https://pubsonline.informs.org/journal/trsc

TRANSPORTATION SCIENCE
Vol. 59, No. 2, March–April 2025, pp. 433–450
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

The Restaurant Meal Delivery Problem with Ghost Kitchens

Gal Neria,[a] Florentin D. Hildebrandt,[b,*] Michal Tzur,[a] Marlin W. Ulmer[b]

[a] Department of Industrial Engineering, Iby and Aladar Fleischman Faculty of Engineering, Tel Aviv University, Tel Aviv 6997801, Israel; [b] Otto-von-Guericke-Universität Magdeburg, 39106 Magdeburg, Germany

*Corresponding author

Contact: galneria@mail.tau.ac.il, https://orcid.org/0000-0001-7089-7210 (GN); florentin.hildebrandt@ovgu.de, https://orcid.org/0000-0003-0298-275X (FDH); tzurm@tauex.tau.ac.il, https://orcid.org/0000-0001-6931-8058 (MT); marlin.ulmer@ovgu.de, https://orcid.org/0000-0003-2499-6570 (MWU)

Received: January 8, 2024
Revised: July 11, 2024; November 27, 2024
Accepted: December 2, 2024
Published Online in Articles in Advance: December 24, 2024

https://doi.org/10.1287/trsc.2024.0510

Copyright: © 2024 INFORMS

Abstract. Restaurant meal delivery has been rapidly growing in the last few years. The main operational challenges are the temporally and spatially dispersed stochastic demand that arrives from customers all over town as well as the customers' expectation of timely and fresh delivery. To overcome these challenges, a new business concept emerged: ghost kitchens. This concept proposes synchronized food preparation of several restaurants in a central facility. Ghost kitchens can bring several advantages, such as fresher food because of the synchronization of food preparation and delivery and less delay because of the consolidated delivery of orders. Exploiting these advantages requires effective operational strategies for the dynamic scheduling of food preparation and delivery. The goal of this paper is providing these strategies and investigating the value of ghost kitchens. We model the problem as a sequential decision process. For the complex decision space of scheduling order preparations, consolidating orders to trips, and scheduling trip departures, we propose a large neighborhood search (LNS) procedure based on partial decisions and driven by analytical properties. Within the LNS, decisions are evaluated via a value function approximation, enabling anticipatory and real-time decision making. In a comprehensive computational study, we demonstrate the effectiveness of our method compared with benchmark policies and highlight the advantages of ghost kitchens compared with conventional meal delivery.

Funding: G. Neria's research is partially supported by the Israeli Smart Transportation Research Center, the Council for Higher Education in Israel, and the Shlomo Shmeltzer Institute for Smart Transportation at Tel Aviv University. F. D. Hildebrandt's research is funded by the Deutsche Forschungsgemeinschaft (DFG) German Research Foundation [Grant 413322447]. M. Tzur's research is partially supported by the Israeli Smart Transportation Research Center. M. W. Ulmer's work is funded by the DFG Emmy Noether Programme [Grant 444657906]. We gratefully acknowledge their support.
Supplemental Material: The online appendix is available at https://doi.org/10.1287/trsc.2024.0510.

Keywords: restaurant meal delivery • ghost kitchens • sequential decision process • large neighborhood search • value function approximation

1. Introduction

The demand for restaurant meal delivery is booming. More and more people order food online and expect a fast and fresh delivery at low cost. The high expectations are often not met in practice, and customers complain about long waits and cold food. At the same time, restaurants and delivery platforms struggle to become profitable with meal delivery. One reason is the limited consolidation potential of orders. Customer orders come in over time from locations all over the city. Restaurants are distributed all over the city as well. Another reason is the limited synchronization between food preparation and delivery routing. Restaurants often serve dine-in customers too and are unable to perfectly synchronize preparation of online orders with their pickup time. In combination, delivery resources are often used ineffectively, and customers experience late and cold food. With these challenges in mind, new business concepts emerge, called ghost kitchens (and related dark kitchens or cloud kitchens) (Shapiro 2023).

In this work, we define a ghost kitchen as a central facility where several restaurants exclusively prepare online orders that are distributed by a joint fleet of vehicles. The ghost kitchen operates with a centralized planning system for coordinating food preparation and vehicle dispatching, communicating plans to cooks and drivers in real-time via a kitchen display system (KDS) and a driver app.

Having several restaurants in one place serving online orders only and planning all operations centrally brings many advantages. For each restaurant, the kitchens can be designed and operated to allow fast and high-quality processing of the online orders (Feldman, Frazelle, and Swinney 2023). Because of the same origin of the food, delivery consolidation can be achieved. Finally, with centralized planning and real-time communication via KDS, consolidation opportunities can be further increased, and food can be ready right on time for delivery (Fresh KDS 2024). Thus, centralized planning can improve the food's freshness (measured by the difference between ready time and time of delivery) compared with conventional delivery, which has limited synchronization between drivers and restaurants.

To exploit these advantages of ghost kitchens, an effective operation of food preparation and vehicle dispatching is necessary to allow for fast and fresh delivery. Delivery vehicles should be dispatched to customers in the same neighborhood. This avoids long travel and congestion of the delivery fleet. To make this possible, the restaurants need to prepare the corresponding food in a way that it is ready for delivery roughly about the same time. Otherwise, at least one customer will receive cold food. At the same time, the restaurant resources should be used efficiently to avoid congestion of orders and long waiting for the corresponding customers. This leads to complex decisions about integrated order sequences and vehicle dispatching with several constraints on freshness and synchronization. All this has to be done in real time as more orders enter the system every minute.

In this research, we propose strategies for the effective operations of ghost kitchens. We model the problem as a sequential decision process and demonstrate that the search over the complex decision space of integrated order scheduling and vehicle dispatching can be replaced by a search over decision representations of significantly reduced dimension. To further improve the search, we embed a novel polynomial algorithm that filters potential candidate decisions (of reduced dimension) via analytical feasibility checks. Then, a full-dimensional decision is generated if and only if feasibility is ensured. Our search over the reduced decision space is performed by a novel large neighborhood search (LNS). Within the LNS, decisions are evaluated via value function approximation (VFA), enabling anticipatory and instant, dynamic, real-time decisions. The VFA is defined independently of the instance size. It is trained on small instances and then transferred to larger instances via transfer learning.

In our computational study, we show the effectiveness of our strategy compared with several benchmark strategies for a wide range of instance settings. We demonstrate that all of our method's components are crucial to achieve effective ghost kitchen operations. We highlight that a careful balance between cook and vehicle resources is important by investigating changing ratios between cooks and vehicles and different freshness constraint parameters. We further observe a trade-off between freshness and delivery time: ensuring fresh food requires more direct trips and reduces flexibility in food preparation. Resources are used less effectively, and the delivery of orders is delayed. We also use our strategy to compare the concept of ghost kitchens to conventional meal delivery systems with several independently operating restaurants. For a clear comparison, we assume that, also for conventional delivery, the platform has full control of the drivers even though in some business models drivers might be partially crowdsourced. With our comparison, we show that, for our instance settings, ghost kitchens improve customer experience significantly by reducing delivery times in ranges of 5%–56%, increasing freshness in ranges of 5%–30%, and reducing the share of delayed customers by 19%–38%. We further see a significant reduction in vehicle travel of 4%–12%.

The contributions of our work are both problem- and method-oriented. To our knowledge, we are the first to analyze the operations of a new and innovative delivery concept that, compared with conventional meal delivery systems, has many advantages, which can be exploited via synchronization of the preparation and dispatching tasks. We demonstrate these advantages in our numerical study, in which improved key performance indicators (KPIs) such as better food quality, faster speed of delivery, and reduced travel times are presented. Based on these results, we present valuable managerial insights when comparing the two delivery concepts, analyzing decision making, and different instance settings. Methodologically, our contribution consists of first reducing the decision space to a significantly lower number of viable candidate solutions by (i) defining a novel reduced decision representation that can be searched more easily than the original one; (ii) reducing the number of candidate decisions even further by filtering, based on analytical results, infeasible decisions; and (iii) developing a novel LNS algorithm to search the remaining decision space, whose search steps are again based on analytical properties. Then, each candidate decision is evaluated via VFA, enabling the incorporation of its anticipated value instantly. In our experiments, we highlight the value of this combination and analyze the impact of balancing off-line training effort with the online execution effort of the LNS. We note that the integration of scheduling and routing is rarely studied in a dynamic environment. Our method has potential adaptability to related problems that involve both scheduling and routing, such as order picking, dispatching, and dynamic production routing. In these contexts, spanning production systems, logistics, and meal delivery, scheduling tasks (e.g., order preparation or production) are closely linked with routing decisions, both of which must adapt dynamically to changing conditions (see Section 6).

The paper is organized as follows: Section 2 provides the literature review focusing on meal delivery and problems with combined scheduling and routing.

Section 3 models the problem as a sequential decision process. Section 4 presents our solution method. Section 5 provides the computational study and comprises an analysis of our policy and a comparison of the business concepts. The paper concludes with Section 6, which discusses the generality of our methodology as well as its applicability to other methods and summarize our work. The paper also provides additional details in an online appendix.

2. Literature Review

Our work addresses the problem field of restaurant meal delivery—its model requires dynamic scheduling and routing—and the methodology designed to handle a large and complex decision space under dynamic and stochastic order arrival. Consequently, our research draws from three key streams of literature:

(i) Meal delivery routing: This stream focuses on strategies for efficiently routing orders from restaurants to customers under dynamic conditions, often addressing order consolidation and timely delivery.

(ii) Combined scheduling and routing: This area studies the integration of scheduling tasks with routing decisions, highlighting the importance of synchronization across operations.

(iii) Methodologies: This stream surveys approaches to search and evaluate large decision spaces in dynamic problems.

In the following, we review the relevant literature in these areas, highlighting their connection to ghost kitchens and identifying the gaps our work seeks to address.

2.1. Problem: Meal Delivery Routing

Work on restaurant meal delivery has surged in recent years, focusing on efficient and flexible routing strategies to deliver meals from different restaurants to dynamically requesting customers. Ghost kitchens, however, present unique characteristics that are not addressed by traditional meal delivery routing studies. These include centralized meal preparation and the synchronization between preparation and dispatching tasks. For example, whereas early works by Reyes et al. (2018); Steever, Karwan, and Murray (2019); Liu (2019); and Ulmer et al. (2021) explore dynamic assignment of orders to vehicles, they do not consider integrated preparation and delivery operations or the centralized structure of ghost kitchens. Similarly, Yildiz and Savelsbergh (2019) analyze bundling strategies but focus on deterministic settings, which do not capture the stochastic and dynamic nature of ghost kitchens. The authors derive insights in the value of bundling operations and give guidelines in demand management and the scheduling of delivery vehicles.

