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

INFORMS JOURNAL ON APPLIED ANALYTICS
Vol. 54, No. 3, May–June 2024, pp. 211–221
ISSN 2644-0865 (print), ISSN 2644-0873 (online)

Data-Driven Order Fulfillment Consolidation for Online Grocery Retailing

Yang Wang,[a] Tong Wang,[b,*] Xiaoqing Wang,[b] Yuming Deng,[b] Lei Cao[b]

[a] IDG Capital, Beijing 100005, China; [b] Alibaba Group, Hangzhou 310052, China
*Corresponding author
Contact: yang_wang@idgcapital.com (YW); buzzbuxx@gmail.com, https://orcid.org/0000-0003-1815-5716 (TW); robin.wxq@alibaba-inc.com (XW); yuming.dym@alibaba-inc.com (YD); huaju.cl@alibaba-inc.com (LC)

Received: September 1, 2022
Revised: May 20, 2023; July 6, 2023
Accepted: July 7, 2023
Published Online in Articles in Advance: October 12, 2023

https://doi.org/10.1287/inte.2022.0068

Abstract. Improving fulfillment efficiency is critical for long-term sustainability of online grocery retailing. In this paper, we study reducing order fulfillment cost by order consolidation. Motivated by the observation that a significant percentage of buyers place multiple orders within a short time interval, we propose a scheme that attempts to consolidate such "multiorders" to reduce the number of parcels and hence, the shipping cost. At the same time, it cannot significantly disturb the existing order fulfillment process or undermine the customer service level. Successful execution of the scheme requires a prediction of multiorder probabilities and a control policy that selectively prioritizes order processing. For the prediction task, we formulate a binary classification problem and use machine-learning algorithms to predict in real time the probability of a multiorder. For the control task, our proposal is to hold arriving orders in a temporary order pool for potential consolidation and to determine the release timing by a dynamic program. The proposed solution is estimated to capture 92.8% of all the multiorders at the cost of holding the orders for about 20.3 minutes on average. This translates to more than 10 million U.S. dollars of order fulfillment cost saving annually.

History: This paper was refereed.

Keywords: order fulfillment • order consolidation • online grocery retailing • data-driven operations

Introduction

Online grocery retailing has become an increasingly more important battleground for all major online and traditional retailers (Griffin 2018). Yet, when compared with other more mature sectors, such as apparel and electronics, online grocery retailing is still far behind in terms of the percentage of spending done online (Saunders 2018). One critical reason lies in the unique challenge it is facing. On one hand, grocery retailing has low gross profit per order because both the average value per order and the gross margin are relatively low. On the other hand, order fulfillment cost is high because a typical grocery order consists of a larger number of items, which are often heavy in weight (e.g., drinks, detergents), bulky in volume (e.g., toilet paper), or demand special care (e.g., cold storage for ice cream). In addition, grocery buyers expect fast (same day and next day in our context) and to-door delivery.

The imbalance between low gross profit and high fulfillment cost makes order fulfillment efficiency the key to the short-term survival and long-term sustainability of online grocery retailing. Industrial practitioners and academic researchers have attempted to approach the problem from various perspectives: (1) strategic location of fulfillment centers (FCs) so as to strike a balance between the need to be decentralized and close to end customers to save delivery costs and the benefit of inventory pooling via centralization (Lim et al. 2017); (2) optimized scheme of matching item inventory and FCs so as to minimize the chance of order splitting, the situation when the items in an order have to be fulfilled from more than one FC and in multiple parcels, or to minimize return-related costs (He et al. 2019); and (3) dynamic assignment of orders to FCs based on real-time inventory information and demand prediction (Xu et al. 2009). See chapter 10 of Fernie and Sparks (2009) for a comprehensive survey of the logistics challenges. We approach the problem from a more operational perspective and study schemes of consolidating orders into fewer parcels to reduce fulfillment cost.

Business Context and Fulfillment Process

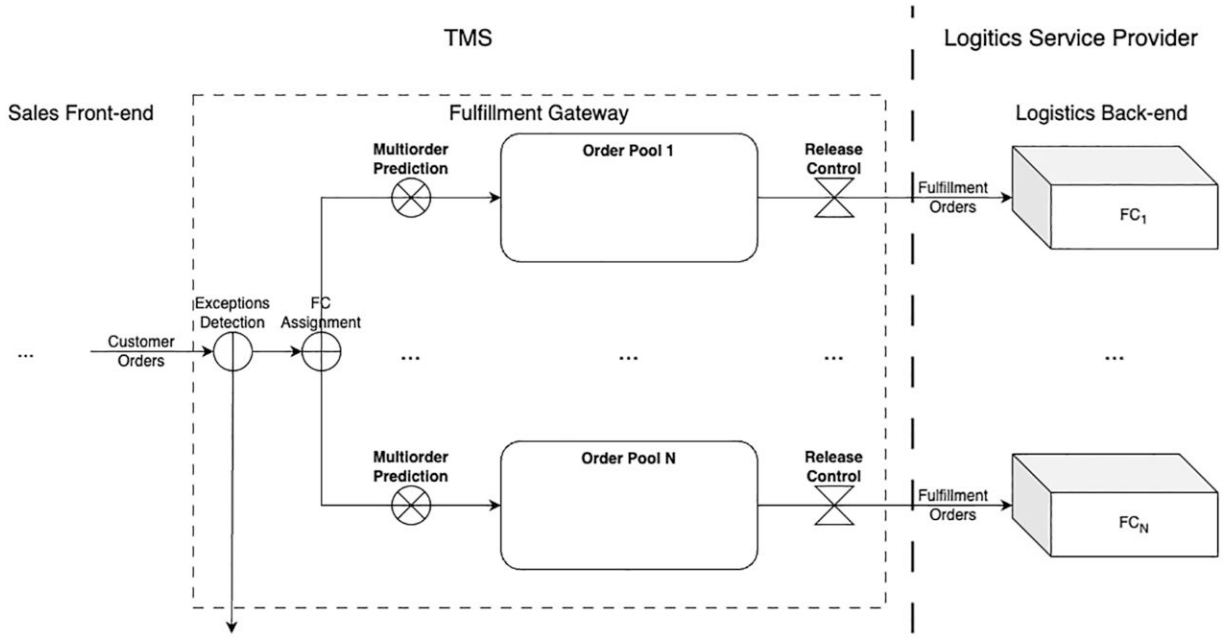
Our business context is an anonymous online grocery retailer, referred to as TMS, based in China. (In this paper, we provide visual illustrations and numerical

summaries and statistics based on real data we obtain from TMS. Because of concerns of confidentiality, we cannot publicize all the data in detail. We regretfully censor all the absolute numbers and only report the order of magnitude, relative proportions, and relative patterns. For the figures, we have to remove the labels and ticks on the axis wherever sensitive data are in place.) The market positioning of TMS is a one-stop online supermarket (with hundreds of thousands of stock-keeping units) with fast delivery (same day or next day). Different from online marketplaces and platforms. such as taobao.com or tmall.com, TMS is a fully integrated retailer that manages its own marketing and sales activities on the demand side and procurement, inventory, and supply chain operations on the supply side as well. Warehousing and delivery to end customers are outsourced to a third-party logistics service provider, which charges TMS a fixed cost per year and a variable cost per parcel delivered.

