Deep Learning-Driven Contextual Stochastic Optimization for Real-Time Order Fulfillment

Tinghan Ye[*]
ISyE, Georgia Tech

Shuaicheng Tong
ISyE, Georgia Tech

Changkun Guan
ISyE, Georgia Tech

Beste Basciftci
University of Iowa

Pascal Van Hentenryck
ISyE, Georgia Tech

Abstract

Order fulfillment optimization is a fundamental challenge in large-scale e-commerce, requiring real-time decisions for every incoming order. For enterprises with extensive fulfillment networks, selecting the optimal fulfillment plan demands balancing operational costs with strict service-level guarantees under uncertainty. To model this problem, this work introduces a two-stage contextual stochastic optimization framework explicitly capturing two sources of uncertainty, delivery timeliness and future inventory consumption. To enable real-time deployment in peak hours, where traditional solvers are computationally prohibitive, an optimization proxy is developed, training deep neural networks to approximate solutions of the underlying stochastic program with high fidelity. Computational experiments on a large-scale JD.com transactional dataset demonstrate that the proposed approach achieves orders-of-magnitude speedups compared to a state-of-the-art commercial solver while preserving similar solution quality. The results establish a scalable paradigm for real-time stochastic optimization in e-commerce logistics, bridging rigorous optimization with deep learning to deliver industrial-scale efficiency.

1 Introduction

Modern e-commerce platforms face the complex logistical challenge of real-time order fulfillment [1]. Each incoming order requires an immediate decision on how to source items from a network of facilities to minimize costs while satisfying delivery promises under significant uncertainty. While traditional operations research methods have addressed this problem [2, 5, 4], they often address future demand uncertainty either implicitly through dual prices or explicitly using deterministic forecasts, while typically overlooking other stochastic elements like delivery timeliness. More advanced frameworks like Contextual Stochastic Optimization (CSO) are beginning to address these complexities by integrating context information to the prediction process, such as incorporating distributional forecasts for delivery time uncertainty in order fulfillment problem [17]. The application of CSO to online, sequential decision-making, however, remains a largely open research area [10].

Even with a suitable online formulation, a practical challenge emerges from the computational latency of these models, especially in high-throughput systems. Solving the underlying stochastic integer program for each order, even with sample average approximation, typically takes from seconds to minutes [17]. This decision time becomes a key consideration, particularly during peak commercial periods. At such peaks, order volumes in the JD.com dataset can exceed 100 orders every minute [12], a throughput that necessitates millisecond-level responses to prevent operational backlogs. This highlights a considerable gap between the capabilities of rigorous optimization models and the operational demands of large-scale e-commerce.

[*]Corresponding author: joe.ye@gatech.edu

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: ML×OR Workshop: Mathematical Foundations and Operational Integration of Machine Learning for Uncertainty-Aware Decision-Making.

This work bridges this gap by introducing a deep learning framework that serves as an *optimization proxy* for the online CSO problem. The core idea is to train a neural network to imitate the decisions of a high-quality (but relatively slow) contextual sample average approximate (C-SAA) solver. The proposed proxy learns a direct mapping from an order's context and predictive scenarios to a near-optimal fulfillment decision, replacing the solver with a single, fast forward pass and a lightweight repair step to ensure feasibility. This approach builds on the growing literature on optimization proxies [19, 9, 6, 13]. However, unlike prior work that assumes no stochasticity or relies on deterministic forecasts, the proxy handles multi-source scenario uncertainty in a sequential fulfillment setting.

This paper makes three-fold contributions. It first formulates the online order fulfillment as a two-stage contextual stochastic program that models uncertainty in both future demand and delivery time. Based on this formulation, it introduces a novel, scenario-embedded deep learning proxy designed to approximate the C-SAA solution and ensure feasible, real-time decisions via a repair mechanism. Finally, the framework's industrial-scale viability is demonstrated through empirical validation on a large-scale transactional dataset from JD.com, where the proxy achieves a 3,000 $\times$ average speedup over the C-SAA solver while maintaining a final solution cost within 1.38% of the oracle's.

2 Problem Formulation

This paper considers the problem of online order fulfillment over a planning horizon $\mathcal{T} = \{1, 2, \ldots, T\}$, where each epoch $t \in \mathcal{T}$ corresponds to the arrival of a customer order. The fulfillment network consists of a set of locations $\mathcal{J}$ and a set of products $\mathcal{I}$. Each order requests quantities $\mathbf{q}^t = (q_i^t)_{i\in\mathcal{I}}$. The current on-hand inventory levels are denoted by $\mathrm{Inv}_{i,j}^t$ for each product $i \in \mathcal{I}$ at each location $j \in \mathcal{J}$. The core task is to create a dynamic fulfillment plan for each arriving order, assigning products to locations to minimize total expected costs while respecting operational constraints (see App. B for a full nomenclature). For each order, a fulfillment decision $\mathbf{z}^t = (z_{i,j}^t)_{i\in\mathcal{I},j\in\mathcal{J}}$ must be made, where $z_{i,j}^t$ is the integer quantity of product i sourced from location j, and belongs to the feasible set $\mathcal{Z}^t = \{z_{i,j}^t \in \mathbb{Z}_+ : \sum_j z_{i,j}^t = q_i^t, \forall i;\ z_{i,j}^t \le \mathrm{Inv}_{i,j}^t, \forall i, j\}$.

The decision-making process is subject to two primary sources of uncertainty. The first is delivery timeliness. Let $\tilde{\mathbf{d}}^t = (\tilde{d}_j^t)_{j\in\mathcal{J}}$ be a random vector where $\tilde{d}_j^t$ represents the deviation in days from the promised delivery date if served from location j. A positive realization d_j^t indicates a late delivery, while a negative value indicates an early one. The realization of d_j^t for the chosen location(s) is observed only after the decision $\mathbf{z}^t$ is executed, analogous to a bandit feedback setting [17]. The second source of uncertainty is future demand, represented by a random vector $\tilde{\mathbf{D}}^t = (\tilde{D}_i^t)_{i\in\mathcal{I}}$ denoting the cumulative demand for each product from $t+1$ to T.

