Supplementary Materials

The following content was not necessarily subject to peer review.

6 Experimental details

6.1 Environment

Here are the paramters for the 3-node example, the exact parameters for the 10-node example must be kept confidential.

node	c_n^h	c_n^b	w_n	$\tilde{w}_n$	demand
middle	26.7	34	170093	12465	$\mathrm{NB}(0.11, 0.0003)$
right	29.5	42	127646	8025	$\mathrm{NB}(0.081, 0.00047)$
replenishment	q_v	$k_v^{\max}$	$k_v^{\max}$		lead time
left	4180	2	0		$\mathrm{ARPois}(6, 20, 0.98)$
right	4100	2	1500		$\mathrm{ARPois}(15, 20, 0.98)$

Both demand distributions are Negative Binomial, $\mathrm{NB}(r, p)$. To reflect realistic fluctuating lead times, we use an autoregressive variant of the Poisson distribution, $\mathrm{ARPois}(\lambda_0, h, \phi)$. This distribution produces each time-step t a lead time τ_t which is drawn from the distribution $\mathrm{Pois}(\lambda_t)$, where λ_t depends on the previously h drawn lead time values τ according to the formula:

$$\lambda_t = \max\left(0, \phi \cdot \frac{\sum_{t' \in [t-h, t]} \tau_t'}{h} + (1-\phi)\lambda_0\right) \tag{5}$$

This produces a time-correlated Poisson distribution which retains an expected value of ${}_0$.

6.2 Details on PPO training

We train feedforward neural nets with PPO. We add skip connections (He et al., 2016) every two layers to enable training deep networks, effectively using 2 residual blocks in both the value and policy networks.

Hyper-parameter	parameter value
no. of environment interactions	10^7
policy network	width 256, depth 4
actor network	width 256, depth 4
activation function	ReLU
discount factor γ	0.99
GAE paramter	0.95
Adam learning rate	$2.5 \cdot 10^{-4}$
batch size	64

6.3 Metastability

The learning curves in Figure 4 highlight the difficulty of deep RL training for multi-echelon supply chain optimization if the reward function is not chosen carefully. Metastability effects occur as there

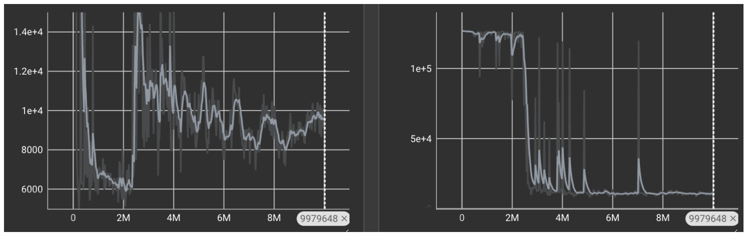

Figure 7: 3-node example, inventory during training at middle node (left) and right node (right)

are suboptimal strategies that manifest local maxima for the parameter vectors of the policy network. As a consequence, gradient based algorithms struggle to improve the suboptimal strategy. To explain what happens we plotted the inventory levels of the 3-node example during training (inventory plots look similar for the 10-node example). The learning curve of Figure 4 shows a sudden improvement after 2.5M environment interactions, preceded by a return drop. This is reflected in Figure 7. The agent finds quickly the suboptimal strategy to optimize the inventory at the middle node while completely sacrificing the terminal right node, running at maximal inventory 127646. At 2.5M iterations the RL agent deviates from the local maximum and explores a new strategy, reducing inventory at the terminal node and increasing inventory at the middle node.

7 Material requirements planning (MRP)

The main algorithmic invention of this article is to combine off-the-shelf RL training (PPO) through imitation learning with rule-based heuristics from operations research (OR). There are several OR algorithms to solve approximately different supply chain problems. For the multi-echelon inventory optimization (MEIO) problem studied in the present article we use a dynamic programming inspired rule-based algorithm that is (with various modifications) implemented in many industry supply chain solutions. We now give a quick overview for the interested RL researcher.

The rule-based algorithm implements a time-phased Material Requirements Planning (MRP I) system to maintain inventory levels above safety stock thresholds across all nodes in the supply chain. Rooted in the foundational work Orlicky (1975), the process begins by exploding dependencies from downstream nodes (e.g., retailers or finished goods) to upstream suppliers, following the hierarchical structure of the multi-echelon supply chain. Inventory projections are calculated in daily time buckets over a fixed $H = 150$ planning horizon. Starting from the current day t, the system computes the projected available balance (PAB) for each subsequent day $s \in [t, t + H]$, accounting for scheduled receipts, planned orders, and demand forecasts. If the PAB is projected to fall below the safety stock level at time T, a planned order is generated to replenish the deficit. Orders are offset by lead times using backward scheduling: for an order requiring τ days of lead time, the release date is set to $T - \tau$. If this calculated release date precedes the current day t, the order is flagged as overdue and scheduled for immediate release. This daily recalibration ensures alignment with the core principle of time-phased net requirement calculation, where material plans are dynamically adjusted to reflect real-time demand and supply conditions. Rule-based MRP algorithms are dynamic-programming, heuristic-based algorithms. It implements a safety-stock approach to managing the supply chain, meaning it predicts the future inventory levels of all nodes in the chain and tries to ensure inventory never falls below the "safety stock" that must be given to the algorithm.

Since we use the MRP algorithm in our examples without multi-material manufacturing steps, we give pseudo-code for a simplified version of MRP. It should be noted that the algorithm is a very simple MRP variant that does not estimate demand expectations and lead times on the run. We do this for a fair comparison to the RL agents, otherwise demand distributions should also be included in the MDP state-space and not be given as part of the model.

There are two novel ideas we add on the standard OR literature.

Algorithm 2 MRP Algorithm (without multi-material manufacturing steps)

Input: expected demands $\mathbb{E}(d)$ and lead times $\mathbb{E}(l)$, safety stocks S_n for all nodes, current generalized inventories $G_{n,t}$ and running orders set O.
for each node n in topological order **do**
  for each time-step $s \in [t, t+H]$ **do**
    $gen(n, \tau) \leftarrow$ amount of additional material by finished orders
    $G_{n,s} \leftarrow G_{n,s-1} + gen(n,s) - \mathbb{E}\left(d(n)\right)$
    if $G_{n,s+1} < S_n$ **then**
      Set number of lots L to minimal number containing at least amount $S_n - G_{n,s+1}$.
      if procurement is possible **then**
        Add to O an order from a source node. L lots, start time: $\max(t, s - \mathbb{E}(l))$
      else
        Choose source node n' that maximizes $\frac{G_{n',s+1}}{\mathrm{num\_outgoing}(n')}$, where $\mathrm{num\_outgoing}(n')$ denotes the number of nodes supplied by node n'.
        Add to O an order from node n' to node n. L lots and start time $\max(t, s - \mathbb{E}(l))$
      end if
    end if
  end for
end for
Output: all orders in O that start at time t. =0

- We interpret MRP(S) $=: \pi$ as a policy. The action (orders) in the state S (inventory level and current order book) are the output orders of the algorithm given above (the orders suggested by the algorithm to be placed at initial time t).
- The safety stock vector S is a required input to the algorithm. We define the reward-based optimal safety stock vector S^* by maximizing the expected reward R under the MRP run defined by the MRP algorithm: $V(S) = \mathbb{E}_{\text{MRP}(S)}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$. Since S is a hyperparameter to the algorithm, it is natural to use a Bayesian optimization algorithm to do so.