Figure 1 provides an illustration of the order fulfillment process. There are three major parts from left to right: sales front end, fulfillment gateway, and logistics back end. The fulfillment process starts when a customer order arrives from the sales front end after payment, confirmation, etc. Before we introduce order fulfillment consolidation, the fulfillment gateway is mainly responsible for exceptions detection and FC assignment. Exceptions may involve wrong address, risk of fraud, virtual products that do not need physical fulfillment, and various ad hoc business rules. Those orders with exception will be redirected into another process that is irrelevant to the context of this paper. In the next step, the remaining orders are assigned to one or more FCs. Here, one needs to solve an assignment/matching problem to minimize the total fulfillment cost, subject to inventory availability constraints. Note that it is common to split a customer order with multiple items into two or more parcels, referred to as fulfillment orders in Figure 1, because not all items are available in one single FC. The following steps, namely multiorder prediction, order pool, and release control as marked in bold in Figure 1, are the key components of our proposed order fulfillment consolidation and will be explained in detail later. The output of the fulfillment gateway is the fulfillment orders, which are linked to the customer orders in a many-to-one fashion because of order splitting (it becomes many to many after order consolidation). The fulfillment orders are then executed by the FCs in the logistics back end. Execution includes item picking, parcel packing and labeling, and then, delivery to the customers.

Among the three parts of the fulfillment process, TMS manages the sales front end and fulfillment gateway, whereas the logistics service provider runs the logistics back end and charges TMS based on the number of fulfillment orders or equivalently, parcels. For TMS to reduce its fulfillment cost, one critical task is to minimize the number of fulfillment orders sent to the logistics back end.

Figure 1. The Order Fulfillment Process Shows How a Customer Order Received in the Sales Front End Goes Through the Fulfillment Gateway and Reaches the Fulfillment Center in the Logistics Back End



Order Fulfillment Consolidation

The starting point of this project is the observation that we may combine multiple customer orders into one fulfillment order and have them fulfilled in one parcel. This (at least) halves the shipping cost paid to the logistics service provider because the charge is on a per-parcel basis. The scheme itself is not an innovation but a rather common practice (Amazon.com 2019).

We use the term *multiorder* to refer to the collection of customer orders that can be fulfilled together. More formally, a multiorder is a set of two or more orders that satisfy the following criteria: the orders (1) are placed by the same buyer, (2) are placed on the same day, (3) are of the same delivery address, (4) are assigned to the same FC, and (5) satisfy the free shipping requirements. (For nonfree-shipping orders, because the customer has paid for shipping, consolidation is not considered to avoid potential public relation or legal issues.)

One simple solution to capture the benefits is to offer a grace period, say half an hour, within which buyers can modify their orders. This essentially allows the buyers themselves to merge their multiorders and hence, to some extent, solve our problem (see Figure 3 in the Data Exploration section for a numerical estimation for different lengths of grace period). However, this way also implies that all the orders will be put on hold throughout the grace period just because a small percentage of them might be modified. Such a scheme of universally delaying all orders is not acceptable in our context because of the same-day/next-day delivery promise.

To exploit the benefit of order fulfillment consolidation while minimizing the disturbance imposed on order processing, we need a solution that can identify orders with high multiorder probability and selectively hold them for an optimized period of time. We propose a data-driven solution that integrates machine learning and dynamic optimization. The solution, which is deployed in the fulfillment gateway, is designed to predict multiorder probabilities for all incoming orders in real time and selectively hold them in a temporary order pool before releasing to the downstream FC. The delay to an order imposed by the solution is optimized to maximize the chance for the order to be hit by another follow-up order and to make a multiorder, subject to a set of operational constraints. JD.com also has a similar practice of holding orders, although the objective is to deal with order cancellations (Medium.com 2018).

By and large, the analytics involve two major tasks: prediction and control. The prediction task is for identification of potential multiorders; upon receiving a new order, we need to predict in real time the probability of whether the same buyer will place one or more orders later in the day that satisfy the previously defined multiorder criteria, which is referred to as a *multiorder hit*. This task corresponds to the multiorder prediction step in Figure 1. Next, given the probability predictions, the control task is concerned about developing a policy to selectively hold orders in the pool to maximize multiorder hits while minimizing the influences brought to the outbound order flow and the downstream order fulfillment process. This corresponds to release control in Figure 1.

The proposed solution is back tested on the real data of TMS. Given a constraint of a maximum 30 minutes of holding time in the order pool, our solution is able to capture 92.8% of all the multiorders with an average holding time of 20.3 minutes. The estimated fulfillment cost saving is more that 10 million U.S. dollars per year.

The rest of the paper is organized as follows. In next section, we present data exploration and visualization, with which we justify our model assumptions and simplifications. The prediction algorithms are introduced in the Prediction Tasks section, and the control problem formulation and our proposed solution are described in detail in the Control Task section. We evaluate the performance in the Performance Evaluation section and conclude the paper in Conclusion. Mathematical formulations and approximation solution are relegated to the appendix.

