Delft University of Technology, Aerospace Engineering (2025) , 1–32
MSc Thesis, Control and Simulation (C&S)

ARTICLE

Real-Time Adaptive Production Scheduling for an E-Grocery Fulfillment Center

Guillermo Van[1] and
supervision by Dr. Patrick Stokkink[3], Dr. Alessandro Bombelli[1] and Ir. van der Maesen[2]

[1]Faculty of Aerospace Engineering, Delft University of Technology, Delft, The Netherlands
[2]Picnic Technologies, Amsterdam, The Netherlands
[3]Faculty of Technology, Policy and Management, Delft University of Technology, Delft, The Netherlands

Abstract
The rapidly growing e-grocery sector demands fulfillment centers (FCs) that operate with high efficiency and robustness, even under strong variability in task durations. Traditional scheduling approaches, such as the Earliest Due Date–First Served (EDD-FS) heuristic, lack real-time adaptability and neglect uncertainty. This study develops a real-time adaptive scheduling framework that combines predictive analytics with optimization to improve the timeliness and reliability of picking operations in a large-scale e-grocery FC. Gradient Boosted Decision Tree (GBDT) models are trained to predict active and inactive batch picking durations, significantly outperforming baseline methods. These predictions feed into two Mixed-Integer Linear Programming (MILP) schedulers: a deterministic variant using point estimates, and a scenario-based variant that incorporates uncertainty via sampled task durations. Evaluated in Picnic's proprietary warehouse simulation system, the deterministic scheduler reduces total delay by 16% and delayed batches by 8.5%, though performance depends on the EDD-FS weighting. The scenario-based scheduler achieves further improvements, decreasing total delay by 6.3–25.6% and maximum delay by 16.6–37.9%, with higher scenario counts improving outcomes at the cost of computation. These results show that integrating predictive models with deterministic and scenario-based optimization enhances schedule robustness compared to heuristic baselines. Future work should validate the framework in live FC operations, explore rolling-horizon and metaheuristic extensions, and assess reinforcement learning as an alternative for adaptive decision-making. This study thus provides a novel methodological framework with both academic significance and direct operational relevance for the rapidly expanding e-grocery sector.

Keywords: E-grocery fulfillment; Production scheduling; Gradient Boosted Decision Trees; Predictive Analytics integrated Optimization; Scenario-based Scheduling

Abbreviations: **EDD-FS** - Earliest Due Date - First Served; **FC** - Fulfillment Center; **GBDT** - Gradient Boosted Decision Trees; **MILP** - Mixed Integer Linear Program

1. Introduction

Over the past decade, the grocery sector has undergone a profound transformation driven by the emergence and rapid growth of e-grocery platforms [1]. Unlike traditional grocery retail, which relies on a store-based model, e-grocery businesses must fulfill and deliver diverse customer orders with short lead times and tight delivery windows. This shift has elevated the importance of opera-

Thesis paper written by Guillermo Van to obtain the Master of Science at the Delft University of Technology. To be defended publicly on the 30th of October 2025.

tional efficiency and agility across the entire supply chain. In particular, the ability to dynamically adapt to fluctuating demand and operational disturbances has become a competitive necessity in ensuring cost efficiency and customer satisfaction [2].

At the heart of the e-grocery supply chain lie the fulfillment centers (FCs). These are dedicated facilities where customer orders are processed through a sequence of tightly orchestrated tasks including replenishment, picking, stacking, and dispatching. Unlike conventional warehouses that rely on batch-based planning and bulk movement of goods, e-grocery FCs operate in near-continuous flow, demanding fine-grained allocation of resources such as human pickers and automated conveyor systems in the production schedule.

The production schedule of FCs has relied traditionally on static or rule-based approaches, with the Earliest-Due-Date-First-Served (EDD-FS) policy being one of the most common [3]. While easy to implement, this strategy lacks real-time flexibility and, as shown by Raman et al. (1989) [4], performs poorly in unbalanced systems. As FCs scale in complexity and throughput, the limitations of traditional scheduling techniques become increasingly apparent. This has led to a growing interest in real-time adaptive scheduling methods that continuously update task plans in response to the evolving system state [5].

Real-time adaptive scheduling, however, introduces a distinct set of challenges. First, any adaptive system must balance responsiveness with stability, ensuring that schedules are updated often enough to remain relevant while avoiding excessive volatility. Next, the system must operate under stringent time constraints, requiring fast inference and optimization even in high-dimensional decision spaces. Finally, a particularly salient challenge in this context is uncertainty. The processes inside an FC are influenced by a wide range of stochastic variables, such as human picking performance and stock availability. These factors introduce substantial variability in task completion times, which can compromise the reliability of production schedules if not properly accounted for [1].

This paper contributes toward advancing real-time adaptive scheduling in FCs through the development of a scheduler that uniquely integrates three key components into a unified system. First, it introduces a machine learning model based on Gradient Boosted Decision Trees (GBDT) to predict task durations, tailored for integration in time-sensitive production environments. Second, it includes an optimization model that allocates task start times while incorporating resource constraints, execution deadlines, and predictive insights. Third, it extends the scheduling properties to incorporate uncertainty via a scenario-based approach, enhancing the robustness of both the predictive and optimization components. Combined, these contributions offer insights and methodologies for the implementation of robust, data-driven adaptive production schedules inside e-grocery FCs.

The remainder of this paper is structured as follows. Section 2 surveys related work on real-time adaptive scheduling, optimization methods, and the handling of uncertainty. Section 3 describes the proposed methodology, including the development of the GBDT prediction model, the MILP-based scheduler, and their integration. Section 4 presents experimental results based on simulations, benchmarking the proposed framework against existing methods. Finally, Section 5 concludes the principal results and proposes directions for future research.

2. Literature review

This chapter reviews prior work on real-time adaptive scheduling relevant for e-grocery fulfillment. It summarizes rescheduling policies, optimization methods that work under short replanning cycles, and approaches to modeling uncertainty. The review closes by diving into several research gaps found in current literature.

2.1 E-grocery FC's adaptive production scheduling properties

The scheduling problem in e-grocery fulfillment operations can be defined as a Resource-Constrained Scheduling Problem (RCSP), in which a finite set of interdependent tasks must be allocated over time to limited resources [6]. Adaptive scheduling is the capability to dynamically revise planned replenishment, picking, or dispatching tasks in response to changes in the current operational state. While uncertainty in task durations falls outside the original framework, it is addressed separately alongside the three key dimensions of adaptive rescheduling defined by Vieira et al. (2003) [7]: rescheduling environment, policy, and method.

Task duration uncertainty is especially pronounced in e-grocery fulfillment due to the high degree of order heterogeneity, fluctuating worker performance, and varying stock availability. Unlike traditional manufacturing systems, where task durations are often standardized and repeatable, e-grocery operations exhibit considerable variability in processing times [8]. This is particularly evident during picking, where task durations are highly sensitive to dynamic system conditions.

Adaptive scheduling problems in e-grocery fulfillment operations can be characterized as dynamic job shop environments with stochastic task durations. While the set of jobs is often finite and known in advance, stochasticity is introduced through variability in workforce performance, equipment reliability, and process bottlenecks. A key contributing factor is the heterogeneity of customer orders, which vary in composition and size. As a result, each order follows a distinct route through the system, engaging different combinations of picking zones and resources. This variability in routing and resource usage justifies modeling the environment as a job shop scheduling problem, where task sequences are job-specific and must be coordinated across shared, constrained resources [9].

In adaptive scheduling, updates can follow either a periodic or event-driven policy. Event-driven rescheduling reacts instantly to disruptions, offering high responsiveness but often resulting in unstable schedules and increased computational overhead [10]. Periodic rescheduling, by contrast, updates the schedule at fixed intervals, allowing multiple system changes to be addressed in a coordinated manner. While this may delay responses to critical disruptions, it improves schedule stability and operator interpretability-qualities particularly valuable in stochastic job shop environments such as e-grocery FCs [11].

In job shop scheduling problems with a finite and known set of jobs, complete schedule regeneration is often the most appropriate rescheduling method. Unlike partial repair or right-shift approaches, where operations are simply postponed while preserving their original sequence, complete regeneration re-optimizes the entire remaining schedule from the current state. This allows the system to account holistically for updated task durations and resource availability [7]. This is particularly effective in environments with stochastic disruptions and interdependent tasks, where earlier decisions may propagate delays downstream. While computationally more demanding, several studies confirm that in stochastic job shop settings with finite job sets, complete regeneration yields better overall performance and schedule quality than reactive or locally myopic strategies [7, 12].

2.2 Optimization methods for adaptive production schedules

Adaptive production scheduling problems have traditionally been addressed using exact solution methods, such as Branch-and-Bound (B&B), Branch-and-Cut (B&C), and decomposition-based methods. These approaches provide strong guarantees on solution quality and are well-suited for relatively small instances with known parameters [13, 14, 15]. However, their high computational load limits their direct applicability in real-time contexts. Recent studies addressing adaptive rescheduling still rely on exact formulations but restrict the problem scope to ensure tractability under uncertainty and short re-optimization cycles [16].

To improve responsiveness, many studies have turned to metaheuristics such as Simulated Annealing (SA), Genetic Algorithms (GA), and Large Neighborhood Search (LNS), which offer flexible and scalable alternatives without requiring problem-specific reformulations [17, 18, 19]. These approaches have been shown to generate near-optimal solutions under time constraints, making them more compatible with real-time adaptation. However, they often require problem-specific tuning and may lack robustness under significant disturbances, especially in job shop environments with a high degree of heterogeneity [20].

More recently, machine learning (ML) methods - including Reinforcement Learning (RL) and hybrid approaches combining predictive analytics with exact solvers - have gained attention for their adaptiveness. For instance, Yamashiro and Nonaka (2021) [16], Li et al. (2020) [21], and Senoner et al. (2023) [22] integrate task duration predictions from ML models - such as Gradient Boosted Decision Trees (GBDT) - into scheduling optimization frameworks, demonstrating improved performance under uncertainty. These hybrid methods enable anticipatory scheduling adjustments, striking a balance between robustness and performance - characteristics especially valuable in the settings of an e-grocery FC.

2.3 Stochastic task durations in adaptive production schedules

Several studies have demonstrated the value of incorporating stochastic elements in production schedules where variability and tight operational constraints coexist. Escudero et al. (2003) [23] developed a multistage stochastic framework for integrated production and transportation planning, showing improved robustness against forecast errors and process disruptions. More recently, Raj et al. (2024) [24] applied a stochastic model to joint picking and delivery operations, resulting in better deadline adherence under volatile picking durations. Similar approaches have been explored in healthcare scheduling, airline operations, and make-to-order production [25, 26]. While adaptive rescheduling is already common in e-grocery FCs, stochastic elements add value by anticipating variability rather than solely reacting to it - strategically allocating resources and buffer times to reduce the need for frequent schedule changes. This anticipatory capability aligns particularly well with the characteristics of job shop-like environments that feature heterogeneous tasks and finite job sets. In this context, single-stage scenario-based stochastic models are particularly suitable, as they embed uncertainty into each rescheduling cycle without requiring long-term scenario trees, making them well aligned with periodic adaptation [27].

2.4 Contributions of this study

This study introduces a unified, operational framework that integrates predictive analytics, optimization, and uncertainty through scenario-based programming to address real-world scheduling challenges and fill key gaps in the academic literature. From a technical standpoint, the combination of predictive analytics and optimization is chosen for its strong potential to deliver both optimal and adaptive scheduling decisions. This hybrid approach enables real-time predictions of picking durations to be embedded into a MILP-based scheduler, effectively implementing a data-driven and anticipatory scheduling strategy for environments with high variability.

From a research perspective, Table 1 highlights several underexplored areas in the existing literature. While recent studies have linked predictive models with optimization, none have extended this framework to incorporate stochasticity. Moreover, the combined integration of predictive uncertainty, order heterogeneity, and real-time adaptive scheduling remains largely unaddressed. This study addresses that gap by applying a scenario-based MILP in a live e-grocery FC setting - making it, to the best of current knowledge, the first to unify these elements into a single operational framework.

Table 1. Comparison of relevant literature with respect to key modeling and application aspects.[1]

Content	Predictive analytics + optimization	Real-time adaptiveness	Finite set of jobs	Minimize tardiness	Include stochasticity	Order heterogeneity
Yamashiro & Nonaka (2021) [16]	X		X			
Frye et al. (2019) [28]	X	X	X	X		
Li et al. (2020) [21]	X	X	X	X		
Senoner et al. (2023) [22]	X		X	X		X
This study	X	X	X	X	X	X

[1] "Predictive analytics + optimization" refers to the integration of ML-based task duration predictions with optimization methods; "Real-time adaptiveness" indicates periodic or event-driven rescheduling; "Finite set of jobs" implies that the job set is known and bounded within each rescheduling cycle; "Minimize tardiness" refers to objective functions that penalize late completions relative to deadlines; "Include stochasticity" refers to the inclusion of stochastic durations during schedule optimization; "Order heterogeneity" denotes variability in task structures due to differences in order content and routing.

3. Methodology

This section outlines the methodology used to build a real-time adaptive scheduler for the picking stage in an e-grocery FC. First, the problem setting is defined, then the development of the gradient-boosted tree models for task-duration predictions is outlined, including a probabilistic variant. Next, the section formulates deterministic and scenario-based MILP schedulers with their assumptions and constraints. Finally, it describes how predictions and optimization are integrated into an integrated scheduler.