Recently, first work on reinforcement learning was proposed by Jahanshahi et al. (2022) to determine anticipatory assignments and rejections of orders, that is, learning the expected future revenue when a decision is made. However, scheduling or routing decisions were not considered. Other work on meal delivery addresses arrival time predictions with the goal of providing accurate delivery time information to customers (Hildebrandt and Ulmer 2022), the scheduling of the workforce to ensure timely delivery without excessive workforce cost (Dai and Liu 2020; Ulmer and Savelsbergh 2020; Auad, Erera, and Savelsbergh 2024), or the zoning of the service area with respect to current and future demand (Ulmer, Erera, and Savelsbergh 2022; Auad, Erera, and Savelsbergh 2023). However, none of the considered studies analyzes ghost kitchens or considers integrated scheduling and routing decisions. In the problem addressed in this paper, we consider the assignment and scheduling of orders in combination with the vehicle dispatching.

To achieve efficiency (and sometimes feasibility) in the overall process, decisions regarding order preparation and vehicle dispatching operations need to be coordinated in an integrated and anticipatory fashion. In ghost kitchens, this synchronization refers to aligning preparation schedules with dispatching tasks to ensure timely delivery, maintaining food quality. Whereas synchronization in meal delivery has not been considered in the literature yet, its importance in vehicle routing problems is highlighted by Drexl (2012), who surveys vehicle routing problems involving resource or task synchronization, such as coordinating multiple vehicle types or aligning delivery time windows.

2.2. Model: Combined Scheduling and Routing

From a modeling perspective, our problem may be considered as a combination of models for dynamic order scheduling, which has been studied only in limited contexts. For example, Xu et al. (2016) and Zhao et al. (2018) investigate dynamic scheduling under stochastic arrivals but do not consider downstream routing decisions. Similarly, Klapp, Erera, and Toriello (2018) study dynamic vehicle dispatching but without integrating upstream scheduling operations. Ghost kitchens require a more holistic approach because of the interdependence between preparation and delivery stages.

Integrated scheduling and routing has been explored in static or deterministic settings as reviewed by Moons et al. (2017). However, Zhang et al. (2019), Hossein Nia Shavaki and Jolai (2021), and Liu et al. (2022) consider online scenarios but assume simplified delivery models, such as predefined destinations or absence of routing complexity. Unlike these studies, ghost kitchens demand real-time coordination between preparation and delivery, making existing models insufficient for this application. Motivating our benchmark policies, Liu et al. (2022) propose an approximate dynamic programming (DP) solution in which simple principles are used to reduce the decision space, for example, shortest processing time in production and first in, first out (FIFO) in delivery. Rijal, Bijvank, and de Koster (2023) suggest integrated optimization of static and deterministic warehouse operations and delivery routing. Rijal, Bijvank, and de Koster (2023) compare sequential and integrated approaches and show the potential of joint optimization.

To conclude, the integration of scheduling and routing is rarely explored in dynamic environments, particularly with anticipation of future order arrivals. Our work addresses this gap by presenting a more general setting relevant to the growing online delivery domain and introducing one of the first anticipatory policies that integrates scheduling and routing optimization.

2.3. Methodology: Search and Evaluation of Complex Decision Spaces

The methodological contribution of our work is to combine the search of a complex decision space with an evaluation via VFA (also known as reinforcement learning (RL)). Value function approximations (and RL methods in general) repeatedly simulate the decision process and store the observed values of decisions in an aggregated form. The values are used for decision making and updated over the simulation. As shown by Hildebrandt, Thomas, and Ulmer (2023), past research focuses on a comprehensive search either without an explicit evaluation or with an explicit evaluation via RL but only for a few potential decisions. However, there are a few exceptions.

Rivera and Mes (2017, 2022) and Heinold, Meisel, and Ulmer (2023) use RL to decide which freight to dispatch. The values are approximated via a linear function based on a set of postdecision state features. Mixed integer programming is used for searching the decision space. A similar concept is proposed by Silva, Pedroso, and Viana (2023) to decide about assignments in urban delivery. However, the approximations are based on a neural net that is linearized such that mixed integer programming can be applied. None of the works considers more complex decision spaces or problems that involve routing decisions with complex constraints. Incorporating these features slows down the search in the decision space, and it makes the VFA more challenging.

Recently, Neria and Tzur (2024) propose using a metaheuristic to search the decision space quickly and evaluate decisions with a VFA. The general idea is similar to our approach. However, whereas our works share the general combination of metaheuristic and VFA, our problem differs significantly from Neria and Tzur (2024). Consequently, the design of both search and evaluation methodologies are different. For our problem, the decision space is very large, and finding feasible decisions is a challenge. Therefore, we propose an alternative decision formulation that reduces the burden to determine a complete decision and allows for fast feasibility checks based on a set of carefully derived propositions. With the more complex decision space and larger problem size, learning the values becomes more challenging as well. Whereas the fast search reduces the training time significantly, simulating systems of realistic size and learning the values is still computationally expensive. Thus, we propose transfer learning to train our policy on small instances and then transfer our trained policy to realistic-sized instances.

3. Problem Statement

In this section, we introduce the restaurant meal delivery problem with ghost kitchens (RMD-GK). First, we describe the problem. Then, we illustrate the model's components with an example. Finally, we formally define the problem as a sequential decision process.

3.1. Problem Description

A ghost kitchen is a facility for the sole purpose of preparing delivery-only meals. It is partitioned into a set of kitchens, each with dedicated cooks, and uses a joint fleet of capacitated vehicles for delivery (e.g., cargo bikes or delivery cars). Each kitchen is associated with a fixed ghost restaurant that we refer to as a food type and prepares only orders from that food type.

Over the course of the day, customers order food from the restaurants. In our setting, each order includes only one of the food types such that an entire order is prepared in the kitchen of the associated ghost restaurant. Each order has a known preparation time and a customer location. Once an order is prepared, a vehicle picks up the order and delivers it (immediately or after some waiting time) to the respective customer with or without other customer orders in the same trip. If orders are bundled and more than one order is dispatched in the same trip, the trip also captures the sequence of orders.

The scheduling of the orders in the restaurants and the scheduling of the delivery vehicles is done by a central information system. This system maintains and updates a schedule over time. The schedule determines for each restaurant which cook prepares which meal order and when. We refer to this as the cook schedule. It further decides about the bundling of orders to trips, their assignment to vehicles, and the departure times of each trip. We refer to this as the vehicle schedule. A schedule is only feasible if each order's freshness is guaranteed when the meal arrives at the customer. Thus, a schedule must ensure for each order that the ready-to-door time (i.e., the difference between the ready time of the food at the restaurant and the arrival time at the customer) does not exceed an order-specific duration. For example, even when a cook is available, the starting time of an order preparation may be postponed, if no vehicle is available to deliver it when the meal preparation is complete, until a fresh delivery can be guaranteed. Customers expect a fast delivery, ideally within 30 minutes after the order is placed. The goal of the platform is to meet this expectation by minimizing the average excess of delivery time (delay) over all orders.

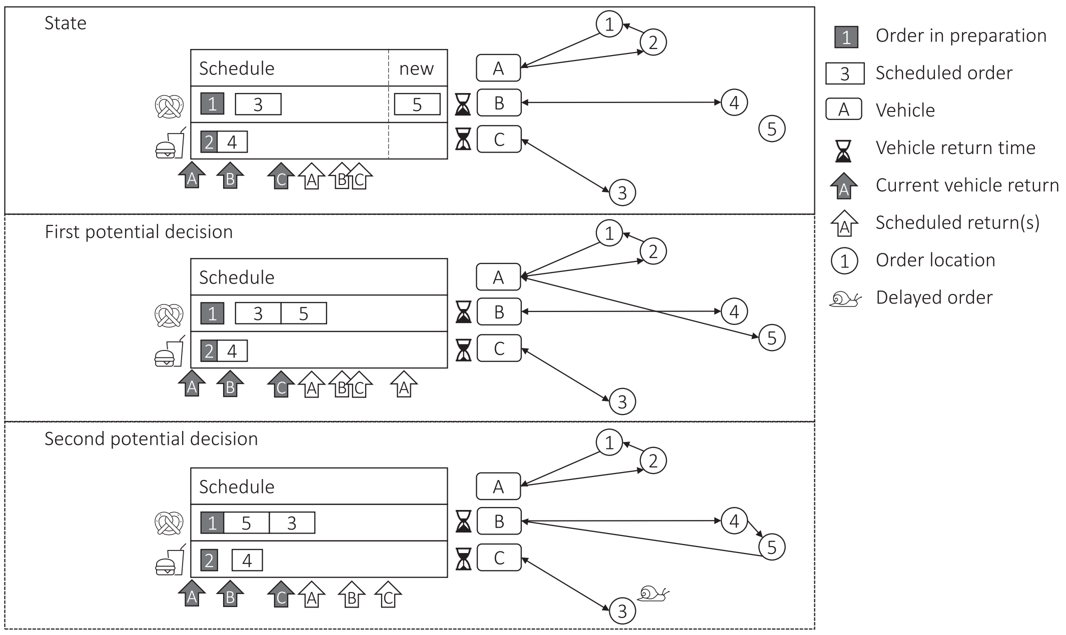
3.2. Example

In the following, we provide a small example to illustrate the RMD-GK and to introduce the logic that is used in the modeling. For the ease of presentation, we omit any notation and numbers. The example is shown in Figure 1. On the top of the figure, a possible state of the system is shown. The middle and bottom parts depict the resulting system postdecision states for two potential decisions. In the example, the schedules of the numbered orders in the kitchens are depicted in the large box on the left. There are two food types (German and U.S.), each with one cook. In the depicted state, five orders are shown in rectangular shapes. The width of the shape indicates the preparation time for each order. Orders currently in preparation are depicted by the gray boxes, and the size of each gray box corresponds to the remaining preparation time. Four of the orders (1, 2, 3, 4) are already scheduled, and one order is new (5). The orders 1, 2, and 5 belong to the first food type (German), whereas the orders 2 and 4 belong to the second food type (U.S.). Orders 1 and 2 are already in preparation. The current scheduling of the orders is (1, 3) and (2, 4) for the first and second food types, respectively. The gap between the finish time of order 1 and the start time of order 3 is the result of the vehicle departure plan, which we discuss next.