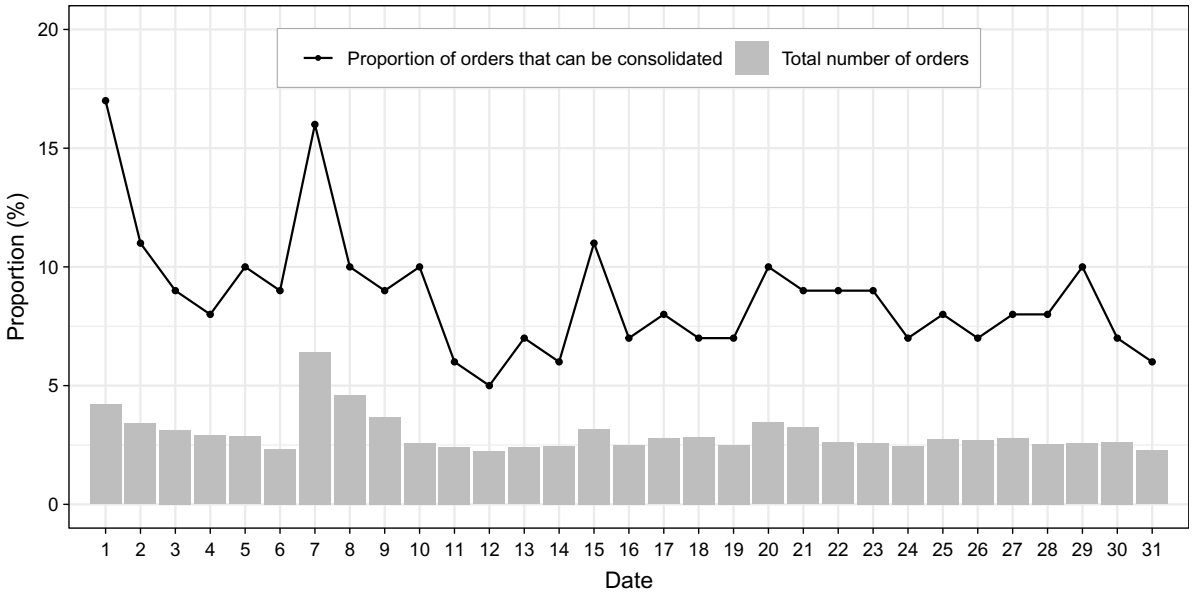
Data Exploration

In this section, we conduct exploratory analysis on the data we obtain from TMS. The analysis shall help us construct simplifying assumptions in our model and justifications to our proposed solution.

First, we try to estimate the potential cost saving by order fulfillment consolidation. Figure 2 plots a daily summary of the proportion of orders that can be consolidated (the line plot) together with the total number of orders in a month (the bar chart). On average, around 8% of all daily orders (in the magnitude of millions at TMS) can be consolidated, and the proportion can be as high as 17% during days with major promotional campaigns, such as days 1 and 7 in Figure 2. If we manage to identify and consolidate all the multiorders, we shall save fulfillment costs by 4% on average. Supposing the delivery cost of a parcel is 1 dollar (because we cannot disclose the actual cost information, we provide a lower bound here) and there are 1 million orders per day, it implies an annual cost saving of more than 15 million dollars.

Next, we zoom in on one typical day (without major promotional campaigns or marketing activities), referred to as day A. One question that we need a concrete answer for is about the multiorder structure; that is, what is the distribution of the number of orders made by a buyer on a day? The proportion of buyers

Figure 2. The Total Numbers of Orders (Bars) and the Proportions of Orders That Can Be Consolidated (Lines) Show the Potential Savings of Order Consolidation



who place a single order is 96%, and 4% place multiple orders. Among the 4%, we find that the majority of multiorders are double orders (79%) and triple orders (10%), and the remaining 11% of customers place four or more orders. In the following analysis, we make a simplifying assumption and focus only on double orders. That is, if a buyer places three orders, the third one is ignored. If there are four orders, they are considered two multiorders.

Figure 3 summarizes the empirical distribution (frequency histogram and cumulative distribution in five-minute time intervals) of interarrival time between the two adjacent orders of a multiorder on day A. By definition, the actual interarrival time can be as long as 24 hours. Here, for clearer visualization, we only plot the distribution in the interval $[0, 120]$ minutes, which covers about 80% of all instances. If we further reduce the interval to 60 or 30 minutes long, the proportions

Figure 3. The Order Interarrival Time (Truncated at 120 Minutes) Follows an Exponential Distribution

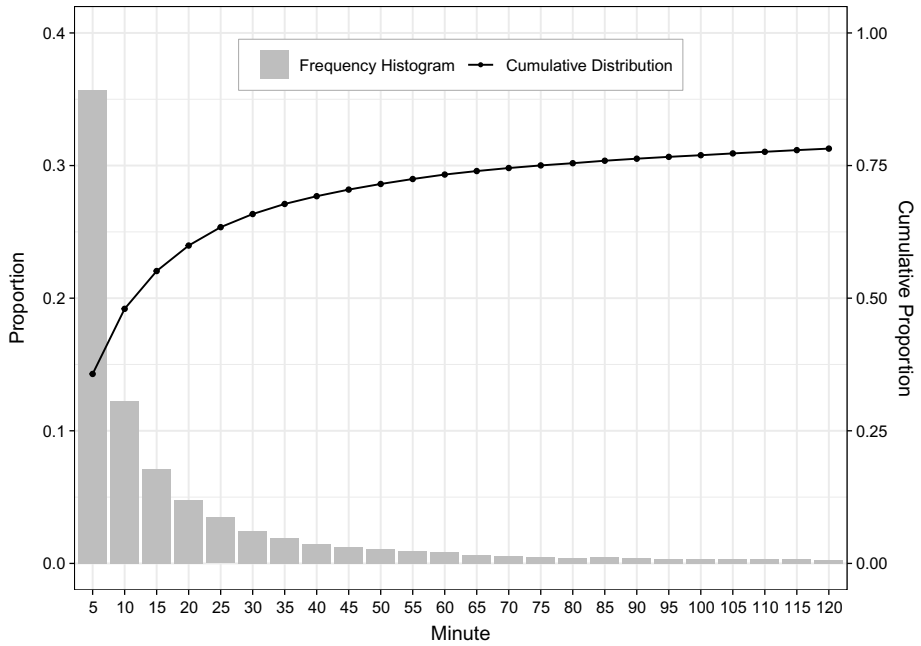
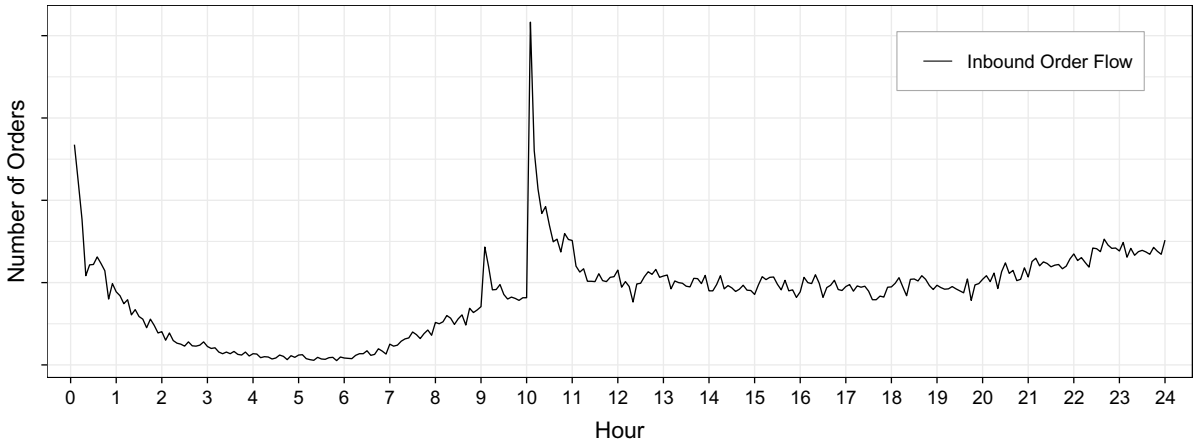


Figure 4. The Number of Order Arrivals at an FC (Data in Five-Minute Intervals) Exhibits a Nonstationary Pattern



become 75% and 65%, respectively. This provides a quick indication on the trade-off between how long the orders need to be held and how many multiorders can be potentially captured.

