Neural Computing and Applications
https://doi.org/10.1007/s00521-021-06129-w

S.I. : ADAPTIVE AND LEARNING AGENTS 2020

Scalable multi-product inventory control with lead time constraints using reinforcement learning

Hardik Meisheri[1] · Nazneen N. Sultana[1] · Mayank Baranwal[1] · Vinita Baniwal[1] · Somjit Nath[1] · Satyam Verma[2] · Balaraman Ravindran[3] · Harshad Khadilkar[1]

Received: 16 November 2020 / Accepted: 15 May 2021

Abstract
Determining optimum inventory replenishment decisions are critical for retail businesses with uncertain demand. The problem becomes particularly challenging when multiple products with different lead times and cross-product constraints are considered. This paper addresses the aforementioned challenges in multi-product, multi-period inventory management using deep reinforcement learning (deep RL). The proposed approach improves upon existing methods for inventory control on three fronts: (1) concurrent inventory management of a large number (hundreds) of products under realistic constraints, (2) minimal retraining requirements on the RL agent under system changes through the definition of an individual product meta-model, (3) efficient handling of multi-period constraints that stem from different lead times of different products. We approach the inventory problem as a special class of dynamical system control, and explain why the generic problem cannot be satisfactorily solved using classical optimisation techniques. Subsequently, we formulate the problem in a general framework that can be used for parallelised decision-making using off-the-shelf RL algorithms. We also benchmark the formulation against the theoretical optimum achieved by linear programming under the assumptions that the demands are deterministic and known apriori. Experiments on scales between 100 and 220 products show that the proposed RL-based approaches perform better than the baseline heuristics, and quite close to the theoretical optimum. Furthermore, they are also able to transfer learning without retraining to inventory control problems involving different number of products.

Keywords Multi-agent reinforcement learning · Supply chain · Scalability and parallelisation

1 Introduction

Optimal control of inventory is one of the most practical and complex problems in supply chain management. The problem entails periodic replenishment of perishable items with different lead times, unit costs and volumes at a store (or warehouse) facing stochastic demands. Crafting an optimal policy for replenishment and the order time of the products is a critical managerial decision that directly impacts the total inventory cost comprising of costs incurred due to understocking/overstocking of products and transportation costs. One of the seminal works involving dynamic programming [19] developed a theoretical characterization of the optimal base-stock policies for simple series systems. However, the complexity of the underlying recursive equations grows significantly with the problem size. Consequently, several approximation algorithms were proposed to address this issue, including, but not limited to stochastic approximation [38], infinitesimal perturbation analysis (IPA) [50], and approximate dynamic programming [9]. However, these methods remain hard to implement in practice [11].

On the other hand, some of the most widely used policies, such as the economic-order-quantity-model and the

Hardik Meisheri and Nazneen N. Sultana contributed equally to this study.

✉ Hardik Meisheri
hardik.meisheri@tcs.com

Balaraman Ravindran
ravi@cse.iitm.ac.in

[1] TCS Research, Mumbai, India

[2] IIT Bombay, Mumbai, India

[3] Robert Bosch Centre for Data Science and AI, IIT Madras, Chennai, India

Published online: 28 May 2021

dynamic-economic-lotsize model, both of which assume deterministic demands [79], do not address realistic scenarios of stochastic demands. Authors in [11] addressed both these issues for a single-stage single-product inventory control problem using a robust linear programming framework over a finite horizon framework. While their approach is computationally scalable, lack of cross-product constraints and backlogging of excess demands are not practically viable in a typical inventory control problem. Consequently, several policies that involve multiple product, multiperiodic constraints have been recently proposed [2, 3, 17, 29, 44, 77]. In this paper, a novel reinforcement learning (RL) approach for a multi-product inventory control problem with different lead times, unit weights and volumes, shelf-lives, and shelf capacities is proposed to address the practical scenarios involving cross-product constraints, stochastic demands and a very large number of products.

RL algorithms have been successfully applied to a broad range of problems in the literature, including virtual gameplay [43, 65], physical systems such as autonomous driving [61] and flight dynamics [46], and problems from the field of operations research [36, 75, 78]. An essential feature of these applications is the ability to plan strategies that maximise a long term discounted future reward under constraints. This property makes RL a natural candidate for decision-making in supply chains, where the key complexity is the large (and possibly variable) number of concurrent decisions that need to be computed in each time period. We model this scenario as a high-dimensional stochastic dynamical system, in order to demonstrate that the subsequent RL formulation can also be useful in related domains. The problem formulation and solution discussed in this paper are part of a larger effort to introduce autonomous, adaptive decision-making in retail supply chains [60].

1.1 Motivating example

Figure 1 illustrates a typical multi-product retail inventory management scenario requiring optimum decision making at various levels. A moderately large retail business may be composed of approximately 1,000 stores, with each store selling up to 100,000 product types. The inventory of products in each store is periodically replenished by trucks (for example, once per day). The full range of products is carried from the warehouse to the store on the same truck (or set of trucks), which imposes total volume and weight constraints on the replenishment decisions. There are multiple trade-offs involved in this process, including maintenance of minimum inventory, minimisation of wastage due to products going past their sell-by dates, fairness across the product range, and capacity sharing on the truck. We focus on the last step in the supply chain, shown by the shaded region in Fig. 1: replenishment of products from the local warehouse, in order to concurrently maintain inventory levels of the entire product range within the store. Inventory within the store is depleted by the sale of products to customers, and through wastage of perishables. Customer demand is stochastic, with noisy predictions provided by an external forecasting system. Depleted inventory is periodically replenished, with the quantity of each product in each delivery being decided by the RL algorithm.

1.2 Modelling assumptions

Real-world operations in a retail supply chain are typically highly dynamic [24], with ad-hoc coordination between the warehouse, store, and logistics/transportation providers. In order to model the problem, we make some simplifying, yet practical, assumptions.

- The prior steps in the supply chain are well organised, so that the warehouse itself has sufficient inventory for each product. This implies that the only constraints on the quantity of replenished products are imposed by the volume and weight constraints of the truck.
- Each product has a fixed designated area for display within the store, so that the maximum number of items of each product that can be stocked is known in advance.
- The volume and weight capacity of the truck for each time period is assumed to be known in advance. In reality, the retailer may decide to send additional truck(s) at the time of shipment, if the demand for inventory is sufficiently high. We do not handle this corner case.
- The deliveries happen periodically at fixed time intervals (though not all products need to be replenished in every delivery).

These assumptions allow us to formulate the problem from an analytical perspective in Sect. 2, propose a reinforcement learning approach to learn replenishment policies in Sect. 4, and to experiment with multiple instances (including transfer learning) in Sects. 5 and 6. The simplified version of the problem is still a good approximation of reality, and results in policies that can be explained and justified from an operational perspective.

1.3 Contributions

The principal contributions of this paper are (1) formulating a *parallelisable* RL approach to solve the multi-product constrained inventory management problem, (2) including realistic business goals such as availability of the entire

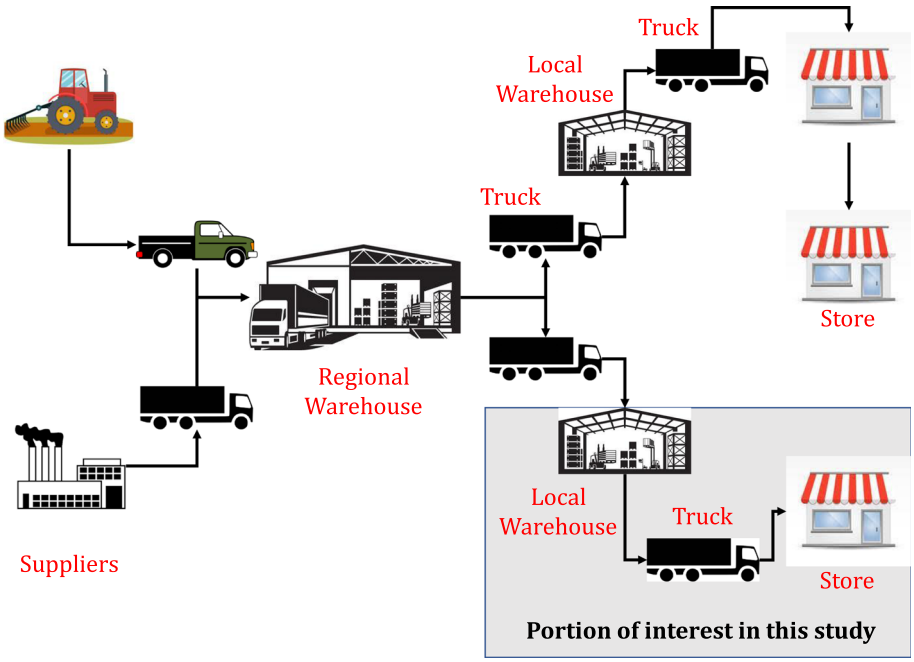

Fig. 1 Flow of products within a retail supply chain. The goal is to maintain inventory of the full range of products in the stores, while minimising costs. The present study considers the last step in the supply chain: movement of products from local warehouse to the store

range of products and minimisation of wastage, and (3) showing that the approach and its learned policy can be transferred without additional training, to instances with a different number of products. A secondary contribution is to emphasise the close relationship between the current problem and the generic control problem in system dynamics. We believe it is possible to use similar approaches for solving other constrained resource allocation problems.

2 Problem description

We now describe a generic dynamical system control problem and show that multi-product inventory management is a special case. We also develop the equivalent reinforcement learning formulation, while the specific solution approach is described in the next section.

2.1 Multivariable dynamical system control and RL formulation

Consider a system with continuous-time dynamics, where inputs are provided at discrete time intervals. This form captures a large variety of real-world systems with digital controllers. The system dynamics between two time steps is given by:

$$\dot{\mathbf{x}}(z) = F(\mathbf{x}(z)) + \mathbf{w}(z), \tag{1}$$

where $\mathbf{x}$ is the vector of p state variables, $F(\mathbf{x}) : \mathbb{R}^p \rightarrow \mathbb{R}^p$ is a (possibly nonlinear) function, $\mathbf{w}$ is external noise, and z denotes continuous time. The initial state $\mathbf{x}(0)$ is arbitrarily specified. The control input is provided at discrete time step t. Without loss of generality, let us assume that $t \in \mathbb{Z}^+$, the set of positive integers. The effect of control is an instantaneous change in the state $\mathbf{x}$,

$$\mathbf{x}(t)^+ = \mathbf{x}(t)^- + B\,\mathbf{u}(t). \tag{2}$$

The control problem specifies that some objective r composed of the system states $\mathbf{x}$, a set of parameters θ, and a discount factor γ, be maximised in the long term. Formally, we state the optimal control problem as the computation of vector $\mathbf{u}^*(t)$ where

$$\mathbf{u}^*(t) = \arg\max \int_t^\infty \gamma^{z-t}\, r(\hat{\mathbf{x}}(z), \mathbf{u}(t), \theta)\, \mathrm{d}z, \tag{3}$$
$$\text{subject to } \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t)) \leq 0, \quad \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) = 0,$$

where $\hat{\mathbf{x}}(z)$ is an estimate of the future state trajectory, $\mathbf{h}$ is a set of inequality constraints on the values of the state and control inputs, and $\mathbf{g}$ is a set of equality constraints. A typical definition of r is a quadratic function composed of error with respect to a reference state trajectory and the control effort expended. We assume that system dynamics F is unknown but the control input matrix B is known, and both F and B could in general be time-varying. However, the variation in nature of F, B, and $\mathbf{w}$ is slow enough to build reasonable estimators $\hat{F}$ and $\hat{\mathbf{w}}$, thus admitting an explicit or implicit predictor for the trajectory $\hat{\mathbf{x}}$ given historical values of $\mathbf{x}$. The noise statistics of $\mathbf{w}$ can be arbitrary.

The generic dynamics of system evolution and control input given by (1) and (2), and the optimal control problem defined by (3), together specify a broad class of control problems for dynamical systems. For example, $\mathbf{x}$ could be the position and velocity of an autonomous robot, while $\mathbf{u}$ is the force or acceleration input. We now show how these relations can be framed as a standard RL problem, and then specialise to the inventory management scenario.

