Simultaneous Decision Making For Stochastic Multi-Echelon Inventory Optimization With Deep Neural Networks As Decision Makers

MOHAMMAD PIRHOOSHYARAN[1] AND LAWRENCE V. SNYDER[1]

[1]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA

LEHIGH
UNIVERSITY

Simultaneous Decision Making for Stochastic Multi-echelon Inventory Optimization with Deep Neural Networks as Decision Makers

Mohammad Pirhooshyaran

Industrial and Systems Engineering

Lehigh University

mop216@lehigh.edu

Lawrence V. Snyder

Industrial and Systems Engineering

Lehigh University

lvs2@lehigh.edu

ABSTRACT

We propose a framework that uses deep neural networks (DNN) to optimize inventory decisions in complex multi-echelon supply chains. We first introduce pairwise modeling of general stochastic multi-echelon inventory optimization (SMEIO). Then, we present a framework which uses DNN agents to directly determine order-up-to levels between any adjacent pair of nodes in the supply chain. Our model considers a finite horizon and accounts for the initial inventory conditions. Our method is suitable for a wide variety of supply chain networks, including general topologies that may contain both assembly and distribution nodes, and systems with nonlinear cost structures. We first numerically demonstrate the effectiveness of the method by showing that its solutions are close to the optimal solutions for single-node and serial supply chain networks, for which exact methods are available. Then, we investigate more general supply chain networks and find that the proposed method performs better in terms of both objective function values and the number of interactions with the environment compared to alternate methods.

Keywords: Stochastic multi-echelon inventory optimization; Deep neural networks; General supply chain networks; Base-stock inventory policies; Simultaneous decision making; Agent-environment interactions

1 Introduction

The central goal in stochastic multi-echelon inventory optimization (SMEIO) is maintaining inventory levels by regulating the order quantities to optimize a cost function. The cost function usually consists of a shortage (penalty) cost plus holding costs, so that there is a tradeoff between ordering too much vs. too little compared to the stochastic demand, each of which incurs its own cost. In this study, we focus on identifying order-up-to levels (OULs) for multiple pairs of nodes in a complex supply chain network (SCN) under a finite decision horizon. Customer demands are stochastic with a known distribution; shipment lead-times are known constants; and decision makers have global information about the system states.

To this end, we use the deep neural network (DNN) paradigm [1] as a means for decision-making. We model general SCNs (we use the terms "mixed" and "general" SCN interchangeably) in which several DNNs can regulate separate inventory levels for different parts of the network jointly, and by interacting with other parts of the SCN environment, yet our model only suggests a order-up-to level for each decision.

Early approaches using machine learning (ML) to optimize inventory in SCNs primarily used finite Markov decision processes (MDP), dating to about two decades ago. Giannoccaro and Pontrandolfo [2] introduce a simulation-based RL to optimize inventory decisions in a three-node serial SCN. Several modified Q-learning [3] algorithms have been presented, with Q-tables containing possible joint state–action spaces for SMEIO [4].

Recently, Oroojlooyjadid et al. [5] explore the well-known beer game inventory problem (a four-node serial SCN) via the deep Q-learning framework [6], which integrates the DNN concept into a Q-learning algorithm. Previous ML studies are mainly focused on cost comparisons between their proposed approach and some known optimal or heuristic policies, and no further inventory policy behaviors have been reported [4, 5]. Moreover, previous studies restrict the SMEIO settings to fit their assumptions. For instance, the action spaces (decisions made by ML approaches) are discretized in advance to make the problem tractable.

In this study, not only do we consider general complex SCNs with multiple decisions to be made at the same time by DNN agents at several echelons of the SCN, but we directly suggest OULs as the DNN's output. In other words, the proposed framework is capable of suggesting interpretable inventory actions. Moreover, there is no restriction on the order quantities (action space) and/or inventory levels (state space) of the proposed method. (For example, they are not discretized.)

As our first contribution, we model the SCNs considering pair-wise (edge) decision makers. Then, we propose a method that finds OULs for complex MEIO systems under a finite horizon. In order to demonstrate that our method is effective, we compare it against classical inventory optimization (IO) models and algorithms. However, these algorithms assume an infinite horizon. Therefore, to make the best comparison possible, we train our model in a finite-horizon setting that approaches steady-state sufficiently well that it is a reasonable approximation for the infinite-horizon setting. We do this by (a) setting the initial conditions carefully, (b) setting the decision horizon, and possibly the warm-up interval, carefully. For instance, we initialize the inventory levels to be lead-time demands. If our method is reasonably close to optimal for these classical IO settings, then we have confidence that it will also be close to optimal for settings that cannot currently be solved by classical IO models. The true value of our method is its ability to solve complex MEIO systems settings.

The rest of the paper is as follows: We briefly review the literature in Section 2, divided into two separate parts, one on SMEIO background and one on ML in the SMEIO framework. Then we explain our model in Section 4. We describe our numerical experiments in Section 5 and we conclude the paper in Section 6. A python package containing an implementation of the framework is available at the paper's repository.

2   ML Background in SMEIO

ML approaches for stochastic inventory optimization have been studied for a long time. Global supply chain management has been studied in [7] via the semi-Markov average reward technique (SMART). Stockheim et al. [8] train

RL agents to explore an optimal job acceptance strategy in a decentralized SCN. Many early works modify different versions of the Q-learning algorithm. Competitive supply chains are studied by [9], where SCN nodes make their decisions separately and independently in an interconnected system.

Oroojlooyjadid et al. [5] study a four-node serial system via deep Q-learning [6]. They study cases in which (non-ML) nodes follow a base-stock policy, as well as cases in which those nodes display irrational behavior and diverge from their expected inventory policy. Zhao and Sun [10] investigate a multi-agent RL framework model to solve SMEIO considering multiple echelons and multiple commodities. Chaharsooghi et al. [4] analyze supply chain ordering management with a focus on the beer game and suggest a reinforcement learning ordering mechanism. Recently, Gijsbrechts et al. [11] extensively investigate the use of deep RL for three otherwise intractable inventory problems—dual sourcing, lost sales, and multi-echelon optimization problems.

Most previous studies consider discrete state and/or action spaces. A countable action set is a necessity for Q-learning convergence [3]; therefore, papers using Q-learning discretize the agent's possible action values. For instance, Giannoccaro and Pontrandolfo [2] explain that in SMEIO, inventory position (IP) has no bound, which implies an infinite-size MDP. Then, they discretize the IP values and associate an integer number to an actual IP interval. New advances that integrate DNN into the RL framework open further opportunities to explore complex SCNs. There exist very few studies in which the policies are approximated by deep neural networks [5], but even these studies do not report the optimal base-stock levels or other inventory policy parameters. In other words, the ML framework aims to minimize the SCN cost function, but the appropriate interpretation of the solution into a near-optimal policy remains unexplored. We aim to provide a framework to present clear base-stock levels for general SCNs. An overview of SMEIO approaches is provided next.

3 Stochastic multi-echelon inventory optimization

Research on SMEIO can be classified according to several dimensions. The number of echelon layers as well as their topological structure are essential factors. One can categorize SMEIO into single-, two-, and multi-echelon networks with serial, divergent, convergent, or general structures [12]. Furthermore, inventory review can be continuous or periodic, meaning that the system's inventory levels evolve smoothly over time or update once in each decision period. Demands can be discrete (mainly Poisson [13], compound Poisson [14], and Erlang [15]), continuous (mainly normal [16]) or general stochastic [17].

Solution approach is another dimension along which to classify SMEIO. Simulation–optimization [18], computational and numerical [5], approximate [13], and exact (theoretical) methods [17] are among the most common solution approaches. In most approaches, we impose a particular form of the inventory policy (e.g., a base-stock policy), and an algorithm is used to obtain a near-optimal set of parameters for that policy to optimize the objective function [19]. This approach is especially prevalent for systems for which the optimal policy structure is unknown, a category which includes distribution systems [20] as well as general networks. However, this approach has some limitations. The methods are dependent on the problem settings, and adjustments in the chosen policy or system structure require the algorithm to be initiated once again. We aim to reduce these limitations by using trainable DNN algorithms in which one can transfer the knowledge (learned weights of agents' networks) from a SCN to a modified one.

Most SMEIO models fall either into the category of *stochastic service models* (SSM) or of *guaranteed service models* (GSM). SSM models allow for the possibility of stockouts at various nodes of the SMEIO system, which then result in stochastic lead times to their downstream nodes. All of the models we have discussed so far are SSM models, as is the approach we propose. In contrast, GSM models impose an upper bound on the demand, which allows all nodes in the system to guarantee delivery within a fixed number of time periods. This guarantee leads to greater tractability (and therefore greater adoption in practice), even for general network topologies, but also requires some strong assumptions, the demand bound chief among them. To the best of our knowledge, no GSM models in the literature can accommodate general holding and stockout cost structures, as our approach can. Finally, we note that the inventory dynamics in SSM and GSM models operate essentially in the same way; the difference lies in how the base-stock levels are optimized, rather than in how the systems are managed. For further discussion of SSM and GSM models, see [12, 21].

4 SMEIO Model and Environment

4.1 SMEIO Model

We consider a multi-echelon supply chain network with a general topology. The network must be connected and may not contain directed cycles; otherwise, any topology is allowed, including assembly nodes (nodes with more than one predecessor) and/or distribution nodes (nodes with more than one successor). We use $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ to denote the SCN graph, in which $\mathcal{N}$ is the set of all nodes, and $\mathcal{E} \subseteq \{(i,j) : i, j \in \mathcal{N}\}$ is the set of all edges. We consider periodic review, with a finite horizon consisting of T periods. Demand at each customer-facing (leaf) node is stochastic and may have any probability distribution, discrete or continuous, so long as the demand is drawn iid from that distribution. Different leaf nodes may have different demand distributions. (Our example networks use normally distributed demands, except when stated otherwise.) Figure 1 illustrates an example of an SCN, which we will return to later in the paper.

Each node in the network accepts raw materials from its supply node(s), processes them into finished goods, and ships the finished goods to its customer node(s). A given node's supply and customer nodes may be other nodes in the network (these are called *internal* supplier or customer nodes) or *external* suppliers or customers. We assume that a node cannot have *both* an internal *and* an external supplier, nor can it have both an internal and an external customer, but both of these assumptions can be relaxed through suitable use of dummy nodes. Let $\mathcal{U}_j$ be the set of immediate upstream nodes (i.e., predecessor nodes) from node j, and $\mathcal{D}_j$ be the set of immediate downstream nodes (successor
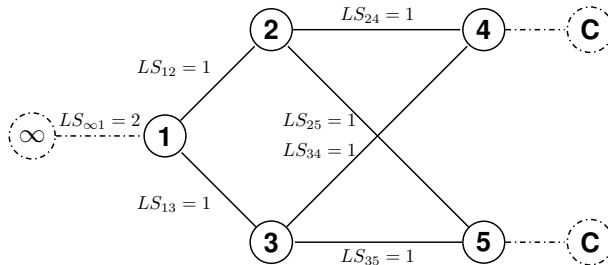

Figure 1: Mixed SCN.

4

nodes) from node j, for $j \in \mathcal{N}$. For modeling convenience, if a node has an external supplier or customer node, these are treated as dummy nodes in the network and are assumed to be contained in $\mathcal{U}_j$ and $\mathcal{D}_j$, respectively. Dummy supplier nodes are assumed to have infinite capacity. (The dummy supplier node for node 1 and the dummy customer nodes for nodes 4 and 5 are indicated in Figure 1 with dashed lines.)

Our framework allows for two different kinds of assembly nodes. An "assembly-*and*" node requires one unit of raw material from *each* of its predecessors—it is an "and" relationship. (It is straightforward to modify our approach to handle cases in which different numbers of units are required from different predecessors.) An "assembly-*or*" node requires one unit of raw material from *any* of its predecessors—it is an "or" relationship.