There are three vehicles (denoted by A, B, and C), which are indicated by the three small squares with soft corners. The hourglasses beside the vehicles indicate that they are currently busy delivering orders. The geographical locations of the orders are shown on the right-hand side. In the depicted state, vehicle A is currently available, and vehicles B and C are out for delivery. Because the details of their current trips and deliveries do not matter, they are not depicted in the example. Only their return times matter, which are indicated by the hourglasses near the vehicles. The gray arrows depicted below the schedule indicate when vehicles become available. Vehicle A just returned, and vehicle B returns before vehicle C. The planned bundles and trips for the vehicles are depicted on the right side of the figure. Specifically, it is planned that vehicle A delivers orders 2 and 1 once they are finished. This sequence might be necessary to ensure freshness because the preparation of order 2 finishes earlier than the preparation of order 1. Once vehicle B (C) returns, it is scheduled to deliver order 4 (3). Because vehicle C becomes available later, the start of preparation of order 3 is postponed. The scheduled later returns of the vehicles are depicted by the white arrows below the schedule.

In the middle and bottom parts of Figure 1, two potential decisions are shown. Each decision integrates the new order in a certain way and consequently updates the preparation schedules, the bundles, and the trips of the vehicles. According to the first decision, the previous cook (or meal) preparation schedule remains the same and order 5 is added to it in a FIFO manner. The same is true for the vehicle schedule in which order 5 is added as a direct trip by vehicle A.

Figure 1. Example for a State and Two Potential Decisions



This results in a second scheduled return for vehicle A. An alternative decision is shown in the bottom part of Figure 1, in which order 5 is inserted before order 3. This allows for a joint delivery of orders 4 and 5, which are geographically close, by vehicle B. To ensure feasibility, the preparation of order 4 is slightly postponed. Note that this decision causes a delay when order 3 is delivered, indicated by the snail.

3.3. Sequential Decision Process

The RMD-GK is stochastic and dynamic. It is stochastic as orders are unknown until they are placed, and their realizations follow known probability distributions in time and space. It is dynamic because decisions are made repeatedly over time. We model the problem as a sequential decision process.

We consider a capture phase $[0, T^c]$ during which orders are placed. We further define a longer operation phase $[0, T]$ with $T > T^c$ sufficiently large to allow the fulfillment of all placed orders. We denote the set of all possible orders by I. Each order is associated with an order time, a food type, a known preparation time, and a customer location (the order's destination). The food types associated with orders are modeled via the set F. Each food type has a corresponding freshness time $\delta_f, f \in F$, which corresponds to the maximum time allowed for the meal order to be delivered at its destination after the meal is prepared. Furthermore, each order should be delivered within a promised delivery time $\tau \in \mathbb{R}$ after order placement, and exceeding this leads to a delay cost. The set of cooks is denoted by C with subsets C_f working on food type $f \in F$. The vehicles are modeled via a set V. The vehicle capacity is measured as the maximum number of orders allowed to be simultaneously carried by a vehicle and is denoted by $\kappa \in \mathbb{N}$. The travel time from the location of order $i \in I \cup \{0\}$ (or the ghost kitchen) to the location of order $j \in I \cup \{0\}$ (or the ghost kitchen) is given by $t^t_{ij} \in \mathbb{R}_{\geq 0}$, where location 0 denotes the location of the ghost kitchen. Next, we present all elements of the sequential decision process, following the modeling framework of Ulmer et al. (2020).

3.3.1. Decision Points.

A decision point k occurs whenever a new order is placed and at the end of the order capture phase at time T^c when no more orders will arrive. As the realization of orders is stochastic, the number of decision points K is a random variable.

3.3.2. States.

A state $S_k \in \mathcal{S}$ comprises all information relevant for decision making. We categorize the state information according to order information, currently planned cook schedules, and currently planned vehicle schedules. First, we describe the order information. Except for the final state S_K in T^c, a state contains a new order. The new order in state S_k at time t_k is denoted by i_k. The set of open orders, that is, orders that have been placed but have not yet left the ghost kitchen for delivery, is denoted by I_k and includes order i_k. Each order $i \in I_k$ is represented by four variables indicating the food type $f_i \in F$ of order i, the time of day at which order i was placed $t^o_i \in [0, T^c]$, the preparation time of the corresponding meal $t^p_i \in \mathbb{R}_{\geq 0}$, and the location of the order l_i. In summary, each order $i \in I_k$ is associated with (f_i, t^o_i, t^p_i, l_i).

Second, we describe the currently planned cook schedules. The cook schedules were decided in the last decision point $k-1$ and pruned in the transition to the current state (see "Transition"). They include all orders in I_k except the new order i_k. In state S_k, the currently planned sequence of orders prepared by cook $c \in C$ is given by $\psi_{kc} = (i^c_1, i^c_2, \ldots) \subset I_k \setminus \{i_k\}$. The set of all preparation sequences is given by $\Psi_k = \{\psi_{kc} | c \in C\}$. The planned time of day at which the preparation of order $i \in I_k \setminus \{i_k\}$ is started is denoted by $t^s_{ki} \in [t^o_i, T]$, where s is a symbol. The corresponding set is given by $t^s_k = \{t^s_{ki} | i \in I_k \setminus \{i_k\}\}$. In summary, the state variables corresponding to the cook schedules are given by (Ψ_k, t^s_k).

Third, we describe the currently planned vehicle schedules. Analogous to the cook schedules, the currently planned vehicle schedules were decided in the last decision point and pruned in the transition to the current state (see "Transition"). The schedules contain all orders in I_k except the new order i_k. The sequence of trips performed by vehicle $v \in V$ is denoted by $\Theta_{kv} = (\theta_{kv1}, \theta_{kv2}, \ldots)$. Each trip $\theta \in \Theta_{kv}$ consists of a sequence of orders $(i^\theta_1, i^\theta_2, \ldots) \subset I_k \setminus \{i_k\}$. The set of all planned trip sequences is given by $\Theta_k = \{\Theta_{kv} | v \in V\}$. The planned time of day at which trip $\theta \in \Theta_k$ departs is denoted by $t^d_{k\theta} \in [t_k, T]$, where d is a symbol. The set of all departure times is denoted by $t^d_k = \{t^d_{k\theta} | \theta \in \Theta_k\}$. The (next) return time of each vehicle $v \in V$ to the ghost kitchen is denoted by $t^r_{kv} \in [t_k, T]$ (r is a symbol), where its value is either t_k if the vehicle is idling or the return time from its last trip that departed before t_k (note that the trip itself is not part of the state S_k). The set of all vehicle return times is given by t^r_k. To summarize, we represent a state as the tuple

$$S_k = (\underbrace{t_k, I_k}_{\text{Orders}}, \underbrace{\Psi_k, t^s_k}_{\text{Cooks}}, \underbrace{\Theta_k, t^d_k, t^r_k}_{\text{Vehicles}}). \tag{1}$$

3.3.3. Decisions.

A decision $x_k \in \mathcal{X}(S_k)$ is an update $x_k = (\Psi^x_k, t^{s,x}_k, \Theta^x_k, t^{d,x}_k)$ on the current plan $(\Psi_k, t^s_k, \Theta_k, t^d_k)$. This update must integrate the new order i_k into the current plan (except for the final state S_K in T^c) but may also change other parts of the plan. For example, it might reassign and reschedule orders to cooks or vehicles to adapt to changes in information. In general, an update on the current plan is feasible only if

- Each order is scheduled for preparation and delivery by exactly one cook and one vehicle.

- The freshness constraint is satisfied, that is, the time between the end of preparation and scheduled time of delivery of an order i is sufficiently small (i.e., less than δ_{f_i}).
- Already started preparations remain unchanged.
- Each order's preparation finishes before its delivery trip departs.
- Only one order is prepared by a cook at a time.
- Each trip of a vehicle departs only after the return from the previous trip.
- Vehicles satisfy the capacity constraint.
- Trips are elementary tours that start and end at the ghost kitchen.

The freedom of modifying the previous schedule combined with the various problem constraints yield a complex decision space $\mathcal{X}(S_k)$, which we model as a mixed integer linear program in Online Appendix A.1.

3.3.4. Postdecision States.

The postdecision state $S_k^x \in \mathcal{S}^x$ represents the state directly after a decision is made but before it transitions to the next state (i.e., before the arrival of the next order i_{k+1}). Thus, given the state $S_k = (t_k, I_k, \Psi_k, t_k^s, \Theta_k, t_k^d, t_k^r)$ and the decision $x_k = (\Psi_k^x, t_k^{s,x}, \Theta_k^x, t_k^{d,x})$, the postdecision state is defined as $S_k^x = (t_k, I_k, \Psi_k^x, t_k^{s,x}, \Theta_k^x, t_k^{d,x}, t_k^r)$.

3.3.5. Costs.

The immediate cost D_k^Δ associated with state S_k and decision $x_k = (\Psi_k^x, t_k^{s,x}, \Theta_k^x, t_k^{d,x})$ is given by the marginal difference in cost of the old plan $(\Psi_k, t_k^s, \Theta_k, t_k^d)$ and the updated plan $(\Psi_k^x, t_k^{s,x}, \Theta_k^x, t_k^{d,x})$ (see Ulmer et al. 2020). The cost of a plan corresponds to the total planned delay of the orders, that is, the delay that would be observed when following the plan. Let $D(\Theta, t^d)$ denote the total delay for a given plan (Ψ, t^s, Θ, t^d). We define D according to

$$D(\Theta, t^d) = \sum_{\theta \in \cup\Theta} \sum_{j=1}^{|\theta|} \max\left(0, t_\theta^d + \left(\sum_{l=1}^{j-1} t^t_{i_l^\theta, i_{l+1}^\theta}\right) - t^o_{i_j^\theta} - \tau\right), \tag{2}$$

where $\cup\Theta$ denotes the union of all sets of trips contained in Θ. In words, we sum over all trips of all vehicles. For each trip, we compute the delay of each order as the maximum of zero and the difference of the time span from order placement to order arrival and the promised delivery time τ. We define the marginal cost (the immediate cost) D_k^Δ as

$$D_k^\Delta(S_k, x_k) = D(\Theta_k^x, t_k^{d,x}) - D(\Theta_k, t_k^d). \tag{3}$$

Note that the marginal cost can be negative in case the delay of the new plan is smaller than the one of the previous plan. Furthermore, note that the sum of the marginal costs over all realized states coincides with the sum of realized delays over all orders. In the special case of the final state S_K in T^c, the process terminates after the decision is made.

3.3.6. Stochastic Information.

The stochastic information $W_{k+1} \in \Omega$ is given by the new order i_{k+1} and corresponding food type $f_{i_{k+1}}$, order placement time $t^o_{i_{k+1}}$, meal preparation time $t^p_{i_{k+1}}$, and order location $l_{i_{k+1}}$. There is a special case in which $W_{k+1} = \emptyset$, that is, no more orders are realized. In that case, the sequential decision process moves to the final state T^c.