3.1 Problem description

This study focuses on the picking stage of the e-grocery fulfillment process - an operation characterized by high variability and a job-shop structure. Given its strong influence on downstream performance and its susceptibility to delays, picking provides a particularly compelling environment to evaluate the benefits of adaptive scheduling. Its position within the overall FC process is shown in Figure 1.

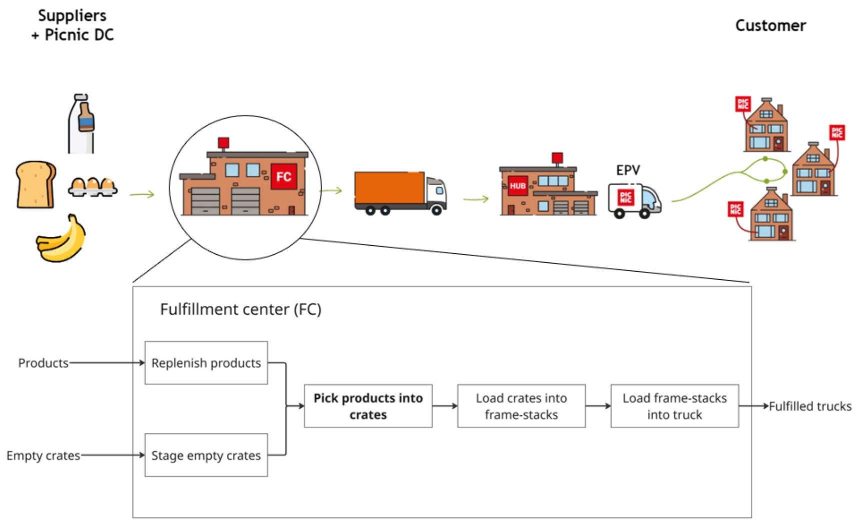
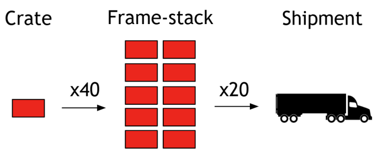
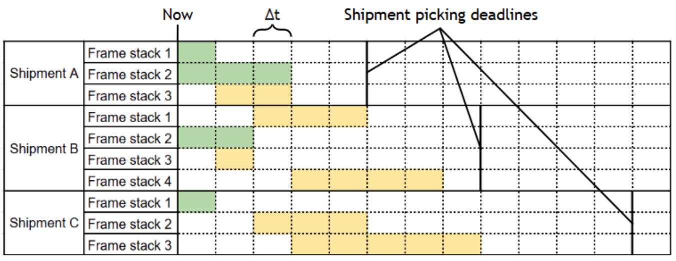


Figure 1. Flow of processes inside an e-grocery FC in the context of its complete supply chain.

The developed scheduling algorithm is specifically designed for the picking operations, the process of placing products inside customer crates, within an e-grocery FC. The customer crates are subsequently assembled together into frame-stacks as part of a shipment, as illustrated in Figure 2a. The shipment picking deadline defines the latest permissible picking completion time for all frame-stacks within a shipment to prevent any downstream delays. Due to the real-time adaptive nature of the scheduling approach, the starting point of the schedule continuously shifts as time progresses and new operational data becomes available, resulting in the rolling schedule structure shown in Figure 2b.



(a) A shipment consists of 20 frame-stacks, each containing 40 crates.



(b) Conceptual visualization of the rolling picking schedule in an e-grocery FC. Green and yellow blocks indicate frame-stacks that have started and not yet started picking, respectively.

Figure 2. Structural and temporal example representation of the picking process in an e-grocery FC.

The methodological framework of this study consists of three interconnected stages: (i) development of a Gradient Boosted Decision Tree (GBDT) model to predict task durations, (ii) construction of a Mixed-Integer Linear Programming (MILP) model to schedule individual frame-stacks, and (iii) integration of both components into a real-time adaptive scheduling system driven by predictive analytics. A high-level conceptual representation of the developed framework is shown in Figure 3.

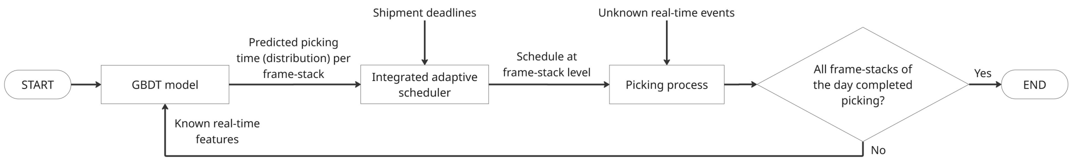

START
GBDT model
Predicted picking time (distribution) per frame-stack
Shipment deadlines
Integrated adaptive scheduler
Schedule at frame-stack level
Unknown real-time events
Picking process
All frame-stacks of the day completed picking?
Yes
END
No
Known real-time features

Figure 3. Top-level structure of the developed real-time adaptive scheduling framework.

3.2 GBDT model

GBDT was selected as the ML-based method for predicting frame-stack completion times due to its strong performance on tabular data, robustness to multicollinearity, and ability to handle missing values natively. These characteristics make GBDT particularly suitable for operational data in FCs, which often includes correlated, sparse, or incomplete records [29]. In addition, recent advances allow GBDT models to estimate predictive uncertainty, enabling integration into stochastic optimization frameworks and making them well-suited for real-time adaptive scheduling under uncertainty [30].

The frame-stacks present in the production schedule can be categorized as either active or inactive. A frame-stack is considered active if at least one of its associated crates has begun the picking process, meaning that picking activities for that frame-stack are already in progress. Conversely, a frame-stack is inactive if none of its crates have been picked yet, and its picking process has not started. The developed GBDT model provides the scheduler with the predicted remaining completion time for active frame-stacks, or the expected total completion time for inactive frame-stacks, as illustrated in the graph of Figure 4.

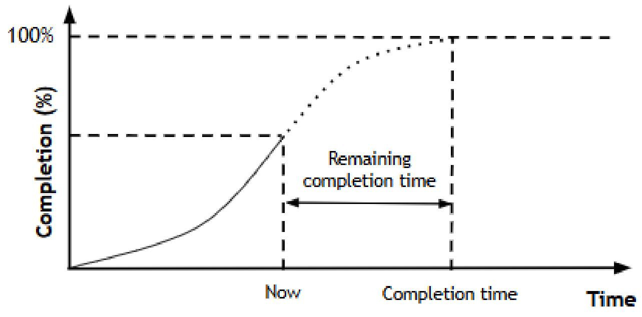


Figure 4. Visualization of the remaining completion time for picking a frame-stack as target variable for the GBDT model to predict.

3.2.1 Feature engineering

The process of developing the GBDT model to predict frame-stacks' completion times is inspired by the Cross-Industry Standard Process for Data Mining (CRISP-DM) as described by Chapman et al. (2000) [31]. The first step, feature selection and preprocessing, is carried out in a structured way by classifying the feature set into *time-dependent* and *time-independent* categories. This classification indicates whether a feature's value changes over the course of the day. Using this distinction as a starting point, the features listed in Table 2 were compiled.

For active frame-stacks - those that have already started the picking process - both time-dependent and time-independent features are available. For inactive frame-stacks, only time-independent features can be used, as no real-time progress information is yet available. Consequently, two separate GBDT models are trained: one using the full feature set for active frame-stacks, and one using only time-independent features for inactive frame-stacks.

Table 2. Overview of the complete feature set and target variable.

Feature / Target	**Time dependent?**	**Description**
Time of day	Yes	Time of the day in absolute minutes since midnight
Number of recent picks	Yes	Number of picks done within the last 30 minutes
Number of recent pickers	Yes	Number of active pickers in the last 30 minutes
Remaining crates to pick	No	Number of crates in the frame-stack that have not finished picking
Weekday	No	Current day of the week in string format
Running picking time	No	Amount of time a frame-stack has been picking
Running completion	No	Fraction of crates in the frame-stack that have completed picking
Temperature zone	No	Temperature zone of the crates in the frame-stack
Upcoming area visits	No	For all remaining crates to be picked, the planned visit count per area
Planned shipment picking deadlines	No	The picking deadline for the entire shipment of the respective frame-stack in minutes since midnight today
Target: Remaining picking time	Yes	Remaining time until the frame-stack is fully picked since (expected) starting time

Before training the GBDT model, the dataset is preprocessed to ensure compatibility with the learning algorithm. Categorical variables, such as the weekday and temperature zone, are encoded into numerical representations using one-hot encoding. Additionally, missing values are handled using the model's built-in support for missing data, and outliers are capped where necessary to prevent undue influence on the training process. Finally, the dataset is split into training and validation sets to evaluate model performance during development as will be described later.

3.2.2 Model architecture

The GBDT model predicts completion times by sequentially fitting an ensemble of shallow decision trees to minimize the squared error loss. At each iteration, a new tree is fitted to the residuals of the ensemble prediction thus far, gradually correcting previous errors. This additive approach enables the model to capture complex non-linear patterns while maintaining relatively low risk of overfitting when properly regularized. A visualization of the process is depicted in Figure 5.

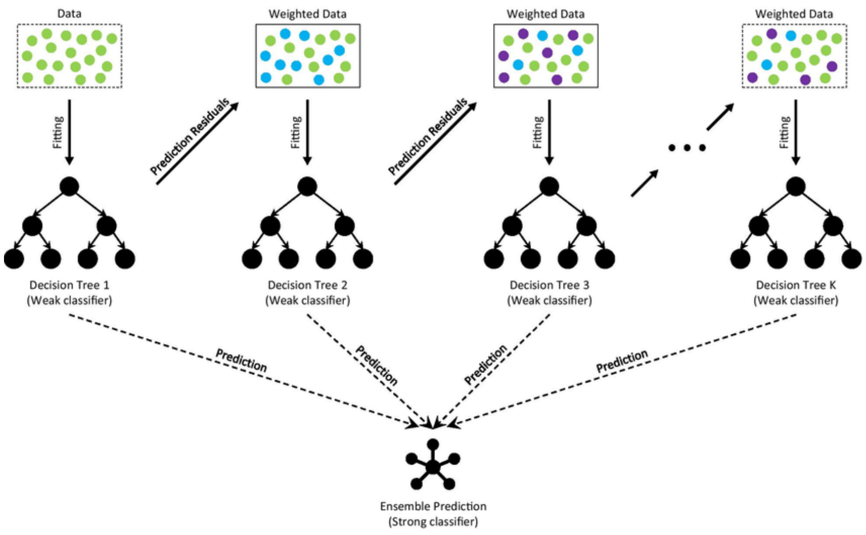


Figure 5. The architecture of a Gradient Boosting Decision Trees (GBDT) model [32].

Several hyperparameters directly influence the model's capacity and generalization. The maximum number of boosting iterations determines the ensemble size, while the learning rate scales the contribution of each tree. To prevent overly deep splits and excessive variance, the maximum tree depth is limited. Furthermore, early stopping is applied by monitoring the validation loss on a reserved fraction of the training set, terminating training once performance stabilizes. A fixed random seed ensures reproducibility across experiments. An overview of all the hyperparameters necessary for training and evaluation is shown in Table 3.

The model architecture is implemented in `Python` using the `scikit-learn` library. The gradient boosting algorithm is instantiated via the `HistGradientBoostingRegressor`, which allows for efficient training on large datasets through histogram-based binning while providing support for missing feature values. Feature preprocessing is handled by `scikit-learn`'s `Pipeline` and `ColumnTransformer` modules, applying one-hot encoding to categorical features via `OneHotEncoder` and standardization to numerical features via `StandardScaler`. This modular pipeline ensures consistent and reproducible data transformations during model fitting and prediction.

3.2.3 Model training and tuning

To train the GBDT model, the available dataset is first split into a training set and a test set. Both sets consist of historic feature data labeled with the true remaining completion times. The training

Table 3. Overview of hyperparameters used in the GBDT model.

Parameter	Description
Max iterations	Number of boosting iterations (i.e., trees). Controls ensemble size and model complexity.
Learning rate	Shrinkage parameter determining the contribution of each tree. Acts as main regularization term.
Maximum tree depth	Maximum depth of individual trees. Limits the complexity of single trees to avoid overfitting.
Validation fraction	Fraction of training data held out for early stopping. Used to monitor generalization performance.
Early stopping	Enables early stopping when validation loss no longer improves.
Random state	Random seed to ensure reproducibility across runs.
Cross-validation folds	Number of folds used during cross-validation.

set is used to fit the model and select hyperparameters, while the test set is strictly held out for final evaluation of model performance.

To assess the generalizability of the model and avoid overfitting, a k-fold cross-validation approach is applied on the training set. Once the optimal hyperparameters are selected based on cross-validation performance, the final model is retrained on the complete training set using the best hyperparameter configuration. The final model is then evaluated once on the test set to obtain an unbiased estimate of the true model accuracy.

The full training procedure is summarized in algorithm 1. In Phase 1, the dataset is preprocessed, split into active and inactive subsets, and outliers are capped. In Phase 2, the model is trained per mode using k-fold cross-validation for hyperparameter tuning, after which the final models are trained on the full training sets and stored. In Phase 3, these models are evaluated on held-out test sets using several performance metrics.

3.3 Stochastic GBDT model variant

The stochastic variant of the GBDT model differs from the deterministic version in that it no longer predicts a single fixed value for each frame-stack's picking duration, but instead estimates a full probability distribution. This enables the model to capture both the expected completion time and the uncertainty around that estimate, providing richer input for downstream decision-making. The model is implemented using the `NGBoost` package, which extends gradient boosting to probabilistic outputs by fitting distributions rather than point estimates. While the input features, preprocessing pipeline, and data splits remain unchanged, the stochastic model is trained to represent the variability observed in historical data.

