DEEP REINFORCEMENT LEARNING FOR INVENTORY OPTIMIZATION WITH NON-STATIONARY UNCERTAIN DEMAND

Henri Dehaybe, Daniele Catanzaro, Philippe Chevalier

REPRINT | 3270

UCLouvain

CORE | LIDAM
Louvain Institute of Data Analysis and Modeling in economics and statistics

Deep Reinforcement Learning for Inventory Optimization with Non-Stationary Uncertain Demand

Henri Dehaybe[a,*], Daniele Catanzaro[a] and Philippe Chevalier[a]

[a]Center for Operations Research and Econometrics (CORE), Université catholique de Louvain, Voie du Roman Pays 34, Louvain-la-Neuve, 1348, Belgium

ARTICLE INFO

Keywords:
Inventory
Lot Sizing
Forecast Evolution
Deep Reinforcement Learning
Non-Stationary Demand

ABSTRACT

We consider here a single-item lot sizing problem with fixed costs, lead time, and both backorders and lost sales, and we show that, after an appropriate training in randomly generated environments, *Deep Reinforcement Learning* (DRL) agents can interpolate in real-time near-optimal dynamic policies on instances with a rolling-horizon, provided a previously unseen demand forecast and without the need to periodically resolve the problem. Extensive computational experiments show that the policies provided by these agents compete, and in some circumstances even outperform by several percentage points of gap, those provided by heuristics based on dynamic programming. These results confirm the importance of DRL in the context of inventory control problems and support its use in solving practical instances featuring realistic assumptions.

1. Introduction

Recent trends in the literature on Operations Management remarked a growing interest of the scientific community towards *Deep Reinforcement Learning* (DRL) approaches to *Inventory Control* (IC) problems characterized by features and dynamics possibly too hard to be modeled and solved by means of classical dynamic or mathematical programming (Vanvuchelen et al., 2020; Oroojlooyjadid et al., 2021; Gijsbrechts et al., 2022; De Moor et al., 2022; van Hezewijk et al., 2022). The idea at the core of these approaches consists of representing the value function of a traditional *Reinforcement Learning* (RL) agent by means of a deep neural network. The replenishment policy is approximated by selecting the order quantity with the highest expected value. More advanced methods directly learn the replenishment policy function of a given IC system by means of a second deep neural network, whose weights encode the parameters of the policy itself. The network is then trained to autonomously learn the replenishment policy, by interacting with a black-box environment that models the system and other arbitrarily complex assumptions. The earliest works pioneering the use of DRL approaches to solution of IC problems considered classical systems characterized by a nontrivial optimal replenishment policy and a stationary demand distribution (Vanvuchelen et al., 2020; Oroojlooyjadid et al., 2021; Gijsbrechts et al., 2022; De Moor et al., 2022; van Hezewijk et al., 2022). This last assumption, however, rarely holds in practice. Here, we introduce a new methodology that aims to extend the use of DRL approaches to IC problems that feature a non-stationary demand and a rolling forecast horizon. This methodology is based on training the policy network on random infinite-horizon problems, provided a forecast for H future periods embedded in the state input. The methodology does not require to train the policy network on forecast data and allows to reuse the learned replenishment policy for many periods despite the non-stationarity of demand. We show here that this methodology can be readily applied to advanced forecast dynamics evolving according to the *Martingale Model of Forecast Evolution* (MMFE) first introduced by Iida and Zipkin (2006) (see Section 3.2). Extensive computational experiments show that in the case of the well-solved *Single-Item Stochastic Lot-Sizing Problem* (SISLSP) with backorders and in presence of a simple prior on the trend of the demand (simulated e.g., by means of an erratic uniform pattern without autocorrelation), the proposed methodology provides replenishment policies that, post-training, interpolates close-to-optimal order quantities given a previously unseen realistically trending forecast. These replenishment policies can be also a preferable alternative to a *Dynamic Programming* (DP) baseline policy even in the case of a highly variable MMFE. On harder lost sales problems, instead, the replenishment policies provided by the deep neural network significantly outperform the approximate DP baseline policy when lead times are sufficiently long.

The article is organized as follows. In Section 2, we review the literature on IC and on previous DRL approaches to solution of IC problems. In Section 3, we state the non-stationary version of the SISLSP; we describe an approximate baseline solution method based on DP and present a generalization of the problem based on the multiplicative MMFE. In Section 4, we present the methodology to solve the non-stationary SISLSP via a DRL approach. In Section 5, we describe the computational experiments obtained when testing this methodology on the backorders, lost sales and updating forecasts versions of the problem. Finally, in Section 6 we draw some preliminary conclusions and we provide perspectives on future research efforts.

Declarations of interest: none

*Corresponding author

henri.dehaybe@uclouvain.be (H. Dehaybe); daniele.cantazaro@uclouvain.be (D. Catanzaro); philippe.chevalier@uclouvain.be (P. Chevalier)

ORCID(s): 0000-0002-1276-9859 (H. Dehaybe); 0000-0001-9427-1562 (D. Catanzaro); 0000-0001-6443-624X (P. Chevalier)

2. Related work

Characterizing the analytic solutions to IC problems engaged for decades the community of Operations Management, especially under the assumption of fixed order costs (Porteus, 2002; Axsäter, 2015). One of the earliest, and historically most important, attempts was proposed by Scarf (1960), who first proved the optimality of the (s_t, S_t) policy of a periodic-review single-item lot sizing problem with uncertain non-stationary demand and fixed order cost. Three years later, Iglehart (1963) extended these results to the case of stationary infinite-horizon problems. Subsequent research efforts focused on generalizing these results, by relaxing some assumptions or by introducing other features. Examples include the introduction of unimodal inventory costs (Veinott, 1966) or concave order costs (Porteus, 1971). These results only apply under the assumption of a stationary demand, which unfortunately rarely holds in practice (Graves, 1999). Further notable attempts to characterize the solutions to inventory management problems concerned the lost sales version of the single-item problem, introduced by Karlin and Scarf (1958). In this context, the (s, S) policy is only proven to be optimal in the zero lead time case with stationary demand distributions (Veinott, 1966). Under more general assumptions, however, the optimal policy is unknown. This fact has justified the frequent use of heuristics and DP-based solution approaches in the literature, such as those described in Nahmias (1979) and Hill and Johansen (2006).

Several heuristic approaches have been proposed in the literature on IC to compute the policy parameters of the SISLSP under non-stationary uncertain demand. Askin (1981) proposed a myopic heuristic procedure to compute the (s_t, S_t) parameters, by selecting the number of periods to cover with the next order quantity. Bookbinder and Tan (1988) discussed the use of alternative policies to the optimal (s_t, S_t) to reduce both computational complexity and planning instabilities. The authors showed that suboptimal policies offer near-optimal performances under a rolling-horizon setting where the parameters are recomputed at every period. Bollapragada and Morton (1999) proposed to speedup the computation of the parameters of the (s_t, S_t) policy by recursively approximating the problem with a stationary demand. Rossi et al. (2015) proposed a mixed integer linear programming formulation for the SISLSP under the *static-dynamic uncertainty strategy* (i.e. the order periods are decided in advance) with both lost sales and backorders. Xiang et al. (2018) used a similar approach for the *dynamic-dynamic uncertainty strategy* (i.e. the order decision is taken for the current period only). Finally, Dural-Selcuk et al. (2020) studied the problem under a receding-horizon setting by using the static-dynamic and the *static uncertainty strategy* (i.e. the order quantities are also decided in advance). That is, the strategy parameters are recomputed at every period but the forecast window shortens to be 1 at the last period. The authors showed that even the static uncertainty strategy becomes competitive in this setting.

In this article we also consider the case of updating forecasts under an MMFE framework. This version of the IC problem was introduced in the literature by Graves et al. (1986), who first described a model that enables the update of the forecast in a multi-stage production process. The acronym MMFE was subsequently proposed by Heath and Jackson (1994) to refer to a general probabilistic model of the evolution of forecasts through time. The authors showed the relevance of MMFE as a way of improving forecasting techniques in a simulation study based on a firm's historical data. Dong and Lee (2003) showed that the base-stock policy remains optimal in the serial system of Clark and Scarf (1960) when the demand is modeled by a MMFE. Iida and Zipkin (2006) developed a computational approach to approximate optimal base-stock policies for a single-item problem. Finally, Norouzi and Uzsoy (2014) derived a better characterization of a MMFE demand process by modeling the evolution of the conditional covariance.

The difficulty posed by the introduction of additional dynamics to classical inventory models motivated several studies to consider them under the new DRL lens. Vanvuchelen et al. (2020) investigated the use of the ubiquitous *Proximal Policy Optimization* (PPO) DRL algorithm to address the joint replenishment problem with a two-dimensional action-space. The approximated policy is a parametric multinomial logistic distribution of the probability of each possible pair of quantities. Oroojlooyjadid et al. (2021) used the *Deep Q-Network* (DQN) algorithm to address the beer game problem, a serial multi-echelon problem commonly used to demonstrate the bullwhip effect in a supply chain. The DQN algorithm approximates only an action-value function, mapping state-action pairs to an expected return. The policy is to select the action with the highest value estimation in the current state by complete enumeration of the action-space. Gijsbrechts et al. (2022) proposed the use of the Asynchronous Advantage Actor-Critic algorithm to address a single-item lost sales model, a dual-sourcing model and a multi-echelon model. The authors proposed to approximate the stochastic policy with a neural network that takes the state of the inventory system as input and predicts the parameters of a categorical distribution over each possible order quantity as an output. De Moor et al. (2022) used the DQN algorithm and a transfer learning method called reward-shaping to stabilize the learning of perishable-goods inventory policies by incorporating knowledge from well-performing heuristics. Finally, van Hezewijk et al. (2022) addressed a stationary problem with capacity constraints and setup costs by adding feasibility masks to the action space. Boute et al. (2021) provided a survey of existing approaches in the literature and discussed other future research directions and challenges for the use of DRL in IC.

3. Problem statement

We consider here a rolling-horizon version of the SISLSP under a non-stationary uncertain demand, introduced by Scarf (1960). We study the problem when a finite-horizon forecast

is given (an hypothesis necessary to accommodate a non-stationary demand), but assuming at the same time that the process will continue beyond the horizon. This is usually addressed by repeatedly solving a finite problem at every time period (Axsäter, 2015). We start by stating the SISLSP and then we present a DP method that is used to approximate the solution to the SISLSP when considering backorders.