3.3.7. Transition.

The transition $S^M : \mathcal{S} \times \mathcal{X} \times \Omega \to \mathcal{S}$ maps a state–decision tuple and stochastic information to the next state $S_{k+1} = S^M(S_k, x_k, W_{k+1})$. The current time is updated to $t_{k+1} = t^o_{i_{k+1}}$. The new set of open orders is given by $I_{k+1} = \bigcup_{\theta \in \Theta_k^x} \{i \in \theta \mid t_{k\theta}^{d,x} > t_{k+1}\} \cup \{i_{k+1}\}$, that is, all orders whose departure times are after t_{k+1}. The schedule of each cook $c \in C$ is truncated to include only orders in I_{k+1}. The new planned sequence of trips for vehicle $v \in V$ is given by truncating trips $\theta \in \Theta_{kv}^x$ with $t_{k\theta}^{d,x} < t_{k+1}$. The return time of vehicle $v \in V$ is updated according to

$$t^r_{k+1,v} = \max\left(t_{k+1}, \max_{\theta \in \Theta_{kv}^x \mid t_{k\theta}^{d,x} < t_{k+1}} t_{k\theta}^{d,x} + t^t_{i^\theta_{|\theta|}, 0} + \sum_{j=1}^{|\theta|-1} t^t_{i_j^\theta, i_{j+1}^\theta}\right). \tag{4}$$

3.3.8. Objective.

A mapping that assigns each state S_k a decision $\pi(S_k) = x_k$ is called a policy and denoted by $\pi \in \Pi$. Given an initial state $S_0 = (0, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$, the objective is to find an optimal policy π^* that minimizes the expected sum of delay. This coincides with minimizing the expected marginal costs over all decision points when starting in S_0 and applying policy π throughout the process:

$$\min_{\pi \in \Pi} \mathbb{E}\left[\sum_{k=0}^{K} D^\Delta(S_k, \pi(S_k)) \mid S_0\right]. \tag{5}$$

4. Solution Method

In this section, we present our solution method. We first give a motivation and an overview in Section 4.1 before defining the two main components: search (Section 4.2) and evaluation (Section 4.3) in detail.

4.1. Motivation and Overview

The problem at hand requires fast and effective decisions. The small example in Figure 1 already illustrates the complexity of finding a feasible and effective decision for cook and vehicle schedules. Even for this small example, the number of potential decisions is already vast. Additionally, it is not clear which of the two decisions shown in the example is more effective as it depends on future orders and decisions. The first decision avoids delay now but binds more vehicle resources, whereas the second decision causes a slight delay now, but saves vehicle resources. The first decision might be advantageous if the expected number of future orders is small, whereas the latter decision becomes more effective in case many orders can be expected in the future. Hence, the evaluation of a decision should anticipate potential future delay when a specific decision is taken. The two challenges of a thorough search of the decision space and an anticipatory evaluation of decisions are captured in the Bellman equation

$$\pi^*(S_k) \in \underset{x \in \mathcal{X}(S_k)}{\arg\min}\; D^{\Delta}(S_k, x) + \mathcal{V}(S_k^x). \tag{6}$$

The Bellman equation defines an optimal decision $\pi^*(S_k)$ from the set of overall decisions $\mathcal{X}(S_k)$ in a state S_k based on the (known) immediate cost $D^{\Delta}(S_k, x)$ and the expected cost to go when taking a decision x. The cost to go is modeled via the value function $\mathcal{V}$, which maps postdecision states S_k^x to the expected future cost $\mathcal{V}(S_k^x)$:

$$\mathcal{V}(S_k^x) = \mathbb{E}\left[\sum_{k'=k+1}^{K} D^{\Delta}(S_{k'}, \pi^*(S_{k'})) \,|\, S_k^x\right]. \tag{7}$$

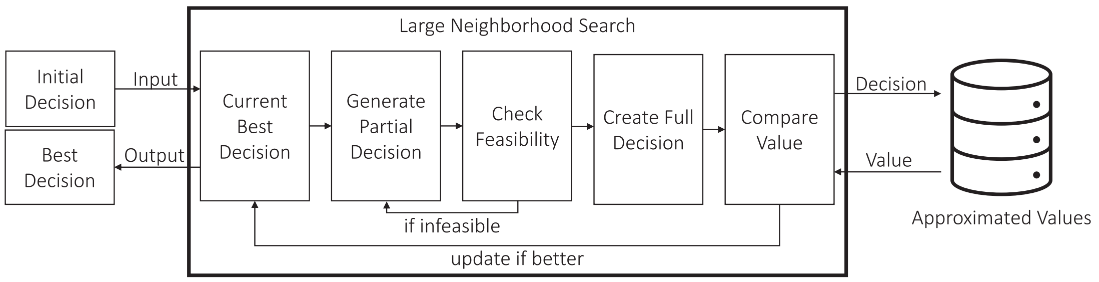
Thus, finding an effective policy poses two main challenges. First, a comprehensive and runtime-efficient search of the decision space, which requires solving a complex combinatorial optimization problem, is required, indicated by $\arg\min_{x \in \mathcal{X}(S_k)}$ in the Bellman equation. Second, the decisions need to be evaluated instantly with respect to their immediate and future impact with the latter represented by the (unknown) value function $\mathcal{V}(S_k^x)$ in the Bellman equation. We address these challenges by combining an LNS of the decision space driven by analytical insights with a VFA tuned via transfer learning.

Specifically, we propose an LNS to search the decision space. Because the decision space is vast, a large number of LNS iterations are required to search it thoroughly. However, many decision candidates turn out infeasible in the end because of the capacity or freshness constraints and need to be discarded. To allow a fast but thorough search and quickly identify infeasible decision candidates, we base our LNS on a condensed representation of a decision, that is, one that reduces the decision space without loss of optimality. Within the LNS, we search on condensed partial decisions, that is, in which the planned sequence of orders at the cooks and trips of vehicles is determined but their exact timing is not. To determine feasibility and timing of the corresponding (partial) sequencing decision, we propose a polynomial algorithm, based on a dynamic programming formulation and a set of analytical propositions.

To evaluate a feasible decision candidate, we propose a VFA. The VFA approximates values $\mathcal{V}(S_k^x)$ via repeated simulations. Once the values are learned, decisions can be evaluated instantly. For the VFA, a postdecision state is represented by a set of features. As illustrated in the example of Figure 1, the value depends on the resources' availability and their balance (i.e., the workload distribution among the cooks and vehicles both individually and relative to one another). Thus, we derive problem-specific features accordingly. A final challenge of the VFA is the runtime. Even though training can be done off-line, the training requires large numbers of simulations with many states and decisions via LNS. Thus, we propose transfer learning, training on a smaller problem size and then adapting the trained policy to larger size instances via fine-tuning. As approximation architecture, we rely on a neural network (NN). The general idea of combining a neighborhood search with an NN-based VFA was proposed by Neria and Tzur (2024).

As our method allows for an integrated optimization of scheduling and routing, anticipating future orders, we denote it anticipatory integrated (AI). The overall procedure when a new order arrives to the system is summarized in Figure 2. In a state, an initial (original) decision is fed to the LNS as the current best decision. Then, over a number of iterations, a new, partial (condensed) decision is generated by a set of operators. The feasibility of the partial decision is checked, and if not feasible, it is discarded, and a new partial decision is generated. This is done by our proposed partial decision feasibility and timing (PDFT) algorithm. In case a partial decision is feasible, the PDFT creates the full (original) decision, and the value of the decision, provided by the stored approximated values, is compared with the current best decision. In case the value is better, the currently best found decision is updated. After a given number of iterations, the procedure returns the

Figure 2. Overview of the Steps of Our AI Method in a Decision State



best found decision. This decision is then implemented in the sequential decision process. In the following, we describe the individual steps of the procedure in detail.

4.2. The Large Neighborhood Search

In this section, we describe our method to search the decision space whenever a new order arrives in the system and a decision needs to be made. We follow the steps proposed in Figure 2. We first introduce the condensed decision representation and, building on it, the notion of a partial decision. We describe the LNS operators for the partial decision space. Finally, we present the PDFT algorithm to check feasibility and create the full decision.

4.2.1. A Condensed Representation of a Decision and a Partial Decision. We start with a formal definition of a condensed decision representation. To that end, we denote by $I_{kf} = \{i \in I_k | f_i = f\}$ the set of open orders of food type $f \in F$. Recall that x_k includes assignments of orders to specific cooks and assignments of trips to specific vehicles. The idea of a condensed decision is to combine together all orders that have the same food type (all trips) instead of assigning each order to a specific cook of the respective food type (specific vehicle).

Definition 1 (Condensed Decision Representation). A condensed decision representation of x_k includes the following components. For each food type $f \in F$, a sequence of processing all orders in I_{kf} is given by $\psi_k^{f,x} = (i_{f1}, i_{f2}, \ldots)$. The set of all preparation sequences (for all food types) is given by $\widehat{\Psi}_k^x = \{\psi_k^{f,x} | f \in F\}$. The sequence of trips to dispatch by all vehicles in V is given by a single vector $\widehat{\Theta}_k^x = (\theta_1, \theta_2, \ldots)$. The associated times $t_k^{s,x}$ and $t_k^{d,x}$ to start each order and depart each trip, respectively, remain as in the original decision representation (see Section 3.3).

That is, $\widehat{\Psi}_k^x$ replaces Ψ_k^x in the original decision representation by storing the sequences of processing all orders by food types instead of by specific cooks. Similarly, $\widehat{\Theta}_k^x$ replaces Θ_k^x in the original decision representation by storing a sequence for all vehicles in V instead of a sequence for each vehicle. Note that, because the times to start each order and depart each trip are identical for the original and its associated condensed representations, they have the same cost. We define a condensed decision to be feasible if there exists an original feasible decision that induces the condensed representation. With this representation, we reduce the decision space by eliminating equivalent decisions in Ψ_k^x and Θ_k^x (because of the homogeneity of cooks of the same food type and vehicles). For the first decision in the small example in Figure 1, the representation of the food type sequences is identical to the cook schedule representations, that is, (1, 3, 5) and (2, 4) for the German and U.S. food types, respectively, because only one cook per food type is available. The representation also contains the corresponding start times for the orders (not explicitly stated in the example). In contrast to the original representation, the vehicle schedule would be represented as $((1,2),(4),(3),(5))$, again, with the corresponding departure times. Next, we present definitions and claims that are used to prove Theorem 1, that the condensed decision space also contains an optimal decision.

