Counterfactual Digital Twin: A Robust Optimization Approach for Automated Warehouses Facing Unknown Shocks

Zhao Fengxia[1,*], Li Hunan[2], Hou Wenyan[1], Qu Xiaoyan[1]

[1]College of Earthquake Engineering and Architectural Safety, University of Emergency Management, Langfang 065201, China
[2]Technical Department, CITIC Press Corporation Limited, Beijing 100020, China

Abstract: Digital twin technology provides decision support by simulating warehouse operations. Traditional approaches, however, rely on historical data to predict the future—once an extreme situation that has never occurred in history emerges (e.g., sudden order surges or simultaneous failures of multiple AGVs), the model fails. This paper proposes a counterfactual digital twin framework that proactively "imagines" various extreme scenarios within the twin model—even combinations that have never occurred in reality—and trains the decision model to learn how to respond in these "virtual extremes." This is analogous to pilots practicing engine failures in simulators rather than waiting to face them during actual flights. A warehouse model is constructed using discrete-event simulation with Python and SimPy. Experimental results show that strategies trained with counterfactual data achieve decision accuracy improved from approximately 50% to over 85%, order completion time reduced by 14%–25%, and performance degradation decreased by 43%–86% when facing unknown shocks. This paper provides a new paradigm for digital twins to transition from "passive prediction" to "proactive preparation".

Keywords: Counterfactual Learning; Digital Twin; Automated Warehouse; Robust Optimization; Decision-making Under Uncertainty

DOI: 10.57237/j.wjmst.2026.02.002

1 Introduction

1.1 Research Background

In automated warehouses, managers make scheduling decisions daily: Should orders be batched and processed together (batch processing), or should each order be processed upon arrival (real-time allocation)? The right choice improves efficiency and reduces energy consumption; the wrong choice leads to congestion, delays, and customer complaints.

Digital twin technology provides a "sandbox" for this problem: build a virtual warehouse on a computer, synchronize real order data, simulate the outcomes of both strategies, and recommend the better one. This sounds perfect, but there is a fatal flaw—the virtual warehouse only predicts the future based on historical data. If something occurs that has never happened in history (e.g., a sudden 5× surge in orders for a particular SKU, or two AGVs failing simultaneously), the virtual warehouse lacks training data for such scenarios, and its recommendations become unreliable.

It is worth noting that digital twin technology encompasses multiple levels. The most fundamental level is "real-time mapping"—synchronizing real warehouse data such as slot occupancy, AGV positions, and order progress to the virtual model, allowing managers to see what is happening in the warehouse, similar to real-time traffic maps. How-

*Corresponding author: Zhao Fengxia, 77428168@qq.com

Received: June 19, 2026; Accepted: July 21, 2026; Published Online: August 5, 2026 http://www.wjmst.net

ever, this is only "seeing the present," still far from "predicting the future." This paper addresses a higher-level capability—how to simulate the future within the twin model, evaluate strategies, and thereby support decision-making. Digital twin technology has been widely adopted in manufacturing and logistics systems [4-6].

1.2 The Nature of the Problem

The academic term for this problem is "out-of-distribution generalization"—in simpler terms: if a scenario was not seen during training, the model cannot handle it during testing.

The conventional solution is to collect more historical data to cover as many situations as possible. However, extreme events are called "extreme" precisely because they occur rarely or never at all. One cannot wait for a warehouse to be destroyed before training the model.

1.3 Our Approach: Counterfactual Learning

We take a different approach—rather than waiting for extreme events to actually occur, we proactively "manufacture" extreme events within the twin model and let the model practice in advance.

For example:

What if order volume suddenly triples?

What if the hottest SKU suddenly accounts for 80% of all orders?

What if two AGVs fail simultaneously?

These situations may never have occurred simultaneously in reality, but within the twin model, we can forcibly combine them to generate a large number of "counterfactual scenarios"—scenarios that "have never actually occurred but are logically possible." We then train a model to learn which strategy to choose under these extreme combinations. By the time the real warehouse encounters a similar shock, the model has already learned to handle such scenarios. Counterfactual learning has been applied in various operations management contexts, such as inventory management [15].