The problem can be modeled as a Markov Decision Process $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ represents the state space defined by a feature map $\mathbf{f}(x) : \mathbb{R}^p \to \mathbb{R}^{m \times p}$, $\mathcal{A}$ denotes the decision or action space $\mathbf{u}(t) \in \mathbb{R}^p$, $\mathcal{T}$ represents transition probabilities from one combination of state and action to the next, $\mathcal{R}$ denotes the rewards, and γ is the discount factor for future rewards. Given the optimisation task as defined in (3), the objective is equivalent to a discounted sum of aggregated discrete rewards $R(n)$, as shown below:

$$\begin{aligned} &\text{Max: } \int_{t}^{\infty} \gamma^{z-t}\, r(\hat{\mathbf{x}}(z), \mathbf{u}(t), \theta)\, dz \\ \equiv &\text{Max: } \sum_{n=t}^{\infty} \left(\gamma^{n-t} \int_{n}^{n+1} \gamma^{z-n}\, r(\hat{\mathbf{x}}(z), \mathbf{u}(t), \theta)\, \mathrm{d}z \right) \\ \equiv &\text{Max: } \sum_{n=t}^{\infty} \gamma^{n-t} R(n). \end{aligned} \tag{4}$$

While (3) is a standard optimisation problem, the arbitrary nature of F, B, and $\mathbf{w}$, the possible nonlinearity of $\mathbf{h}$ and $\mathbf{g}$, and the online response requirement of computing $\mathbf{u}(t)$ make it a difficult proposition to solve using traditional approaches. Even with simple linear forms of F, large dimensionality of $\mathbf{x}$ and $\mathbf{u}$ can make the problem intractable. On the other hand, the reward structure (4) readily admits the use of RL for computing the inputs $\mathbf{u}(t)$. Model-free RL can implicitly model the estimator for noise as well as the future state trajectory in order to maximise the discounted long-term reward, requiring only (1) a feature set $\mathbf{f}(x)$ as the input, and (2) a mechanism (such as simulation) for rolling out the effects of actions on the states in a closed-loop fashion.

2.2 Inventory management as an optimal control problem

We instantiate the generic system dynamics with a multi-product inventory management scenario, illustrated in Fig. 2. As mentioned before, this is the last step of the retail supply chain from Fig. 1. Our goal is to maintain sufficient inventory levels of all products to ensure availability for sale, while simultaneously minimising wastage due to spoiling of perishable products. The former objective is due to business requirements, while the latter objective is directly linked to incurred cost. We define the state $\mathbf{x}$ in (1) to represent the inventory levels of all products normalised by the corresponding shelf capacity. The depletion of inventory is modelled by the system dynamics, with the sale of products representing the noise variable $\mathbf{w}$, and the spoiling of perishable inventory representing the internal dynamics F. The inventory has to be periodically replenished in a quantity equal to the control variable $\mathbf{u}$. Inequality constraints are imposed by the maximum inventory levels of each product as defined by available shelf space, and replenishment $\mathbf{u}$ in each time step is constrained in terms of total weight and volume by the load carrying capacity of the truck.

Assume each element x_i of state $\mathbf{x}$ to be the inventory level of product i, and the rate of product sales is the noise $\mathbf{w} \geq \mathbf{0}$ (with a negative sign, as in (5)). Inventory depletion due to spoiling of perishables is assumed to be at a fixed proportional rate a_i for each product. The dynamics is thus $F(\mathbf{x}(z)) = A\,\mathbf{x}(z)$, where matrix A is a constant diagonal matrix with entries $-a_i$. Larger values of a_i are for products that perish correspondingly faster[1]. Since $\mathbf{u}(t)$ represents the replenishment actions at time t, its dimension is equal to that of $\mathbf{x}$ and B is the identity matrix. Recall that we denote continuous time by z and the discrete instants of replenishment by t. The dynamics are thus explicitly given by:

$$\begin{aligned} \dot{\mathbf{x}}(z) &= \begin{pmatrix} -a_1 & \dots & 0 \\ \vdots & -a_i & \vdots \\ 0 & \dots & -a_p \end{pmatrix} \mathbf{x}(z) - \mathbf{w}(z), \\ \mathbf{x}(t)^+ &= \mathbf{x}(t)^- + \mathbf{u}(t). \end{aligned} \tag{5}$$

The inventory levels x_i and control inputs u_i are assumed to be continuous variables. In the case of unit-based products (for example, groceries), this assumption is approximately true as long as the maximum shelf capacity is significantly larger than the size of individual units.

The inventory level of product i between two time periods can be propagated by integrating relation (5), which depends on the noise $\mathbf{w}(z)$. Note that the diagonal form of A implies that the inventory equations can be solved independently between replenishment instants. We assume that the integral of the noise (total sales) in one time step is $W_i(t-1) = \int_{t-1}^{t} w_i(z)\mathrm{d}z$. Since the time period is assumed to last one unit, the average rate of sale within the time period is also equal to $W_i(t-1)$, which we shorten to W_i for notational simplicity. Assuming that the rate W_i is a constant in a given time period, the rate of depletion for i

[1] For products with fixed expiry dates, wastage happens at discrete time instants t with $a_i = 0$. The quantity of wastage is known in advance, and can be absorbed in $\mathbf{u}(t)$.

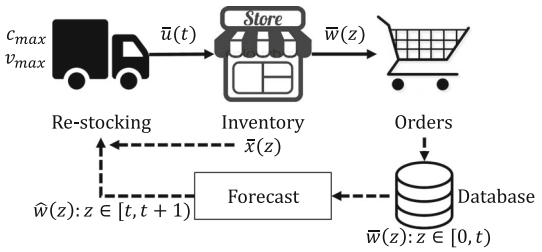

Fig. 2 The inventory management problem considered in this work

is $(-a_i x_i - W_i)$. Integrating (5) gives the inventory at the end of the time period,

$$x_i(t)^- = e^{-a_i} x_i(t-1)^+ - \frac{W_i}{a_i}(1 - e^{-a_i}). \tag{6}$$

Since the inventory cannot be a negative value, we assume that orders for any products that have no inventory are rejected and therefore $w_i(z) = 0 \; \forall z \in [z', t)$ if $x_i(z') = 0$. We note that even the functional form of the relationship (6) is not assumed known to the RL agent. It must either model the state and control relationships explicitly, or use a model-free technique as described in the next section. However, the relation (6) defines the state update relationship for each product in the inventory, which we use in the environment (simulation) portion of the RL algorithm. Assuming that the inventory levels and control inputs are normalised to the range [0, 1], the set of applicable constraints is as follows.

$$\mathbf{0} \leq \mathbf{x}(t) \leq \mathbf{1} \tag{7}$$

$$\mathbf{0} \leq \mathbf{u}(t) \leq \mathbf{1} \tag{8}$$

$$\mathbf{0} \leq \mathbf{x}(t)^- + \mathbf{u}(t) \leq \mathbf{1} \tag{9}$$

$$\mathbf{v}^\intercal \mathbf{u}(t) \leq v_{\max} \tag{10}$$

$$\mathbf{c}^\intercal \mathbf{u}(t) \leq c_{\max} \tag{11}$$

Here, constraints (7) and (8) are related to the range of acceptable values of each product. Constraint (9) states that the level of inventory just after replenishment ($\mathbf{x}(t)^+$ according to (2)) cannot exceed the maximum inventory level. Constraints (10) and (11) set maximum values on the total volume $v_{\max}$ and weight $c_{\max}$ of products replenished at a single time step, mimicking transportation capacity limitations. Column vectors $\mathbf{v}$ and $\mathbf{c}$ are constants corresponding to the unit volume and weight multipliers for each product.

Inventory management is a multi-objective optimisation problem, with direct costs relating to (1) the reduction of inventory for some products to 0, commonly known as out-of-stock, and (2) the quantity $q_{\text{waste},i}(t)$ of products wasted (spoiled) during the time period ending at t. In addition, we wish to ensure that some products are not unfairly preferred over others when the system is stressed (for example, when the capacities $v_{\max}$ and $c_{\max}$ are too small to keep up with product sales). Therefore, we include a fairness penalty on the variation in inventory levels across the product range, from the 95th to the 5th percentile (denoted by $\Delta\mathbf{x}(t)_{.05}^{.95}$) across all products. We also add a penalty $\Omega(t)$ on the total orders refused during the time interval, aggregated over all products. The objective (reward) to be maximised is defined in (12). Since no control intervention is allowed between two time steps, all the terms can be considered to be aggregate values received at time t. The objective is defined by:

$$R(t) = 1 - \underbrace{\frac{p_{\text{empty}}(t)}{p}}_{\text{Outofstock}} - \underbrace{\frac{p_{\text{critical}}(t)}{p}}_{\text{Criticallevel}} - \underbrace{\frac{\sum_i q_{\text{waste},i}(t)}{p}}_{\text{Wastage}} - \underbrace{\Delta\mathbf{x}(t)_{.05}^{.95}}_{\text{Percentilespread}} - \underbrace{\Omega(t)}_{\text{Refusedorders}}, \tag{12}$$

where p is the total number of products (size of $\mathbf{x}$), $p_{\text{empty}}(t)$ is the number of products with $x_i = 0$ at the end of the period $[t-1, t)$. $p_{\text{critical}}(t)$ is the number of products with $x_i < \chi_i$ at the end of the period $[t-1, t)$ which ensures that each product maintains it minimum presentation level (χ_i) or critical inventory. Since the maximum value of $p_{\text{empty}}(t)$, $p_{\text{critical}}(t)$ and $\Omega(t)$ is equal to p, the maximum value of $q_{\text{waste},i}(t-1, t)$ is 1, and the maximum difference in inventory between two products is also 1, the theoretical range of the objective/reward is $-4 \leq R(t) \leq 1$. For practical purposes, the individual terms will be smaller than 1, and the majority of rewards should be in the range $[-1, 1]$. The goal of the algorithm is to maximise the discounted sum of this reward as per (4), given the dynamics (5) and the constraints (7)–(11).

2.3 Inventory management with variable lead time

In the previous section, the formulation of optimal inventory management assumes that products arrive at the store almost instantly after the decision to replenish is made. However, delay in manufacturing and other logistics constraints necessitate delay in procuring products once the replenishment decision is made. Thus, the replenishment decision at any time t must also account for any such delays. The time delay between the between taking decisions and replenishment being reflected in the inventory is termed as the lead time in the supply chain literature. In this section, we formulate a scenario where there are different fixed lead times products. Figure 3, shows a visual schematic of products with different lead times. Blue,

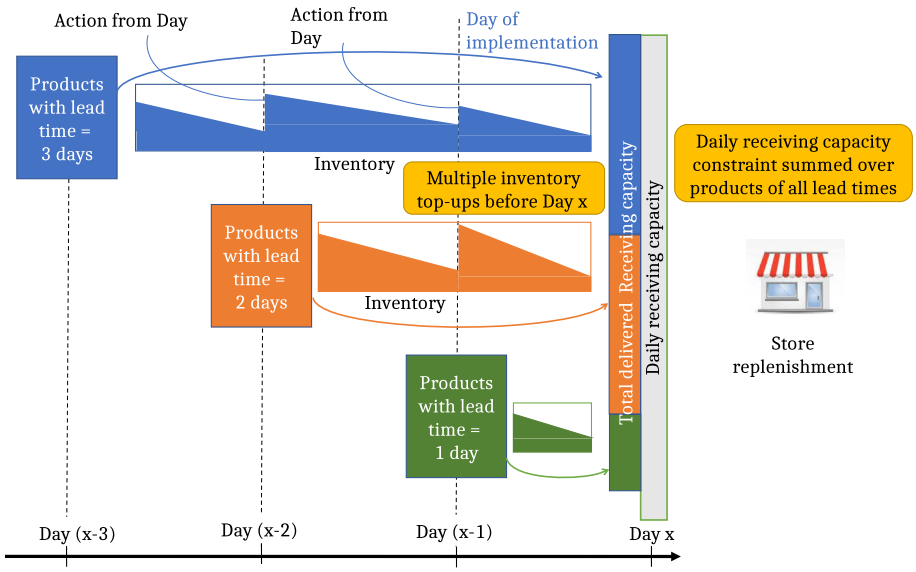

Fig. 3 Illustration of delivery process with products having three different lead times

orange and green refer to three products with 3, 2, and 1 lead times, respectively.

The major difference with respect to the previous formulation is that decisions for higher lead time products need to be taken before lower lead time products. This creates a challenge for truck capacity and/or other joint constraints where decisions are not taken at the same time and hence cannot be modified if they violate the capacity constraints. Apart from the set of constraints (7–11) relating to acceptable values, level of inventory, the maximum level of inventory and truck volume and weight restrictions, the inclusion of variable lead time includes some additional constraints. Similar to the estimator for sales in the stores in the next period, we assume the existence of another estimator to predict the inventory at $t + l_i - 1$ which is when the replenishment actions from the previous time steps will take effect.

$$\hat{x}(t + l_i - 1) = \mathbf{F}_1(x(t), \mathbf{U}(t - l_i : t), \mathbf{W}(t : t + l_i - 1)) \tag{13}$$

where $\mathbf{U}(t - l_i : t)$ is the amount of replenishment quantities for the current lead time l_i products from the beginning which only takes implemented after a lag of l_i delivery moments. $\mathbf{W}(t : t + l_i - 1)$ is the sales forecast estimator which leads to compounding errors in the $\mathbf{F}_1$ estimator because of noisy forecast in longer horizon. Here, l_i is the lead time for each individual i products. Noise in a multi-step forecast is directly proportional to horizon.

