

An Intelligent Multi-Agent Architecture for Autonomous Supply Chain Decision-Making in Large-Scale Retail Systems

Suresh Kumar

Walmart, USA

Corresponding author: Suresh Kumar, Email: sureshkumar89@gmail.com

Large retail networks must reconcile local demand signals with shared inventory, transportation capacity, supplier commitments, and service objectives. A decision that improves one store can consume resources required elsewhere, while delayed information can turn individually reasonable replenishment actions into network instability. This paper proposes an intelligent multi-agent architecture that separates demand estimation, inventory planning, capacity coordination, execution authorization, and performance assurance. Intelligence is treated as the capacity to use observations, evaluate alternatives, and adapt within explicit authority boundaries; it does not require every agent to use a large language model. The architecture combines local decision autonomy with regional resource arbitration, a versioned operational state, and a deterministic execution gateway. A mathematical formulation defines inventory transitions, feasible allocations, cost, and service measures. An original synthetic experiment evaluates three interpretable policies across 320 store-product streams, four operating scenarios, and 30 paired random seeds. A further capacity sensitivity experiment tests five supply ratios. Under combined promotion and capacity disruption, adaptive proportional allocation achieves a mean normalized cost of 0.6404 artificial cost units per requested unit and a 93.397% fill rate. A greedy urgency coordinator produces a higher cost of 0.7129 and a lower fill rate of 92.603%. These findings reject the assumption that adding coordination automatically improves performance and motivate explicit policy comparison before deployment. The contribution is an auditable architecture, a bounded numerical demonstration, and a validation protocol for expanding toward large retail networks. The study does not claim a deployed system, trained multi-agent reinforcement learning model, or commercial savings. Its principal conclusion is that autonomous retail decision-making requires both effective resource allocation and independently enforced operational constraints.

Keywords: Multi-agent systems; retail supply chains; autonomous decision-making; inventory allocation; artificial intelligence; simulation; operational governance.

1 Introduction

1.1 Decision Coupling in Retail Networks

Retail supply chains contain decisions whose effects cross organizational and physical boundaries. A store replenishment request changes the workload at a distribution center. A distribution decision changes shelf availability, promised delivery dates, and the quantity that can be committed to other channels. A supplier delay changes the value of an expedited shipment, while a promotion can make a previously adequate allocation rule insufficient. These interactions create a control problem: observations arrive at different times, objectives are only partly aligned, and an action may remain in transit long after the decision that created it.

The term autonomous decision-making should therefore refer to a controlled operational process, rather than the unrestricted delegation of purchasing or movement authority to software. A useful system can recognize demand changes, generate feasible proposals, compare their consequences, and execute approved actions without requiring a person to review every routine replenishment. However, autonomy must remain conditional on data quality, policy validity, available capacity, and the magnitude of potential harm. Otherwise, faster decisions can simply accelerate the propagation of errors.

Multi-agent systems offer one way to organize this problem. Different agents can represent stores, product categories, distribution centers, or specialized analytical functions. The decomposition is attractive because different responsibilities have different information needs and decision frequencies. A demand agent may update an estimate daily, while an exception agent may respond immediately to a rejected shipment. The central research issue is how these agents should cooperate when locally desirable actions compete for the same resources.

1.2 Research Problem and Motivation

A common architectural weakness is to specify a list of intelligent agents without defining how their conflicting proposals become one executable plan. An inventory agent might recommend additional stock, a transport agent might propose consolidation, and a finance agent might reject the resulting expenditure. Without an allocation mechanism, a decision deadline, and an authoritative commitment record, the interaction remains a conversation rather than an operational control system. This paper addresses that gap through explicit proposal, allocation, reservation, authorization, and reconciliation stages.

The problem is particularly visible during scarcity. When aggregate demand exceeds available shipping capacity, accurate local forecasts alone cannot ensure high service. A coordinator must decide which requests to satisfy and when. An apparently sensible rule, such as fully serving the location with the lowest coverage first, may concentrate inventory in ways that increase aggregate shortages. The experimental component deliberately tests this possibility instead of assuming that more complex coordination is beneficial.

1.3 Contributions and Scope

The first contribution is a reference architecture that separates analytical advice from authority over physical commitments. The second is a formal decision model connecting local inventory positions to shared dispatch limits and network outcomes. The third is a reproducible synthetic experiment that demonstrates how a coordination rule can be safe with respect to capacity yet inferior in economic and service performance. The fourth is an implementation and evaluation protocol covering message semantics, failure recovery, policy promotion, and expansion to a larger network.

The scope is retail replenishment and allocation, with procurement, transportation, and assortment functions described as architectural extensions. The implemented experiment represents 40 stores and eight products through 320 inventory streams sharing one dispatch pool. It does not simulate a complete multi-echelon enterprise. Large-scale capability is consequently an architectural objective and an evaluation requirement, not an experimentally established result. This distinction is maintained throughout the paper.

2 Related Work and Research Positioning

2.1 Agent Oriented Supply Chain Systems

Fox, Barbuceanu, and Teigen [1] establish agent-oriented supply-chain management as a substantive research direction rather than a recent consequence of conversational AI. Their work provides a foundation for assigning responsibilities to interacting software entities. The relevant inheritance for this paper is organizational decomposition: separate agents can hold different knowledge and participate in coordinated planning. The proposed architecture does not claim to originate that idea. Its emphasis is the operational boundary between an agent proposal and a durable resource commitment.

The practical motivation for revisiting agent architectures is that retailers can now receive frequent updates from transaction systems, fulfillment services, and logistics partners. More observations create opportunities for adaptive control, but they also increase the number of inconsistent or overlapping decisions. An architecture must therefore define the ownership of each quantity, the age of each observation, and the point at which an allocation becomes authoritative. These are design requirements derived for the present proposal rather than empirical conclusions attributed to an earlier system.

2.2 Reinforcement Learning and Inventory Decisions

Oroojlooyjadid and colleagues [2] examine a deep Q-network approach to the beer game, connecting reinforcement learning with decentralized inventory decisions and partial information. That work motivates learning policies that account for delayed consequences. It does not establish that a learned policy will dominate interpretable replenishment rules in every retail setting. Differences in topology, shortage treatment, information access, and cost structure can change the comparison.

Hubbs and colleagues [3] introduce OR-Gym to support reinforcement learning research on operations research problems, including inventory-related environments and comparisons with optimization or heuristic methods. Yang and colleagues [4] present MABIM, a multi-echelon, multi-commodity inventory benchmark designed to expose challenges such as scale, interactions, and nonstationarity. These efforts support the methodological principle that policy claims should be evaluated against a stated environment contract. The present experiment is a separate, smaller simulator and does not claim to reproduce either benchmark.

A recent preprint by Barati and Hu [5] explicitly emphasizes that information access, feasibility rules, topology, and KPI definitions influence inventory-policy rankings. Its status as a preprint is retained here. This manuscript follows the same general concern for comparison transparency by holding generated demand, dispatch limits, shortage treatment, and initialization constant across the three tested policies. The small simulator remains intentionally narrower than a broad benchmark suite.

2.3 Cooperative Learning and Constraints

Lowe and colleagues [6] develop an actor-critic approach for multi-agent settings in which training can use additional information about other agents. Rashid and colleagues [7] propose QMIX, using a

monotonic factorization of joint action values to support centralized training with decentralized execution. These methods inform possible future policy modules. Neither method is trained or evaluated in this paper, and the numerical results should not be interpreted as a comparison with them.

Achiam and colleagues [8] study constrained policy optimization, distinguishing reward improvement from constraint satisfaction during learning. The architecture adopts a related separation at execution time: a policy may recommend an action, but an independent gateway checks whether that action is authorized and feasible. This is an architectural design choice, not a transfer of the theoretical guarantees of constrained policy optimization to the proposed system.

2.4 Event Semantics and Risk Management

GS1 EPCIS 2.0 [9] provides a standard for expressing and sharing visibility events. It is relevant to identifying business objects, locations, and events that describe movement through a supply chain. A retailer implementing the proposed architecture could map its physical visibility events to that standard while retaining separate internal records for forecasts, proposals, and reservations. Conformance would require an implementation-specific assessment; no standards certification is claimed.

The NIST AI Risk Management Framework 1.0 [10] provides a voluntary structure for governing, mapping, measuring, and managing AI risks. Its relevance here is the need to treat operational performance, authority, monitoring, and accountability as related concerns. The concrete gateway, policy registry, and rollback mechanisms described later are the paper's proposed implementation choices. They are not presented as mandatory controls specified by NIST.

2.5 Research Gap and Positioning

Table 1 summarizes the boundaries of the literature used to position the contribution. The review is a targeted narrative review of foundational and directly relevant work. It is not a systematic review, does not claim exhaustive coverage through 2026, and does not infer the absence of competing architectures from a limited source selection. Novelty is therefore stated narrowly: the manuscript integrates an explicit execution contract with a reproducible adverse coordination example and a staged retail validation design.

Table 1 Literature positioning and contribution boundaries

Research strand	Representative source	Relevance to this paper	Boundary
Agent organization	Fox et al. [1]	Decomposition of supply chain responsibilities	Agent concepts are established
Inventory learning	Oroojlooyjadid et al. [2]	Delayed and decentralized inventory control	No DQN reproduced here
Benchmark design	Hubbs et al. [3]; Yang et al. [4]; Barati and Hu [5]	Explicit environment and evaluation contracts	This simulator is independent
Cooperative learning	Lowe et al. [6]; Rashid et al. [7]	Potential future learned policy modules	No MARL training performed
Constrained learning	Achiam et al. [8]	Separate performance from constraints	No inherited safety theorem
Events and governance	GS1 [9]; NIST [10]	Semantic and governance foundations	Proposed design is not certified

3 Research Method and Questions

3.1 Architecture Construction and Numerical Demonstration

The research follows an architecture-design and simulation approach. First, the retail decision problem is decomposed into responsibilities, information flows, and authority boundaries. Second, a simplified replenishment model is defined so that selected interactions can be executed and inspected. Third, three policies are compared using identical random demand paths. Finally, the observed weaknesses are translated into requirements for a more complete implementation. This sequence makes the numerical study a test of a narrow mechanism rather than a validation of every architectural component.