At period t, the forecast for the H upcoming periods is given by a sequence of demand distributions

$$F_{t:t+H-1} := \{D_k : k \in t, \dots, t+H-1\}$$

of the exponential family (Schäl, 1976), with respective probability density functions ϕ_k. This model assumes that

1. the demand distributions are *non-stationary* (i.e. they are not identical across periods);
2. a fixed order cost is incurred to place orders of non-zero quantities;
3. the lead time L between order and delivery is a fixed integral number of periods;
4. unsatisfied demand is back-ordered;
5. there is a per period discount factor $0 < \gamma < 1$.

The process starts with an initial inventory level z_1. At the beginning of each period, one must decide to raise the inventory position by ordering a quantity q_t. The inventory reaches a level $z_t + q_{t-L}$, available to satisfy the demand of the period that realizes according to its distribution, $d_t \sim D_t$. The inventory level thus reaches $z_{t+1} = z_t + q_{t-L} - d_t$. For now, we only consider the case of back-ordered unsatisfied demand, i.e., we assume that customers unsatisfied by the lack of available inventory are willing to wait for a replenishment (the inventory level z_t may be negative). Finally, a new forecast demand distribution D_{t+H} becomes available.

The classical version of this problem proposed by Scarf (1960) includes: a linear holding cost h per positive unit of inventory carried between two periods; a linear backorder cost b per negative unit of inventory carried between two periods (i.e. unsatisfied demand); and a fixed order cost K if an order occurred in this period. A linear order cost c per unit ordered can be considered but it is often set to 0, as observed in Pochet and Wolsey (2006).

Let $y_t := z_t + \sum_{k=1}^{L} q_{t-k}$ be the *inventory position* in period t before ordering and $w_t := y_t + q_t$ the inventory position right after ordering but before the demand realizes. Given an order q_t, the expected holding and stock-out costs at period $t+L$, when the order arrives, is the convex function

$$I_{t+L}(w_t) = \begin{cases} \displaystyle\int_0^{w_t} h(w_t - \xi)\phi_{t:t+L}(\xi)d\xi & \\ \qquad + \displaystyle\int_{w_t}^{\infty} b(\xi - w_t)\phi_{t:t+L}(\xi)d\xi & w_t \geq 0 \\ \displaystyle\int_0^{\infty} b(\xi - w_t)\phi_{t:t+L}(\xi)d\xi & w_t < 0 \end{cases} \tag{1}$$

where $\phi_{t:t+L}$ is the probability density function of the sum of the demands from periods t to $t+L$. The ordering cost is

$$P(q_t) = \begin{cases} K + c \cdot q_t & q_t > 0 \\ 0 & q_t = 0. \end{cases} \tag{2}$$

The *Single-Item Stochastic Lot-Sizing Problem* (SISLSP) with a starting inventory level y_t can, then, be formulated with the following functional equation of the expected value of the discounted cost

$$C_t(y_t) = \min_{q_t \geq 0} \left\{ P(q_t) + I_{t+L}\left(y_t + q_t\right) + \gamma E_{t+1}(y_t + q_t) \right\} \tag{3}$$

where

$$E_t(w_{t-1}) = \int_0^{\infty} C_t(w_{t-1} - \xi)\phi_t(\xi)d\xi \tag{4}$$

is the expected total cost for periods t to H if period $t-1$ reached the inventory position w_{t-1}. The boundary condition $C_{t+H-L}(\cdot) = 0$ is equivalent to assuming that the process stops beyond the horizon. When the forecast horizon H is long enough, this condition is unimportant as the most distant forecasts are insignificant in the computation of the policy. This, therefore, can be used as a good approximation of the infinite-horizon problem.

In the lost sales version of the SISLSP, negative inventory levels are brought to zero after that the *shortage* costs (instead of the *backorder* costs) have been paid. This change dramatically impacts on (1) by making its expression hard to compute in practical time.

3.1. An approximate dynamic programming algorithm for the SISLSP

In the case of backorders, Scarf (1960) proved that the optimal policy to the finite-horizon version of the SISLSP consists of checking whether the inventory position y_t is below a threshold s_t: if this should be the case, then order up to the inventory level $q_t = S_t - y_t$; otherwise, do not order. Because the demand is non-stationary, the (s_t, S_t) values differ for each period (Iglehart, 1963).

The sequence of optimal (s_t, S_t) parameters to problem (3) can be approximated by means of a DP algorithm that was first briefly described by Bollapragada and Morton (1999) as a benchmark to evaluate their heuristic. This algorithm, whose pseudocode is outlined in Algorithm 1, starts at period $H - L$, by computing the policy parameters (s_{H-L}, S_{H-L}); then, it proceeds backward in time. The algorithm relies on the K-convexity of the function

$$G_t(w) = c \cdot w + I_{t+L}(w) + E_{t+1}(w), \tag{5}$$

which is the expected cost function of reaching the inventory position w when ignoring the fixed ordering cost (Scarf, 1960). At each iteration t, a line search on (5) with a fixed step size δ starts from an upper bound to S_t to find a (sub)optimal inventory position S_t; then it continues

Algorithm 1: Approximate Policy based on DP

Input: The instance parameters: $L, b, K, h, F_{1:t+H-1}, \gamma, c$; a step size: δ; an upper bound to S: UB_S

```
for t ∈ [H − L, ..., 1] do
    descending ← true;
    w ← UB_S;
    g ← G_t(w);
    S_t ← w ;
    g_min ← g;
    while g_min ≤ g + K ∨ descending do
        w ← w − δ;
        g ← G_t(w);
        if g ≤ g_min then
            g_min ← g;
            S_t ← g;
        else
            descending ← false;
    s_t ← w
```

searching until finding s_t at the inventory position costing K more than that of S_t. The task of computing $C_{t+1}(w_{t+1})$ is simplified by the fact that the pair (s_{t+1}, S_{t+1}) is known from the previous iteration and the values of $E_{t+2}(w_{t+1})$ can be memoized from the previous iteration for all w_{t+1}. E_{t+1} can be computed with any numerical integration tool. We used a simple Riemann sum on an interval upper bounded by a large quantile of ϕ_{t+1}. The analytic expression of the solution of (1) can be obtained by means of a symbolic solver such as Wolfram Mathematica.

The introduction of the lost sales in the SISLSP breaks the K-convexity property at the core of Scarf's optimality proof of the (s_t, S_t) policy. The policy computed by assuming backorders, however, may constitute a good heuristic to approximate the optimal policy for the lost sales case (Axsäter, 2015). This is especially true when the h/b ratio is low, as stock-outs will be rare and negligible in presence of a fixed order cost. In their experiments, Hill and Johansen (2006) concluded that the (s_t, S_t) policy is near-optimal, and likely optimal in many cases. In the rolling-horizon framework, only the (s_1, S_1) pair is used and the problem must be resolved at every period by using the new forecast window. Due to the boundary condition, the discretization of the units, and the approximate integration of E_t, this method is approximating the optimal policy as the length of the window increases and as the step size decreases. In the case of a short forecast horizon, the finite-horizon boundary condition is no longer a good approximation. We circumvent this issue by augmenting the forecast with 32 stationary periods equal to the average demand.

3.2. The Martingale model of forecast evolution

The *Martingale Model of Forecast Evolution* (MMFE) was introduced by Heath and Jackson (1994) as a framework to model the dynamics of forecasting. We will exploit the MMFE in the subsequent sections to model the periodic revision of the demand that may occur in practice. For the sake of completeness, we briefly recall here the fundamental aspects of the MMFE and we introduce some basic notation and definitions that will prove useful in the remainder of the article.

Let $D_{k,t}$ denote the distribution that models the demand forecast for period k, given the information available at time $t \leq k$. The mean of $D_{k,t+1}$ (i.e., the revised forecast, hereafter denoted as $\mathbb{E}[D_{k,t+1}]$) is obtained by multiplying the mean of $D_{k,t}$ by an update factor u_k, i.e.,

$$\mathbb{E}[D_{k,t+1}] = \mathbb{E}[D_{k,t}]u_k, \forall k \geq t+1.$$

In the context of the MMFE, this updating policy is referred to as the *multiplicative update* (Iida and Zipkin, 2006). Another type of updating policy available in the MMFE is referred to as the *additive update* and consists of adding a *revision term* $\Delta\mu_k$ to $\mathbb{E}[D_{k,t}]$, i.e., $\mathbb{E}[D_{k,t+1}] = \mathbb{E}[D_{k,t}] + \Delta\mu_k$. Because the MMFE only considers updates of the expected demand, we assume that the standard deviation is also updated such that the *Coefficient of Variability* (CoV) (i.e., the ratio $\mathbb{E}[D_{k,t}]/\sigma_{k,t}$, where $\sigma_{k,t}$ represents the standard deviation of $D_{k,t}$) remains constant. In general, u_k is a random variable with positive support and unit mean and may be correlated to $u_{j\neq k,t}$. In this work we will assume that u_k is log-normally distributed, i.e.,

$$u_k \sim Logn(-\frac{\Sigma_k}{2}, \Sigma_k), \tag{6}$$

where the exponential decaying of the variance

$$\Sigma_k = \Sigma_1 \cdot \varkappa^{k-t-1} \tag{7}$$

is used to model the fact that less information is available to update distant forecasts. In (7), Σ_1 denotes the variance of u_{t+1} and $\varkappa \in (0, 1)$ is a scalar representing the decay factor. As in Iida and Zipkin (2006), we consider no correlation between the update factors belonging to different periods.

Henceforth, we refer to the version of the SISLSP that includes a periodic revision of the forecast according to the above framework as SISLSP-MMFE. No solution method is currently described in the literature to solve the SISLSP-MMFE under fixed ordering cost and a base-stock policy would perform poorly under these settings. To identify a baseline policy for the SISLSP-MMFE, we used the DP method presented in Section 3.1.

4. A new DRL solution approach to non-stationary versions of the SISLSP

In this section, we present a DRL approach to learn replenishment policies for the non-stationary versions of the SISLSP described in Section 3. Before proceeding, we recall that *Reinforcement Learning* (RL) methods require an interactive simulation of a Markov Decision Process, called *environment*, to learn on. In Section 4.1, we present a possible approach to construct an environment in the case

of a non-stationary demand and we provide a toy example in Section 4.2. In Section 4.3, we briefly recall the foundations of the *Proximal Policy Optimization* (PPO) algorithm, much used in the literature of DRL to learn neural network-based control policies. We rely upon this algorithm to carry out computational experiments on the non-stationary SISLSP. In Section 4.4, we discuss some issues introduced by the presence of a fixed order cost and propose a possible strategy to overcome them.