3 Related work in control and learning-based approaches

We classify prior literature related to this work into two categories: an overview of decision-making algorithms for dynamical systems and a description of prior work specific to inventory management. While the mathematical aspects of these problems have been studied extensively, we posit that our dimension-agnostic meta-model (covered in Sect. 4) is novel in the literature.

3.1 Decision-making algorithms for dynamical systems

3.1.1 Traditional control techniques

In Sect. 2, we showed that the inventory management problem is related to the multivariable dynamical system control problem. Dynamical systems are typically distinguished based on the form of state that they aim to control. The simpler form is a scalar state such as the rate of a chemical reaction [28] or temperature of a boiler [41]. A more complex version is the multivariable control problem, involving a vector of (possibly interdependent) states. The evolution of the state vector in the canonical multivariable system is modelled using matrix differential equations [47]. The preferred control approach for multivariable problems is to use a linear time invariant (LTI) model of the system and to design a controller using classical methods in the frequency domain or using state feedback [13, 27]. Non-linear versions of the problem are solved using techniques such as sliding mode [72] and model predictive control [42], while robust control under stochastic disturbances is handled using techniques such as H_∞ [22]. The key

takeaway from these techniques is that they address one complex aspect of the problem in isolation, focusing on one of (1) stability and robustness, (2) high dimensionality of state space, or (3) system identification. By contrast, we require an approach that will handle all these aspects simultaneously. We, therefore, turn to data-driven methods.

3.1.2 Data-driven control approaches

There exists a large volume of existing literature on adaptive control [4, 32], where the control law is defined as a functional relationship, while the parameters are computed using empirical data. However, adaptive control typically requires analytical models of the control and adaptation laws. Approximate dynamic programming (ADP) [8, 49] has a similar dependence on analytical forms of the value function, at least as a weighted sum of basis functions. It also requires explicit state transition probabilities and stage costs, which may not be available in the current context. The closest form of ADP for problems of the current type is the literature on adaptive critics [64], which has considerable overlap with reinforcement learning.

ADP in the policy space solves the problem by using policy gradients to compute the optimal policy parameters, each of which defines a stationary policy. ADP has been used in prior literature for relatively large task allocation problems in transportation networks [26, 71]. These studies use nonlinear approximations of the value function, but the forms are still analytically described. Furthermore, they require at least a one-step rollout of the policy. This may not be feasible in the current context, since the dimensionality is high and each action is continuous (or at least finely quantised), and the goal is not to track some reference signal as in the standard linear quadratic regulator (LQR) [47].

Imitation learning (IL) is a well-known approach for learning from expert behaviour without having any need of a reward signal and with the simplicity of a supervised learning. This approach assumes that expert decisions have considered all the constraints of the system in order to accomplish the objective. IL has been used in variety of problems including games [54], 3D games [30], and robotics [23]. The inherent problems of design complexity and performance limitations apply here as well; to the definition of the expert policy rather than to the IL algorithm. Additionally, the general form of the problem may not admit an obvious expert policy to train with.

3.1.3 RL techniques

Value-based methods in reinforcement learning are easy to use [43, 73], while advantage actor critic (A2C) [37] and deep deterministic policy gradients (DDPG) [40] have been shown to work well on continuous action spaces. Trust region policy optimisation [56] and proximal policy optimisation [57] have also proven effective for optimal control using RL. Branching DQN [70] is able to handle multiple actions, but has a significant growth in complexity with the size of the state space. DDPG and DQN are categorised under off-policy algorithms as they use older policy samples to improve upon current policy, whereas A2C, TRPO and PPO are on-policy algorithms use only current policy data to improve the policy. We benchmark and test DQN (off-policy) and PPO (on-policy) over two different datasets to better investigate the applicability of RL algorithms towards inventory management.

3.2 Traditional approaches to inventory management

Supply chain control problems have been extensively studied in the operations research literature [67, 69], and we cannot reasonably hope to cover all prior work in this paper. Therefore, we focus on a representative cross section of papers. We divide the literature into three broad algorithmic approaches: (1) exact optimisation, (2) heuristics and meta-heuristics, and (3) learning-based techniques. While there are many studies of each type, ones that handle a large number of products, non-zero lead times, noisy forecasts, and fast response times simultaneously are very rare. We cover these studies in detail below.

3.2.1 Exact optimisation

There exist studies that use mixed-integer linear programming (MILP) [18, 68] and related techniques such as branch-and-cut [20] for inventory management of multiple products. However, these formulations are developed without consideration of lead times, and for a small number of products (typically fewer than 10) and short time horizons. The reason for this limitation is the large computation time for exact optimisation approaches, as we demonstrate in Sect. 6 with a formulation specifically developed for solving the current replenishment problem (but with full prior knowledge of demand). Due to their large computational times, these approaches are also difficult to use in a rolling horizon manner (since one would need to run the algorithm in every time step).

One workaround towards computational tractability is to model the replenishment policy as piecewise linear, and to solve the problem using second-order cone optimisation

(SOC) [58]. The authors of this study showed that the algorithm can handle multiple time periods and lead times, and achieves polynomial complexity (much better than exponential approaches such as dynamic programming). However, it is demonstrated for a single product, and is unlikely to be tractable for a large number of products. Similar limitations are observed in other near-optimal approximations [10, 51]. The key takeaway appears to be that for large scale multi-product, multi-stage problems with lead times, one must turn to approximate approaches.

3.2.2 Heuristics and meta-heuristics

The most common set of heuristics are threshold policies defined by (R, s, S) [45, 74, 76], where the policy is to replenish a product to an order-up-to level S at review instants R, whenever the inventory falls below a threshold s. Some studies compute these thresholds and order-up-to levels using domain knowledge [21, 45], while others use optimisation approaches with known demand distributions [76]. In either case, the class of control policies is defined by (R, s, S) and one does not have the option of time-specific, forecast-driven quantities; nor do these studies consider capacity sharing between multiple products. In special circumstances, replenishment rules may be defined using mathematical formulae [14, 66] if the demand characteristics can be modelled analytically. For the results described in Sect. 6, we use the popular threshold policy as a baseline, with some modifications to ensure satisfaction of the multi-product capacity constraint.

Among the class of meta-heuristic algorithms, a few studies have used genetic algorithms (GA) [1, 52] for the inventory management problem. The approach is to either use GA for computing the order-up-to level S as defined above, or else to solve the multi-stage replenishment problem directly (which is the problem of concern in this paper). However, given the computational complexity and running time of GA, such studies are limited to single-product scenarios. Simulated annealing (SA) has been used for solving the multi-product multi-stage inventory management problem for perishable products [59]. However, this approach suffers from the same computational problems; the study reports a run time of more than 1 hour for a 600 product, 15 step scenario. It also does not consider lead times.

We can summarise these studies by noting that heuristic algorithms possess the required computational characteristics and ability to handle the essential constraints, but are limited to threshold-type replenishment policies. Meta-heuristics can be used for more customised replenishment decisions, but their computational requirements are prohibitive. A major part of this problem is their inability to reuse learning from previous runs. Therefore, we identify learning-based approaches as the most promising class of algorithms for the present work.

3.2.3 Learning-based approaches

Apart from reinforcement learning, approaches that can reuse prior training for speeding up online decision-making can be classified into at least four types: Adaptive control (AdC) [4], model predictive control (MPC) [15], approximate dynamic programming [8, 49], and imitation learning [55]. Of these, AdC and MPC require analytical models of the system model, the former for deriving the control law, while the latter for rolling out over multiple time periods. These methods are thus more suited to physical dynamical systems. ADP [8, 49] also depends on analytical forms of the value function, and requires explicit state transition probabilities and stage costs. However, some variants such as adaptive critics [62–64] and adaptive dynamic programming [48] have been previously used for the inventory management problem (sometimes called the product dispatching problem). These studies either use explicit value function forms [48], or a centralised critic [63] where neural networks are employed. There are some residual problems with generalisability and scalability, both of which are addressed by reinforcement learning (described below). The final set of techniques before moving to RL is called imitation learning, which learns from expert behaviour assuming that the ideal policy is able to maximise, whatever objective is being targeted. The inherent problem here is that the general form of the problem may not admit an obvious expert policy to train with. While this is feasible in special cases [5], this is not possible in the present instance.

Some prior studies use reinforcement learning for the inventory management problem [25, 33, 35], but principally for the single-product version. The idea of using RL for such problems goes back several decades [74], though it was initially proposed for the single-product scenario. They also focus on specific goals, such as maximum profit [25], minimum operating cost [35], or maintaining target inventory [33]. Prior work on a simpler version of the problem [6] focussed on the use of a complex simulation framework in a closed loop with RL for training. The model did not explicitly incorporate system constraints (truck weight and volume capacity), and the business objectives were quantified in simplified form. In the present work, we motivate the modelling approach thoroughly, describe the RL formulation in detail, incorporate constraints into the model, and experiment on problem instances with different number of products (including transfer learning). Apart from the problem scale (we report instances up to 100,000 products over 900 time steps), we also explicitly account for lead times in replenishment,

which is a crucial and often overlooked aspect of the problem.

3.3 Complexity of current problem in comparison with literature

Before moving to the description of our formulation, we wish to emphasise the novel nature of such real-world problems in comparison with the more popular applications of RL in games.

1. We note that real-world RL problems have no clear distinction between the 'modelled environment' and the rest of the world, because these are open systems. In the case of replenishment, the supply chain is affected by not just customer demand and its internal constraints, but by several other socio-economic, geo-political, and meteorological factors. It is thus difficult to design an effective simulator for training the algorithms.
2. There is no obvious reward function, because the performance of the system is subjective to some extent (no clear winning or losing outcomes). Ill-designed reward functions can lead to undesirable behaviours in the algorithm. For example, an emphasis on 'average availability of products' can lead to unfair policies, focussing only on products that have slower sales.
3. The size of the task can vary during the course of use. For example, new products get introduced and old products are discontinued frequently in the inventory management problem. Any proposed solution needs to be agnostic towards such variations.
4. Unlike adversarial situations, models developed for the current problem must be amenable to scrutiny. They must also train and work on reasonably small hardware.
5. Finally, the accumulation and preparation of data about the system are a challenging task in itself.

In the next two sections, we describe a formulation that takes into account these challenges, and tries to address them in as computationally efficient a way as possible. They key to addressing the explainability and scalability challenges is the definition of a single-product meta model, which allows us to parallelise the decision-making and make it scale-agnostic.

4 Methodology

As mentioned in Sect. 3, we benchmark our solution using two popular algorithms, DQN [43] and PPO [57]. While DQN is off-policy algorithm which uses previous policy samples to improve its value estimation, PPO is an on-policy data which has been proven to be state of the art for various tasks such as board games, robotic arm dexterity and even complex games such as Dota2. These two different algorithms prove our methodology is robust and algorithm agnostic since any off-the-shelf RL algorithm can be used. We have used a four-layer neural network for DQN with the last layer providing the Q values for each actions. For PPO, we used a similar network as DQN with an addition of a value network layer, while all other layers are shared between policy and value networks.

4.1 No lead time

We note that the order rate $\mathbf{w}(z)$ plays a key role in the system dynamics (5)–(6). For simplicity, we define an estimator for the sales rate w_i of each product i in the form of a trailing average of sales in the most recent T time periods,

$$\hat{w}_i(z) = \frac{\int_{t-T}^{t} w_i(z')\mathrm{d}z'}{T}, \forall\, z \in [t, t+1)\, i \in \{1, \ldots, p\}. \tag{14}$$

It must be remarked that the above forecasting algorithm returns a very crude approximation of the actual demand. Instead, one can leverage more sophisticated forecasting algorithms [7, 16, 31], such as recurrent neural network (RNN)-based approaches to provide a better estimate of the true demand; however, design of the most appropriate forecasting algorithm is not the main focus of this paper. Instead, we rely on the simpler averaging based forecasting algorithm and still achieve near-optimal performance for inventory control. A better forecasting algorithm would only aid the RL algorithm in further improving its capability towards optimal replenishment decision. Note that all the competing algorithms tested in Sect. 6 use the same values of forecast orders. Since we assume that each period lasts for a unit of time, the forecast for aggregate orders, $\hat{W}_i(t)$, is also given by (14).

The primary challenges in applying RL to this problem are (1) the large number of products p, (2) handling the shared capacity constraints (10)–(11), and (3) the fact that the number of products can change over time. We describe an algorithm for parallelised computation of replenishment decisions, by cloning the parameters of the same RL agent for each product and computing each element of the vector $\mathbf{u}(t)$ independently. The advantage of this approach is that it splits the original problem into constant-scale sub-problems. Therefore, the same algorithm can be applied to instances where there are a very large (or variable) number of products. Despite parallelisation, we handle the shared capacity constraints as follows.