The evaluated mechanism is how inventory requests compete for a common dispatch budget. Demand estimation and allocation are separated so that the effect of replacing a static forecast can be distinguished from the effect of replacing proportional allocation. The adaptive proportional and coordinated urgency policies share the same forecasting rule and inventory target formula. Their difference is the rationing mechanism. The comparison therefore provides a controlled diagnostic of that mechanism under the stated assumptions.

The architecture itself is assessed through logical consistency, clearly assigned state ownership, and failure-case reasoning. These forms of assessment can identify missing responsibilities and unsafe transitions, but they cannot establish throughput, security resistance, or field effectiveness. Accordingly, implementation tests and pilot criteria are specified as future work rather than reported as completed experiments.

3.2 Research Questions

RQ1 asks how specialized agents can retain local decision responsibility while respecting shared physical and financial constraints. The architectural response is hierarchical allocation with a distinct execution authority. RQ2 asks whether a simple urgency coordinator improves outcomes compared with proportional allocation under common demand and capacity conditions. The numerical experiment addresses this question directly and produces a negative answer for the tested rule and scenarios.

RQ3 asks how performance should be measured so that aggregate efficiency does not hide poor service in individual store-product streams. The study reports fill rate, normalized cost, inventory, and the worst-stream fill rate within each replication. RQ4 asks what additional evidence is required before extending a bounded allocation experiment to autonomous control across a large retail enterprise. The response is a staged evaluation plan covering richer physical models, distributed execution, operational data, and monitored deployment.

These questions avoid an unsupported hypothesis that autonomy or multi-agent complexity must be beneficial. They also permit a useful conclusion when a coordination policy performs poorly. A system architecture should support rejecting a candidate policy; it should not require every newly introduced intelligent component to outperform its predecessor.

3.3 Evidence Classification

Three kinds of statements appear in the paper. Literature statements are linked to numbered references. Architectural proposals describe how a system could be organized and are not represented as deployed capabilities. Experimental statements are computed from the synthetic simulator described in Sections 8 and 9. This classification is necessary because a complete enterprise diagram can easily appear more empirically established than the code that supports it.

No retailer supplied transaction logs, warehouse data, supplier contracts, or customer records. All numerical demand is generated synthetically. No survey, interview, customer intervention, or experiment with human participants was conducted. The proposed governance process anticipates later organizational deployment, but the present paper makes no claim about employee adoption, customer satisfaction, or return on investment.

4 Retail System Requirements

4.1 Functional Responsibilities

The target architecture should estimate demand, maintain inventory positions, propose replenishment, allocate scarce resources, authorize commitments, observe execution, and learn from outcomes. Each responsibility must expose an input contract and an output contract. A forecast is not an order, and an order request is not a shipment. Treating these objects as interchangeable can double-count stock or create commitments unsupported by available capacity.

At store level, the system needs sellable on-hand inventory, outstanding allocations, expected receipts, reservations for customer orders, and known adjustments. At regional level, it needs capacity by facility and time window, destination eligibility, transport restrictions, and service priorities. At enterprise level, it needs spending limits, policy versions, supplier eligibility, and the ability to suspend automation. Different deployments may place these responsibilities in existing services rather than create a new service for every agent.

The architecture must support routine cases and exceptions. Routine replenishment can follow a preapproved policy within a local spending envelope. A request involving a new supplier, a disputed inventory record, or an unusually large exposure should be escalated. The threshold is a business design parameter and must be recorded explicitly. It should not be inferred opportunistically by a conversational model during execution.

4.2 Nonfunctional Requirements

Consistency requirements depend on the business object. Analytical forecasts can tolerate eventual consistency because they are recommendations about future demand. Capacity reservations require stronger concurrency control because the same remaining capacity cannot be sold or assigned twice. Audit records require durability because a later investigation must reconstruct the decision even after the forecasting model has changed.

Availability should also be scoped by function. Losing a demand service should not prevent a store from reading its last approved target or recording a sale. Losing the capacity authority should prevent new unverified network commitments while allowing already authorized work to proceed where operationally appropriate. This partition is preferable to a single universal failure mode in which every component either remains fully autonomous or stops entirely.

Latency targets should follow the decision horizon. A nightly procurement plan, a same-day dispatch allocation, and an urgent fulfillment exception have different deadlines. The paper does not invent a universal millisecond service-level objective. Instead, it recommends measuring event age, queue delay, solver time, authorization time, and acknowledgement time separately. Their combined distribution determines whether the system meets an operational cutoff.

Table 2 Proposed system requirements and evidence needed

Requirement	Proposed mechanism	Evidence required before deployment
No capacity oversubscription	Atomic reservation by resource owner	Concurrent reservation tests
Traceable decisions	Versioned proposals and event identifiers	Complete replay of sampled orders
Bounded autonomy	Policy-specific authority envelopes	Approval and rejection tests
Local resilience	Cached approved fallback rules	Partition and recovery exercises
Effective allocation	Policy comparison under common inputs	Out-of-sample cost and service results
Large-scale operation	Regional partitions and sparse communication	Load tests at target network size

4.3 Decision Authority and Human Roles

People remain responsible for defining service targets, approving policy promotion, deciding acceptable exposure, and resolving material exceptions. This responsibility should be represented in the system through named roles and explicit approval records. A supervisor who can approve emergency transfers should not automatically have authority to change supplier payment details or model training permissions.

The practical aim is to reduce repetitive review while preserving review where judgment matters. An exception should explain the proposed action, the constraint that triggered review, the alternative of doing nothing, and the operational deadline. A long natural-language narrative is less useful than a concise statement tied to verifiable quantities. If the record cannot identify the affected stock, capacity, and policy, the system should not present the action as ready for approval.

5 Proposed Multi Agent Architecture

5.1 Structural Overview

Figure 1 presents the proposed architecture. Operational sources feed a validated event layer, which updates a versioned state service. Specialized agents read scoped views of that state and submit proposals. A regional coordinator reconciles proposals that compete for shared resources. A policy and safety gateway checks the selected plan before execution adapters create commitments in operational systems. Execution observations return through the event layer, closing the control loop.

The diagram deliberately distinguishes the analytical path from the execution path. A demand agent can revise a forecast without acquiring warehouse authority. A coordinator can select a preferred plan without creating a purchase order. The execution gateway is the final authority check, but it is not the only protection: data validation, constrained proposals, and reservation ownership reduce the number of unsafe requests reaching that point.

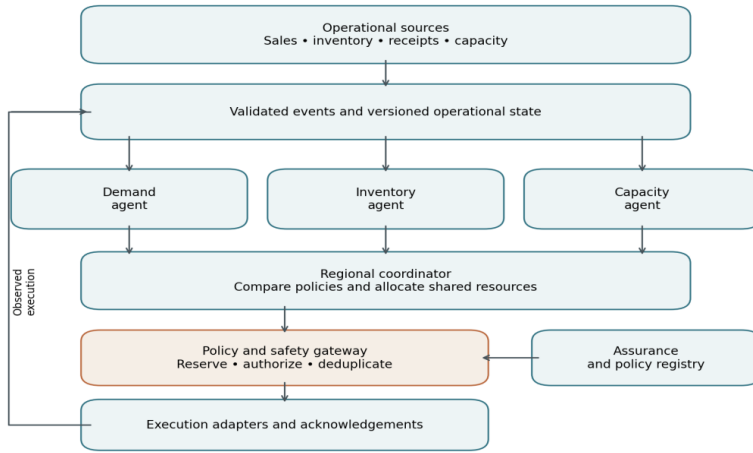


Figure 1 Proposed multi-agent architecture and execution boundary. Solid arrows indicate the main decision flow; the feedback path returns observed execution to the operational state. This is a conceptual design, not a deployed-system diagram.

The state service is logically shared but need not be implemented as one physical database. Inventory ownership can be partitioned by location and product, while capacity ownership can be partitioned by facility and dispatch window. What matters is that each mutable resource has an unambiguous owner and a version against which a decision is evaluated. Agents can use cached views for analysis, but a commitment must be checked against the authoritative resource state.

5.2 Agent Responsibilities

The demand agent produces estimates and uncertainty descriptors, including the observation window, product identifier, location, and model version. The inventory agent converts these estimates into bounded replenishment requests. The capacity agent exposes the available quantities and operating restrictions for a dispatch period. The coordination agent selects a feasible joint allocation. The assurance agent compares predicted and observed outcomes and recommends keeping, limiting, or withdrawing a policy.

Procurement and transportation agents extend the architecture when the physical model includes supplier lead times, order minimums, vehicle routes, and delivery appointments. They should not be represented as omniscient components. A procurement agent may know approved suppliers and contractual terms but have incomplete visibility into actual production. A transport agent may know scheduled vehicle capacity but require new observations to detect a late arrival.

An optional explanation agent can summarize structured decision records for planners. It should not become the source of numerical truth. Explanations must be generated from the same accepted proposal, cost estimates, constraints, and execution status that the gateway used. If a language model is used, it should be prevented from transforming retrieved supplier text or free-form messages into new authority over orders.