Definition 2 (Symmetric Decisions). Given a state S_k, two decisions x and x' are symmetric if they include the same set of trips as well as the same sets of starting times $t_k^{s,x}$ and $t_k^{d,x}$ for orders and trips.

Two symmetric decisions can only differ in the specific cook and vehicle assignments.

Claim 1 (Condensed Representation of Symmetric Decisions). *Given a state S_k, decisions are represented by the same condensed decision representation if and only if they are symmetric decisions.*

Proof of Claim 1. This is immediately derived from Definitions 1 and 2. □

Claim 2 (Symmetric Decision Costs). *Given a state S_k, the resulting postdecision states of symmetric decisions have the same immediate cost and the same cost to go.*

Proof of Claim 2. In each pair of symmetric decisions, the starting preparation and departure times of all orders and trips, respectively, are identical, and hence, the arrival time of each order $i \in I_k$ to the customer is identical, that is, the total delay is the same (the immediate cost). For the same reason, at each point of time $t \geq t_k$, the number of available cooks of each food type and the number of available vehicles is the same in both postdecision states. As cooks and vehicles are homogeneous and their indices are, therefore, interchangeable, the cost to go is also identical. □

Theorem 1. *Given a state S_k, an optimal decision can be found by searching over all (feasible) condensed decision representations rather than over all original decision representations.*

Proof of Theorem 1. Every original decision representation has a corresponding condensed decision representation, and every feasible condensed decision is associated with a nonempty set of (feasible) original decision representations. Thus, all original decision representations are included in the search, and only feasible original decision representations are considered. Moreover, all original representations that are associated with the same condensed representation are symmetric and have an equal immediate cost and cost to go (Claims 1 and 2). □

Consequently, we use the condensed decision representation in our search heuristic to reduce the decision space. Toward a further reduction of the decision space, we next present the notion of a partial decision, which reduces the decision space we have to explore at any given state.

Definition 3 (Partial Decision). A partial decision includes only the sequences of a condensed decision, that is, the sets of preparation and trip sequences, $\widehat{\Psi}_k^x$ and $\widehat{\Theta}_k^x$, respectively.

For the example in Figure 1, this would leave the food type sequences (1, 3, 5) and (2, 4) for the German and U.S. food types, respectively, without the preparation starting times. It would also contain the condensed vehicle sequence $((1,2),(4),(3),(5))$ but without the departure times.

4.2.2. The LNS Procedure. We now define our LNS procedure operating on partial decisions. We start by generating an initial decision using an FIFO procedure (see Algorithm 1 in Online Appendix A.2). We generate new partial decisions using operators that alter task sequences in $\widehat{\Psi}_k^x$ and $\widehat{\Theta}_k^x$ as described below. The feasibility of each partial decision is checked using a polynomial algorithm (see the PDFT in Online Appendix A.4). If a partial decision is feasible, the algorithm assigns tasks to cooks and vehicles as well as task starting times. A pseudocode of the LNS is given by Algorithm 2 in Online Appendix A.2.

The FIFO procedure generates the initial decision by appending the new order i_k to the planned schedule $(\Psi_k, t_k^s, \Theta_k, t_k^d)$ (as defined by the state variables of S_k), that is, by assigning i_k to a cook and a vehicle trip. It starts by assigning order i_k to the first available cook $c \in C_{f_{i_k}}$, initially at the earliest time $t \geq t_k$ in which c is first available after the last scheduled order in ψ_{kc} is prepared (according to the plan in Ψ_k and t_k^s). Then, it searches for all vehicles $v \in V$ that have fewer than κ orders assigned on their last scheduled trips according to Θ_k and their departure time is no earlier than $t + t_i^p$, the time order i_k can be ready. The selected trip is the one that can get i_k to the customer with minimum tardiness. We note that, if order i_k joins a tour with other orders, the trip is determined to minimize the total tardiness over the orders included in it (and i_k) such that it is feasible. If there is no existing scheduled trip the new order can join, then it is assigned on a new trip that departs as soon as both the order finishes preparation and there is an available vehicle. If the earliest time a vehicle can reach i_k extends its freshness constraint, then the preparation time by cook c, t, is postponed such that the order arrives fresh. The described procedure to insert order i_k to the planned schedules $(\Psi_k, t_k^s, \Theta_k, t_k^d)$ without additional search steps is used in Section 5.3 as a first benchmark, which we refer to as FIFO. This heuristic represents what is often done in practice. A pseudocode of the FIFO benchmark is given by Algorithm 1 in Online Appendix A.2.

Based on the initial decision (or current decision), new decisions are generated to search the decision space as follows. Let $\psi_k^{f,x} = (i_{f1}, i_{f2}, \dots)$, $\forall f \in F$ and $\widehat{\Theta}_k^x$ be the partial decision of the current decision, referred to as the current partial decision. We generate the next decision by using one of seven operators to alter one of the sequences in $\psi_k^{f,x}$, $\forall f \in F$ or $\widehat{\Theta}_k^x$ such that we obtain a new partial decision, represented by new sequences: $\psi_k^{f,x'}$, $\forall f \in F$ and $\widehat{\Theta}_k^{x'}$ (where $\widehat{\Psi}_k^{x'} = \{\psi_k^{f,x'}, \forall f \in F\}$). The operators are randomly selected with equal probabilities. The operators were chosen according to our understanding of the problem domain based on analytical properties as well as random procedures that ensure a large portion of the decision space can be covered. The individual operators are described in detail in Online Appendix A.3. Generally, operators 1 and 2 make changes to the order preparation sequences, operators 3 and 4 change the trips sequence, and operators 5–7 change the vehicle trips.

After an operator is applied, we check if the new partial decision is feasible using the PDFT, which we describe in the next section. If not, we discard it. Otherwise (if it is feasible), the PDFT also derives timing sets $t_k^{s,x'}$ and $t_k^{d,x'}$. If the decision has a lower (cost) value than the current one (see Section 4.3), it becomes the new current partial decision from which the search continues.

4.2.3. The PDFT. We next summarize our PDFT algorithm (for all details, we refer to Online Appendix A.4). The input of the PDFT is a state S_k and a partial decision, $\psi_k^{f,x}$, $\forall f \in F$ and $\widehat{\Theta}_k^x$. Its output is whether the partial decision is feasible or not. If feasible, it additionally provides the starting times for order preparations and the departure times for trips. To describe the PDFT, we first sketch the problem it is aimed to solve, referred to as the assignment and timing subproblem (ATP): given a state S_k in the RMD-GK problem and a partial decision, that is, set of sequences of orders to prepare for each food type and a sequence of trips to dispatch, $\psi_k^{f,x}$, $\forall f \in F$ and $\widehat{\Theta}_k^x$, respectively, the goal of the ATP is to assign the orders (trips) to the cooks (vehicles) and determine feasible task starting times such that the orders and trips are processed according to the given sequences and the total delay of the open orders in I_k (given by S_k) is minimized. In Online Appendix A.4, we present a DP formulation of the ATP. We further present a set of analytical propositions that narrow down the state and decision space of the ATP without loss of optimality. We also present properties of the optimal solution to the DP formulation, which form the basis to the PDFT algorithm.

The PDFT starts with the first state of the DP of the ATP and assigns orders and trips to cooks and vehicles, respectively, as soon as possible, one after another according to the obtained states along the DP. When obtaining infeasible states in the ATP because of freshness and/or synchronization between orders in the same trip and/or constraints on the given sequences (according to the partial decision), it returns to the first assigned order for which its starting time could have caused the state's infeasibility. Then, it postpones this order only as much as necessary and also some of the orders that followed it (along the DP) as well as some of the trips so that the infeasible state becomes feasible (if possible). This means that the final result of the PDFT is that all trips depart as soon as possible given state S_k and the partial decision, which minimizes the delay. We design stopping conditions when no feasible solution exists and observe that the PDFT determines infeasibility of a partial decision if and only if it is infeasible, and otherwise, it solves the ATP to optimality. In our computational experiments, we limit the number of iterations to 25. However, in about 99% of cases, the PDFT terminates earlier because either infeasibility was confirmed or a feasible decision was found. A detailed analysis of the functionality of PDFT can be found in Online Appendix A.7.

4.3. Decision Evaluation

Ideally, we want to evaluate each feasible decision x_k in the LNS exactly, that is, with respect to the sum of its immediate cost $D^{\Delta}_k(S_k, x)$ and the expected cost to go of the postdecision state, $\mathcal{V}(S^x_k)$. Whereas the immediate cost can be computed directly as defined in Equation (3), the value function is unknown and may only be approximated. As decisions are evaluated in every state and every iteration of the LNS, it is crucial that the approximation of the value function is computationally fast. For that reason, we use a VFA given by a NN $\widehat{\mathcal{V}}_{\omega}$, using a set of features associated with the postdecision state. These features are based on aggregated values of the postdecision state. In our case, the aggregated postdecision states are defined by summary statistics reflecting our available resources, that is, the current status of cooks and vehicles (see Online Appendix A.5). The aggregation of the postdecision state is required for two reasons. First, the dimension of the postdecision state can be vast and, therefore, impede the approximation accuracy of the VFA, a phenomenon known as the curse of dimensionality. Second, we employ the same NN for instances of different numbers of cooks and vehicles, a practice that is called transfer learning, which requires the dimension of the network's input to remain fixed. Thus, aggregation of postdecision states of different sizes to the same features is required. In Online Appendix A.7, we show the value of transfer learning.

The mapping of aggregated postdecision states to their estimated cost to go is defined by the NN's weights ω, fitted once in an extensive off-line simulation. Afterward, the fitted VFA $\widehat{\mathcal{V}}_{\omega}$ can be invoked directly, when employed in the online simulation to evaluate the cost to go of a decision. To fit the network's weights ω, we minimize the expected deviation of the NN's estimate of the cost to go $\widehat{\mathcal{V}}_{\omega}(\mathcal{F}(S^x_k))$ from the observed cost to go $\sum_{t=k+1}^{K} D^{\Delta}_k$ for each realized postdecision state S^x_k. The observed costs to go are only known in hindsight, that is, after simulating the entire day. Thus, we only update the weights in the NN at the end of each simulated day. To do so, we save all observed tuples of aggregated postdecision states and its associated observed costs to go in a memory. When updating the NN at the end of a day, we consider all previously observed tuples in the memory (observed during this day and previous days) by randomly selecting a batch of tuples from the memory. This is done to avoid statistically dependent samples in the update step. Then, we compute the mean deviation (as given by the mean squared error) of the network's approximated value and the observed value. The gradient of the deviation with respect to our weights ω serves as a search direction to update the weights ω and, therefore, to improve the quality of our VFA. We repeat this off-line simulation until the NN's approximation quality converges. We then use the approximated values in the Bellman equation when searching the decision space via LNS. In the special case of the last, deterministic decision point at time T^c, the future cost to go is zero, and therefore, we set the value function to zero as well. We describe the details of the NN features and architecture and its update procedure in Online Appendix A.5.