The complete training and evaluation workflow for the stochastic model is summarized in algorithm 2. In Phase1, the same data preparation steps as in the deterministic pipeline are applied. In Phase2, each mode is trained using an `NGBoost` regressor that assumes a *LogNormal* predictive distribution and minimizes the negative log-likelihood loss. The choice of a *LogNormal* distribution is motivated by expert insights from Picnic, who identify the general shape of frame-stack completion times as log-normally distributed. Hyperparameters are tuned via k-fold cross-validation, using the standard performance metrics (RMSE, MAE, MAPE) along with the probabilistic Continuous Ranked Probability Score (CRPS). The CRPS provides a unified measure of both the sharpness and calibration of probabilistic forecasts by comparing the cumulative distribution function (CDF) of the predicted distribution to the empirical CDF of the observed value as formalized in Equation 1. In this expression, $F(z)$ denotes the predicted CDF evaluated at value z, and $1_{\{z\geq y\}}$ is the indicator function that equals 1 if $z \geq y$ and 0 otherwise, where y is the observed target value. In Phase 3, the

Algorithm 1: Deterministic GBDT Model Development Pipeline

```
1:  Phase 1: Data Preparation
2:  Load complete historic picking dataset file $\mathcal{D}$.
3:  Split $\mathcal{D}$ into four subsets based on timestamps and running completion: train-active,
      train-inactive, test-active, test-inactive.
4:  Apply remaining time cap: discard rows with REMAINING_PICKING_TIME exceeding threshold
      $T_{\max}$.
5:  Define hyperparameter grid $\mathcal{G}$ for grid search.
6:  Phase 2: Model Training
7:  foreach mode ∈ {Active, Inactive} do
8:      Extract features $X$ and targets $y$ from corresponding training dataset (train-active or
          train-inactive).
9:      Construct preprocessing pipeline with standardized (numerical) and one-hot encoded
          (categorical) features.
10:     Define a HistGradientBoostingRegressor that minimizes the squared-error (L2) loss.
11:     foreach combination of hyperparameters $\theta \in \mathcal{G}$ do
12:         Apply parameters $\theta$ to pipeline.
13:         Perform $k$-fold cross-validation with metrics: RMSE, MAE, MAPE.
14:         Compute mean validation performance.
15:     Fit pipeline on full training data using $\theta^*$ ($\theta$ with best validation score).
16:     Save trained model to disk.
17: Compute mean train-inactive target values for baseline.
18: Phase 3: Model Testing
19: foreach mode ∈ {Active, Inactive, Baseline} do
20:     Use corresponding test dataset (test-active or test-inactive).
21:     if mode = Baseline then
22:         Predict constant mean on test targets.
23:     if mode ≠ Baseline then
24:         Load saved model from disk.
25:         Predict remaining picking time on test set.
26:     Compute metrics: RMSE, MAE, MAPE, residual variance and standard deviation.
```

saved models are evaluated on held-out test sets. In addition to point-estimate metrics, the CRPS and the predicted standard deviation (in minutes) are reported to quantify the predictive performance of the distribution.

$$\mathrm{CRPS}(F, y) = \int_{-\infty}^{\infty} \left(F(z) - 1_{\{z \geq y\}}\right)^2 \, \mathrm{d}z \tag{1}$$

3.4 MILP model

3.4.1 Model description

The MILP model generates a frame-stack-level schedule that is updated each time the optimizer is triggered. During each update, the model reallocates start times for all currently inactive frame-stacks in a way that maximizes the cumulative weighted slack: the time buffer between a frame-stack's expected completion and its associated shipment picking deadline. To ensure the schedule is operationally feasible, the model enforces several constraints. These include limitations on the

Algorithm 2: Stochastic GBDT Model Development Pipeline

```
1:  Phase 1: Data Preparation
2:  Same as for algorithm 1
3:  Phase 2: Model Training
4:  foreach mode ∈ {Active, Inactive} do
5:      Extract features $X$ and targets $y$ from corresponding training dataset (train-active or
          train-inactive).
6:      Construct preprocessing pipeline with standardized (numerical) and one-hot encoded
          (categorical) features.
7:      Define an NGBRegressor that minimizes the negative log-likelihood under a LogNormal
          distribution.
8:      Train using probabilistic gradient boosting with negative log-likelihood as the loss function.
9:      foreach combination of hyperparameters $\theta \in \mathcal{G}$ do
10:         Apply parameters $\theta$ to pipeline.
11:         Perform $k$-fold cross-validation with metrics: RMSE, MAE, MAPE, CRPS.
12:         Compute mean validation performance.
13:     Fit pipeline on full training data using $\theta^*$ ($\theta$ with best validation score).
14:     Save trained model to disk.
15: Fit LogNormal distribution on train-inactive target values for baseline.
16: Phase 3: Model Testing
17: foreach mode ∈ {Active, Inactive, Baseline} do
18:     Use corresponding test dataset (test-active or test-inactive).
19:     if mode = Baseline then
20:         Predict precomputed LogNormal distribution on test targets.
21:     if mode ≠ Baseline then
22:         Load saved model from disk.
23:         Predict remaining picking time distribution on test set.
24:     Compute metrics: RMSE, MAE, MAPE, CRPS, and predicted standard deviations.
```

allowable start window for each frame-stack, the maximum number of frame-stacks that can be picked concurrently, and a required time gap between consecutive frame-stack start times. These constraints reflect both physical resource limits and embedded execution logic within the FC.

Two variants of the model have been developed. The deterministic MILP uses point estimates from the GBDT model and serves as a baseline under nominal conditions. The scenario-based MILP, by contrast, incorporates predictive uncertainty by evaluating multiple possible duration outcomes for each frame-stack drawn from a predicted distribution. This allows the scheduler to hedge against a wider range of potential delays, aiming to improve reliability and reduce the risk of extreme deadline violations.

3.4.2 Modeling assumptions

The formulation of the MILP scheduler relies on a number of key modeling assumptions to ensure tractability and alignment with the operational constraints of the FC environment. First, the objective of maximizing cumulative weighted buffer time is assumed to align with the higher-level goal of minimizing the risk of tardiness. By allocating buffer ahead of shipment deadlines - weighted more heavily for short-horizon frame-stacks - the model indirectly minimizes expected delays and improves schedule robustness, especially under uncertainty.

Second, the number of frame-stacks allowed to begin picking in each time-slot is statically constrained. This upper bound is determined based on the total number of frame-stacks to be scheduled within the day and the number of discrete time-slots available in the planning horizon. This assumption ensures a controlled and smooth distribution of start times across the shift, avoiding clustering that could overload resources.

Third, the capacity constraint is derived from the physical limitations of the frame-stack buffer zone - an intermediate holding area between picking completion and the dispatch process. This buffer has a fixed size, and exceeding its capacity would lead to operational bottlenecks or require unmodeled interventions such as early crate loading. Therefore, the MILP includes a hard cap on the number of overlapping picking tasks to prevent infeasible solutions.

Finally, it is assumed that task durations are exogenous and predicted prior to optimization using the GBDT model. These predictions are taken as inputs to the MILP in either deterministic or stochastic form, and the scheduling model does not update durations within the optimization loop. This separation allows the scheduling problem to remain linear and computationally efficient while still benefiting from data-driven duration estimates.

3.4.3 Deterministic MILP model formulation

The sets with indices, parameters, decision variables and constraints for the deterministic MILP model variant are formulated below.

Sets and Indices

- $\mathcal{I} = \{1, \ldots, N\}$: all frame-stacks, indexed by i.
- $\mathcal{I}^{\text{act}} \subseteq \mathcal{I}$: the subset of "active" frame-stacks.
- $\mathcal{T} = \{0, 1, \ldots, T_{\max}\}$: set of time-slots (each of length Δ minutes).

Parameters

- D_i: deadline of frame-stack i, in minutes since epoch.
- T_0: schedule start time, in minutes since epoch.
- L_i: picking duration of frame-stack i (minutes).
- Δ: length of one time-slot (minutes).
- C: maximum number of concurrent frame-stacks per time-slot (capacity).
- Y: maximum number of concurrent starting frame-stacks per time-slot.
- $W_i = \dfrac{w}{\max\{\epsilon, D_i - T_0\}}$: weight giving higher priority to short-horizon frame-stacks (with small $\epsilon > 0$ to avoid division by zero and configurable parameter w).

Decision Variables

- $x_{i,t} \in \{0, 1\}$: 1 if frame-stack i starts in time-slot t, 0 otherwise.

Objective

Maximize the weighted slack between predicted picking finishing time and respective picking deadline:

$$\max_{x} \sum_{i \in \mathcal{I}} W_i \left(D_i - \sum_{t \in \mathcal{T}} (t\Delta + T_0 + L_i) x_{i,t} \right) \tag{2}$$

Constraints

1. **Start time assignment** – Each frame-stack must start in exactly one time-slot:

$$\sum_{t \in \mathcal{T}} x_{i,t} = 1 \quad \forall\, i \in \mathcal{I} \tag{3}$$

2. **Active frame-stack locking** – Force pre-assignment for active frame-stacks:

$$x_{i,0} = 1 \quad \forall\, i \in \mathcal{I}^{\text{act}} \tag{4}$$

3. **Capacity constraint** – At each time-slot $a \in \mathcal{T}$, the number of frame-stacks that are active must not exceed the available picking capacity C. A frame-stack is considered active at time-slot a if it was scheduled to start at a time-slot t such that a lies within its picking duration. The constraint ensures that the total number of such frame-stacks remains within the allowed limit:

$$\sum_{i \in \mathcal{I}} \sum_{\substack{t \in \mathcal{T} \\ t \le a < t + \lceil L_i/\Delta \rceil}} x_{i,t} \le C \quad \forall\, a \in \mathcal{T} \tag{5}$$

4. **Minimum start-gap constraint** – At any slot t, at most Y inactive frame-stacks may start:

$$\sum_{i \in \mathcal{I} \setminus \mathcal{I}^{\text{act}}} x_{i,t} \le Y \quad \forall\, t \in \mathcal{T} \tag{6}$$

5. **Variable domains**:

$$x_{i,t} \in \{0,1\} \quad \forall\, i \in \mathcal{I},\ t \in \mathcal{T} \tag{7}$$

3.4.4 Scenario-based MILP model formulation

Instead of relying on a single point estimate per task, the scenario-based model evaluates multiple possible duration outcomes across scenarios, allowing it to hedge against potential delays while optimizing a unified schedule. By introducing soft capacity constraints that permit scenario-specific overflow with penalized violations, the model balances schedule robustness with operational performance under uncertainty.

The modeling process can be viewed as comprising two conceptual stages:

- **Sampling phase (pre-model)**: Following the Sample Average Approximation (SAA) framework, each frame-stack $i \in \mathcal{I}$ has an uncertain picking duration L_i. This is modeled as a lognormally distributed random variable, based on expert opinions as discussed before, with parameters estimated by the stochastic GBDT model. Formally,

$$L_i \sim \text{LogNormal}(\mu_i, \sigma_i^2),$$

  where μ_i and σ_i are scenario-independent parameters derived from predicted mean and standard deviation. Based on this distribution, K independent samples $\{L_i^s\}_{s=1}^{K}$ are drawn to construct a scenario set $\mathcal{S} = \{1, \ldots, K\}$, each assigned equal probability $p_s = \frac{1}{K}$.

- **Optimization phase**: Given the sampled durations $\{L_i^s\}$, a scenario-based MILP is solved that determines a single start time per frame-stack and introduces per-scenario overflow variables to accommodate capacity flexibility.

The sets with indices, parameters, decision variables and constraints for the scenario-based MILP model are formulated below.

Sets and Indices

- $\mathcal{I}$, $\mathcal{I}^{\text{act}}$, and $\mathcal{T}$ same as for deterministic MILP model.
- $\mathcal{T}^s_{i,a} = \{t \in \mathcal{T} \mid t \leq a < t + \lceil L^s_i/\Delta \rceil\}$: time-slots t such that frame-stack i is active at time-slot a in scenario s if started at t.
- $\mathcal{S} = \{1, \dots, K\}$: scenario indices, indexed by s.

Parameters

- D_i, T_0, Δ, C, Y, and W_i same as for deterministic MILP model.
- L^s_i: picking duration of frame-stack i in scenario s (minutes).
- p_s: probability of scenario s, such that $\sum_{s \in \mathcal{S}} p_s = 1$.
- ρ: penalty parameter for capacity overflow (per time-slot).

Decision Variables

- $x_{i,t} \in \{0, 1\}$: 1 if frame-stack i starts in time-slot t, 0 otherwise (same as for deterministic MILP model).
- $v^s_a \geq 0$: capacity overflow at time-slot a in scenario s.