Table 3 Agent inputs outputs and authority

Agent	Principal input	Principal output	Authority
Demand	Demand history and context	Forecast with provenance	Analytical only
Inventory	Forecast and inventory position	Replenishment request	Within proposal limits
Capacity	Facility and transport availability	Resource envelope	Publishes owned resource state
Coordinator	Requests and resource envelope	Joint allocation proposal	Selection without execution
Execution gateway	Allocation and approvals	Authorized commitment	Explicit scoped write permission
Assurance	Outcomes and policy registry	Evaluation and alerts	May request suspension
Explanation	Structured decision record	Planner explanation	No transaction authority

5.3 Hierarchy and Local Autonomy

A flat system in which every agent communicates with every other agent becomes difficult to reason about as the number of stores and products grows. The proposed hierarchy groups local inventory decisions under regional resource coordinators. Enterprise governance supplies common policy rules and exposure limits, while regional coordinators arbitrate resources whose coupling is local. Cross-region transfers are treated as explicit interregional decisions rather than hidden side effects of local replenishment.

Local autonomy means an agent can select an action within its approved envelope using the observations available to it. It does not mean the agent can unilaterally consume shared dispatch capacity. For uncoupled actions, such as refreshing a forecast, local execution can be immediate. For coupled actions, such as requesting a shipment from a constrained facility, the agent submits a proposal and waits for an allocation or a reasoned rejection.

The hierarchy must not suppress information that matters for joint decisions. A regional coordinator should receive enough detail to distinguish urgent shortages from speculative buffer requests. It need not receive every raw transaction. Summary messages can include requested quantity, expected shortage, inventory coverage, service priority, latest acceptable arrival, and the uncertainty associated with the forecast. The usefulness of each field should be tested rather than assumed.

5.4 Decision Lifecycle

Figure 2 illustrates the lifecycle of a replenishment decision, including rejection and recovery branches. A proposal is first associated with a state version and an expiry time. The coordinator evaluates it against competing requests. The selected allocation is reserved before authorization. If authorization succeeds, an execution adapter sends the commitment to the relevant operational system. An acknowledgment is recorded separately from a physical receipt because acceptance of an order does not mean goods have arrived.

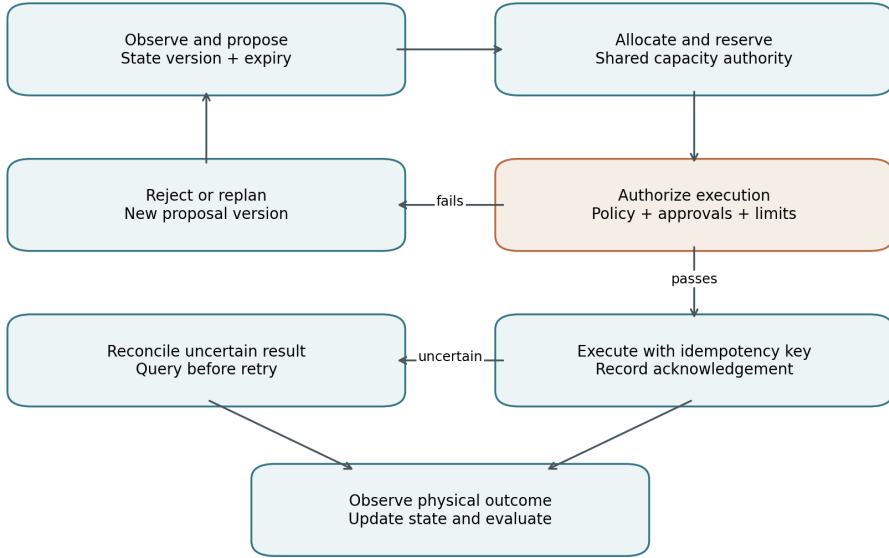


Figure 2 Proposed decision lifecycle with expiry rejection and reconciliation paths. Replanning creates a new versioned proposal; it does not silently alter a previously committed order.

A failed acknowledgement is an ambiguous state, not proof that execution did not occur. The adapter must query by idempotency key before retrying or compensating. Similarly, a late inventory correction can invalidate an uncommitted proposal but should not erase a historical decision. The system should preserve what was known when the original action was authorized and then create a corrective action if needed.

6 Formal Decision Model

6.1 State and Timing

Consider a set of store–product streams indexed by i and discrete periods indexed by t . The state includes sellable inventory, scheduled pipeline arrivals, demand estimates, and dispatch capacity. Let $I_{i,t}$ denote inventory immediately before the arrival event in period t . Let $A_{i,t}$ be inventory arriving at the start of that period, and let $D_{i,t}$ be requested demand. The fulfilled quantity is the smaller of available inventory and demand. Unfilled demand is treated as lost sales in the implemented experiment.

$$S_{i,t} = \min(I_{i,t} + A_{i,t}, D_{i,t}) \quad (1)$$

$$U_{i,t} = D_{i,t} - S_{i,t} \quad (2)$$

$$I_{i,t+1} = I_{i,t} + A_{i,t} - S_{i,t} \quad (3)$$

The notation in Equation 3 represents end-of-period carryover before the next arrival. Replenishment is decided after demand has been observed, and a dispatch decision $q_{i,t}$ arrives at the start of period $t + L$. In the experiment, L equals three periods. Because the decision occurs after current demand, the inventory position includes end-of-period stock and outstanding receipts due in subsequent periods. New requests do not become immediately available inventory.

This event order is consequential. A simulator that ships before observing demand solves a different information problem from one that ships after observing demand. A simulator that allows a shipment ordered at the end of day t to satisfy demand earlier on that day introduces future information and physically impossible service. The present model makes these boundaries explicit so that later policy comparisons can preserve them.

6.2 Inventory Targets and Requests

Let $m_{i,t}$ denote the estimate used for replenishment. The static policy uses the initial base mean for every period. The two adaptive policies update the estimate using an exponentially weighted average of observed requested demand with smoothing coefficient 0.2. All policies use the same four-period target structure and the same square-root buffer. The target is a heuristic rather than an optimized quantile for the full nonstationary environment.

$$m_{i,t} = 0.8m_{i,t-1} + 0.2D_{i,t} \quad (4)$$

$$B_{i,t} = 4m_{i,t} + 1.65\sqrt{4m_{i,t}} \quad (5)$$

$$r_{i,t} = \max(0, B_{i,t} - P_{i,t}) \quad (6)$$

Here $P_{i,t}$ is end-of-period on-hand inventory plus outstanding receipts, and $r_{i,t}$ is the requested replenishment quantity. The four-period target combines a three-period physical delay with a one-period review allowance. The multiplier 1.65 is a fixed design parameter. Because forecasts are smoothed and demand includes weekly variation, it must not be interpreted as guaranteeing a particular realized service probability.

The simulator allows fractional inventory and dispatches as a fluid approximation. Demand is integer-valued, but proportional rationing can create fractional replenishment. This simplifies comparisons of allocation logic while avoiding a rounding rule that would introduce an additional policy difference. Production implementation would require case-pack constraints, rounding, and a residue allocation rule, each of which should be evaluated separately.

6.3 Shared feasibility and objective

Every dispatch quantity must be nonnegative and no greater than the corresponding request. The sum across streams must not exceed the available dispatch capacity C_t . These restrictions define a feasible allocation set. Additional production constraints could include store storage limits, SKU eligibility, delivery windows, minimum order quantities, and financial exposure. They are architectural extensions rather than constraints silently included in the current experiment.

$$0 \leq q_{i,t} \leq r_{i,t}, \quad \sum_i q_{i,t} \leq C_t \quad (7)$$

$$J = \sum_t \left[0.1 \sum_i I_{i,t+1} + 5 \sum_i U_{i,t} + 0.2 \sum_i q_{i,t} \right] \quad (8)$$

Equation 8 uses artificial cost units: 0.1 per unit of end-of-period inventory, 5 per lost unit, and 0.2 per dispatched unit. These weights are selected for an illustrative shortage-sensitive inventory setting. They are not retailer accounting estimates and should not be converted directly into currency. All policies are assessed under the same weights.

Normalized cost divides J over the scoring window by requested demand over that window. This permits comparison across random networks with different base demand totals. Fill rate is fulfilled units divided by requested units. The lowest stream-level fill rate is also reported for each replication, followed by an average across replications. This quantity is not a percentile and should not be described as a worst-case guarantee over all possible demand paths.

Table 4 Mathematical symbols and units

Symbol	Meaning	Unit
$D_{i,t}$	Requested demand	Units per period
$S_{i,t}$	Fulfilled demand	Units per period
$U_{i,t}$	Lost demand	Units per period
$I_{i,t}$	Sellable inventory before arrivals	Units
$A_{i,t}$	Scheduled arrivals	Units per period
$P_{i,t}$	On-hand plus outstanding receipts after demand	Units
$m_{i,t}$	Replenishment demand estimate	Units per period
$r_{i,t}$	Replenishment request	Units
$q_{i,t}$	Accepted dispatch quantity	Units
C_t	Shared dispatch limit	Units per period
L	Dispatch-to-arrival delay	Periods
J	Holding shortage and dispatch cost	Artificial cost units

7 Coordination and Intelligence Mechanisms

7.1 Proportional Allocation

The proportional allocator accepts all requests when their sum fits within the dispatch limit. During scarcity, it applies the same acceptance fraction to every positive request. A stream requesting twice as much receives twice the allocation, regardless of its current inventory coverage. This rule is easy to implement and audit, and it avoids a full ranking of individual streams. It is not economically optimal in general because requested quantity is not necessarily proportional to marginal service value.

$$q_{i,t} = r_{i,t} \min \left(1, \frac{C_t}{\sum_j r_{j,t}} \right) \quad (9)$$

The implementation handles a zero aggregate request without division by zero. Because the scalar acceptance fraction lies between zero and one, nonnegative requests produce nonnegative dispatches bounded by the request. Summing Equation 9 gives a total no larger than capacity. This property is a direct consequence of the rule, not a result learned from the simulation.