4.1.1 Rewards

The key challenges with computation of individual elements of $\mathbf{u}$ are (1) ensuring that the system-level constraints (10)–(11) are met, and (2) that all products are treated fairly. Both challenges are partially addressed using the reward structure. The fairness issue is addressed using the percentile spread term in (12), since it penalises the agent if some products have low inventories, while others are at high levels. The volume and weight constraints are introduced as soft penalties in the following 'per-product' reward definition, adapted for individual decision-making.

$$\begin{aligned} R_i(t) =& 1 - b_{i,\text{empty}}(t) - b_{i,\text{critical}}(t) - q_{\text{waste},i}(t) \\ & - \Delta \mathbf{x}(t)_{.05}^{.95} - \Omega_i(t) - \alpha \max(\rho - 1, 0), \end{aligned} \tag{15}$$

where $b_{i,\text{empty}}(t)$ and $b_{i,\text{critical}}(t)$ are binary variables indicating whether inventory for product i is below 0 and χ_i, respectively, in the current time period, α is a constant parameter, and ρ is the ratio of total volume or weight requested by the RL agent to the available capacity. We formally define this as:

$$\rho = \max\left(\frac{\mathbf{v}^\intercal \mathbf{u}(t)}{v_{\max}}, \frac{\mathbf{c}^\intercal \mathbf{u}(t)}{c_{\max}}\right).$$

Equation (15) defines the reward that is actually returned to the RL agent, as opposed to the true system reward defined in (12). If the aggregate actions output by the agent (across all products) does not exceed the available capacity ($\rho \leq 1$), then the average value of (15) is equal to (12). This implies that maximising $R_i(t)$ is equivalent to maximising $R(t)$, as long as system constraints are not violated. The last two terms of (15) are common to all products at time t.

4.1.2 State and action space

Table 1 lists the features used for computing the replenishment quantity for each product. The first two features relate to the instantaneous state of the system with respect to product i. The next three inputs are product meta-data, relating to either long-term or constant behaviour. The shelf-life $T_s(i)$ is the normalised inverse of the *average inventory loss* for product i (the decrease in inventory not accounted for by orders, in a given time period). This is also computed empirically, and acts as an implicit estimator for the dynamics A.

Table 1 State space representation

Notation	Explanation
$x_i(t)$	Current inventory level
$\hat{W}_i(t)$	Forecast aggregate orders in $[t, t+1)$
v_i	Unit volume
c_i	Unit weight
$T_s(i)$	Shelf-life
$\mathbf{v}^\intercal \hat{\mathbf{W}}(t)$	Total volume of forecast for all products
$\mathbf{c}^\intercal \hat{\mathbf{W}}(t)$	Total weight of forecast for all products

The meta-data distinguish between different product characteristics when they are processed sequentially by the same RL agent, by mapping individual products to the same feature space. The last two features in Table 1 are derived features that provide indications of total demand on the system, with respect to the various constraints. These indicators act as inhibitors to the control action for product i, if the total demand on the system is high. They also help the agent correlate the last term in the observed rewards (the capacity exceedance penalty) with the state inputs. The output of the RL agent is $u_i(t)$, which is the desired action for product i at time t. Individual actions are concatenated to form $\mathbf{u}(t)$, according to the workflow shown in Fig. 4.

4.2 Deterministic lead time

We now proceed to define a RL formulation for solving the inventory management problem with variable lead times across products. In contrast to the single store multiple products inventory optimisation with no lag between taking decisions and inventory being updated, this section focuses on solving the problem with different products having deterministic lead times (time lag between the decisions taken and actual update of inventory). We model this as a multi-agent decision problem among products having different lead times in addition to the previous formulation of processing a single product in parallel. At each time step t and for each l_i we compute each $u_i(t)$ independently. We create a new RL agent corresponding to l_i, and each agent learns the different dynamics of that lead time. The agents still play independently from one another, except that in the state space, we give information about the total aggregate demand on the system due to other agents. The training schema is done using a stage-wise learning strategy in which the learning process is broken down into a number of related sub-tasks that are completed stage-by-stage. The advantage of this approach is that training the higher lead time products initially would fix the volume and weight constraints since the total volume is shared among all lead time products.

In order to satisfy the volume constraint, we propose a roll-out mechanism to predict the soft actions of lower lead time products while taking decisions for the higher lead time products. As shown in Fig. 5, if the number of lead times in the dataset is (3, 2, 1), we roll-forward the decisions for lead time 1 till the maximum lead time which is 3 in this example and freeze only the first actual decision u_{11}

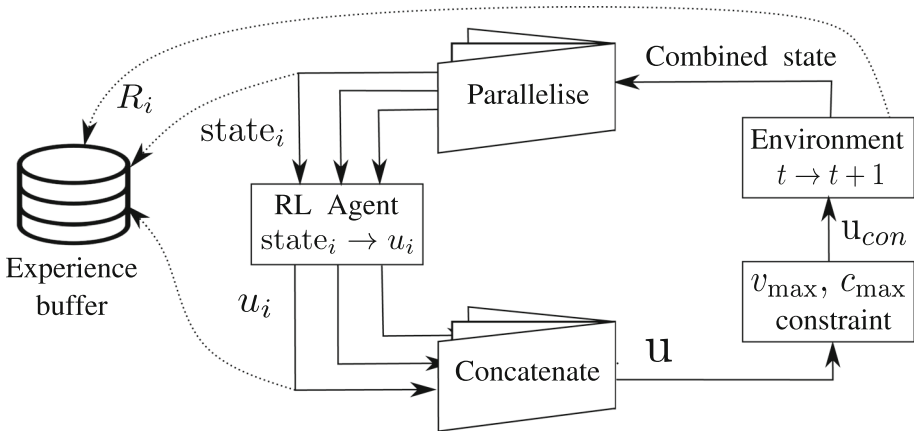

Fig. 4 Workflow of RL computation. We show only the flowchart when the RL agent is being used for decision-making. Depending on the algorithm, the policy and/or value networks are updated periodically by training on the stored data in the experience buffer

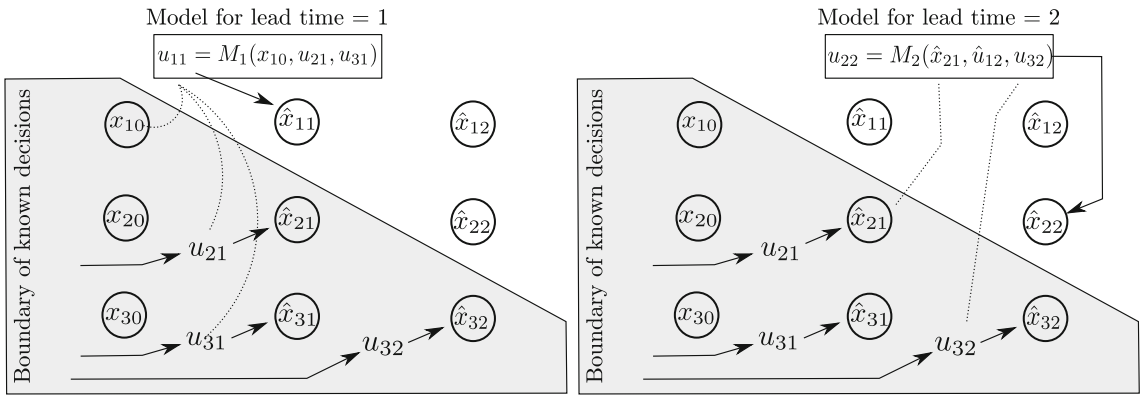

Fig. 5 Rollout mechanism to satisfy truck volume constraint. Here, variables with a hat are projections and the ones which are without a hat are fixed quantities. M is the model, x is the inventory level, u is the replenishment decision (hat for soft decision, no hat for fixed decision). In the subscript, the first digit is lead time and the second digit is days. On a given day 0, decisions within the shaded region have been computed on previous days. We first compute u_{11}, the fixed replenishment decision for day 1, for the product with a lead time of 1. This is followed by u_{22}, the fixed decision for day 2, product with lead time of 2

which is a function of $M_1(x_{10}, u_{21}, u_{31})$, while all other actions $\hat{u}_{12}$ and $\hat{u}_{13}$ are soft actions which helps in the decision making of higher lead time products 3 and 2. During the decision making if soft or the fixed actions are not present, then we use forecast sales information as the replenishment action, e.g. computation of $\hat{u}_{12}$. Similarly for the prediction of the lead time 2 products using model M_2, we use lead time 1 soft action $\hat{u}_{12}$ and the previous replenishment action u_{32} of lead time 3 , $u_{22} = M_2(\hat{x}_{21}, \hat{u}_{12}, u_{32})$ and freeze this action as shown in the schematic below. And action prediction for lead time 3 is done using M_3 model with the information of lower lead time soft actions $\hat{u_{13}}$ and $\hat{u_{23}}$, $u_{22} = M_3(\hat{x}_{32}, \hat{u}_{13}, \hat{u}_{23})$ and this process repeats for every delivery moment replacing the current delivery moment actions as the actual action to be implemented. The objective is to output the replenishment quantities such that the individual lead time products satisfy all the systems constraints as well as share the volume and weight capacity of trucks among the different lead time products at a particular time instance.

4.2.1 Rewards

The key challenges with computation of individual elements of u for a problem with variable lead time products are (1) ensuring that the system-level constraints (10)-(11) are met by not over-utilization of volume and weight by higher lead time products since higher lead time decisions are taken upfront, and (2) that all products are treated fairly. Both challenges are partially addressed using the reward structure described in (12). The fairness issue is addressed by the percentile spread term (denoted by $\Delta x_{l_i}(t + l_i + 1)_{.05}^{.95}$), on the variation in inventory levels across the product range, applied uniformly to all the l lead-time products. The volume and weight constraints are introduced as soft and hard capacity exceedance in the following 'per-product' reward definition, adapted for individual decision-making.

$$R_{i,l_i}(t+l_i+1) = 1 - b_{i,l_i,\text{empty}}(t+l_i+1) - b_{i,l_i,\text{critical}}(t+l_i+1) - q_{i,l_i,\text{waste}}(t+l_i+1) - e_{i,l_i}(t) - \Omega_{i,l_i}(t+l_i+1) - \max(\rho_l - 1) - \Delta x_l(t+l_i+1)_{.05}^{.95} \quad (16)$$

$\Omega_{i,l_i}(t+l_i+1)$ is the demand quantities that are refused in the current time period. Equation (16) defines the reward that is actually returned to the RL agent, as opposed to the true system reward defined in (12). ρ_l is the ratio of total volume or weight of all the current lead-time products requested by the agent to the available truck capacity. We formally define as,

$$\rho_l = \max\left(\frac{\mathbf{v}_l^T\,\mathbf{u}_l(t)}{v_{\text{avail}}}, \frac{\mathbf{c}_l^T\,\mathbf{u}_l(t)}{c_{\text{avail}}}\right).$$

$e_{i,l_i}(t)$ can be described as soft capacity exceedance penalty which gives the indication of other agent policies whether its aggressive or conservative and is defined as the ratio of total estimated volume to its total truck capacity.

$$e_{i,l_i} = \left(\frac{\sum_{i,l'\neq l}\eta_{i,l_i'} * v_{i,l_i'} + \sum_i u_{i,l_i} * v_{i,l_i}}{v_{\max}} - 1\right)$$

where,

$$\eta_{i,l_i'} = \begin{cases} u_{i,l_i'}, & l_i < l_i' \\ \hat{u}_{i,l_i'}, & \text{otherwise} \end{cases}$$

4.2.2 State and action space

Table 2 lists the features used for the element-wise computation of replenishment quantity for each product. The framework also allows us to compute replenishment quantities for each product independently, while receiving some global information in the form of aggregated demand of other lead time products on the system. Details about each of these features are as follows.

- The current inventory level $x_{i,l_i}(t)$ indicates the instantaneous state of the system with respect to each product for the current lead time l_i.
- The projected inventory is computed by simultaneously updating the current inventory $x_{i,l_i}(t)$ with the previous replenishment quantities and subtracting the store order forecast in the time interval $(t:(t+l_i+1))$.
- The forecast estimator, which is a trailing average of sales in the most recent T time periods, predicts the forecast estimate during the $((t+l_i):(t+l_i+1))$ time period for product i with lead time l.
- The capacity occupied by other agents for the day of replenishment indicates how much free space is available for the current set of decisions. They also help the agents to correlate the soft capacity exceedance penalty in the RL reward term with the state inputs.
- I_{empty} represents the binary indication of inventory going to stock-out.
- The last two terms are the products meta-data which help to distinguish between different product characteristics when they are processed sequentially by the same RL agent, by mapping individual products to the same feature space.

The output of the RL agent is $u_{i,l_i}(t+l_i+1)$, which is the desired action at time t for product i to be implemented at time $(t+l_i+1)$.