Objective
Maximize the expected weighted slack minus a penalty for capacity violations:

$$\max_{x,v} \sum_{s \in \mathcal{S}} p_s \left[\sum_{i \in \mathcal{I}} W_i \left(D_i - \sum_{t \in \mathcal{T}} (t\Delta + T_0 + L^s_i) x_{i,t} \right) - \rho \sum_{a \in \mathcal{T}} v^s_a \right] \tag{8}$$

Constraints

1. **Start time assignment**, **active frame-stack locking**, **minimum start-gap** and the $x_{i,t}$ **variable domain** constraints are the same as Equation 3, Equation 4, Equation 6, and Equation 7 of the deterministic MILP model respectively.
2. **Soft capacity constraint (per scenario)** – At each time-slot $a \in \mathcal{T}$ and for every scenario $s \in \mathcal{S}$, the number of frame-stacks that are active must not exceed the available picking capacity C. A frame-stack is considered active at time-slot a in scenario s if it was scheduled to start at a time-slot t such that a lies within its scenario-specific picking duration, i.e., $t \leq a < t + \lceil L^s_i/\Delta \rceil$. To ensure feasibility while allowing for variability in predicted durations, this constraint permits capacity violations, which are absorbed into a non-negative overflow variable v^s_a. The extent of the violation is penalized in the objective, discouraging overload while maintaining flexibility:

$$\sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}^s_{i,a}} x_{i,t} \leq C + v^s_a \quad \forall\, a \in \mathcal{T},\ \forall\, s \in \mathcal{S} \tag{9}$$

3. **Overflow variable bounds** – Capacity overflow must be non-negative:

$$v^s_a \geq 0 \quad \forall\, a \in \mathcal{T},\ \forall\, s \in \mathcal{S} \tag{10}$$

3.5 Integrated adaptive scheduler

The final methodological component is the integration of the predictive and optimization modules into a unified real-time adaptive scheduling framework. The integrated scheduler serves as the decision-making layer that repeatedly monitors system progress, updates predictions, and re-optimizes the schedule during operations. This setup ensures that schedules remain responsive to dynamic changes in picking progress, resource availability, and forecast uncertainty.

The scheduler is triggered at fixed intervals, defined by the trigger period P, throughout the picking shift. Each time the trigger condition is met, the scheduler executes a three-step pipeline. First, the most recent operational data is collected, and the predictive model is invoked for every frame-stack depending on its status. If a frame-stack is active, the model provides a prediction of its remaining completion time; if it is inactive, the total expected completion duration is estimated; and if it is already completed, the frame-stack is excluded from further scheduling. Depending on the chosen mode, the deterministic GBDT outputs point estimates, while the stochastic NGBoost variant produces probability distributions that represent the uncertainty in completion times.

Algorithm 3: Integrated Adaptive Scheduler Pipeline (Deterministic and Scenario-Based)

```
1:  if current simulation time t is a multiple of the trigger period P then
2:      Phase 1: Predictions
3:      Collect recent picking activity: $n_{\text{picks}}$, $n_{\text{pickers}}$.
4:      foreach frame-stack $f \in F$ do
5:          if status($f$) = active then
6:              if scheduler mode is deterministic then
7:                  predict remaining picking time $\hat{T}_f^{\text{rem}}$.
8:              else if scheduler mode is scenario-based then
9:                  predict probability distribution $\mathcal{D}_f^{\text{rem}}$.
10:         else if status($f$) = inactive then
11:             if scheduler mode is deterministic then
12:                 predict picking completion duration $\hat{T}_f^{\text{comp}}$.
13:             else if scheduler mode is scenario-based then
14:                 predict probability distribution $\mathcal{D}_f^{\text{comp}}$.
15:         else if status($f$) = completed then
16:             continue (exclude).
17:     Phase 2: Scheduling of Frame-Stacks
18:     if scheduler mode is deterministic then
19:         Generate updated schedule $S(t)$ using $\{\hat{T}_f^{\text{rem}}, \hat{T}_f^{\text{comp}}\}$ and deadlines $\{\tau_f^{\text{dl}}\}$.
20:     else if scheduler mode is scenario-based then
21:         Construct a finite set of scenarios $\Omega$ from $\{\mathcal{D}_f^{\text{rem}}, \mathcal{D}_f^{\text{comp}}\}$.
22:         Solve the scenario-based scheduling model over $\Omega$ to obtain $S(t)$, with the objective aggregated across
              scenarios and completion times represented by their mean values in $S(t)$.
23:     Assign recommended start times $\hat{\tau}_f^{\text{start}}$ for all inactive frame-stacks.
24:     Phase 3: Prioritization and Execution Adjustments
25:     if scheduler mode is deterministic then
26:         Update prioritization function $\pi(f, t)$ using $\hat{T}_f^{\text{rem}}$, $\hat{\tau}_f^{\text{start}}$, and slack $(\tau_f^{\text{dl}} - t)$.
27:     else if scheduler mode is scenario-based then
28:         Update prioritization function $\pi(f, t)$ using risk-adjusted summaries of $\mathcal{D}_f^{\text{rem}}$ and $\mathcal{D}_f^{\text{comp}}$ together with slack
              $(\tau_f^{\text{dl}} - t)$.
29:     Execute frame-stacks in order of $\pi(f, t)$.
```

Using these predictions, the MILP optimization model then generates an updated picking schedule. In deterministic mode, the predicted durations are treated as fixed inputs, and the optimizer allocates start times for inactive frame-stacks such that weighted slack is maximized while respecting operational constraints. In scenario-based mode, samples are drawn from the predictive distributions to create a finite set of scenarios, and the model evaluates trade-offs across them.

Finally, the updated schedule is transformed into actionable picking priorities. The optimizer's recommended start times are compared against the current time and shipment deadlines, after which a prioritization function $\pi(f, t)$ determines the order in which frame-stacks should be released to pickers. In deterministic mode this function relies on predicted completion times and slack, whereas in scenario-based mode it also incorporates distributional information such as the mean and variance of predicted durations to account for risk. The resulting execution plan is implemented in the FC until the next trigger event, at which point the cycle repeats. In this way, the adaptive scheduler continuously aligns the planned schedule with the real-time operational state, mitigating disruptions and reducing the likelihood of deadline violations under uncertainty. A formalized description of the pipeline is provided in algorithm 3.

4. Results and Discussion

This section reports the experimental evaluation of the proposed algorithm at one of Picnic's FCs. First, the predictive accuracy of the GBDT models is assessed and interpreted through feature importance analysis. Next, the performance of the deterministic and scenario-based MILP schedulers is benchmarked against the EDD-FS heuristic and hindsight-optimal baselines. Finally, a sensitivity analysis investigates the robustness of the schedulers under varying parameter settings.

4.1 Experimental setup

The experimental evaluation is conducted using a case study based on one of Picnic's FCs, designed to assess the predictive performance of the proposed GBDT models and the scheduling efficiency of both the deterministic and scenario-based MILP schedulers. The setup is structured in three parts: model development, simulation environment, and performance metrics.

4.1.1 GBDT model development setup

The predictive models are trained on historical picking data from January 1 to May 1, 2025, and tested on an independent dataset from May 21 to May 27, 2025. This independent dataset was selected to span a full week without public holidays, ensuring that the evaluation reflects typical operational conditions. For both the deterministic and stochastic variants, the training pipeline follows a 5-fold cross-validation strategy to avoid overfitting and to tune hyperparameters. A picking time cap of 600 minutes is applied to discard extreme outliers. The deterministic model optimizes squared-error loss using the HistGradientBoostingRegressor, while the stochastic variant is implemented with NGBoost under a LogNormal distribution to represent predictive uncertainty. The full configuration of feature ranges and hyperparameters is summarized in Table 4.

4.1.2 Simulation setup

The developed scheduling algorithms are evaluated inside Picnic's proprietary Warehouse Simulation System (WSS), a Python-based discrete-event simulation environment that digitally replicates the operations of the FC. The simulator provides a detailed emulation of physical processes, including sensors, conveyors, and the behavior of objects such as crates and frame-stacks. By reproducing the flow of shipments and the constraints of the picking and buffering process, WSS enables testing of alternative scheduling strategies under realistic and controlled conditions without interfering with live operations. For this study, the schedulers are evaluated in a simulation of one of Picnic's FCs on January 24, 2025, chosen as a representative day. The environment contains 11 shipments with deadlines throughout the entire day, 105 frame-stacks in total, all in the ambient temperature zone, and includes a 30-minute warm-up phase before the first pick to stabilize initial conditions. The

Table 4. Overview of the GBDT model's configuration of input space ranges and parameters for training and evaluation.

Feature / Target	Range
Time of day	0–1439 min
Number of recent picks	0–5000
Number of recent pickers	0–60
Remaining crates to pick	0–40
Weekday	All days of the week
Running picking time	0–600 min
Running completion	0–1
Temperature zone	*Ambient*, *Chilled*
Upcoming area visits	0–1500
Planned shipment picking deadlines	0–1439 min
Target: Remaining picking time	0–600 min

Parameter	Value
Deterministic GBDT model	
Training dataset	01/01/2025 until 01/05/2025 (active: 1,328,864 rows, inactive: 33,222 rows)
Test dataset	21/05/2025 until 27/05/2025 (active: 77,039 rows, inactive: 19,260 rows)
CV folds	5
Picking time cap	600 min
Learning rate grid	[0.05, 0.1, 0.3]
Learning rate	0.1
Boosting rounds grid	[100, 200, 300]
Boosting rounds	200
Early stopping	True
Validation fraction	0.1
Stochastic GBDT model	
Training dataset	01/01/2025 until 01/05/2025 (active: 1,328,864 rows, inactive: 33,222 rows)
Test dataset	21/05/2025 until 27/05/2025 (active: 77,039 rows, inactive: 19,260 rows)
CV folds	5
Picking time cap	600 min
Learning rate grid	[0.05, 0.1, 0.3]
Learning rate	0.1
Boosting rounds grid	[100, 200, 300]
Boosting rounds	200
Distribution shape	LogNormal

deterministic and scenario-based MILP models are configured with identical operational parameters: a frame-stack capacity of 70, a maximum of 4 concurrent starting frame-stacks, a time-slot discretization of 6 minutes in the schedule, an EDD-FS priority weight of 1, and a trigger period for rescheduling of 6 minutes. The scenario-based variant additionally incorporates uncertainty by sampling 20 duration scenarios from the stochastic GBDT predictions. All configured parameters are summarized in Table 5.

4.1.3 Performance metrics

The evaluation is conducted across three dimensions. First, the predictive accuracy of the GBDT models is measured using RMSE, MAE, and MAPE for deterministic outputs, and additionally CRPS for probabilistic outputs. Second, scheduler performance is assessed through operational KPIs: objective value of the actual process in hindsight, average makespan per shipment, total delay, delayed frame-stacks count, and maximum delay. Third, computational reliability is examined by tracking rescheduling times over the course of the simulation runs.

4.2 GBDT model results

4.2.1 Model validation and testing

Model testing and validation proceed in two steps. First, five-fold cross-validation on the training period tunes hyperparameters and assesses generalization, with RMSE, MAE, and MAPE for de-

Table 5. Overview of the configured parameters in the simulation setup for both the simulation environment as well as the implemented scheduling model.

Parameter	Value
Picking environment	
Simulation date	24/01/2025
Number of shipments	11
Number of frame-stacks	105
Temperature zone	Ambient
Warm-up before first pick	30 min
MILP model	
Frame-stack capacity (C)	70
Frame-stack starting capacity (Y)	4
Time-slot length (Δ)	6 min
Short-horizon weight (w)	1
Trigger period	6 min
Number of scenarios (K in scenario-based MILP)	20

terministic models and CRPS for stochastic ones. Second, the selected models are evaluated on an independent test week against simple baselines, using paired comparisons where relevant. Results are reported separately for active and inactive frame-stacks to match their respective model's performance.

Five-fold cross-validation indicates that the models generalize well. As shown in Table 6, fold-to-fold variability is very small (active deterministic: RMSE 33.74 ± 0.11 min, MAE 24.91 ± 0.06 min). The ranking is consistent across folds: active models outperform inactive ones, and stochastic variants show slightly higher point errors while providing calibrated distributions (CRPS 39.96 ± 0.043 min for active vs. 57.97 ± 0.10 min for inactive). The narrow standard deviations across metrics support the conclusion that performance reflects robust, generalizable behavior rather than fold-specific deviations.

Table 6. Cross-validation results for the deterministic and stochastic GBDT models for assessing the generalizability of the trained model. Values are reported as mean ± standard deviation over $k = 5$ folds.

Model type	RMSE (min)	MAE (min)	MAPE (%)	CRPS (min)
Deterministic model				
GBDT model (active frame-stacks)	33.74 ± 0.11	24.91 ± 0.06	54.31 ± 0.27	-
GBDT model (inactive frame-stacks)	53.21 ± 0.59	39.87 ± 0.18	20.24 ± 0.10	-
Stochastic model				
GBDT model (active frame-stacks)	37.45 ± 0.07	27.94 ± 0.09	59.53 ± 0.26	39.96 ± 0.043
GBDT model (inactive frame-stacks)	54.84 ± 0.41	40.68 ± 0.34	20.67 ± 0.15	57.97 ± 0.10

When evaluated on the independent test set, the results in Table 7 confirm that both the deterministic and stochastic GBDT models outperform their baselines. For active frame-stacks, the deterministic model reduces RMSE by 10.29 minutes and MAE by 8.14 minutes relative to the mean-duration baseline, while the stochastic model reduces RMSE by 7.67 minutes and MAE by 6.12 minutes. Improvements are also observed for inactive frame-stacks, though with smaller margins. The statistical comparison in Table 8 confirms that these gains are highly significant across all metrics, with p-values below 0.001.

Figure 6 and Figure 7 show how predictive performance evolves across running completion for active frame-stacks. In Figure 6, RMSE and MAE generally decline as completion increases - consistent with the fact that the remaining time shrinks and the model benefits from more informative signals later in the process. By contrast, MAPE rises sharply near full completion because the true remaining time (the denominator) approaches zero. Figure 7 indicates that the stochastic model improves both mean accuracy and uncertainty quantification with increasing completion, i.e., predictive uncertainty shrinks and the model's mean estimates become more reliable near full completion. For the same reason, this is expected: as the remaining time shrinks and the signal becomes richer, absolute errors and predictive uncertainty decline.