Proportional allocation can also create undesirable incentives if agents can exaggerate their requests. In the numerical experiment, requests are generated by a fixed common formula, so strategic inflation is absent. A production system should impose request ceilings, record the target formula, and compare proposals with realized demand. Otherwise, the allocation mechanism may reward the agent that asks most aggressively rather than the location with the strongest operational need.

7.2 Greedy Urgency Coordination

The coordinated urgency policy ranks streams by inventory coverage, defined as inventory position divided by the current adaptive demand estimate. It then serves requests in ascending coverage order, fully satisfying each request until capacity is exhausted. Ties use stable stream order. The rule interprets low coverage as urgency and introduces a central comparison that proportional allocation does not perform.

$$\text{coverage}_{i,t} = \frac{P_{i,t}}{\max(m_{i,t}, \varepsilon)} \quad (10)$$

The resulting dispatch remains feasible because every accepted quantity is the minimum of the request and remaining capacity. However, feasibility does not imply that allocating a complete target replenishment to the most urgent stream maximizes marginal service. A full request can include buffer inventory whose value is lower than the first units needed by another stream. The greedy rule does not update relative coverage after each unit is assigned; it effectively makes large allocation blocks.

This distinction provides a mechanism for poor performance. A stream with low initial coverage can consume capacity sufficient to move it far above another stream that receives nothing. The next period may reverse their ranking, producing oscillating allocations. Because shipments arrive after a delay, a locally urgent replenishment can also become excess stock before it arrives. The experiment measures the consequences of this specific rule and does not generalize its weaknesses to every coordination method.

7.3 Candidate Improvements and Learned Modules

A more careful coordinator could allocate capacity incrementally according to marginal expected shortage reduction, use a rolling-horizon optimizer, or equalize projected coverage through a water-filling procedure. These are candidate improvements, not evaluated algorithms in the present results. Their comparison should preserve the same forecasting inputs and capacity limits so that any gain can be attributed to the allocation method rather than additional information.

A learned policy could propose target adjustments, urgency scores, or allocation weights. Keeping the action space bounded can make evaluation and execution checks easier than permitting unrestricted order generation. For example, a policy might select a multiplier within an approved interval around an interpretable base-stock target. The gateway would still enforce stock eligibility, resource constraints, and spending authority after the proposal is generated.

Training such a module would require an explicit objective, an environment reflecting the action delay, and held-out scenarios that include demand shifts and operational failures. The reward should not omit service or fairness penalties simply because those outcomes are difficult to measure. Conversely, encoding every business concern in a single reward can obscure nonnegotiable constraints. This architecture therefore retains constraints outside the learned model even when learning is introduced.

7.4 Model Selection and Policy Registry

Every candidate policy should have a registry entry containing its executable version, input schema, parameter values, training provenance when applicable, evaluation scenarios, and permitted scope. Promotion should be a documented decision supported by comparisons with the incumbent. A policy that improves average cost but materially worsens low-service locations should not be promoted without an explicit business decision about that tradeoff.

The adverse result in this paper illustrates why a registry must support rejection and rollback. The urgency rule is plausible enough to appear in an architectural proposal, but the measured comparison does not justify replacing proportional allocation with it. An assurance agent should be able to recommend retaining the simpler policy while recording the tested alternative and the reason it was rejected. Intelligence at the system level includes this capacity to decline an unhelpful adaptation.

8 Synthetic Experimental Design

8.1 Network and Demand Generation

The experiment contains 320 store–product inventory streams, interpreted as 40 stores with eight products each. Streams are flattened in store-major order. Each replication samples a base demand mean independently from a uniform distribution between 5 and 25 units per day. A shared weekly sinusoid with amplitude 0.12 modifies the daily mean. Requested demand is then drawn from a Poisson distribution with that mean. Conditional on the generated rates, demand draws are independent across streams and periods.

The term stationary scenario refers to the baseline scenario without promotion or capacity disruption. It retains deterministic weekly seasonality and is therefore not strictly stationary in the distributional sense. The label is used for readability in tables; the generating process is explicitly seasonal. Across a replication, the sampled base means remain constant. Across replications, both those means and demand realizations vary.

The promotion affects streams 0 through 79, corresponding to ten stores under the stated indexing convention. Their demand rate is multiplied by 1.8 for periods 70 through 99 inclusive. This is a demand surge rather than a fully modeled commercial promotion: price, margin, advertising, substitution, and customer response are not simulated. The system does not receive advance knowledge of the promotion. The adaptive estimators respond only after requested demand is observed.

8.2 Capacity and Event Timing

Normal dispatch capacity equals 1.10 times the sum of the sampled base means. The disruption scenario multiplies that capacity by 0.65 during periods 85 through 109 inclusive. Thus the capacity reduction is 35% relative to the normal dispatch limit, not a reduction to 35%. The combined scenario overlaps the promotion and disruption, creating a period in which selected streams face increased demand while the common dispatch pool contracts.

Inventory begins at the four-period base-demand target including the square-root buffer, and the pipeline begins empty. Every daily cycle processes arrivals, fulfills demand, updates the adaptive estimate, calculates outstanding inventory position, generates requests, allocates dispatch capacity, and schedules accepted shipments to arrive three days later. The physical supply source is unlimited apart from the daily dispatch cap. There is no upstream stock balance, supplier production process, or vehicle route in the simulator.

All policies receive the same generated requested demand, including the portion that cannot be fulfilled. This is a strong information assumption: observed sales alone would censor demand during stockouts. The study uses full requested demand to isolate allocation effects from demand reconstruction. Later experiments should test incomplete demand observation and estimation error because that change may alter both target behavior and policy ranking.

Table 5 Executed simulation configuration

Parameter	Value	Interpretation
Inventory streams	320	40 stores $\times$ 8 products
Horizon	180 periods	One period interpreted as one day
Scored periods	30 through 179	First 30 periods excluded
Replication seeds	100 through 129	30 paired replications
Base demand	Uniform 5 to 25	Sampled once per stream and seed
Weekly amplitude	0.12	Sinusoidal rate multiplier
Promotion	$\times 1.8$ in periods 70–99	First 80 streams only
Normal capacity	$1.10 \times$ total base mean	Shared dispatch pool
Disruption	$\times 0.65$ in periods 85–109	35% loss of normal capacity
Lead time	3 periods	End-period dispatch to start-period receipt
Forecast smoothing	0.20	Requested demand observed after service
Target	$4m + 1.65\sqrt{(4m)}$	Same structure for every policy
Cost weights	0.1 holding; 5 shortage; 0.2 dispatch	Artificial units

8.3 Policies and Comparison Controls

The static proportional policy combines an unchanged base-demand estimate with proportional rationing. The adaptive proportional policy replaces only the estimate with the exponentially weighted update. The coordinated urgency policy uses the same adaptive estimate and target formula but replaces proportional rationing with the greedy coverage ranking. All three policies receive identical physical constraints and demand realizations within each seed and scenario.

There are four scenarios, three policies, and 30 seeds, yielding 360 primary simulation runs. A separate sensitivity experiment evaluates the two adaptive policies at normal-capacity ratios of 0.90, 1.00, 1.10, 1.20, and 1.30 under the combined scenario, again with 30 seeds. This contributes 300 runs. The 1.10 sensitivity setting deliberately repeats the corresponding primary setting as a consistency check; it does not provide 60 independent replications for that case.

No parameters are optimized on the test outcomes. The rule definitions and cost weights are fixed for the executed comparison. The policies are heuristics, so there is no training set or training convergence curve. Describing the results as trained AI performance would be incorrect. The architecture can accommodate learning, but the executed study evaluates an interpretable adaptive component and a coordination mechanism.

8.4 Metrics and Uncertainty

The primary metric is normalized total cost over the scoring window. Secondary metrics are aggregate fill rate, mean end-of-period inventory, and the minimum stream-level fill rate within a replication. The analysis first computes each metric per replication and then averages across the 30 seeds. It does not treat every store-day observation as an independent statistical replicate.

For selected comparisons, paired differences are computed using the same seed for the two policies. A two-sided 95% Student *t* interval uses 29 degrees of freedom. These intervals characterize variation under the selected synthetic generator. They do not account for uncertainty in the structural assumptions, cost weights, or business interpretation. A narrow interval can coexist with low external validity when the simulator omits important operational mechanisms.

The initial 30 periods are excluded to reduce initialization effects. No formal stationarity or warm-up-length study is claimed. The scoring window includes the promotion, disruption, and post-event recovery. Inventory and dispatch costs are charged during that window, including shipments scheduled near the horizon end. There is no terminal salvage value. A longer horizon or terminal adjustment could affect absolute costs and should be included in subsequent robustness work.

9 9 Results

9.1 Overall Policy Comparison

Table 6 reports the means across 30 replications for each scenario. Figure 3 compares normalized cost with uncertainty intervals. In the baseline seasonal scenario, all policies achieve high aggregate service, and cost differences are small. Under promotion, adaptive proportional allocation improves normalized cost relative to the static rule. Under capacity disruption, both proportional policies remain close in cost, while the urgency coordinator is more expensive.