4.1. An RL encoding for the SISLSP

An environment is a Markov Decision Process that simulates a sequential decision problem to solve. An environment is associated with a set $\mathcal{X}$ of states that it may assume. An *agent* is an entity that is distinguished from the environment and that can learn from it by means of *interactions*, i.e., actions that cause the transition of the environment from the current state to a new one and the return of a reward. Specifically, let $\mathcal{A}$ denote the set of actions that an agent can undertake. Then, a transition of the environment occurs at time t when the agent observes the state $x_t \in \mathcal{X}$ and undertakes an action $a_t \in \mathcal{A}$ based on a policy $\pi : \mathcal{X} \to \mathcal{A}$. The environment then returns a scalar reward r_t drawn from a *reward function* $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ and moves to a new state x_{t+1}. The agent-environment interaction gives rise to a sequence of transitions (x_t, a_t, r_t, x_{t+1}) until a terminal state is reached, or the simulation is arbitrarily stopped in the case of an infinite Markov Decision Process. The sequence of states, actions and rewards from the initial to the last state is called an *episode*. The *transition dynamics* of the process describe how the environment will transition depending on its state x_t, the action a_t, and possibly some other stochastic component. The goal of an agent is to learn the policy that maximizes its expected discounted sum of rewards. Thus, in order to formulate an IC problem as a RL environment, we need to specify the sets of states and actions, the reward function, and the transition dynamics.

States

A particular state is an encoding that includes all the information about past agent-environment interactions (i.e. it has the *Markovian* property, Sutton and Barto (2018)). In this article, we explicitly embed the parameters of the forecast demand distributions in the state, in addition to the inventory position:

$$x_t = (M(D_t), M(D_{t+1}), \dots, M(D_{t+H-1}), z_t, q_{t-L}, \dots, q_{t-1})$$

where H is a forecast horizon and $M(D)$ returns all the necessary parameters to describe the distribution D (e.g. the mean and the standard deviation in the case of Gaussians).

Actions

The simplest encoding of the action is the direct embedding of the order quantity $a_t = (q_t)$. In this article, we instead consider the use of an alternative encoding $a_t = (s_t, S_t)$. The environment is programmed internally to reinterpret this pair as an order quantity q_t. As explained in Section 4.4, this is crucial to the stability of the DRL algorithm due to the discontinuous nature of the optimal policy of lot sizing problems with fixed costs.

Reward function

The environment returns the ordering and inventory costs paid at time t as the negative rewards r_t computed from the function $r(x_t, a_t)$ that depends on the state of the environment and the action undertaken by an interacting agent. Ordering costs are deterministic and given by Equation (2). The inventory costs equal $h \cdot \max(0, z_{t+1}) + b \cdot \left|\min(0, z_{t+1})\right|$ where $z_{t+1} = z_t + q_{t-L} - d_t$ where d_t is the realized demand at time t.

Although Scarf's results can be generalized to other cost functions, as shown in (Porteus, 1971), here we only consider the case of linear ordering, holding, and stock-out costs, mostly because the baseline DP policy parameters are easier to compute. Any different assumption would just involve changing accordingly the reward signal of the environment.

Transition dynamics

The dynamics of the of environment are characterized by a *state-transition function* that generates one next-state given an action. In our case, given the action (q_t), the next state is simply

$$x_{t+1} = (M(D_{t+1}), \dots, M(D_{t+H}), z_{t+1}, q_{t-L+1}, \dots, q_t).$$

That is, the forecast window is shifted by one period, the inventory level is updated according to the realized demand and the quantity ordered L periods ago, and the on-order queue is appended with the latest order (q_t) quantity. An implementation of an environment with lost sales needs only to set $z_{t+1} = 0$ if it is negative at the end of period t. In the case of a SISLSP-MMFE, the parameters of the forecast are multiplied by a random vector at the end of each period, as we explain in Section 3.2. Since there is no terminal state in infinite problems, interactions may optionally be interrupted to form episodes after a fixed number of periods.

Initial state distribution

RL algorithms typically learn from transition data generated by several episodes of interactions with an agent. When starting a new episode, the environment may be reset to a fixed or to a random initial state. Formally, the state variables may be partitioned into subsets characterized by their independent distributions (univariate for single-element subsets, multivariate otherwise). For example, the parameters of each forecast distribution may be independently drawn for each period or follow an autocorrelated process. For the sake of simplicity, we consider here just univariate initial distributions. This is without loss of generality as RL methods do not require this assumption. We denote $\bar{Z}_1$ as the distribution of the initial inventory level (z_1); $\bar{D}_t^n$ for the n^{th} parameter of the forecast distribution at period t; and $\bar{Q}_k$ for the on-order quantity ordered at time $1 - k$ (q_{1-k}). The family of each distribution need not to be the same for each state variable and may be discrete or continuous. Notably, a subset of

state variables may be initialized deterministically to a fixed number.

Randomizing the initial state of the forecast is a core feature of the proposed approach for two reasons.

First, the randomization of the initial state is an instance of an *exploring start* that allows the agent to explore the state space more efficiently and thus better generalize (Sutton and Barto, 2018). Some states may otherwise be unlikely to be reached with the current policy and consequently prevent the agent from learning their value.

Second, this ensures that the agent does solve the rolling-horizon problem with a forecast window of H periods and no more. If the forecasts beyond the horizon are predictable, because always identical from episode to episode, then the agent will learn it and its policy will solve a longer-horizon problem. Randomizing the initial state of the forecast parameters acts as an uninformative prior that provides little information to extrapolate beyond the horizon.

A key implication of this latter fact is that, once trained, the agent can be used to compute the order decision indefinitely, as long as the training environment remains relevant to the actual inventory system. To the best of our knowledge, this is the first method that achieves this in an infinitely-rolling-horizon setting with non-stationary demand. Consequently, as a by-product, the policy learned by the agent does not learn to solve a single instance of the original problem, but rather all instances that lie on the support of the initial state distribution.

4.2. A toy example

We show the construction and a transition of a rolling-horizon RL environment following the steps outlined in Sections 4.1. Consider an instance of the SISLSP problem arising when setting $H = 4$, $L = 1$, $K = 100$, $h = 1$, $b = 10$. The demand at period t is forecasted with a Poisson distribution $Poi(\lambda_t)$ with parameter λ_t. The state of the environment at time t will thus be

$$x_t = \left(\lambda_t, \lambda_{t+1}, \lambda_{t+2}, \lambda_{t+3}, z_t, q_{t-1}\right)$$

The initial state distribution used to generate λ_t, $\forall t \geq 1$ is a single binomial distribution to randomly generate the Poisson parameter at each period. We choose the following initial state distributions:

- $\lambda_t \sim \bar{D}^{\lambda} := B(20, 0.5), \forall t$,
- $z_1 \sim \bar{Z}_1 := U(-20, 20)$,
- $q_0 = 5$,

where U denotes a uniform distribution, B a binomial distribution, and q_0 is a deterministic initial state variable. Following the initial distribution outlined above, we generate the initial state

$$x_1 = (7, 13, 10, 15, -2, 5)\,.$$

The policy of the agent returns the action $a_1 = (19)$, giving rise to the transition that follows. The 5 on-order quantities become available and the realized demand $d_1 = 6$ is randomly drawn from the Poisson distribution $Poi(7)$, thus the new inventory level becomes $z_2 = -2 + 5 - 6 = -3$. Finally, a new forecast parameter for the demand in period 5 becomes available and generated from the initial distribution: $\lambda_5 \sim B(20, 0.5) = 5$. As a result, the new state is

$$x_2 = (13, 10, 15, 5, -3, 19)\,,$$

and the reward is

$$\begin{aligned} r_1 := r(x_1, a_1) = &- (100 + 10 \cdot |\min(0, -3)| \\ &+ 1 \cdot \max(0, -3)) = -130. \end{aligned}$$

This completes a single transition.

Should the environment be that of a lost sales model, the transition to the next state finishes by setting $z_2 = \max(z_2, 0)$. In the case of forecast updates according to a MMFE and given Σ_k for all $k \in 2, \ldots, 5$, the transition requires to randomly generate update factors u_k for $k \in 2, \ldots, 5$ according to Equation (6). The revised forecast is obtained by setting $\lambda_k = \lambda_k \cdot u_{k+t}, \forall k \in 2, \ldots, 5$.

4.3. The proximal policy optimization algorithm

The *Proximal Policy Optimization* (PPO) algorithm is a well-known DRL algorithm introduced by Schulman et al. (2017) to train neural networks. The PPO algorithm belongs to the family of *actor-critic* algorithms as it approximates two functions by using neural networks. The first network, called the *critic*, approximates the state value function that estimates expected discounted sum of future rewards of a state for a given policy π

$$V_{\pi}(x_t) = \mathbb{E}\left[\sum_{k=t}^{\infty} \gamma^{k-t} r\left(x_k, \pi\left(x_k\right)\right)\right] \tag{8}$$

where $0 < \gamma < 1$ is a discount factor and the expectation is with respect to the transition probabilities and the stochastic policy distribution.

The second neural network, called the *actor*, approximates the policy $\pi(x_t)$. The PPO algorithm learns a *stochastic policy*, in contrast to a deterministic policy. In particular, the actor does not directly predict the action, but the parameters of a distribution from which the action is sampled. This allows a local exploration of the actions during training. At test time, only the mode of the learned policy is used to approximate the optimal policy. The goal of this network (agent) is to find the policy π that maximizes $R(\pi) = \mathbb{E}\left[V_{\pi}(x_1)\right]$ with respect to the initial state distribution.

The policy distribution may be discrete or continuous. We differ from existing IC literature in that the actor policy is a continuous multivariate Gaussian distribution with diagonal covariance, whereas Vanvuchelen et al. (2020) use a discrete multinomial distribution where each class is a possible action from a discretized set of actions. We use this approach for two reasons. First, it is more data efficient because a continuous distribution naturally incorporates the fact that two close order quantities lead to similar results.