Table 7. Predictive performance metrics from both the deterministic and stochastic GBDT model compared against the baseline model evaluated on the test dataset. Note that the deterministic baseline model is a collective mean on the training dataset for the inactive frame-stacks, whereas the stochastic baseline fits a LogNormal distribution to the same target values. Also, the GBDT model for active frame-stacks is only evaluated here for frame-stacks with zero running completion. Values are reported as mean ± standard deviation over the number of instances in the test dataset (*n* = 2,450).

Model type	RMSE (min)	MAE (min)	MAPE (%)	Residual variance (min^2)	CRPS (min)	Training time (sec)	Prediction time (sec)
Deterministic model							
GBDT model (active frame-stacks)	39.09 ±49.28	30.51 ±24.44	15.10 ±14.14	1526	–	14.03	0.10 ±0.05
GBDT model (inactive frame-stacks)	46.58 ±61.55	35.50 ±30.16	18.14 ±20.17	2170	–	0.8800	0.10 ±0.06
Baseline (mean duration)	49.38 ±60.74	38.65 ±30.74	19.91 ±19.76	2438	–	–	–
Stochastic model							
GBDT model (active frame-stacks)	41.71 ±53.72	32.53 ±26.12	16.21 ±15.88	1737	52.14 ±17.58	3911	0.035 ±0.021
GBDT model (inactive frame-stacks)	46.39 ±61.32	35.13 ±30.30	17.92 ±20.05	2158	54.75 ±21.34	41.26	0.031 ±0.028
Baseline (LogNormal distribution)	49.38 ±60.74	38.65 ±30.74	19.87 ±19.66	2438	69.25 ±44.08	–	–

Table 8. Paired statistical analysis of the predictive performance metrics from Table 7 for the deterministic and stochastic GBDT models compared against the baseline. Note that Δ is the difference between the mean of the respective model and baseline performance.

Model type	RMSE (min)		MAE (min)		MAPE (%)		CRPS (min)	
	Δ	*p-value*	Δ	*p-value*	Δ	*p-value*	Δ	*p-value*
Deterministic model								
GBDT model (active frame-stacks)	-10.29	0.0000	-8.140	0.0000	-4.810	0.0000	–	–
GBDT model (inactive frame-stacks)	-2.800	0.0000	-3.150	0.0000	-1.770	0.0000	–	–
Stochastic model								
GBDT model (active frame-stacks)	-7.670	0.0000	-6.120	0.0000	-3.660	0.0000	-17.11	0.0000
GBDT model (inactive frame-stacks)	-2.990	0.0000	-3.520	0.0000	-1.950	0.0000	-14.50	0.0000

The learning dynamics and residuals of the deterministic models are summarized in Figure 8. Panels (a) and (b) display the training and validation RMSE curves across boosting iterations for the active and inactive models, respectively. In both cases, the training error decreases steadily, confirming that the models continue to learn useful patterns. For the active model in Panel (a), the validation RMSE closely tracks the training curve, showing no divergence and indicating good generalization without overfitting. By contrast, the inactive model in Panel (b) exhibits a widening gap between training and validation performance: while the training error keeps improving, the validation error quickly plateaus. This pattern reflects overfitting and highlights the limited predictive power of the inactive feature set. Importantly, the inactive model was trained on roughly forty times less data than the active model, which further constrains its ability to generalize - small datasets are more prone to variance and cannot capture the full variability of the process. Hence, the weaker performance of the inactive model arises from a combination of insufficient feature signal and restricted data availability. This explains the superior performance of the active GBDT model compared to the inactive one, as reported in Table 7 and Table 8. Panel (c) plots the residuals between predicted and true remaining picking times across the target range. Both models show residuals distributed symmetrically around zero, indicating no systematic bias. Nevertheless, a downward tendency is visible: large remaining times are often underestimated, while small remaining times are slightly overestimated. This regression-to-the-mean effect arises because GBDT models trained with squared-error loss, combined with a skewed target distribution, tend to pull predictions toward average values [33]. Taken together, these results demonstrate that the active GBDT model leverages informative progress features and sufficient training data to achieve strong generalization, while the inactive model suffers from both limited explanatory power and reduced sample size. Moreover, despite some regression-to-the-mean effects at the extremes, the models remain well-calibrated overall, establishing a solid foundation for integrating predictive outputs into the subsequent scheduling framework.

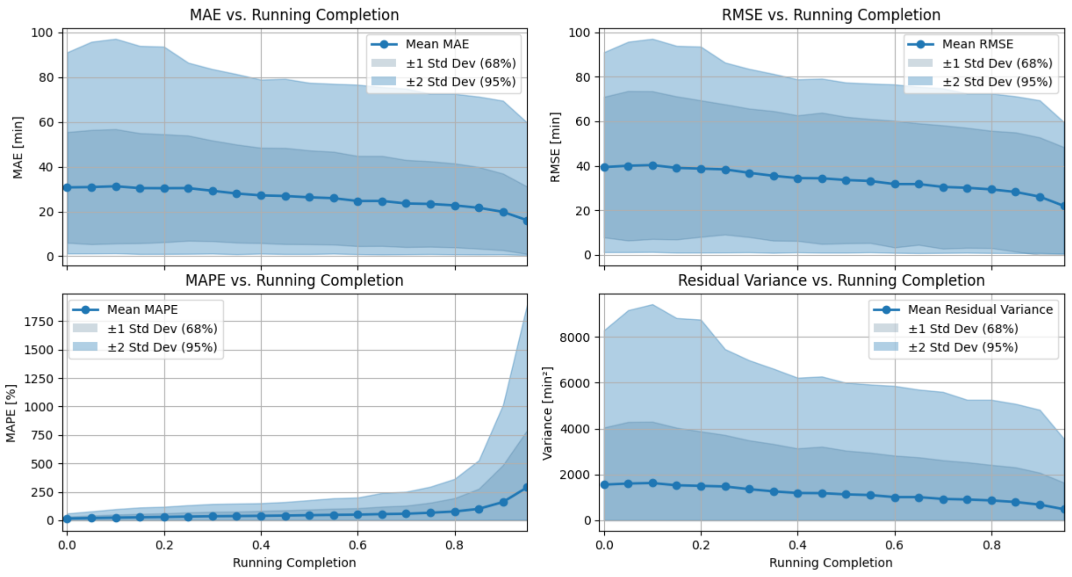


Figure 6. Performance of the deterministic GBDT model across running completion levels for active frame-stacks. The bottom-left graph shows a sharp rise in MAPE near full completion due to the denominator (remaining picking time) approaching zero. Shaded areas indicate 68% and 95% confidence intervals around the mean.

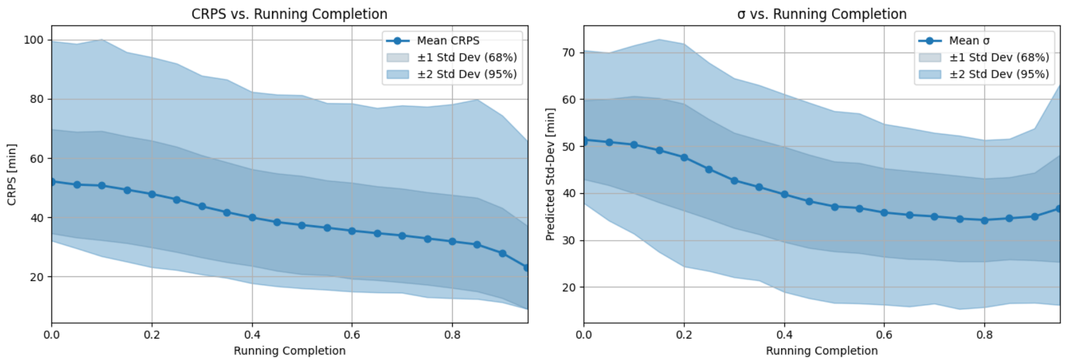
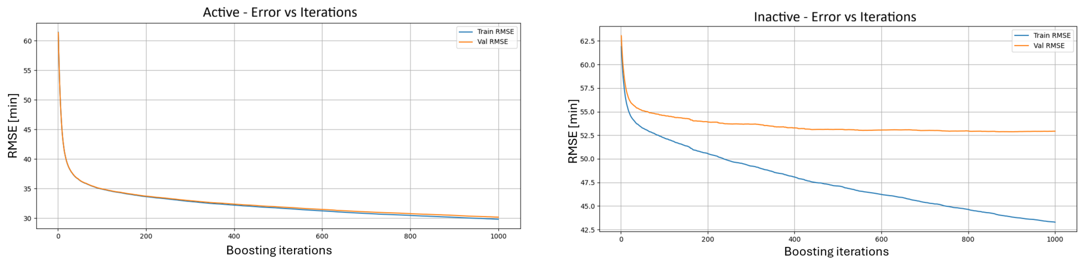
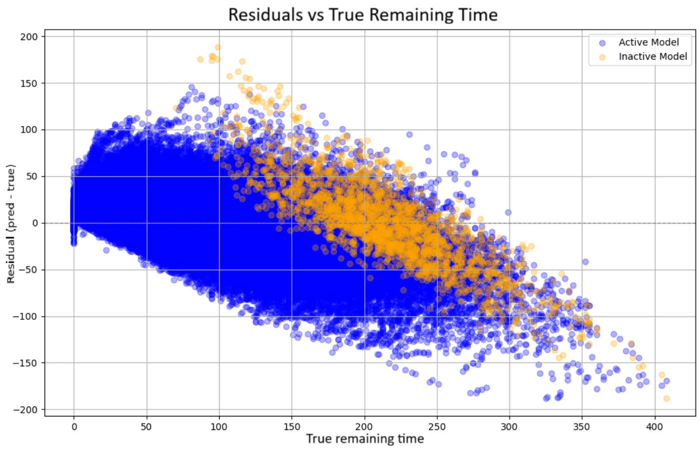


Figure 7. Predictive performance (CRPS) and predicted standard deviation σ of the stochastic GBDT model across different levels of running completion for active frame-stacks. Shaded areas indicate 68% and 95% confidence intervals around the mean.



(a) Active model.

(b) Inactive model.

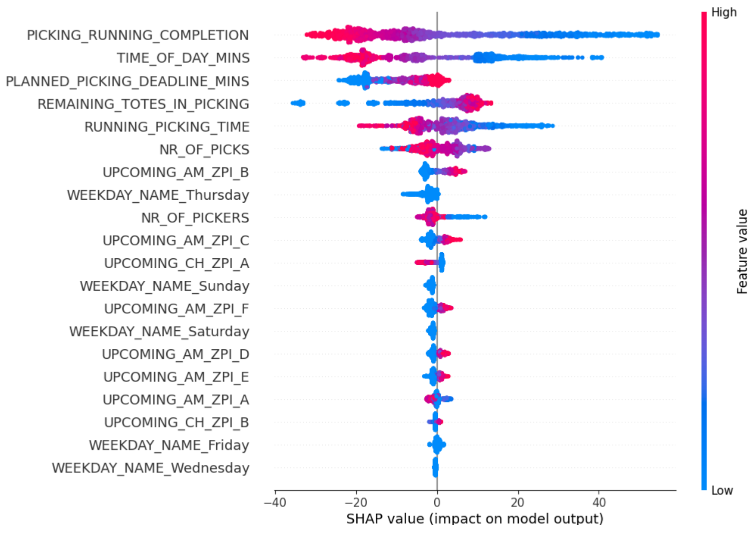


(c) Residuals between predicted and true remaining picking times across the target range for both the active (blue) and inactive (yellow) frame-stacks.

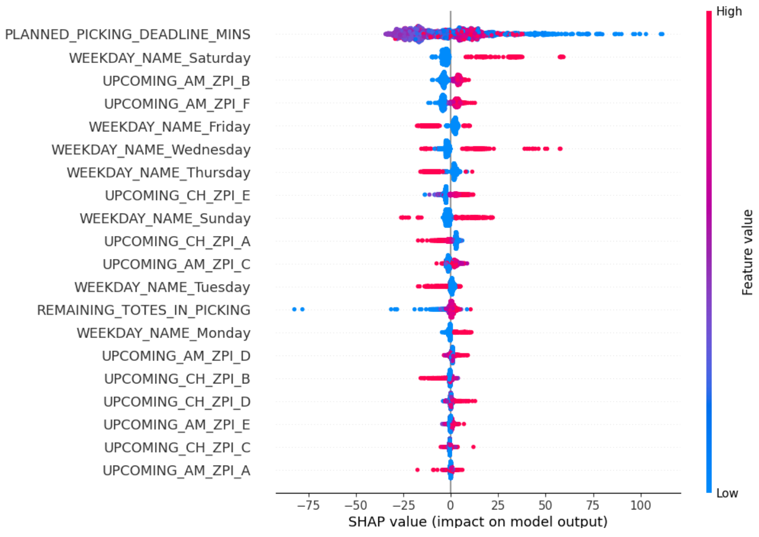
Figure 8. (a) Training (1,328,864 datapoints) and validation (77,039 datapoints) RMSE across boosting iterations for the active model, (b) training (77,039 datapoints) and validation (19,260 datapoints) RMSE across boosting iterations for the inactive model, and (c) residuals between predicted and true remaining picking times across the target range for the deterministic model.

4.2.2 Feature importance

Figure 9 summarizes the relative importance of input features for the deterministic GBDT models using SHAP values. The active model in Panel (a) is strongly driven by progress-related features, in particular PICKING_RUNNING_COMPLETION, TIME_OF_DAY_MINS, PLANNED_PICKING_DEADLINE_MINS, and REMAINING_TOTES_IN_PICKZONE. These features directly capture how far the picking process has advanced and how much work remains, which explains their dominant predictive impact. High completion values (shown in red) are associated with lower predicted remaining times, while low values (blue) correspond to longer expected durations, which aligns with operational intuition.



