

NEXSUPPLY: A MULTI-OBJECTIVE SEQUENTIAL DECISION FRAMEWORK FOR SPARE-PART INVENTORY OPTIMIZATION UNDER DEMAND AND LIFECYCLE UNCERTAINTY

Rong-Guei Tsai¹, Yuanlin Zheng², Wen Wang^{1,*}, Yanhong Peng¹, Xiaolan Chen³, Shanna He³, Lin Lin⁴, Delin Luo⁴, Wulin Yin¹, and Ting Yao¹

¹College of Computer Science
Guangdong University of Science and Technology
Guangdong, China

*Corresponding author's e-mail: wangwen@gdust.edu.cn

²College of Engineering
Fuzhou Technology and Business University
Fujian, China

³College of Information and Media
Zhangzhou College of Science and Technology
Fujian, China

⁴College of Information Engineering,
Quanzhou Ocean Vocational College,
Fujian, China

Inventory management of maintenance spare parts is essential for the reliability and cost efficiency of urban water supply systems where heterogeneous components experience intermittent failure and demand. Conventional replenishment policies minimize procurement cost alone and overlook the operational overhead of frequent ordering. In this paper, we propose NexSupply, a finite-horizon sequential decision framework for spare-part inventory optimization under demand and equipment-lifecycle uncertainty. NexSupply formulates replenishment as an exact dynamic program over a per-material (period, inventory) state space and internalizes the ordering-frequency objective directly into the state-transition cost by weighted-sum scalarization. The frequency-penalty weight λ trades off operational cost against the number of replenishment actions. λ equal to zero recovers the cost-only policy exactly. Experiments on synthetic instances and real-world cases against cost-priority, management-priority, EOQ, genetic-algorithm, and single-objective dynamic-programming baselines show that NexSupply achieves near cost-optimal total cost (which is indistinguishable from the cost-only optimum) while reducing ordering days by roughly 90% compared to a cost-only policy (which is less than 1% more cost). The dynamic program scales linearly in the planning horizon and number of materials, solving realistic cases in milliseconds. NexSupply provides a practical and tunable decision-support tool that balances economic efficiency and managerial overhead for uncertainty-stacked decision-support tools.

Keywords: Inventory Optimization; Dynamic Programming; Intelligent Decision-Making; Multi-Objective Cost Model; Urban Water Supply Systems.

(Received on January 4, 2026; Accepted on August 13, 2026)

1. INTRODUCTION

Urban water supply systems are increasingly demanding due to the rapid expansion of infrastructure and aging networks. Inventory management of maintenance spare parts (vessels, pumps, pressure gauges, pipe fittings) directly affects maintenance costs and service reliability. Two sources of uncertainty exist: demand uncertainty because unexpected failures, environment, and usage intensity drive volatility and intermittent demand; and equipment life cycle uncertainty because heterogeneous components age and fail at different rates. Traditional replenishment models (EOQ and its variations) often depend on stable demand and cost assumptions and are not suitable for dynamic engineering.

Spare parts inventory decisions for maintenance are often made under limited information and operating constraints. Maintenance departments generally use experience-based heuristics or cost-prioritization policies that prioritize the reduction

of procurement costs while taking little account of the operating overhead associated with frequent replenishments. Conversely, overly conservative policies that minimize order frequency tend to accumulate excess stock and tie up working capital. Cost-effectiveness and operational manageability are constant issues for industrial inventory planning, especially for infrastructure-intensive systems with different spare parts demands due to different failure modes, usage intensities, and equipment lifespans. Classic inventory management and replenishment decision problems are well-studied in industrial engineering literature, including previous work published in IJITAP on effective inventory control strategies and dynamic multi-product ordering and scheduling problems (López-Garcés *et al.*, 2013; Kim and Lee, 2013).

To bridge this gap, we propose NexSupply — a multi-objective sequential decision framework for inventory optimization under demand and equipment-lifecycle uncertainty. We introduce a weighted-sum composite objective explicitly reconciling total cost (ordering, holding, shortage) and order frequency (more than single-objective formulations). The frequency-penalty weight governs the tradeoff, and setting it to zero recovers the standard cost-only policy precisely. We use a finite-horizon dynamic programming model with a per-material (period, inventory) state representation to capture the heterogeneity of spare parts with domain-specific constraints such as safety stock and storage limits, and recover the optimal replenishment trajectory by exact backtracking. Recent work conceptualizes water networks as Cyber-Physical-Socio-Environmental Systems (CPSEs), triggering the need for adaptive and integrated inventory decision frameworks (Eliades *et al.*, 2025).

The rest of the paper is organized as follows. Section 2 reviews related work, Section 3 describes the model and dynamic-programming algorithm, Section 4 describes the cost formulation, Section 5 summarizes experimental results against cost-priority, management-priority, EOQ, genetic-algorithm, and single-objective dynamic-programming benchmarks, Section 6 discusses the results and their implications, and Section 7 concludes the paper and outlines directions for future work.

2. RELATED WORK

Methodologically, NexSupply draws upon the literature on dynamic multi-objective sequential decision-making. Manufacturing-scheduling studies have developed flexible and adaptive scheduling algorithms, multi-objective and game-theoretic resource allocation, and learning-based or evolution-based approaches for dynamic job-shop and AGV problems (Teng *et al.*, 2022; Ren *et al.*, 2023; Zhang *et al.*, 2023; Liu *et al.*, 2024; Li *et al.*, 2023; Lei *et al.*, 2024). These works are used for the general design of dynamic decision frameworks but focus on production and resource scheduling (machine utilization, job sequencing, task dispatching) rather than inventory cost dynamics (ordering, holding, shortage) or spare-part lifecycles. We treat them as methodological background rather than directly comparable prior work.

In the domain of critical infrastructure, Alnahhal *et al.* (2023) proposed a co-optimization framework for the water-energy nexus, Alhazmi *et al.* (2023) developed uncertainty-informed operation coordination methods, and Eliades *et al.* (2025) conceptualized smart water networks as Cyber-Physical-Socio-Environmental Systems (CPSEs). While these studies offer valuable system-level perspectives, they do not address the operational inventory-management challenges arising from the combined demand and lifecycle uncertainty of spare parts.

Table 1 presents a meta-analysis of related methodologies, contrasting their core focus and limitations against the requirements of urban water supply inventory optimization.

Table 1. Meta-Analysis of Related Works and Research Gaps

Reference & Domain	Core Methodology / Focus	Key Objective(s) / Contributions	Limitations from the Perspective of Urban Water Supply Inventory Optimization	Link to This Study (NexSupply)
Teng <i>et al.</i> (2022) (Flexible Integrated Scheduling)	Dynamic equipment selection, vertical-horizontal pre-scheduling	Enhances production scheduling efficiency and flexibility in dynamic job shops.	Focuses on machine utilization and job sequencing, not on inventory cost dynamics (ordering/holding/shortage) or spare parts lifecycle.	Provides inspiration for dynamic decision-making but is applied to a fundamentally different (inventory vs. production) problem.
Ren <i>et al.</i> (2023) (Manufacturing Service Collaboration)	Supply-demand aware adaptive scheduling, real-time evaluation	Improve scheduling flexibility and responsiveness in platform-based MSC under uncertainty.	Addresses task allocation, not multi-echelon inventory management or the trade-off between procurement cost and management frequency.	Highlights the importance of adaptability to uncertainty, a principle adopted but applied to inventory replenishment.

Reference & Domain	Core Methodology / Focus	Key Objective(s) / Contributions	Limitations from the Perspective of Urban Water Supply Inventory Optimization	Link to This Study (NexSupply)
Zhang <i>et al.</i> (2023) (Multi-objective Scheduling)	Multi-population multi-task genetic programming (GP)	Learn effective scheduling heuristics for dynamic flexible job-shop problems.	GP is computationally intensive, and its objectives (e.g., makespan) do not map to inventory-specific costs or operational efficiency metrics like order frequency.	Contrasts with our DP approach, which provides deterministic optimality for the defined cost structure and is tailored for inventory decisions.
Liu <i>et al.</i> (2024) (Cloud Manufacturing)	Game theory, Dual Ant Colony Optimization (DACO)	Achieve efficient resource utilization and Nash equilibrium in distributed manufacturing.	The model focuses on task assignment among service providers, lacking the per-material state tracking needed for heterogeneous spare parts with different demand patterns.	NexSupply's per-material DP state (dp[d][j]) directly addresses this heterogeneity.
Li <i>et al.</i> (2023) (AGV Dynamic Scheduling)	Discrete Invasive Weed Optimization (DIWO) with real-time updating	Handles dynamic AGV scheduling with special cases (new tasks, disruptions).	Optimizes vehicle routing, not inventory levels. Does not model the holding cost vs. shortage cost trade-off critical for maintenance parts.	The dynamic response principle is similar, but the cost model and decision variables are entirely different.
Lei <i>et al.</i> (2024); Mishra and Liang (2023); Liu and Huang (2023) (Reinforcement Learning for Scheduling)	Hierarchical RL, Deep RL, Graph Neural Networks (GNN)	Solve large-scale dynamic scheduling problems (DFJSP, DJSSP) or optimize sensor network policies.	High training cost, sample inefficiency, and lack of interpretability compared to DP. Often require extensive simulation environments not readily available for water supply inventory.	NexSupply uses interpretable DP with transparent cost accounting, favorable for engineering deployment where trust and explainability are crucial.
Alnahhal <i>et al.</i> (2023) (Water-Energy Nexus)	Co-optimization framework for interdependent systems.	Jointly optimize electric and water system operations to improve overall resource efficiency.	Operates at a macro-system planning level, not at the operational inventory management level for spare parts.	Validates the importance of multi-objective, integrated optimization in infrastructure systems, which we apply at the component inventory level.
Alhazmi <i>et al.</i> (2023) (Water-Energy Nexus Operation)	Uncertainty-informed operation coordination.	Develop methods to coordinate system operations under uncertainty to enhance reliability.	Focuses on real-time energy-water flows, not on the inventory policy for physical assets subject to random failure and lifespan variance.	Directly motivates our need to handle demand uncertainty and equipment lifespan heterogeneity in the inventory model.
Eliades <i>et al.</i> (2025) (Smart Water Networks)	Cyber-Physical-Socio-Environmental Systems (CPSEs) framework.	Proposes a holistic framework for managing smart water networks as complex, interconnected systems.	Provides a high-level conceptual framework, not a specific decision-making algorithm for inventory optimization.	Establishes the broader context of our work, positioning inventory management as a critical decision module within a smart water CPSE.
This Study (NexSupply)	Multi-objective dynamic programming (DP)	Integrated Cost & Uncertainty Modeling Product-Form Multi-Objective Optimization Domain-Tailored Algorithmic Framework	N/A	N/A