The described LNS procedure to update the planned schedules $(\Psi_k, t^s_k, \Theta_k, t^d_k)$ without evaluation via VFA is used in Section 5.3 as a second benchmark that we refer to as integrated. That is, the evaluation of each decision by this benchmark is made with respect to the immediate cost $D^{\Delta}_k(S_k, x)$ only.

5. Numerical Study

In this section, we describe instances and benchmark policies, analyze the performance of our method as well as the value of ghost kitchens, and derive managerial insights.

5.1. Instances

For all experiments, we assume service in Iowa City with customer locations from Ulmer et al. (2021). The ghost kitchen hosts $|J| = 5$ restaurants and is positioned in the center of Iowa City. Vehicles travel in the OpenStreetMap network with free-floating travel times (Boeing 2017). We assume that orders can be placed the entire day ($T^c = 1{,}440$). However, they usually accumulate around lunch and dinner times. For this reason, setting $T = T^c + 120$ is sufficient in our experiments.

Following the literature, we assume two demand peaks around lunch and dinner time. The number of lunch and dinner orders is given by random variables *OD* and *OL*, each following a normal distribution. We assume that more orders are placed in the city center during lunch time, whereas orders are more frequent in residential areas during dinner time. We provide detailed information on how we sample instances in Online Appendix A.6.

Orders are equally likely for all restaurants. The preparation times of orders are drawn from a log-normal distribution that mimics the long-tail behavior observed in practice (Mao et al. 2022). We assume decreasing expected preparation times over the five restaurants in steps of one minute from 10 to 6 minutes representing different cuisines or efficiency between the restaurants. We assume a delivery promise of $\tau = 30$ minutes, a freshness constraint of $\delta_f = 20$ minutes for all food types $f \in F$, and a vehicle capacity of $\kappa = 3$ orders (we note that the capacity is rarely reached in our experiments because of the freshness constraint).

We define three main instance settings. In the small instance setting, we assume $|C_j| = 1$ cook for each restaurant j. We further assume $|V| = 5$ vehicles. The expected number of orders is 64 during lunch time and it is 100 for dinner time. For the medium setting, we keep the resources the same, but increase demand by 25%. For the large settings, we double the numbers of cooks and vehicles as well as the demand compared with the medium instances. For all settings, we create 300 days of order realizations for evaluation.

5.2. Tuning and Benchmark Methods

We train our VFA on the small instances over 10,000 simulations on realizations different than the 300 used for evaluation. We observe convergence after 4,000 simulations. In the training and implementation, we set the number of LNS iterations to 70 for each state (which takes a fraction of a second). We fine-tune the VFA each for the medium and large instances with an additional 1,000 samples.

We create four benchmark policies: two are problem-oriented and two are method-oriented. The latter two can be found in Online Appendix A.7. The two problem-oriented benchmarks follow the concepts of Rijal, Bijvank, and de Koster (2023) and D'Haen, Braekers, and Ramaekers (2023) for combined optimization of warehouse operations and vehicle routing, respectively.

- FIFO: This policy implements the first in, first out starting solution in every state. Therefore, it solves the cooking part first and then the routing part. As the FIFO solution is effective when considering the cooks in isolation and ignoring vehicles, it can be seen as the sequential approach proposed in Rijal, Bijvank, and de Koster (2023).
- Integrated: This policy considers cooking and routing together but does not anticipate future developments. It applies the same LNS as our policy but minimizes immediate delay only.

5.3. Policy Comparison

First, we compare our policy with the problem-oriented benchmarks. Table 1 summarizes the results of our suggested solution method on the small, medium, and large instances compared with the two problem-oriented benchmark policies. The table lists (i) the instance type, (ii) the KPI examined ("late orders" refer to orders with a positive delay), (iii)–(v) the tested policies, and (vi and vii) the average improvement of our AI heuristic over FIFO and integrated, respectively, for the corresponding instance type and KPI. All values are rounded to one digit, but the improvement is calculated on the original values.

Depending on the instance and method, the average delay values range between 4.9 and 26.8 minutes, which falls within ranges observed in practice (Mao et al. 2022). We observe that our policy improves all KPIs for all three instance settings. For the small instances, the average delay can be reduced substantially by 17.0% compared with integrated and even by 106.5% compared with FIFO. The same tendency can be observed for the medium and large instances even though the improvements are less extreme with 9.4% and 68.1% for the medium instances and 18.5% and 42.2% for the large instances.

The stepwise increase in solution quality from FIFO to integrated to our policy confirms the effectiveness of both parts of our policy: the LNS part for searching and the VFA part for evaluating the decision space. The already large difference between FIFO and integrated shows the value of joint optimization of scheduling and routing. The superiority of our policy for the medium and large instances also confirms the success of our transfer learning approach. In Online Appendix A.9, we further show that, whereas the average delay depends on the net travel time from the ghost kitchen to the customer, our policy improves this KPI for customers across the city. It also shows that the average delay is linked to the restaurant's performance. Longer expected preparation times result in more delays for the respective customers of the restaurant.

Whereas the objective of our problem and methodology is to minimize delay, we also observe significant improvements in other KPIs. Our method reduces the percentage of customers experiencing delay, the average delay, and the maximum delay a customer experiences per day. Thus, we have fewer dissatisfied customers, and these customers are dissatisfied less. We also observe that the average click-to-door time decreases; thus, customers receive their food faster. This result is not self-evident as Ulmer et al. (2021) show that optimization for delay may trade fast

Table 1. Average Results for the Small, Medium, and Large Instances

Instances	KPI	FIFO	Integrated	AI	Percentage improvement over FIFO	Percentage improvement over integrated
Small	Average delay, minutes	10.1	5.8	4.9	106.5	17.0
	Percentage of late orders	43.4	36.4	34.3	26.4	6.1
	Average delay of late orders, minutes	22.2	14.7	13.3	66.8	10.8
	Maximum delay, minutes	52.6	45.4	44.0	19.5	3.1
	Average click-to-door, minutes	34.7	29.9	29.3	18.5	2.1
	Average number of orders/trip	1.3	1.3	1.4	6.7	3.0
	Average total travel time, minutes	2,543.6	2,476.5	2,416.8	5.2	2.5
	Average cooks finish time, minutes	1,271.5	1,261.9	1,260.1	0.9	0.2
Medium	Average delay, minutes	26.8	17.5	15.9	68.1	9.4
	Percentage of late orders	60.8	54.9	54.5	11.6	0.8
	Average delay of late orders, minutes	44.0	31.5	29.0	51.9	8.9
	Maximum delay, minutes	101.4	82.2	78.4	29.3	4.8
	Average click-to-door, minutes	53.2	43.5	42.2	26.2	3.2
	Average number of orders/trip	1.3	1.4	1.4	10.5	4.2
	Average total travel time, minutes	3,154.9	3,020.0	2,940.5	7.3	2.7
	Average cooks finish time, minutes	1,321.6	1,295.3	1,289.9	2.4	0.4
Large	Average delay, minutes	15.3	12.8	10.8	42.2	18.5
	Percentage of late orders	46.8	44.4	44.1	6.0	0.7
	Average delay of late orders, minutes	32.5	28.4	24.0	35.5	18.3
	Maximum delay, minutes	69.8	64.8	60.5	15.3	7.1
	Average click-to-door, minutes	40.3	37.7	36.0	12.0	4.7
	Average number of orders/trip	1.5	1.5	1.5	4.9	4.2
	Average total travel time, minutes	5,926.7	5,869.3	5,679.1	4.4	3.4
	Average cooks finish time, minutes	1,294.8	1,289.0	1,280.2	1.2	0.7

deliveries for fewer late deliveries. Finally, we observe that our policy increases the average bundle size and reduces the average travel times by several percentages. Evidently, there is a significant difference between AI and integrated. When looking closer at the decision making, we observed that the main differences in bundling between the two policies occur during the lunch and dinner peak times. That is, the anticipation of AI allows preparing for the peaks and, consequently, leads to more effective decision making during the peaks.

Finally, we observed that with AI, cooks finish preparing all orders between 10 and 30 minutes earlier compared with FIFO and between 1.8 to 8.8 minutes earlier compared with integrated. For a closer analysis of the workforce utilization, we refer to Online Appendix A.8, in which we show that AI leads to a more effective utilization especially at the beginning of peaks and to a substantially smaller backlog at the end of the peaks. In essence, the reduction in travel times and the increase in bundling opportunities does not only come from an elaborate search of the decision space, but also from the anticipation of future orders.

5.4. The Value of Ghost Kitchens

Setting up a ghost kitchen is a strategical decision that might come with several advantages in delivery times, resource usage, and food freshness compared with conventional meal delivery operations. In the following, we quantify these advantages in a new experiment based on our previous three instances that we evaluated in Section 5.3. We use the same realization of demand including customer locations, order times, food type, and preparation times. However, each food type is now prepared by a corresponding independent restaurant that is located within Iowa City. The locations of the restaurants are obtained by a k-medoid clustering of real restaurant locations in Iowa City, for which k is given by the number of food types. We add Gaussian noise (with a mean of zero and a variance of 20% of the expected preparation time) to the preparation times in order to represent that traditional restaurants are affected by disturbances in their preparation process (Giousmpasoglou, Ladkin, and Marinakou 2022). Unlike ghost kitchens, we assume that independent restaurants cannot synchronize preparation schedules across locations, limiting bundling opportunities. Each restaurant operates a single queue per cook with orders assigned to the cook with the shortest queue and prepared in a first in, first out manner. The schedule cannot be modified ex post. This limitation means freshness constraints cannot be guaranteed in this setting and are, therefore, excluded. To ensure a fair comparison, we also relax the freshness constraints in the ghost kitchen setting during this experiment.

The delivery process is handled by a fleet of vehicles operated by the platform. In practice, independent restaurants consolidate orders to balance delivery efficiency and costs (e.g., Uber Eats and DoorDash), which employ bundling strategies to optimize delivery routes (e.g., Reyes et al. 2018, Ulmer et al. 2021). We use the assignment and routing policy of Ulmer et al. (2021) combined with a repositioning strategy in which vehicles return to the closest restaurant if they are idle.