(a) SHAP summary plot for the active GBDT model.



(b) SHAP summary plot for the inactive GBDT model.

Figure 9. SHAP summary plots showing feature importance and impact on predicted picking time for both the active (a) and inactive (b) GBDT models. Each point represents a frame-stack, with color indicating the feature value (red = high, blue = low). Note that only the twenty most important features are included. Also, every feature that starts with WEEKDAY_NAME_ or UPCOMING_ belongs to the the weekday and upcoming area features that are interpreted like this by the model due to the application of one-hot encoding.

The inactive model in Panel (b), which excludes progress features, instead relies on contextual variables such as `PLANNED_PICKING_DEADLINE_MINS`, weekday indicators, and upcoming wave-related features. This shift illustrates that in the absence of direct signals of progress, the model resorts to weaker proxies such as temporal and planning information. While these features carry some explanatory power, their influence is less direct, leading to the poorer predictive accuracy observed for the inactive variant.

Overall, these results highlight the critical role of progress-related information in predicting remaining picking times even more. When such features are included, the model learns relationships that are both operationally interpretable and quantitatively predictive, whereas their exclusion forces reliance on indirect signals and degrades performance.

4.3 Scheduler results

4.3.1 Performance

The performance results for the adaptive schedulers are summarized in Table 9. A top-down perspective reveals three key findings. First, both adaptive schedulers improve over the EDD–FS heuristic across nearly all KPIs. Second, the scenario-based variant (at K=80, meaning 80 scenarios included) delivers the strongest and most consistent gains. Third, the scenario-based variant offers additional protection against extreme outcomes, reflected in significant lower delays.

On the scheduling objective (larger is better, indicating higher cumulative weighted slack), the deterministic scheduler achieves a significant improvement over the heuristic baseline (+1.272), while the scenario-based scheduler with $K = 80$ yields an even stronger significant gain (+2.256). These values give an indication on the performance of the rest of the metrics.

In terms of flow efficiency, all adaptive schedulers reduce average makespan per shipment compared to the heuristic. The deterministic scheduler achieves the largest and only significant decrease of 8.221 minutes (-4.0%). Although the reductions for the scenario-based models are not statistically significant, the results indicate that incorporating uncertainty does not undermine overall throughput, and that efficiency gains remain consistent across variants.

With respect to timeliness, differences between the adaptive approaches become more pronounced. The deterministic scheduler reduces total delay substantially by 229.3 minutes (-16.5%) and cuts the number of delayed frame-stacks by 5.0 (-8.5%) on average. The $K = 20$ scenario-based scheduler achieves moderate but non-significant improvements, while the $K = 80$ variant delivers the strongest effect in terms of total delay, cutting it by 357 minutes (-25.6%) while its effect on the delay count is not significant. Moreover, the scenario-based scheduler markedly lowers the maximum delay significantly by 13.6 minutes (-16.6%) and 31 minutes (-37.9%) when using 20 and 80 scenarios respectively, outperforming the deterministic scheduler. This demonstrates that increasing the number of scenarios strengthens the model's ability to hedge against extreme outcomes.

Finally, hindsight "upper bound" benchmarks illustrate the gap to perfect foresight. Objective values reach around 10, total delay falls to roughly 300–370 minutes, and delay counts drop to about 16–18 on average. These values set an unattainable ceiling in practice but highlight that the $K = 80$ scenario-based scheduler closes the gap most effectively, especially on timeliness.

Taken together, the results indicate that adaptive scheduling substantially improves operational performance compared to the heuristic baseline. The deterministic scheduler offers strong and consistent gains, while the scenario-based approach with $K = 20$ provides intermediate results. Crucially, increasing the number of scenarios to 80 combines the benefits of robustness with superior performance across timeliness metrics, demonstrating that scenario-based modeling with sufficient scenario richness can outperform deterministic scheduling not only in mitigating tail risks but also in

Table 9. Merged evaluation and significance summary. Each cell shows the mean ± standard deviation. For the two implemented schedulers, the line below reports the mean difference versus the EDD-FS benchmark (Δ; percentage change for all but the objective), with significance stars from a two-sided Welch's t-test across replications ($n = 10$): * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$. Results without stars are not statistically significant.

Simulation type	Objective	Avg. makespan (min/shipm.)	Total delay (min)	Delay count	Maximum delay (min)
EDD-FS heuristic (benchmark)	-2.986 ± 1.027	203.3 ± 9.766	1393 ± 188.1	59.10 ± 4.818	81.90 ± 16.85
Deterministic scheduler					
Proposed scheduler	-1.714 ± 0.8155 (+1.272)***	187.0 ± 27.27 (−8.221, −4.0%)**	1164 ± 143.9 (−229.3, −16.5%)***	54.10 ± 5.364 (−5.000, −8.5%)**	79.80 ± 13.60 (−2.100, −2.6%)
Upper bound benchmark	10.30 ± 1.437	206.0 ± 2.238	335.9 ± 90.24	16.40 ± 2.271	73.10 ± 23.76
Scenario-based scheduler (K=20)					
Proposed scheduler	-2.579 ± 0.7063 (+0.4069)	197.9 ± 4.038 (−5.459, −2.7%)	1306 ± 121.9 (−87.20, −6.3%)	56.70 ± 2.541 (−2.400, −4.1%)	68.30 ± 13.53 (−13.60, −16.6%)*
Upper bound benchmark	9.666 ± 0.9507	208.3 ± 6.978	369.4 ± 128.7	18.80 ± 5.245	59.20 ± 28.39
Scenario-based scheduler (K=80)					
Proposed scheduler	-0.7299 ± 0.3463 (+2.256)***	199.8 ± 2.994 (−3.500, −1.7%)	1036 ± 50.80 (−357.0, −25.6%)***	57.30 ± 2.791 (−1.800, −3.0%)	50.90 ± 5.043 (−31.00, −37.9%)***
Upper bound benchmark	11.24 ± 0.9835	203.3 ± 4.389	306.4 ± 53.89	15.83 ± 4.275	47.29 ± 13.28

improving average outcomes.

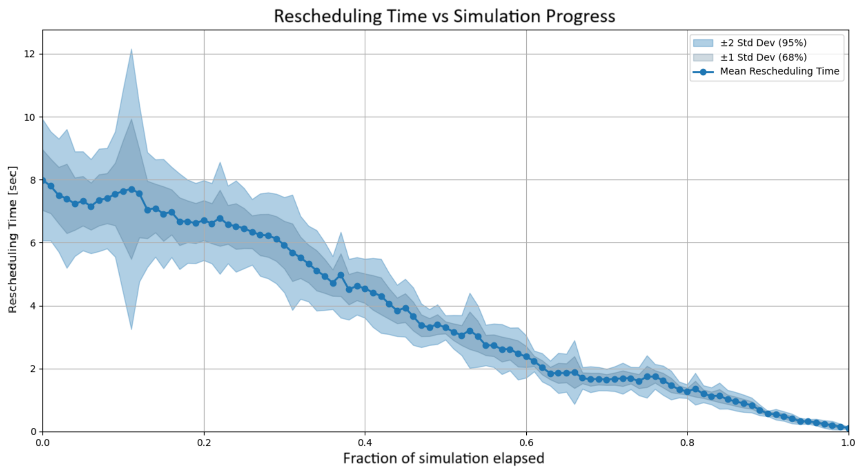
4.3.2 Computational reliability

To be operationally viable, the schedulers must re-optimize reliably within each trigger period. Figure 10 shows the rescheduling times for both variants over the course of the simulation.

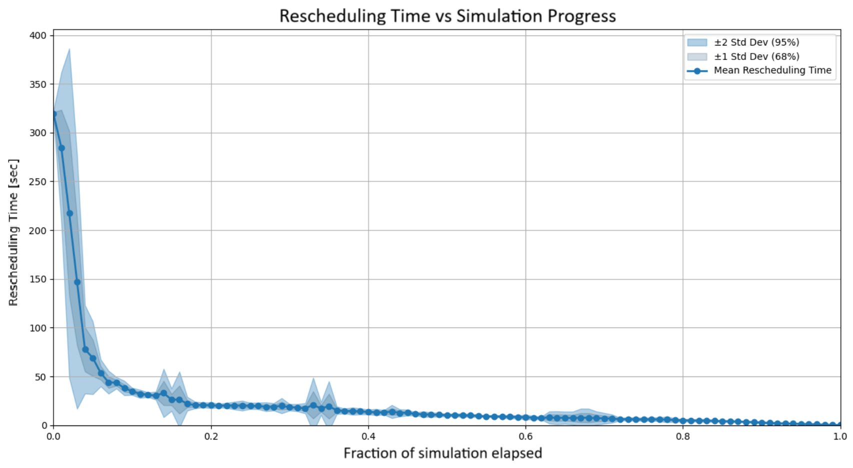
For the deterministic scheduler, computation is consistently fast, averaging 1–8 seconds during most of the shift. Times peak at around 10–12 seconds early in the day when many frame-stacks are still unscheduled, but steadily decline toward zero as the horizon shrinks. The narrow confidence intervals indicate stable performance across replications, with runtimes decreasing in an almost linear fashion as the number of tasks reduces step by step.

The scenario-based scheduler requires substantially more time at the start of the shift - up to 250–300 seconds - because the optimization must evaluate multiple scenarios (20 in this case) for every frame-stack. After the first few triggers, however, runtimes drop sharply. An explanation of this steeper, negative-exponential decline could be that fixing a single start time simultaneously eliminates large blocks of constraints across all scenarios, while shrinking start windows render many of the remaining constraints redundant. Beyond this structural effect, solver behavior also plays a role: the scenario-based model starts out so large that Gurobi operates in a *slow regime*, where presolve, LP relaxations, and branching scale poorly. Once the first few fixes reduce the model below this critical threshold, Gurobi can prune aggressively, causing runtimes to collapse. The deterministic model, by contrast, never reaches such extreme problem sizes, so its runtime decreases more smoothly and predictably, resulting in a near-linear decline.

In summary, both schedulers are computationally tractable for real-time use. The deterministic variant is consistently fast and predictable, while the scenario-based variant starts with a heavier computational burden but quickly converges to similar runtimes as the shift progresses. If shorter trigger



(a) Deterministic scheduler.



(b) Scenario-based scheduler.

Figure 10. Rescheduling time profile (in seconds) expressed as a function of simulation elapsed. Shaded areas indicate 68% and 95% confidence intervals around the mean.

periods were needed in practice, this could be supported by using a rolling time horizon that restricts optimization to tasks within a cutoff window, keeping problem sizes bounded and rescheduling times within operational limits.

4.3.3 Schedule representation

The deterministic scheduler's output can be illustrated through the example in Figure 11. Each horizontal bar represents the picking duration of a frame-stack, with colors grouping frame-stacks that belong to the same shipment. The visualization shows how the scheduler distributes workload across the day while balancing shipment deadlines and capacity constraints. Aligned start times indicate points where multiple frame-stacks are launched together within the allowed start-gap, whereas staggered sequences reflect the prioritization of short-horizon shipments. This highlights how the model's objective of maximizing weighted slack translates into a concrete picking plan that structures both workload distribution and shipment timing.

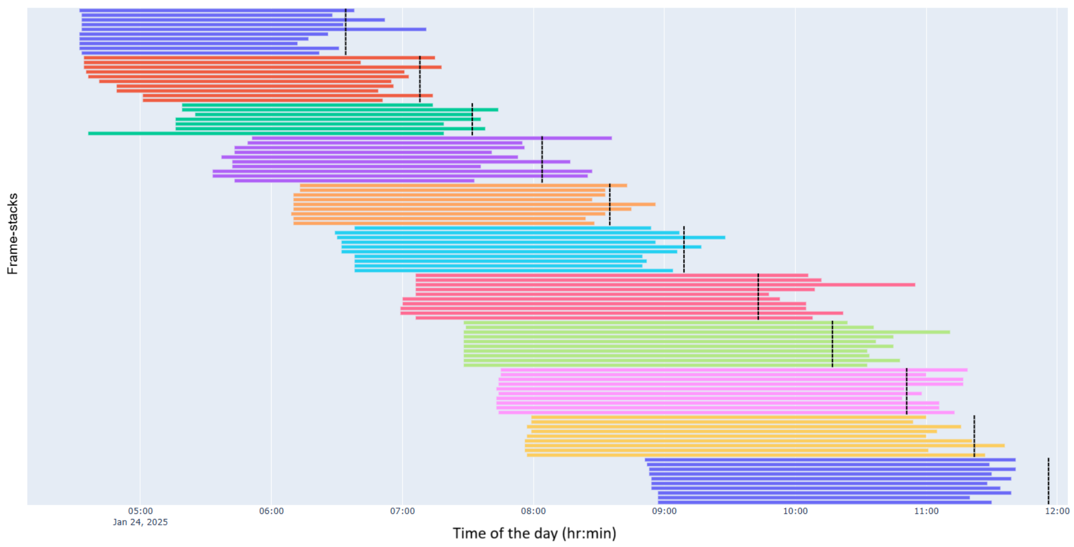


Figure 11. Actual picking process in the simulation when the adaptive schedule is generated by the deterministic model. Each horizontal bar represents a frame-stack's picking time from start to finish; colors denote frame-stacks from the same shipment with their respective deadline visualized by a black vertical line.

4.4 Sensitivity analysis

Beyond average performance, the practical value of a scheduling approach depends on its robustness under different parameter choices. Small changes in trigger period, priority weighting, or the number of sampled scenarios may shift the balance between efficiency, timeliness, and computational effort. To explore these trade-offs, the sensitivity analysis varies three levers: trigger period, EDD–FS priority weight, and, for the scenario-based scheduler, scenario count.