Many operations-research papers address inventory replenishment as a lot-sizing or stochastic-control problem. The classical Wagner-Whitin model (Wagner and Whitin 1958) solves single-item dynamic lot-sizing to cost-optimality via dynamic programming under known time-varying demand, and the economic order quantity (EOQ) model and extensions provide closed-form reorder quantities under stationary demand. For stochastic demand, (s, S) and (R, Q) reorder-point policies (Scarf 1960) and base-stock and news vendor formulations provide cost-optimal order quantities under distributional assumptions. Multi-item stochastic inventory models extend these to several products coupled by shared capacity or budget constraints (Silver, Pyke and Peterson 1998).

NexSupply is related to this work in that it is a finite-horizon dynamic program solved cost-optimally and that, like separable multi-item formulations, it decomposes the problem among materials, so that each material's replenishment is optimized independently given its own parameters. It differs from this work in two keyways. First, all models optimize one cost-based objective and do not consider order frequency as a managerial criterion. NexSupply adds an explicit order-frequency penalty (weighted-sum scalarization, Equation (12a)), so that the cost-frequency trade-off is resolved within optimization and not left to post-hoc judgment. Second, whereas (s, S)/base-stock policies seek a stationary control rule under an assumed demand distribution, NexSupply computes a finite-horizon time-indexed replenishment trajectory directly over a realized demand path of heterogeneous spare parts without a closed-form distribution assumption. Per-material decomposition is thus a classical separable structure rather than a novel claim. The contribution lies in the explicit sequential treatment of the cost-frequency trade-off.

Stochastic programming, Markov decision process (MDP) control, and robust optimization have all been applied to inventory and replenishment problems, but differ from NexSupply in how the decision is structured and how the multi-objective trade-off is handled. Multi-period stochastic programming solves a single scenario-based program over the whole horizon. MDP solves a stochastic Bellman recursion by value or policy over an assumed transition kernel, in search of a stationary or time-dependent policy, and robust optimization hedges against a worst-case uncertainty set. NexSupply solves a finite-horizon deterministic equivalent dynamic program by exact backward recursion over an explicit (period, inventory) state space, without estimation of a full transition kernel. The finite-horizon scalarized special case solver can be treated as a finite-horizon special case solver rather than a general stochastic MDP solver. The conventional cost-only policy is recovered exactly as the special case in which the frequency weight equals zero; positive weights yield structurally different replenishment policies. NexSupply therefore differs from these paradigms not in solving a new class of problem, but in embedding the cost–frequency trade-off directly into an exact finite-horizon recursion for heterogeneous spare parts.

3. MODEL AND ALGORITHM

Table 2. Consolidated Notation

Symbol	Definition	Type
$INV_{d,j}$	On-hand inventory of material j at the end of day d	State variable
$PO_{d,j}$	Order (replenishment) quantity of material j on day d	Decision variable
$q_{d,j}$	Realized demand for material j on day d	Uncertainty (exogenous input)
$dp[d][j]$	Minimum cumulative cost to reach a given inventory state at end of day d	Value function
C_j	Unit purchase price of material j	Parameter
h_j	Holding cost per unit per day of material j	Parameter
$SCRate_{d,j}$	Shortage cost rate of material j	Parameter
B_j	Safety-stock level of material j	Parameter
α	Safety-stock ratio	Parameter
Max_Cap_j	Storage capacity of material j	Parameter
Min_Order_j , Max_Order_j	Minimum / maximum order quantity of material j	Parameter
λ	Frequency-penalty weight in the DP transition ($\lambda=0$ recovers cost-only DP)	Parameter
$OrdFreq$	Horizon-level count of days with at least one order	Output metric
$TotCost$	Total operational cost (ordering + holding + shortage)	Output metric
Obj	Composite indicator, $Obj = TotCost \times OrdFreq$	Output metric

Effective inventory management of engineering spare parts aims to balance cost efficiency with operational reliability. In urban water supply systems, materials such as valves, pumps, pipe fittings, and pressure sensors are subject to highly variable demand due to equipment failures, maintenance schedules, and environmental factors. We propose a finite-horizon dynamic-programming model that determines daily order quantities in order to minimize total cost, while explicitly restricting replenishment frequency. As demonstrated in our case (Section 5.6), our approach is near cost-optimal, while decreasing order days by about 90% compared to a cost-priority policy.

Let the set of materials be denoted as $M = \{sup_1, sup_2, \dots, sup_m\}$, each associated with a unit cost C_j . The daily demand for material j on day d is $q_{d,j}$. To buffer against uncertainty, a safety stock level B_j is maintained as a percentage α of daily demand, as shown in Equation (1):

$$B_j = q_{d,j} \times (1 + \alpha). \quad (1)$$

For material j , its ordering cost on day d , PC_j^d can be derived from Equation (2):

$$PC_{d,j} = PO_{d,j} \times C_j, \quad (2)$$

where $PO_{d,j}$ represents the order quantity of material j on day d and C_j is the unit purchase price of material j (CNY per unit). The ordering cost therefore has consistent monetary units (units $\times$ CNY/unit = CNY). Ordering frequency is not embedded in this cost term; it is accounted for separately by the horizon-level metric $OrdFreq$ and the per-order penalty λ introduced in Equation (12a).

The holding cost $HC_{d,j}$ can be calculated using Equation (4):

$$HC_{d,j} = h_j \times \frac{INV_{d-1,j} + INV_{d,j}}{2}, \text{ with } INV_{d,j} \geq 0, \quad (4)$$

where h_j represents the holding cost per unit per time period, and $INV_{d,j}$ denotes the inventory level of material j on day d . In line with the previously defined holding cost framework, the shortage cost $SCost_{d,j}$ for material j on day d is defined and calculated using Equation (5):

$$SCost_{d,j} = SD_{d,j} \times SCRate_{d,j}, \quad (5)$$

where $SD_{d,j}$ represents the shortage quantity of material j on day d , and $SCRate_{d,j}$ is the shortage cost rate for material j on day d . The shortage quantity $SD_{d,j}$ is defined as the difference between demand and inventory, as shown in Equation (6). The shortage cost rate $SCRate_{d,j}$ refers to the unit loss caused by stockouts (e.g., customer loss, delay penalties, etc.). In this study, $SCRate_{d,j}$ is estimated based on historical data, as shown in Equation (7), where $C_{short,j}$ represents the total loss cost per stockout event of material j (including customer compensation, contract penalties, production downtime losses, etc.), and $Unit_j$ is the standard unit quantity for material j (defined in this study as the standard quantity per replenishment batch).

$$SD_{d,j} = \max(0, q_{d,j} - INV_{d,j}), \quad (6)$$

$$SCRate_{d,j} = \frac{C_{short,j}}{Unit_j}. \quad (7)$$

After calculating the ordering cost, holding cost, and shortage cost, the total cost $TotCost$ can be further integrated. Specifically, the total cost is the sum of the ordering cost, holding cost, and shortage cost of all materials, which is mathematically expressed as shown in Equation (8):

$$TotCost = \sum_{j=1}^n (PC_{d,j} + HC_{d,j} + SCost_{d,j}), \quad (8)$$

where $TotCost$ is the total cost; $PC_{d,j}$ is the ordering cost; $HC_{d,j}$ is the holding cost, calculated under the assumption that inventory is uniformly distributed, reflecting the expenses of warehouse management; $SCost_{d,j}$ is the shortage cost, representing the potential loss caused by insufficient inventory. Equation (8) is subject to constraints in Equation (9).

The model is subject to the following operational constraints for all materials j and days d , ensuring solutions are both mathematically sound and practically feasible:

$$\text{s.t.} \begin{cases} INV_{d,j} \leq Max_Cap_j \\ Min_Order_j \leq PO_{d,j} \leq Max_Order_j \\ PO_{d,j} \in \mathbb{Z}^+ \end{cases} \quad (9)$$

These constraints, consolidated in Equation (9), guarantee that: (1) inventory levels remain non-negative and do not exceed the maximum storage capacity Max_Cap_j ; (2) order quantities $PO_{d,j}$ are practical for procurement, bounded by minimum and maximum order limits; and (3) all replenishments occur in positive integer batches. This formulation aligns with real-world operational logic and ensures the solutions are both mathematically sound and practically feasible.