We summarize our results in Table 2. The table lists (i) the instance type, (ii) the reported KPIs including the mean freshness of orders for the setting of (iii) one ghost kitchen without freshness constraints using the AI policy, (iv) for the same setting but using the FIFO policy, and (v) the setting of independent dine-in restaurants as well as (vi) the ghost kitchen's (using the AI policy) improvement on the setting of independent dine-in restaurants for each KPI.

5.4.1. Service Quality. We observe that ghost kitchens provide higher service quality and operational efficiency compared with independent restaurants. This includes lower average delays in ranges of 5%–56%, reduced percentages of late orders in ranges of 18%–38%, and shorter maximum delays in ranges of 43%–74%. The centralized planning in ghost kitchens ensures that meals are prepared in alignment with the delivery driver's departure time. This leads to reduced idle times and fresher food. In contrast, independent restaurants cause delays and loss of freshness because of uncertain preparation times and decentralized planning. The average delay of late orders is the only KPI that worsens (for medium and large instances). This is likely caused by statistical artifacts because of fewer delayed orders.

5.4.2. Resources and Costs. Ghost kitchens reduce travel times by 4%–12% which offers several advantages: Operational costs decrease as platforms spend less on fuel, maintenance, and driver compensation. Drivers can deliver more orders in the same shift, increasing productivity. Shorter routes also reduce emissions, supporting sustainability goals, and enhance driver welfare by minimizing time spent on the road.

5.5. Problem Analysis

In this section, we perform a sensitivity analysis and study the value of collaboration. In our main experiments for the large setting, we assume a freshness constraint of $\delta_f = 20$ minutes, a delivery promise of $\tau = 30$ minutes, $|C_j| = 2$ cooks per restaurant, $|V| = 10$ vehicles, moderate standard deviations $\sigma(t_i^p \mid f_i = f)$ in preparation times for all customers $i \in I$, and food types $f \in F$ and expected demand values of $\mathbb{E}[OL] = 160$ and $\mathbb{E}[OD] = 250$. We now vary the parameters individually and analyze their impact on the KPIs. The results of our AI policy are shown in Table 3.

Table 2. Comparison of One Ghost Kitchen with Multiple, Independent Restaurants

Instances	KPI	Ghost kitchen (AI)	Ghost kitchen (FIFO)	Independent restaurants	Percentage improvement over independent restaurants
Small	Average delay, minutes	4.5	6.4	7.0	55.6
	Percentage of late orders	34.6	43.4	47.6	37.6
	Average delay of late orders, minutes	12.0	13.8	14.6	21.7
	Maximum delay, minutes	41.4	42.7	63.2	52.7
	Average click-to-door, minutes	29.1	31.2	32.6	12.0
	Average freshness, minutes	15.7	17.8	19.3	22.9
	Average number of orders/trip	1.5	1.8	1.3	13.3
	Average total travel time, minutes	2,389.2	2,304.1	2,672.5	11.9
Medium	Average delay, minutes	13.1	16.2	13.8	5.3
	Percentage of late orders	52.5	60.4	62.2	18.5
	Average delay of late orders, minutes	24.2	26.2	22.1	–8.7
	Maximum delay, minutes	75.7	71.1	108.9	43.9
	Average click-to-door, minutes	39.3	42.7	40.8	3.8
	Average freshness, minutes	22.6	25.9	23.8	5.3
	Average number of orders/trip	1.7	2.0	1.5	11.8
	Average total travel time, minutes	2,902.8	2,770.9	3,014.3	3.8
Large	Average delay, minutes	7.8	9.7	8.5	9.0
	Percentage of late orders	43.8	44.8	52.0	18.7
	Average delay of late orders, minutes	17.4	21.2	16.3	–6.3
	Maximum delay, minutes	51.4	51.8	89.2	73.5
	Average click-to-door, minutes	33.4	34.9	34.7	3.9
	Average freshness, minutes	16.6	15.6	21.5	29.5
	Average number of orders/trip	1.8	1.8	1.5	20.0
	Average total travel time, minutes	5,417.3	5,540.2	6,057.8	11.8

Table 3. Sensitivity Analysis

						Delay, minutes		
Instance	Parameter	Original value	Updated value	Average number of orders/trip	Percentage of late orders	Average	Maximum	Average late orders
L1	$\delta_f, \forall f \in F$	20	15	1.2	53.1	24.1	99.5	45.6
L2	$\delta_f, \forall f \in F$	20	25	1.8	43.8	7.8	51.4	17.4
L3	τ	30	25	1.5	55.0	12.9	64.8	23.3
L4	τ	30	35	1.5	37.4	8.7	55.0	22.4
L5	$\|C_j\|, \forall j \in J$	2	1	1.3	72.8	51.7	262.5	71.0
L6	$\|C_j\|, \forall j \in J$	2	3	1.5	39.7	10.6	60.0	26.0
L7	$\|V\|$	10	7	1.5	71.7	53.6	190.0	74.8
L8	$\|V\|$	10	13	1.5	25.1	3.1	41.4	11.4
L9	$\sigma(t_i^p)$	$\sigma(t_i^p)$	0	1.6	40.7	9.9	54.4	23.9
L10	$\sigma(t_i^p)$	$\sigma(t_i^p)$	$2 \cdot \sigma(t_i^p)$	1.5	53.2	18.4	119.2	34.4
L11	$\mathbb{E}[OL], \mathbb{E}[OD]$	160,250	144, 225	1.5	36.4	6.4	44.8	16.7
L12	$\mathbb{E}[OL], \mathbb{E}[OD]$	160,250	176, 275	1.5	52.2	19.9	88.2	38.0
Large				1.5	44.1	10.8	60.5	24.0

5.5.1. Freshness Constraint.

The first two instances, L1 and L2, decrease and increase the freshness constraint by five minutes, respectively. We observe that the average delay differs significantly between the two instances. One reason is that, with a more relaxed freshness constraint, the number of orders per trip can be increased significantly (1.2 to 1.5 and even 1.8). This consolidation frees vehicle resources and allows for faster deliveries. This illustrates a trade-off for restaurants and ghost kitchens. If customers want even fresher food, they likely pay for it by longer waiting for their meal to get prepared and delivered. We further note that average and maximum delay values are also indicators for overall working times for cooks and vehicles. Thus, to ensure fresh food, the platform has to pay its employees longer.

5.5.2. Delivery Promise.

Instances L3 and L4 decrease and increase the delivery promise by five minutes, respectively. The difference in delivery promises between the two instances is almost equal to the difference in observed maximum delay (9.8 minutes) for the two instances. The difference in average delay is about 6.4 minutes, thus, smaller than 10 minutes, because in L4 some deliveries arrive earlier than the deadline.

5.5.3. Resources.

Instances L5 to L8 vary the number of cooks (L5, L6) and the number of vehicles (L7, L8). The results indicate that reducing any resources leads to a tight bottleneck and unacceptably high delays, whereas increasing any resources improves the service quality. This confirms our expectation. Notably, when the number of cooks is reduced in L5, the number of orders/trip decreases as well (from 1.5 to 1.3). Whereas bundling increases efficiency in the vehicle utilization, it likely reduces efficiency for the cooks because of the synchronization requirements for orders that share trips. Our method recognizes that cooks are the bottleneck, and vehicles are dispatched more often instead of enforcing consolidation. In essence, the ghost kitchen provider should carefully balance the two resources required for the operations as a lack in one can hardly be compensated by the other. In Online Appendix A.10, we analyze the impact of the number of cooks on workload distribution between cooks and vehicles. Whereas the average cook workload is determined by the total preparation time divided by the number of cooks (and, therefore, decreases linearly from L5 to L to L6), the vehicle workload depends on trip decisions. Specifically, when cooks are the primary bottleneck (as in L5), the vehicle workload increases despite the unchanged set of orders because of reduced bundling efficiency. Interestingly, in L6, with a higher number of cooks, vehicles benefit from improved timing of food preparation, which helps reduce the total delay objective. However, this comes at the cost of less efficient bundling, resulting in higher travel times compared with L.

5.5.4. Preparation Times.

Instances L9 and L10 vary the standard deviation of preparation times by either setting it to zero or doubling it. We observe that the value of no variance in preparation time (9.9 average delay) is only slightly smaller compared with our standard instance (10.8 average delay). This suggests that extended preparation times alone are not the primary cause of increased delays. When the variance is high, the system experiences a breakdown with an average delay of 40.0 minutes. This suggests that, whereas the system can tolerate minor variations in preparation times, it is crucial for the ghost kitchen provider to keep these fluctuations within manageable limits.

5.5.5. Demand.

Instances L11 and L12 decrease and increase demand by 10%. As expected with less demand, average delay decreases, and with more demand,

Table 4. The Value of Collaborating Cooks and Vehicles

	Cooks			Vehicles	
KPI	No	Pairwise	Full	Pairwise	Full
Average delay, minutes	16.7	10.8	10.0	34.3	10.8
Percentage of late orders	51.5	44.1	38.2	51.8	44.1
Average delay of late orders, minutes	32.2	24.0	25.2	65.8	24.0
Average click-to-door, minutes	42.8	36.0	34.4	60.5	36.0
Maximum delay, minutes	89.7	60.5	54.6	181.8	60.5
Average number of orders/trip	1.5	1.5	1.5	1.3	1.5

average delay becomes significantly worse. Thus, accurate demand predictions are crucial to manage resources accordingly.

Finally, we analyze the value of shared resources between the restaurants. In our initial setting, we assumed full collaboration among the vehicle fleet and pairwise collaboration of two cooks per restaurant. We now vary the collaborations between no, pairwise, and full for the two types of resources individually. The results are shown in Table 4. No collaboration between cooks means that we assume 10 food types with single cooks instead of five food types with two cooks each. We observe that the delay increases in this case. Thus, even the pairwise collaboration is already very beneficial. The delay reduces even further, but only slightly, for the full collaboration case in which every cook can prepare every food (i.e., one large restaurant). The limited improvement is likely because the vehicles become the main bottleneck in that case. For vehicles, we observe a similar but more severe result. Delay goes up and orders per trip go down. Assigning dedicated vehicles to each restaurant is undesirable as it negates a key advantage of the ghost kitchen model: centralized, anticipatory preparation scheduling and coordinated deliveries across multiple restaurants. This experiment reveals two main insights: First, a careful balance of the resources is required for effective operations. Second, the central dispatching of a joint fleet is one of the major advantages of a ghost kitchen.

6. Discussion and Opportunities for Future Work

In this work, we analyze the concept of ghost kitchens and show how anticipatory and integrated optimization of scheduling and routing can improve operations and customer experience. We present a novel solution method to synchronize between the first (preparation) and second (dispatching) stages of a ghost kitchen's operation.