In contrast, an actor with a multinomial distribution must try each action several times to learn their values separately. Second, it is more scalable in the dimensions of $\mathcal{A}$. With a multinomial policy distribution, the number of parameters to estimate grows exponentially and the approach becomes untractable.

We propose here that the actor approximates the parameters of a Gaussian distribution to sample a (s_t, S_t) pair. In other words, the output of the neural network is

$$\pi(x_t) = \left(\mu_{s_t}, \mu_{S_t}, \sigma_{s_t}, \sigma_{S_t}\right)$$

and during training

$$a_t = \left(s_t \sim N(\mu_{s_t}, \sigma_{s_t}), S_t \sim N(\mu_{S_t}, \sigma_{S_t})\right).$$

At test time, the action is $a_t = (\mu_{s_t}, \mu_{S_t})$. Actions that are not feasible are projected to the nearest feasible solution (e.g. negative order quantities become a no-order action). For more details on how the PPO algorithm trains its neural networks, we refer the reader to Schulman et al. (2017).

Continuous DRL is typically less stable and harder to train, Engstrom et al. (2020) explain how one must carefully pay attention to implementation details. Additionally it does not interact well with fixed order costs due to the discontinuous nature of the optimal policy function. We discuss this aspect in Section 4.4. As in Schulman et al. (2017), we use the Generalized Advantage Estimation (GAE) method of Schulman et al. (2016) to estimate the advantage of the actions sampled during training. This method uses an exponentially weighted average of the learned value of all the state encountered after an action to reduce the bias induced by the estimation error of the critic neural network.

4.4. Learning under fixed order costs

The presence of a fixed order cost makes much harder the task of learning the correct value estimates by using one-step bootstrapping. Let (x_t, a_t, r_t, x_{t+1}) be a transition where a non-zero quantity q_t was ordered. The incurred reward r_t is very low due to the large fixed cost whereas the benefit of an increased inventory will only be visible L periods in the future. The short-term advantages (computed with the GAE) are negative if the values of x_{t+1} and the subsequent states have not been well learned yet, as it is the case at the beginning of the training phase. This fact discourages the policy of ordering. Which in turn reduces the chances of encountering high inventory states and thus increases the time before the critic effectively learns that ordering is beneficial in the long run. This effect is amplified when the fixed order cost or the lead time increases. Furthermore, the exploration of the action-space via local sampling around a mean action discourages to ever move from ordering 0 to a large batch quantity. Around a mean of, say, $q_t = 0$, the most likely positive order quantities are small. These small batches will reinforce the agent towards never ordering because they incur the fixed cost without significantly improving the inventory position. It is therefore crucial that the critic rapidly learns the value of high inventory position states. For this reason, we train the critic for more epochs per iteration than the actor and use a warmup time to prevent the actor from diverging due to incorrect value estimation at the beginning of training, as detailed in Section 5.1. Finally, as known from the work of Scarf (1960), the optimal q_t policy with respect to the inventory position is discontinuous since q_t is a function of the (s_t, S_t) parameters and the inventory position. A neural network, on the other hand, is a continuous function approximator. Approximating discontinuous functions typically leads to extremely large gradients and is source of high instability in deep learning. A possible way to cope with this difficulty consists of parameterizing the actor network to output a distribution for (s_t, S_t) instead of (q_t). This parameterization has the benefit of having much smoother gradients. It is worth noting here that this reparametrization does not reduce the space of policies that the agent can approximate. In the literature on IC, the (s_t, S_t) policy parameters are computed without the knowledge of the inventory position w_t. This is a fundamental difference with this DRL approach where the output (s_t, S_t) distribution of the neural network is still conditioned on the full state vector, including on the inventory position, and thus the actor can output parameters that vary with respect to w_t. As such, it is still a universal policy approximator and it does not limit the method to the (s_t, S_t) policy in its classical IC meaning; it is merely a smoothed way to indirectly express a desired order quantity.

5. Computational experiments

In this section, we empirically evaluate the ability of the DRL approach to learn a policy for the versions of the non-stationary SISLSP described in Section 3. We consider, in particular, three series of experiments. The first series aims to measure the performance of the approach on problem instances where the DP baseline is approximately optimal. We show that a single agent trained on an environment with an erratic demand prior is able to interpolate close-to-optimal policies for several realistic trends that are not encountered during the training phase. We also show that the performance of the DRL approach is not impacted by a reduction of the forecast horizon. In the second series, we experiment on the case where the demand forecast is periodically revised according to a MMFE. Finally, we show in the last series that the approach can outperform the heuristic DP baseline for some instances of the lost sales version of the problem.

We start by presenting the details of our implementation of the PPO algorithm in Section 5.1. We then carry out the experiments on near-optimally solved instances in Section 5.2, the experiments on the SISLSP-MMFE instances in Section 5.3, and the experiments on lost sales instances in Section 5.4. Finally, we provide visual insights on the learned policies in Section 5.5.

5.1. Implementation details

Hyperparameters	Values
Critic learning rate	10^{-4}
Actor learning rate warmup bounds	$10^{-6} \rightarrow 10^{-4}$
Actor warmup factor	$\sqrt[2000]{100}$
Actor learning rate decay bounds	$10^{-4} \rightarrow 10^{-5}$
Actor learning rate decay	$\sqrt[13000]{0.1}$
Maximum gradient norm	0.5
Discount factor (γ)	0.99
GAE weight (λ)	0.9
Clip decay bounds (ϵ)	$0.2 \rightarrow 0.05$
Entropy weight (β)	10^{-2}
Episodes per iteration	30
Critic epochs	5
Actor epochs	1
Batch size	252
Training iterations	10000

Table 1
Values of the hyperparameters of the PPO algorithm.

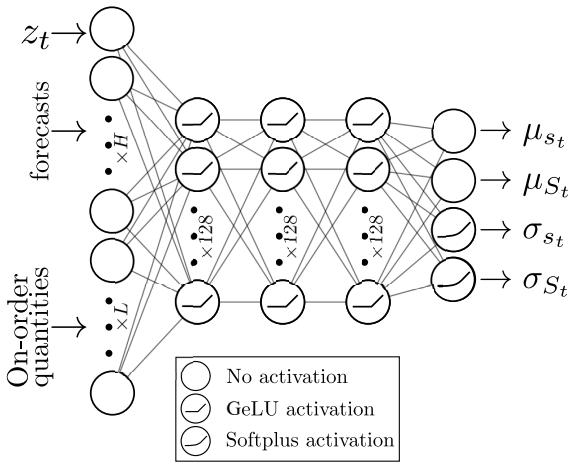

Figure 1: Architecture of the actor neural network.

The PPO algorithm is known to be fairly insensitive to the choice of hyperparameters. We used the recommended hyperparameters in most cases and we lowered the number of simulations and neural network update epochs as much as it was possible without losing performances (Schulman et al., 2017; Engstrom et al., 2020; Cobbe et al., 2021). They are detailed in Table 1. The only hyperparameter that required tuning was the gradient step size (learning rate) used to update the neural networks. We observed that different learning rates for the actor were needed depending on the problem parameters. To address all instances with the same hyperparameters, we used an exponentialy decaying learning schedule to vary its value during training. To ensure stability and better convergence of the actor, we also used an exponential warmup during the first 2000 training iterations. This warmup time is necessary to let the critic learn before letting the actor make too large updates, this addition proved to be critical for the stability of the method. The critic has a constant learning rate. We also linearly decayed the ϵ clipping parameter (see Schulman et al. (2017)) to help convergence at the end of training. We used the state-of-the-art stochastic gradient descent optimizer to train the neural networks, Adam (Kingma and Ba, 2015).

We implemented both neural networks as Multi-Layer Perceptrons (MLP) composed of three hidden layers with 128 neurons having the GeLU activation function (Hendrycks and Gimpel, 2016) and randomly initialized by following the orthogonal scheme (Saxe et al., 2014). The rationale underlying the use of an MLP rather than more recent sequential architectures such as an LTSM or a Transformer is that the performance cost is not justified in our situation where the state is Markovian. Such architectures are mainly used in DRL when dealing with partially observable states. The critic network is characterized by a linear output activation. The output neurons of the actor network that estimate the mean of the policy parameters have linear activations and those that estimate the standard deviation have a standard *softplus* activation ($sp(x) := log(1 + e^x)$). The complete architecture of the actor network is shown in Figure 1. The critic network is identical except for the output layer that has a single neuron with no activation.

At each iteration, episodes of 52 periods are simulated by using the latest policy to generate transitions. Then, both the actor and the critic networks are updated. The actornetwork is updated for a single epoch per iteration while additional epochs are performed for the critic network, as suggested by (Cobbe et al., 2021) and discussed in Section 4.4. The advantages are estimated by using GAE (Schulman et al., 2016) with $T = 52$, the full episode length. We scaled the rewards to zero mean and $1/T$ standard deviation with a running estimate of their moments to normalize the value network (Andrychowicz et al., 2021). The gradient is clipped to have a maximum norm of 0.5. The code is written in Julia 1.8.1 (Bezanson et al., 2017) (LLVM 13.0.1) and includes the deep learning package Flux 0.13 and its internal auto-differentiation package, Zygote (Innes, 2018). The three experiments ran on different cluster nodes, and thus with varying hardware configurations that are detailed in Table 2. Environment simulations were performed in parallel on the CPU cores while gradient computations were carried out on a GPU. We trained multiple agents in parallel by allocating one GPU to each task. The training time is highly hardware dependent: experiments that ran with more CPU cores per agent (number of GPUs divided by the number of cores) took up to 2.5 less time to complete. The implementation, the experimentation datasets, the experimental data, and the code of the fully reproducible experiments are available at `https://github.com/HenriDeh/DRL_MMULS/tree/single-item`

5.2. Near-optimality

In this section, we describe the first series of experiments that we carry out on SISLSP instances that are approximately optimally solved by the DP baseline presented in Section 3.1. We first present the testing and the training setups and then we detail the results of the experiments.

Experiment	GPU	CPU	DDR4 RAM	Time/agent
Near-optimality	4 NVIDIA RTX 3090	AMD EPYC 7352 24 cores (2.3 GHz)	515Gb	53
Updating Forecasts	2 NVIDIA A10	Intel Xeon Gold 6346 12 cores (3.1 GHz)	98 Gb	20
Lost Sales	2 NVIDIA RTX 3090	AMD EPYC 7352 12 cores (2.3 GHz)	98 Gb	25

Table 2
Hardware configurations of the different experiments and approximative time (expressed in minutes) to train one agent. Each GPU was dedicated to train one agent at a time.