4.2.3 Training mechanism

For the computation of each individual decisions $u_{i,l_i}(t)$ for a given lead time l, we use separate RL agents. Information sharing among the agents is done with sharing other agents action information into the current state of the system. To address the non-stationarity challenge in this multi-agent reinforcement learning (MARL) scenario which usually occurs when multiple agents learn concurrently thus affecting the reward of other agents, and the evolution of the state, we employ stage-wise training of individual agents starting from higher lead time. Suppose we have 3

Table 2 State space representation for product i with lead-time l_i

Notation	Explanation
$x_{i,l_i}(t)$	Current inventory level
$\hat{x}_{i,l_i}(t+l_i+1)$	Projected inventory
$\hat{\mathbf{W}}_{i,l_i}((t+l_i):(t+l_i+1))$	Forecast at the current lead-time
$\sum_{l'\neq l}\mathbf{v}^\intercal\,\eta_{l'}$, where $\eta = u$ if $l<l'$ else $\hat{u}$	Capacity occupied by other agents
I_{empty}	Binary indicating inventory stock-out
v_i	Unit volume
w_i	Unit weight

separate lead times l_1, l_2 and l_3, with $l_1 > l_2 > l_3$ then, the training regime is as follows:

- Train the l_1 lead time model, while l_2 and l_3 lead time products using heuristics.
- Train the l_2 lead time model , with l_1 using trained policy and l_3 using heuristic
- Train the l_3 lead time model, with l_1 and l_2 using the trained policy

5 Experimental details

We now evaluate the proposed RL algorithm, along with other baselines, on the data derived from a public source [34]. We further provide a linear programming (LP)-based theoretical upper bound for any inventory control algorithm and later benchmark the RL algorithm against it.

5.1 Data for experiments

We use a public data set for brick and mortar stores [34] as the basis for the experimentation. The original data set includes purchase data for 50,000 product types and 60,000 unique customers. However, it does not contain meta-data about the products (dimensions, weight) and also does not specify date of purchase (although it specifies time of day and the day of week). Instead, it measures the number of days elapsed between successive purchases by each customer. We obtain data in the format required by the current work, by assigning a random date to the first order of each unique customer while respecting the day of week given in the original data set. This implicitly assigns specific date and time stamps to each purchase. Additionally, we assign dimension and weight to each product type based on the product description. Since this is a manual process, we only use two independent subsets of 220 and 100 products respectively from the full data set, from products listed under the 'grocery' category.

The data set thus formed spans a period of 349 days. We assume that stock replenishment happens 4 times a day, resulting in 1396 time periods. Of these, we use the first 900 time periods for training, and use the final 496 time periods for testing. The semi-synthetic data sets with 220 and 100 product types are used for comparison against other approaches. In both cases, the volume and weight constraints ($v_{\max}$ and $c_{\max}$) are set slightly lower than the average sales volume and weight, ensuring that constraints (10)–(11) are active. In addition, we randomly generate lead time for the product while taking care that lead times are not greater than their shelf-life.

As described in Sect. 4, we investigate the RL and baseline approach in two separate scenarios with no lead time and deterministic lead times. For our experiments, for both datasets, we consider 3 different lead times, 10, 6 and 2 time periods. For RL algorithms, we quantised the replenishment values for each product i in 14 discrete actions (0, 0.005, 0.01, 0.0125, 0.015, 0.0175, 0.02, 0.03, 0.04, 0.08, 0.12, 0.2, 0.5, 1). These discrete actions correspond to normalised replenishment quantities by maximum shelf capacity for that product. We train DQN and PPO in each experiment for no lead time and each phase in deterministic lead time for 500 episodes. ϵ was exponentially decayed (decay rate as 0.9942) for DQN and maintained at 0.01 value after 400 episodes, replay buffer size was kept at 2^4 with batch size as 32. Policy, value and entropy coefficients for PPO were set to 1, 0.5 and 0.01 respectively as default values from [57]. Adam was used as optimiser with AMSGrad [53] for both DQN and PPO with 0.001 as the learning rate. We observe that these algorithms converge with minor variance in 500 episodes. During each training episode, we generate random initial inventories for each product. These random initial inventories are used for all the experiments across algorithms (RL, Heuristic, LP) and scenarios (no lead time, deterministic lead time). We ran experiments for 10 and 5 random seeds for no lead time and deterministic lead time respectively for statistical significance.

5.2 Baseline algorithms

5.2.1 Heuristic based on proportional control

We use a modified version of s-policy [45], a standard heuristic in the literature which aims to maintain inventory at some predefined constant level. If we define $\mathbf{x}^*$ to be the *target* level of inventories for the products, the desired replenishment quantity is designed to satisfy forecast sales and reach $\mathbf{x}^*$ by the end of the time period. The expression is given by:

$$\mathbf{u}_{pr}(t) = \max\left[0, \mathbf{x}^* + \hat{\mathbf{W}}(t) - \mathbf{x}(t)^-\right]. \tag{17}$$

The desired action $\mathbf{u}_{pr}(t)$ according to (17) satisfies constraints (8) and (9), since $\mathbf{0} \leq \mathbf{x}^* \leq \mathbf{1}$. However, it is not guaranteed to satisfy the total volume and weight constraints (10) and (11). Therefore, the final control vector is computed by proportional reduction analogous to (4.2.1).

5.2.2 Optimal bounds using linear programming

Linear programs (LP) and its robust variants have long been used in the context of single-store, single-product inventory optimisation [11, 12, 39]. While LP approaches do not easily extend to more complex scenarios involving heterogeneous lead times and multi-product constraints,

they seem to provide optimal performance bounds if implemented successfully. To this end, we formulate an online LP with perfect information, i.e., the true or realised demands $\mathbf{W}$ are known apriori. It must be noted that having a perfect information on true demands does not necessitate special treatment for the multi-lead time scenario. Thus, we restrict ourselves to the simple case of zero lead time. As before, we use $\mathbf{x}(t)^-$, $\mathbf{u}(t)$, $\mathbf{v}$ and $\mathbf{c}$ to denote normalised inventory at the beginning of demand point t, normalised replenishment at demand point t, unit volume and unit weight multipliers for all products, respectively.

The LP aims to optimise a linear combination of various components that constitute the overall business reward described in (12). Additionally, the LP objective also includes components that linearise *maximum* and *minimum* constraints. Among other objectives, the LP must minimise the total demand lost at any time due to insufficient inventory. The demand lost at time t is defined as:

$$\text{Demand lost at time } t = \max\left(0, \mathbf{W}(t) - \mathbf{x}(t)^- - \mathbf{u}(t)\right). \tag{18}$$

The maximum constraints can be linearised using an additional LP variable, $\mathbf{D}(t)$ and the following set of constraints:

$$\begin{aligned} &\mathbf{D}(t) \geq 0, \\ &\mathbf{D}(t) \geq \mathbf{W}(t) - \mathbf{x}(t)^- - \mathbf{u}(t). \end{aligned} \tag{19}$$

One possible approach to minimise the lost demand is to replenish quantities in bulk subject to truck constraints and shelf capacity constraints. While this approach may work for products with relatively large shelf-lives, shelf-life constraints require the replenishment to meet the expected demands closely in order to avoid wastage. Therefore, we impose additional constraints on replenishment in order to keep the inventory levels below the wastage level, i.e.,

$$\mathbf{u}(t) \leq \text{Wastage-level} - \mathbf{x}(t)^- + \mathbf{W}(t), \tag{20}$$

where Wastage-level is the ideal inventory level to be targeted. Finally, for each product i that gets replenished at time t, it is desired that the product gets consumed within the next $T_s(i)$ time instants, where $T_s(i)$ denotes the shelf-life of the ith product. Assuming a first-in first-out (FIFO) roll-out model, this can be achieved using the following set of constraints,

$$\begin{aligned} &\mathbf{q}(t) \geq 0, \\ &\mathbf{q}_i(t) \geq \mathbf{u}_i(t) - \sum_{\tau=t}^{t+T_s(i)} \mathbf{W}_i(\tau), \quad \forall i \in \{1, 2, \ldots, p\} \end{aligned} \tag{21}$$

where $\mathbf{q}(t)$ needs to be minimised. Finally, we aim to maintain critical inventory levels in order to ensure a minimum presentation level for each product, i.e., we introduce additional set of constraints given by:

$$\mathbf{G}(t) \leq \chi - (\mathbf{x}(t)^- + \mathbf{W}(t)), \tag{22}$$

where χ denotes the acceptable critical inventory level. We are now ready to formulate the LP as:

$$\begin{aligned} \max_{\mathbf{u}(t), \mathbf{D}(t), \mathbf{q}(t), \mathbf{G}(t)} \quad & \mathbf{u}(t) - \beta_1 \mathbf{D}(t) - \beta_2 \mathbf{q}(t) - \beta_3 \mathbf{G}(t) - \beta_4 \Delta \mathbf{x}(t)^{.95}_{.05}, \\ \text{subject to,} \quad & (8) - (11), (19) - (22) \text{ hold.} \end{aligned}$$

Here $\beta_1, \beta_2, \beta_3, \beta_4 > 0$ denote the relative weights for different cost components.

6 Results and discussion

In this section, we evaluate RL algorithms against the baselines as mentioned in the previous section. We present results during the training phase for RL and heuristic and evaluate and compare it with baselines on the testing dataset. We evaluate all the algorithms over the business reward that is defined in the problem formulation.

6.1 No lead time

It can be observed from Fig. 6 that both DQN and PPO converge to a similar value for the average business reward on the 100 products dataset, and significantly outperform the proportional control based heuristic. Interestingly, DQN marginally outperforms PPO on the 220 products dataset, as seen in Fig. 7. Possible reasons for difference in performance between two RL algorithms across datasets are further investigated in Sect. 6.3.

Table 3 shows performance over business reward on testing samples for both the datasets. To understand the robustness of the trained RL algorithms for out of distribution states, we also investigate the performance on a dataset on which it is not trained. We note that the 100 product and the 220 product datasets are completely different and there is no overlap between these two. We can observe that both of these algorithms are able to beat the heuristic even when they are trained on different datasets and provide us with a soft signal about RL algorithms learning the environment dynamics well. The LP-based upper bound provides us with theoretical upper bounds for a given environment as it uses the perfect information to generate actions. We observe that DQN is able to achieve 89% and 94% whereas 85% and 84% of the theoretical upper bound for the 100 and the 220 datasets, respectively.

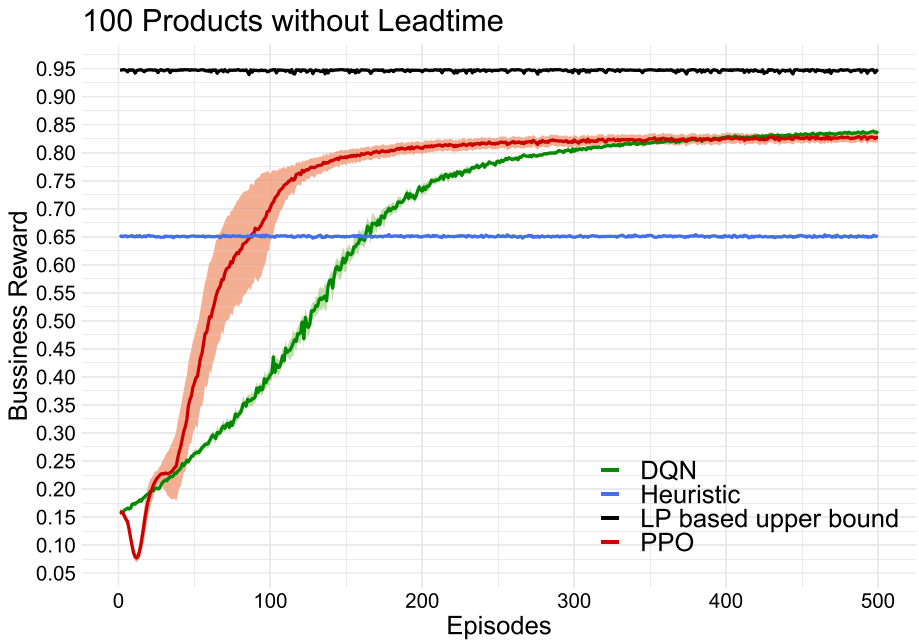

Fig. 6 Business reward vs training episodes for 100 product dataset with no lead times. Solid line represents mean across 10 random seeds and shaded area depicts the 95% confidence interval