4.4.1 Effect of scheduler parameters on performance

The three tuning levers affect different KPIs and by different magnitudes, with details as shown in Appendix 1. Trigger period mainly shows up in average makespan and total delay. For the deterministic scheduler a shorter period improves makespan, with a significant difference between 3 and 6 minutes (p-adj = 0.010). For the scenario-based scheduler the 6 vs 12 minute contrast is also significant for makespan (p-adj = 0.014). Effects on total delay move in the expected direction but are not significant for the reported contrasts. The EDD–FS priority weight shifts the objective, makespan, and total delay. For the deterministic scheduler the improvements when moving from 1.0 to 5.0 are modest but relatively significant for the objective (p-adj = 0.092) and total delay (p-adj = 0.052), while effects for the scenario-based scheduler remain mild and not significant. Scenario count (scenario-based only) has the strongest and most consistent effect on maximum delay, with significant reductions as the count increases (for example 1 vs 40, p-adj = 0.0285; 1 vs 60, p-adj = 0.0018; 1 vs 80, p-adj < 0.001). At higher counts total delay also improves, with several significant contrasts (for example 3 vs 60, p-adj = 0.0041; 20 vs 60, p-adj = 0.0004; 20 vs 80, p-adj = 0.0001; 40 vs 80, p-adj = 0.0005). Effects on makespan are small and mostly not significant. Overall, trigger frequency changes primarily affect general flow metrics, weight tuning trades early slack against premature starts with modest statistical signals, and adding scenarios mainly trims tail risk and, once sufficiently rich, reduces overall delay.

4.4.2 Managerial insights

This subsection translates the empirical findings into actionable guidance for planners. Rather than focusing on statistical significance, we summarize how each tuning lever should be used under real operational constraints - balancing punctuality, flow efficiency, and compute budget. We first discuss the trigger period as the cadence that keeps plans aligned with reality, then the EDD–FS priority weight as the knob that trades early slack creation against premature starts, and finally the scenario count as the driver of risk awareness versus runtime. The aim is to provide simple rules of thumb and default settings that are robust across days, with clear signals for when to deviate.

As shown in Figure 12, the deterministic scheduler remains relatively stable across trigger periods, with only minor changes in objective and makespan. By contrast, the scenario-based scheduler exhibits a clear deterioration when the trigger period increases: objective values decline, average makespan lengthens, and total delay rises. This behavior can be attributed to the high variability of distributional predictions, which require frequent re-optimization to prevent drift. Maximum delay is largely unaffected for both schedulers. Overall, the deterministic scheduler tolerates a wider range of trigger periods, whereas the scenario-based scheduler benefits from shorter, more frequent triggers to control variability. Therefore, the choice of trigger period should reflect the inherent trade-off between the two approaches: the deterministic scheduler maintains strong performance even at moderate periods, while the scenario-based scheduler requires shorter periods to offset its higher predictive variability and only then approaches the efficiency levels of the deterministic variant.

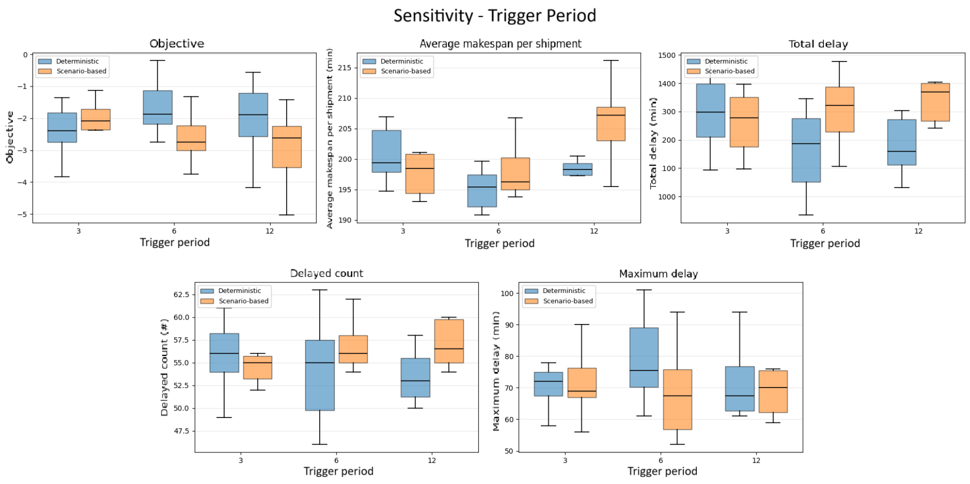


Figure 12. Impact of trigger period (3, 6 and 12 minutes) on deterministic and scenario-based scheduling performance in boxplot format with the median, interquartile range, and whiskers visible.

The results in Figure 13 show that the deterministic scheduler achieves an optimum at intermediate weight values, with both lower and higher extremes reducing performance by either postponing urgent tasks too much or forcing inefficient early starts. In contrast, the scenario-based scheduler reacts more mildly: its objective decreases slightly, while makespan and total delay improve gradually as the weight increases, making its behavior more similar to the deterministic variant. Delay count remains largely unaffected for both schedulers, indicating that the main effect of the weight lies in shifting when delays occur rather than how many. Maximum delay is minimized for the scenario-based scheduler at intermediate weights, while the deterministic model remains comparatively flat. Overall, the EDD–FS priority weight should be tuned to the planning objective: moderate weights provide the best balance for the deterministic scheduler, whereas higher weights help the scenario-based scheduler hedge against late departures and reduce its performance gap towards the deterministic variant.

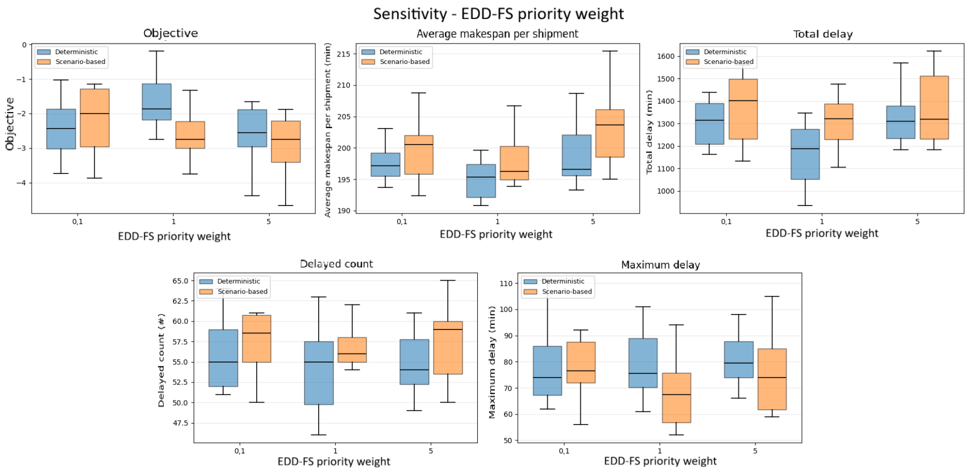


Figure 13. Impact of the EDD-FS priority weight (0.1, 1.0 and 5.0) of the scheduler on deterministic and scenario-based scheduling performance in boxplot format with the median, interquartile range, and whiskers visible.

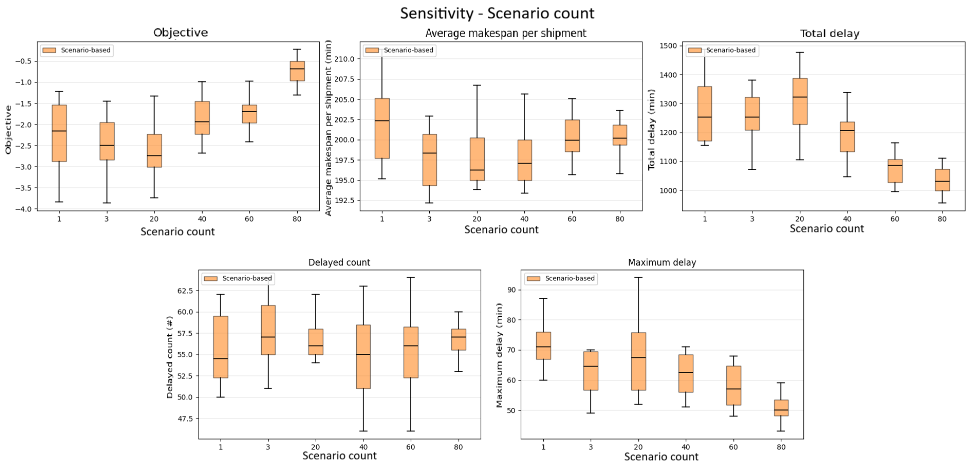


Figure 14. Impact of the scenario count (3, 20, 40, 60, and 80) of the scheduler on scenario-based scheduling performance in boxplot format with the median, interquartile range, and whiskers visible.

As shown in Figure 14, increasing the number of sampled scenarios mainly improves the scheduler's ability to hedge against risk rather than changing average flow. Objective values become less negative, while total delay and maximum delay clearly decrease once the scenario set is large enough to capture the tails of the predictive distribution, with a noticeable improvement around a scenario count of 40. At lower counts such as 1, 3 or 20, the scheduler under-samples possible outcomes, resulting in higher variability and worse median values. When moving to 40, 60 and 80 scenarios, the distributions tighten and median performance improves, although the gains diminish between 60 and 80. In practice, a scenario count of 40 to 60 provides a balanced trade-off: sufficient coverage of uncertainty to reduce extreme delays, without incurring disproportionate computational costs.

5. Conclusion

This study addressed the increasing societal need for adaptive and reliable scheduling methods in e-grocery FCs. As demand for online groceries continues to rise, operational efficiency and robustness against variability in task durations become critical to meeting customer expectations while maintaining cost-effectiveness. Existing scheduling approaches - such as the widely used EDD-FS heuristic - lack the predictive and adaptive capabilities required in such stochastic, high-throughput environments.

To fill this methodological gap, this study proposed a unified framework combining predictive analytics with optimization, extended into a stochastic scenario-based variant. Specifically, GBDT models were developed to predict picking durations and integrated into MILP schedulers. This hybrid approach enables real-time adaptive scheduling that anticipates uncertainty rather than merely reacting to it.

The results demonstrate the value of this framework. The GBDT models significantly outperformed baseline predictors, with active frame-stacks achieving strong accuracy improvements due to the inclusion of progress-related features. The deterministic scheduler reduced total delay by 16% and the number of delayed frame-stacks by 8.5%. Sensitivity analysis further showed that its performance depends on the balance of the EDD-FS weight, highlighting the need to calibrate between not excessively advancing late tasks and preserving algorithm-driven frame-stack priorities.

The scenario-based scheduler extended these benefits by explicitly accounting for uncertainty. Depending on the number of scenarios, total delay decreased by 6.3% to 25.6% and maximum delay dropped by 16.6% to 37.9%. Higher scenario counts consistently yielded better performance, though at the cost of increased computational load. This illustrates the trade-off between robustness and tractability in stochastic optimization. Importantly, the scenario-based variant with sufficient scenario richness not only mitigated extreme delays but also improved average outcomes beyond the deterministic scheduler.

While the presented framework demonstrates clear improvements in predictive accuracy and scheduling performance, several limitations must be acknowledged. First, the evaluation was conducted exclusively in a simulation environment rather than in live operations, which may not fully capture real-world disturbances such as picker variability, stockouts, or equipment failures. Second, the scope of the scheduling model was restricted to the picking stage and did not account for upstream or downstream processes, limiting its holistic applicability. Finally, the GBDT model was trained on historical data from a specific period; if operational processes or workforce behavior change over time, the predictive accuracy may degrade, leading to a mismatch between the model's assumptions and future reality.

Future work should focus on validating these findings through real-world trials, where the proposed algorithm is expected to perform even better, as the GBDT model was trained on actual operational rather than simulated data. In addition, robustness should be assessed under other operational perturbations—such as shipment delays and cancellations—whose effects were not captured in this study. Methodological extensions could involve improving the predictive accuracy of the inactive-frame model, implementing rolling-horizon strategies to reduce the solution space, and exploring metaheuristics to accelerate optimization. Finally, reinforcement learning could be investigated as an alternative paradigm for adaptive decision-making under uncertainty, leveraging data-driven policies that continuously improve through experience.

In conclusion, this study demonstrates that integrating predictive models with deterministic and scenario-based optimization methods can substantially improve the timeliness and reliability of e-grocery fulfillment schedules. By bridging predictive analytics with optimization and extending into scenario-based scheduling, this study introduces a novel methodological framework that is both academically significant and directly applicable to the rapidly growing e-grocery sector.

References

[1] Nils Boysen, René de Koster, and Felix Weidinger. “Warehousing in the e-commerce era: A survey”. In: *European Journal of Operational Research* 277.2 (2019), pp. 396–411. DOI: 10.1016/j.ejor.2018.08.023.

[2] Miguel Rodríguez García, Iria González Romero, Ángel Ortiz Bas, and J. Carlos Prado-Prado. “E-grocery retailing: From value proposition to logistics strategy”. In: *International Journal of Logistics Research and Applications* 25.10 (2022), pp. 1381–1400. DOI: 10.1080/13675567.2021.1900086.

[3] Prita Meilanitasari and Seung-Jun Shin. “A Review of Prediction and Optimization for Sequence-Driven Scheduling in Job Shop Flexible Manufacturing Systems”. In: *Processes* 9.8 (2021). DOI: 10.3390/pr9081391.