The plot also suggests that the interarrival time can be reasonably approximated by an exponential distribution or a mixture of multiple exponential distributions. Later in our analysis, we shall treat the interarrivals as exponential and rely on the distribution's memoryless property; the multiorder probability remains constant regardless of how long an order has stayed in the order pool.

The order arrival pattern at one of the FCs on day A can be visualized in Figure 4. The nonstationarity of the arrivals is apparent and not surprising, yet attention should be paid to the spikes at 0, 9, and 10 hours. These spikes are driven by small-scale but frequent flash sales that have been scheduled to start at those time points. Simple myopic control policies will suffer from these shocks imposed on the arrivals. A reasonable solution should incorporate such flash sales into its predictions and control the order pool size with the upcoming shocks in mind.

Prediction Tasks

In this section, we introduce the multiorder prediction algorithm. In addition, the control task also requires characterizing the uncertainty about the intensity of incoming order flow. This calls for another prediction algorithm to predict the upcoming order arrivals in the rest of the day. The two prediction tasks are described in the following subsections.

Multiorder Prediction

This part is a relatively standard binary classification problem, with the target Y variable being, for a given order, whether there will be a multiorder hit by the end of the day (labeled as positive and negative, respectively). Note that here the time horizon for multiorder prediction is set to the whole day. Apparently, delaying the fulfillment of an order for a whole day is not always practical. Later in the control task, we shall introduce parameter L, the maximum amount of time in the order pool, which is to be adjusted based on the actual business setting.

Because of concerns about business secrets, we are not able to fully disclose the complete set of features (X variables) and their relative importance in the model. A selected subset of representative features is categorized in groups and presented in the following.

1. Buyer basic features: member score, membership level, purchase-power level
2. Buyer statistical features: number of coupons on hand, shop visit frequency, page view count, purchase frequency, number of items added to shopping cart but removed later
3. Buyer multiorder features: order cancellation frequency with TMS, multiorder count with TMS
4. Order basic features: item count, total order value, total quantity, total number of categories
5. Order statistical features: total average daily sales of items in the order
6. Real-time features: real-time activities such as clicks, add to cart, and voucher redemption

The corresponding data are retrieved from TMS's database on a daily basis. Because of the sheer size of the data (millions of orders per day) and imbalance of positive and negative samples (recall that, on average, there are about 4% positive multiorder instances and 96% negative ones), we construct the training data set with 500,000 positive samples and 2,000,000 negative samples, randomly sampled from the orders in the most recent six weeks.

With the training data set, we experiment with a number of predictive machine-learning models, including gradient-boosted decision trees (GBDTs) (Chen and Guestrin 2016), Transformer (Vaswani et al. 2017), and

Table 1. We Experiment with Various Prediction Models and Evaluate Their Performance in Terms of AUC and Latency

Model	Description	AUC	Latency range (milliseconds)
V0 GBDT	Baseline GBDT model using feature groups 1, 3, and 4	0.745	[5, 15]
V1 GBDT with extended features	V0 + feature groups 2 and 5	0.756	[5, 20]
V2 GBDT with real-time features	V1 + feature group 6	0.809	[5, 20]
V3 Transformer	Prediction model using Transformer	0.821	[30, 50]
V4 Transformer + feature cross	V3 + feature interactions	0.829	[30, 50]
V5 Transformer + resampling	V4 + resampling on error-prone data points	0.833	[40, 60]
V6 BERT	Item embedding using BERT pretraining	0.841	[60, 100]

BERT (Devlin et al. 2019), and a number of feature combinations based on the groups of features listed previously. Given the nature of this paper, we have skipped a lengthy description of the models' setup and implementations and provide a brief summary in Table 1.

The output of the classification algorithms includes the multiorder probability and the label of "positive" or "negative." Because the following control task consumes only the multiorder probability, the prediction performance is measured by out-of-sample area under the receiver operating characteristic curve (AUC), which is a number between zero and one, with one being perfect prediction. The result is also listed in Table 1. We find that as we employ more powerful models (in terms of both the structural complexity and the sheer number of parameters) and introduce more relevant features, the AUC can be further improved.

The model eventually deployed is V2. This is because of another constraint imposed; the deployed model must be able to complete multiorder prediction for an order within 40 milliseconds. Otherwise, the fulfillment gateway will treat it as time-out and release the order immediately. As shown in Table 1, this constraint effectively rules out all the Transformer- and BERT-based models. Even for GBDT, we have to adjust the hyperparameters to speed it up. The final setup is presented in Table 2. The final model of choice is the result of balancing prediction accuracy and execution latency.

Order Flow Prediction

The other important input of the follow-up control task is real-time prediction of the number of orders for the rest of the day. As will be described in more detail in the section Control Task, we discretize time into five-minute intervals when applying the control algorithm.

Table 2. Hyperparameters Used in the GBDT Model

Hyperparameter	Value
Number of trees	100
Learning rate	0.05
Max leaf count	32
Sample ratio	0.8
Feature ratio	0.8

Therefore, we will need to predict the number of orders for all five-minute intervals in the rest of the day.

We rely on an existing real-time prediction system at TMS that has been built for other purposes. It generates hourly predictions and updates every five minutes. The algorithm is a GBDT-based regression using features such as historical hourly order arrivals and revenues, current day order arrivals and revenues so far, last five-minute orders and revenues, and information on promotional activities (such as flash sales).

We estimate the historical proportion of orders in each five-minute interval within each hour of a day and apply the proportions to the hourly prediction to obtain five-minute predictions.

Control Task

In the following, we first describe the problem setup of the control task. Its detailed mathematical formulation and solution procedures are relegated to the appendix. Then, we introduce two other policies that serve as benchmarks for comparison.

Problem Setup

We formulate a Markov decision process-based model that decides which orders to release from the pool at a given time and a given state of the system. The problem is a discrete-time, finite-horizon, dynamic stochastic control problem. The planning horizon is one day, which is discretized into 288 five-minute periods, denoted by $t = 1, 2, \ldots, T$. The system state is characterized by the orders held in the pool, their associated multiorder probabilities, and their stay so far.