The per-epoch cost function $C(\mathbf{z}^t, \tilde{\mathbf{d}}^t)$ combines deterministic shipping costs with penalties for the stochastic delivery deviations. To encourage consolidation, a discount $\beta \in (0, 1)$ is applied to the shipping cost c_{ij}^{ship} at any location j that sources two or more units for the order. Asymmetric penalties, $\gamma^+ \gg \gamma^- > 0$, are applied for late ($\tilde{d}_j^{t+} = \max\{0, \tilde{d}_j^t\}$) and early ($\tilde{d}_j^{t-} = \max\{0, -\tilde{d}_j^t\}$) deliveries. The cost is given by: $C(\mathbf{z}^t, \tilde{\mathbf{d}}^t) = \sum_j \left[\left(\sum_i c_{ij}^{\mathrm{ship}} z_{i,j}^t \right) \left(1 - \beta\, \mathbf{1}\{\sum_i z_{i,j}^t \ge 2\}\right) + \left(\gamma^+ \tilde{d}_j^{t+} - \gamma^- \tilde{d}_j^{t-} \right) \sum_i z_{i,j}^t \right]$.

At the start of epoch t, a contextual vector $\mathbf{X}^t \in \mathcal{X}$ is observed, containing features about the order, customer, and network state. This information is predictive of the uncertainties in delivery times and future demands, enabling data-driven decision-making. After the fulfillment decision $\mathbf{z}^t$ is executed, the system transitions to the next epoch, $t+1$, where a new contextual vector $\mathbf{X}^{t+1}$ is observed.

Two-Stage Contextual Stochastic Model Determining the optimal fulfillment decision $\mathbf{z}^t$ at each epoch requires accounting for its impact on future periods. This sequential nature gives rise to a multi-stage stochastic program, which is typically intractable. Therefore, a simplification mechanism is employed: at epoch t, a two-stage contextual stochastic program is solved to determine the immediate fulfillment decision $\mathbf{z}^t$. This model uses a stage aggregation assumption, collapsing future epochs $(t+1, \ldots, T)$ into a single second stage to manage complexity [8]. In this formulation, the first-stage decision $\mathbf{z}^t$ is made before the realization of current-period and future uncertainties. The second-stage

recourse decision, $\mathbf{v}^t = (v^t_{i,j})_{i\in\mathcal{I},j\in\mathcal{J}}$, represents the aggregate fulfillment plan for future orders, where $v^t_{i,j}$ represents the future inventory allocated to product i from location j.

The problem at each epoch t is formulated as: $\min_{\mathbf{z}^t\in\mathcal{Z}^t} \mathbb{E}_{\tilde{\mathbf{d}}^t,\tilde{\mathbf{D}}^t\sim\mathbb{P}(\cdot\,|\,\mathbf{X}^t)}\Big[\, C\big(\mathbf{z}^t,\,\tilde{\mathbf{d}}^t\big) + \mathcal{Q}\big(\mathbf{z}^t,\tilde{\mathbf{D}}^t\big)\,\Big]$, where $\mathcal{Q}(\mathbf{z}^t,\tilde{\mathbf{D}}^t)$ is the second-stage cost function representing the expected future shipping cost: $\mathcal{Q}(\mathbf{z}^t,\tilde{\mathbf{D}}^t) = \min_{\mathbf{v}^t\geq 0}\ \{\sum_{i,j} c^{\text{ship}}_{i,j} v^t_{i,j}$ s.t. $\sum_j v^t_{i,j} = \tilde{D}^t_i, \forall i; z^t_{i,j} + v^t_{i,j} \leq \text{Inv}^t_{i,j}, \forall i,j\}$. The first-stage decision $\mathbf{z}^t$ affects the second stage by consuming inventory. The deterministic second-stage cost implicitly assumes that the decision $\mathbf{z}^t$ does not alter future cost uncertainties.

Contextual Sample Average Approximation (C-SAA) To solve the above stochastic program, the expectation over the distribution $\mathbb{P}(\cdot|\mathbf{X}^t)$ is approximated by a finite set of E scenarios, $\tilde{\Omega}^t = \{\omega_1,\ldots,\omega_E\}$, sampled based on the context vector $\mathbf{X}^t$. Each scenario $\omega \in \tilde{\Omega}^t$ consists of a realization of the delivery time deviations, $\mathbf{d}^t_\omega$, and the cumulative future demands, $D^t_{i,\omega}$. Define $\boldsymbol{v}^t_\omega = (v^t_{i,j,\omega})_{i,j,\omega}$ as the extensive second-stage allocation decisions under scenario ω. This approach yields the C-SAA formulation [14, 17], a large-scale deterministic mixed-integer program given by the following extensive form: $\min_{\mathbf{z}^t\in\mathcal{Z}^t,\mathbf{v}^t_\omega\geq 0}\big\{\sum_\omega \Big(C(\mathbf{z}^t,\mathbf{d}^t_\omega) + \sum_{i,j} c^{\text{ship}}_{i,j} v^t_{i,j,\omega}\Big)$ s.t. $\sum_j v^t_{i,j,\omega} = D^t_{i,\omega}, \forall i,\omega; z^t_{i,j} + v^t_{i,j,\omega} \leq \text{Inv}^t_{i,j}, \forall i,j,\omega\big\}$. App. D summarizes the steps for C-SAA.