Parameters	Experiment values	
Lost sales	false	true
h	1	1
b	$10, \mathbf{25}, 50$	$50, \mathbf{75}, 100$
CoV	$0.1, \mathbf{0.2}, 0.3$	$0.1, \mathbf{0.2}, 0.3$
K	$0, 80, \mathbf{360}, 1280$	$0, 80, \mathbf{360}, 1280$
c	0	0
L	$0, 1, \mathbf{2}, 4, 8$	$0, 1, \mathbf{2}, 4, 8$
H	$4, 8, 16, \mathbf{32}$	$4, 8, 16, \mathbf{32}$

Table 3
Parameters of the learning environments. The default values are emphasized in bold.

Trend	μ_t
Constant μ, $\mu \in \{5, 10, 15\}$	μ
Linear decline	$15 - t \cdot \frac{10}{136}$
Linear growth	$5 + t \cdot \frac{10}{136}$
Seasonal 1	$\left[1 + 0.5 \cdot \sin\left(1 \cdot 2\pi \frac{t}{52}\right)\right] 10$
Seasonal 2	$\left[1 + 0.5 \cdot \sin\left(2 \cdot 2\pi \frac{t}{52}\right)\right] 10$
Seasonal 4	$\left[1 + 0.5 \cdot \sin\left(4 \cdot 2\pi \frac{t}{52}\right)\right] 10$

Table 4
Time series used to generate the test forecasts for 8 different trends.

Training setup

To simplify the experimental setup, we considered the case where all demand distributions are Gaussians with identical Coefficients of Variation (CoV) across periods. This allowed to only embed the means of the Gaussians into the state, as the standard deviation can be inferred. We trained the agents on SISLSP environments with different *environment parameters* to evaluate the sensitivity of the method to their values. We changed the backorder/shortage cost (b), the CoV, the fixed order cost (K), the lead time (L), the forecast horizon (H), and whether sales are lost. The K values were selected to have respective deterministic EOQ of 0, 4, 8 and 16 periods. In the case of lost sales, the shortage cost (b) values were significantly higher than backorder values to ensure the DP baseline is near-optimal, and to avoid ill-posed problem instances where the optimal policy is to never-order. The variable production cost was kept to 0 for simplicity. We allowed one environment parameter to change at a time while the others were kept to default values. Ten agents were trained for each of the 30 combinations described in Table 3 (15 of both backorders and lost sales configurations). The initial distributions of the inventory and of the expected demands of the training environments were $z_1 \sim \bar{Z}_1 = U(-\mu \cdot (L+1), 2\mu \cdot (L+1))$ and $\mu_t \sim \bar{D}^{\mu} = U(0, 20)$, respectively. The on-order quantities were always initialized to 0.

Testing setup

In order to test the ability of the agents to predict near-optimal policy parameters given non-stationary demand forecasts they never encountered during the training phase, we generated a test dataset of 136-periods-long forecasts with five types of realistic trends, inspired from those used by Rossi et al. (2015): three constant trend, three seasonal trends with different frequencies, a linear growth trend and a linear decline trend. These trends are detailed in Table 4. The constant trends test the ability to perform when the average expected demand is not that of the training environment ($\mu = 10$); the linear decline and growth trends model a threefold decrease or increase in demand over the 136 weeks; and the seasonal trends represent sinusoidal demands with one, two and four periods over the two years, respectively. These forecasts were used in combination with the varying training environment parameters (Table 3) to generate unique test problem instances.

The policies learned by the agents are evaluated by comparing to two baselines. The first is the (s_t, S_t) policies obtained by using the DP method described in Section 3.1, with a step size δ of 1 and always assuming backorders. This baseline is approximately optimal when shortages are backordered and heuristic in the case of lost sales. The baseline policy parameters are computed in a rolling-horizon fashion by executing Algorithm 1 at each period to recover the current period (s_t, S_t) parameters. The second baseline, which we refer to as the *Simple* baseline, computes (s_t, S_t) policy parameters for every period as follows. The threshold s_t is the $b/(b+h)$ quantile of the distribution of the total demand during the lead time. The reorder level S_t is $s_t + EOQ$ where the EOQ is computed with the usual formula

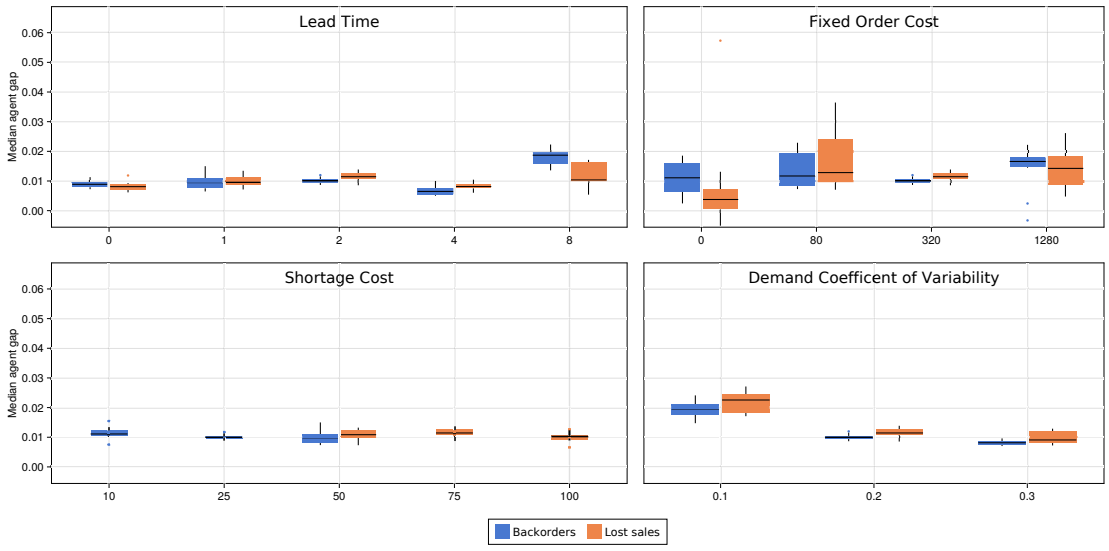

Figure 2: Whisker plots of the median agent gaps with respect to the DP baseline for the near-optimality experiments. Each data point represents one agent, each whisker plot thus represents ten agents.

$EOQ = \sqrt{2 \cdot D \cdot K/h}$, where D is the average expected demand of the next H periods.

The average return of a policy (baseline or learned) on an instance is estimated with 1000 Monte Carlo simulations of the first 104 periods with an initial inventory of $z_1 = L \cdot \mu$ and no on-order quantities. The performance of a policy for a single instance is its average return over the 1000 simulations. The *gap* of an agent for one instance is the difference between its performance and the baseline performance, divided by the baseline performance.

A trained DRL agent is evaluated on the 8 test instances that combine the test forecasts and its training environment parameters. We useed the median of the gaps of the individual instances as a measure of the overall performance of an agent. With 8 forecasts and a total of 15 different environment parametrizations (1 default configuration, 2 b variants, 2 CoV variants, 3 K variants, and 4 L variants, and 3 H variants), for both backorders and lost sales, we effectively tested the DRL approach on 120 unique backorder and 120 unique lost sales instances, each never encountered by the agents during the training phase.

Results

Figure 2 shows the distribution of the median gap of the agents with respect to the DP baseline over the 8 forecasts. Agents where $H \neq 32$ are not included in this figure. The results showed that the DRL approach performs consistently over repeated trials and thus the boxes are narrow in most environment settings. We only observed agents that are less performing on average in the case of a low setup cost. Specifically, 2 agents out of 240 reach an average gap greater than 4%, one when $K = 0$ and one when $K = 80$.

Table 5 reports the average, best and worst median gaps achieved with respect to the baseline methods for each environment parametrization. Overall, the average median gap achieved by the agents with respect to the DP baseline never exceeds 2.2% on each environment. All the agents converged to a policy competitive with the DP baseline and the simple heuristic is greatly outperformed in all settings excepted when $K = 0$, where the simple baseline even outperforms the DP baseline. This result shows that the proposed DRL approach approximates near-optimal policies for a wide range of forecasts trends, unknown to the agent. Figure 3 shows how the deep RL approach performs with reduced forecast horizons, ranging from 4 to 32 periods. This has no significant impact on the relative performance compared to the DP heuristic, which, as a reminder, solves for an instance with a forecast augmented with 32 stationary periods equal to the average in-horizon expected demand. This result shows that the choice of a uniform initial state distribution as a prior for the demand beyond the horizon is similar to the use of a stationary boundary condition for the DP method. In the opposite case, we would have observed an increase or a decrease in performance as the horizon shortens.

Figure 4 shows that the agents appropriately learned most forecast trends. The obvious exception is the *constant 5* trend (see Table 4), where most agents approximated a suboptimal policy. Surprisingly, we cannot observe the same effect for the *constant 15* trend. A possible explanation for this difference is that the relative difference between 5 and 10 is greater than between 15 and 10. This suggests that the mean demand during training should at least be close to that of the average real demand (10 here). A second exception is that some agents did not generalize well to the *seasonal 4* trend. The uppermost outliers of this box are all from agents training in environments with a lead time of 8 periods. This indicates that it is due to a difficulty in generalizing to different forecasts in this type of environment, rather than a

				DP			Simple
L	b	K	CoV	Average	Best	Worst	Average
Backorders							
2	25	0	0.2	1.11	0.24	1.85	1.44
2	25	80	0.2	1.35	0.73	2.29	-8.88
2	25	320	0.1	1.95	1.47	2.41	-4.69
2	10	320	0.2	1.15	0.74	1.56	-8.67
0	25	320	0.2	0.9	0.72	1.13	-5.38
1	25	320	0.2	0.96	0.65	1.5	-6.14
2	25	320	0.2	1.02	0.87	1.2	-6.63
4	25	320	0.2	0.68	0.5	1.0	-7.36
8	25	320	0.2	1.8	1.36	2.23	-6.1
2	50	320	0.2	0.99	0.74	1.5	-5.65
2	25	320	0.3	0.82	0.7	0.96	-8.0
2	25	1280	0.2	1.39	-0.33	2.22	-4.3
Lost Sales							
2	75	0	0.2	0.88	-0.51	5.73	1.29
2	75	80	0.2	1.68	0.71	3.65	-7.04
2	75	320	0.1	2.2	1.71	2.71	-3.33
2	50	320	0.2	1.06	0.73	1.32	-5.59
0	75	320	0.2	0.83	0.62	1.19	-4.1
1	75	320	0.2	1.0	0.71	1.34	-4.64
2	75	320	0.2	1.14	0.86	1.39	-5.04
4	75	320	0.2	0.81	0.6	1.04	-5.85
8	75	320	0.2	1.21	0.54	1.71	-5.26
2	100	320	0.2	1.01	0.65	1.29	-4.88
2	75	320	0.3	0.98	0.72	1.29	-6.41
2	75	1280	0.2	1.46	0.48	2.61	-3.1

Table 5
Summary of the near-optimality experiments (Section 5.2). Each row summarizes the ten agents that were trained with the environment parameters in the four leftmost columns. The four rightmost columns give the average, best, or worst of the ten agents' median gaps (in %) with respect to either the DP method or the Simple heuristic (see the headers).

difficulty posed by the forecast trend alone, since we did not see this difference with shorter lead times.