Table 6 Mean outcomes across 30 synthetic replications

Scenario	Policy	Cost / unit	Fill %	Inventory	Worst fill %
Stationary	Static	0.3828	99.995	8,761.0	99.454
Stationary	Adaptive	0.3789	99.945	8,449.8	98.594
Stationary	Urgency	0.3819	99.888	8,459.6	97.929
Promotion	Static	0.4443	98.275	8,059.5	91.573
Promotion	Adaptive	0.4136	98.526	7,119.1	95.983
Promotion	Urgency	0.4654	97.839	8,058.3	94.138
Disruption	Static	0.5250	96.283	7,034.7	94.038
Disruption	Adaptive	0.5272	96.144	6,809.1	94.667
Disruption	Urgency	0.5986	95.201	8,061.0	92.631
Combined	Static	0.6427	93.538	6,609.6	84.950
Combined	Adaptive	0.6404	93.397	6,151.1	90.962
Combined	Urgency	0.7129	92.603	7,866.5	86.704

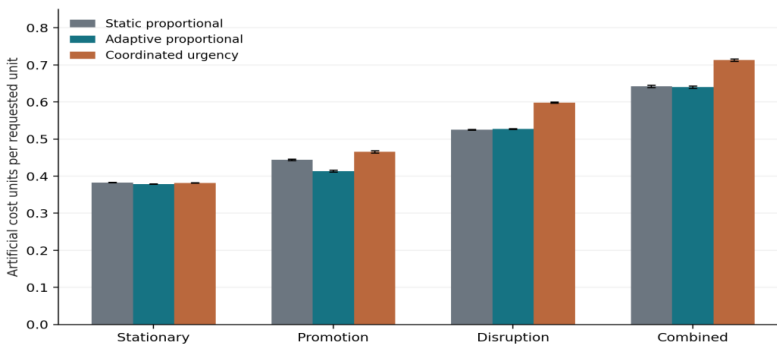


Figure 3 Normalized cost by policy and scenario. Bars show means across 30 synthetic replications; error bars show 95% Student t intervals for each policy mean. Stationary denotes the baseline seasonal scenario without a promotion or capacity shock.

In the combined scenario, mean normalized cost is 0.6427 for static proportional allocation, 0.6404 for adaptive proportional allocation, and 0.7129 for coordinated urgency. The urgency policy therefore increases cost by approximately 11.3% relative to adaptive proportional allocation under these assumptions. Aggregate fill rate also declines, from 93.397% under adaptive proportional allocation to 92.603% under coordinated urgency. This is not a tradeoff in which higher cost purchases better aggregate service.

The static policy has slightly higher aggregate fill than the adaptive proportional policy in the combined scenario, but it holds more inventory and has substantially poorer service in its worst-served stream. The result demonstrates why a single KPI cannot establish dominance. Adaptive demand estimation changes where inventory is held and how shortages are distributed. A manager choosing between these policies would need to evaluate both the network total and the distribution of service across locations and products.

9.2 Paired Differences and Interpretation

Table 7 Paired differences in the combined scenario urgency minus adaptive

Metric	Mean difference	95% paired t interval
Cost per requested unit	+0.0725	[+0.0717, +0.0733]
Fill rate percentage points	-0.7941	[-0.8077, -0.7804]
Mean inventory units	+1715.4621	[+1700.3042, +1730.6200]
Worst-stream fill percentage points	-4.2581	[-4.5097, -4.0064]

Table 7 reports coordinated urgency minus adaptive proportional allocation in the combined scenario. Positive cost and inventory differences are unfavorable for urgency, while negative fill-rate differences indicate poorer service. Pairing reduces noise from changes in the sampled network and demand path because each pair shares both. The comparison supports rejecting this urgency rule as a replacement within the tested environment.

It does not establish that all central coordination is inferior. Both allocation policies are centrally applied to a shared capacity constraint in the simulator; the labels distinguish their rules rather than distributed software topology. The experiment therefore compares allocation mechanisms, not a real distributed multi-agent deployment against a centralized enterprise optimizer. The architectural coordinator remains useful as an explicit place to evaluate and select among mechanisms.

The numerical result is consistent with a concentration problem: filling complete requests in rank order leaves some streams with large replenishment blocks and others with no dispatch. Mean inventory is higher even though aggregate service is lower. This pattern supports investigating the greedy block assignment as a cause, but it is not a complete causal decomposition. A unit-by-unit allocation trace or a comparison with water-filling would be needed to establish how much of the gap is attributable to that specific mechanism.

9.3 Temporal Service Behavior

Figure 4 shows daily network fill rates averaged across seeds in the combined scenario. The promotion and disruption windows are indicated separately. The curves provide context that the overall scoring-window averages cannot show: a controller may perform adequately before the event, deteriorate while capacity is constrained, and retain excess inventory after demand returns toward its base level.

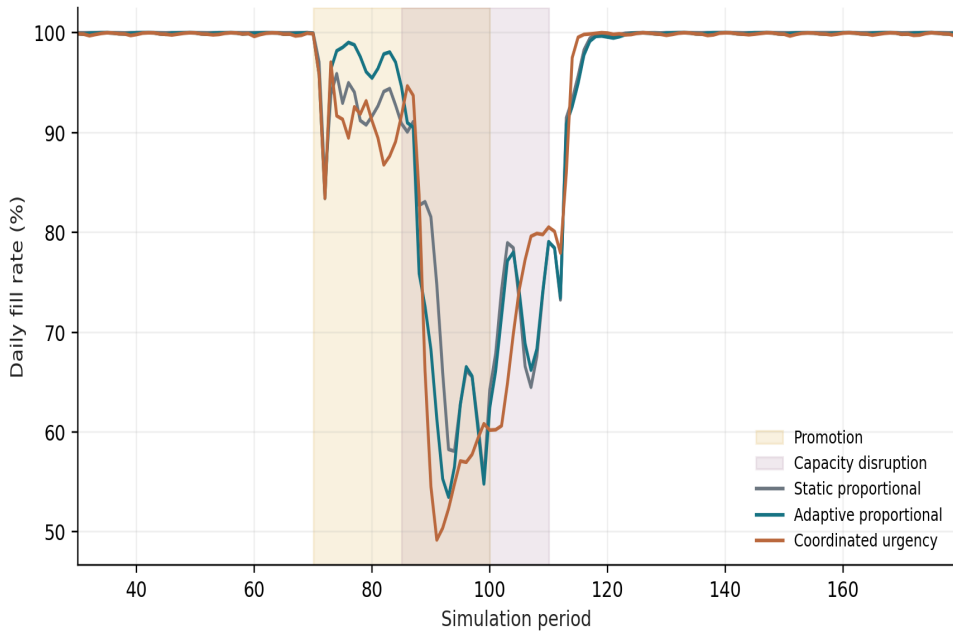


Figure 4 Daily aggregate fill rate in the combined synthetic scenario. Lines average seed-level daily fill rates; shaded windows identify promotion and dispatch disruption. These daily averages are not identical to the demand-weighted whole-window fill rates in Table 6.

Interpretation must respect the three-period transit delay. A dispatch decision affects a later service period, so a response immediately after a demand surge cannot fully restore availability that has already been lost. Likewise, replenishment based on a recently elevated estimate can arrive after the promotion ends. This delay is one reason the numerical study includes a post-event period rather than terminating exactly when the disruption ends.

The graph should not be read as a measured recovery-time experiment. The paper has not defined or estimated a formal recovery threshold with sustained service requirements. Such a metric could be added by specifying, in advance, a service floor and a minimum number of consecutive periods above that floor. Defining the threshold after seeing the trajectories could favor one policy and weaken the comparison.

9.4 Inventory and Service Distribution

Figure 5 plots mean inventory against aggregate fill rate in the combined scenario. A desirable direction is higher service with less inventory, but an economic decision also depends on dispatch and shortage costs. Adaptive proportional allocation holds substantially less inventory than the urgency policy while providing higher aggregate service. This reinforces the conclusion that the tested urgency rule does not make effective use of the available stock.

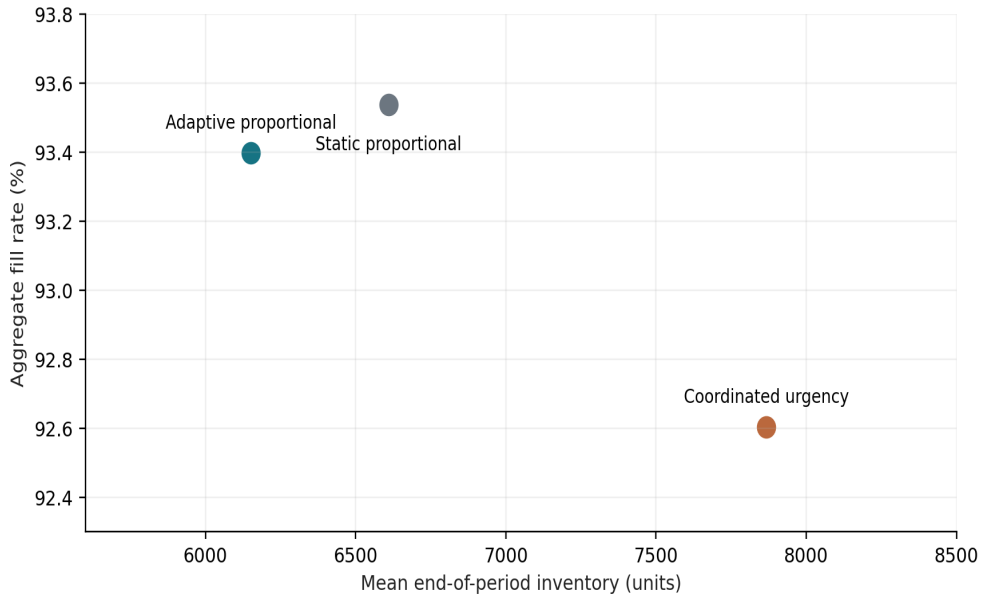


Figure 5 Inventory and service tradeoff in the combined synthetic scenario. Each marker is a policy mean across 30 replications. Axes are intentionally restricted to the observed range for comparison; this is not a claim of a complete Pareto frontier.