3 A Scenario-Embedded Proxy for the Two-Stage Program

Solving the two-stage contextual stochastic program via SAA can be computationally demanding for real-time fulfillment systems during peak periods. This section details a practical surrogate pipeline that delivers low-latency fulfillment decisions (see schematic overview in App. A). The pipeline consists of three stages: (i) generate predictive scenarios, (ii) use a learned optimization proxy to predict a first-stage allocation, and (iii) apply a lightweight repair procedure to enforce feasibility. The details of the model features and neural network architectures are available in Apps. F and H.

Scenario Generation For future demand, a A Multi-Horizon Quantile Recurrent Forecaster (MQRNN) model [15] predicts per-period quantiles over the lookahead horizon. These are then summed and interpolated to produce a family of demand scenario sets, $\boldsymbol{\Omega}^t_{\text{dem}} = (\tilde{\Omega}^t_{i,\text{dem}})_{i\in\mathcal{I}}$, where each set $\tilde{\Omega}^t_{i,\text{dem}} = \{\hat{D}^t_{i,1},\ldots,\hat{D}^t_{i,E}\}$ contains the E possible cumulative demand outcomes for product i. This summation implements the stage aggregation assumption, converting per-period forecasts into the cumulative demand required by the second stage. Similarly, for delivery time, a multi-quantile network generates a corresponding family of time deviation scenario sets, $\boldsymbol{\Omega}^t_{\text{time}} = (\tilde{\Omega}^t_{j,\text{time}})_{j\in\mathcal{J}}$, where each set $\tilde{\Omega}^t_{j,\text{time}} = \{\hat{d}^t_{j,1},\ldots,\hat{d}^t_{j,E}\}$ contains the E possible outcomes for location j.

The Optimization Proxy The proxy framework treats each product line within an order as an independent decision. This simplifies the task of predicting an integer allocation vector $\mathbf{z}^t_i$ into a multiclass classification problem: for each product, the proxy predicts the single most suitable location j^* from which to source its entire quantity q^t_i. This product-level decomposition allows a single model to handle orders of varying sizes. The proxy, a deep neural network f_θ, is trained to minimize an empirical risk objective over a dataset $\mathcal{N}$ of historical product-order instances (see App. G for details on the training data collection process): $\theta^* = \arg\min_\theta \frac{1}{|\mathcal{N}|}\sum_{n\in\mathcal{N}} \mathcal{L}\big(f_\theta(\mathbf{X}^n), \mathbf{z}^{n,*}\big)$, where $\mathbf{X}^n$ contains the contextual features for instance n, $\mathbf{z}^{n,*}$ is the corresponding target decision vector from the C-SAA solver, and $\mathcal{L}$ is the per-instance loss function detailed below. For notational clarity, the product index i is omitted in the subsequent descriptions of the architecture and loss, as they apply to a single product instance.

The network architecture uses three parallel MLP encoders to process distinct inputs: (i) standard contextual features $\mathbf{X}^n_{\text{std}}$, (ii) the current inventory vector $\mathbf{Inv}^t$, and (iii) the high-dimensional scenarios $\boldsymbol{\Omega}^n_{\text{dem}}$ and $\boldsymbol{\Omega}^n_{\text{time}}$. The scenario encoder [7] uses a two-branch design to produce a fixed-size embedding of future demand and costs. The concatenated embeddings from all three branches are fed to an output head that produces logits $\boldsymbol{\ell} \in \mathbb{R}^J$ over the fulfillment locations.

The per-instance loss $\mathcal{L}$ is a hybrid function, $\mathcal{L} = \mathcal{L}_{\text{clf}} + \lambda_{\text{cons}}\mathcal{L}_{\text{cons}}$. The primary component is the cross-entropy classification loss, $\mathcal{L}_{\text{clf}}$, which trains the model to predict the location j that

received the largest allocation from the C-SAA solver for instance n. To incorporate operational constraints into training, a constraint violation term, $\mathcal{L}_{\text{cons}}$, is added. This term uses the Gumbel-Softmax estimator to create a differentiable approximation of the argmax operation on the logits: $\tilde{a}_j = \text{softmax}\left(\frac{\ell_j + g_j}{\tau}\right)$, where $g_j \sim \text{Gumbel}(0, 1)$. This soft, one-hot assignment $\tilde{\mathbf{a}} = (\tilde{a}_j)_{j \in \mathcal{J}}$ is then used to compute the expected inventory shortfall, forming the constraint loss: $\mathcal{L}_{\text{cons}} = \max(0, q^n - \sum_j \min\{q^n\, \tilde{a}_j, \text{Inv}_j^t\})$.

Inference with Repair Procedure At inference, the proxy's output logits $\boldsymbol{\ell}$ are converted to probabilities $\mathbf{p}$ via the softmax function, and the location with the highest probability, $j^* = \arg\max_j p_j$, is selected. A deterministic, greedy repair procedure then ensures feasibility. It first attempts to assign the full quantity q^n to location j^*, clipping the amount by the available inventory $\text{Inv}_{j^*}^t$. If any quantity remains unfulfilled, it is greedily assigned to the other locations in descending order of their softmax probability p_j until the demand is fully met.

4 Computational Study

The framework is evaluated in a simulated online environment using a large-scale e-commerce dataset from JD.com [12] (see App. E for data preprocessing details). Both training and evaluation are restricted to decision epochs in the daily peak window (06:00–18:00). Instance setup and per-unit shipping-cost construction are detailed in App. C. The first three weeks of data are used for training and the final week for evaluation, where a counterfactual simulator generates realistic delivery-time outcomes for unobserved decisions [17]. Performance is benchmarked against the C-SAA model, which serves as a high-quality oracle. To ensure statistical robustness, all results are averaged over 50 independent simulation runs. The primary metrics are cumulative objective value (cost) and decision latency (average and maximum run time). Full implementation details are provided in App. I.