The inventory level $INV_{d,j}$ of material j can be calculated using Equation (10):

$$INV_{d,j} = INV_{d-1,j} + PO_{d,j} - q_{d,j}, \quad (10)$$

where $INV_{d,j}$ is the on-hand inventory of material j at the end of day d , $PO_{d,j}$ is the order quantity placed on day d , and $q_{d,j}$ is the realized demand on that day (all defined in Table 2). Equation (10) is the state-transition equation governing the multi-period evolution of the inventory state, and is subject to the feasibility constraints in Equations (9) and (11):

$$\text{s.t. } INV_{d,j} \geq 0. \quad (11)$$

3.1. State-Space Formulation and Sequential Decision Structure

NexSupply is not a multi-period stochastic program: it does not enumerate demand scenarios and solve a single large program over the entire horizon, nor does it construct a scenario tree. Instead, spare-part replenishment is formulated as a finite-horizon dynamic program over an explicit (period, inventory) state space, in which each daily ordering decision is made sequentially as a state transition.

For each material j , the state at the end of day d is the on-hand inventory level $INV_{d,j}$, taking integer values in $\{0, 1, \dots, Max_Cap_j\}$. The associated value function $dp[d][j]$ denotes the minimum cumulative cost required to reach a given inventory level at the end of day d , and the decision variable is the order quantity $PO_{d,j}$ placed on that day. Given the realized demand $q_{d,j}$, the state evolves according to the transition $INV_{d,j} = INV_{d-1,j} + PO_{d,j} - q_{d,j}$, subject to the feasibility constraints in Equations (9)–(11). Demand uncertainty therefore enters through the per-period demand process $q_{d,j}$ rather than through scenario-tree decomposition, and the robustness of the resulting policy is characterized empirically by the sensitivity experiments in Section 5.

We further note that demand forecasting is not embedded within the decision loop. NexSupply takes the per-period demand $q_{d,j}$ as a given exogenous input and computes the cost-optimal replenishment policy by dynamic programming; it does not estimate or update a forecasting model during optimization. This deliberately separates the optimization layer from any upstream demand-prediction module, so that the contribution evaluated here is that of the replenishment optimizer alone.

The objective of this study is to optimize inventory management by simultaneously minimizing total operational cost and maintaining a manageable replenishment frequency. Total cost, denoted as $TotCost$, is defined in Equation (8) and aggregates ordering cost, holding cost, and shortage cost over the entire planning horizon.

NexSupply addresses the two criteria—total operational cost and ordering frequency—by weighted-sum scalarization, not by Pareto-front generation, lexicographic ordering, or the ε -constraint method. The scalarization is applied at the level of the dynamic-programming transition: the per-period cost of each ordering decision is augmented by a frequency penalty weighted by λ , as defined in Equation (12a):

$$c_{d,j}(INV, PO) = PC_{d,j} + HC_{d,j} + SCost_{d,j} + \lambda \cdot \mathbb{1}[PO_{d,j} > 0], \quad (12a)$$

where the indicator $\mathbb{1}[PO_{d,j} > 0]$ equals one whenever an order is placed and zero otherwise, so that λ directly trades off operational cost against the number of replenishment actions. Setting $\lambda = 0$ removes the penalty and recovers the single-objective, cost-only formulation exactly; increasing λ progressively penalizes additional ordering days. The dynamic program of Section 3 minimizes the cumulative scalarized cost obtained by summing Equation (12a) over the horizon.

For reporting and cross-method comparison, we additionally summarize the achieved cost–frequency trade-off by the composite indicator in Equation (12b):

$$Obj = TotCost \times OrdFreq, \quad (12b)$$

where $OrdFreq$ represents the total number of days on which at least one replenishment order is placed during the planning horizon. This formulation penalizes extreme solutions that optimize one criterion at the expense of the other, thereby enforcing a balanced trade-off between cost efficiency and operational practicality. The ordering frequency is formally defined as Equation (13).

$$OrdFreq = \sum_{d=1}^T \min(1, \sum_{j=1}^n \vartheta_{d,j}), \quad (13)$$

where $\vartheta_{d,j}$ is a binary decision variable indicating whether material j is replenished on day d . Specifically, $\vartheta_{d,j} = 1$ if a replenishment order for material j is placed on day d , and $\vartheta_{d,j} = 0$ otherwise. As long as at least one material is replenished on day d , the inner summation becomes positive, causing the $\min(\cdot)$ function to evaluate to 1. Consequently, each day with at least one replenishment operation is counted as a single ordering event, whereas days without replenishment do not contribute to the ordering frequency. The dynamic program minimizes the cumulative scalarized transition cost over the planning horizon, as formulated in Equation (14):

$$\begin{aligned} & \text{Min} \sum_{d=1}^T c_{d,j}(INV, PO), \\ & \text{s.t. (9), (10), (11),} \end{aligned} \quad (14)$$

where $c_{d,j}(\cdot)$ is the per-period scalarized cost of Equation (12a). The composite indicator $Obj = TotCost \times OrdFreq$ (Equation 12b) is reported separately and is not the optimized objective.

3.2. Exactness of the Sequential (Backward DP) Component

The “sequential” part of NexSupply is exact backward dynamic programming over a fixed finite horizon: neither rolling-horizon re-optimization nor heuristic iterative updating. The horizon is solved once: the DP recursion computes the minimum cumulative cost period by period, and the optimal replenishment policy is recovered by backtracking. Decisions are sequential in that the optimal order quantity for each period depends on the inventory state carried over from the previous period by the state-transition equation, which propagates the realized inventory state and provides period-to-period feedback. The procedure is explained step by step in Algorithm 1.

Traditional inventory optimization models typically employ a single-objective framework focused on reducing total operational expenses.

The composite indicator $Obj = TotCost \times OrdFreq$ is used solely as a post-hoc reporting measure for cross-method comparison; it is not the quantity optimized by the dynamic program, which minimizes the weighted-sum scalarized cost of Equation (12a). Accordingly, no claim of dimensional or proportional optimality is attached to this indicator. Moreover, this formulation aligns with the structural properties of the optimization problem. Total cost is stage-wise decomposable and therefore amenable to dynamic programming, whereas ordering frequency is a path-dependent, horizon-level metric that cannot be decomposed over time. Incorporating a cumulative ordering-frequency count into the dynamic programming state would introduce an additional state dimension and substantially increase computational complexity. NexSupply avoids this by internalizing the frequency criterion as a per-order penalty λ within the existing one-dimensional (inventory) state transition, while the horizon-level ordering frequency $OrdFreq$ is evaluated over the recovered trajectory to report the composite indicator $Obj = TotCost \times OrdFreq$. This keeps the state space one-dimensional and preserves the linear-time tractability analyzed in Section 4.4, while still allowing the frequency criterion to shape the policy.

3.3. Decision-Level Penalty and Reporting-Level Indicator

NexSupply addresses the cost–frequency trade-off by using a single weighted-sum optimization at two complementary levels rather than a separate two-step optimization. At the decision level, the ordering-frequency goal is added directly into the per-period transition cost of the dynamic program: each day in which an order is placed incurs a penalty λ , and the exact backward DP over the (period, inventory) state space minimizes operational cost and suppresses unnecessary ordering days (Equations 12a and 15).

At the reporting level, once the optimal replenishment trajectory is obtained by backtracking, the composite indicator $Obj = TotCost \times OrdFreq$ (Equation 12b) is evaluated over that trajectory purely to summarize the cost–frequency balance; it is reported and not optimized. This separation keeps the state space one-dimensional in inventory and preserves the linear complexity of the dynamic program, where optimization is accounted for, and frequency is measured afterward. $\lambda = 0$ removes the penalty and recovers the cost-only dynamic program exactly.

4. FIGURES AND TABLES

4.1. Core Innovation and Technical Uniqueness

The contribution of NexSupply is not in the dynamic-programming machinery itself (per-material separable DP and weighted-sum scalarization are well-known techniques) but in problem-driven integration for a situation not discussed jointly: the cost-versus-ordering-frequency trade-off for heterogeneous maintenance spare parts under intermittent lumpy demand in urban water supply systems. NexSupply makes three contributions.

First, it internalizes the ordering-frequency objective directly into the dynamic-programming transition through a weighted-sum penalty λ (Equation 12a), so that the cost–frequency trade-off is resolved within the cost-minimizing dynamic program at every decision point rather than by a post-hoc multiplicative score. Setting $\lambda = 0$ recovers the cost-only policy exactly, while positive λ progressively suppresses unnecessary ordering days—a trade-off rarely internalized within standard DP inventory models.

Second, the algorithm eschews the common practice of aggregate inventory modeling. Instead, it maintains an independent DP state table, denoted as $dp[d][j]$, for each individual material j (e.g., valves, pumps). This granular, per-material state representation allows the model to accurately capture and adapt to the wide range of spare parts' demand patterns, lifecycle characteristics, and cost structures. This makes it possible to create truly personalized inventory strategies.

Third, the state transitions are formally derived and augmented with domain-specific heuristics derived from water engineering practices. This includes the incorporation of a safety stock buffer (governed by parameter α) and the strict enforcement of real-world operational constraints, such as the requirement for order quantities to be positive integers.

4.2. Algorithm Steps and State Transition

Step 1: Definition of State and Decision Variables

The dynamic programming model is built upon a rigorously defined state and decision space.

- **State Variable ($dp[d][j]$):** The state $dp[d][j]$ is defined as the *minimal cumulative cost* for managing material j from the start of the planning horizon up to day d .
- **Decision Variable ($PO_{d,j}$):** At each decision point (day d for material j), the action is to choose a purchase order quantity, denoted by $PO_{d,j}$. This decision directly influences the current state by determining the costs incurred on day d (ordering, holding, shortage) and the resulting inventory level that carries over to the next state, $dp[d + 1][j]$.