To draw an analogy: pilots cannot wait until an engine actually fails in mid-air to learn how to handle it; they must repeatedly practice engine failures in simulators. Counterfactual twin is the "flight simulator" for warehouse schedulers.

1.4 Main Contributions

First introduction of counterfactual learning into warehouse digital twins, endowing the model with the ability to "imagine extremes."

Design of a counterfactual scenario generation methodology—actively varying key variables such as order volume, sales concentration, and equipment failure rates to create combinations absent from historical data.

Demonstration through simulation experiments that this "advance practice" significantly improves warehouse performance under real shocks.

2 Problem Description

2.1 The Warehouse Model

We study a typical small-to-medium automated warehouse with parameters shown in Table 1.

Table 1 Baseline Warehouse Parameters

Parameter	Value
Racking	10 columns × 5 levels × 2 rows (100 slots total)
Stacker crane	Horizontal speed 2 m/s, vertical speed 1 m/s
AGVs	2 units, speed 1 m/s
Daily operation hours	8 hours
SKU categories	50
Average order arrival rate	10 orders/hour

Managers make one key decision daily: which scheduling strategy to use today?

Strategy A (Batch Processing): Accumulate orders every 15 minutes and release them as a batch.

Strategy B (Real-time Allocation): Process each order immediately upon arrival without waiting.

Different strategies have different advantages under different circumstances. The right choice leads to fast order completion; the wrong choice leads to congestion and delays. Warehouse operation optimization has been extensively studied in the literature [7-9].

2.2 Definition of Counterfactual Scenario

Consider a concrete example:

Fact: Over the past 30 days, a certain SKU averaged 50 orders per day, with a maximum of 80 orders on a single day.

Counterfactual: What if this SKU suddenly receives 300 orders today (nearly 4× the historical maximum)?

The "300 orders" has never appeared in history, but it is entirely logically possible (e.g., driven by a promotional campaign or social media). We do not need to wait for it to actually happen; instead, we directly set the order volume to 300 in the twin model, run the simulation, and observe the results.

This process of "actively changing inputs and observing outcomes" is counterfactual generation. We are not "predicting the future" but "imagining another potential reality."

2.3 Causality: Beyond Simple Correlation

We adopt the structural causal model (SCM) framework [1] to represent causal relationships. For a comprehensive treatment of causal inference, see Pearl [1], Bareinboim and Pearl [2], and Peters et al. [3].

One might ask: why not just train a model on historical data—input order volume and equipment status, output a recommended strategy?

The problem is that certain combinations of variables have never co-occurred in historical data. For example, "high order volume" and "high failure rate" may never have occurred simultaneously, so the model does not know how to handle both at once. It can only guess, and its guessing accuracy is only about 50% (no better than a coin flip).

What we need is causality, not correlation. Causality tells us: if we actively increase order volume while simultaneously setting an AGV to a failed state, how will performance change—even if this combination has never been seen in history?

In a simulation model, because all rules (equipment speeds, failure probabilities, order generation logic) are defined by us, we fully know the effect of "intervening" on any variable. The simulator is inherently a perfect "causal machine."

3 Counterfactual Twin Framework

3.1 Overall Architecture

The figure below illustrates our framework, consisting of five modules connected end-to-end to form a closed loop.

Figure 1 Overall architecture of the counterfactual digital twin framework

The closed loop is embodied in:

1) Real-time data synchronization: The twin model continuously receives order and equipment status data from the real warehouse.
2) Execution feedback: After the twin-recommended strategy is executed in the real warehouse, actual completion times, energy consumption, and other data are transmitted back to the twin model for subsequent training or model calibration.
3) Continuous iteration: After daily operations conclude, newly generated data (including shock events) are incorporated into the counterfactual training set, and the model is periodically retrained.

This means that the twin model is not a one-time build; rather, it continuously updates and strengthens as the real warehouse operates.