Results and Analysis Table 1 summarizes the performance of C-SAA and the proposed optimization proxy. The results reveal a favorable trade-off between solution quality and computational speed. The proxy achieves near-optimal performance, incurring a minimal 1.38% increase in total cost compared to C-SAA. Interestingly, this small cost increase is coupled with a slight improvement in a key operational metric, as the proxy reduces the average late delivery rate. This suggests the proxy learns a robust and effective policy that, while not perfectly optimized for the full cost function on every instance, is consistent at the crucial task of avoiding late deliveries.

This near-optimal performance is achieved with a massive reduction in computational latency. The proxy reduces the average decision time from 6.063 seconds to just 2 milliseconds, a speedup of over 3,000 times. The proxy reduces the maximum run time from 88.56 seconds to a manageable 0.699 seconds, crucial for real-time peak-hour operations. These results confirm that for a negligible optimality gap, the optimization proxy successfully bridges the gap between high-fidelity stochastic optimization and the stringent low-latency requirements of large-scale industrial applications.

Table 1: Cumulative Realized Metrics (Avg $\pm$ SE) across 50 Simulation Replications and Run Time.

Model	**C-SAA**	**Proxy**
Avg Objective Value (↓)	93,983 $\pm$ 99	95,284 $\pm$ 91
Avg Late Delivery Rate (%) (↓)	9.59 $\pm$ 0.02	9.56 $\pm$ 0.02
Avg Run Time (s) (↓)	6.063	0.002
Max Run Time (s) (↓)	88.56	0.699

5 Conclusion and Future Work

This paper introduced a deep learning proxy for online stochastic order fulfillment, demonstrating its effectiveness in reducing decision latency by over three orders of magnitude compared to a C-SAA solver while maintaining favorable solution quality in peak hours. This work confirms the industrial viability of using proxies to deploy complex stochastic optimization models in real time. Promising future directions include integrating reinforcement learning [19, 3] and using conformal prediction to provide statistical guarantees for more complex fulfillment problems under uncertainty [18].

References

[1] Acimovic, J. and Farias, V. F. (2019). The fulfillment-optimization problem. In *Operations Research & Management Science in the age of analytics*, pages 218–237. INFORMS.

[2] Acimovic, J. and Graves, S. C. (2015). Making better fulfillment decisions on the fly in an online retail environment. *Manufacturing & Service Operations Management*, 17(1):34–51.

[3] Akhlaghi, V. E., Zandehshahvar, R., and Van Hentenryck, P. (2025). Propel: Supervised and reinforcement learning for large-scale supply chain planning. *arXiv preprint arXiv:2504.07383*.

[4] Andrews, J. M., Farias, V. F., Khojandi, A. I., and Yan, C. M. (2019). Primal-dual algorithms for order fulfillment at Urban Outfitters, Inc. *INFORMS Journal on Applied Analytics*, 49(5):355–370.

[5] Jasin, S. and Sinha, A. (2015). An LP-based correlated rounding scheme for multi-item ecommerce order fulfillment. *Operations Research*, 63(6):1336–1351.

[6] Ojha, R., Chen, W., Zhang, H., Khir, R., Erera, A., and Van Hentenryck, P. (2025). Outbound load planning in parcel delivery service networks using machine learning and optimization. *Transportation Science*.

[7] Patel, R. M., Dumouchelle, J., Khalil, E., and Bodur, M. (2022). Neur2SP: Neural two-stage stochastic programming. *Advances in neural information processing systems*, 35:23992–24005.

[8] Powell, W. B. (2022). Designing lookahead policies for sequential decision problems in transportation and logistics. *IEEE Open Journal of Intelligent Transportation Systems*, 3:313–327.

[9] Qi, M., Shi, Y., Qi, Y., Ma, C., Yuan, R., Wu, D., and Shen, Z.-J. (2023). A practical end-to-end inventory management model with deep learning. *Management Science*, 69(2):759–773.

[10] Sadana, U., Chenreddy, A., Delage, E., Forel, A., Frejinger, E., and Vidal, T. (2025). A survey of contextual optimization methods for decision-making under uncertainty. *European Journal of Operational Research*, 320(2):271–289.

[11] Salari, N., Liu, S., and Shen, Z.-J. M. (2022). Real-time delivery time forecasting and promising in online retailing: When will your package arrive? *Manufacturing & Service Operations Management*, 24(3):1421–1436.

[12] Shen, M., Tang, C. S., Wu, D., Yuan, R., and Zhou, W. (2024). JD. com: Transaction-level data for the 2020 MSOM data driven research challenge. *Manufacturing & Service Operations Management*, 26(1):2–10.

[13] Van Hentenryck, P. (2025). Optimization learning. *arXiv preprint arXiv:2501.03443*.

[14] Verweij, B., Ahmed, S., Kleywegt, A. J., Nemhauser, G., and Shapiro, A. (2003). The sample average approximation method applied to stochastic routing problems: a computational study. *Computational optimization and applications*, 24(2):289–333.

[15] Wen, R., Torkkola, K., Narayanaswamy, B., and Madeka, D. (2017). A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*.

[16] Yang, H., Huang, T., and Chen, K. (2025). Marrying theory and practice: Product network-inspired deep learning for sales forecasting. *Available at SSRN 5118746*.

[17] Ye, T., Cheng, S., Hijazi, A., and Van Hentenryck, P. (2025a). Contextual stochastic optimization for omnichannel multicourier order fulfillment under delivery time uncertainty. *Manufacturing & Service Operations Management*.

[18] Ye, T., Hijazi, A., and Van Hentenryck, P. (2025b). Conformal predictive distributions for order fulfillment time forecasting. *arXiv preprint arXiv:2505.17340*.