6.1. Method Generality

Our solution method consists of several components whose usefulness are not limited to the RMD-GK and can also be used in other problems that have a large decision space and/or real-time decision making under uncertainty. For instance, the notion of a condensed decision representation can be used to reduce the search space (ideally without loss of optimality) in problems in which homogeneous resources are considered, for example, vehicle fleet management and parallel machine scheduling. Similarly, searching over partial decisions and determining timing by a separate algorithm can be a successful approach for various scheduling and assignment problems. This is the case when the starting times of tasks can be defined as a subproblem of the scheduling problem, which may be easier to solve given the assignment of tasks at the resources. This motivates searching over assignment decisions only and determining the starting times according to a separate algorithm. Finally, our methodology can be used in other complex problems in which two sequential stages need to be coordinated, for example, when facing joint dynamic optimization of warehouse operations and vehicle routing (Rijal, Bijvank, and de Koster 2023). In such cases, infeasibility often arises because of the need to synchronize the stages. Analytical analysis that excludes a substantial number of infeasible solutions can significantly enhance the efficiency of the solution method as demonstrated for the RMD-GK.

6.2. Future Research

Our numerical study reveals that synchronization achieves a significant improvement in all important KPIs and that interesting and insightful connections exist between the availability and use of the resources allocated to the two stages. This is, again, a promising direction for future research for other problems with similar characteristics. There are also several additional avenues for future research in problem and method.

6.2.1. Hybrid Systems and Customer Order Flexibility.

In our work, we show that operating a ghost kitchen has several advantages compared with conventional meal delivery systems by analyzing both systems in isolation, each with a dedicated fleet. Future research could explore hybrid systems that integrate both traditional restaurants and ghost kitchens as this could combine the strengths of both models, potentially improving flexibility, resource utilization, and responsiveness to varying demand. Additionally, studying how ghost kitchens can be implemented with crowdsourced delivery services could uncover scalable and cost-effective approaches to meet peak demands more efficiently. Further, we assume that customers still order from one restaurant only. However, an additional advantage of ghost kitchens might be that customers may order from different restaurants and receive their food at the same time (Arslan, Agatz, and Klapp 2021). Future research may extend the presented model and methodology to allow for multiple orders and analyze its impact on the system's performance because such an option may lead to less flexibility in the ghost kitchen and longer delivery times. We further show that the freshness constraint has a significant impact on the delivery times.

6.2.2. Freshness Constraints, Cook Collaboration, and Preparation Time Variance. Future research may exploit this insight, for example, by dynamically relaxing the constraint during peak times. Our results also indicate that the collaboration among cooks can be beneficial. The setup and assignment of cooks may deserve a closer investigation in the future. For example, some all-around cooks may be assigned dynamically between the restaurants (or even to vehicles) based on realized and expected demand. We further see that the variance in preparation times of the restaurants determines success or failure of the entire operation. Future work may investigate this in more detail, for example, by analyzing how a single restaurant's performance impacts the service quality of the other restaurants or by modeling preparation time uncertainty.

6.2.3. Equity. Moreover, exploring the fairness between customers, cooks, and vehicles could be an intriguing avenue for future research, potentially leading to an even more balanced and equitable system-wide performance.

6.2.4. Methodology. In our work, we present a method that searches and evaluates complex scheduling and routing decisions in an integrated fashion. There are a variety of possible extensions. In our method, we condense the decision space to search by using a condensed partial decision representation and, if feasible, create the full decision for evaluation via our VFA. In the future, VFA features may be developed to allow evaluation already on the condensed partial decision level. Whereas our PDFT algorithm was able to identify feasibility in the first iterations, future research may combine our analytical checks with checks based on machine learning (van der Hagen et al. 2022). Another extension could be to use the trained neural net for guiding the search within the LNS, for example, by prescribing the operators to use in a state.

Acknowledgments
The authors thank Julius Hoffmann for his valuable advice on the literature on stochastic dynamic customer order scheduling. Finally, the authors thank the associate editor and two referees for their valuable comments that helped improve the paper.

References
Arslan AM, Agatz N, Klapp MA (2021) Operational strategies for on-demand personal shopper services. *Transportation Res. Part C Emerging Tech.* 130:103320.

Auad R, Erera A, Savelsbergh M (2023) Courier satisfaction in rapid delivery systems using dynamic operating regions. *Omega* 121: 102917.

Auad R, Erera A, Savelsbergh M (2024) Dynamic courier capacity acquisition in rapid delivery systems: A deep *q*-learning approach. *Transportation Sci.* 58(1):67–93.

Boeing G (2017) OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Comput. Environ. Urban Systems* 65:126–139.

Dai H, Liu P (2020) Workforce planning for O2O delivery systems with crowdsourced drivers. *Ann. Oper. Res.* 291(1):219–245.

D'Haen R, Braekers K, Ramaekers K (2023) Integrated scheduling of order picking operations under dynamic order arrivals. *Internat. J. Production Res.* 61(10):3205–3226.

Drexl M (2012) Synchronization in vehicle routing—A survey of VRPs with multiple synchronization constraints. *Transportation Sci.* 46(3):297–316.

Feldman P, Frazelle AE, Swinney R (2023) Managing relationships between restaurants and food delivery platforms: Conflict, contracts, and coordination. *Management Sci.* 69(2):812–823.

Fresh KDS (2024) What is a KDS? The ultimate guide to kitchen display systems. Accessed November 25, 2024, https://www.fresh.technology/blog/what-is-a-kds.

Giousmpasoglou C, Ladkin A, Marinakou E (2022) *The Emergence of Ghost Kitchens in the Restaurant Industry: Operational and Labour Perspectives* (EuroCHRIE, Apeldoorn, Netherlands).

Heinold A, Meisel F, Ulmer MW (2023) Primal-dual value function approximation for stochastic dynamic intermodal transportation with eco-labels. *Transportation Sci.* 57(6):1452–1472.

Hildebrandt FD, Ulmer MW (2022) Supervised learning for arrival time estimations in restaurant meal delivery. *Transportation Sci.* 56(4):1058–1084.

Hildebrandt FD, Thomas BW, Ulmer MW (2023) Opportunities for reinforcement learning in stochastic dynamic vehicle routing. *Comput. Oper. Res.* 150:106071.

Hossein Nia Shavaki F, Jolai F (2021) A rule-based heuristic algorithm for joint order batching and delivery planning of online retailers with multiple order pickers. *Appl. Intelligence* 51(6):3917–3935.

Jahanshahi H, Bozanta A, Cevik M, Kavuk EM, Tosun A, Sonuc SB, Kosucu B, Başar A (2022) A deep reinforcement learning approach for the meal delivery problem. *Knowledge-Based Systems* 243:108489.

Klapp MA, Erera AL, Toriello A (2018) The dynamic dispatch waves problem for same-day delivery. *Eur. J. Oper. Res.* 271(2):519–534.

Liu Y (2019) An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones. *Comput. Oper. Res.* 111:1–20.

Liu H, Wang Y, Lee LH, Chew EP (2022) An approximate dynamic programming approach for production-delivery scheduling under non-stationary demand. *Naval Res. Logist.* 69(4):511–528.

Mao W, Ming L, Rong Y, Tang CS, Zheng H (2022) On-demand meal delivery platforms: Operational level data and research opportunities. *Manufacturing Service Oper. Management* 24(5): 2535–2542.

Moons S, Ramaekers K, Caris A, Arda Y (2017) Integrating production scheduling and vehicle routing decisions at the operational decision level: A review and discussion. *Comput. Indust. Engrg.* 104:224–245.

Neria G, Tzur M (2024) The dynamic pickup and allocation with fairness problem. *Transportation Sci.* 58(4):821–840.

Reyes D, Erera A, Savelsbergh M, Sahasrabudhe S, O'Neil R (2018) The meal delivery routing problem. Technical report, Georgia Institute of Technology, Atlanta.

Rijal A, Bijvank M, de Koster R (2023) Dynamics between warehouse operations and vehicle routing. *Production Oper. Management* 32(11):3575–3593.

Rivera AEP, Mes MR (2017) Anticipatory freight selection in intermodal long-haul round-trips. *Transportation Res. Part E Logist. Transportation Rev.* 105:176–194.

Rivera AEP, Mes MR (2022) Anticipatory scheduling of synchromodal transport using approximate dynamic programming. *Ann. Oper. Res.* 318(1):685–712.

Shapiro A (2023) Platform urbanism in a pandemic: Dark stores, ghost kitchens, and the logistical-urban frontier. *J. Consumer Culture* 23(1):168–187.

Silva M, Pedroso JP, Viana A (2023) Deep reinforcement learning for stochastic last-mile delivery with crowdshipping. *EURO J. Transportation Logist.* 12:100105.

Steever Z, Karwan M, Murray C (2019) Dynamic courier routing for a food delivery service. *Comput. Oper. Res.* 107:173–188.

Ulmer MW, Savelsbergh M (2020) Workforce scheduling in the era of crowdsourced delivery. *Transportation Sci.* 54(4):1113–1133.

Ulmer MW, Erera A, Savelsbergh M (2022) Dynamic service area sizing in urban delivery. *OR Spectrum* 44(3):763–793.

Ulmer MW, Goodson JC, Mattfeld DC, Thomas BW (2020) On modeling stochastic dynamic vehicle routing problems. *EURO J. Transportation Logist.* 9(2):100008.

Ulmer MW, Thomas BW, Campbell AM, Woyak N (2021) The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times. *Transportation Sci.* 55(1):75–100.

van der Hagen L, Agatz N, Spliet R, Visser TR, Kok L (2022) Machine learning–based feasibility checks for dynamic time slot management. *Transportation Sci.* 58(1):94–109.

Xu X, Zhao Y, Wu M, Zhou Z, Ma Y, Liu Y (2016) Stochastic customer order scheduling to minimize long-run expected order cycle time. *Ann. Oper. Res.* 248(1–2):515–529.

Yildiz B, Savelsbergh M (2019) Provably high-quality solutions for the meal delivery routing problem. *Transportation Sci.* 53(5): 1372–1388.

Zhang J, Liu F, Tang J, Li Y (2019) The online integrated order picking and delivery considering pickers' learning effects for an O2O community supermarket. *Transportation Res. Part E Logist. Transportation Rev.* 123:180–199.

Zhao Y, Xu X, Li H, Liu Y (2018) Stochastic customer order scheduling with setup times to minimize expected cycle time. *Internat. J. Production Res.* 56(7):2684–2706.