Objective. Recall that the ultimate goal of the control task is to maximize multiorder hits. Whether a hit will be eventually captured depends on how long the order stays in the pool and its associated multiorder probability. Because of this observation, we construct the objective function to be the total stay of the orders in the pool, weighted by their multiorder probabilities.

Constraints. The optimal solution will be trivial if without constraints—for example, just holding all the orders as long as possible, which is clearly not appealing in

practice. In our context, various concerns have been raised by the managers of TMS, and we translate them into the following constraints.

- Maximum flow constraint. The FC operations impose a limit on our outbound fulfillment order flow in terms of the maximum number of fulfillment orders released from the pool in each period t, denoted by $\overline{F}_t$. This is to minimize flow disturbance and to accommodate the existing order-processing capacity schedule. (Jointly optimizing the order-processing schedules is another interesting problem; this is left to the next stage of the project and is not within the scope of this paper.)
- Maximum stay constraint. For each order, there is a maximum amount of time in the pool, denoted by L (periods).
- Pool size constraint. There should be a cap on the total number of orders held in the pool at any time, denoted by C_t.
- End-of-horizon constraint. No orders should be carried over to the next day; the pool has to be emptied by the end of the day.

We shall incorporate these constraints in our model in terms of either constraints or dualized penalty terms in the objective function.

Mathematical Formulation and Solution Procedures

The problem is formulated as a dynamic program (DP). Considering all the complications and implementation requirements, we develop a way to simplify the state space and propose an approximation that solves the linear programming (LP) relaxation of the original DP on a rolling basis. Given the technical nature of this part, we relegate the details to the appendix.

Benchmark Policies

To better understand the impact of the prediction and control tasks in isolation, we evaluate two benchmark models.

We first consider a simple threshold policy that serves as a lower-bound benchmark. The idea is to fully utilize the output of the multiorder probability prediction but to replace the optimal control part with an easy-to-obtain and easy-to-implement heuristic rule. This way, we can conduct a comparison between the benchmark and our proposed solution and evaluate the marginal contribution of solving the optimal control problem. The heuristic rule works as follows; for each order received, if its multiorder probability is larger than a threshold $\overline{p}$, it goes into the order pool and waits for a fixed amount of time $\overline{\tau}$ (unless $t+\overline{\tau} > T$). On one hand, the heuristic is a generalization of the known practice we observed; with $\overline{p} = 0$ and $\overline{\tau}$ set to the length of the grace period, all orders will be universally held for the same amount of time. On the other hand, it can be considered a lower bound of our proposed policy because it only focuses on the multiorder probability and ignores the arrival time of the orders and the flow constraints.

An upper-bound policy of our proposed solution is to solve the LP relaxation with the actual observation, as opposed to the real-time prediction, of order arrivals as input. This essentially gets rid of the impact of the arrival flow uncertainty and provides us an estimation of the performance in the ideal setting.

Performance Evaluation

The proposed solution has been deployed online at TMS, together with several other algorithm modules that detect malicious orders, potential returns, and potential arbitragers. It is hard to decouple our solution from the online system and evaluate its performance and benefit in isolation. In this section, we present a back test of the proposed solution together with the benchmark policies on the real data from TMS.

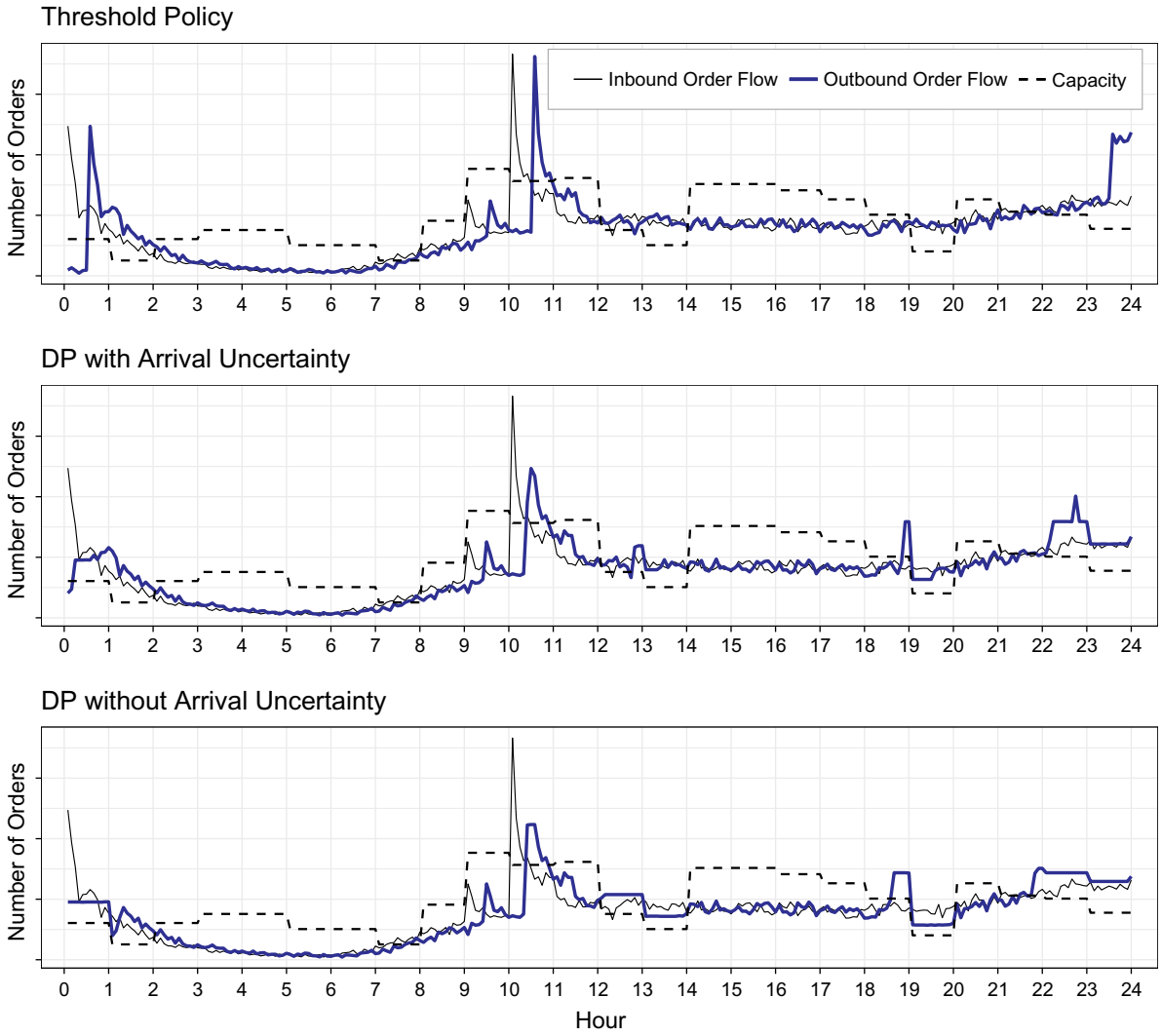
The following constraint parameters are specified by the managers of TMS: (1) maximum stay: 30 minutes ($L = 6$ periods), which is decided based on the pattern on Figure 3; (2) pool size constraint C_t: 20% of the total daily orders for all t (this constraint is never binding in our tests); and (3) maximum flow penalty: c_t^F is set to 10 for all $t = 1, \ldots, 272$ and is 100 for $t = 273, \ldots, 288$, which is to avoid depleting the pool entirely at the end. We are not allowed to disclose $\overline{F}_t$, which is the FC's processing capacity. In this study, we use a set of synthesized numbers just for the purpose of illustration. The numbers are plotted in Figure 5 as dashed lines.