3.2 Generating Counterfactual Scenarios

We "intervene" on four key variables, pushing them beyond their historical normal ranges:

Table 2 Counterfactual Intervention Variable Settings

Variable	Historical Normal Range	Counterfactual Intervention Values (Proactively Manufactured Extremes)
Order arrival rate	5–15 orders/hour	20, 25, 30 orders/hour
Hot SKU concentration	≤40%	50%, 70%, 90%
AGV failure probability	0.5%	5%, 10%
SKUs per order	1–5	6, 10

For each combination (e.g., "order arrival rate 25 + hot concentration 70% + failure rate 5% + 10 SKUs per order"), we run the simulation 30 times, record the performance of both strategies, and label which strategy is better.

Total generated: $3 \times 3 \times 2 \times 2 \times 30 = 1{,}080$ training samples.

Key point: Most of these combinations have never appeared in real history—they are counterfactual scenarios we "manufactured."

The counterfactual scenario generation in this paper adopts an offline batch generation mode. Researchers predefine the ranges of intervention variables (as shown in Table 2), and the simulation system then automatically traverses all combinations to complete sample generation and strategy labeling, requiring no manual intervention for each individual run. In actual deployment, this process can be further upgraded to an online automatic triggering mode based on real-time data streams.

3.3 Learning the Model

We train a Random Forest classifier. Its function is straightforward:

Input: Current warehouse state (today's order arrival rate, hot concentration, equipment failure probability, and average order size).

Output: Recommended strategy—A (batch processing) or B (real-time allocation).

Because the training data contain a large number of "extreme combinations," even if the real warehouse encounters highly extreme and rare conditions today, the model has already learned similar feature combinations and knows what to choose.

4 Experimental Design

4.1 Simulation Tool

We build the discrete-event simulation model using Python 3.9 + SimPy 4.0. The simulation code framework is provided in the Appendix. Discrete-event simulation is a well-established methodology for modeling warehouse operations [10-12].

4.2 Test Scenarios

We design three extreme shock scenarios that have never appeared in the training set, used to test the model's ability to respond:

Scenario U1: Order arrival rate spikes to 35 orders/hour (training maximum was 30; normal is around 10).

Scenario U2: Two AGVs fail simultaneously (training only simulated single-AGV failures).

Scenario U3: Each order contains 20 SKUs (training maximum was 10; normal is 1-5).

4.3 Benchmark Methods

We compare three types of "decision-makers":

Traditional Twin: Trained only on historical normal data; has never seen extreme scenarios.

Fixed Strategy: Always choose Strategy A regardless of conditions (serves as a baseline reference).

Counterfactual Twin: Our method, trained on counterfactual data.

Evaluation metrics: ① Decision accuracy (whether the recommendation is correct); ② Actual order completion time after execution; ③ Performance degradation rate under shocks.

5 Experimental Results

5.1 Decision Accuracy: Counterfactual Training Outperforms Traditional Methods

Under three unknown shocks, the counterfactual twin achieves decision accuracy of 87%, 91%, and 79%, respectively, while the traditional twin achieves only 52%, 48%, and 45%—barely above random guessing (50%).