5.3. Updating forecasts

We performed experiments with a similar setup to those described in Section 5.2 with the addition of forecast updates that follow an MMFE. The training and testing setups as well as the implementation details are almost identical to the first experiment. The only change is that the parameters in Table 3 were fixed to their default values. We instead varied the parameters $\Sigma_1 \in \{0.03, 0.05, 0.1\}$ and $\varkappa \in \{0.7, 0.8, 0.9, 0.95, 0.99\}$, this time in a full factorial fashion. The results are plotted in Figure 5 for both backorders and lost sales. Table 6 contains the detailed performance results. The DP method remains a well-performing choice in the case of backorders. However, the DRL approach becomes more interesting in the case of lost sales as the variability increases. This is likely due to poorer performances of the DP heuristic as the likelihood of shortages increases with the variability induced by the forecast revisions.

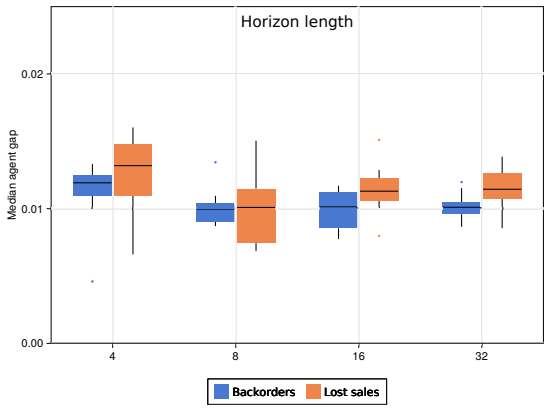

Figure 3: Whisker plots of the median agent gaps with respect to the DP baseline for the near-optimality experiments with varying horizons. Each data point represents one agent, each whisker plot thus represents ten agents.

5.4. Lost sales

The high shortage costs in the Lost Sales column of Table 3 lead to DP policies with shortage rates of less than 1%. In this section, we specifically focus on lost sales environments with parameters that allow for a higher shortage rate at optimality. Lowering the shortage cost is, however, not sufficient as this quickly leads to instances where the optimal policy is to not participate (i.e. never ordering), due to the high order cost. We avoided this issue by using a Poisson demand distribution instead of a Gaussian. The higher uncertainty of the demand induced by the long tail of the Poisson distribution creates more frequent shortages and thus penalizes more a non-participation. In these experiments we varied the parameters in a full factorial fashion as follows: $L \in \{2, 4, 8\}, b \in \{5, 10\}, K \in \{10, 20, 30\}$. In these settings, the shortage rates of the corresponding DP policies ranged from 2% to 9%.

The results are plotted in Figure 6 and summarized in Table 7. The results shows that the DRL approach provides competitive policies as the lead time increases, albeit with a higher variance as can be seen with the height of the boxes. Longer lead times are harder to learn in general, due to the delay between an action and the resulting reward, and thus the agents do not all converge to similarly performing policies. Nevertheless, the deep RL approach is able to outperform both the DP and the Simple heuristic in some settings, by a large gap in the best cases.

5.5. Visualization of the learned policies

Figure 7a compares the DP policy and the output of a learned DRL policy with respect to the inventory position (y). For the visualization, we selected the best performing agent (1.00% median gap) for the instance with backorders, $K = 360, b = 10, CoV = 0.2, L = 4$, and the constant 10 forecast. The on-order quantities in the state are set to zero, thus the inventory position consists solely of the inventory level. The red lines refer to the DP method and the blue lines to the agent. The solid lines are the order

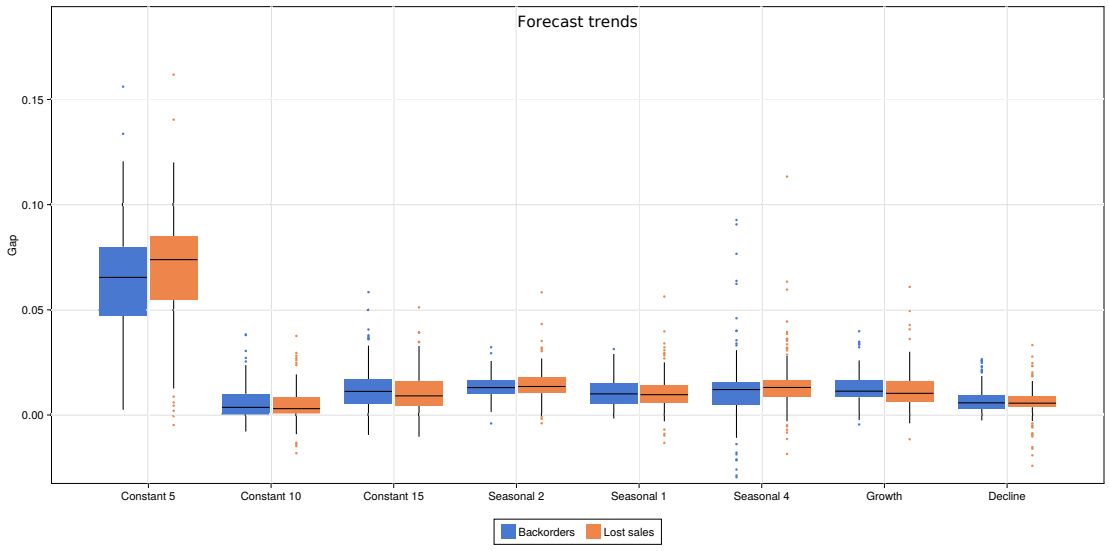

Figure 4: Whisker plots of the gap of the agents for each forecast with respect to the DP baseline. Each data point represents one test instance, each whisker plot thus represents the performance one the respective forecast trends of each of the 120 agents (backorders or lost sales).

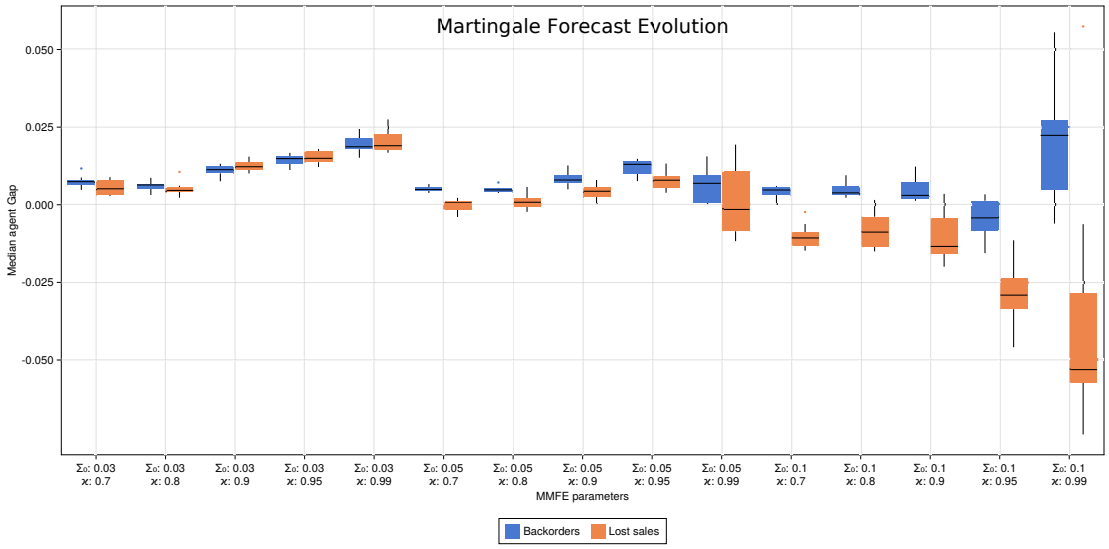

Figure 5: Whisker plots of the median agent gaps for the Updating Forecasts experiments with varying MMFE parameters. The gaps are computed with respect to the average cost of the dynamic programming (s_t, S_t) policy of each instance, recomputed at each step to account for the updated forecast. Each data point represents one agent, each whisker plot thus represents ten agents.

quantities prescribed by the respective policies (given their (s, S) output), the dashed and dash-dotted lines are the s and S parameters respectively (both constant for the DP method).

Unlike the DP baseline, the s output of the agent decreases as a function of y. However, in effect, there is still an implicit unique ordering threshold at the point where the dashed blue line crosses the identity line (dotted black line). This behavior arises from the fact that, during training, s is sampled from the Gaussian distribution of the agent's stochastic policy. Therefore, a very low mean when the inventory position is high ensures that the chances of sampling an s that would trigger an order are low. Conversely, a higher mean for s when the position is low decreases the chances of not ordering when it is needed. Although the S prediction is slightly increasing with the inventory position, and therefore not constant with respect to y, the agent learned a policy with a reorder point close to a true (s, S) policy.