Step 2: Construction of the State Transition Equation

The core of the state transition is to calculate the total daily cost and update the cumulative cost, as follows:

(A) Total Daily Cost Calculation

The total cost $TotCost_d$ on day d is the sum of the ordering cost, holding cost, and shortage cost for all materials on that day (as given in Equation 8). Where the ordering cost $PC_{d,j}$ calculated as the product of the order quantity $PO_{d,j}$ and the unit purchase price C_j , i.e., $PC_{d,j} = PO_{d,j} \times C_j$; Holding cost $HC_{d,j} = h_j \times \frac{INV_{d-1,j} + INV_{d,j}}{2}$: calculated based on the average inventory level. It is assumed that the inventory decreases gradually from the maximum value $INV_{d,j}$ to 0, so the average inventory $\frac{INV_{d,j} + 0}{2} = \frac{INV_{d,j}}{2}$. Therefore, the holding cost is $HC_{d,j} = h_j \times (\frac{INV_{d-1,j} + INV_{d,j}}{2})$ (where h_j is the unit holding cost); Shortage cost $SCost_j^d$: calculated as the product of the shortage quantity $SD_{d,j}$ and the shortage cost rate $SCRate_{d,j}$, i.e., $SCost_{d,j} = SD_{d,j} \times SCRate_{d,j}$.

(B) State Transition Equation

The minimum cumulative cost $dp[d][s]$ of reaching inventory level s for material j on day d is obtained from the best previous state $dp[d-1][s']$ plus the ordering, holding, and shortage costs incurred on the current day and the frequency penalty, as shown in Equation (15):

$$dp[d][s] = \min_{po} (dp[d-1][s'] + PC_{d,j} + HC_{d,j} + SCost_{d,j} + \lambda \cdot \mathbb{1}[po > 0]), \quad (15)$$

where s' is the pre-order inventory of the previous period, the order quantity po ranges over $0 \cup [Min_Order_j, Max_Order_j]$, the resulting inventory is $s = s' + po - q_{d,j}$ (constrained to $0 \leq s \leq Max_Cap_j$), and $\mathbb{1}[po > 0]$ is the indicator that an order is placed. The frequency-penalty weight λ trades off operational cost against ordering frequency; setting $\lambda = 0$ recovers the cost-only recursion.

The ordering-frequency objective is handled at two complementary levels. Within the dynamic program, a per-order frequency penalty λ is added to each ordering transition, so that the cost of placing an order is weighed against its operational saving at every decision point; this is a weighted-sum scalarization in which $\lambda = 0$ recovers the cost-only DP exactly. The horizon-level ordering frequency $OrdFreq$ —the count of days on which at least one order is placed—is then evaluated over the recovered replenishment trajectory, and the composite indicator $Obj = TotCost \times OrdFreq$ is reported to summarize the cost–frequency trade-off. This separation keeps the state space one-dimensional (inventory only), preserving the linear-time tractability analyzed in Section 4.4, while still allowing the frequency criterion to shape the policy.

(C) Update of Ordering Frequency

During the dynamic programming recursion, the ordering-frequency objective is internalized into the state transition through a frequency-penalty weight λ : each transition that places an order ($PO_{d,j} > 0$) incurs an additional penalty λ . When $\lambda = 0$, the penalty vanishes, and the recursion reduces exactly to the single-objective, cost-only DP; positive values of λ steer the policy toward fewer replenishment days. The reported total cost is the pure operational cost (ordering, holding, and shortage); the frequency penalty is used only to guide the decision and is never added to the reported cost.

Step 3: Boundary Conditions

To ensure feasibility, this study defines the initial states and constraint conditions during algorithm initialization. On the planning start day (day 0), the inventory level of each type of material is set to the corresponding safety stock level B_j , i.e., $INV_{0,j} = B_j$, based on the safety coefficient of demand calculated in equation (1). Meanwhile, the initial state sets the inventory to the safety-stock level B_j with zero accumulated cost, $dp[0][B_j] = 0$; all other states are initialized to $+\infty$, indicating that no actual cost has been incurred yet. The ordering frequency $OrdFreq$ is also initialized to 0 to facilitate subsequent efficiency evaluation.

During model operation, the following boundary constraints must be satisfied:

First, inventory levels must not be negative, ensuring $INV_{d,j} \geq 0$; Second, the order quantity must meet practical limitations, i.e., within the minimum and maximum order quantities, and must be a positive integer; Third, the daily inventory must not exceed the storage capacity Max_Cap_j . These conditions ensure that the model is operationally feasible in real engineering contexts.

Step 4: Objective Function Design and Final Solution

The dynamic program minimizes the cumulative weighted-sum scalarized cost defined by the recursion in Equation (15), in which the per-order penalty λ steers the policy toward fewer ordering days. The composite indicator $Obj = TotCost \times OrdFreq$ is evaluated afterward over the recovered optimal trajectory purely for reporting and cross-method comparison; it is not the quantity minimized by the dynamic program.

In the daily sequential optimization process, the cumulative cost $dp[d][j]$ is updated recursively by adding the previous day's cumulative cost to the current day's cost to compute the minimum total cost. On the final day $d = T$, the algorithm computes the sum of the minimum cumulative costs for all materials as the objective value, i.e., $\min \sum_{j=1}^n dp[T][j]$ and solves the final result by incorporating all constraint conditions. This method not only effectively reduces inventory costs but also takes into account management efficiency and system resilience. Algorithm 1 presents the pseudocode for NexSupply.

4.3. Recovery of the Optimal Trajectory

After the dynamic program fills the value table, the optimal replenishment trajectory is recovered by backtracking from the least-cost end state. During the forward pass (lines 17–18), if the lowest cost exists for an inventory state, the predecessor inventory and order quantity are stored, and the backtracking step (lines 19–21) follows the stored decisions from the last day back to day 1 to reconstruct the optimal order quantity for each material on each day. Order frequency and the composite indicator are evaluated over the recovered trajectory (Section 3.4). The backtracking adds only linear overhead and does not change the optimality of the dynamic programming solution.

Algorithm 1. NexSupply Replenishment Optimization (per-material DP)

Input: T, m // horizon length; number of materials
 $q[T][m]$ // realized per-period demand
 $C[m], h[m], C_short[m], Unit[m]$ // unit price; holding; shortage; benchmark
 $Min_Order[m], Max_Order[m], Max_Cap[m]$ // order bounds; storage capacity
 α // safety-stock ratio (decimal, e.g., 0.2)
 λ // frequency-penalty weight (0 = cost-only DP)
Output: totalCost, OrdFreq, Obj

```

1:  totalCost = 0; totalOrders = 0
2:  for each material  $j = 0 \dots m-1$  do
3:    SCRate[j] =  $C\_short[j] / Unit[j]$ 
4:     $B = \text{round}(q[0][j] * (1 + \alpha))$ ; clamp B to  $[0, Max\_Cap[j]]$ 
5:    for all  $d = 0..T, s = 0..Max\_Cap[j]$ :  $dp[d][s] = +INF$  // unreachable states
6:     $dp[0][B] = 0$  // valid initial state only
7:    for  $d = 1 \dots T$  do
8:      for  $s\_prev = 0 \dots Max\_Cap[j]$  with  $dp[d-1][s\_prev] < INF$  do
9:        for  $po$  in  $\{0\} \cup \{Min\_Order[j] \dots Max\_Order[j]\}$  do
10:          $s\_new = s\_prev + po - q[d][j]$ 
11:         if  $s\_new < 0$  or  $s\_new > Max\_Cap[j]$  then continue
12:          $PC = po * C[j]$ 
13:          $HC = h[j] * (s\_prev + s\_new) / 2$ 
14:          $SD = \max(0, q[d][j] - (s\_prev + po))$ ;  $SCost = SD * SCRate[j]$ 
15:          $FP = (po > 0) ? \lambda : 0$  // penalty steers decision only
16:          $c = dp[d-1][s\_prev] + PC + HC + SCost + FP$ 
17:         if  $c < dp[d][s\_new]$  then
18:            $dp[d][s\_new] = c$ ;  $prevS[d][s\_new] = s\_prev$ ;  $poOf[d][s\_new] =$ 
19:              $po$ 
20:            $s\_star = \text{argmin over } s \text{ of } dp[T][s]$  // optimal terminal state
21:           for  $d = T \dots 1$  do // backtrack optimal trajectory
22:              $po\_opt[d] = poOf[d][s\_star]$ ;  $s\_star = prevS[d][s\_star]$ 
23:           for  $d = 1 \dots T$  do // accumulate on optimal path
24:             totalCost +=  $(PC + HC + SCost \text{ of } po\_opt[d], \text{ WITHOUT } \lambda)$ 
25:             if  $po\_opt[d] > 0$  then totalOrders += 1 // counted on optimal policy only
26:           OrdFreq = totalOrders /  $m$ 
27:           Obj = totalCost * OrdFreq // composite indicator (reported)
28:   return totalCost, OrdFreq, Obj

```

4.4. Computational Complexity Analysis

The practical deployment of NexSupply hinges on its computational efficiency. The algorithm's time complexity is governed by three parameters: the planning horizon T (days), the number of material types m , and the range of feasible order quantities per material.