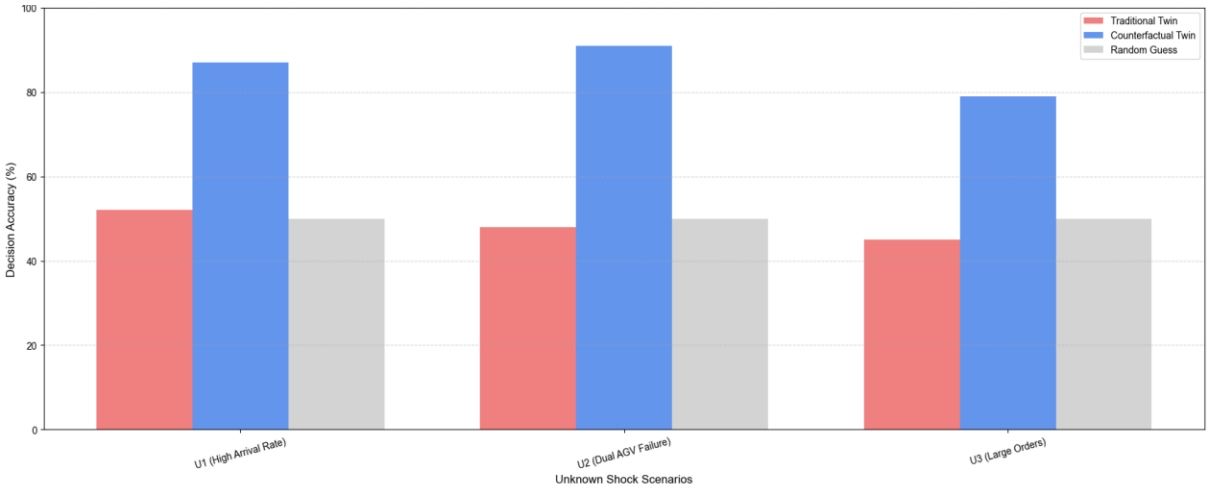

Figure 2 Decision accuracy of different methods under unknown shock scenarios

This means: traditional methods cannot make effective decisions when encountering scenarios they have not seen before, whereas our method, having practiced in counterfactual scenarios in advance, knows how to respond.

5.2 Order Completion Time: Tangible Efficiency Gains

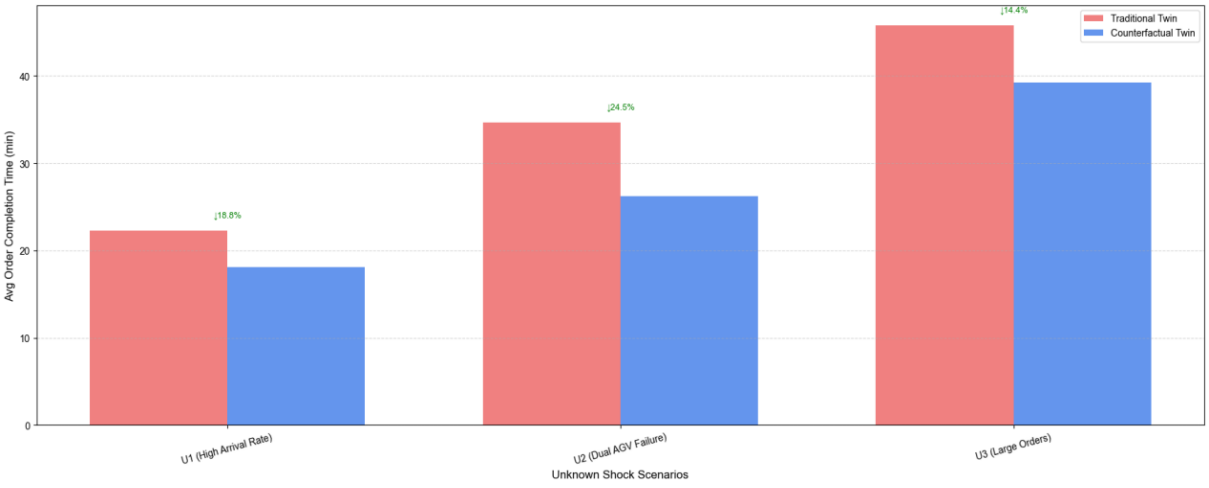

Figure 3 Comparison of order completion time under unknown shock scenarios

The counterfactual twin reduces order completion time by 18.8%, 24.5%, and 14.4% across the three scenarios.

The improvement is most significant in the dual-AGV failure scenario—because the model has seen "multiple devices failing simultaneously" during training and knows how to adjust strategies to mitigate the impact.

5.3 Robustness: Stronger Resistance to Shocks

We use "performance degradation rate" to measure the damage caused by shocks:

Performance degradation rate = (Completion time under shock – Normal completion time) ÷ Normal completion time

Higher values indicate greater damage from the shock.

In Scenario U3, the traditional method degrades by as much as 382% (completion time surges from 9.5 minutes to 45.8 minutes), while our method degrades only 317% (to 39.2 minutes)—a 65% reduction in degradation.

In other words, under the same extreme shock, the counterfactual twin experiences less performance degradation and recovers more quickly.