[19] Yuan, E., Chen, W., and Van Hentenryck, P. (2022). Reinforcement learning from optimization proxy for ride-hailing vehicle relocation. *Journal of Artificial Intelligence Research*, 75:985–1002.

A Schematic Overview

Figure 1 provides a schematic overview of the proposed approach. Offline, contextual signals drive forecasts and scenario generation, and a C-SAA oracle yields labels to train the proxy; online, the proxy consumes the same inputs and outputs a feasible fulfillment plan with low latency.

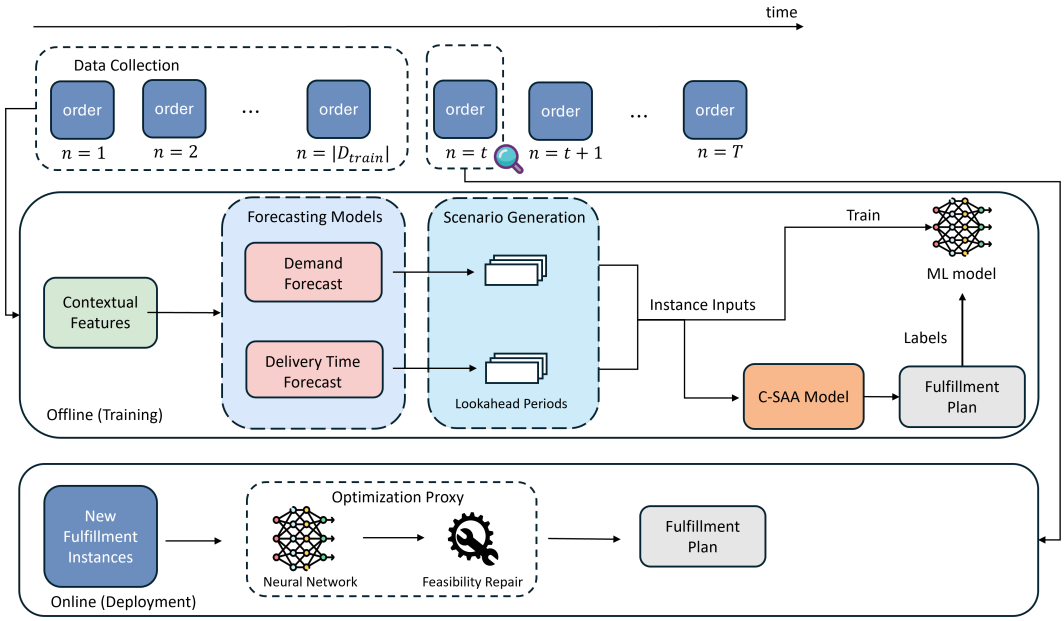

Figure 1: Schematic Overview of the Proposed Framework.

B Nomenclature

Conventions: Bold symbols are vectors; calligraphic symbols are sets; tildes denote random quantities; realizations are plain (no tilde).

Symbol	Meaning
Sets, indices, horizon	
$\mathcal{T} = \{1, \dots, T\}$	Planning horizon (epochs). t indexes epochs; T is the final epoch.
$\mathcal{I}$, i, $I{=}\lvert\mathcal{I}\rvert$	Set of products, its index, and cardinality.
$\mathcal{J}$, j, $J{=}\lvert\mathcal{J}\rvert$	Set of fulfillment locations, its index, and cardinality.
$\mathcal{X}$	Context/covariate space.
$\tilde{\Omega}^t$, $E{=}\lvert\tilde{\Omega}^t\rvert$	Scenario set at epoch t and its size.
$\tilde{\Omega}^t_{i,\mathrm{dem}}$, $\tilde{\Omega}^t_{j,\mathrm{time}}$	Per-product demand scenarios; per-location delivery-time scenarios at epoch t.
$\mathcal{N}$, n	Training set of product-level instances and its index.
Order, state, inventory	
$\mathbf{q}^t = (q^t_i)_{i\in\mathcal{I}}$	Requested units of the order arriving at epoch t (units).
$\mathrm{Inv}^t_{i,j}$	On-hand inventory of product i at location j at epoch t (units).
$\mathbf{Inv}^t = (\mathrm{Inv}^t_j)_{j\in\mathcal{J}}$	Location-level inventory vector used by the proxy (units).
$\mathbf{X}^t$	Context/features observed at the start of epoch t.
Decisions	
$\mathbf{z}^t = (z^t_{i,j})$	First-stage fulfillment decision for order t; $z^t_{i,j}$ units of i from j (units).
$\mathcal{Z}^t$	Feasible set at epoch t: $\{z^t_{i,j}\in\mathbb{Z}_+ : \sum_j z^t_{i,j}{=}q^t_i,\ z^t_{i,j}\le\mathrm{Inv}^t_{i,j}\}$.
$\mathbf{v}^t = (v^t_{i,j})$	Second-stage (aggregated future) fulfillment decision (units).
$\mathbf{v}^t_\omega = (v^t_{i,j,\omega})$	Second-stage decision in scenario ω (extensive form, units).
j^*	Location chosen by the proxy at inference, $j^*{=}\arg\max_j p_j$.
Uncertain parameters	
$\tilde{\mathbf{d}}^t = (\tilde{d}^t_j)_{j\in\mathcal{J}}$	Delivery-time deviation if served from each location (days).
d^t_j	Realized deviation for location j; $d^t_j{>}0$ late, $d^t_j{<}0$ early (days).