Each node has a finished-goods inventory that contains items that have been processed but not yet shipped to a customer. It also has one or more raw-material inventories, one for each supplier (including the external supplier, if any). When a node receives raw material items from its supplier(s), those items are placed into the raw material inventories. The node immediately processes as much raw material inventory as possible to produce finished goods. If node j is an assembly-*and* node, then it processes

$$R_{jt} = \min_{i \in \mathcal{U}_j} \{IL^r_{jit}\} \tag{1}$$

items into finished goods in period t, and if it is an assembly-*or* node, then it processes

$$R_{jt} = \sum_{i \in \mathcal{U}_j} \{IL^r_{jit}\} \tag{2}$$

items, where IL^r_{jit} is the number of units of the raw material from node i that are in inventory at node j in period t. R_{jt} units are immediately added to node j's finished goods inventory. For an assembly-*and* node, R_{jt} units are subtracted from each of node j's raw material inventories. For an assembly-*or* node or a non-assembly node (i.e., a node with at most one predecessor), all units are removed from the raw material inventory.

The shipment lead time for orders placed by node j from its predecessor $i \in \mathcal{U}_j$ is denoted L_{ij}. Shipment lead-times are deterministic, but due to possible upstream shortages, downstream nodes might experience stochastic lead-time. The processing time to convert raw materials to finished goods at a given node is assumed to be zero, though this assumption can be relaxed by adding dummy nodes whose shipment lead times equal the processing times.

If a node has insufficient inventory to meet its demands in a given period, the available inventory is allocated to customer nodes in proportion to the size of their orders in that period, and the remaining demands are backordered. Backorders are modeled as negative finished-goods inventory, as is common in the literature. If the inventory level is IL, then the number of items on hand is IL^+ and the number of backorders is IL^-, where $x^+ = \max\{x, 0\}$; and $x^- = \max\{-x, 0\}$. Backorders may occur at any node, whether it has internal or external customers. However, since a node never processes more items than it has raw material inventory for, raw material inventories are always non-negative.

Holding and shortage costs may be arbitrary linear or nonlinear functions of the on-hand inventory (including in-transit inventory) and backorders, respectively, at the end of a period. In particular, the *state variables* at a given node $j \in \mathcal{N}$, evaluated at the end of period t, are as follows:

- IL_{jt} = the inventory level of finished goods

- BO_{jkt} = the backorders at node j that are owed to customer node k ($k \in \mathcal{D}_j$); note that $\sum_{k \in \mathcal{D}_k} BO_{jkt} = IL_{jt}^-$

- IL_{jit}^r = the inventory level of raw material i at node j ($i \in \mathcal{U}_j$)

- IT_{jkt} = the inventory in transit (being shipped) from node j to node $k \in \mathcal{D}_j$

The *cost functions* at node j are as follows:

- $h_{ij}(\cdot)$ = the holding cost function for items from node $i \in \mathcal{U}_j$ that are held in raw material inventory at j or as a component of node j's finished goods inventory or of in-transit inventory from node j to node $k \in \mathcal{D}_j$

- $p_{jk}(\cdot)$ = the stockout cost function for backorders at node j that are owed to node $k \in \mathcal{D}_j$

Then the total cost incurred in period t is given by

$$c_t = \sum_{j \in \mathcal{N}} \sum_{i \in \mathcal{U}_j} h_{ij} \left(IL_{jit}^r + IL_{jt}^+ + \sum_{k \in \mathcal{D}_j} IT_{jkt} \right) + \sum_{j \in \mathcal{N}} \sum_{k \in \mathcal{D}_j} p_{jk} \left(BO_{jkt} \right) \tag{3}$$

where $h_{ij}(\cdot)$ and $p_{jk}(\cdot)$ are general (possibly non-linear or non-convex) functions. In this calculation, an item from node i that is shipped to j is counted in the holding cost function $h_{ij}(\cdot)$ when it is in raw-material inventory at node j, in finished-product inventory at node j, and in transit from node j to its customer(s). This approach for calculating holding costs is somewhat non-standard, but is meant to provide more flexibility, and many common settings are special cases. For example, in a distribution system (each node has at most one predecessor) in which node j has a holding cost charged on finished-goods inventory and inventory in transit to its successors, we can simply set $h_{ij}(x) = h_j x$ for all $(i, j) \in \mathcal{E}$. Moreover, we note that it is straightforward to modify our approach for cost functions with other functional forms. It is also worth mentioning that if one wants to consider an added holding cost value for finished items, one could model this by adding a dummy raw material.

At the end of the planning horizon (after period T ends), any remaining inventory or backorders at node j are reimbursed or charged according to a *salvage function* $v_j(x)$. That is, at the end of the horizon, the system incurs a cost of

$$\sum_{j \in \mathcal{N}} v_j(IL_{jT}), \tag{4}$$

where $v_j(x)$ may be positive (indicating a cost), negative (indicating a revenue), or zero, for either positive or negative values of x. Salvage values and costs are a common mechanism in inventory models to avoid end-of-horizon effects such as excess inventory buildup or selloffs near the end of the horizon.

To facilitate traversing through the SCN graph, we number the nodes with integer values. We assume that the nodes are numbered $1, \ldots, N$ in ascending order by their total shipment lead-times from the infinite source. That is, if node i has a longer total lead-time than j as calculated from the infinite source, then $i > j$. Ties are broken arbitrarily, and duplicate node labels are not allowed.

6

4.2 State Variables and Sequence of Events

Each node j follows a base-stock policy to place orders from its predecessors, and it may use a different order-up-to level (or base-stock level) for each predecessor. In particular, we use OUL_{ji} to represent the order-up-to level used by node $j \in \mathcal{N}$ when it places orders from predecessor $i \in \mathcal{U}_j$. The OUL_{ji} values may be chosen by our DNN agent or by some other mechanism.

The sequence of events at each node j in each time period t is as follows:

1. The demand D_{jkt} is observed from each $k \in \mathcal{D}_j$. If k is an internal customer, then D_{jkt} is the order quantity placed by node k, and if k is an external customer, then D_{jkt} is an exogenous random variable.

2. For each predecessor $i \in \mathcal{U}_j$, node j orders $OUL_{ji} - IP_{jit}$ units from predecessor node i, where

$$IP_{jit} = IL_{j,i,t-1} - \sum_{k \in \mathcal{D}_j} D_{jkt} + IT_{i,j,t-1} + BO_{i,j,t-1} \tag{5}$$

   is the inventory position of item-i materials at node j immediately before the order is placed. Note that it includes only the raw-material inventory of item i at node j, and not the finished goods inventory at node j.

3. For each predecessor $i \in \mathcal{U}_j$, node j receives all items that were shipped from node i L_{ij} time periods ago. (There are $S_{i,j,t-L_{ij}}$ such units.) Raw material and in-transit inventories are updated as

$$IL_{jit}^r = IL_{j,i,t-1}^r + S_{i,j,t-L_{ij}}$$
$$IT_{ijt} = IT_{i,j,t-1} - S_{i,j,t-L_{ij}}.$$

4. Node j processes R_{jt} units, where R_{jt} is given by (1) or (2) depending on whether node j is an assembly-*and* node or an assembly-*or* node. (If j is not an assembly node, then the two equations are equivalent.) The raw-material inventory levels are further updated as

$$IL_{jit}^r = IL_{jit}^r - R_{jt} \tag{6}$$

   if node j is an assembly-*and* node, and as

$$IL_{jit}^r = 0 \tag{7}$$

   otherwise. The finished goods inventory is updated as:

$$IL_{jt} = IL_{j,t-1} + R_{jt}.$$

5. For each successor $k \in \mathcal{D}_j$, node j ships S_{jkt} units to node k. If $IL_{jt} \geq \sum_{k \in \mathcal{D}_j}(D_{jkt} + BO_{j,k,t-1})$, then node j has sufficient inventory to meet all of its backorders and new demands; the shipment quantity and new backorder level are given by

$$S_{jkt} = D_{jkt} + BO_{j,k,t-1} \tag{8}$$
$$BO_{jkt} = 0 \tag{9}$$

7

If, instead, $IL_{jt} < \sum_{k \in \mathcal{D}_j}(D_{jkt} + BO_{j,k,t-1})$, then available inventory is allocated proportionally according to the current demands:

$$S_{jkt} = (D_{jkt} + BO_{j,k,t-1})\frac{D_{jkt}}{\sum_{l \in \mathcal{D}_j} D_{jlt}} \tag{10}$$

$$BO_{jkt} = BO_{j,k,t-1} - S_{jkt}. \tag{11}$$

In either case, the finished-goods and in-transit inventory levels are updated as

$$IL_{jt} = IL_{jt} - \sum_{k \in \mathcal{D}_j} D_{jkt} \tag{12}$$

$$IT_{jkt} = IT_{jkt} + S_{jkt}. \tag{13}$$

(Note that only new demands, not old backorders, are subtracted from IL_{jt} since old backorders are already counted as negative inventory in IL_{jt}. Note also that demands are subtracted whether or not they are actually shipped out, since the inventory level decreases in either case, either by a reduction in on-hand inventory or an increase in backorders.)

6. Holding and stockout costs are assessed according to (3). The ending raw-material inventory levels are given by (6) or (7); the ending finished-goods inventory level is given by (12); the ending backorders are given by (9) or (11); and the ending in-transit inventories are given by (13).

The sequence described above is the sequence of events for each node. However, the events are split into two phases: In the first phase, the nodes follow events 1–3 in order from downstream to upstream, and in the second phase, the nodes follow events 4–6 in order from upstream to downstream. That is, the downstream-most nodes place their orders to their predecessors, their predecessors place their orders, etc.; then the upstream-most nodes ship units to their successors, who ship units to their successors, etc.

Note also that the multi-period newsvendor problem (in which there is a single node, which can hold inventory and backorders from one period to the next) can be modeled using the framework above by setting the lead time to 1. (The newsvendor problem is often described as having zero lead time, but it also uses a different sequence of events, in which we observe the demand after we place the order. Setting the lead time to 1 converts our sequence of events to this one.)

The notation is summarized in Table 1.

4.3 Interaction between Agents and Environment

Each node j must choose the order-up-to level (OUL) it uses when placing order from each of its predecessor nodes $i \in \mathcal{U}_j$; we refer to the OULs as being chosen "for the edge (i, j)." Decisions for each edge (i, j) may be made by a separate DNN agent, or using some exogenous mechanism. For ease of exposition, we will assume that *all* edges are governed by a DNN agent, but it is straightforward to adapt our method if some edges have non-DNN decision makers. This structure implies that there can be up to $|\mathcal{E}|$ independent DNN decision makers.

Table 1: Notation summary.

Inputs		
	T	Length of the planning horizon
	$\mathcal{N}, \mathcal{E}$	Sets of nodes and edges in SCN
	$\mathcal{U}_j, \mathcal{D}_j$	Sets of nodes upstream from and downstream from (respectively) node j
	$h_{ij}(\cdot)$	Holding cost function for items from node i in raw material, finished goods, or in-transit inventory at node j
	$p_{ij}(\cdot)$	Stockout cost function for stockouts at node i that are owed to node j
	L_{jk}	Lead time for items shipped from node j to node k
State Variables		
	D_{jkt}	Demand placed by node k to node j in period t
	S_{jkt}	Number of units shipped from node j to node k in period t
	IL_{jt}	Finished-goods inventory level at node j at the end of period t
	IL^r_{jit}	Raw-material inventory level at node j for items from node i at the end of period t
	BO_{jkt}	Backorders at node j owed to node k at the end of period t
	IT_{jkt}	Inventory in transit from node j to node k at the end of period t
	IP_{jit}	Inventory position of item-i materials at node j in period t immediately before order is placed
Decision Variables		
	OUL_{ji}	Order-up-to level (base-stock level) used by node j when placing orders from node i
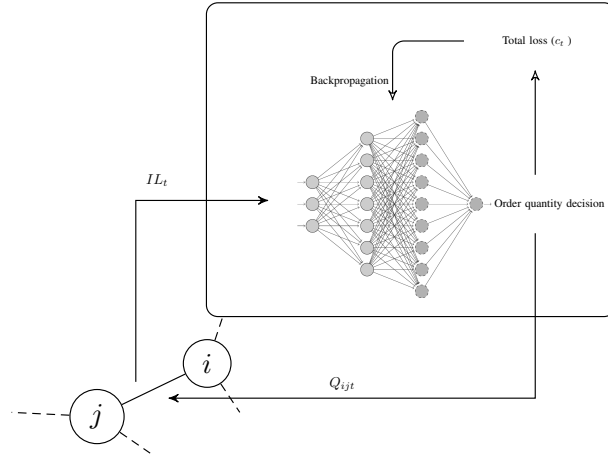
Figure 2: DNN-SMEIO framework.