The mean of the worst-stream fill rates is 84.950% for static proportional allocation, 90.962% for adaptive proportional allocation, and 86.704% for coordinated urgency. These stream-level minima identify a distributional concern that is hidden by aggregate service above 92% for every policy. The adaptive proportional rule provides the strongest value on this measure among the tested policies in the combined scenario.

A minimum is sensitive to the number of streams and to chance variation. If the network expanded from 320 to many thousands of streams, the worst observed value could decrease even if typical service remained unchanged. Future reporting should therefore include lower quantiles, service by demand segment, and the count of streams below an agreed threshold. The present minimum is a diagnostic, not a complete fairness assessment.

9.5 Capacity Sensitivity

Figure 6 evaluates the adaptive policies across five normal-capacity ratios in the combined scenario. Each ratio retains the same 0.65 disruption multiplier. Increasing the ratio therefore raises both normal and disrupted capacity. This experiment varies resource availability rather than changing the severity multiplier independently. Table 8 provides the corresponding mean costs and fill rates.

Table 8 Combined-scenario capacity sensitivity means across 30 seeds

Capacity ratio	Policy	Cost / unit	Fill rate %
0.90	Adaptive	1.1059	81.453
0.90	Urgency	1.2005	81.536
1.00	Adaptive	0.7207	90.808
1.00	Urgency	0.8159	90.182
1.10	Adaptive	0.6404	93.397
1.10	Urgency	0.7129	92.603
1.20	Adaptive	0.5560	95.471
1.20	Urgency	0.6138	94.821
1.30	Adaptive	0.4996	96.828
1.30	Urgency	0.5538	96.152

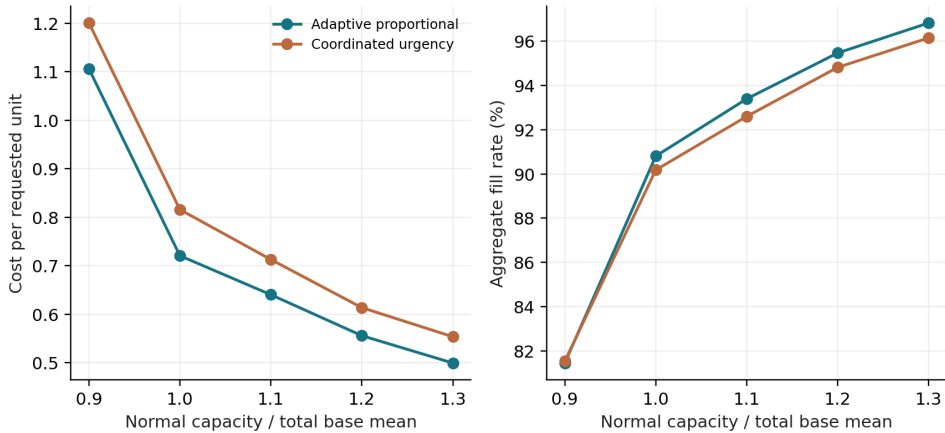


Figure 6 Sensitivity to normal dispatch capacity in the combined synthetic scenario. Lines join mean outcomes at five tested capacity ratios; intermediate ratios were not simulated. Each point represents 30 paired replications.

The sensitivity analysis helps distinguish an isolated adverse setting from a broader pattern across the tested capacity range. It does not demonstrate robustness to a different lead-time distribution, a different shortage cost, or a different number of streams. These dimensions remain fixed. Capacity is also modeled as a scalar quantity, whereas a real facility may face simultaneous constraints on labor, dock doors, refrigerated space, vehicle availability, and SKU-specific stock.

9.6 Feasibility Checks

Every executed primary and sensitivity run checks that inventory and allocations remain nonnegative. Dispatch totals are compared against capacity with a numerical tolerance of one millionth of a unit. No capacity exceedances were recorded. This result is expected from the algebra of proportional scaling and the remaining-capacity update in the greedy allocator, and it also checks their implementation in the simulator.

The checks do not validate the architecture's proposed distributed reservation gateway. The simulation executes serially and has no concurrent workers, delayed acknowledgements, duplicated messages, or

adversarial inputs. It therefore cannot establish exactly-once business execution or resistance to concurrent overbooking. Those properties require separate software and integration tests against an implemented gateway.

10 Discussion

10.1 What the Negative Result Contributes

The experiment demonstrates that an architecture can become more coordinated in its information processing while becoming worse in its resource allocation. The urgency controller receives a network-wide ordering of coverage and uses that ordering consistently. Its failure is not the absence of information or a violation of capacity. The failure is that the chosen mapping from information to allocation does not balance the value of replenishment across streams effectively.

This distinction matters for intelligent supply-chain systems. Adding agents, communication, or a shared planning layer creates opportunities for better control, but those opportunities must be converted into a good decision rule. A coordinator that makes poor assignments can amplify resource imbalance more efficiently than independent planners. The architecture should therefore make policy replacement easy to test and difficult to deploy without evidence.

The result also clarifies the role of explanation. An urgency-based decision is easy to justify in words: the selected location had the lowest coverage. Yet a locally convincing explanation does not establish network efficiency. Explanations should expose the rule and its assumptions, while quantitative evaluation determines whether the rule achieves the intended outcomes. A fluent explanation cannot substitute for a controlled comparison.

10.2 Forecast Quality Versus Decision Quality

The adaptive policies use the same demand estimate, so their different results arise from allocation behavior and its effect on later inventory. This is useful because improvements in forecasting are often evaluated without considering the decisions that consume the forecast. Even a responsive forecast can produce poor outcomes when replenishment requests are converted into large, uneven dispatch blocks. The static and adaptive proportional policies provide a second distinction. Adaptation can reduce overstock and improve service in affected streams while producing a slightly different aggregate fill rate. This means forecast responsiveness is not a universal benefit independent of the target formula, safety buffer, and capacity environment. An accurate estimate of recent demand may still be a poor predictor of demand when a promotion abruptly ends.

A fuller system should evaluate forecast modules using decision outcomes as well as statistical prediction metrics. That does not mean training only on downstream cost. Forecast accuracy, calibration, and data validity remain useful diagnostic measures. The objective is to keep the interface transparent enough to identify whether a problem arises in estimation, target construction, allocation, or execution.

10.3 Implications for Large Retail Systems

The proposed decomposition can support large networks only if communication and state ownership reflect actual resource coupling. A store does not need every other store's transaction history to generate a replenishment request. A regional allocator does need the requests competing for its dispatch capacity. A cross-region transfer process needs a joint view of the source reservation and destination acceptance. These different needs argue for scoped communication rather than universal broadcasting.

For n streams, proportional allocation is linear in n after requests are computed. The implemented urgency rule includes sorting and is approximately $O(n \log n)$, followed by a linear allocation pass. These algorithmic statements describe the local numerical operations. They do not predict enterprise throughput because serialization, database contention, network delays, and integration calls may dominate runtime in a deployed system.

The 320-stream experiment is too small and too simplified to substantiate enterprise-scale claims. A credible scale evaluation would vary streams, products, facilities, regional boundaries, and event rates while recording decision latency and resource contention. It would also test whether partitioning the allocation problem materially changes service outcomes. A system that is fast only because it ignores important cross-region coupling is not a successful large-scale solution.

10.4 Economic and Operational Interpretation

Artificial costs provide a consistent comparison, but commercial interpretation requires additional work. A lost unit may have different consequences across products: immediate margin loss, customer dissatisfaction, substitution, contractual penalties, or future demand reduction. Holding cost may depend on shelf life, financing, storage constraints, and markdown risk. A single pair of coefficients cannot capture those differences.

The dispatch cost in the simulator is linear per unit. Real transport costs can include fixed vehicle costs, minimum loads, route interactions, and time windows. A policy that creates many small allocations may therefore incur costs omitted here. Proportional rationing can appear attractive in a fluid model and become less attractive when case packs or vehicle economics are enforced. That possibility strengthens the need for richer validation rather than invalidating the present bounded comparison.

Operational adoption should begin with a category and region where stock balances can be reconciled reliably and where the action envelope is narrow. The first business outcome to establish is that executed decisions match authorized decisions and that measured service remains acceptable. Broader automation can follow only after the organization understands exceptions, costs, and recovery behavior under normal and disrupted conditions.

11 Data and Integration Design

11.1 Canonical State and Identifiers

A production implementation needs stable identifiers for products, locations, inventory lots where relevant, orders, shipments, and capacity windows. A human-readable product name is not a sufficient join key because names change and may be reused. The state service should retain source identifiers and mapping versions so that a forecast or allocation can be traced back to the operational records on which it depended.

Inventory should be separated into sellable, reserved, damaged, quarantined, and in-transit quantities when the business requires those distinctions. A single stock field can hide material constraints. For example, a store may physically possess units that cannot be sold because they are reserved for online collection or isolated for quality review. The inventory agent should use the quantity available for its actual decision, not whichever stock field is easiest to retrieve.

Events need both an event time and an ingestion time. Event time records when the business occurrence happened; ingestion time records when the decision platform received it. Their difference helps diagnose stale information. A delayed receipt should not automatically be interpreted as a new physical movement

if it refers to a shipment already reconciled by another source. Deduplication must depend on business identity as well as transport-level message identity.

11.2 Proposal and Execution Records

Table 9 specifies a proposed minimum decision record. The purpose is to connect an action to its evidence and authority. The record should be sufficient to reconstruct why a quantity was proposed, how it changed during allocation, which constraints were checked, and what operational system accepted it. The exact schema can vary, but these relationships should not be lost when components are split across services.