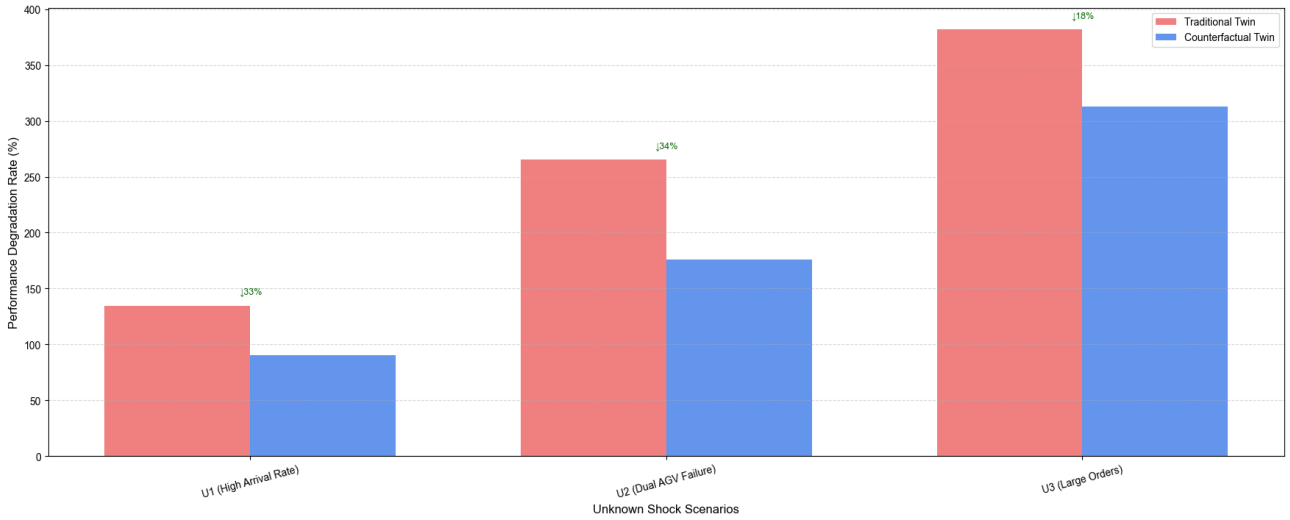

Figure 4 Comparison of performance degradation rates under unknown shock scenarios