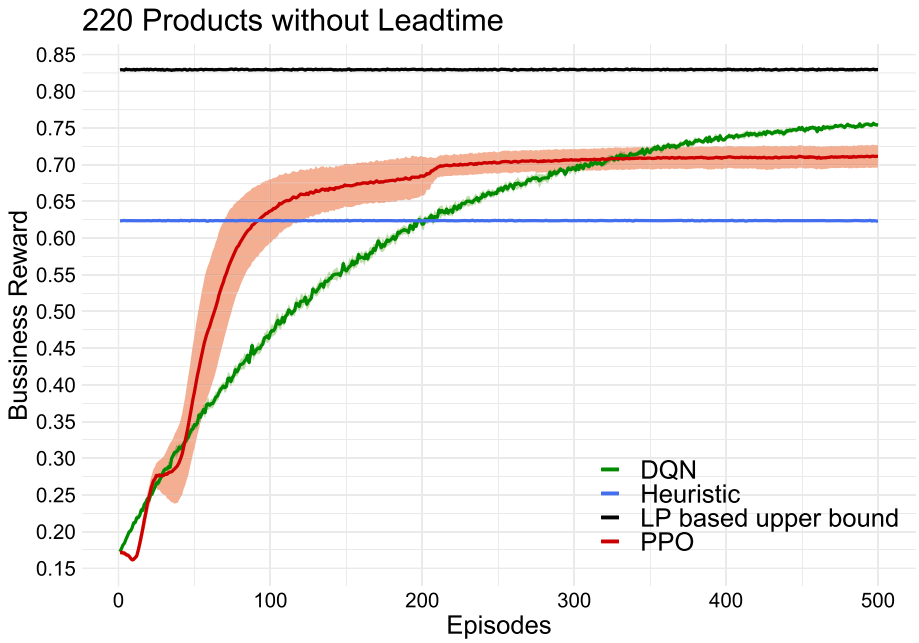

Fig. 7 Business reward vs training episodes for 220 product dataset with no lead times. Solid line represents mean across 10 random seeds and shaded area depicts the 95% confidence interval

6.2 Deterministic lead time

In Figs. 8 and 9, we present the results of training DQN and PPO for 100 and 220 datasets. The training process consists of three phases as mentioned previously. The first phase comprises of training the highest lead time agent (10 in our case), while the other two lead time policies are guided by the heuristic. The lower lead time models are subsequently trained, and used in conjunction with already trained agents for higher lead times. The performance improves progressively as different lead time agents get trained. It is also observed that both of these algorithms significantly outperform the heuristic.

Table 4 shows the training and testing results with business reward as the metric after the end of phase 3. We also assess the robustness of RL trained agent to perform in out of distribution states and observe that PPO performs better when compared to DQN, which is the opposite of what we observed in the no lead time case. One of the possible reasons for PPO dominating in the robustness claim is as compared to earlier, combined training epochs

Table 3 Training and test performance of with no lead time in products

Algorithm	Training	Testing
100 products		
DQN	0.837	0.857
PPO	0.827	0.829
Heuristic	0.651	0.647
DQN (trained on 220 products)	–	0.831
PPO (trained on 220 products)	–	0.771
LP based upper bound	0.948	0.952
220 products		
DQN	0.754	0.760
PPO	0.711	0.704
Heuristic	0.62	0.608
DQN (trained on 100 products)	–	0.694
PPO (trained on 100 products)	–	0.634
LP-based upper bound	0.829	0.808

across phases are 1500 as compared to 500 and hence able to learn better on environment dynamics.

6.3 Ablation studies

In order to better understand the policies that RL algorithms have learned, we run ablation studies over policies to deconstruct their actions.

6.3.1 Policy heatmaps

In Figs. 10 and 11, we plot heatmaps for policy actions on the 100 and the 220 datasets, respectively. We query the RL agent for a given forecast and current inventory value and use mean of actions (replenishment quantity) to generate heatmaps. These heatmaps are generated for the no lead time scenario. We can observe that DQN is much more aggressive in its policy as there are higher replenishment quantities on an average where inventory is below 0.2 (which is stock out level χ). Where forecasts are relatively high, it asks for high replenishment even when current inventory levels are comfortable (right half of the plot). This is probably due to high forecasts indicating a higher variance in demand in the training data, which can lead to stock-outs if the demand spike is high enough. The same behaviour is not seen in Fig. 11, because the 220-product case must share capacity among a larger number of products, removing the option to replenish speculatively.

6.3.2 Testing no lead time models on the deterministic lead time scenario

We further test the already trained RL models (trained on no lead time scenario) on deterministic lead time scenarios for any degradation in performance. Figure 12 shows the performance over 496 delivery moments in testing data for 100 products dataset. We can observe that PPO and DQN which are trained with deterministic lead times are able to maintain the same level of business reward as the episode

Fig. 8 Training results for 100 product dataset with 10, 6 and 2 lead time. Solid line represents mean across 5 random seeds and shaded area depicts the 95% confidence interval

Fig. 9 Training results for 220 product dataset with 10, 6 and 2 lead time. Solid line represents mean across 5 random seeds and shaded area depicts the 95% confidence interval

Table 4 Training and test performance of with deterministic lead time products

Algorithm	Training	Testing
100 products		
DQN	0.803	0.809
PPO	0.828	0.826
Heuristic	0.516	0.510
DQN (trained on 220 products)	–	0.572
PPO (trained on 220 products)	–	0.747
LP based upper bound	0.948	0.952
220 products		
DQN	0.714	0.709
PPO	0.722	0.722
Heuristic	0.466	0.466
DQN (trained on 100 products)	–	0.561
PPO (trained on 100 products)	–	0.636
LP based upper bound	0.829	0.808

progress, whereas both DQN and PPO which are trained with no lead time scenario drops down significantly from initial point specifically DQN. We further investigate this case by observing degradation in different lead times are presented in Table 5, we observe that there is a significant drop in performance for products with 2 lead times as compared to PPO. This is due to aggressive nature of DQN as we observed in Fig. 10 and 11 which leaves no truck volume and weight capacity for lower lead time products. We calculated the difference in replenishment quantity before and after truck capacity fulfilment and we see that on an average there is a reduction of 0.1 quantities across the products for DQN, whereas for PPO, it is 0.001 which reinforces our hypothesis.

6.3.3 How does correlation between actual orders and forecast affect performance?

The performance of any unbiased approach for optimal inventory control must correlate positively with the accuracy of demand forecast. LP-based upper bound provides theoretical upper bound under the assumption that the forecasted demand is perfectly correlated with the true demand. We further investigate this relationship between forecasted demand with true demand where, both for the baseline heuristic, as well as the DQN algorithm under varying accuracy of the forecasted demand. The correlation (or accuracy) of the demand forecast with respect to the true demand is gradually increased, to an extent that the forecasted demand is perfectly correlated with the true demand in the extreme case. Figure 13 depicts the business rewards over different sets of correlations generated by adding normal noise to actual orders. We can observe that when forecast is equal to actual orders (correlation = 1), RL does seem to approach LP performance. Besides, RL is also able to handle low correlated forecasts better as compared to heuristics. Moreover, the gain in business reward is monotonic in the correlation, indicating that there are no potential artifacts in the learning process. The performance of the DQN algorithm is a function of the demand forecast, and the algorithm is not biased towards working in only one (low or high) correlation regime.

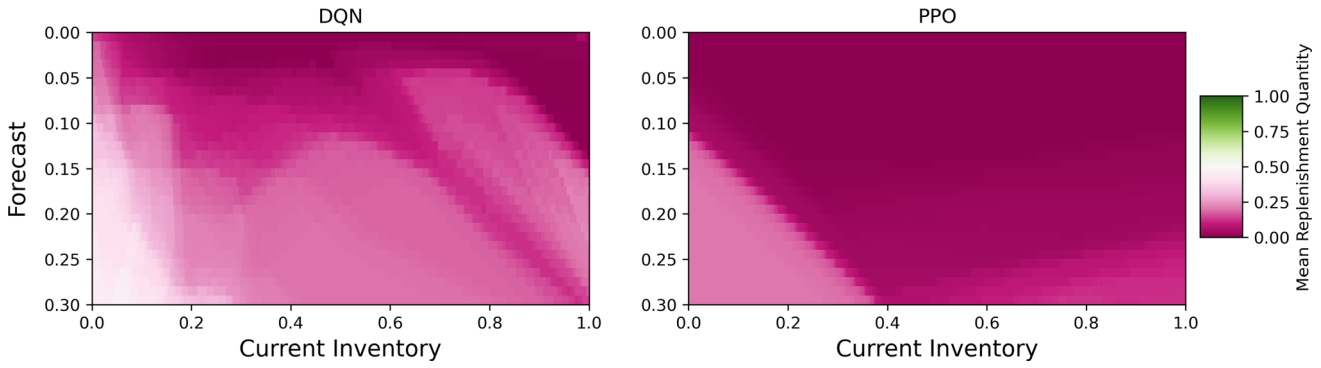

Fig. 10 Policy heatmap for 100 product dataset

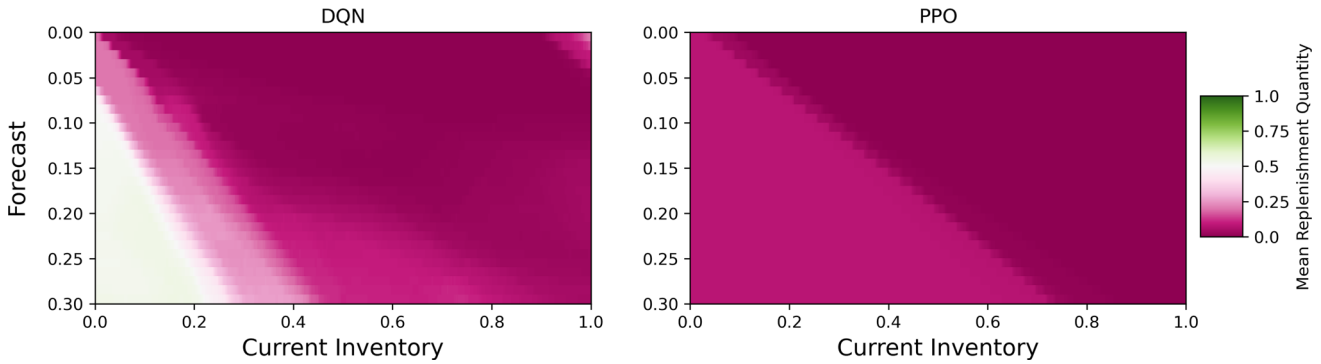

Fig. 11 Policy heatmap for 220 product dataset

Fig. 12 Testing no lead time trained models on with lead time scenario

6.3.4 Product wise rewards

Given the global truck constraints and shelf-capacity constraints, it is likely that products with very large demands may fall short on availability, thereby incurring a decline in average reward. Figures 14 and 15 depict product-wise average business rewards for the 100 and the 220 product dataset. Products on x-axis are sorted in descending order of average business reward. Note that the product-specific average business rewards are not very dissimilar, indicating that the algorithm does a good job at maintaining fairness across multiple products. Having said that, it is a property of the datasets where certain products have very high actual orders sometimes higher than shelf capacities of the stores, specifically seen in the 220 product dataset.

Table 5 Comparison between RL models learned with lead times and no lead times over lead time scenario

	DQN			PPO		
	10 lead time	6 lead time	2 lead time	10 lead time	6 lead time	2 lead time
100 Prod						
With lead time	0.745	0.822	0.839	0.833	0.825	0.823
No lead time	0.441	0.412	− 0.110	0.194	0.263	0.489
220 Prod						
With lead time	0.650	0.753	0.715	0.649	0.756	0.741
No lead time	0.366	0.329	− 0.116	0.466	0.597	0.627

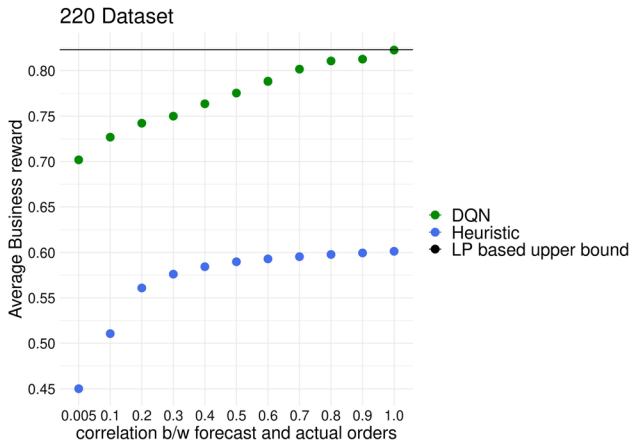

Fig. 13 X-axis denotes correlation between actual orders and forecast, Y-axis denotes the business rewards after training

6.3.5 Performance of RL algorithms across varying average orders

Scatter plot between orders and business rewards provides significant evidence that DQN is able to handle products with higher orders much better as compared to PPO as shown in Fig. 16. In addition, we can observe that the 220 product dataset has significant higher average orders for some products when compared to the 100 product dataset. The reason for performance variations between DQN and PPO across different dataset is a function of type of products and not the number of products.