Figure 7b provides the same comparison of the DP baseline but with one of the agents of the Lost Sales experiments in Section 5.4 (-4.47% median gap). The environment

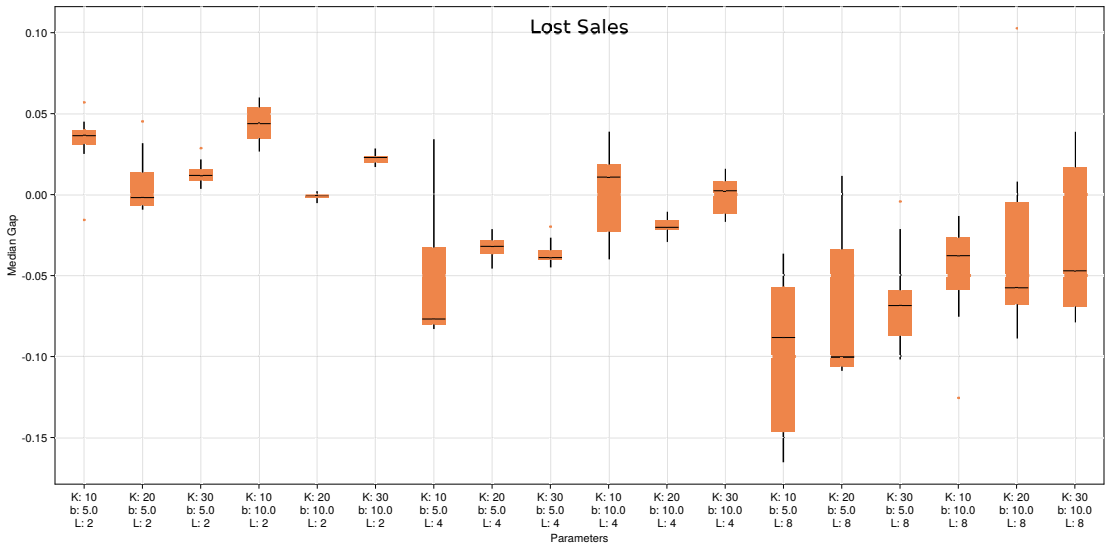

Figure 6: Whisker plots of the median gap for the Lost Sales experiments for varying environment parameters. The gaps are computed with respect to the average cost of the DP policy of each instance. Each data point represents one instance, each whisker plot thus represents ten instances.

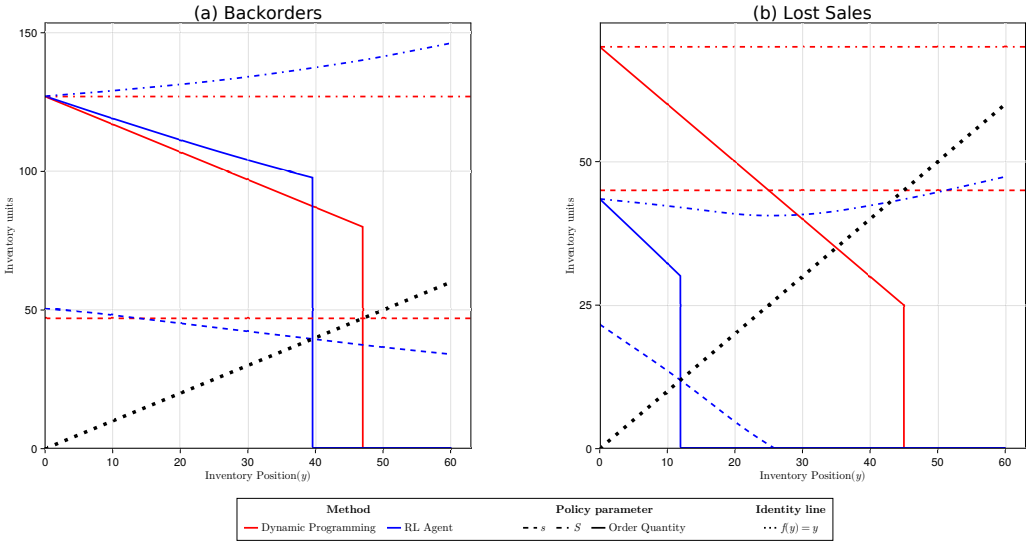

Figure 7: Comparison of the DP (s, S) policies (red) and learned deep RL policies (blue) in a backorders environment (left) and a lost sales environment (right). The dashed and dash-dotted lines are the predicted (s, S) parameters respectively, and the solid lines are the resulting order quantities with respect to the inventory position.

parameters are $K = 30, L = 4, b = 5$, and the trend is the constant 10. The reorder threshold of the DRL policy is significantly lower than that of the DP baseline. This is expected in the case of lost sales because shortages are less expensive in this setting due to only being penalized once, whereas backorders accumulate for possibly several periods until resolved by a delivery. Since the DP baseline solves for a backorder policy, it is unsurprising that it over-conservatively holds more safety stocks than an agent trained in a lost sales environment.

Figure 8 compares how the baseline DP policy and the prediction of the agent used for Figure 7a evolve as the forecast window evolve over time under the seasonal 1 trend. The input state of the actor uses an inventory level z_t equal to the point where it outputs $s_t = z_t$. This reorder threshold is obtained with a line search on z_t (there are no on-order quantities). Evidently, the approximated policies vary as time progresses and the predicted (s_t, S_t) parameters follow a trend that resembles the demand trend, similarly to the DP baseline. The DRL policy is smoother than the DP baseline; this is due to the discretization of the DP algorithm where small errors accumulate during the backward computation of the policy and lead to a sudden jump at some iterations.

		DP			Simple
Σ_0	$\varkappa$	Average	Best	Worst	Average
Backorders					
0.03	0.7	0.75	0.48	1.17	-5.65
0.03	0.8	0.6	0.31	0.87	-5.83
0.03	0.9	1.11	0.76	1.32	-6.51
0.03	0.95	1.44	1.12	1.67	-7.44
0.03	0.99	1.94	1.52	2.45	-8.72
0.05	0.7	0.51	0.38	0.67	-5.17
0.05	0.8	0.49	0.38	0.72	-5.46
0.05	0.9	0.84	0.5	1.27	-6.5
0.05	0.95	1.21	0.76	1.48	-7.56
0.05	0.99	0.6	0.01	1.56	-9.13
0.1	0.7	0.41	-0.01	0.6	-4.14
0.1	0.8	0.49	0.23	0.95	-4.28
0.1	0.9	0.47	0.13	1.23	-5.83
0.1	0.95	-0.47	-1.56	0.33	-7.41
0.1	0.99	1.93	-0.61	5.55	-7.95
Lost Sales					
0.03	0.7	0.55	0.28	0.89	-3.4
0.03	0.8	0.51	0.23	1.06	-3.6
0.03	0.9	1.26	1.01	1.55	-4.13
0.03	0.95	1.52	1.22	1.8	-5.25
0.03	0.99	2.03	1.68	2.75	-6.53
0.05	0.7	-0.02	-0.39	0.22	-2.55
0.05	0.8	0.1	-0.23	0.57	-2.85
0.05	0.9	0.41	0.01	0.8	-3.88
0.05	0.95	0.8	0.39	1.33	-5.31
0.05	0.99	0.09	-1.17	1.94	-6.85
0.1	0.7	-1.02	-1.48	-0.23	-0.89
0.1	0.8	-0.82	-1.5	0.16	-1.32
0.1	0.9	-1.08	-2.0	0.35	-3.18
0.1	0.95	-2.87	-4.59	-1.14	-5.53
0.1	0.99	-3.81	-7.4	5.74	-8.9

Table 6
Summary of the Updating Forecasts experiments (Section 5.3). Each row summarizes the ten agents that were trained with the environment parameters in the two leftmost columns. The four rightmost columns give the average, best, or worst of the ten agents' median gaps (in %) with respect to either the DP method or the Simple heuristic (see the headers).

Finally, Figures 9a and 9b show the heatmap of the gradients of the (s, S) outputs (top and bottom heatmaps respectively) of the same policy neural networks as in Figure 7, with respect to the elements in the forecast window. Each row of the heatmaps correspond to a forecast trend. The gradients are not significantly affected by the input trend. We can observe that the network of Figure 9a only uses the $L+1$ first periods to decide the reorder threshold s, and around 7 more periods to decide S. In Figure 9b, period $L+1$ is almost exclusively used by the neural network to define the reorder threshold.

6. Conclusion

			DP			Simple
K	L	b	Average	Best	Worst	Average
10	2	5.0	3.26	-1.56	5.73	-4.32
20	2	5.0	0.71	-0.93	4.48	-11.51
30	2	5.0	1.35	0.4	2.86	-11.55
10	2	10.0	4.36	2.65	6.03	1.06
20	2	10.0	-0.11	-0.53	0.25	-7.28
30	2	10.0	2.21	1.72	2.84	-5.64
10	4	5.0	-5.31	-8.3	3.4	-10.46
20	4	5.0	-3.3	-4.55	-2.13	-12.77
30	4	5.0	-3.63	-4.48	-1.97	-13.35
10	4	10.0	0.21	-3.98	3.86	-0.74
20	4	10.0	-1.95	-2.92	-1.06	-7.08
30	4	10.0	-0.05	-1.68	1.61	-6.28
10	8	5.0	-10.0	-16.54	-3.63	-12.81
20	8	5.0	-6.82	-10.9	1.18	-13.93
30	8	5.0	-6.47	-10.21	-0.43	-14.68
10	8	10.0	-4.77	-12.54	-1.32	-4.06
20	8	10.0	-3.18	-8.88	10.27	-6.31
30	8	10.0	-2.9	-7.89	3.85	-7.66

Table 7
Summary of the Lost Sales experiments (Section 5.4). Each row summarizes the ten agents that were trained with the environment parameters in the three leftmost columns. The four rightmost following columns give the average, best, or worst of the ten agents' median gaps (in %) with respect to either the DP method or the Simple heuristic (see the headers). The three rightmost columns give the count of agents with a median gap within selected intervals.

In this article we presented a methodology that allows to extend the use of DRL approaches to IC problems that feature a non-stationary demand and a rolling forecast horizon. This methodology bases on the PPO algorithm and allows to approximate the optimal policies of versions of the Single-Item Stochastic Lot-Sizing Problem with non-stationary demand forecasts (i) by augmenting the state encoding with the forecast of the upcoming demand in a fixed-length window and (ii) by training the agents in environments with randomized demand forecasts. This strategy allowed the reuse of the trained neural networks without the need to retrain in subsequent periods. To the best of our knowledge, this is the first method that addresses non-stationary IC problems with a DRL algorithm. The proposed methodology also does not require forecast data to train agents. Specifically, because the forecast is embedded in the state encoding and only known at test time, this approach approximates a generalized replenishment policy given any forecast in the support of the environment initial state distribution. We achieved this by using an extensive dataset of instances consisting of different problem parameters and forecasts.