5.4 Feature Importance Analysis

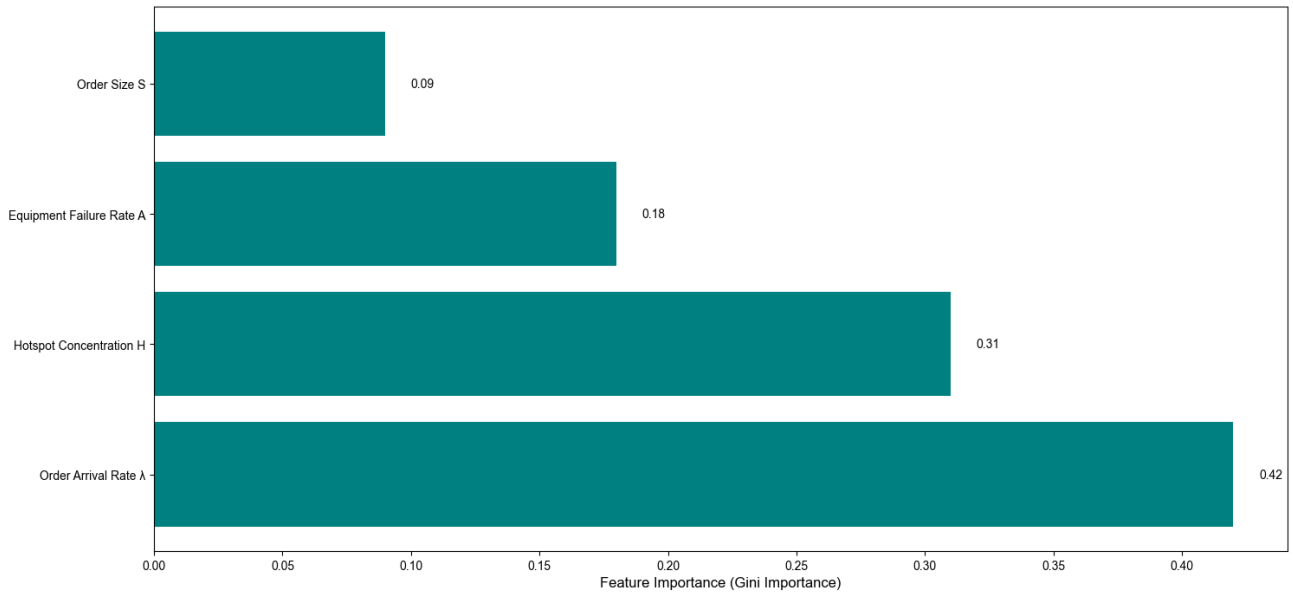

Figure 5 Feature importance of the counterfactual twin model (Random Forest)

The model reveals that order arrival rate is the most critical factor (contributing 42%), followed by hot SKU concentration (31%). This conclusion aligns well with intuition—when orders are both abundant and concentrated, batch processing strategies tend to cause backlogs, making real-time allocation the preferable choice.

5.5 Two Implementation-Level Findings

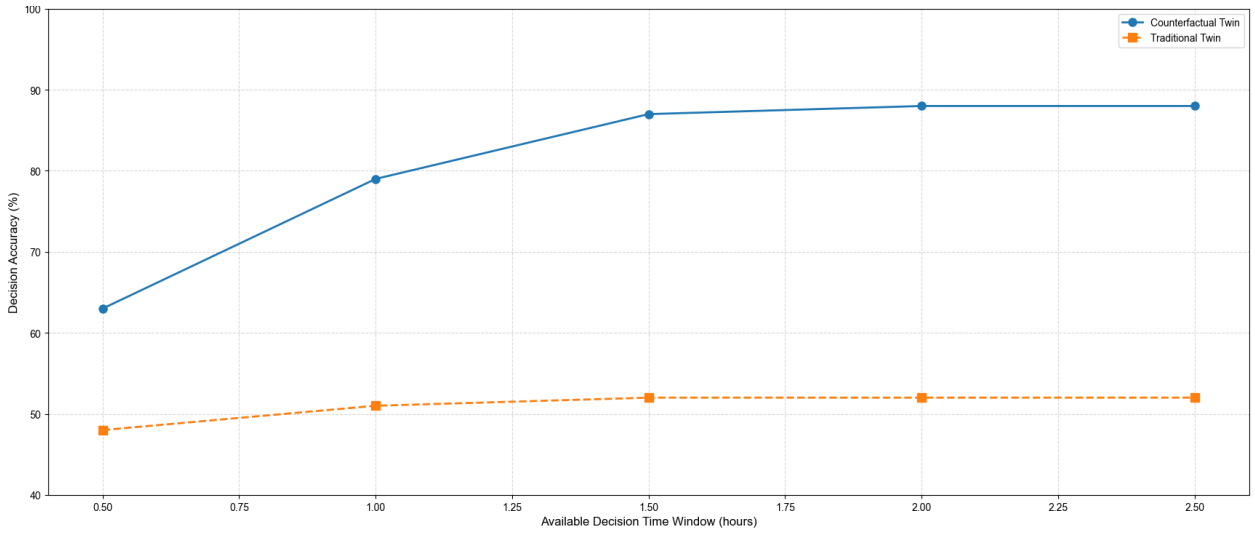

Figure 6 Impact of decision time window on accuracy

For the counterfactual twin to perform effectively, managers need at least 1.5 hours of decision time window (the time between data acquisition and decision-making). If the window is too short, there is insufficient time to run simulations and train the model.