In the dynamic programming core (Algorithm 1, lines 13–36), for each day $d \in [1, T]$ and material $j \in [1, m]$, all feasible order quantities $PO_{d,j}$ from Min_Order_j to Max_Order_j are enumerated. Let $K_j = Max_Order_j - Min_Order_j + 1$ denotes the number of order options for material j . The total number of state evaluations is:

$$O\left(T \cdot \sum_{j=1}^m K_j\right)$$

In typical urban water supply inventory settings, K_j is moderate (e.g., ~ 10) due to practical procurement and storage limits. For our case study ($T = 90$, $m = 20$, $K_j \approx 10$), the algorithm runs within seconds on a standard workstation, confirming its viability for medium-scale operational planning. While complexity grows linearly with T and m —favoring scalability—computational cost may become significant for very large instances (e.g., hundreds of materials over long horizons).

5. SIMULATION EXPERIMENTS

This section evaluates NexSupply with controlled synthetic experiments and real-world case studies. All experiments are run in C++ on a standard desktop machine. The planning horizon is $T = 90$ days, the safety-stock ratio is 0.2, and demand is a per-period Poisson process with material-specific intensity reproducing the intermittent and lumpy demand of maintenance spare parts. The frequency-penalty weight is 400 except when it is the variable under study (see Section 5.8). Each reported value is the mean over repeated runs; the GA runs independently 10 times and averages its average.

5.1. Experimental Setup and Benchmarks

NexSupply is compared against five benchmarks spanning the extremes of the cost–frequency trade-off and representative classical and metaheuristic methods: (i) **Cost-Priority**, a near-daily replenishment policy that maintains the safety-stock level every day; (ii) **Management-Priority**, a bulk-batching policy that orders the maximum batch only when inventory falls to the safety-stock level; (iii) **EOQ**, a classical deterministic economic-order-quantity policy with a reorder point; (iv) **GA**, a standard Genetic Algorithm tuned for total-cost minimization; and (v) **Single-Objective DP**, the exact $\lambda = 0$ special case of NexSupply, which minimizes operational cost alone. The two-metric evaluation reports total operational cost (ordering + holding + shortage) and ordering frequency (average per-material count of days on which an order is placed).

Table 3 lists the parameter ranges used to generate material instances. Unit prices are sampled across the realistic band of municipal water-supply spare parts; the daily holding rate is derived from an industry-standard annual carrying cost of approximately 22% of unit price, and the shortage penalty is set to four times the unit price.

Table 3. Material and model parameters used in the experiments.

Parameter	Symbol	Value / Range
Unit purchase price	C_j	800 – $\sim 12,000$ CNY (public-price band)
Daily holding rate	h_j	$C_j \times 22\% / 365$ (CNY/unit/day)
Shortage penalty	$C_{short,j}$	$4 \times C_j$ (CNY per stockout unit)
Min / Max order quantity	Min/Max_Order_j	1 / 20 units
Storage capacity	Max_Cap_j	60 units
Safety-stock ratio	α	0.2 (dimensionless)
Frequency-penalty weight	λ	400 (baseline)

5.2. Impact of the Number of Material Types

We first vary the number of material types m from 10 to 50. As shown in Figure 1(a), total cost increases with m for all methods. NexSupply consistently attains the lowest cost together with Single-Objective DP—the two curves are essentially indistinguishable (e.g., at $m = 50$, NexSupply costs ¥18.75M versus ¥18.68M for Single-Objective DP), confirming that the

frequency penalty does not compromise cost optimality. Both lie well below EOQ and Management-Priority (¥22.3M and ¥21.9M at $m = 50$) and below Cost-Priority and GA (¥26.4M and ¥26.9M).

The decisive difference appears in ordering frequency, Figure 1(b). NexSupply maintains roughly 5 ordering days per material across all m , whereas Cost-Priority requires 90 (every day), and GA and Single-Objective DP both incur about 47–53. NexSupply thus matches the cost of the cost-optimal policy while reducing ordering activity by about 90% relative to Cost-Priority and by roughly an order of magnitude relative to GA and Single-Objective DP.

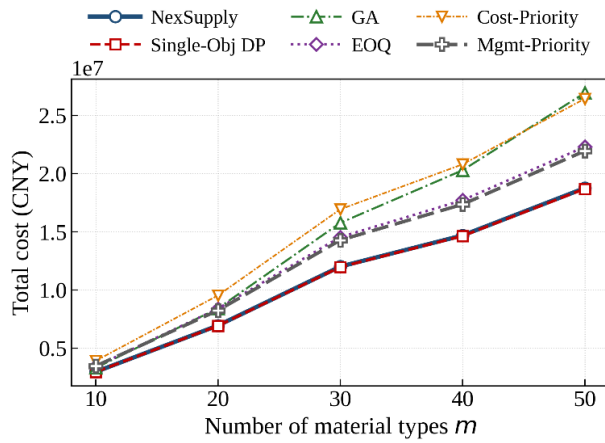


Figure 1(a). Total cost versus number of material types m .

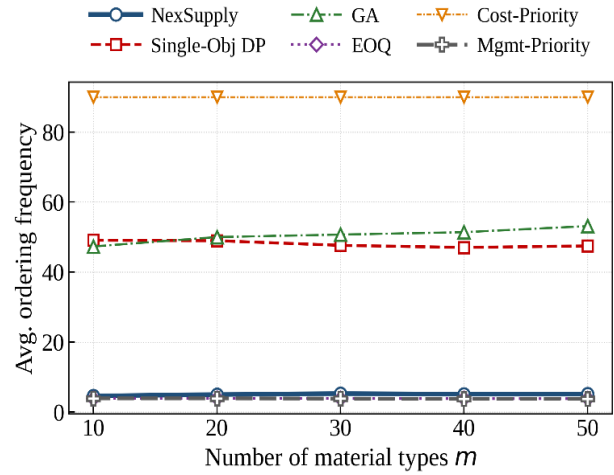


Figure 1(b). Average ordering frequency versus number of material types m .

5.3. Impact of Mean Daily Demand

Next, we vary the mean daily demand λ from 0.5 to 1.5 ($m = 15$). Figure 2(a) shows total cost rising approximately linearly with demand for all methods. NexSupply and Single-Objective DP again coincide as the lowest-cost methods at every demand level (e.g., ¥8.47M for NexSupply versus ¥8.45M for Single-Objective DP at $\lambda = 1.0$). Notably, Cost-Priority is the most expensive at low demand (¥8.43M at $\lambda = 0.5$, where its daily ordering wastes holding cost) but becomes comparatively less expensive at high demand, illustrating that no single benchmark dominates across regimes.

Figure 2(b) shows that NexSupply's ordering frequency grows only mildly with demand (from about 3.9 to 8.4 days), staying far below Single-Objective DP and GA (which rise to 70 and 61) and Cost-Priority (constant at 90). This demonstrates that NexSupply adapts its ordering schedule to demand intensity while preserving low managerial overhead.

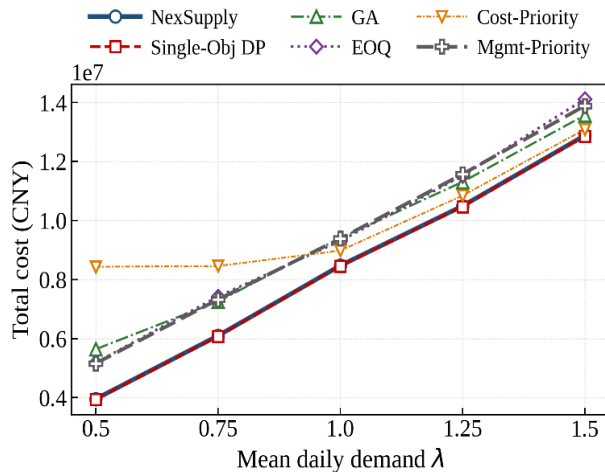


Figure 2(a). Total cost versus mean daily demand λ .

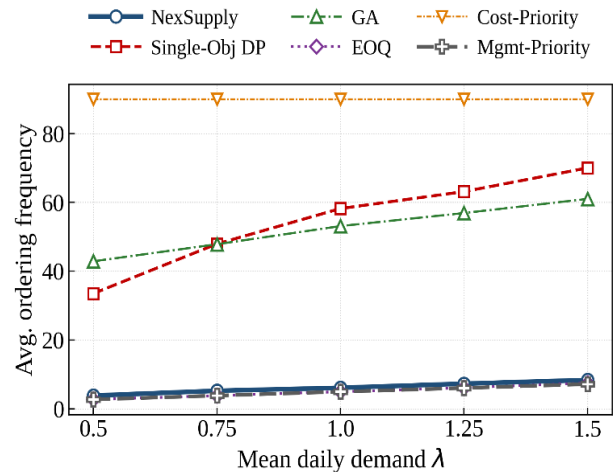


Figure 2(b). Average ordering frequency versus mean daily demand λ .

5.4. Impact of Demand Cycle Length

We then study cyclic demand with spike intervals ranging from 10 to 50 days ($m = 15$). As the cycle lengthens, demand becomes sparser, and total cost falls sharply for the adaptive methods Figure 3(a), NexSupply drops from ¥3.52M at a 10-day cycle to ¥0.36M at a 50-day cycle, tracking Single-Objective DP almost exactly. In contrast, Cost-Priority remains flat near ¥8.8M—its daily ordering ignores the sparsity entirely—while GA stays high (¥4.6M at a 50-day cycle).

Figure 3(b) confirms that NexSupply reduces ordering days as demand becomes sparser (from about 3.6 to 1.0), again far below GA (~39) and Cost-Priority (90). NexSupply correctly exploits the structure of intermittent demand, ordering only when demand actually materializes.