6.3.6 Fairness across products

The fairness across product types is a critical concern with the retailers. We trained an DQN agent omitting percentile term from its reward (DQN_wpr). The percentile term in the business reward structure indeed captures the fairness. This is reflected in the performance of the RL algorithm in Fig. 17. For brevity, only the DQN algorithm is examined for fairness. The red and cyan curves in Fig. 17b indicate the performance of the DQN algorithm with and without the fairness terms, respectively. It can be seen that the normalized replenishment quantity is consistent across products when the fairness term is included in the model.

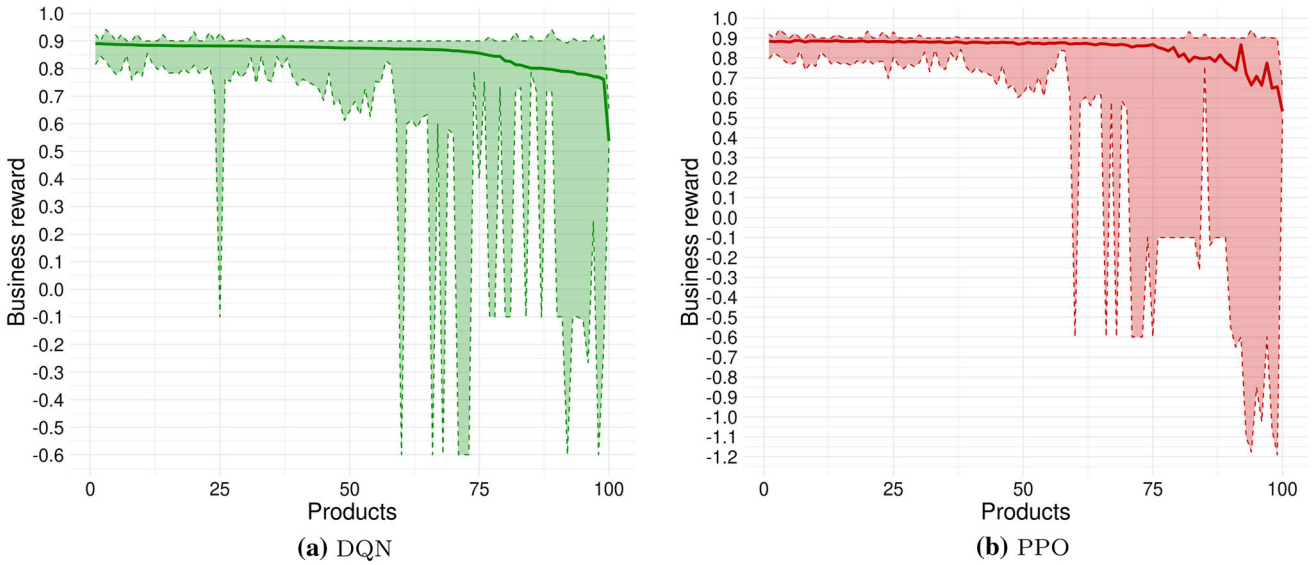

Fig. 14 Product wise mean (solid line), min–max (shaded region) of business reward for 100 product dataset. Products on X-axis are sorted in decreasing orders of Business reward

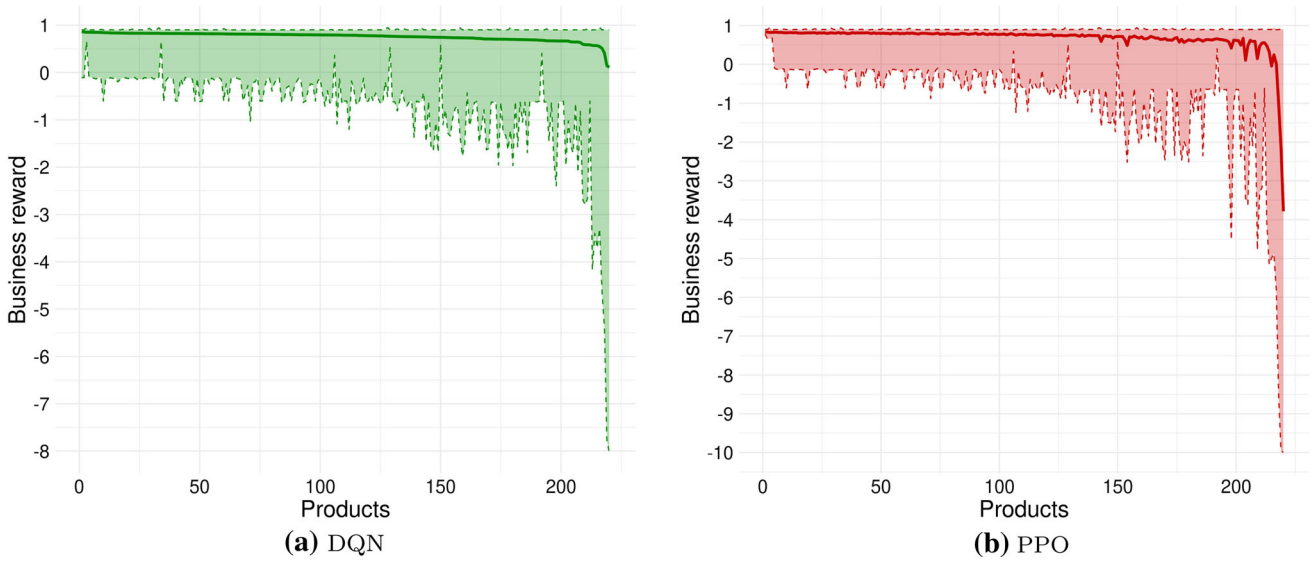

Fig. 15 Product wise mean (solid line), min–max (shaded region) of business reward for 220 product dataset. Products on X-axis are sorted in decreasing orders of Business reward

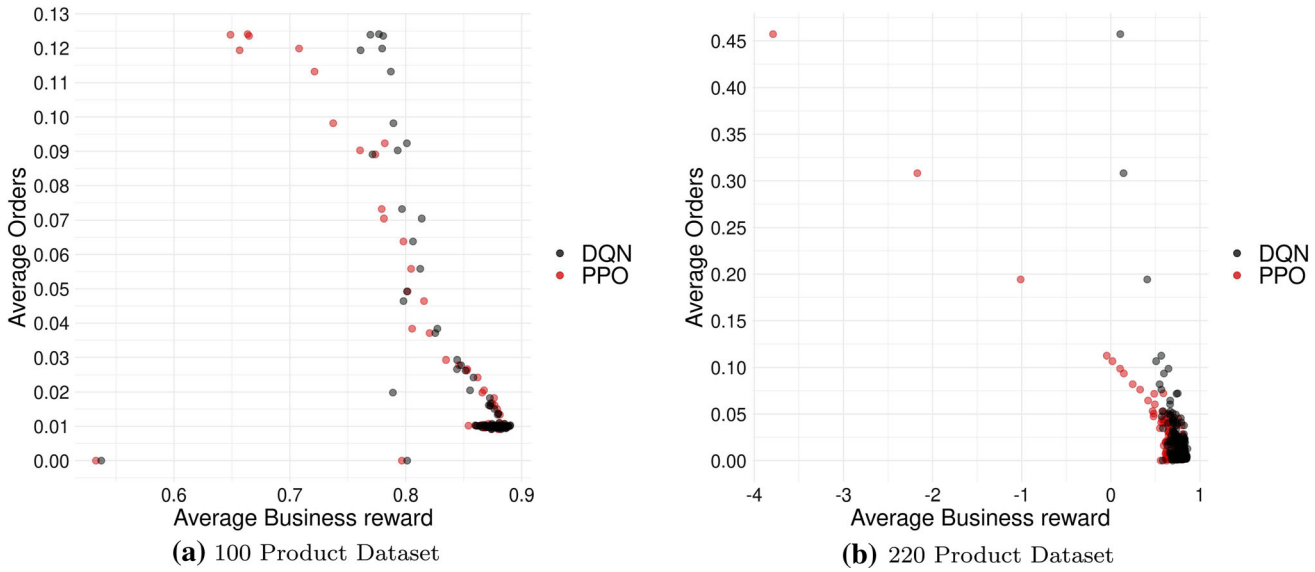

Fig. 16 Scatter plot showing performance of DQN and PPO as a function of average orders

Moreover, the contribution of individual product towards the average business reward is consistent with the inclusion of fairness term (Fig. 17a).

6.3.7 Reinforcement learning over linear programming

Finally, we describe the difficulty in using linear programming itself as the algorithm for replenishment, instead of reinforcement learning. The first challenge is the assumption that the true demand is known (as assumed in the benchmark results). Once this assumption is relaxed and we need to use forecasts, the performance of LP reduces. Secondly, when we introduce lead times and noisy forecasts at the same time, the complexity of the LP becomes several times greater (it needs to be converted into a robust optimisation framework). The third and final challenge is the computational time requirements of LP in comparison with RL. Figure 18 shows the time required to process replenishment quantity decisions for a single delivery moment for RL and LP. The x-axis represents the number of products and y-axis represents time in seconds in a log scale. We observe that there is order of magnitude difference in time complexity between the two algorithms at inference. Since real-world retailers have product lists in the range of 10^5–10^6 items, LP will quickly becomes

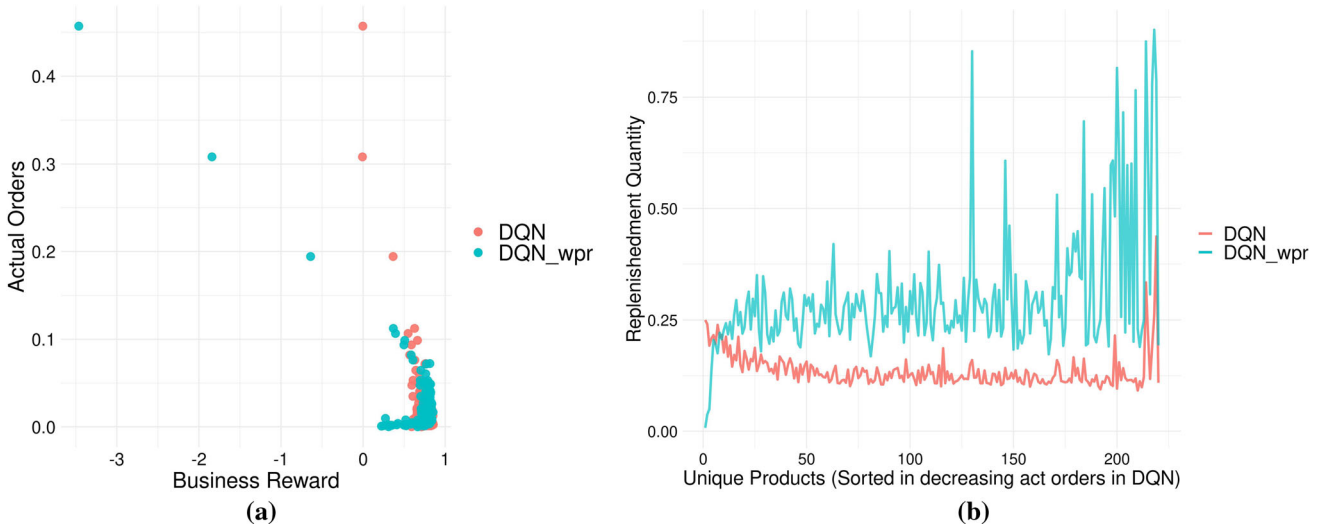

Fig. 17 **a** Scatter plot showing performance of DQN and DQN_wpr (without percentile reward) as a function of average orders. **b** X-axis: Products are sorted in decreasing order of actual orders (color figure online)

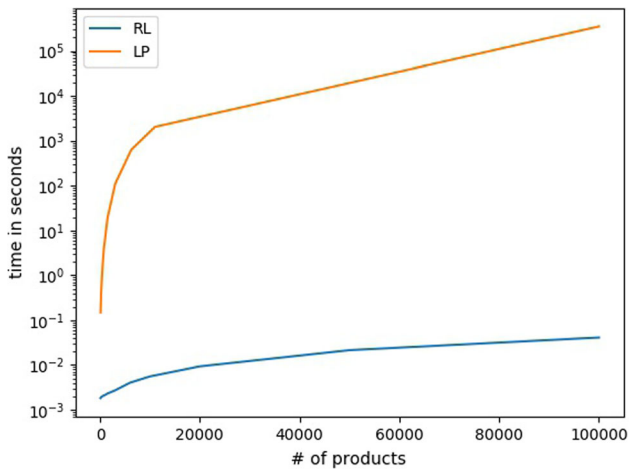

Fig. 18 Computational time requirements for RL and LP

intractable, even with perfect information. By contrast, the RL solution is parallelisable and therefore much more scalable.

7 Conclusion

The key takeaway from the results is that while each RL approach has its advantages and disadvantages (stability, ease of training), the parallelised decision-making framework is able to yield high solution quality with low online computational cost, and is also able to learn to stay within the aggregate system capacity constraints. Given the broader class of problems from which we derived the multi-product inventory management scenario, we wish to identify problems in related areas that can be solved using a similar RL approach. In addition, we also wish to extend the current store inventory management capabilities with a multi-agent approach for supply chain decision making, encompassing warehouse and transportation management.