Table 9 Proposed minimum decision record

Field group	Example fields	Purpose
Identity	decision_id; proposal_id; idempotency_key	Deduplicate and trace the action
Context	location_id; product_id; period; state_version	Define decision scope and state
Evidence	forecast_version; observation_time; source_event_ids	Reconstruct the information used
Quantities	requested_qty; allocated_qty; committed_qty	Separate proposal from execution
Constraints	capacity_window; reservation_id; policy_version	Identify feasibility and authority
Lifecycle	status; expiry_time; acknowledgement_id	Support recovery and reconciliation
Accountability	agent_version; approval_role; reason_code	Explain responsibility and exceptions

A proposal should expire when its evidence becomes too old or its dispatch window closes. Expiry should release uncommitted reservations through a controlled transition. An execution acknowledgement received after expiry requires reconciliation because the external system may already have accepted the order. The platform must not create a second order merely because its local timeout elapsed.

Reason codes should remain structured even when planners receive a natural-language explanation. Examples include capacity insufficient, data stale, spending limit exceeded, product ineligible, or policy suspended. Structured reasons enable monitoring and prevent the organization from relying on inconsistent free-form explanations when comparing rejection patterns across regions.

11.3 Feature Quality and Missing Observations

Input validation should check impossible quantities, missing identifiers, inconsistent units, and timestamps outside the expected range. A sudden inventory increase might be a receipt, a correction, or a duplicate. The validation layer should avoid making an unsupported choice when the source does not identify the event. It can quarantine the record or reduce the autonomy of decisions that depend on it.

Missingness should be represented as missingness, not silently replaced with zero. A missing demand observation does not mean no customers requested the product. A missing capacity report does not mean unlimited capacity. Imputation policies should depend on the field and the consequence of error. For some analytical features, using a recent value with an age flag may be reasonable; for authority-bearing quantities, a stale value may require suspending new commitments.

Monitoring should distinguish distribution shift from pipeline failure. If demand drops sharply across every product in one store at exactly the same time, a disconnected sales feed may be more plausible than a simultaneous change in customer preference. An intelligent architecture needs these operational checks before applying increasingly complex forecasting models to unreliable inputs.

12 Reliability Security and Governance

12.1 Concurrency and Commitment Safety

Suppose two coordinators read the same remaining dispatch capacity and both produce feasible plans relative to that snapshot. The combined plans can exceed capacity even though each passes an isolated check. The capacity owner must therefore reserve or commit through an atomic operation using the current version of the resource. A failed version check should trigger a new allocation attempt rather than an unconditional retry of the old plan.

The invariant can be expressed as the sum of active commitments and valid reservations remaining no greater than the capacity of the resource window. Reservations need bounded lifetimes, clear ownership, and a transition to committed or released. A timer alone is not enough: if an external system accepted an action but the acknowledgement was delayed, releasing the reservation could make the same capacity available twice. Reconciliation must resolve uncertain outcomes before that resource is reused.

Exactly-once message delivery is not assumed. The proposed execution adapter uses an idempotency key tied to the business action and queries the external system after an ambiguous failure. Duplicate messages can then be recognized without duplicating the shipment or order. This is a design requirement for an eventual implementation; the numerical simulator does not exercise it.

12.2 Failure Recovery

Agent failure should not automatically erase proposals or resource ownership. A restarted agent should recover its last durable state and inspect pending decisions before generating new ones. The system should be able to distinguish a proposal that was never selected from a selected allocation awaiting authorization and from an authorized order awaiting confirmation. Collapsing all three into a generic pending status makes recovery unsafe.

A regional communication partition requires a scoped fallback. Stores may continue recording demand and using previously approved local information, while new shared-capacity commitments remain controlled by the authoritative region. If local emergency authority is permitted, it should consume a preallocated envelope rather than an assumed share of currently unknown capacity. The envelope must be included in the enterprise exposure calculation before the partition occurs.

Recovery should also address inventory corrections. When a cycle count reveals less stock than the system expected, the correction changes future decisions but does not rewrite historical service. An exception process should determine whether the discrepancy arose from shrinkage, timing, damaged goods, or a data defect. That diagnosis influences whether to change replenishment policy, improve data capture, or suspend affected automation.

Table 10 Failure modes and proposed controls

Failure mode	Operational consequence	Proposed control	Validation method
Duplicate execution request	Duplicate order or shipment	Business idempotency key	Replay the same request
Stale capacity snapshot	Resource overbooking	Versioned atomic reservation	Concurrent competing plans
Delayed acknowledgement	Uncertain commitment	Query before retry	Timeout after external acceptance
Missing demand feed	Under-ordering	Freshness checks and fallback	Feed outage injection

Invalid product mapping	Wrong stock movement	Identifier validation	Deliberate mapping mismatch
Agent restart	Lost or repeated work	Durable lifecycle recovery	Restart during each transition
Harmful policy drift	Persistent service loss	Monitored rollback	Replay controlled demand shifts

12.3 Access Control and Untrusted Content

Agents should have permissions aligned with their responsibilities. Forecasting requires read access to selected data and write access to analytical outputs. Order execution requires more restricted write permissions and an explicit policy boundary. A single shared credential for every agent defeats the separation of responsibilities described in the architecture.

External supplier messages and free-form documents should be treated as data, especially when a language model is used to interpret them. A message that describes a delivery delay must not be able to change the agent's permissions, disable approval rules, or instruct the system to send data to a new endpoint. Structured extraction should produce a proposed fact with provenance; trusted validation and authorization components should decide how that fact affects execution.

Audit retention should be sufficient to reconstruct decisions while avoiding unnecessary copies of customer or employee information. Most replenishment decisions can be evaluated using aggregated product and location quantities. Where customer-level records are required for fulfillment, access should remain scoped to that purpose. The present synthetic study contains no personal records and does not test a privacy-preserving learning mechanism.

12.4 Policy Oversight and Accountability

A governance board or designated operations owner should approve the scope within which an autonomous policy may act. The approval should identify the covered regions and products, maximum exposure, performance floors, fallback behavior, and review date. These parameters make the meaning of approved autonomy concrete and avoid treating a model's general availability as permission to act everywhere.

Monitoring should trigger review when performance changes materially, when input quality declines, or when the rate of overridden decisions rises. A high override rate can indicate a poor policy, misunderstood constraints, or an interface that fails to explain the decision. It should prompt investigation rather than automatic retraining. Retraining on overrides without understanding their causes can encode inconsistent human behavior into the next policy.

The assurance component should be organizationally capable of stopping execution within its defined remit. If its only output is an alert that no one is responsible for reviewing, it does not provide effective oversight. Suspension should preserve ongoing commitments and shift new decisions to an approved fallback; it should not blindly cancel shipments already in motion.

13 Deployment and Evaluation Roadmap

13.1 Stage One Offline Replay

The first production-oriented stage should reconstruct historical inventory, requests, dispatches, and receipts from operational records. This exercise tests whether the organization can create a consistent

state history. If replayed stock diverges from recorded stock, the discrepancy should be resolved before policy performance is evaluated. Otherwise, a policy may appear beneficial because it is operating on an impossible historical state.

Offline comparison should use chronological splits and preserve the information available at each decision time. A forecast cannot use a corrected record that arrived after the original dispatch decision unless the purpose is explicitly to measure the value of improved data. Demand reconstruction during stockouts must be documented. Where requested demand is unobserved, alternative plausible reconstructions should be used to measure sensitivity.

The incumbent replenishment process should be reproduced sufficiently well to provide a meaningful baseline. Comparing a proposed optimizer with an intentionally weak static rule may demonstrate a technical improvement without establishing business value. At minimum, the evaluation should include the current operational rule, a well-tuned interpretable policy, and a constrained optimization reference on tractable subsets.

13.2 Stage Two Shadow Operation

In shadow operation, agents generate proposals from live data but do not execute them. Planners can inspect differences from actual decisions and identify missing constraints. The stage is especially valuable for discovering practical issues such as case packs, store receiving hours, supplier blackout dates, and product restrictions that were not represented in the synthetic model.

A shadow policy cannot be evaluated simply by assuming every recommendation would have produced its predicted outcome. Actual operations follow another policy, so the observed inventory trajectory does not equal the counterfactual trajectory under the candidate. A simulator or careful causal evaluation is needed to estimate the effect of replacing the incumbent. The shadow stage primarily validates integration, information timing, proposal quality, and constraint coverage.

The exit criteria should include complete decision traceability, acceptable data freshness, stable execution-ready schemas, and a documented explanation for material differences from planner decisions. Meeting these criteria does not yet establish cost savings. It establishes that the system is sufficiently understood to consider a bounded operational trial.

13.3 Stage Three Bounded Autonomy

A pilot should begin with a limited region, product set, and action size. Its design must consider interference: if treated and control stores share the same constrained distribution capacity, one group's allocations can affect the other. Randomizing individual stores without accounting for this coupling can bias the measured effect. Cluster-level allocation or carefully designed time-based comparisons may be more appropriate, depending on the network.

The pilot should predefine outcome measures and stopping rules. Cost, fill rate, low-service locations, exception rate, and manual workload should be measured together. A policy that saves planner time but creates persistent stockouts should not be described as successful without an explicit assessment of that tradeoff. The organization should also record why operators override recommendations so that adoption problems can be separated from model problems.

Rollback should be exercised before the pilot starts. The test should show that new decisions can switch to the incumbent policy while already committed orders remain visible and reconciled. A nominal rollback button that leaves duplicate pending work or loses reservation state is insufficient. Operational reversibility is part of the deployment design, not an administrative step added after a failure.

13.4 Stage Four Scale Validation

Scale validation should vary the number of stores, products, resource pools, and concurrent decision windows. Measurements should include median and tail decision latency, backlog growth, rejected reservations, duplicate suppression, and reconciliation time. Hardware and service configurations should be reported so that observed performance is interpretable and reproducible.