Multiorder probabilities are grouped according to the following $K = 4$ intervals: $[0, 0.2), [0.2, 0.5), [0.5, 0.8), [0.8, 1]$, with the corresponding p_k being $0.1, 0.35, 0.65, 0.9$. In the threshold policy, $\overline{\tau} = 6$ and $\overline{p} = 0.15$, which is manually tuned with reference to the receiver operating characteristic curve of the multiorder prediction algorithm.

The key performance indicators (KPIs) specified by the management of TMS are (1) multiorder captures, measured by the percentage of multiorders captured by the algorithm over the total number of multiorders within the same time frame (i.e., 30 minutes in our test); and (2) average stay in pool, the average time spent in the pool. Table 3 lists the KPI we summarized for two days, day A being the typical day we study in the Data Exploration section and day B being another day with major promotional activities.

Not surprisingly, the upper-bound policy, DP without uncertainty, dominates our proposed solution in both KPIs; it has higher multiorder captures (95.5%) and lower average stay (19.4 minutes). The more interesting comparison is between the lower-bound threshold policy and our solution. We find that the threshold policy is better at capturing multiorders (96.7% versus

Figure 5. (Color online) The Outbound Order Flows Under the Three Policies Show Different Smoothness



92.8%) but at the cost of a much longer pool stay (8.6 minutes or 42% more on day A and 3.2 minutes or 14% more on day B). The analysis sheds light on the trade-off between the two KPIs and provides us with a guideline on how to relax the maximum stay constraint. If we are able to relax the maximum stay constraint from 30 minutes to, say, 60 minutes, we can evaluate multiorder captures and average stay in a similar manner.

Another shortcoming of the threshold policy is its ignorance of the flow constraints. Figure 5 illustrates the outbound order flow (thick blue lines) under the three policies described in the previous section together with

Table 3. A Summary of KPIs of the Three Policies for a Typical Day (Day A) and a Day with Major Promotion (Day B)

Day	Proportion of multiorders (within 30 minutes), %	KPI	Threshold policy	DP with uncertainty	DP without uncertainty
Day A (typical)	3.0	Multiorder captures	96.7%	92.8%	95.5%
Day A (typical)	3.0	Average stay in pool (minutes)	28.9	20.3	19.4
Day B (promotion)	6.9	Multiorder captures	93.5%	88.8%	89.2%
Day B (promotion)	6.9	Average stay in pool (minutes)	26.3	23.1	22.5

the inbound order flow (thin black lines), which is the same as in Figure 4, and the capacity $\overline{F}_t$ (dashed lines). We observe that the threshold policy (the top plot) tries to hold most of the orders (because of the low probability threshold that we set) and keep them for 30 minutes in the pool. Because the policy is myopic, a significant surge happens in the last 30 minutes of the day, which is because of releasing expiring orders in the pool and all new arrivals at the same time. The DP-based policies perform better in terms of smoothing the outbound flow, especially at the end. Although smoothing the flow is not our objective here, one can fine-tune the penalty c_t^F and achieve a smoother outbound flow. Our proposed solution (the middle plot), suffering from imperfect prediction of future order arrivals, has more spikes and fluctuations when compared with the policy without arrival uncertainty (the bottom plot).

Conclusion

Our analysis has demonstrated the power of analytics. Data analysis helps us identify the phenomenon of multiorders and provides an initial estimate of the potential benefit, machine learning leverages on the rich data about buyers' past behavior to produce accurate predictions of multiorder probabilities and order arrivals, and operations research tools convert this information into actions that significantly improve business performance without violating operational constraints.

Empowered by analytics, the online grocery retailer can capture 92.8% of multiorders and reduce its order fulfillment cost by 3%, which translates to more than 10 million dollars per year. A cost reduction by 3%, which may not seem significant in some business contexts, could determine the survival or bankruptcy of online grocery retailers, which often struggle to obtain a positive net profit margin. Moreover, all of this comes at a small cost; all the change that we bring to the existing order fulfillment process is resequencing the orders and an average delay of about 20 minutes.

In addition to the economic savings, there are also environmental benefits because of order fulfillment consolidation. For example, having the items packed in one carton box instead of two separate ones helps reduce the usage of packing materials. Also, to some extent, it reduces the shipping weight and volume, which in turn, makes shipping more efficient. A detailed evaluation of such carbon emission reduction is conducted in another project.

Appendix. Mathematical Formulation and Solution Procedures

A.1. Sequence of Events

At the beginning of period t, the existing pool of orders is reviewed. Let $\mathcal{S}_t$ denote the initial set of orders in the pool, where each order is identified by its arrival time and multiorder probability. We can partition the pool set according to the orders' arrival time; that is, $\mathcal{S}_t = \mathcal{S}_{t,t-L} + \cdots + \mathcal{S}_{t,t-1}$, where $\mathcal{S}_{t,s}$, $s = t-L,\ldots,t-1$ is the set of orders that arrived in period s and are still in the pool at the beginning of t. Note that because the maximum stay in the pool is L, the oldest set of orders includes those that arrived in $t-L$. We further partition each set $\mathcal{S}_{t,s}$ into K groups based on the orders' multiorder probabilities. The partition is done according to a set of predefined probability intervals. The result is K groups of orders; that is, $\mathcal{S}_{t,s} = \mathcal{S}_{t,s}^1 + \cdots + \mathcal{S}_{t,s}^K$, with the superscript denoting the probability group. The multiorder probability of an order, or the probability group an order is in, is assumed to be constant because of the exponential assumption justified in the Data Exploration section.

For tractability, we do not further differentiate orders in the same group $\mathcal{S}_{t,s}^k$ and treat them all with the same multiorder probability, the middle value of the corresponding probability interval denoted by p^k, where $0 < p^1 < \cdots < p^K < 1$. This way, the orders can be identified only by their arrival time and probability group. Letting $S_{t,s}^k$ be the number of orders in set $\mathcal{S}_{t,s}^k$, we have matrix $\mathbf{S}_t = [S_{t,s}^k]$ fully characterize the pool state.