[4] N. Raman, F. B. Talbot, and R. V. Rachamadugu. “Due Date Based Scheduling in a General Flexible Manufacturing System”. In: *Journal of Operations Management* 8.2 (1989), pp. 115–127. URL: https://deepblue.lib.umich.edu/bitstream/handle/2027.42/146830/joom115.pdf.

[5] Dimitrios Mourtzis, Georgios Michalos, and Ioannis Lekakos. “Advances in Adaptive Scheduling in Industry 4.0”. In: *Frontiers in Manufacturing Technology* 2 (2022). DOI: 10.3389/fmtec.2022.937889.

[6] Willy Herroelen and Roel Leus. “Resource-Constrained Project Scheduling: A Survey of Recent Developments”. In: *International Journal of Production Research* 45.6 (2007), pp. 1073–1106.

[7] Guilherme E.P. Vieira, Jeffrey W. Herrmann, and Enver Lin. “Rescheduling manufacturing systems: A framework of strategies, policies, and methods”. In: *Journal of Scheduling* 6.1 (2003), pp. 39–62. DOI: 10.1023/A:1022235519958.

[8] Gyanesh Raj, Debjit Roy, Rene de Koster, and Vishal Bansal. “Stochastic modeling of integrated order fulfillment processes with delivery time promise: Order picking, batching, and last-mile delivery”. In: *European Journal of Operational Research* 316.3 (2024), pp. 1114–1128. DOI: 10.1016/j.ejor.2024.03.003.

[9] Ming Zhang, Yang Lu, Youxi Hu, Nasser Amaitik, and Yuchun Xu. “Dynamic Scheduling Method for Job-Shop Manufacturing Systems by Deep Reinforcement Learning with Proximal Policy Optimization”. In: *Sustainability* 14.9 (2022). DOI: 10.3390/su14095177.

[10] Tao Zhang, Tianzuo Zhao, Hongyan Chu, Congbin Yang, Yueze Zhang, and Jun Yan. “A Dynamic Scheduling Method Combining Iterative Optimization and Deep Reinforcement Learning to Solve Sudden Disturbance Events in a Flexible Manufacturing Process”. In: *Mathematics* 13.1 (Dec. 2024), p. 4. DOI: 10.3390/math13010004.

[11] A. Gharbi and J. Haddad. “Dynamic scheduling in flexible manufacturing systems: A review”. In: *International Journal of Production Research* 57.12 (2019), pp. 3742–3766. DOI: 10.1080/00207543.2018.1519260.

[12] Duygu Ünal and Begüm Kaçar. “A reactive scheduling approach for the stochastic job shop problem”. In: *Computers & Industrial Engineering* 139 (2020), p. 106179. DOI: 10.1016/j.cie.2019.106179.

[13] David R. Morrison, Sheldon H. Jacobson, Jason J. Sauppe, and Edward C. Sewell. “Branch-and-Bound Algorithms: A Survey of Recent Advances in Searching, Branching, and Pruning”. In: *Discrete Optimization* 19 (2016), pp. 79–102. DOI: 10.1016/j.disopt.2016.01.005.

[14] Willy Herroelen and Roel Leus. “Project Scheduling under Uncertainty: Survey and Research Potentials”. In: *European Journal of Operational Research* 165.2 (2005), pp. 289–306. DOI: 10.1016/j.ejor.2004.04.002.

[15] Marco E L"ubbecke and Jacques Desrosiers. *Selected topics in column generation.* Springer, 2005, pp. 65–138.

[16] Hirochika Yamashiro and Hirofumi Nonaka. "Estimation of processing time using machine learning and real factory data for optimization of parallel machine scheduling problem". In: *Operations Research Perspectives* 8 (2021), p. 100196. DOI: 10.1016/j.orp.2021.100196. URL: https://doi.org/10.1016/j.orp.2021.100196.

[17] S. Hartmann. "A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling". In: *Naval Research Logistics* 45.7 (1998), pp. 733–750.

[18] Paul Shaw. "Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems". In: *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming (CP).* Springer, 1998, pp. 417–431.

[19] David Vázquez-Noguerol, Marta Gené-Gil, and Albert Corominas. "Order Batching and Picker Routing in E-Commerce Warehouses: A Genetic Algorithm Approach". In: *Computers & Industrial Engineering* 165 (2022), p. 107952.

[20] S. Abderrazzak, A. Hamid, and S. Omar. "Adaptive Large Neighborhood Search for the Just-In-Time Job-shop Scheduling Problem". In: *2022 International Conference on Control, Automation and Diagnosis (ICCAD).* Lisbon, Portugal: IEEE, 2022, pp. 1–6. DOI: 10.1109/ICCAD55197.2022.9853973.

[21] Yuanyuan Li, Stefano Carabelli, Edoardo Fadda, Daniele Manerba, Roberto Tadei, and Olivier Terzo. "Machine learning and optimization for production rescheduling in Industry 4.0". In: *The International Journal of Advanced Manufacturing Technology* 110 (2020), pp. 2445–2463. DOI: 10.1007/s00170-020-05850-5.

[22] Julian Senoner, Bernhard Kratzwald, Milan Kuzmanovic, Torbjørn H. Netland, and Stefan Feuerriegel. "Addressing distributional shifts in operations management: The case of order fulfillment in customized production". In: *Production and Operations Management* (2023). arXiv: 2304.11910 [stat.AP]. URL: https://arxiv.org/abs/2304.11910.

[23] Luis F. Escudero, Andres Garin, Maria Teresa Ortuño, Carlos Perea, and Guillermo Perez. "An approach for strategic supply chain planning under uncertainty based on stochastic 0-1 programming". In: *Journal of Global Optimization* 26.1 (2003), pp. 137–157.

[24] Gyanesh Raj, Debjit Roy, René de Koster, and Vishal Bansal. "Stochastic modeling of integrated order fulfillment processes with delivery time promise: Order picking, batching, and last-mile delivery". In: *European Journal of Operational Research* 316.3 (2024), pp. 1114–1128.

[25] Man Yiu Tsang, Karmel S. Shehadeh, Frank E. Curtis, Beth Hochman, and Tricia E. Brentjens. "Stochastic Optimization Approaches for an Operating Room and Anesthesiologist Scheduling Problem". In: *arXiv preprint arXiv:2204.11374* (2022).

[26] R. K. Cecen and F. A. Çetek. "A stochastic programming model for the aircraft sequencing and scheduling problem considering flight duration uncertainties". In: *The Aeronautical Journal* 126.1304 (2022), pp. 1736–1751. DOI: 10.1017/aer.2022.17.

[27] José Niño-Mora. "Stochastic Scheduling". In: *Encyclopedia of Optimization.* Ed. by Christodoulos A. Floudas and Panos M. Pardalos. Springer, 2008, pp. 3818–3824. DOI: 10.1007/978-0-387-74759-0_665.

[28] Mark Frye, Dávid Gyulai, József Bergmann, and Robert H. Schmitt. "Adaptive Scheduling through Machine Learning-based Process Parameter Prediction". In: *MM Science Journal, Special Issue on HSM2019 – 15th International Conference on High Speed Machining.* MM Science Journal. 2019, pp. 3060–3066. DOI: 10.17973/MMSJ.2019_11_2019051.

[29] Tianqi Chen and Carlos Guestrin. "XGBoost: A scalable tree boosting system". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM. 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.

[30] Yutong Yan, Yichuan Zhou, and James Micklewright. “Gradient boosting machines with uncertainty estimation: survey and comparative evaluation”. In: *Machine Learning* 111.10 (2022), pp. 3733–3767. DOI: 10.1007/s10994-022-06182-1.

[31] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth. *CRISP-DM: Step-by-step data mining guide*. https://www.scirp.org/reference/referencespapers?referenceid=1592779. Accessed: 2025-04-07. 2000.

[32] Haowen Deng, Youyou Zhou, Lin Wang, and Cheng Zhang. “Ensemble learning for the early prediction of neonatal jaundice with genetic features”. In: *BMC Medical Informatics and Decision Making* 21.1 (2021), p. 338. DOI: 10.1186/s12911-021-01701-9. URL: https://doi.org/10.1186/s12911-021-01701-9.

[33] Jerome H. Friedman. “Greedy Function Approximation: A Gradient Boosting Machine”. In: *The Annals of Statistics* 29.5 (2001), pp. 1189–1232. URL: https://www.cse.iitb.ac.in/~soumen/readings/papers/Friedman1999GreedyFuncApprox.pdf.

Appendix 1. Pairwise KPI contrast sensitivity analysis

Table 10. Pairwise contrasts for Trigger period (3.0, 6.0, 12.0), EDD–FS weight (0.1, 1.0, 5.0), and Scenario count (1, 3, 20, 40, 60, 80), shown for deterministic and scenario-based schedulers. $\hat{\Delta}$ is the mean difference (second minus first measurement); p-adj is the multiplicity-adjusted p-value (Sidak correction).

Scheduler	**Factor**	**Contrast**	**Objective**		**Avg. makespan (min/shipm.)**		**Total delay (min)**		**Delay count**		**Max. delay (min)**	
			$\hat{\Delta}$	*p-adj*	$\hat{\Delta}$	*p-adj*	$\hat{\Delta}$	*p-adj*	$\hat{\Delta}$	*p-adj*	$\hat{\Delta}$	*p-adj*
Deterministic	Trigger period	3.0 vs 6.0	-0.6609	0.23	5.6630	0.0100	135.9000	0.1000	1.6000	0.8300	-9.0000	0.2200
		3.0 vs 12.0	-0.3963	0.7400	2.3280	0.5000	87.6000	0.4800	2.4000	0.5500	-0.7000	1.0000
		6.0 vs 12.0	0.2646	0.9000	-3.3350	0.1300	-48.3000	0.8700	0.8000	0.9800	8.3000	0.4000
Deterministic	EDD–FS weight	0.1 vs 1.0	-0.7486	0.1700	2.7240	0.1900	138.7000	0.0700	1.6000	0.8600	-1.1000	1.0000
		0.1 vs 5.0	0.1299	0.9800	-1.0470	0.9400	-16.0000	0.9800	1.1000	0.9200	-1.8000	0.9900
		1.0 vs 5.0	0.8785	0.0920	-3.7710	0.2300	-154.7000	0.0520	-0.5000	0.9900	-0.7000	1.0000
Scenario-based	Trigger period	3.0 vs 6.0	0.4109	0.5400	0.9190	0.9700	-41.0000	0.8200	-1.8000	0.3000	3.2000	0.9200
		3.0 vs 12.0	0.8045	0.2100	-6.7070	0.0650	-100.2000	0.4300	-2.9000	0.2200	-2.2000	0.9800
		6.0 vs 12.0	0.3936	0.7400	-7.6260	0.0140	-59.2000	0.8100	-1.1000	0.8700	-5.4000	0.8000
Scenario-based	EDD–FS weight	0.1 vs 1.0	1.3020	0.4300	1.6710	0.8100	59.7000	0.7300	0.6000	0.9700	9.1000	0.3300
		0.1 vs 5.0	1.5920	0.2800	-3.6370	0.4200	-8.2000	1.0000	-0.3000	1.0000	0.5000	1.0000
		1.0 vs 5.0	0.2903	0.8100	-5.3080	0.1000	-67.9000	0.6900	-0.9000	0.9400	-8.6000	0.5300
Scenario-based	Scenario count	1 vs 3	-0.1078	0.9886	-4.5100	0.1369	81.7730	0.8909	2.0000	0.6865	-2.8000	0.9644
		1 vs 20	-0.2590	0.8668	-4.2180	0.1790	121.6730	0.7157	1.3000	0.8358	-3.6000	0.8566
		1 vs 40	0.4691	0.4716	-4.2000	0.1835	13.6730	0.9993	-0.6000	0.9916	-9.9000	0.0285
		1 vs 60	0.6358	0.1979	-1.5660	0.8163	-107.3270	0.7673	0.0000	1.0000	-14.0000	0.0018
		1 vs 80	1.5899	0.0009	-2.3070	0.5860	-148.0270	0.5580	1.9000	0.6380	-21.0000	0.0000
		3 vs 20	0.1512	0.9548	-0.2920	0.9979	-39.9000	0.8707	0.7000	0.9574	0.8000	0.9994
		3 vs 40	0.0050	1.0000	-0.3100	0.9976	-46.1000	0.8417	2.6000	0.5772	-7.9000	0.6133
		3 vs 60	0.7436	0.0446	2.9440	0.2137	-189.1000	0.0041	-2.0000	0.7609	-11.2000	0.2856
		3 vs 80	1.6977	0.0001	2.2030	0.4480	-229.8000	0.0009	-0.1000	0.9999	-18.2000	0.0415
		20 vs 40	-0.1462	0.9816	-0.0180	1.0000	-6.2000	0.9994	1.9000	0.7185	-8.7000	0.3495
		20 vs 60	0.8947	0.0121	2.6520	0.2990	-229.0000	0.0004	-1.3000	0.8908	-10.4000	0.1460
		20 vs 80	1.8489	0.0000	1.9110	0.5715	-269.7000	0.0001	0.6000	0.9457	-17.4000	0.0081
		40 vs 60	0.1667	0.8603	2.6340	0.3092	-121.0000	0.0073	0.6000	0.9936	-4.1000	0.5550
		40 vs 80	1.1208	0.0003	1.8930	0.5833	-161.7000	0.0005	2.5000	0.5349	-11.1000	0.0043
		60 vs 80	0.9542	0.0002	-0.7410	0.9274	-40.7000	0.2983	1.9000	0.7390	-7.0000	0.0736