$d_j^{t+} := \max\{0, d_j^t\}$, $d_j^{t-} := \max\{0, -d_j^t\}$	Positive/negative parts of d_j^t (days).
$\tilde{\mathbf{D}}^t = (\tilde{D}_i^t)_{i \in \mathcal{I}}$	Cumulative future demand from $t{+}1$ to T (units).
$\mathbf{d}_\omega^t$, $D_{i,\omega}^t$	Scenario-ω delivery-time deviations and future demand realizations.
Costs and objectives	
c_{ij}^{ship}	Deterministic per-unit shipping cost for product i from location j.
$\beta \in (0,1)$	Consolidation discount if $\sum_i z_{i,j}^t \geq 2$ at location j. Default value is 0.5.
$\gamma^+ \gg \gamma^- > 0$	Asymmetric penalties for late (γ^+) and early (γ^-) delivery (per unit-day). Default values are 4.0 and 0.5.
$C(\mathbf{z}^t, \tilde{\mathbf{d}}^t)$	Per-epoch cost: shipping plus deviation penalties.
$\mathcal{Q}(\mathbf{z}^t, \tilde{\mathbf{D}}^t)$	Second-stage (future) shipping cost given $\mathbf{z}^t$.
$p(\omega)$	Probability of scenario ω in C-SAA.
Distributions and operators	
$\mathbb{P}(\cdot \mid \mathbf{X}^t)$	Conditional law of uncertainties given context $\mathbf{X}^t$.
$\mathbb{E}[\cdot]$	Expectation. $\lvert\cdot\rvert$: cardinality. $\mathbb{Z}_+$: nonnegative integers $\{0, 1, 2, \dots\}$.
$\mathbf{1}\{\cdot\}$	Indicator function (1 if condition holds; 0 otherwise).
Scenario generation	
$\mathbf{\Omega}_{\text{dem}}^t = (\tilde{\Omega}_{i,\text{dem}}^t)_{i \in \mathcal{I}}$	Family of demand-scenario sets at epoch t.
$\tilde{\Omega}_{i,\text{dem}}^t = \{\hat{D}_{i,1}^t, \dots, \hat{D}_{i,E}^t\}$	E cumulative-demand outcomes for product i at epoch t.
$\mathbf{\Omega}_{\text{time}}^t = (\tilde{\Omega}_{j,\text{time}}^t)_{j \in \mathcal{J}}$	Family of delivery-time scenario sets at epoch t.
$\tilde{\Omega}_{j,\text{time}}^t = \{\hat{d}_{j,1}^t, \dots, \hat{d}_{j,E}^t\}$	E deviation outcomes for location j at epoch t.
Learning (proxy) notation	
f_θ, θ	Neural proxy mapping features to location logits; its trainable parameters.
$\boldsymbol{\ell} \in \mathbb{R}^J$, $\mathbf{p}$=softmax($\boldsymbol{\ell}$)	Output logits over locations; corresponding probabilities.
$\mathbf{z}^{n,*}$	Target decision (from C-SAA) for training instance n.
$\mathcal{L}$, $\mathcal{L}_{\text{clf}}$, $\mathcal{L}_{\text{cons}}$	Total, classification, and constraint-violation losses.
λ_{cons}	Weight on the constraint-violation loss.
$\tilde{a}_j$	Gumbel-Softmax relaxed one-hot assignment probability for location j.
$\tau > 0$	Temperature used in the Gumbel-Softmax relaxation.
$g_j \sim \text{Gumbel}(0,1)$	i.i.d. Gumbel noise for the relaxation.
q^n	Requested units for the product in training instance n (units).
Convenience	
$x^+ := \max\{0, x\}$, $x^- := \max\{0, -x\}$	Positive/negative parts of a scalar x (used for deviations).

C Fulfillment Problem Instance Setup

The instance is defined over a peak window $\mathcal{T}_{\text{peak}} = [t_{\text{start}}, t_{\text{end}}] \subseteq \mathcal{T}$ within a single operating day. Initial inventories are not observed; for each product i and distribution center (DC) j, the starting level is set to the maximum historical sales from that DC over a calibration window, $\text{Inv}_{i,j}^0 := \max_{\tau \in \mathcal{W}} \text{sales}_{i,j}(\tau)$. Central-warehouse stock is scaled and buffered with a fixed scale $\alpha = 10$ and per-item safety stock $s_i^{\text{saf}} = 100$, i.e., $\text{Inv}_{i,\text{CW}}^0 := 10\,\text{Inv}_{i,\text{CW,base}}^0 + 100$.

Per-unit shipping cost construction. Shipping costs are synthesized by region proximity and whether the *origin* DC is a central warehouse (CW). Let $r(j)$ be the region of DC j and $\mathcal{J}_{\text{cw}}$ the set of CWs (DCs whose IDs also appear as region IDs). Define a base per-unit shipping *rate* between an origin DC j and a destination DC (or destination region proxy) j' as $s_{j,j'} \in \{\kappa_1, \dots, \kappa_6\}$ according to the tiering below; invoice-level variability is modeled multiplicatively by

$$\hat{s}_{j,j'} \;=\; s_{j,j'}\,\varepsilon_{j,j'}, \qquad \varepsilon_{j,j'} \sim \text{LogNormal}(0, \sigma^2),\;\; \sigma = 0.05.$$

In the model cost, the notation c_{ij}^{ship} denotes the applicable per-unit shipping cost for a unit of product i sourced from location j for the current order; operationally, c_{ij}^{ship} is taken from $\hat{s}_{j,j'}$ given the order's destination j' (or destination region).

Table 3: Shipping-cost tiers and parameters used in the deterministic calculation.