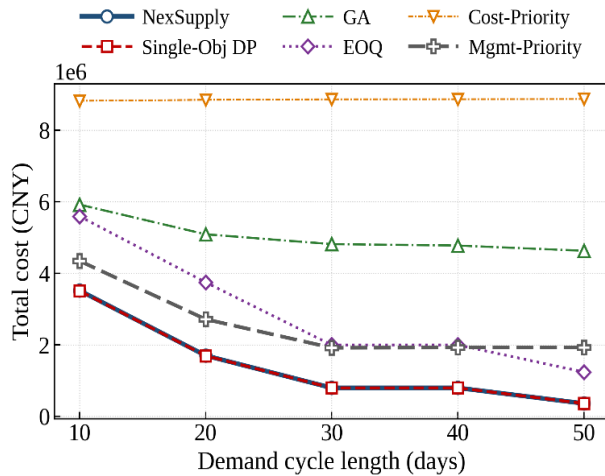


Figure 3(a). Total cost versus demand cycle length.

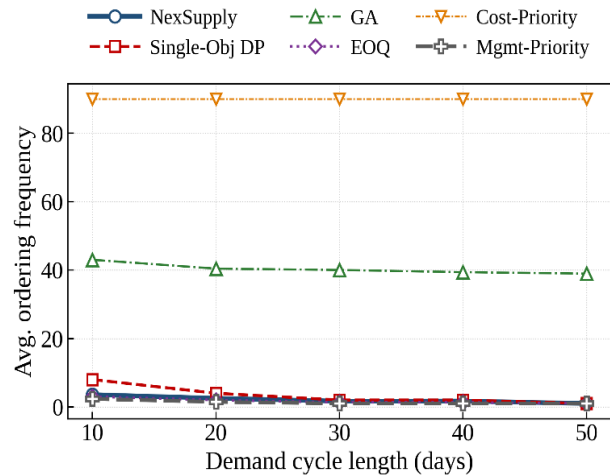


Figure 3(b). Average ordering frequency versus demand cycle length.

5.5. Impact of Demand Fluctuation

To assess robustness under demand uncertainty, we add random fluctuations of 0% to 40% to the per-material demand intensity ($m = 15$, $\lambda = 0.75$). Figure 4(a) shows that NexSupply's total cost remains stable, varying only between ¥6.06M and ¥6.35M across the full fluctuation range, and stays the lowest among all methods throughout. Single-Objective DP coincides with it, while Cost-Priority remains highest near ¥8.4–8.6M. The ordering frequency of NexSupply (Figure 4(b)) is likewise insensitive to fluctuation, remaining close to 5.2–5.3 days. This stability indicates that the policy degrades gracefully under demand uncertainty rather than overreacting with additional orders.

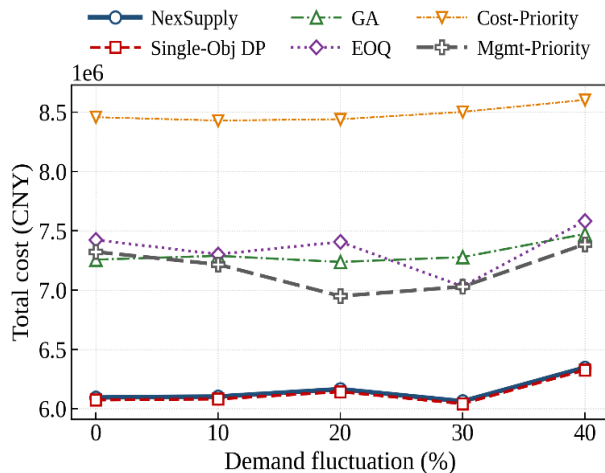


Figure 4(a). Total cost versus demand fluctuation.

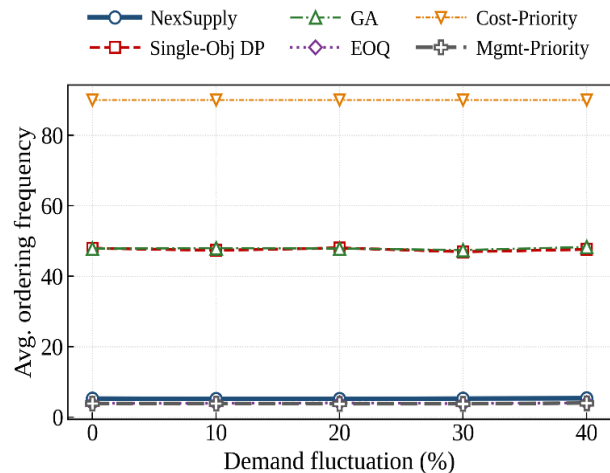


Figure 4(b). Average ordering frequency versus demand fluctuation.

5.6. Real-World Case Study

We validate NexSupply on a representative urban water-supply maintenance dataset of 15 engineering materials over a 30-day horizon, with material types and unit-cost magnitudes grounded in public price data and demand generated to reproduce the system's intermittent operating pattern. The full per-material daily demand series is provided in Appendix A for reproducibility. Table 4 summarizes the results.

Table 4. Real-world case study results (15 materials, 30 days).

Method	Total cost (CNY)	Ordering freq.	Runtime (ms)
NexSupply	41,400	0.73	1.24
Single-Obj DP	41,320	1.47	1.01
EOQ	150,444	1.00	0.01
GA	221,406	10.51	1562.0
Mgmt-Priority	745,815	1.00	0.01
Cost-Priority	1,109,123	30.00	0.01

NexSupply achieves the lowest total cost (¥41,400), marginally below the cost-only Single-Objective DP (¥41,320), while using fewer ordering days on average (0.73 versus 1.47). Both are dramatically cheaper than Cost-Priority (¥1,109,123 with 30 ordering days) and Management-Priority (¥745,815), and well below EOQ (¥150,444) and GA (¥221,406), as visualized in Figure 5(a). The result confirms that, under realistic, sparse maintenance demand, the composite policy attains near-cost-optimal expenditure with very few replenishments.

Figure 5(b) plots NexSupply's daily cost over the horizon, showing the characteristic lumpy profile of spare-part demand: most days incur no cost, punctuated by a few replenishment events (notably around days 8–10, 13, and 24). For example, the expansion-joint material is replenished only once, on day 2 (7 units), rather than through repeated small orders—illustrating how NexSupply concentrates procurement into a few well-timed actions.

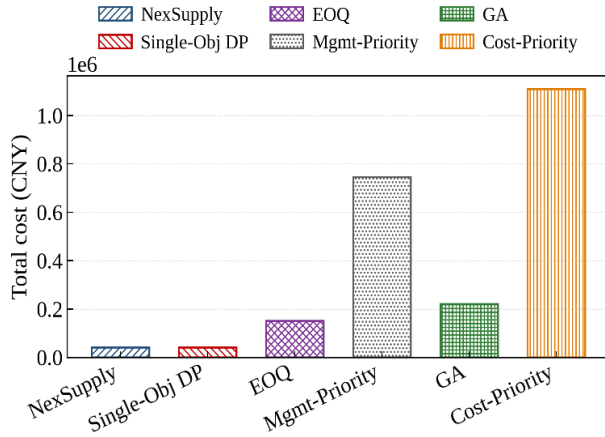


Figure 5(a). Total cost by method in the real-world case study.

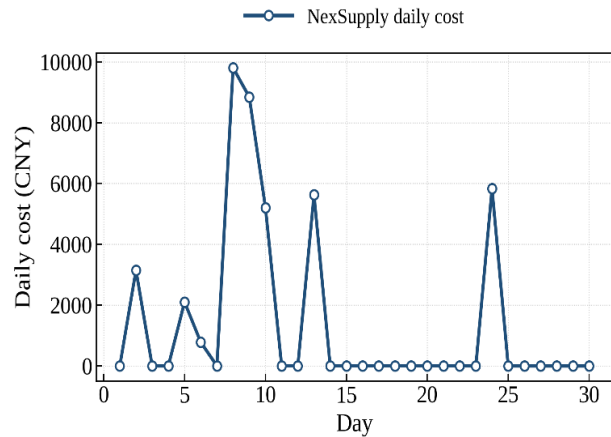


Figure 5(b). NexSupply daily cost over the 30-day horizon.

5.7. Computational Scalability

We measure runtime as the planning horizon T grows from 30 to 180 days ($m = 20$) and as the number of materials m grows from 10 to 100 ($T = 90$). Figure 6(a) and 6(b) (log scale) show that NexSupply scales linearly in both dimensions, consistent with its $O(T \cdot \sum_j K_j)$ complexity: runtime rises from 2.2 ms to 9.0 ms as T increases, and from 2.2 ms to 22.9 ms as m increases. NexSupply and Single-Objective DP have essentially identical runtimes, confirming that the frequency penalty adds negligible overhead.

GA is one to two orders of magnitude slower (104–611 ms over the T range and up to 1226 ms at $m = 100$), while the rule-based EOQ and Cost-Priority policies are fastest (sub-millisecond) but, as shown above, far costlier or far more order-intensive. NexSupply therefore offers near-cost-optimal solutions at millisecond speed, remaining practical for large instances.