The evaluation should include failure while under load. A system can meet normal latency targets yet recover poorly when a coordinator restarts during the busiest dispatch period. Tests should introduce delayed messages, unavailable dependencies, hot inventory partitions, and bursts of corrections. The acceptance criterion is not merely eventual completion but preservation of resource invariants and acceptable operational delay.

Table 11 Proposed deployment gates

Stage	Primary question	Evidence for advancement
Offline replay	Can the state and incumbent decisions be reconstructed?	Reconciled histories and controlled comparisons
Shadow operation	Are proposals timely and operationally meaningful?	Valid constraints and complete traces
Bounded pilot	Does execution improve agreed outcomes?	Prespecified comparison and working rollback
Regional expansion	Does performance persist across settings?	Segment results and manageable exceptions
Enterprise scale	Do reliability and latency hold at target load?	Load failure and recovery evidence

14 Limitations and Threats to Validity

14.1 Model Realism

The simulator represents one shared dispatch pool with unlimited upstream supply. It does not model a distribution center inventory balance, supplier disruption propagation, inbound procurement, multiple echelons, vehicle routing, perishability, returns, substitution, or product-specific capacity. These omissions limit the relationship between the numerical results and real retail economics. They also explain why the full enterprise architecture is broader than the implemented evaluation.

All streams share a three-period deterministic lead time. Real lead times may vary by supplier, destination, weekday, weather, and operating congestion. The urgency rule could behave differently if some destinations can receive stock sooner than others. A complete allocation policy should compare expected service consequences at arrival, not only current coverage. This extension requires a stochastic transit model and reliable lead-time observations.

Fractional allocation removes case-pack and unit-rounding effects. The cost model omits fixed dispatch costs and terminal salvage. The initial inventory buffer and 30-period exclusion are design choices rather than calibrated operational values. Each choice should be varied before interpreting the ranking as a stable business recommendation.

14.2 Information and Statistical Validity

Requested demand is fully observed after service, even when inventory is insufficient. This avoids sales censoring but may overstate what a retailer can know. The adaptive estimators are therefore given a

cleaner signal than a sales-only implementation would receive. A later study should compare full demand observation, censored sales, and reconstructed demand under identical physical conditions.

The 30 seeds provide replication within one family of synthetic environments. They do not represent 30 independent retailers or 30 observed commercial deployments. Student *t* intervals summarize uncertainty across these generated replications and are conditional on the environment structure. They should not be used to claim universal superiority or a guaranteed percentage improvement.

The sensitivity experiment varies capacity only. No inference is made about robustness to other demand distributions, correlated shocks, different smoothing coefficients, unequal shortage costs, or a larger action space. The repeated 1.10 setting is retained for consistency checking and is not counted as independent evidence when discussing precision.

14.3 Implementation and Novelty Boundaries

The numerical program is a serial array-based simulation. It does not instantiate a message broker, distributed agents, a policy registry service, an execution gateway, or a production digital twin. Consequently, no measured claim is made about distributed throughput, fault tolerance, cyber resilience, model-serving latency, or deployment cost. The architecture diagrams specify intended relationships rather than implemented services.

The study does not introduce a new reinforcement learning algorithm. It evaluates two adaptive heuristic policies and one static reference policy. The intelligent architecture is proposed as a framework for observation, bounded decisions, and policy evaluation. Its contribution should not be overstated as the first use of agents or AI in supply-chain management.

The literature review is selective and primarily uses directly relevant publications and official standards. Some referenced works are preprints, explicitly identified as such. A venue-specific submission would benefit from a broader systematic search and a comparison with the closest recent implemented architectures. No author identity, institutional affiliation, funding award, or field dataset has been invented to fill those gaps.

15 Future Research

A direct next experiment is to replace blockwise urgency allocation with incremental marginal allocation while keeping the existing demand paths and target formula unchanged. This would test the proposed mechanism behind the observed inventory concentration. A second experiment should use a rolling-horizon allocation optimizer with explicit transit timing and compare its benefit against its computational cost. These comparisons would make the architectural coordination module more concrete without prematurely introducing a complex learned policy.

A richer simulator should introduce multiple distribution centers, finite upstream stock, stochastic replenishment lead times, and lateral transfers. Adding perishability would permit an explicit tradeoff between availability and waste. Adding case packs and route-level fixed costs would test whether the proportional rule's fluid allocations remain attractive when execution becomes discrete. Each extension should be introduced with a baseline and conservation checks rather than combined into an opaque model all at once.

Learning-based extensions should begin with bounded action spaces and reproducible training protocols. Candidate approaches include learned target multipliers, graph-based state representations, or cooperative value functions. The evaluation should report training seeds, environment steps, held-out

scenarios, compute, model versions, and failures. A policy that requires extensive tuning for each region may be less operationally useful than a simpler rule with slightly worse average cost but stable behavior. Further work should investigate service distribution explicitly. A coordinator could enforce minimum allocations or penalize persistent deprivation of selected streams. Such constraints may increase total cost while improving service equity, so the tradeoff should be measured and approved. The objective is to make priorities visible rather than hide them inside an unexplained score.

Finally, a full prototype should test the commitment lifecycle under concurrency. The most important software evidence would include duplicate suppression, atomic reservation, recovery from ambiguous execution, and complete reconstruction of accepted decisions. These tests would validate the architecture's distinctive execution boundary, which is only specified conceptually in the present paper.

16 Conclusion

This paper proposes an intelligent multi-agent architecture for autonomous supply-chain decision-making in large retail systems. The design assigns forecasting, replenishment, resource coordination, execution authorization, and assurance to distinct responsibilities. A versioned operational state and explicit commitment lifecycle connect those responsibilities while preserving the difference between analytical recommendations and authorized physical actions.

The synthetic evaluation shows why policy quality must be tested separately from architectural plausibility. Across the combined promotion and disruption scenario, the greedy urgency coordinator increases mean normalized cost from 0.6404 to 0.7129 and reduces aggregate fill rate from 93.397% to 92.603% relative to adaptive proportional allocation. All tested allocations remain within dispatch capacity, demonstrating that feasible behavior can still be economically and operationally inferior.

The supported conclusion is therefore conditional and practical: autonomous coordination should be introduced through controlled comparisons, independently enforced constraints, traceable execution, and working fallback policies. The proposed architecture provides a structure for that process, while the experiment supplies a reproducible caution against assuming that greater agent coordination automatically creates better outcomes. Enterprise-scale effectiveness remains to be established through richer models, implemented services, and monitored retail pilots.

17 Research Transparency

This manuscript reports an original synthetic numerical demonstration and a proposed architecture. It does not report observations from an operating retailer. No human participants or personal data were used. Commercial savings, production readiness, and regulatory certification are not claimed. Author names and affiliations have been omitted because they were not supplied. Funding and conflict-of-interest declarations must be completed by the submitting authors rather than inferred.

The full simulation source is reproduced in Appendix B. It generates the primary results and capacity-sensitivity results using seeds 100 through 129. The graph and table summaries in this manuscript are calculated from those outputs. The code requires Python and NumPy; the preparation environment versions are recorded with the reproducibility details. Architectural figures are original schematic illustrations, and quantitative graphs are plotted from the synthetic outputs rather than generated as decorative imagery.

References

- [1] M. S. Fox, M. Barbuceanu, and R. Teigen, “Agent-Oriented Supply-Chain Management,” *International Journal of Flexible Manufacturing Systems*, vol. 12, pp. 165–188, 2000. <https://doi.org/10.1023/A:1008195614074>
- [2] A. Oroojlooyjadid, M. Nazari, L. Snyder, and M. Takáč, “A Deep Q-Network for the Beer Game: A Deep Reinforcement Learning algorithm to Solve Inventory Optimization Problems,” *arXiv:1708.05924*, 2017, subsequently revised. Preprint record. <https://arxiv.org/abs/1708.05924>
- [3] C. D. Hubbs, H. D. Perez, O. Sarwar, N. V. Sahinidis, I. E. Grossmann, and J. M. Wassick, “OR-Gym: A Reinforcement Learning Library for Operations Research Problems,” *arXiv:2008.06319*, 2020. Preprint. <https://arxiv.org/abs/2008.06319>
- [4] X. Yang, Z. Liu, W. Jiang, C. Zhang, L. Zhao, L. Song, and J. Bian, “A Versatile Multi-Agent Reinforcement Learning Benchmark for Inventory Management,” *arXiv:2306.07542*, 2023. Preprint. <https://arxiv.org/abs/2306.07542>
- [5] R. Barati and Q. V. Hu, “gym-invmgmt: An Open Benchmarking Framework for Inventory Management Methods,” *arXiv:2605.11355*, 2026. Preprint. <https://arxiv.org/abs/2605.11355>
- [6] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, “Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments,” *arXiv:1706.02275*, 2017. <https://arxiv.org/abs/1706.02275>
- [7] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, “QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning,” *Proceedings of Machine Learning Research*, vol. 80, pp. 4295–4304, 2018. <https://proceedings.mlr.press/v80/rashid18a.html>
- [8] J. Achiam, D. Held, A. Tamar, and P. Abbeel, “Constrained Policy Optimization,” *Proceedings of Machine Learning Research*, vol. 70, pp. 22–31, 2017. <https://proceedings.mlr.press/v70/achiam17a.html>
- [9] GS1, “EPC Information Services Standard,” Release 2.0, 2022. <https://ref.gs1.org/standards/epcis/2.0.o/>
- [10] National Institute of Standards and Technology, “Artificial Intelligence Risk Management Framework AI RMF 1.0,” NIST AI 100-1, 2023. <https://doi.org/10.6028/NIST.AI.100-1>