The agents are trained by interacting with an *environment* consisting of a simulation of the SCN described above. In particular, the agent makes decisions for multiple *episodes*, each of which consists of T time periods. At the beginning of each episode k, the DNN chooses order-up-to levels OUL_{ji} for all $(i, j) \in \mathcal{E}$ and sends these levels to the environment. The environment simulates the SCN to calculate the cost of the current OULs, given by

$$\mathcal{C}_k = \sum_{t=1}^{T} c_t, \tag{14}$$

where c_t is as given by equation (3). The DNN weights are then updated, new OULs are chosen, and a new episode begins. When the DNNs are trained, the output OUL_{ji} converges to a single OUL. In practice, the weight-update procedure happens considering mini-batches of episodes together for computational purposes. The inputs to the DNN are largely irrelevant and can be set in any number of ways. This is because the DNN is optimizing an objective function, rather than trying to determine labels for a given input. In our numerical experiments, we use the inventory positions as the input, but the DNN could just as easily be given a vector of 1s as its input.

Figure 2 shows the proposed framework, in which a DNN is responsible for deciding the quantities of items ordered by node j from node i. We use fully connected DNNs, with several possible hidden layers for each decision maker. Moreover, a batch normalization procedure is considered after every network layer, which considerably stabilizes the learning procedure.

We allow the agents to be optimized by individual DNNs or by a single DNN. That is, networks corresponding to different agents can be completely separate, or they can share their first few layers.

The loss function used to train the DNN(s) is given by the per-episode costs $\mathcal{C}_k$ given in (14). (A similar approach was used by [22]. This is in contrast to the more common approach in which the DNN loss function measures the distance between an estimated and actual value.) To optimize the weights of the DNN network(s), we use the adaptive moment estimation (Adam) optimizer [23] with fixed learning rate. Each environment consists of many hyperparameters that might need to be tuned.

5   Numerical Experiments

We report the results of our experiments on single-node, serial, assembly and mixed SCNs. At first, we investigate the effectiveness of the model against infinite-horizon classical IO models to check whether our proposed method can produce results that are close to those produced by established methods for classical IO problems. Then, we tackle more complex structures.

5.1   Framework Structure, Optimization Algorithm and Hyperparameter Tuning

In this study, we utilize Spearmint Bayesian optimization, and in particular, the Gaussian process expected improvement (GPEI) method [24]. Figure 3 shows a sketch of GPEI Spearmint. The number of shared layers (if any) between separate networks, the number of hidden layers for each network, the learning rate, the activation function type, the number of nodes in each layer and the size of the mini-batches are considered as the tunable hyperparameters.

Table 2 shows the values we considered for the parameters to be tuned. In most of our experiments, the best settings are to use totally separated DNNs (Multi-agent structure = 0) with Softplus activation functions, learning rate 0.01, and 4 hidden layers. The results appear to be quite quite insensitive to the mini-batch size.

Because the episodes have a finite horizon, the initial values of the state variables can have a significant impact on the training process. Of course, in practice, one would set the initial values to the actual values of the inventory levels at

Table 2: Hyperparameter values for DNN-SMEIO framework.

	parameter	possible values	notes
1	multi-agent structure	{0,1}	0: totally separated, 1: sharing first layers
2	# hidden layers	{1,2,3,4,5}	total number of layers is this number plus 2
3	# shared layers	{1,2,3}	only used if multi-agent structure = 1
4	activation function	softplus, leakyReLU, ReLU	
5	# nodes in each layer	{8,12,16,24,36,48}	
6	mini-batch size	{1,2,5,10}	
7	learning rate	{0.0005,0.001,0.005,0.01,0.02}	
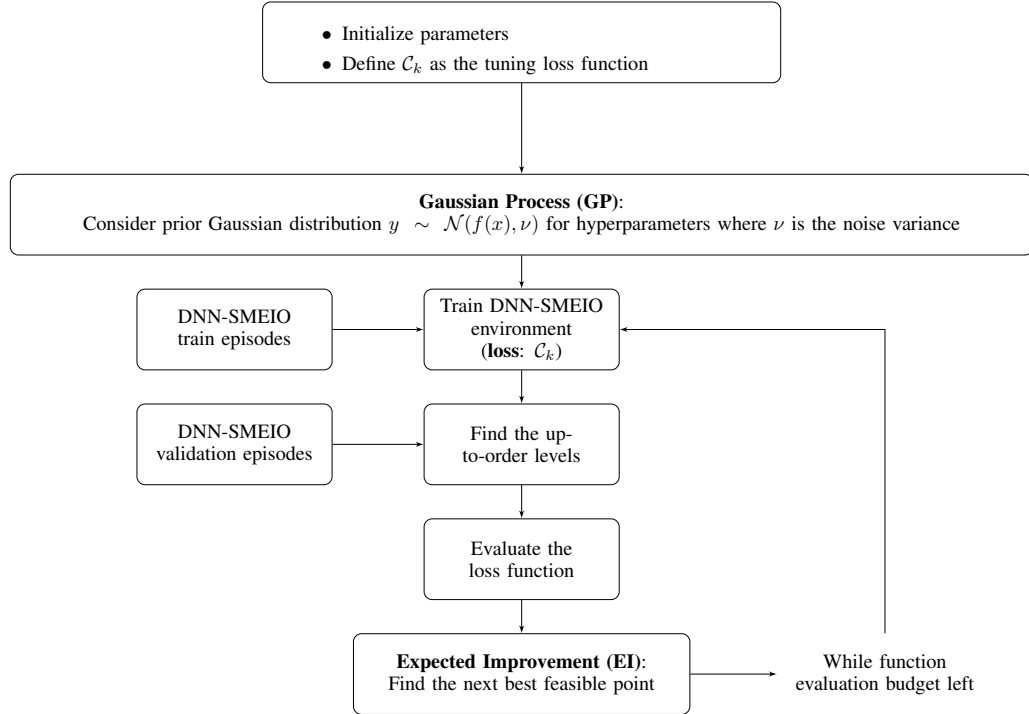
Figure 3: Tuning DNN-SMEIO via GPEI Bayesian optimization

Table 3: OUL and cost comparison for single-node inventory instances

| | | $L = 0$ | | | | $L = 1$ | | | |
| | demand | analytical | | DNN | | analytical | | DNN | |
case #	distribution	OUL	cost	OUL	cost	OUL	cost	OUL	cost
1	$\mathcal{N}(10,1)$	0	0	0	0	10.67	12.71	10.68	12.71
2	$\mathcal{N}(10,2)$	0	0	0	0	11.35	25.42	11.50	25.47
3	$\mathcal{N}(50,1)$	0	0	0	0	50.67	12.71	50.58	12.75
4	$\mathcal{N}(50,5)$	0	0	0	0	53.37	63.56	53.30	63.59
5	$\mathcal{N}(100,1)$	0	0	0	0	100.67	12.71	100.77	12.75
6	$\mathcal{N}(100,5)$	0	0	0	0	103.37	63.56	103.28	63.58
7	$\mathcal{N}(100,10)$	0	0	0	0	106.74	127.11	106.79	127.12

the start of the horizon. For training, however, in each episode we chose to initialize the raw-material inventories to 0 and the finished-goods inventory levels to the lead-time demand. That is, we set $IL_{j0} = \mu_j L_j$ and $IL_{ji0}^r = 0$ for all $j \in \mathcal{N}$ and $i \in \mathcal{U}_j$. The system is therefore "primed" with enough inventory to meet the upcoming expected demand.

5.2 Single-Node SCN

In this section, we consider a simple SCN consisting of a single node. In particular, the node has holding and stockout costs of $h = 10$ and $p = 30$, respectively. The demand per period is normally distributed; we consider various means and standard deviations. We set the holding cost $h = 10$ and shortage cost $p = 30$ for all the cases. This means critical ratio equals 0.75. We consider a short time horizon of $T = 2$ periods, with no salvage value ($v(x) = 0$ for all x). We consider two settings for the lead time, one in which $L = 1$ and one in which $L = 0$. In the $L = 0$ case, the decision maker sees the demand and then places an order; there is no stochasticity. The optimal action is simply to

order the realized demand value, and the optimal cost is 0. This is not a typical setting, but we examine it to evaluate the learning process of the DNN-SMEIO framework in a very simple case.

The $L = 1$ case is mathematically equivalent to the newsvendor problem. (Recall that in the classical newsvendor problem, the order is received before the demand is observed, whereas the opposite is true in our sequence of events. Therefore, our model is equivalent to the newsvendor model if we set $L = 1$.)

Table 3 reports the 7 instances we tested. The table shows the demand distribution, the optimal OUL and average cost per period, and the OUL and average cost given by the DNN, for both the $L = 0$ and $L = 1$ cases. The optimal OULs were found using an analytical approach. For the DNN solutions, we calculate the cost using the base-stock policy simulation. One can see that the results are very close; the DNN finds near-optimal OULs for these instances. The largest relative error between DNN-SMEIO approach and the analytical solutions are 1.32% and 0.31% for the OUL and cost values, respectively.

Figure 4 illustrates the loss function and OUL convergence curves for instances 1 and 7 from Table 3 and for both lead-time settings. We train the DNN-SMEIO for 50000 training episodes. The x-axes contain 500 points. After each 100 episodes of training, a new set of episodes is conducted for testing. The cost and OULs are calculated, which corresponds to a single point in the respective figures. One can stop the training much sooner than 50000 episodes. For instance, case 1 reaches its best result after only 1500 training episodes. We let the process continue only to demonstrate the method's stability. The black dotted lines for the OULs are the true optimal values. However, for the loss function, we simulate the process identically to the one that DNN-SMEIO is trained over (i.e., we consider the same initialization and episode horizon).

5.3 Serial SCN

In this section, we consider instances of serial SCNs. First, we discuss the comparison structure through a three-echelon serial system, which is illustrated in Figure 5 and used as an example by Snyder and Shen [12]. The network has 3 nodes and therefore 3 separate decisions to be made. There is a single external customer with normally distributed demand $\mathcal{N}(5, 1)$. The (local) holding costs increase as one moves downstream, and there is a shortage cost only at the furthest downstream node. Snyder and Shen [12] report the optimal OULs for this system under an infinite-horizon, continuous-review environment, based on the Clark–Scarf recursive method [25, 26]. However, as mentioned previously, our environment is a finite-horizon, periodic-review environment. Although the two environments are not strictly comparable, our time horizon (we use $T = 10$) is long enough so that the system approximately reaches steady state and can therefore be compared to the infinite-horizon case. Moreover, after the agents report their OULs, we use a simple base-stock policy simulation to find the cost. Nevertheless, because of the inherent differences between our assumptions (finite horizon, periodic review, cost evaluation by simulation) and those of the Clark-Scarf method (infinite horizon, continuous review, analytical cost evaluation), we also consider a third approach for optimizing the OULs: derivative-free optimization (DFO). We use the Trust Region DFO (DFO-TR) method [27], which is a model-based DFO method and is well-known to use as few function evaluations as possible. DFO can be used to explore the best possible OULs, so we use them as an additional benchmark against which to compare the DNN-SMEIO method.