Next, a new set of orders $\mathcal{I}_t$ arrives at the system. Multiorder hits are identified and released immediately. Let $\epsilon_{t,s}^k$ be the number of orders in set $\mathcal{S}_{t,s}^k$ that are hit by multiorders. The updated pool state is

$$S_{t,s}^{\prime k} = S_{t,s}^k - \epsilon_{t,s}^k, \text{ for } s = t-L,\ldots,t-1 \tag{A.1}$$

and for all $k = 1,\ldots,K$. On the other hand, those new orders that remain, denoted by $\mathcal{I}_t'$, have their multiorder probability estimated by the prediction algorithm developed in the Multiorder Prediction section. The same scheme of partition by probability can be applied to $\mathcal{I}_t'$—that is, $\mathcal{I}_t' = \mathcal{I}_t'^1 + \cdots + \mathcal{I}_t'^K$. Let $I_t'^k$ be the number of orders in $\mathcal{I}_t'^k$. These orders are then merged into the pool. This implies that

$$S_{t,t}^{\prime k} = I_t'^k. \tag{A.2}$$

In the following formulation, we refer to $\mathbf{S}_t$ as the initial state before multiorder clearance and $\mathbf{S}_t' = [S_{t,s}^{\prime k}]$, for $s = t-L,\ldots,t$ and $k = 1,\ldots,K$, as the updated state after clearance.

Given the updated state $\mathbf{S}_t'$ in period t, the decision is how many and which orders from the pool should be released. Let $\mathbf{O}_t = [O_{t,s}^k]$, for $s = t-L,\ldots,t$ and $k = 1,\ldots,K$, denote the decision variables. It is a matrix of dimension $(L+1)\times K$, with $O_{t,s}^k$ being the number of orders that arrived in group k at time s and are released at t.

The challenge that lies in the optimization is twofold. The first is about how many orders to release in each period so that the constraints are not violated. The second is about how to prioritize orders when releasing. For example, consider an order with a high multiorder probability that has already stayed for $L-1$ periods in the pool versus a newly arrived order with a lower probability. The answer is not obvious and may depend on the whole system state.

After releasing the orders, the pool state that will be carried to the next period $t+1$ can be updated:

$$S_{t+1,s}^k = S_{t,s}^{\prime k} - O_{t,s}^k, \text{ for } s = t-L+1,\ldots,t. \tag{A.3}$$

A.2. Dynamic Program Formulation

To this end, we can write down the dynamic program. Let $V_t(\mathbf{S}'_t)$ be the value function with a given updated state $\mathbf{S}'_t$; then, the DP recursion is

$$V_t(\mathbf{S}'_t) = \max_{\mathbf{O}_t} \left\{ \sum_{s=t-L+1}^{t} \sum_{k=1}^{K} p^k S^k_{t+1,s} - c^F_t \left(\sum_{s=t-L}^{t} \sum_{k=1}^{K} O^k_{t,s} - \overline{F}_t \right) + \mathbf{E}[V_{t+1}(S'_{t+1})] \right\}, \tag{A.4}$$

$$V_{T+1}(\cdot) = 0. \tag{A.5}$$

On the right-hand side of Equation (A.4), the first term is the incremental benefit of keeping orders in the pool, measured by the total number of orders carried to $t+1$, weighted by their corresponding multiorder probabilities. The second term is the penalty for violating the max flow constraint, with c^F_t being the per-unit penalty in period t. Here, we choose to incorporate the max flow constraint in the objective in order to guarantee a feasible solution. The third term is the future value-to-go from period $t+1$ onward. Equation (A.5) is the boundary condition.

In addition to the state dynamics (A.1), (A.2), and (A.3), the aforementioned recursion is also subject to the following constraints.

Constraint 1 (maximum stay). All existing L-period-old orders must be released from the pool (for $k = 1, \ldots, K$):

$$O^k_{t,t-L} = S'^k_{t,t-L}.$$

Constraint 2 (pool size). The total orders in the pool at the end of t cannot be more than C_t:

$$\sum_{s=t-L+1}^{t} \sum_{k=1}^{K} S^k_{t+1,s} \le C_t.$$

Constraint 3 (end of horizon). The pool must be empty at the end of the horizon:

$$\mathbf{S}_{T+1} = 0.$$

Constraint 4 (nonnegativity):

$$\mathbf{S}_t \ge 0, \ \mathbf{O}_t \ge 0.$$

A.3. LP Relaxation

In practice, the aforementioned dynamic program is not tractable because of high dimensionality. We shall employ LP-based approximation on a rolling basis (Bertsekas 2012). The main idea is that in any period t, given the updated state $\mathbf{S}'_t$, we treat all the random variables $\mathbf{I}'_{t+1}, \ldots, \mathbf{I}'_T$ and $\epsilon_{t+1}, \ldots, \epsilon_T$ as deterministic and replace them with our predictions. The resulting problem is deterministic and can be solved efficiently by LP. The optimal solution obtained from the LP covers the decisions for all the remaining periods $t, \ldots, T$. Yet, we only implement $\mathbf{O}_t$ for period t. The same process is repeated in the next period after updating the system state and future order predictions.

The LP we need to solve in period t can be written as

$$\max_{\mathbf{O}_t, \ldots, \mathbf{O}_T} \sum_{\tau=t}^{T} \left\{ \sum_{s=\tau-L+1}^{\tau} \sum_{k=1}^{K} p^k S^k_{\tau+1,s} - c^F_\tau \left(\sum_{s=\tau-L}^{\tau} \sum_{k=1}^{K} O^k_{\tau,s} - \overline{F}_\tau \right) \right\}, \tag{A.6}$$

subject to the following.