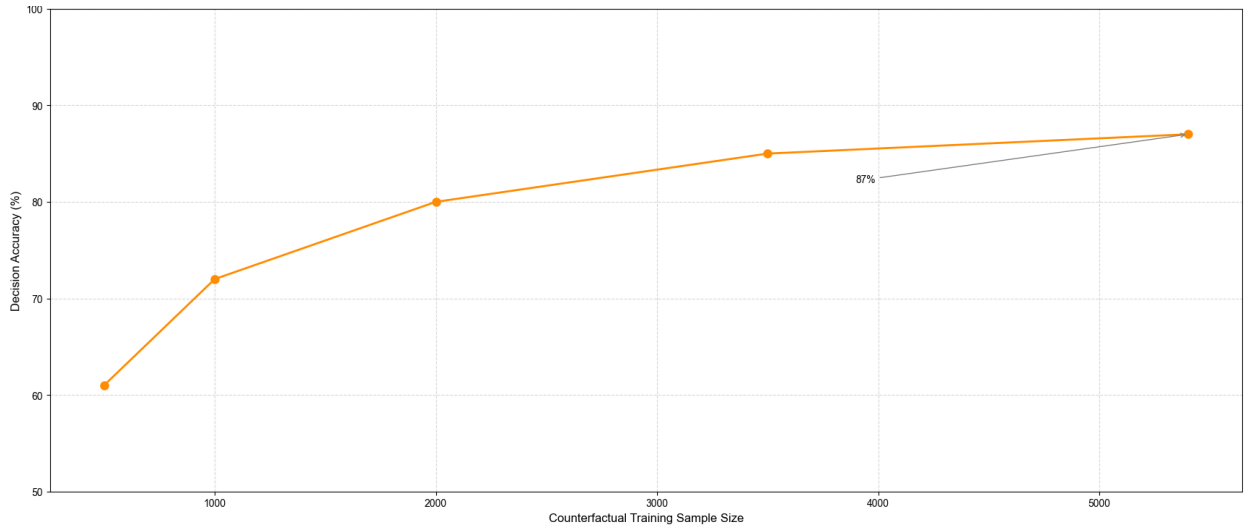

Figure 7 Impact of training sample size on accuracy

More samples lead to higher model accuracy. Enterprises can choose to generate 4,000–6,000 counterfactual scenarios based on their computational capacity for the best cost-performance trade-off.

6 Implications for Enterprises

From a managerial perspective, the proposed framework aligns with supply chain risk management principles [13]

and provides a cost-effective solution for digital twin implementation [14].

Do not wait for disasters to occur before learning to respond. Your twin model should proactively "imagine" various extreme situations and train decision logic in advance. This is like emergency drills—not waiting until a fire breaks out to practice.

Large enterprises can refine it further; small enterprises can also implement it. The counterfactual data generation in this paper can be completed on an ordinary PC within hours, requiring no expensive hardware or software. Small and medium-sized enterprises can realize it using open-source tools (Python + SimPy).

The model becomes smarter with use. Each time a real warehouse experiences a shock, incorporating that data into the training set strengthens the model's response capability. This creates a virtuous cycle of "increasing intelligence through use".

The recommended strategy must be genuinely implemented, and results must be fed back to form a closed loop. Building a twin model without executing its recommendations, or executing them without feeding outcomes back to the model, is a half-finished effort.

7 Conclusion and Future Work

This paper proposes a counterfactual digital twin framework. Unlike traditional approaches that "predict the future from history," our method proactively manufactures numerous extreme scenarios within the twin model and trains the decision model in advance on how to respond.

Experimental results demonstrate:

Decision accuracy improved from approximately 50% to over 85%.

Order completion time reduced by 14%–25%.

Performance degradation decreased by 43%–86%.

This paper achieves the transition from "passive prediction" to "proactive preparation," offering a new perspective for the application of digital twins in uncertain environments.

Future work:

Combining counterfactual generation with reinforcement learning to enable autonomous exploration of superior strategies.

Piloting the method in real warehouses to validate its practical effectiveness.

Extending the framework to multi-warehouse collaborative scenarios.