The methods see the optimal OULs neither at the initialization nor during the training process. The problem settings are reported in Table 4, and the analytical, DNN-SMEIO and DFO results are presented in Table 5. As one can see,
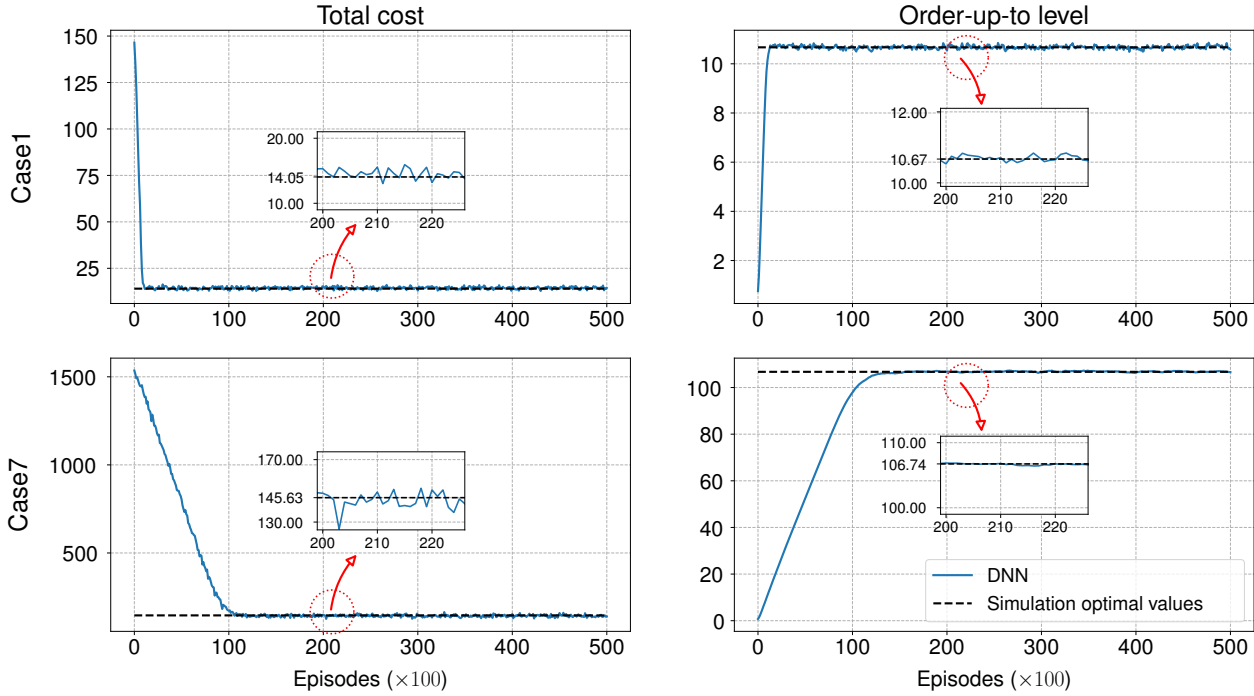
Figure 4: Loss function and OUL convergence curves for single-node inventory instances case 1 and 7.
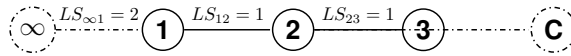
Figure 5: Three-echelon serial SCN.

the DNN results are quite close to the ones suggested by the analytical approach. For instance, the difference between the best cost for the case discussed above (case 3) achieved by the proposed framework (47.90) and the cost obtained by simulation using the exact OULs (47.65) is less than 1%. DFO gives considerably worse results for cases 7 and 8 but performs as well as the other approaches for the rest of the cases.

The convergence curves for the loss function and the OULs of serial SCN case 3 are provided in Figure 6. Training and testing procedures are similar to those discussed in the previous section. One can see that the DNN-SMEIO method is stable to a great extent.

5.4 Assembly SCNs

We consider two separate assembly structures, which are illustrated in Figure 7. Each structure contains three echelons. There are 10 and 11 OUL decisions to be made for assembly SCNs 1 and 2, respectively. For each of these structures, we consider 5 separate cases having different holding and shortage costs, lead-times and demand distributions. We use DFO, coordinate descent (CD), and enumeration as heuristic benchmarks to compare our method against. For CD and enumeration, each candidate solution is evaluated by simulating the system for 3 trials, each consisting of 200 periods; one solution is considered superior to another if the mean of the 3 total costs (one per trial) for that solution

Table 4: Serial SCNs settings

case	# echelons	demands	holding cost (per item)	shortage cost (per item)	lead-time
1	2	$\mathcal{N}(3, 0.5)$	$(5, 8.2)$	$(0, 25.5)$	$(1, 1)$
2	2	$\mathcal{N}(6, 1.5)$	$(1.9, 4.1)$	$(0, 11.3)$	$(2, 1)$
3	3	$\mathcal{N}(5, 1)$	$(2, 4, 7)$	$(0, 0, 37.12)$	$(2, 1, 1)$
4	3	$\mathcal{N}(50, 3)$	$(5, 10, 25)$	$(0, 0, 50)$	$(2, 1, 1)$
5	3	$\mathcal{N}(100, 5)$	$(25, 25, 50)$	$(0, 0, 100)$	$(1, 2, 2)$
6	3	$\mathcal{N}(100, 10)$	$(10, 20, 30)$	$(0, 0, 100)$	$(1, 1, 1)$
7	4	$\mathcal{N}(3, 0.4)$	$(4, 5.75, 7.90, 10.8)$	$(0, 0, 0, 35.5)$	$(1, 1, 1, 1)$
8	4	$\mathcal{N}(5, 1.2)$	$(5, 5, 5, 10)$	$(0, 0, 0, 30)$	$(1, 1, 1, 1)$
9	5	$\mathcal{N}(80, 4)$	$(10, 20, 30, 40, 50)$	$(0, 0, 0, 0, 200)$	$(1, 1, 1, 1, 1)$
10	5	$\mathcal{N}(25, 2)$	$(5, 10, 25, 50, 50)$	$(0, 0, 0, 0, 150)$	$(2, 1, 1, 1, 1)$

Table 5: OUL and cost comparison for serial SCNs

|	Analytical		DNN		DFO	
case	OULs	cost	OULs	cost	OULs	cost
1	$(2.91, 3.64)$	22.21	$(2.91, 3.72)$	22.34	$(1.22, 5.10)$	22.55
2	$(12.58, 7.60)$	23.07	$(12.58, 7.65)$	23.17	$(12.05, 7.58)$	23.20
3	$(10.69, 5.53, 6.49)$	47.65	$(10.08, 5.39, 6.64)$	47.90	$(10.54, 5.35, 6.57)$	50.01
4	$(101.45, 51.40, 52.7040)$	879.88	$(99.29, 51.03, 52.71)$	885.63	$(97.02, 53.59, 53.02)$	885.49
5	$(71.026, 228.29, 207.04)$	10568.23	$(87.71, 204.63, 208.90)$	10625.01	$(79.33, 211.20, 208.51)$	10695.88
6	$(99.53, 102.58, 114.05)$	3630.14	$(93.36, 103.42, 114.26)$	3651.63	$(95.83, 100.87, 117.90)$	3638.18
7	$(2.78, 3.13, 3.19, 3.60)$	63.39	$(2.78, 3.13, 3.19, 3.74)$	63.84	$(-12.03, -8.86, -3.93, 1.90)$	592.51
8	$(-3.80, 9.80, 9.80, 6.35)$	101.48	$(1.48, 6.12, 7.00, 6.46)$	104.04	$(-4.96, -3.69, -1.90, 0.02)$	674.62
9	$(80.15, 80.15, 81.17, 81.68, 86.99)$	8559.85	$(76.83, 78.02, 79.60, 81.62, 87.40)$	8678.38	$(80.49, 77.62, 80.04, 77.45, 92.18)$	8585.50
10	$(51.57, 26.30, 25.05, 20.25, 33.01)$	2500.79	$(48.40, 25.65, 24.02, 22.90, 30.12)$	2581.41	$(49.44, 25.51, 23.04, 23.82, 31.17)$	2527.1
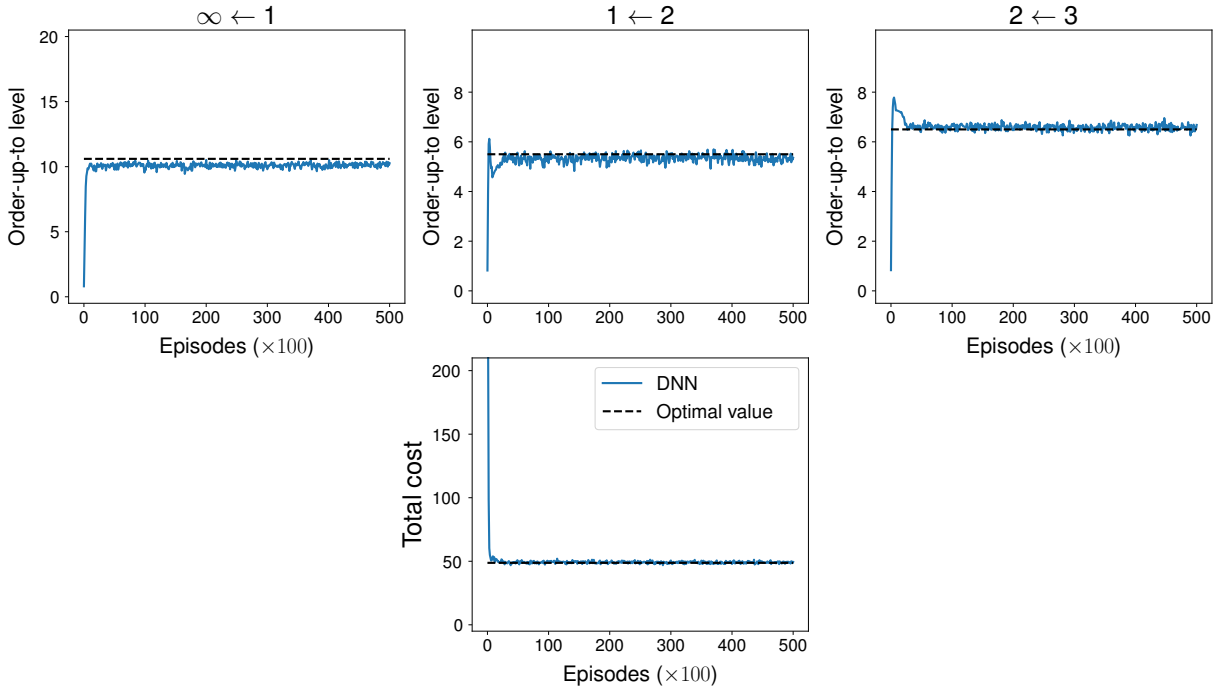
Figure 6: Total cost and OUL decisions convergence curves for serial SCN case 3

Table 6: Cost comparisons for assembly SCNs

	Case	DNN-SMIO	CD	enumeration	DFO
assembly1	1	40.55	40.27	40.34	233.45
	2	103.77	101.59	101.47	482.63
	3	163.15	161.30	161.13	441.43
	4	37.49	35.97	35.98	139.77
	5	29.04	27.53	27.45	36.03
average		74.80	73.33	73.27	324.32
assembly2	1	93.94	90.40	90.54	116.38
	2	23.00	22.43	22.48	25.75
	3	86.61	82.67	82.32	90.71
	4	34.62	34.04	34.17	42.35
	5	30.98	28.19	27.96	62.76
average		53.83	51.55	51.49	67.59

is smaller than that of the other. Lower and upper bounds for both CD and enumeration are set equal to $0.75D$ and $2D$, respectively, where D is the mean lead-time demand observed by the node. For enumeration, the resulting range was discretized into 10 equal intervals. Moreover, we restricted the OULs to be equal at all nodes within a given echelon, which is optimal given the symmetries in the SCNs, and which therefore reduces the search space. Note that we did not make the same restriction for the DNN method, meaning that we are providing an advantage to the benchmark methods. The complete specifications of the parameters of the assembly system instances, as well as the OULs suggested by each method, are presented in Tables 1 and 2 of the supplement.

For all solutions produced by all methods, we evaluated the cost by simulating the system for 10 trials, each consisting of 10,000 periods. The mean of the 10 total costs (one per trial) is reported in the Table 6. We observe that the costs of the solutions suggested by DNN are very close to those from CD and enumeration, even though the search space was restricted as described above for CD and enumeration but not for DNN.

DFO is inferior compared to the other approaches. For four of the cases, the DFO OULs did not converge to reasonable values, resulting in a very large cost. There are at least three possible reasons for this. First, we may have exceeded the number of variables that can be properly handled by DFO. Second, DFO methods are generally local methods, and DFO may become trapped in local minimum. Third, the performance of DFO heavily depends on the starting point, and we may have started from OULs that are far from the optimal ones, although we tried to alleviate its effect by starting with lead-time demands as initial inventory levels.