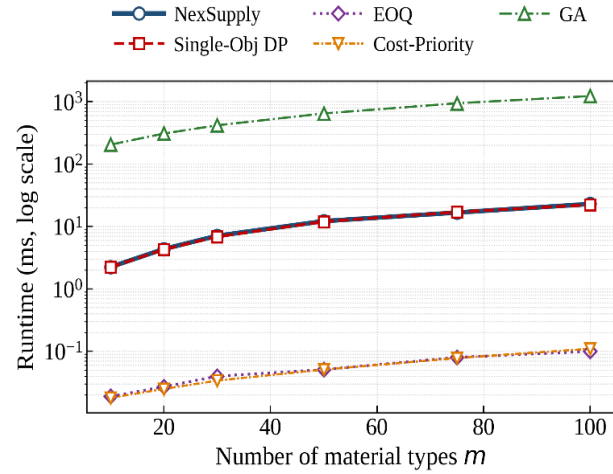
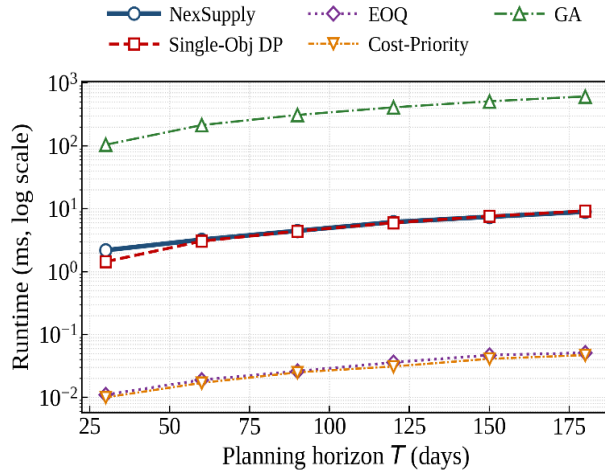


Figure 6(a). Runtime versus planning horizon T (log scale).

Figure 6(b). Runtime versus number of material types m (log scale).

5.8. Sensitivity Analysis

Finally, we sweep three key parameters one at a time around the baseline ($m = 15$, $T = 90$). Varying the holding-rate multiplier (0.5 – $4\times$) raises NexSupply's ordering frequency modestly from about 4.3 to 9.4 days (Figure 8(a)), as higher holding cost makes it economical to order more often and hold less; total cost stays nearly flat (¥6.09M–¥6.11M). Varying the shortage-rate multiplier (0.5 – $4\times$) leaves NexSupply's cost and frequency essentially unchanged (Figure 8(b)), indicating that its safety-stock buffer already avoids stockouts.

The frequency-penalty weight λ is the central design parameter. Figure 7 shows the resulting trade-off: at $\lambda = 0$, NexSupply reduces exactly to Single-Objective DP (frequency ≈ 47.9 , cost ¥6.07M); as λ increases, ordering frequency falls steeply—to 9.4 at $\lambda = 100$, 5.3 at $\lambda = 400$, and 3.9 at $\lambda = 1500$ —while total cost rises by less than 1% (to ¥6.11M). This confirms that a clean Pareto trade-off is available: a large reduction in ordering activity is purchased at negligible cost, and the baseline $\lambda = 400$ sits in the favorable knee of this curve.

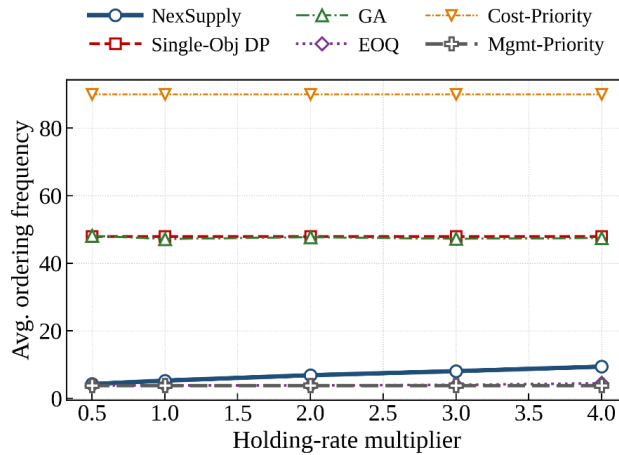


Figure 7. Ordering frequency versus frequency-penalty weight λ . At $\lambda = 0$, NexSupply coincides with Single-Objective DP; increasing λ sharply reduces ordering frequency at negligible additional cost.

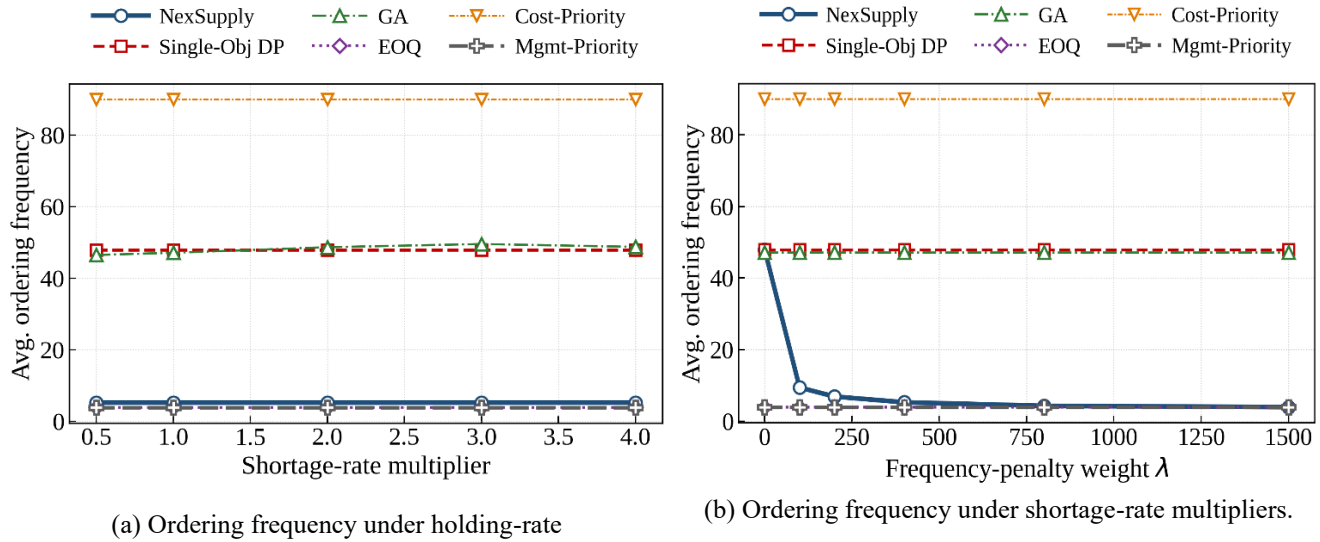


Figure 8. Ordering frequency

6. DISCUSSION

The experimental results in Section 5 admit a single, consistent explanation for NexSupply's behavior across all conditions. Because the ordering-frequency penalty is internalized directly into the per-period transition cost of the dynamic program, rather than evaluated after the fact, the optimizer can suppress unnecessary replenishment days and select cost-optimal order quantities on the days it orders. NexSupply costs no more than the single-objective dynamic program's cost-only optimum, but orders are about an order of magnitude smaller. As shown in Figure 7, at $\lambda = 0$ the method drops to the cost-only optimum, and as λ grows, the number of orders drops steeply while total cost rises by less than 1%. The tradeoff is highly asymmetric, and the practical consequence is that a large reduction in managerial overhead can be achieved at a small cost.

These results directly relate to the inventory and lot-sizing literature discussed in Section 2. Classical models such as Wagner–Whitin dynamic lot-sizing and (s, S) or (R, Q) reorder-point policies minimize procurement cost alone and deal with ordering frequency implicitly via fixed setup costs. NexSupply differs in that the number of orders is itself a first-class objective, modeled within the same dynamic program that minimizes costs. Comparison with EOQ and cost- and management-priority heuristics gives an idea of what this buys in practice: the rule-based policies either over-order (cost-priority, near-daily replenishment) or over-stock (management-priority, bulk batching) and are dominated by NexSupply on the combined cost-and-frequency criteria. The genetic algorithm (a modern metaheuristic baseline) approaches NexSupply on cost but has 1 to 2 orders of magnitude higher runtime and far more ordering days, suggesting the value of an exact structure-exploiting formulation.

A cautionary note: NexSupply is not always the single cheapest policy in every situation. In demand-fluctuation experiments (Figure 4) and at some demand levels (Figure 2), its cost essentially matches the single-objective dynamic program (i.e., it does not undercut every benchmark), and the cost-priority policy can become relatively less expensive under high demand intensity. This is expected and consistent with the design: NexSupply does not claim global cost minimization against all policies, but rather near cost-optimal expenditure with a drastically reduced ordering frequency. The composite objective is reported as an indicator of this balance and not a quantity that NexSupply minimizes against all alternatives. As a whole, it is not so much novelty at the algorithmic level as the problem-specific framing of a problem-specific trade-off implicit in the inventory literature, but that we take ordering frequency as a first-class goal within an exact finite-horizon dynamic program, and characterize the cost-frequency frontier in terms of material count, demand level, demand-cycle length, and demand fluctuation on real city water supply spare-part data. NexSupply transforms a management problem handled by arbitrary judgment into a transparent, reproducible, and parameter-controllable decision. We consider this explicit operational framing and its empirical description (and not the dynamic programming machinery as a whole) as the main contribution, and one directly applicable to other maintenance-inventory environments with intermittent demand. Given this distinction, we avoid the over-claiming that affected earlier versions of this work and treat the contribution as a good trade-off rather than universal dominance.

The scaling results (Figure 6) further clarify the method. The dynamic program scales linearly in both the planning horizon and the number of materials, solving cases with up to 100 materials in 10ms, which is good for periodic procurement planning in municipal water-supply operations. Combined with the tunable weight λ , NexSupply can be used for interactive decision support: a planner can adjust λ to any point on the cost–frequency curve and obtain the correct replenishment schedule essentially in seconds.