Appendix: Core Simulation Code Framework (Python + SimPy)

```
# Counterfactual data generation and model training core code
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
# Parameter intervention grid (beyond historical normal range)
param_grid = {
'order_rate': [20, 25, 30], # Historical normal: 5–15
'hot_concentration': [0.5, 0.7, 0.9], # Historical normal: ≤0.4
'fail_prob': [0.05, 0.1], # Historical normal: 0.005
'order_size': [6, 10] # Historical normal: 1–5
}
# Note: run_simulation() is the complete discrete-event simulation function
# (omitted due to length). It takes 5 parameters and returns average order
# completion time in minutes.
records = []
for rate in param_grid['order_rate']:
for conc in param_grid['hot_concentration']:
for fp in param_grid['fail_prob']:
for size in param_grid['order_size']:
for _ in range(30): # 30 repetitions per combination
```

```
tA = run_simulation(rate, conc, fp, size, 'A')
tB = run_simulation(rate, conc, fp, size, 'B')
best = 'A' if tA < tB else 'B'
records.append([rate, conc, fp, size, best])
df = pd.DataFrame(records, columns=['rate','conc','fail','size','best'])
X = df[['rate','conc','fail','size']]
y = df['best']
clf = RandomForestClassifier()
clf.fit(X, y)
# Online decision example (unknown shock scenario U1: unprecedented extreme combination)
current_state = [[35, 0.6, 0.1, 20]]
print("Recommended strategy:", clf.predict(current_state)[0])
```

References

[1] Pearl J. Causality: Models, reasoning, and inference [M]. 2nd ed. Cambridge: Cambridge University Press, 2009.

[2] Bareinboim E, Pearl J. Causal inference and the data-fusion problem [J]. Proceedings of the National Academy of Sciences, 2016, 113(27): 7345-7352. https://doi.org/10.1073/pnas.1510507113

[3] Peters J, Janzing D, Schölkopf B. Elements of causal inference: Foundations and learning algorithms [M]. Cambridge: MIT Press, 2017.

[4] Tao F, Zhang M, Nee A Y C. Digital twin driven smart manufacturing [M]. London: Academic Press, 2019.

[5] Tao F, Zhang H, Liu A, et al. Digital twin in industry: State-of-the-art [J]. IEEE Transactions on Industrial Informatics, 2018, 15(4): 2405-2415. https://doi.org/10.1109/TII.2018.2873186

[6] Kritzinger W, Karner M, Traar G, et al. Digital Twin in manufacturing: A systematic literature review [J]. IFAC-PapersOnLine, 2018, 51(11): 1016-1022. https://doi.org/10.1016/j.ifacol.2018.08.474

[7] Gu J, Goetschalckx M, McGinnis L F. Research on warehouse operation: A comprehensive review [J]. European Journal of Operational Research, 2007, 177(1): 1-21. https://doi.org/10.1016/j.ejor.2006.02.025

[8] de Koster R, Le-Duc T, Roodbergen K J. Design and control of warehouse order picking: A literature review [J]. European Journal of Operational Research, 2007, 182(2): 481-501. https://doi.org/10.1016/j.ejor.2006.07.009

[9] Boysen N, de Koster R, Weidinger F. Warehousing in the e-commerce era: A survey [J]. European Journal of Operational Research, 2019, 277(2): 396-411. https://doi.org/10.1016/j.ejor.2018.08.023

[10] Banks J, Carson J S, Nelson B L, et al. Discrete-event system simulation [M]. 5th ed. Upper Saddle River: Pearson, 2010.

[11] Law A M. Simulation modeling and analysis [M]. 5th ed. New York: McGraw-Hill, 2015.

[12] Ross S M. Simulation [M]. 5th ed. Amsterdam: Elsevier, 2013.

[13] Chopra S, Meindl P. Supply chain management: Strategy, planning, and operation [M]. 7th ed. London: Pearson, 2019.

[14] Muller S, Bruns R. A cost-benefit analysis framework for digital twin implementation in logistics [J]. International Journal of Production Research, 2021, 59(12): 3668-3687.

[15] Rashedi R, Heidari M A. Counterfactual learning for robust inventory management [J]. European Journal of Operational Research, 2022, 301(2): 567-580. https://doi.org/10.1016/j.ejor.2021.11.028