5.5 Mixed SCN

From an inventory optimization perspective, there are no known analytical solutions for mixed SCNs, other than computationally intensive, enumeration-based approaches. Therefore, providing a stable numerical approach is greatly desirable. In this section, we demonstrate the performance of our method in finding OULs for multiple nodes of a mixed SCN simultaneously. We consider the SCN illustrated in Figure 1. The network has 3 echelons, 5 nodes, 7 edges and two customers having independent stochastic normal distributions $\mathcal{N}(5,1)$ as their demands. The third echelon nodes, 4 and 5, are "assembly-*and*" nodes (see the definition in Section 4.), and nodes 2 and 3 are distribution nodes, making the SCN a mixed one. We consider a time horizon of $T = 10$. All 7 OULs are required to be optimized
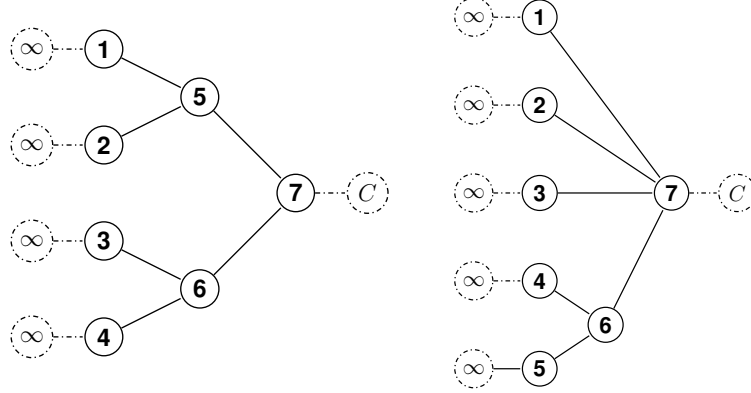
Figure 7: Assembly SCN structures. Left figure: assembly SCN (1) and the right figure assembly SCN (2).

Table 7: Mixed SCN parameters

Echelon	Edge	Holding cost (per item)	Shortage Cost (per item)	Shipment lead-time	Order lead-time	Initialization
1	(0,1)	2	4	2	0	40
2	(1,2)	4	12	1	0	10
2	(1,3)	4	12	1	0	10
3	(2,4)	7	37.12	1	0	5
3	(2,5)	7	37.12	1	0	5
3	(3,4)	7	37.12	1	0	5
3	(3,5)	7	37.12	1	0	5

simultaneously. Table 7 shows the parameters used in the study. Figure 8 illustrates the convergence curves of the DNN approach. The black dotted lines are the best results, corresponding to the minimum costs achieved by DNN.

We allow the algorithm to restart the learning procedure from scratch using the best OULs found previously as the new initial inventory levels until there is no extra improvement in the objective value. We do this for two reasons. First, we do not know whether the objective functions of mixed SCN inventory problems are convex or nonconvex. In case we are minimizing a nonconvex objective, we do not wish the optimization algorithm to be stuck in a local minimum. Second, we have no idea how far away the initial inventory levels are from the optimal OULs. Considering that we are dealing with finite-horizon ($T = 10$) episodes, this can significantly slow the method's learning procedure. Hence, we use the OULs found the previous time as the initial inventory levels of the next learning procedure. We set stopping criteria so that the procedure terminates when there is less than a 1% change in the objective value.

One can observe the following based on Figure 8:

- The jumps in the subplots are due to the restart procedure discussed previously. The importance of better initialization can be seen in the total cost spike at the restart. Although the OUL decisions are random and not necessarily close to their optimal values, at the restart, the total cost is almost half of the cost at the beginning of the procedure.

- The farther downstream the node is, the noisier the OULs are.

- Similar to the previous examples, the agents farther upstream need fewer episodes to be optimized compared with those downstream.
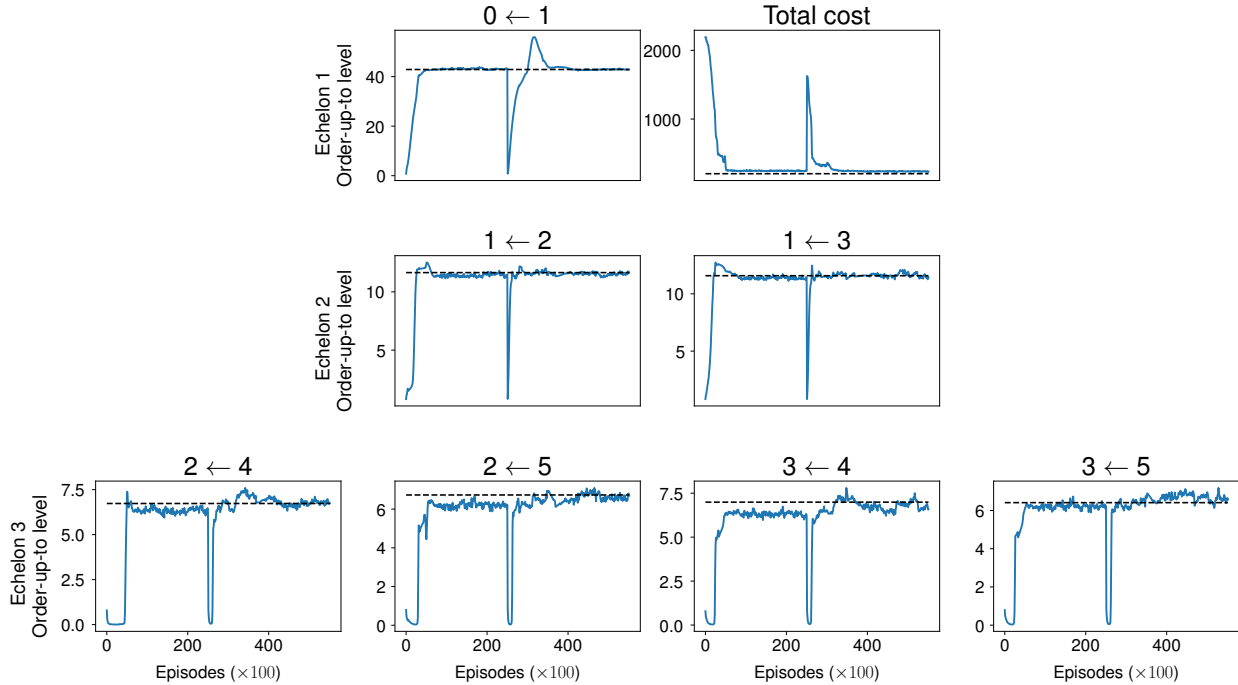
Figure 8: Total cost and OUL decisions for mixed SMEIO.

Table 8: Top 5 randomized solutions and DNN results comparison for mixed SCN

| Echelon | Edge | Random search parameters | OULs | | | | | |
			5 best randomly generated solutions					DNN
1	(0,1)	$40 + \|\mathcal{N}(0,4)\|$	40.00	43.58	41.22	45.50	41.19	42.87
2	(1,2)	$10 + \|\mathcal{N}(0,2)\|$	13.54	12.84	10.57	12.49	13.07	11.65
2	(1,3)	$10 + \|\mathcal{N}(0,2)\|$	13.72	12.02	11.57	10.62	13.21	11.58
3	(2,4)	$5 + \|\mathcal{N}(0,2)\|$	6.31	7.58	5.08	6.99	8.03	6.73
3	(2,5)	$5 + \|\mathcal{N}(0,2)\|$	6.50	5.85	5.77	5.36	9.07	6.73
3	(3,4)	$5 + \|\mathcal{N}(0,2)\|$	5.84	7.79	7.54	6.36	5.14	6.99
3	(3,5)	$5 + \|\mathcal{N}(0,2)\|$	5.16	5.16	9.45	5.39	8.00	6.41
Total Cost			215.05	214.72	214.45	212.97	211.90	208.80

Because there are no reliable algorithms for general mixed systems in the literature, we use a randomized search as a benchmark to compare with the DNN-SMEIO framework. We consider 100 separate randomly generated solutions and for each run we average over 2000 episodes. (Note that this is more than 20 times the number of environment interactions that the DNN approach requires; recall that DNN-SMEIO reaches its best results in at most 10^4 episodes.)

We select candidate solutions randomly by setting each node's OUL equal to the mean demand of the node plus the absolute value of a random, zero-mean, normal random variate. We use random variates with larger standard deviations for nodes with larger demand means to allow for a search space that is likely to contain good solutions. Table 8 shows the parameters for the random variates, as well as a comparison between the 5 best solutions found using this method and the solution found by the DNN approach. As one can see, the DNN performs better than all of the randomly generated solutions.

Table 9: Comparison results between DFO, Spearmint and DNN approaches for Mixed SCN

	Echelon	Edge	Initial Value Choosing		Results with 25 function evaluations		Best results without restrictions		DNN
			DFO	Spearmint	DFO	Spearmint	DFO	Spearmint	
	1	(0,1)	40	[40, 48]	47.69	41.45	43.73	43.80	42.87
	2	(1,2)	10	[10, 14]	12.45	12.34	11.46	11.45	11.65
	2	(1,3)	10	[10, 14]	12.62	11.76	11.46	11.49	11.58
OUL decisions	3	(2,4)	5	[5, 9]	5.51	5.39	5.77	5.80	6.73
	3	(2,5)	5	[5, 9]	5.58	5.54	5.77	5.77	6.73
	3	(3,4)	5	[5, 9]	5.53	7.02	5.77	5.78	6.99
	3	(3,5)	5	[5, 9]	5.40	5.63	5.78	5.78	6.41
Total Cost					215.21	214.66	206.35	206.36	208.80

We compare the performance of the proposed method with two additional alternatives: derivative-free optimization (DFO) [28] and Spearmint Bayesian optimization [24]. Each DFO or Spearmint step can be reduced to three parts: first, a suggestion of OULs; second, an independent simulation run to obtain the objective value for that suggestion; third, a suggestion of a new set of OULs based on an optimization algorithm. We consider 2000 episodes to allow each simulation run to converge. We test two cases, one in which there is no explicit bound on the number of function evaluations and one in which we restrict them to make sure the number of interactions between agents and the environment stays equal for the DFO, Spearmint, and DNN approaches. For this instance, 25 function evaluations for DFO or Spearmint were conducted, each of which consists of 2000 episodes, so we allowed DNN to use 50,000 episodes to reach its result. The only stopping criteria for the cases without an upper bound on the number of function evaluations are either having 100 algorithm steps without any improvement or having the last 10 improvements be less than 0.5% of the cost function value. In addition, DFO-TR requires an initial guess to start the algorithm. We use the lead-time demand means for this purpose. Spearmint, however, requires an interval for each decision variable. We consider lead-time demand means as the lower bounds and an acceptable range (at least twice the lead-time demand standard deviation) to cover possible solutions. Both algorithms might suffer from a dependence on these initial conditions.

Table 9 shows the comparison results. DFO and Spearmint find marginally better values in terms of objective function value for the cases without any upper bound on the number of function evaluations. However, restricting them to have the same number of interactions with the environment as the DNN has results in inferior performance of these alternatives compared to the DNN. Furthermore, because the optimal OULs for this mixed SCN structure happen to be close to the lead-time demand means, this biases the experiment in favor of DFO and Spearmint, which are given the lead-time demand means (or a small interval containing them) as initial values. In the next section, however, we investigate a more realistic case study and explore the comparison further.