Geographic Condition (Origin DC j, Dest. DC j')	**Tier symbol**	**Value**
Same DC (j=j'), origin not CW	κ_1	1.0
Same DC (j=j'), origin is CW	κ_2	1.6
Same region ($r(j)$=$r(j')$), origin not CW	κ_3	1.5
Same region ($r(j)$=$r(j')$), origin is CW	κ_4	2.2
Different region ($r(j) \neq r(j')$), origin not CW	κ_5	2.0
Different region ($r(j) \neq r(j')$), origin is CW	κ_6	3.0
Lognormal noise s.d.	σ	0.05

For completeness, the tier selection is

$$s_{j,j'} = \begin{cases} \kappa_1, & j = j',\ j \notin \mathcal{J}_{\text{cw}}, \\ \kappa_2, & j = j',\ j \in \mathcal{J}_{\text{cw}}, \\ \kappa_3, & j \neq j',\ r(j) = r(j'),\ j \notin \mathcal{J}_{\text{cw}}, \\ \kappa_4, & j \neq j',\ r(j) = r(j'),\ j \in \mathcal{J}_{\text{cw}}, \\ \kappa_5, & r(j) \neq r(j'),\ j \notin \mathcal{J}_{\text{cw}}, \\ \kappa_6, & r(j) \neq r(j'),\ j \in \mathcal{J}_{\text{cw}}. \end{cases}$$

D C-SAA Algorithm Pseudocode

Algorithm 1 provides the pseudocode for the complete C-SAA procedure.

Algorithm 1 Two-Stage C-SAA with Candidate Generation and Evaluation

Require: Number of candidates S; generation scenarios N_1; evaluation scenarios N_2
Ensure: Candidate set $\{\mathbf{z}_s^t\}_{s=1}^S$ with estimated objectives $\{\bar{f}_s\}_{s=1}^S$

procedure GENERATECANDIDATES(S, N_1)
  for $s = 1$ to S **do** ▷ Candidate generation
    $\Omega_{s,N_1} \leftarrow$ SAMPLESCENARIOS(N_1) ▷ Demand & fulfillment cost
    $(\mathbf{z}_s^t, \hat{f}_{s,N_1}) \leftarrow$ SOLVESAA(Ω_{s,N_1})
  end for
  return $\{\mathbf{z}_s^t\}_{s=1}^S$
end procedure

procedure EVALUATECANDIDATES($\{\mathbf{z}_s^t\}_{s=1}^S, N_2$)
  $\Omega_{N_2} \leftarrow$ SAMPLESCENARIOS(N_2) ▷ Fresh, larger set
  for $s = 1$ to S **do** ▷ Candidate evaluation
    for each $\omega \in \Omega_{N_2}$ **do** ▷ Scenario evaluation
      $\phi_{s,\omega} \leftarrow$ SOLVESECONDSTAGE($\mathbf{z}_s^t, \omega$) ▷ Solve: $\min \sum_{i,j} c_{i,j}^{ship} v_{i,j,\omega}$ s.t. $\sum_j v_{i,j,\omega} = \hat{D}_{i,\omega}\ \forall i$; $z_{s,i,j}^t + v_{i,j,\omega} \leq I_{i,j}^t\ \forall i,j$; $v_{i,j,\omega} \geq 0$
    end for
    $\bar{f}_s \leftarrow \dfrac{1}{N_2} \displaystyle\sum_{\omega \in \Omega_{N_2}} \left(\phi_{s,\omega} + C_\omega(\mathbf{z}_s^t, \mathbf{X}^t) \right)$ ▷ Statistical upper bound estimate
  end for
  return $\{\bar{f}_s\}_{s=1}^S$
end procedure

$\{\mathbf{z}_s^t\}_{s=1}^S \leftarrow$ GENERATECANDIDATES(S, N_1)
$\{\bar{f}_s\}_{s=1}^S \leftarrow$ EVALUATECANDIDATES($\{\mathbf{z}_s^t\}_{s=1}^S, N_2$)
$s^\star \leftarrow \arg\min_{s \in \{1,\ldots,S\}} \bar{f}_s$ ▷ Select best candidate by estimated objective
return $(\mathbf{z}_{s^\star}^t, \bar{f}_{s^\star})$

E Data Preprocessing

Entries with missing delivery promise days and duplicate (`order_ID, sku_ID`) pairs were removed. Records with missing `brand_ID`, single-item gift orders, and multi-package orders (more than one unique `package_ID`) were excluded. Timestamps were parsed to compute end-to-end delivery time (`order_time`→`arr_time`) and DC-to-DC travel time (`ship_out_time`→`arr_station_time`); records with negative durations were discarded. Delivery hours were converted to days via $\lceil \text{hours}/24 \rceil$ (counting same-day as 1 day), delivery deviation was defined as `delivery_time_days` minus `promise_delivery_days`, and records with `delivery_time_days` > 5 or deviations outside $[-5, 5]$ days were removed.

F Model Feature Details

F.1 Delivery-Time Forecasting Features

Feature design follows prior e-commerce delivery-time models [11], combining time-of-order signals, order complexity/price incentives, and an operations snapshot of the origin/destination network, with simple user/location covariates; data are time-sorted and split by calendar cutoffs.

F.2 Demand Forecasting Features

In line with recent retail practice [16], per-SKU series are resampled to a fixed bucket and augmented with compact calendar numerics, order/customer/product covariates, and SKU/brand identifiers; sliding windows provide history and prediction contexts for the encoder–decoder.

F.3 Optimization Proxy Features

Each row corresponds to one *(order, SKU)* decision context, organized into three blocks:

Standard per-row covariates. Configured order/DC covariates after categorical factorization describe the sourcing context (customer/location tiers, order size/complexity, promised speed, destination code). The ordered units for this SKU are recorded as a scalar quantity, and integer indices for SKU and brand are provided for embedding layers.

Scenario covariates. A fixed-length demand vector captures short-term uncertainty in required units, and a fixed-shape cost matrix (scenarios $\times$ DCs) represents per-unit fulfillment costs under the same scenarios. All rows share a consistent DC ordering; rows with unexpected scenario counts are excluded to keep tensor shapes uniform.