Funding None.

Declarations

Conflicts of interests The authors declare that they have no conflict of interest.

References

1. Abdelmaguid TF, Dessouky MM (2006) A genetic algorithm approach to the integrated inventory-distribution problem. Int J Prod Res 44(21):4445–4464
2. Aharon BT, Boaz G, Shimrit S (2009) Robust multi-echelon multi-period inventory control. Eur J Oper Res 199(3):922–935
3. Akbari AA, Karimi B (2015) A new robust optimization approach for integrated multi-echelon, multi-product, multi-period supply chain network design under process uncertainty. Int J Adv Manuf Technol 79(1–4):229–244
4. Åström KJ, Wittenmark B (2013) Adaptive control. Courier Corporation
5. Baniwal V, Kayal C, Shah D, Ma P, Khadilkar H (2019) An imitation learning approach for computing anticipatory picking decisions in retail distribution centres. In: 2019 American control conference (ACC). IEEE, pp 4186–4191
6. Barat S, Khadilkar H, Meisheri H, Kulkarni V, Baniwal V, Kumar P, Gajrani M (2019) Actor based simulation for closed loop control of supply chain using reinforcement learning. In: Proceedings of the 18th international conference on autonomous agents and multiAgent systems. International Foundation for Autonomous Agents and Multiagent Systems, pp 1802–1804
7. Barlas Y, Gunduz B (2011) Demand forecasting and sharing strategies to reduce fluctuations and the bullwhip effect in supply chains. J Oper Res Soc 62(3):458–473

8. Bertsekas DP (2005) Dynamic programming and optimal control, chap 6, vol 1. Athena scientific Belmont, MA
9. Bertsekas DP, Tsitsiklis JN (1995) Neuro-dynamic programming: an overview. In: Proceedings of 1995 34th IEEE conference on decision and control, vol. 1. IEEE, pp 560–564
10. Bertsimas D, Georghiou A (2015) Design of near optimal decision rules in multistage adaptive mixed-integer optimization. Oper Res 63(3):610–627
11. Bertsimas D, Thiele A (2004) A robust optimization approach to supply chain management. In: International conference on integer programming and combinatorial optimization. Springer, pp 86–100
12. Bertsimas D, Thiele A (2006) A robust optimization approach to inventory theory. Oper Res 54(1):150–168
13. Bouabdallah S, Noth A, Siegwart R (2004) Pid vs lq control techniques applied to an indoor micro quadrotor. In: Proceedings of The IEEE international conference on intelligent robots and systems (IROS). IEEE, pp 2451–2456
14. Cachon G, Fisher M (1997) Campbell soup's continuous replenishment program: evaluation and enhanced inventory decision rules. Prod Oper Manage 6(3):266–276
15. Camacho EF, Alba CB (2013) Model predictive control. Springer Science & Business Media
16. Carbonneau R, Laframboise K, Vahidov R (2008) Application of machine learning techniques for supply chain demand forecasting. Eur J Oper Res 184(3):1140–1154
17. Cárdenas-Barrón LE, Treviño-Garza G (2014) An optimal solution to a three echelon supply chain network with multi-product and multi-period. Appl Math Modell 38(5–6):1911–1918
18. Caro F, Gallien J (2010) Inventory management of a fast-fashion retail network. Oper Res 58(2):257–273
19. Clark AJ, Scarf H (1960) Optimal policies for a multi-echelon inventory problem. Manage Sci 6(4):475–490
20. Coelho LC, Laporte G (2014) Optimal joint replenishment, delivery and inventory management policies for perishable products. Comput Oper Res 47:42–52
21. Condea C, Thiesse F, Fleisch E (2012) Rfid-enabled shelf replenishment with backroom monitoring in retail stores. Decis Supp Syst 52(4):839–849
22. Doyle JC, Glover K, Khargonekar PP, Francis BA (1989) State-space solutions to standard h/sub 2/and h/sub infinity/control problems. IEEE Trans Autom Control 34(8):831–847
23. Duan Y, Andrychowicz M, Stadie B, Ho J, Schneider J, Sutskever I, Abbeel P, Zaremba W (2017) One-shot imitation learning. In: NIPS 31
24. Fernie J, Sparks L (2018) Logistics and retail management: emerging issues and new challenges in the retail supply chain. Kogan page publishers
25. Giannoccaro I, Pontrandolfo P (2002) Inventory management in supply chains: a reinforcement learning approach. Int J Prod Econ 78(2):153–161
26. Godfrey GA, Powell WB (2002) An adaptive dynamic programming algorithm for dynamic fleet management, I: single period travel times. Transp Sci 36(1):21–39
27. Golnaraghi MF, Kuo BC (2017) Automatic control systems. McGraw-Hill Education
28. Gustafsson TK, Waller KV (1983) Dynamic modeling and reaction invariant control of ph. Chem Eng Sci 38(3):389–398
29. Harifi S, Khalilian M, Mohammadzadeh J, Ebrahimnejad S (2020) Optimization in solving inventory control problem using nature inspired emperor penguins colony algorithm. J Intell Manuf 1–15
30. Harmer J, Gisslen L, del Val J, Holst H, Bergdahl J, Olsson T, Sjoo K, Nordin M (2018) Imitation learning with concurrent actions in 3d games. arXiv preprint arXiv:1803.05402
31. Hofmann E, Rutschmann E (2018) Big data analytics and demand forecasting in supply chains: a conceptual analysis. Int J Logist Manage
32. Ioannou PA, Sun J (1996) Robust adaptive control, vol 1. PTR Prentice-Hall Upper Saddle River, NJ
33. Jiang C, Sheng Z (2009) Case-based reinforcement learning for dynamic inventory control in a multi-agent supply-chain system. Exp Syst Appl 36(3):6520–6526
34. Kaggle: Instacart market basket analysis data. https://www.kaggle.com/c/instacart-market-basket-analysis/data. Accessed Aug 2018
35. Kara A, Dogan I (2018) Reinforcement learning approaches for specifying ordering policies of perishable inventory systems. Exp Syst Appl 91:150–158
36. Khadilkar H (2019) A scalable reinforcement learning algorithm for scheduling railway lines. IEEE Trans Intell Transp Syst 20(2):727–736
37. Konda V, Tsitsiklis J (2000) Actor-critic algorithms. In: Advances in neural information processing systems, pp 1008–1014
38. Kushner HJ, Clark DS (2012) Stochastic approximation methods for constrained and unconstrained systems, vol 26. Springer Science & Business Media
39. Lee H, Pinto JM, Grossmann IE, Park S (1996) Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management. Indus Eng Chem Res 35(5):1630–1641
40. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2015) Continuous control with deep reinforcement learning. CoRR arxiv:abs/1509.02971
41. Liu XJ, Chan C (2006) Neuro-fuzzy generalized predictive control of boiler steam temperature. IEEE Trans Energy Con 21(4):900–908
42. Mayne DQ, Rawlings JB, Rao CV, Scokaert PO (2000) Constrained model predictive control: stability and optimality. Automatica 36(6):789–814
43. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al (2015) Human-level control through deep RL. Nature 518(7540):529
44. Mousavi SM, Hajipour V, Niaki STA, Aalikar N (2014) A multi-product multi-period inventory control problem under inflation and discount: a parameter-tuned particle swarm optimization algorithm. Int J Adv Manuf Technol 70(9–12):1739–1756
45. Nahmias S, Smith SA (1994) Optimizing inventory levels in a two-echelon retailer system with partial lost sales. Manage Sci 40(5):582–596
46. Ng AY, Coates A, Diel M, Ganapathi V, Schulte J, Tse B, Berger E, Liang E (2006) Autonomous inverted helicopter flight via reinforcement learning. In: Experimental robotics IX. Springer, pp 363–372
47. Ogata K, Yang Y (2002) Modern control engineering, vol 4. Prentice Hall
48. Papadaki KP, Powell WB (2003) An adaptive dynamic programming algorithm for a stochastic multiproduct batch dispatch problem. Naval Res Logist 50(7):742–769
49. Powell WB (2007) Approximate dynamic programming: solving the curses of dimensionality, vol 703. John Wiley & Sons
50. Proth JM, Sauer N, Wardi Y, Xie X (1996) Marking optimization of stochastic timed event graphs using ipa. Disc Event Dyn Syst 6(3):221–239
51. Qiu R, Shang J (2014) Robust optimisation for risk-averse multi-period inventory decision with partial demand distribution information. Int J Prod Res 52(24):7472–7495
52. Radhakrishnan P, Prasad V, Gopalan M (2009) Inventory optimization in supply chain management using genetic algorithm. Int J Comput Sci Netw Sec 9(1):33–40

53. Reddi SJ, Kale S, Kumar S (2019) On the convergence of adam and beyond. arXiv preprint arXiv:1904.09237
54. Ross S, Bagnell JA (2010) Efficient reductions for imitation learning. In: Proceedings of the international conference artificial intelligence and statistics (AISTATS)
55. Schaal S (1999) Is imitation learning the route to humanoid robots? Trends Cogn Sci 3(6):233–242
56. Schulman J, Levine S, Abbeel P, Jordan M, Moritz P (2015) Trust region policy optimization. In: International conference on machine learning, pp 1889–1897
57. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347
58. See CT, Sim M (2010) Robust approximation to multiperiod inventory management. Oper Res 58(3):583–594
59. Shaabani H, Kamalabadi IN (2016) An efficient population-based simulated annealing algorithm for the multi-product multi-retailer perishable inventory routing problem. Comput Indus Eng 99:189–201
60. Shah D (2020) The six aces to thrive in supply chain 4.0. https://www.tcs.com/blogs/six-aces-to-thrive-in-supply-chain-4-0
61. Shalev-Shwartz S, Shammah S, Shashua A (2016) Safe, multi-agent, reinforcement learning for autonomous driving. arXiv preprint arXiv:1610.03295
62. Shervais S (2000) Adaptive critic design of control policies for a multi-echelon inventory system. Ph.D. thesis, Portland State University
63. Shervais S, Shannon TT, Lendaris GG (2003) Intelligent supply chain management using adaptive critic learning. IEEE Trans Syst Man Cybern Part A Syst Humans 33(2):235–244
64. Si J, Barto AG, Powell WB, Wunsch D (2004) Handbook of learning and approximate dynamic programming, vol 2. John Wiley & Sons
65. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M et al (2016) Mastering the game of go with deep neural networks and tree search. Nature 529(7587):484
66. Silver EA (1979) A simple inventory replenishment decision rule for a linear trend in demand. J Oper Res Soc 30(1):71–75
67. Silver EA (1981) Operations research in inventory management: a review and critique. Oper Res 29(4):628–645
68. Smith SA, Agrawal N (2000) Management of multi-item retail inventory systems with demand substitution. Oper Res 48(1):50–64
69. Song JS, van Houtum GJ, Van Mieghem JA (2020) Capacity and inventory management: Review, trends, and projections. Manuf Serv Oper Manage 22(1):36–46
70. Tavakoli A, Pardo F, Kormushev P (2018) Action branching architectures for deep reinforcement learning. In: Thirty-second AAAI conference on artificial intelligence
71. Topaloglu H, Powell W (2006) Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems. INFORMS J Comput 18(1):31–42
72. Utkin V, Guldner J, Shi J (2009) Sliding mode control in electro-mechanical systems. CRC Press
73. Van Hasselt H, Guez A, Silver D (2016) Deep reinforcement learning with double q-learning. In: AAAI, vol 2. Phoenix, AZ, p 5
74. Van Roy B, Bertsekas DP, Lee Y, Tsitsiklis JN (1997) A neuro-dynamic programming approach to retailer inventory management. In: Proceedings of the 36th IEEE conference on decision and control, vol 4, pp 4052–4057. https://doi.org/10.1109/CDC.1997.652501
75. Verma R, Saikia S, Khadilkar H, Agarwal P, Srinivasan A, Shroff G (2019) An RL framework for container selection and ship load sequencing in ports. In: International conference on autonomous agents and multi agent systems
76. Visentin A, Prestwich S, Rossi R, Tarim SA (2021) Computing optimal (r, s, s) policy parameters by a hybrid of branch-and-bound and stochastic dynamic programming. Eur J Oper Res
77. Yang L, Li H, Campbell JF, Sweeney DC (2017) Integrated multi-period dynamic inventory classification and control. Int J Prod Econ 189:86–96
78. Zhang W, Dietterich T (1995) A reinforcement learning approach to job-shop scheduling. In: International joint conference on artificial intelligence. Montreal, Canada
79. Zipkin P (2000) Foundations of inventory management. McGraw-Hill Companies, Incorporated. https://books.google.co.in/books?id=rjzbkQEACAAJ