5.6 Complex SCN

In this section we introduce a case study of a general-structured SCN with realistic settings such as nonlinear holding and penalty costs and salvage values. For this comparison, We consider trust-region derivative-free optimization (DFO-TR) [27, 29], GPEI Bayesian along with simple yet powerful random search techniques as alternative methods. We consider the complex SCN shown in Figure 9. The SCN contains 7 nodes and 13 edges. Nodes 5, 6 and 7 are "assembly-*and*" nodes. That is, to produce one item at any of these nodes, one item from each of its predecessors is required. In this case study, holding costs are considered to be piecewise linear instead of linear. When the number
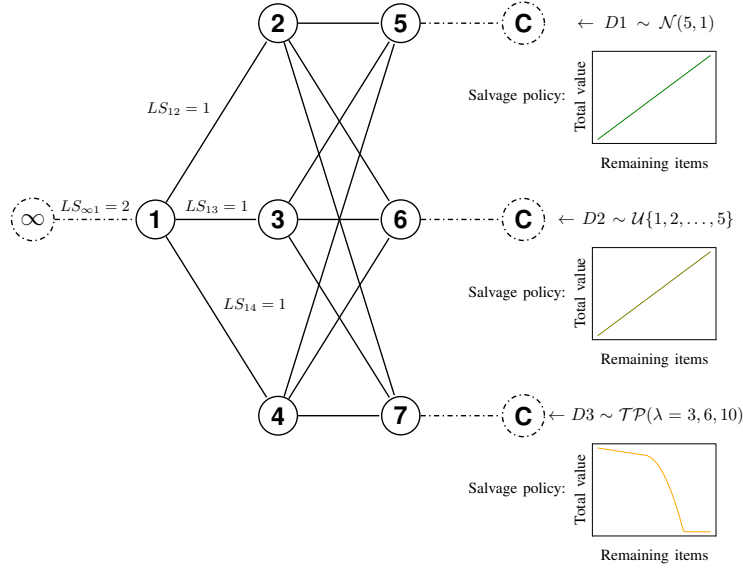
Figure 9: Mixed SCN with customer demands and salvage policies.

of items in inventory increases beyond a certain threshold, the cost per item decreases. On the other hand, shortage penalty costs are considered to be nonlinear, and the cost per item increases when the shortage quantity is greater. There are three leaf nodes that see customer demands. One follows an independent normal distribution $\mathcal{N}(5,1)$, the second follows a discrete uniform distribution $\mathcal{U}\{1,2,\ldots,5\}$, and the third follows a two-sided truncated Poisson distribution $\mathcal{TP}(\lambda = 3; 6; 10)$, where 3 is the distribution parameter and 6 and 10 are the beginning and the end of the possible values, respectively. We consider $T = 10$ as the episode horizon length.

We assume that the three leaf nodes have different salvage functions that represent the cost or reward incurred based on the inventory level at the end of the horizon. Nodes 5 and 6 have linear rewards, with different slopes; for example, these might model situations in which excess inventory can be sold for a per-unit cost at the end of the horizon. However, for node 7, we assume a nonlinear salvage reward policy to clear the remaining inventory. Up to a threshold, the price per item is considered to be high because there are only few items left. The price per item then reduces drastically to reach the second threshold and thereafter it is fixed.

Table 10 provides further details about this problem instance. In the table, $(x < \text{threshold}, f_1(x), f_2(x))$ means that when $x < \text{threshold}$, the function $f_1(x)$ is used, and afterwards $f_2(x)$ is used. The initial inventories are set equal to the demand mean. For instance, $\mathbb{E}(\mathcal{TP}(\lambda = 3, 6, 10)) = 7.58$ is considered as the initialization values for the $(2 \leftarrow 7), (3 \leftarrow 7)$ and $(4 \leftarrow 7)$ decisions.

First, we compare the DNN-SMEIO method with the randomized approach. We consider 400 runs of 5000 episodes each. Table 11 shows the results of this comparison. The intervals of possible OULs used for the randomized approach contain the best solution found by DNN-SMEIO. One can see the clear advantage of the proposed method over the randomized approach.

We further investigate a comparison between DFO, Spearmint, and the proposed DNN approach. We do not put an upper limit on the number of interactions with the environment for the DFO and Spearmint approaches. (This biases

Table 10: Case study mixed SCN parameters[*]

Edge	Holding cost (for x units on hand)	Shortage Cost (for x backorders)	Shipment lead-time	Salvage reward (for x units on hand)	Initialization
(0,1)	$2x$	$4x$	2	—	45.24
(1,2)	$(x < 3, 4x, 3x)$	$(x < 3, 12x, 4x^2)$	1	—	15.08
(1,3)	$(x < 3, 4x, 3x)$	$(x < 3, 12x, 4x^2)$	1	—	15.08
(1,4)	$(x < 3, 4x, 3x)$	$(x < 3, 12x, 4x^2)$	1	—	15.08
(2,5)	$(x < 3, 7x, 6x)$	$(x < 3, 36x, 12x^2)$	1	$1.25x$	5.00
(2,6)	$(x < 3, 7x, 6x)$	$(x < 3, 36x, 12x^2)$	1	$1.5x$	2.50
(2,7)	$(x < 3, 7x, 6x)$	$(x < 3, 36x, 12x^2)$	1	$(x < 2, 15 - 0.5x, max(-3.5x^2 + 14x, 3))$	7.58
(3,5)	$(x < 3, 7x, 6x)$	$(x < 3, 36x, 12x^2)$	1	$1.25x$	5.00
(3,6)	$(x < 3, 7x, 6x)$	$(x < 3, 36x, 12x^2)$	1	$1.5x$	2.50
(3,7)	$(x < 3, 7x, 6x)$	$(x < 3, 36x, 12x^2)$	1	$(x < 2, 15 - 0.5x, max(-3.5x^2 + 14x, 3))$	7.58
(4,5)	$(x < 3, 7x, 6x)$	$(x < 3, 36x, 12x^2)$	1	$1.25x$	5.00
(4,6)	$(x < 3, 7x, 6x)$	$(x < 3, 36x, 12x^2)$	1	$1.5x$	2.50
(4,7)	$(x < 3, 7x, 6x)$	$(x < 3, 36x, 12x^2)$	1	$(x < 2, 15 - 0.5x, max(-3.5x^2 + 14x, 3))$	7.58

[*] All of the costs listed below equal 0 if $x < 0$.

Table 11: Top 5 randomized solutions and DNN results comparison for mixed SCN

			OULs					
Echelon	Edge	Random search parameters	5 best randomly generated solutions					DNN
1	(0,1)	$45.24 + |\mathcal{N}(0,50)|$	102.73	102.56	111.44	100.40	100.40	101.44
2	(1,2)	$15.08 + |\mathcal{N}(0,5)|$	19.50	17.96	22.71	22.75	15.91	18.75
2	(1,3)	$15.08 + |\mathcal{N}(0,5)|$	14.25	14.51	18.72	18.81	19.84	20.82
2	(1,4)	$15.08 + |\mathcal{N}(0,5)|$	15.51	15.10	20.05	20.51	21.85	21.48
3	(2,5)	$5 + |\mathcal{N}(0,5)|$	7.52	6.91	6.06	5.33	6.26	7.29
3	(2,6)	$2.5 + |\mathcal{N}(0,5)|$	7.75	10.38	5.51	8.95	7.16	6.95
3	(2,7)	$7.58 + |\mathcal{N}(0,5)|$	9.38	8.83	12.52	12.29	9.84	10.28
3	(3,5)	$5 + |\mathcal{N}(0,5)|$	8.03	7.87	9.65	9.39	6.47	7.21
3	(3,6)	$2.5 + |\mathcal{N}(0,5)|$	6.99	6.55	4.08	5.19	5.73	6.06
3	(3,7)	$7.58 + |\mathcal{N}(0,5)|$	11.87	11.14	8.17	8.52	9.17	9.48
3	(4,5)	$5 + |\mathcal{N}(0,5)|$	7.04	7.23	9.11	5.16	5.02	6.11
3	(4,6)	$2.5 + |\mathcal{N}(0,5)|$	8.51	6.70	8.85	5.54	5.15	6.5
3	(4,7)	$7.58 + |\mathcal{N}(0,5)|$	10.10	9.84	10.48	8.25	7.71	9.38
Total Cost			534.06	526.75	522.18	519.68	514.69	478.61

the experiment in favor of alternative approaches.) Table 12 tabulates the results. One can see the clear advantage of the proposed method over the alternatives. Both DFO and Spearmint fail to find results that are nearly as good as the randomized method. We also emphasize the fact that choosing a suitable interval for the Spearmint method requires expert knowledge, trial and error or multiple runs of the algorithm. For instance, based on the results reported in Table 11, we already knew that OULs greater than 100 should be considered for the $0 \leftarrow 1$ decision. Consequently, we assumed a large interval of $[45, 150]$ for this decision variable, but without this prior knowledge, one might miss important regions for the decision variable. Figure 10 illustrates the learning process and the optimal values for all 13 OULs. The jumps in the subplots are related to the restarting procedure discussed in the previous section. We stop the procedure simply when there is no gain anymore in terms of objective function values. The best cost values found before restarting the training procedures were 626.17, 479.68, and 478.61. Therefore, we stop restarting the training procedure after two retrainings. Figure 10 provides the following insights:

- When the cost structures are more complex such as nonlinear (potentially very large) shortage penalties, the optimal solutions are biased towards ordering extra items in the upstream echelons. DNN-SMEIO sets the $(0 \leftarrow 1)$ OUL to 101.44, while the average lead-time demand that node 1 realizes is only $3 \times 15.08 = 45.24$. The difference between the two values is considerable, and if a practitioner without this knowledge uses

20
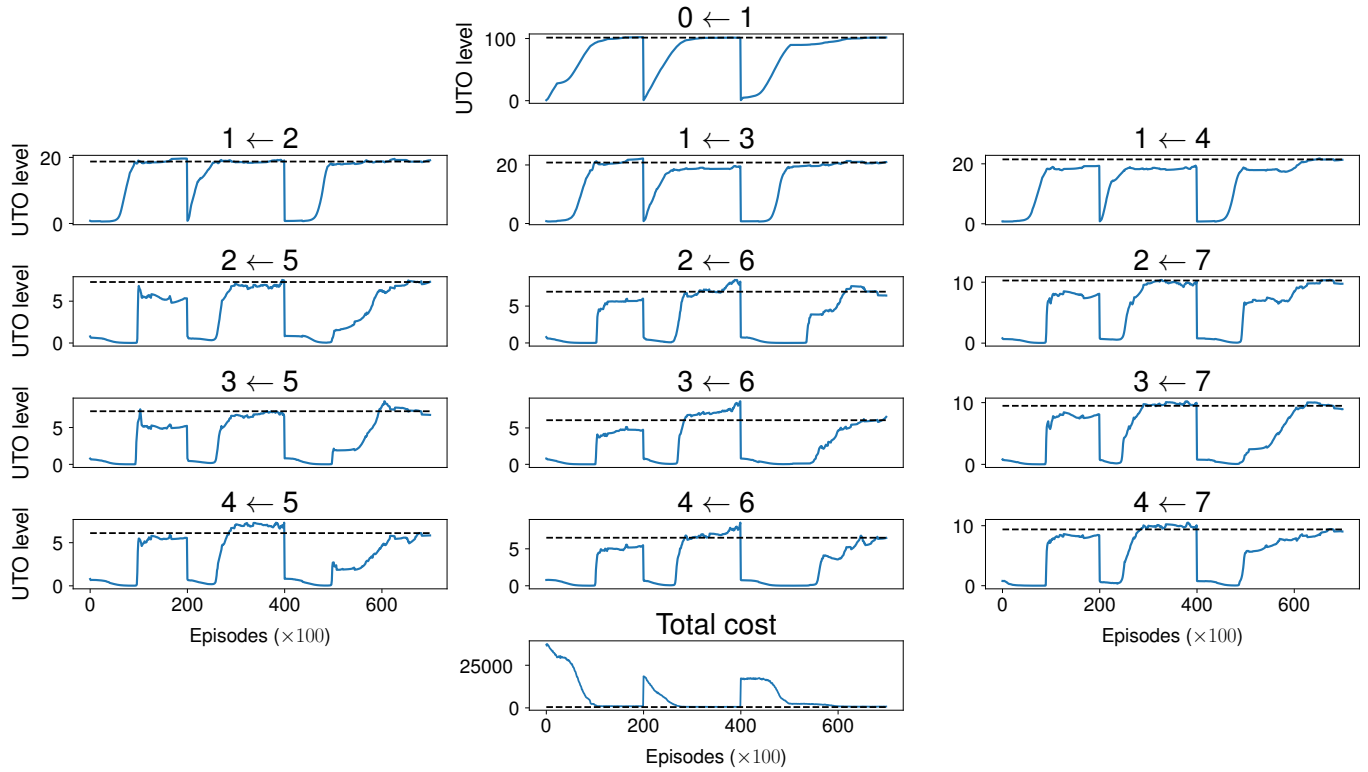
Figure 10: Total cost and OUL decisions for the case study SCN.

enumeration or an *ad hoc* approach, it is probable that the best value found by the DNN-SMEIO method would not even be in the interval selected by the practitioner.