The three series of experiments carried out in Section 5 showed that the agents can learn close-to-optimal policies on backorder and easy lost sales instances and that they can additionally learn in environments where the forecasts update periodically according to a MMFE. Moreoveor, the

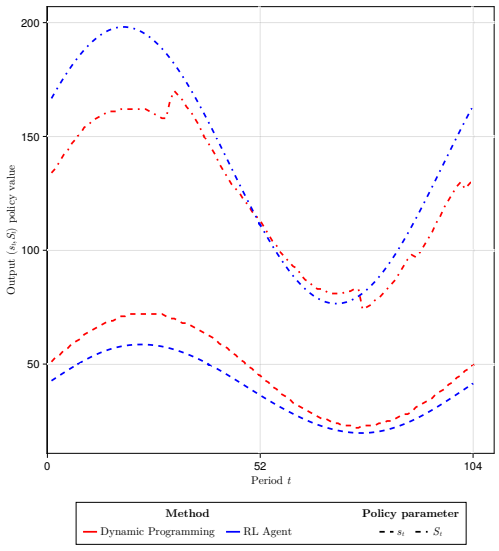

Figure 8: Evolution over time periods of the (s, S) predicted by an agent (blue) and the baseline DP policy (red).

experiments also showed that the agents can learn policies that outperform a DP algorithm in the case of lost sales.

As many machine learning methods, DRL cannot offer any performance or stability of convergence guarantees. Further research effort is thus necessary to alleviate this issue. Notably, the ensembling of several trained models is a frequently used tool in other machine learning areas to compose a more robust model that could prove useful in DRL as well. Pre-training the agents to imitate Simple heuristics or coupling this approach to techniques such as reward shaping (e.g. De Moor et al. (2022)) could also be ways to speed up and stabilize learning.

We also note that this work is also the first application of DRL to IC that uses a continuous policy. This feature paves the way to future research efforts aiming at effectively scaling the approach to large multi-item inventory problems with stochastic demand (Boute et al., 2021).

7. Acknowledgments

This research was carried out under a research mandate provided by the Université de Louvain. The second author acknowledges support from the Fondation Louvain via the research grant COALESCENS and the Belgian National Fund for Scientific Research (FRS-FNRS) via the grant PDR 40007831. Computational resources have been provided by the supercomputing facilities of the Université de Louvain (CISM/UCLouvain) and the *Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles* (CÉCI) funded by the *Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS)* under convention 2.5020.11 and by the Walloon Region.

References

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., Bachem, O., 2021. What matters for on-policy deep actor-critic methods? A large-scale study, in: International Conference on Learning Representations.

Askin, R.G., 1981. A procedure for production lot sizing with probabilistic dynamic demand. AIIE Transactions 13, 132–137.

Axsäter, S., 2015. Inventory Control. volume 225. Springer.

Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B., 2017. Julia: A fresh approach to numerical computing. SIAM Review 59, 65–98.

Bollapragada, S., Morton, T.E., 1999. A simple heuristic for computing nonstationary (s, S) policies. Operations Research 47, 576–584.

Bookbinder, J.H., Tan, J.Y., 1988. Strategies for the probabilistic lot-sizing problem with service-level constraints. Management Science 34, 1096–1108.

Boute, R.N., Gijsbrechts, J., van Jaarsveld, W., Vanvuchelen, N., 2021. Deep reinforcement learning for inventory control: A roadmap. European Journal of Operational Research In-press. doi:https://doi.org/10.1016/j.ejor.2021.07.016.

Clark, A.J., Scarf, H., 1960. Optimal policies for a multi-echelon inventory problem. Management Science 6, 475–490.

Cobbe, K.W., Hilton, J., Klimov, O., Schulman, J., 2021. Phasic policy gradient, in: Proceedings of the 38th International Conference on Machine Learning, PMLR. pp. 2020–2027.

De Moor, B.J., Gijsbrechts, J., Boute, R.N., 2022. Reward shaping to improve the performance of deep reinforcement learning in perishable inventory management. European Journal of Operational Research 301, 535–545.

Dong, L., Lee, H.L., 2003. Optimal policies and approximations for a serial multiechelon inventory system with time-correlated demand. Operations Research 51, 969–980.

Dural-Selcuk, G., Rossi, R., Kilic, O.A., Tarim, S.A., 2020. The benefit of receding horizon control: Near-optimal policies for stochastic inventory control. Omega 97, 102091.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., Madry, A., 2020. Implementation matters in deep rl: A case study on ppo and trpo, in: International Conference on Learning Representations.

Gijsbrechts, J., Boute, R.N., Van Mieghem, J.A., Zhang, D.J., 2022. Can deep reinforcement learning improve inventory management? performance on lost sales, dual-sourcing, and multi-echelon problems. Manufacturing & Service Operations Management 24, 1349–1368. doi:https://doi.org/10.1287/msom.2021.1064.

Graves, S.C., 1999. A single-item inventory model for a nonstationary demand process. Manufacturing & Service Operations Management 1, 50–61.

Graves, S.C., Meal, H.C., Dasu, S., Qui, Y., 1986. Two-stage production planning in a dynamic environment, in: Multi-stage production planning and inventory control. Springer, pp. 9–43.

Heath, D.C., Jackson, P.L., 1994. Modeling the evolution of demand forecasts with application to safety stock analysis in production/distribution systems. IIE transactions 26, 17–30.

Hendrycks, D., Gimpel, K., 2016. Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 .

van Hezewijk, L., Dellaert, N., Woensel, T.V., Gademann, N., 2022. Using the proximal policy optimisation algorithm for solving the stochastic capacitated lot sizing problem. International Journal of Production Research 0, 1–24. doi:10.1080/00207543.2022.2056540.

Hill, R.M., Johansen, S.G., 2006. Optimal and near-optimal policies for lost sales inventory models with at most one replenishment order outstanding. European Journal of Operational Research 169, 111–132.

Iglehart, D.L., 1963. Optimality of (s, S) policies in the infinite horizon dynamic inventory problem. Management Science 9, 259–267.

Iida, T., Zipkin, P.H., 2006. Approximate solutions of a dynamic forecast-inventory model. Manufacturing & Service Operations Management 8, 407–425.

Innes, M., 2018. Flux: Elegant machine learning with julia. Journal of Open Source Software 3, 602. doi:10.21105/joss.00602.

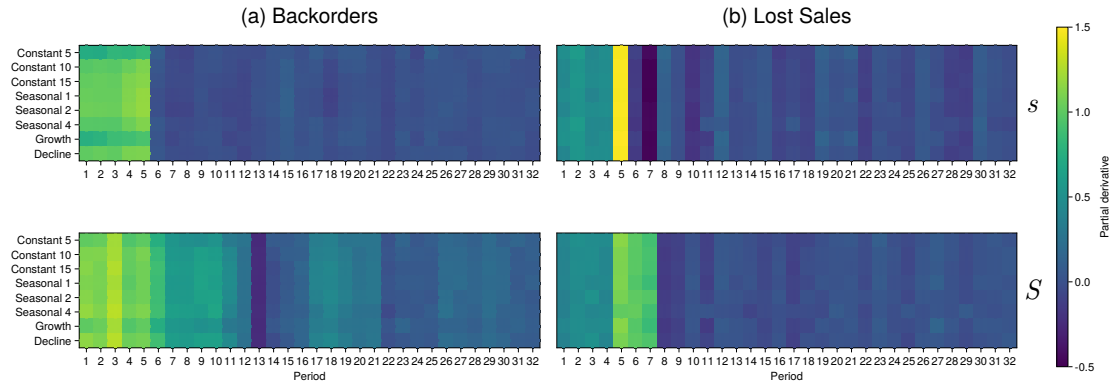

Figure 9: Heatmaps of the gradients of s (top) and S (bottom) with respect to the forecast input for each forecast trend. The agents inspected are that used in Figure 7a (left) and Figure 7b (right), trained in backorders and lost sales environments respectively.

Karlin, S., Scarf, H., 1958. Inventory models of the Arrow-Harris-Marschak type with time lag. Stanford University Press. pp. 155–178.

Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization, in: Proceedings of the 3rd International Conference for Learning Representations, San Diego, California.

Nahmias, S., 1979. Simple approximations for a variety of dynamic leadtime lost-sales inventory models. Operations Research 27, 904–924.

Norouzi, A., Uzsoy, R., 2014. Modeling the evolution of dependency between demands, with application to inventory planning. IIE Transactions 46, 55–66.

Oroojlooyjadid, A., Nazari, M., Snyder, L., Takáč, M., 2021. A deep q-network for the beer game: Deep reinforcement learning for inventory optimization. Manufacturing and Service Operations Management .

Pochet, Y., Wolsey, L.A., 2006. Production planning by mixed integer programming. Springer Science & Business Media.

Porteus, E.L., 1971. On the optimality of generalized (s, S) policies. Management Science 17, 411–426.

Porteus, E.L., 2002. Foundations of Stochastic Inventory Theory. Stanford University Press, CA.

Rossi, R., Kilic, O.A., Tarim, S.A., 2015. Piecewise linear approximations for the static–dynamic uncertainty strategy in stochastic lot-sizing. Omega 50, 126–140.

Saxe, A.M., McClelland, J.L., Ganguli, S., 2014. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, in: International Conference on Learning Representations.

Scarf, H.E., 1960. The optimality of (S, s) policies in the dynamic inventory problem .

Schäl, M., 1976. On the optimality of (s, S)-policies in dynamic inventory models with finite horizon. SIAM Journal on Applied Mathematics 30, 528–537.

Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P., 2016. High-dimensional continuous control using generalized advantage estimation, in: Proceedings of the 4th International Conference on Learning Representations.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 .

Sutton, R.S., Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Vanvuchelen, N., Gijsbrechts, J., Boute, R., 2020. Use of proximal policy optimization for the joint replenishment problem. Computers in Industry 119, 103239.

Veinott, Jr, A.F., 1966. On the opimality of (s, S) inventory policies: New conditions and a new proof. SIAM Journal on Applied Mathematics 14, 1067–1083.

Xiang, M., Rossi, R., Martin-Barragan, B., Tarim, S.A., 2018. Computing non-stationary (s, s) policies using mixed integer linear programming. European Journal of Operational Research 271, 490–500.