Some limitations temper these conclusions and motivate future work. First, materials are separable, and we do not model cross-material coupling (shared procurement budget, joint-ordering discount, shared storage contention), and such coupling would break the per-material decomposition and require a higher-dimensional state or an additional layer. Second, demand is treated as an exogenous per-period input: NexSupply optimizes over a given demand path but does not forecast demand, so deployment performance depends on the quality of the upstream demand estimates. Third, although runtime scales linearly, the per-material state space grows with storage capacity, which could be important for materials with very large capacity limits. These limitations are revisited as directions for future work in the Conclusion.

7. CONCLUSION

This study presented NexSupply, a multi-objective finite-horizon dynamic-programming framework for spare-part inventory optimization in urban water supply systems under demand and equipment-lifecycle uncertainty. The work offers three principal contributions:

- (1) A weighted-sum scalarization that internalizes the ordering-frequency objective directly into the dynamic-programming state transition through a penalty weight λ , explicitly capturing the trade-off between operational cost and managerial efficiency; setting λ to zero recovers the conventional cost-only policy exactly.
- (2) A tailored algorithm design featuring a per-material (period, inventory) state representation and domain-informed constraints, with the optimal replenishment trajectory recovered by exact backtracking, enabling adaptation to heterogeneous demand patterns and equipment lifecycles.
- (3) Extensive experimental data on synthetic instances and a real-world case study comparing cost-priority, management-priority, EOQ, genetic-algorithm, and single-objective dynamic-programming baselines show that NexSupply attains near cost-optimal total cost (indistinguishable from the cost-only optimum) while reducing order days by around 90% compared to the cost-priority policy and scaling linearly in both the planning horizon and the number of materials.

Building on the limitations discussed in Section 6, natural directions for future work include learning-based demand forecasting (e.g., LSTM or transformer models) upstream of the optimizer, extensions to coupled multi-material constraints, and deep reinforcement-learning and robust-optimization variants.

REFERENCES

- Teng, H., Xie, Z., Yue, X., and Cao, W. (2022) A flexible integrated scheduling algorithm for dynamic determination processing equipment considering the root-subtree vertical and horizontal pre-scheduling. *IEEE Access*, 10, 132298–132314.
- Ren, J., Cheng, Y., Xiang, F., and Tao, F. (2023) Platform-based manufacturing service collaboration: A supply-demand aware adaptive scheduling mechanism. *IEEE Transactions on Industrial Informatics*, 19(2), 1768–1777.
- Zhang, F., Mei, Y., Nguyen, S., and Zhang, M. (2023) Multitask multiobjective genetic programming for automated scheduling heuristic learning in dynamic flexible job-shop scheduling. *IEEE Transactions on Cybernetics*, 53(7), 4473–4486.
- Liu, S., Li, L., Zhang, L., and Shen, W. (2024) Game theory-based dynamic event-driven service scheduling in cloud manufacturing. *IEEE Transactions on Automation Science and Engineering*, 21(1), 618–629.
- Li, Z., Sang, H., Pan, Q., Gao, K., Han, Y., and Li, J. (2023) Dynamic AGV scheduling model with special cases in matrix production workshop. *IEEE Transactions on Industrial Informatics*, 19(6), 7762–7770.
- Qi, Z., and Zhang, R. (2024) Coordinated scheduling of micro-grid combined heat and power based on dynamic feedback correction and virtual penalty cost. *IEEE Transactions on Power Systems*, 39(3), 4975–4986.
- Lei, K., Guo, P., Wang, Y., Zhang, J., Meng, X., and Qian, L. (2024) Large-scale dynamic scheduling for flexible job-shop with random arrivals of new jobs by hierarchical reinforcement learning. *IEEE Transactions on Industrial Informatics*, 20(1), 1007–1018.

- Mishra, S., and Liang, J.-M. (2023) Dynamic sleep scheduling for wireless TP sensor transmissions based on reinforcement learning. *IEEE Sensors Letters*, 7(11), 1–4.
- Liu, C.-L., and Huang, T.-H. (2023) Dynamic job-shop scheduling problems using graph neural network and deep reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53(11), 6836–6848.
- Wang, K. (2024) Dynamic on-demand collaborative charging scheduling based on game theory. *IEEE Journal of the Electron Devices Society*, 12, 824–830.
- Ali, K. B., Bechikh, S., Louati, A., Louati, H., and Kariri, E. (2024) Dynamic job shop scheduling problem with new job arrivals using hybrid genetic algorithm. *IEEE Access*, 12, 85338–85354.
- Ma, Y. (2022) Optimal dynamic transmission scheduling for wireless networked control systems. *IEEE Transactions on Control Systems Technology*, 30(6), 2360–2376.
- Hajibabai, L., and Mirheli, A. (2022) A game-theoretic approach for dynamic service scheduling at charging facilities. *IEEE Transactions on Intelligent Transportation Systems*, 23(12), 23919–23932.
- Nguyen, S., Thiruvady, D., Zhang, M., and Alahakoon, D. (2022) Automated design of multipass heuristics for resource-constrained job scheduling with self-competitive genetic programming. *IEEE Transactions on Cybernetics*, 52(9), 8603–8616.
- You, W., Xu, Z., and Zhao, S. (2024) A two-layer approach for the decentralized multi-project scheduling problem sharing multi-skilled staff. *IEEE Access*, 12, 29204–29221.
- Zhang, K., Gao, Q., Lü, J., Ogorzałek, M., and Deng, Y. (2024) A subgraph-based hierarchical Q-learning approach to optimal resource scheduling for complex industrial networks. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 71(5), 2146–2156.
- Yu, Y., Xu, Z., and Zhao, S. (2023) A two-stage algorithm based on 12 priority rules for the stochastic distributed resource-constrained multi-project scheduling problem with multi-skilled staff. *IEEE Access*, 11, 29554–29565.
- Mao, H., Yuan, J., Mao, Y., and Zhang, S. (2024) Efficient priority rules for resource allocation of stochastic decentralized multi-project scheduling problem. *IEEE Access*, 12, 112729–112741.
- Sun, X., Sang, H., Zhao, Y., and Wu, W. (2024) A hybrid multi-threaded parallel iterated greedy algorithm for distributed flowshop group scheduling problems with preventive maintenance. In *Proc. 2024 7th Int. Conf. Adv. Algorithms Control Eng. (ICAACE)*, Shanghai, China, 117–121.
- Su, C. (2021) Research on electrolytic aluminum pot tending machine scheduling based on improved greedy algorithm. In *Proc. 2021 Int. Symp. Comput. Technol. Inf. Sci. (ISCTIS)*, Guilin, China, 303–307.
- Deng, J., Yang, R., Liang, J., Wu, Y., and Wang, H. (2024) Research on models based on greedy algorithms, dynamic programming algorithms and grid discretization algorithms. In *Proc. 2024 IEEE 2nd Int. Conf. Control, Electron. Comput. Technol. (ICCECT)*, Jilin, China, 847–852.
- Xue, X., Ai, X., Fang, J., Jiang, Y., Cui, S., Wang, J., Ortmeier, T. H., and Wen, J. (2024) Continuous approximate dynamic programming algorithm to promote multiple battery energy storage lifespan benefit in real-time scheduling. *IEEE Transactions on Smart Grid*, 15(6), 5744–5760.
- Alnahhal, Z., Shaaban, M. F., Hamouda, M. A., Majozi, T., and Bardan, M. A. (2023) A water-energy nexus approach for the co-optimization of electric and water systems. *IEEE Access*, 11, 28762–28770.
- Alhazmi, M., Dehghanian, P., Nazemi, M., and Oikonomou, K. (2023) Uncertainty-informed operation coordination in a water-energy nexus. *IEEE Transactions on Industrial Informatics*, 19(5), 6439–6449.

Eliades, D. G., Malialis, K., Vrachimis, S., and Polycarpou, M. M. (2025) Smart water networks as cyber-physical-socio-environmental systems. *IEEE Transactions on Industrial Cyber-Physical Systems*, 3, 95–105.

Kim, B. S., and Lee, W.-S. (2013) A multi-product dynamic inbound ordering and shipment scheduling problem at a third-party warehouse. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 20(1–2).

López-Garcés, J. A., Mendoza, A., and Masini-Aguilera, J. (2013) A classic and effective approach to inventory management. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 20(5–6).

Wagner, H. M., and Whitin, T. M. (1958) Dynamic version of the economic lot size model. *Management Science*, 5(1), 89–96.

Scarf, H. (1960) The optimality of (S,s) policies in the dynamic inventory problem. In *Mathematical Methods in the Social Sciences*. Stanford, CA, USA: Stanford University Press.

Silver, E. A., Pyke, D. F., and Peterson, R. (1998) *Inventory Management and Production Planning and Scheduling*, 3rd ed. New York, NY, USA: Wiley.

APPENDIX A – PER-MATERIAL DAILY DEMAND SERIES

The complete per-material daily demand series used in the real case study of Section 5.6 is provided here in the Appendix. Columns represent the 15 engineering materials and rows represent the 30-day planning time frame.

Table A1. Per-material daily demand series used in the real case study (15 engineering materials over a 30-day time frame).

All values represent units demanded; most days have zero demand, which is consistent with the intermittent and lumpy demand of spare parts.

Material ID Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1
14	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
15	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
16	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Material ID Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
22	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
23	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
24	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0