Constraint 0 (state dynamics). For all $\tau = t, \ldots, T$ and $k = 1, \ldots, K$,

$$S'^k_{\tau,s} = S^k_{\tau,s} - \epsilon^k_{\tau,s}, \text{ for } s = \tau - L, \ldots, \tau - 1,$$

$$S'^k_{\tau,\tau} = I'^k_\tau,$$

$$S^k_{\tau+1,s} = S'^k_{\tau,s} - O^k_{\tau,s}, \text{ for } s = \tau - L + 1, \ldots, \tau.$$

Constraint 1 (maximum stay). For all $\tau = t, \ldots, T$ and $k = 1, \ldots, K$,

$$O^k_{\tau,\tau-L} = S'^k_{\tau,\tau-L}.$$

Constraint 2 (pool size). For all $\tau = t, \ldots, T$,

$$\sum_{s=\tau-L+1}^{\tau} \sum_{k=1}^{K} S^k_{\tau+1,s} \le C_\tau.$$

Constraint 3 (end of horizon).

$$\mathbf{S}_{T+1} = 0.$$

Constraint 4 (nonnegativity). For all $\tau = t, \ldots, T$,

$$\mathbf{S}_\tau \ge 0, \ \mathbf{O}_\tau \ge 0.$$

In order to solve the aforementioned LP, we need input on $\mathbf{I}'_{t+1}, \ldots, \mathbf{I}'_T$ and $\epsilon_{t+1}, \ldots, \epsilon_T$. In the section on Order Flow Prediction, we have introduced a real-time prediction algorithm. The problem is that the granularity of the prediction is not enough. From the algorithm, we can only obtain the total number of orders in each period τ, namely $|\mathcal{I}_\tau|$. To split the total number, we estimate the historical percentage of orders that are allocated to each probability group k and apply that percentage to $|\mathcal{I}_\tau|$ to obtain $\mathbf{I}_\tau = [I^1_\tau, \ldots, I^K_\tau]$. Another simplification is to ignore ϵ_τ's and treat them as zero. This is because the number of multiorders constitutes only several percent of the total orders, which is negligible when compared with the prediction error. This way, we can approximate $\mathbf{I}'_\tau$ with $\mathbf{I}_\tau$.

References

Amazon.com (2019) About combined shipments. Accessed December 1, 2019, https://www.amazon.com/gp/help/customer/display.html?nodeId=201910270.

Bertsekas DP (2012) *Dynamic Programming and Optimal Control*, vol. II, 4th ed. (Athena Scientific, Nashua, NH).

Chen T, Guestrin C (2016) XGBoost: A scalable tree boosting system. *KDD '16 Proc. 22nd ACM SIGKDD Internat. Conf. Knowledge Discovery Data Mining*, 785–794.

Devlin J, Chang M-W, Lee K, Toutanova K (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. *Proc. 2019 Conf. North Amer. Chapter Assoc. Comput. Linguistics Human Language Tech. Vol. 1 (Long Short Papers)* (Association for Computational Linguistics, Kerrville, TX), 4171–4186.

Fernie J, Sparks L (2009) *Logistics & Retail Management: Emerging Issues and New Challenges in the Retail Supply Chain*, 3rd ed. (Kogan Page, London).

Griffin M (2018) Categories shifting online. Accessed September 1, 2022, https://www.npd.com/news/thought-leadership/2018/top-categories-shifting-online/.

He L, Rong Y, Shen Z-JM (2019) Product sourcing and distribution strategies under supply disruption and recall risks. *Production Oper. Management* 29(1):9–23.

Lim MK, Mak H-Y, Shen Z-JM (2017) Proximity and agility considerations in supply chain design. *Management Sci.* 63(4):1026–1041.

Medium.com (2018) Going the extra mile in customer service: How does JD.com make order cancellation so easy? Accessed

September 1, 2022, https://medium.com/jd-technology-blog/going-the-extra-mile-in-customer-service-how-does-jd-com-make-order-cancellation-so-easy-a00ebc7fd1ee.

Saunders N (2018) Online grocery & food shopping statistics. Accessed September 1, 2022, https://www.onespace.com/blog/2018/08/online-grocery-food-shopping-statistics/.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. *Proc. 31st Internat. Conf. Neural Inform. Processing Systems (NIPS'17)* (ACM, New York), 6000–6010.

Xu PJ, Allgor R, Graves SC (2009) Benefits of reevaluating real-time order fulfillment decisions. *Manufacturing Service Oper. Management* 11(2):340–355.

Verification Letter

Tao Fang, Director of Supply Chain Operations, TMall Supermarket, Alibaba Group, No. 969 West Wenyi Road, Yuhang, District, Hangzhou, China, writes:

"I am writing to testify to the project described in the paper titled 'Data-Driven Order Fulfillment Consolidation for Online Grocery Retailing,' which is being submitted to *INFORMS Journal on Applied Analytics* for review.

"The project started in 2018 at TMall Supermarket (codenamed TMS in the paper), which is an online grocery retailing business of the Alibaba Group. TMall Supermarket serves customers all around mainland China, processes more than a million orders per day, and has an annual revenue of tens of billions RMB. I was in charge of supply chain operations at TMall Supermarket and engaged the team of authors from Alibaba DChain, a business unit that specializes in providing digital solutions for supply chain management, to optimize our order fulfillment process to further cut cost.

"The project focused on order fulfillment consolidation. Through explorative data analysis, they found promising opportunities in consolidating orders so as to save the shipping cost paid to our logistics service provider, which charged us on a per parcel basis. We started from simple manually defined rules; then, the solution evolved into a much more complicated but powerful form with machine learning and dynamic optimization. To date, the system is still running on our order fulfillment system.

"The benefit of the project is straightforward and significant. In most other initiatives we have pursued to improve our supply chain operations, the monetary benefit is often hard to evaluate or prove in a counterfactual way. This project has an easily verifiable benefit: One can calculate how many orders could have been consolidated and how many were. In addition, for each consolidation, we have a clear number on the shipping cost saved. I cannot disclose this specific number, but in total, our estimated savings is around 100 million RMB per year. This was evaluated in 2019. As our business continues to grow, we expect the savings to grow proportionally."

Yang Wang is a data scientist at IDG Capital. This work was done when he was a senior algorithm engineer on the Supply Chain Fulfillment Team of Alibaba Group. His research interests include the combination of operations research, stochastic optimization, and large-scale online optimization in the areas of logistics and retail industry. He obtained his PhD in electrical engineering from Tsinghua University.

Tong Wang is a director at Alibaba Group leading the Supply Chain Forecasting Team. Prior to Alibaba, he was an associate professor at the National University of Singapore. His research is on information and flexibility in supply chain management, with a special focus on retail analytics. He obtained his PhD in decision sciences from INSEAD, France.

Xiaoqing Wang is a director at Alibaba Group leading the Supply Chain Fulfillment and Service Optimization Team. His research interests include the combination of operation research, stochastic optimization, and large-scale online optimization in the areas of logistics and retail industry. He obtained his PhD in systems engineering from Northeastern University.

Yuming Deng is a director at Alibaba Group leading the Digital Supply Chain Department. His research interests are assortment planning, network planning, pricing strategy, forecasting, inventory optimization, and simulation-based optimization and its applications. He obtained his PhD in operations research and industrial engineering from the University of Texas at Austin.

Lei Cao is a senior algorithm engineer on the Supply Planning Team of Alibaba Group. His research focuses on inventory management and revenue management in the retail industry with deep learning, reinforcement learning, and data-driven optimization algorithms. He obtained his PhD from the University of the Chinese Academy of Sciences.