- In consensus with what we observed in the previous examples, the upstream echelons learn their OUL values more quickly than those downstream.

We further conduct a cost comparison between the DFO and DNN approaches for 4 new instances based on the complex SCN structure shown in Figure 9. Table 13 shows the cost comparison results for the new instances plus the one already discussed earlier on the structure. We focused on different salvage costs and demand distributions for this comparison. We refer the reader to Table 3 of the supplement for the structure details and suggested OULs. As can be seen, the DNN approach outperforms DFO for all five cases.

Table 12: Comparison results between DFO, Spearmint and DNN approaches for case study SCN

| | | | Alternatives | | | | DNN |
| | | | Initializing | | results | | |
	Echelon	Edge	DFO	Spearmint	DFO	Spearmint	
	1	(0,1)	45.24	[45, 150]	99.15	93.86	101.44
	2	(1,2)	15.08	[15, 45]	15.94	29.93	18.75
	2	(1,3)	15.08	[15, 45]	16.01	24.40	20.82
OUL decisions	2	(1,4)	15.08	[15, 45]	15.99	26.01	21.48
	3	(2,5)	5	[2.5, 15]	5.20	5.08	7.29
	3	(2,6)	2.5	[2.5, 15]	3.13	3.24	6.95
	3	(2,7)	7.58	[2.5, 15]	7.62	8.02	10.28
	3	(3,5)	5	[2.5, 15]	5.41	5.21	7.21
	3	(3,6)	2.5	[2.5, 15]	2.84	3.41	6.06
	3	(3,7)	7.58	[2.5, 15]	7.52	8.22	9.48
	3	(4,5)	5	[2.5, 15]	5.44	5.17	6.11
	3	(4,6)	2.5	[2.5, 15]	2.77	3.37	6.5
	3	(4,7)	7.58	[2.5, 15]	7.52	8.37	9.38
Total Cost					644.41	618.44	478.61

Table 13: Cost comparisons for complex mixed SCN instances

Instance	DNN-SMIO	DFO
1	380.95	402.41
2	419.13	442.42
3	407.83	408.27
4	379.31	400.04
5	478.61	644.41
average	426.43	446.32

6 Conclusion

This research studies simultaneous decision-making for stochastic multi-echelon inventory optimization with arbitrary SCN topologies, demand distributions, and cost structures, using deep neural networks as decision makers, considering a finite-horizon. We introduce pairwise modeling of SMEIOs and associate a DNN to each edge in need of decision making. The DNNs constantly interact with their environment (the supply chain network) and aim to learn the OULs minimizing the total network cost. We assume that the demand distribution as well as the inventory levels are known to all agents. Our research is one of the first works considering deep neural networks as joint decision makers in an SMEIO framework that can suggest clear and interpretable OULs as an output.

The findings indicate the effectiveness of the method both in terms of its accuracy compared to analytical exact solutions and versus alternatives such as DFO and Spearmint Bayesian optimization, as well as in terms of the computational expense (interactions with environment) compared to enumeration methods. The convergence curves shown for the single-node newsvendor, serial, and mixed systems validate the stability of the framework. For mixed supply networks with more advanced cost schemes, findings suggest that upstream echelons hold more items compared to their demand means than downstream echelons. For example, the OUL found by DNN-SMEIO for the first echelon $(0 \leftarrow 1)$ in the complex SCN shown in Figure 9 is more than twice the demand it sees.

This study can be extended to cases in which the demands are auto-correlated and/or the decision makers/DNNs have partial information about the structure of the SCN. Another future study can be conducted to investigate closed-loop supply chain networks. In addition, considering time dependent OULs would be a desirable improvement to this method.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[2] Ilaria Giannoccaro and Pierpaolo Pontrandolfo. Inventory management in supply chains: a reinforcement learning approach. *International Journal of Production Economics*, 78(2):153–161, 2002.

[3] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[4] S Kamal Chaharsooghi, Jafar Heydari, and S Hessameddin Zegordi. A reinforcement learning model for supply chain ordering management: An application to the beer game. *Decision Support Systems*, 45(4):949–959, 2008.

[5] Afshin Oroojlooyjadid, MohammadReza Nazari, Lawrence Snyder, and Martin Takáč. A deep q-network for the beer game: A reinforcement learning algorithm to solve inventory optimization problems. *ArXiv preprint arXiv:1708.05924*, 2017.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *ArXiv preprint arXiv:1312.5602*, 2013.

[7] Pierpaolo Pontrandolfo, Abhijit Gosavi, O Geoffrey Okogbaa, and Tapas K Das. Global supply chain management: a reinforcement learning approach. *International Journal of Production Research*, 40(6):1299–1317, 2002.

[8] Tim Stockheim, Michael Schwind, and Wolfgang Koenig. A reinforcement learning approach for supply chain management. In *1st European Workshop on Multi-Agent Systems, Oxford, UK*, 2003.

[9] Tim van Tongeren, Uzay Kaymak, David Naso, and Eelco van Asperen. Q-learning in a competitive supply chain. In *2007 IEEE International Conference on Systems, Man and Cybernetics*, pages 1211–1216. IEEE, 2007.

[10] Gang Zhao and Ruoying Sun. Application of multi-agent reinforcement learning to supply chain ordering management. In *2010 Sixth International Conference on Natural Computation*, volume 7, pages 3830–3834. IEEE, 2010.

[11] Joren Gijsbrechts, Robert N Boute, Jan A Van Mieghem, and Dennis Zhang. Can deep reinforcement learning improve inventory management? performance on dual sourcing, lost sales and multi-echelon problems. *SSRN*, 2019.

[12] Lawrence V. Snyder and Zuo-Jun Max Shen. *Fundamentals of Supply Chain Theory*. Wiley, Hoboken, NJ, 2nd edition, 2019.

[13] Engin Topan, Z Pelin Bayındır, and Tarkan Tan. Heuristics for multi-item two-echelon spare parts inventory control subject to aggregate and individual service measures. *European Journal of Operational Research*, 256 (1):126–138, 2017.

[14] Lina Johansson, Danja R Sonntag, Johan Marklund, and Gudrun P Kiesmüller. Controlling distribution inventory systems with shipment consolidation and compound poisson demand. *European Journal of Operational Research*, 280(1):90–101, 2020.

[15] Shimon Bitton, Izack Cohen, and Morris Cohen. Joint repair sourcing and stocking policies for repairables using erlang-a and erlang-b queueing models. *IISE Transactions*, pages 1–16, 2019.

[16] Barry R Cobb. Lead time uncertainty and supply chain coordination in lost sales inventory models. *International Journal of Inventory Research*, 3(1):5–30, 2016.

[17] Amar Sapra. Dual sourcing in a serial system. *Production and Operations Management*, 26(12):2163–2174, 2017.

[18] Guangyin Xu, Jihao Feng, Fenglei Chen, Heng Wang, and Zhenfeng Wang. Simulation-based optimization of control policy on multi-echelon inventory system for fresh agricultural products. *International Journal of Agricultural and Biological Engineering*, 12(2):184–194, 2019.

[19] Ton de Kok, Christopher Grob, Marco Laumanns, Stefan Minner, Jörg Rambau, and Konrad Schade. A typology and literature review on stochastic multi-echelon inventory models. *European Journal of Operational Research*, 269(3):955–983, 2018.

[20] Awi Federgruen and Paul Zipkin. Approximations of dynamic, multilocation production and inventory problems. *Management Science*, 30(1):69–84, 1984.

[21] Stephen C. Graves and Sean P. Willems. Supply chain design: Safety stock placement and supply chain configuration. In A. G. de Kok and Stephen C. Graves, editors, *Supply Chain Management: Design, Coordination and Operation*, volume 11 of *Handbooks in Operations Research and Management Science*, chapter 3. Elsevier, Amsterdam, 2003.

[22] Afshin Oroojlooyjadid, Lawrence V Snyder, and Martin Takáč. Applying deep learning to the newsvendor problem. *IISE Transactions*, 52(4):444–463, 2020.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ArXiv preprint arXiv:1412.6980*, 2014.

[24] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.

[25] Andrew J Clark and Herbert Scarf. Optimal policies for a multi-echelon inventory problem. *Management Science*, 6(4):475–490, 1960.

[26] F. Chen and Y.-S. Zheng. Lower bounds for multi-echelon stochastic inventory systems. *Management Science*, 40(11):1426–1443, 1994.

[27] Afonso S Bandeira, Katya Scheinberg, and Luís N Vicente. Computation of sparse low degree interpolating polynomials and their application to derivative-free optimization. *Mathematical Programming*, 134(1):223–257, 2012.

[28] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*, volume 8. Siam, 2009.

[29] Mohammad Pirhooshyaran, Katya Scheinberg, and Lawrence V Snyder. Feature engineering and forecasting via derivative-free optimization and ensemble of sequence-to-sequence networks with applications in renewable energy. *Energy*, 196:117136, 2020.

Supplementary Material

1   Details of Assembly SCN Experiments

The complete specifications of the parameters of the assembly system instances in Section 5.4 of the main article, as well as the OULs suggested by each method, are presented in Tables 1 and 2 of the supplement.

2   Complex SCN Cost Comparison

Table 3 shows the structure details of the different experiments done for the Complex SCN studied in Section 5.6 of the main paper as well as suggested OULs by DNNSMIO and DFO. H, local holding cost coefficients, stockout cost coefficients and salvage reward coefficients

Table 1: Assembly(1) SCN cases details

Instance 1

							OUL		
Node	Predecessor	H	S	LT	Demand	DNNSMIO	CD	Enumeration	DFO
1	–	0.25	0	2	–	26.91	25.77	26.72	6.08
2	–	0.25	0	2	–	26.8	25.77	26.72	27.32
3	–	0.25	0	2	–	26.86	25.77	26.72	21.77
4	–	0.25	0	2	–	26.85	25.77	26.72	26.89
5	1	0.8	0	1	–	13.55	13.87	13.36	12.20
5	2	0.8	0	1	–	13.57	13.87	13.36	14.75
6	3	0.8	0	1	–	13.58	13.87	13.36	12.85
6	4	0.8	0	1	–	13.57	13.87	13.36	15.36
7	5	1.9	10	1	$N(13,(1.2)^2)$	14.64	15.08	15.16	14.41
7	6	1.9	10	1	$N(13,(1.2)^2)$	14.64	15.08	15.16	14.46
				Cost		40.55	40.26	40.34	233.45

Instance 2

							OUL		
Node	Predecessor	H	S	LT	Demand	DNNSMIO	CD	Enumeration	DFO
1	–	2	0	2	–	10.08	7.50	7.5	10.19
2	–	2	0	2	–	10.08	7.50	7.5	11.83
3	–	2	0	2	–	10.13	7.50	7.5	1.00
4	–	2	0	2	–	10.12	7.50	7.5	11.15
5	1	4	0	1	–	5.42	6.04	3.75	4.63
5	2	4	0	1	–	5.42	6.04	3.75	4.67
6	3	4	0	1	–	5.38	6.04	3.75	1.46
6	4	4	0	1	–	5.33	6.04	3.75	4.62
7	5	7	37.12	1	$N(5,(1)^2)$	6.54	8.58	10.69	6.81
7	6	7	37.12	1	$N(5,(1)^2)$	6.59	8.58	10.69	6.99
				Cost		103.77	101.59	101.47	482.63

Instance 3

							OUL		
Node	Predecessor	H	S	LT	Demand	DNNSMIO	CD	Enumeration	DFO
1	–	0.4	0	2	–	40.98	36.67	35.55	39.66
2	–	0.4	0	2	–	40.99	36.67	35.55	22.78
3	–	0.4	0	2	–	41.03	36.67	35.55	12.47
4	–	0.4	0	2	–	41.07	36.67	35.55	46.63
5	1	0.9	0	2	–	42.83	45.19	41.11	39.61
5	2	0.9	0	2	–	43.29	45.19	41.11	42.00
6	3	0.9	0	2	–	43.26	45.19	41.11	42.41
6	4	0.9	0	2	–	42.17	45.19	41.11	40.04
7	5	2.1	15	2	$N(20,(3)^2)$	46.19	47.85	52.22	47.02
7	6	2.1	15	2	$N(20,(3)^2)$	46.14	47.85	52.22	48.82
				Cost		163.15	161.30	161.13	441.43