Inventory & base-cost summaries. An inventory vector enumerates on-hand units for this SKU across DCs at decision time, accompanied by simple summaries (total, mean, max, min) to indicate tightness and dispersion. Base shipping-cost summaries to the destination DC (mean, max, min across origin DCs) provide coarse cost signals independent of scenario variation.

G Training Data Collection Details

The pre-training period spans three consecutive weeks and is partitioned chronologically into two disjoint subsets: a *forecast-train* split $\mathcal{D}_{\text{fcast}}$ comprising the first two weeks and a *proxy-train* split $\mathcal{D}_{\text{proxy}}$ comprising the subsequent one week; the following week is reserved for evaluation. Scenario generators for demand and delivery-time deviations are fit exclusively on $\mathcal{D}_{\text{fcast}}$. Using the fitted generators, context-conditional scenarios are produced for each order in $\mathcal{D}_{\text{proxy}}$, and the C-SAA algorithm is solved with those scenarios and contemporaneous inventories to obtain a high-quality fulfillment decision $\mathbf{z}^{*,n}$ for each instance n. The proxy is then trained on inputs derived from $\mathcal{D}_{\text{proxy}}$ (context features, inventories, and generated scenarios) with $\mathbf{z}^{*,n}$—or its implied single-DC label used by the classifier—as supervision. No data from $\mathcal{D}_{\text{proxy}}$ or from the evaluation week is used when fitting the scenario generators, preventing leakage and hindsight bias while aligning training labels with the oracle policy the proxy is designed to imitate. For the final evaluation only, the scenario generators are refit on the full training set $\mathcal{D}_{\text{train}} = \mathcal{D}_{\text{fcast}} \cup \mathcal{D}_{\text{proxy}}$.

H Neural Network Details

H.1 Delivery Time Forecasting Model (MQMLP)

The multi-quantile MLP (MQMLP) delivery time model is a feed-forward MLP that predicts multiple delivery-time quantiles. Inputs consist of numerical features and origin/destination DC IDs; the DC IDs are mapped through learned embeddings and concatenated to the numerical features before entering the network. The output layer returns one value per requested quantile.

Non-crossing quantiles. To avoid quantile crossing, the head is built around the median (0.5) as an anchor. The model predicts (i) a median value and (ii) nonnegative "gaps" for higher and lower quantiles via a `softplus` transform. Higher quantiles are the median plus cumulative upward gaps; lower quantiles are the median minus cumulative downward gaps. This construction produces nondecreasing quantiles by design.

Bounded support. Predictions are mapped to a fixed interval with a monotone sigmoid: $[t_{\min}, t_{\max}] = [1, 5]$. Concretely,

$$\hat{q} \;=\; t_{\min} + (t_{\max} - t_{\min}) \cdot \sigma(\text{raw}).$$

Training objective. Training uses the pinball (quantile) loss summed over all predicted quantiles. Early stopping is applied on a validation split.

Hyperparameters

Setting	Value
Hidden width	192
MLP layers	4
Dropout	0.15
Batch size	128
Learning rate	1.7×10^{-3}
Weight decay	7.8×10^{-4}
DC embedding dims (ori, des)	(8, 8)

H.2 Demand Forecasting (MQRNN)

The MQRNN demand model follows an encoder–decoder design. Two input encoders map (i) the history window (calendar features, order-statistics, and SKU/brand IDs via learned embeddings) and (ii) the prediction window (calendar features and the same IDs) into compact contexts. An LSTM then processes the history context concatenated with the observed target series. A two-level decoder produces multi-horizon quantiles: a global MLP builds horizon-agnostic and horizon-specific contexts, and a local head refines each horizon.

Non-crossing & non-negative quantiles. The quantile head is centered at the median as an anchor. It predicts nonnegative "gaps" for higher and lower quantiles using a `softplus` transform and accumulates them outwards from the median, which ensures ordered (non-crossing) quantiles. A final `ReLU` enforces non-negative forecasts.

Training objective. Training uses the pinball (quantile) loss summed over all horizons and quantiles, with early stopping on a validation split.

Hyperparameters.

Setting	Value
LSTM hidden width	128
LSTM layers	2
Dropout	0.53
Batch size	32
Learning rate	4.5×10^{-4}
Weight decay	3.0×10^{-5}
SKU / Brand embedding dims	$(16,\ 12)$

H.3 The Optimization Proxy

The proxy is a modular MLP that emulates the single-DC allocation decision for each (order, SKU) pair. Inputs include (i) standard per-row features (order/DC signals and ID embeddings for SKU and brand), (ii) stochastic scenarios (scalar demand draws and per-DC cost vectors, aggregated across scenarios), and (iii) the on-hand inventory vector. Three MLP submodules process the standard features, the scenario features (separate demand/cost branches with mean aggregation and fusion), and the inventory; their outputs are concatenated and passed to an output MLP that produces DC logits.

Hyperparameters.

Setting	Value
Hidden width	256
MLP layers (per block)	2
Dropout	0.20
SKU / Brand embedding dims	$(8,\ 6)$
Batch size	128
Learning rate	2.0×10^{-4}
Constraint loss weight λ_{cons}	1.25

I Implementation Details

The evaluation proceeds in a rolling-horizon manner, where for each order arriving in the test week, the model makes a fulfillment decision, and the system state is updated. The C-SAA oracle is solved using Gurobi 11.0 with four threads. For each decision, run the SAA procedure detailed in Appendix D: $S = 10$ candidate solutions are generated using $N_1 = 50$ scenarios each, and the best is selected after evaluation on a common set of $N_2 = 500$ scenarios. Each MIP instance is solved to a strict 0.01% gap tolerance. The proxy model and all forecasting models were trained on a cluster equipped with NVIDIA RTX 6000 GPUs.