Instance 4

							OUL		
Node	Predecessor	H	S	LT	Demand	DNNSMIO	CD	Enumeration	DFO
1	–	0.3	0	2	–	10.12	9.21	10.27	1.26
2	–	0.3	0	2	–	10.19	9.21	10.2	7.59
3	–	0.3	0	2	–	10.73	9.21	10.27	9.89
4	–	0.3	0	2	–	10.21	9.21	10.27	1.60
5	1	0.8	0	2	–	12.74	11.55	10.27	8.06
5	2	0.8	0	2	–	12.38	11.55	10.27	7.26
6	3	0.8	0	2	–	12.55	11.55	10.27	5.27
6	4	0.8	0	2	–	12.20	11.55	10.27	7.94
7	5	2	15	2	$N(5,(1)^2)$	12.25	12.62	13.05	15.52
7	6	2	15	2	$N(5,(1)^2)$	12.25	12.62	13.05	13.81
				Cost		37.49	35.97	35.98	139.77

Instance 5

							OUL		
Node	Predecessor	H	S	LT	Demand	DNNSMIO	CD	Enumeration	DFO
1	–	0.5	0	1	–	5.01	4.38	3.75	8.79
2	–	0.5	0	1	–	5.02	4.38	3.75	7.35
3	–	0.5	0	1	–	5.11	4.38	3.75	8.91
4	–	0.5	0	1	–	5.06	4.38	3.75	5.23
5	1	1	0	1	–	7.04	6.23	6.52	3.07
5	2	1	0	1	–	7.07	6.23	6.52	5.71
6	3	1	0	1	–	6.98	6.23	6.52	3.08
6	4	1	0	1	–	6.91	6.23	6.52	5.74
7	5	3	16	1	$N(5,(1)^2)$	6.33	6.49	6.52	5.75
7	6	3	16	1	$N(5,(1)^2)$	6.23	6.49	6.52	6.38
				Cost		29.04	27.53	27.45	36.03

Table 2: Assembly(2) SCN cases details

Instance 1

Node	Predecessor	H	S	LT	Demand	DNNSMIO	Suggested OUL CD	Enumeration	DFO
1	–	2	0	1	–	5.23	3.97	3.75	6.76
2	–	2	0	1	–	5.01	3.97	3.75	6.53
3	–	2	0	1	–	5.22	3.97	3.75	6.80
4	–	2	0	1	–	4.96	3.97	3.75	6.80
5	–	2	0	1	–	4.92	3.97	3.75	5.93
6	4	4	0	1	–	5.9	5.49	5.83	0.80
6	5	4	0	1	–	6.31	5.49	5.83	5.92
7	1	7	40	1	N(5,(1)^2)	5.89	7.77	7.91	6.64
7	2	7	40	1	N(5,(1)^2)	6.97	7.77	7.91	5.75
7	3	7	40	1	N(5,(1)^2)	5.82	7.77	7.91	6.10
7	6	7	40	1		6.33	7.77	7.91	6.27
					Cost	93.9432	90.4087	90.5432	116.38

Instance 2

Node	Predecessor	H	S	LT	Demand	DNNSMIO	Suggested OUL CD	Enumeration	DFO
1	–	0.3	0	1	–	9.93	8.98	7.5	10.89
2	–	0.3	0	1	–	9.89	8.98	7.5	10.49
3	–	0.3	0	1	–	9.98	8.98	7.5	11.01
4	–	0.3	0	1	–	9.86	8.98	7.5	11.24
5	–	0.3	0	1	–	9.87	8.98	7.5	10.79
6	4	0.5	0	1	–	10.99	10.53	10.27	5.59
6	5	0.5	0	1	–	11.39	10.53	10.27	10.47
7	1	0.9	3.5	1	N(10,(1)^2)	12.47	12.54	14.44	10.88
7	2	0.9	3.5	1	N(10,(1)^2)	12.31	12.54	14.44	11.25
7	3	0.9	3.5	1	N(10,(1)^2)	12.5	12.54	14.44	11.12
7	6	0.9	3.5	1	N(10,(1)^2)	11.85	12.54	14.44	11.28
					Cost	23.00	22.43	22.48	25.75

Instance 3

Node	Predecessor	H	S	LT	Demand	DNNSMIO	Suggested OUL CD	Enumeration	DFO
1	–	0.5	0	1	–	10.7	11.1868	10.27	11.57
2	–	0.5	0	1	–	10.43	11.1868	10.27	19.27
3	–	0.5	0	1	–	10.71	11.1868	10.27	11.59
4	–	0.5	0	1	–	10.27	11.1868	10.27	13.77
5	–	0.5	0	1	–	10.24	11.1868	10.27	13.63
6	4	3	0	1	–	11.06	11.1868	10.27	5.71
6	5	3	0	1	–	11.07	11.1868	10.27	11.56
7	1	6	25	1	N(10,(2)^2)	11.54	12.4788	13.05	11.81
7	2	6	25	1	N(10,(2)^2)	11.94	12.4788	13.05	11.78
7	3	6	25	1	N(10,(2)^2)	10.87	12.4788	13.05	11.77
7	6	6	25	1	N(10,(2)^2)	11.62	12.4788	13.05	11.90
					Cost	86.61	82.67	82.32	90.71

Instance 4

Node	Predecessor	H	S	LT		DNNSMIO	Suggested OUL CD	Enumeration	DFO
1	–	0.6	0	1	–	7.26	6.2706	5.25	7.47
2	–	0.6	0	1	–	7.16	6.2706	5.25	7.95
3	–	0.6	0	1	–	7.1	6.2706	5.25	6.10
4	–	0.6	0	1	–	6.93	6.2706	5.25	5.57
5	–	0.6	0	1	–	6.88	6.2706	5.25	7.52
6	4	1.1	0	1	–	7.72	7.53	7.19	10.76
6	5	1.1	0	1	–	8.02	7.53	7.19	5.42
7	1	2.5	5.4	1	N(7,(1)^2)	8.22	8.72	10.11	7.60
7	2	2.5	5.4	1	N(7,(1)^2)	7.87	8.72	10.11	7.87
7	3	2.5	5.4	1	N(7,(1)^2)	7.73	8.72	10.11	8.18
7	6	2.5	5.4	1	N(7,(1)^2)	7.43	8.72	10.11	9.30
					Cost	34.62	34.04	34.17	42.35

Instance 5

Node	Predecessor	H	S	LT	Demand	DNNSMIO	Suggested OUL CD	Enumeration	DFO
1	–	0.25	0	1		12.11	10.55	8.25	14.88
2	–	0.25	0	1	–	11.97	10.55	8.25	14.55
3	–	0.25	0	1	–	11.9	10.55	8.25	14.48
4	–	0.25	0	1	–	11.5	10.55	8.25	4.95
5	–	0.25	0	1	–	11.59	10.55	8.25	14.22
6	4	0.59	0	1	–	13.52	12.62	12.83	2.07
6	5	0.59	0	1	–	13.59	12.62	12.83	13.95
7	1	0.88	49.7	1	N(11,(2)^2)	14.86	18.44	20.47	15.89
7	2	0.88	49.7	1	N(11,(2)^2)	14.49	18.44	20.47	15.05
7	3	0.88	49.7	1	N(11,(2)^2)	14.51	18.44	20.47	14.55
7	6	0.88	49.7	1	N(11,(2)^2)	14.29	18.44	20.47	17.67
					Cost	30.98	28.19	27.96	62.76

Table 3: Cost comparison cases: structure details and suggested OULs

Instance 1

Node	Predecessor	H	S	R	LT	Demand	Suggested OUL DNNSMIO	Suggested OUL DFO
1	–	2	4	1.25	1	–	49.25	52.22203
2	1	4-3	12-4	1.5	1	–	17.75	18.48301
3	1	4-3	12-4	1.5	1	–	17.45	18.42489
4	1	4-3	12-4	1.5	1	–	17.36	18.40821
5	2	7-6	36-12	1.25	1	–	7.3	5.34762
5	3	7-6	36-12	1.25	1	–	6.7	5.29257
5	4	7-6	36-12	1.25	1	–	6.79	5.33311
6	2	7-6	36-12	1.25	1	–	7.55	5.32846
6	3	7-6	36-12	1.25	1	–	6.8	5.31856
6	4	7-6	36-12	1.25	1	–	6.99	5.30957
7	2	7-6	36-12	1.25	1	N(5,(1)^2)	7.92	5.30319
7	3	7-6	36-12	1.25	1	N(5,(1)^2)	6.64	5.30318
7	4	7-6	36-12	1.25	1	N(5,(1)^2)	7.21	5.28193
						cost	380.95	402.41

Instance 2

Node	Predecessor	H	S	R	LT	Demand	Suggested OUL DNNSMIO	Suggested OUL DFO
1	–	2	4	2	1	–	56.53	57.33026
2	1	4-3	12-4	2	1	–	18.64	20.50994
3	1	4-3	12-4	2	1	–	19.96	18.88558
4	1	4-3	12-4	2	1	–	19.07	19.12507
5	2	7-6	36-12	2	1	–	7.8	7.24254
5	3	7-6	36-12	2	1	–	6.91	4.22521
5	4	7-6	36-12	2	1	–	12.21	6.3352
6	2	7-6	36-12	2	1	–	8.36	6.94252
6	3	7-6	36-12	2	1	–	7.56	4.40232
6	4	7-6	36-12	2	1	–	11.83	7.46706
7	2	7-6	36-12	2	1	N(6,(1)^2)	8.19	7.31975
7	3	7-6	36-12	2	1	N(4,(1)^2)	5.97	4.39467
7	4	7-6	36-12	2	1	N(6,(2)^2)	11.8	7.27793
						cost	419.13	442.42

Instance 3

Node	Predecessor	H	S	R	LT	Demand	Suggested OUL DNNSMIO	Suggested OUL DFO
1	–	2	4	1	1	–	53.73	56.46892
2	1	4-3	12-4	2	1	–	18.87	19.30803
3	1	4-3	12-4	2	1	–	19.43	19.16313
4	1	4-3	12-4	2	1	–	18.75	19.61025
5	2	7-6	36-12	1	1	–	6.67	5.41518
5	3	7-6	36-12	1	1	–	7.28	5.61027
5	4	7-6	36-12	1	1	–	12.64	7.17493
6	2	7-6	36-12	1	1	–	7.49	5.40836
6	3	7-6	36-12	1	1	–	8.19	5.44619
6	4	7-6	36-12	1	1	–	10.67	6.90065
7	2	7-6	36-12	1	1	N(5,(1)^2)	7.55	5.90935
7	3	7-6	36-12	1	1	N(5,(1.2)^2)	8.59	4.75065
7	4	7-6	36-12	1	1	N(6,(1.5)^2)	9.25	7.14491
						cost	407.83	408.27

Instance 4

Node	Predecessor	H	S	R	LT	Demand	Suggested OUL DNNSMIO	Suggested OUL DFO
1	–	2	4	2.5	1	–	48.56	52.62784
2	1	4-3	12-4	2.5	1	–	17.56	17.88147
3	1	4-3	12-4	2.5	1	–	16.9	17.9517
4	1	4-3	12-4	2.5	1	–	17.62	16.90722
5	2	7-6	36-12	2.5	1	–	7.11	5.15377
5	3	7-6	36-12	2.5	1	–	7.85	7.34585
5	4	7-6	36-12	2.5	1	–	7.22	4.16108
6	2	7-6	36-12	2.5	1	–	7.19	5.48897
6	3	7-6	36-12	2.5	1	–	8.39	7.18886
6	4	7-6	36-12	2.5	1	–	6.39	4.39452
7	2	7-6	36-12	2.5	1	N(5,(1)^2)	7.64	5.83787
7	3	7-6	36-12	2.5	1	N(6,(1)^2)	8.75	6.75788
7	4	7-6	36-12	2.5	1	N(4,(1.5)^2)	6.25	4.72713
						cost	379.31	400.047