Using Reinforcement Learning for a Large Variable-Dimensional Inventory Management Problem

Hardik Meisheri, Vinita Baniwal,
Nazneen N Sultana, Harshad Khadilkar
TCS Research
Mumbai, India
[hardik.meisheri,vinita.baniwal,nn.sultana,harshad.khadilkar]@tcs.com

Balaraman Ravindran
IIT Madras
Chennai, India
ravi@cse.iitm.ac.in

ABSTRACT

This paper evaluates the applicability of reinforcement learning (RL) to multi-product inventory management in supply chains. The novelty of this problem with respect to supply chain literature is (i) we consider concurrent inventory management of a large number (hundreds) of products under realistic constraints such as shared capacity, and (ii) the number of products (size of the problem) can change frequently, implying that the RL agent needs to work in this regime without retraining. We approach the 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 reinforcement learning framework that can be used for parallelised decision-making, and use the advantage actor critic (A2C) and deep Q-network (DQN) algorithms with quantised action spaces to solve the problem. Experiments on scales between 100 and 220 products show that these approaches perform better than other baseline algorithms. They are also able to transfer learning without retraining, when the number of products change.

KEYWORDS

Reinforcement Learning; Supply Chain; Inventory Control

1 INTRODUCTION

Reinforcement Learning (RL) algorithms have been successfully applied to a broad range of problems in literature, including virtual gameplay [30, 47], physical systems such as autonomous driving [44] and flight dynamics [36], and problems from the field of operations research [22, 55, 56]. An essential feature of these applications is the ability to plan strategies that maximize 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 [43].

Motivating example: In this paper, we demonstrate the application of RL to a multi-product two-echelon retail inventory management scenario, illustrated in Figure 1. 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 tradeoffs 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 Figure 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 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.

Modelling assumptions: Real-world operations in a retail supply chain are typically highly dynamic [13] , with ad-hoc coordination between the warehouse, store, and logistics/transportation providers. In order to model the problem, we make some simplifying assumptions.

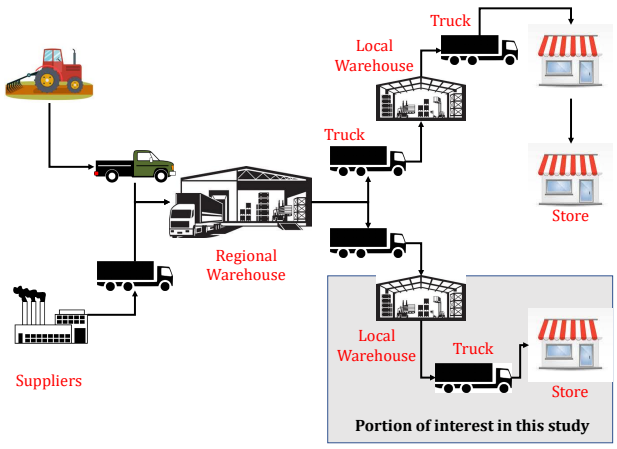

Figure 1: Flow of products within a retail supply chain. The goal is to maintain inventory of the full range of products in the stores, while minimising costs. The present study considers the last step in the supply chain: movement of products from local warehouse to the store.

- We assume that the prior steps in the supply chain are well organized, so that the warehouse itself has sufficient inventory of all products. This implies that the only constraints on the quantity of replenished products are imposed by the volume and weight constraints of the truck.
- We assume that 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 high enough. We do not handle this corner case. Furthermore, we assume that deliveries happen after fixed time intervals.
- In this work, we assume that there is no lag between the computation of replenishment decisions and their implementation in the store. This is merely a simplifying assumption for the purposes of analysis. For handling a finite time lag, it is possible to use estimated inventory levels of products by projecting their current rate of sale. We have shown successful training in a simpler version of the problem in prior work [2].

These assumptions allow us to formulate the problem from an analytical perspective in Section 3, propose a reinforcement learning approach to learn replenishment policies in Section 4, and to experiment with multiple instances (including transfer learning) in Section 5. 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.

Contributions: The principal contributions of this paper are (i) formulating a *parallelisable* RL approach to solve the multi-product constrained inventory management problem, (ii) including realistic business goals such as availability of the entire range of products and minimisation of wastage, and (iii) showing that the approach and its learned policy can be transferred without additional training, for 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 RELATED WORK

We classify prior literature related to this work into various categories, including the description and traditional approaches for control of dynamical systems, data-driven approaches such as adaptive control, approximate dynamic programming (ADP) and imitation learning (IL), decision-making approaches for supply chain and inventory management, and the use of reinforcement learning in related problem areas. While the mathematical aspects of these problems have been studied extensively, we posit that our dimension-agnostic formulation of the problem (covered in Section 4) is novel in literature.

Control of dynamical systems: In Section 3, we show 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 [17] or temperature of a boiler [28]. 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 [37]. 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 [5, 16]. Non-linear versions of the problem are solved using techniques such as sliding mode [53] and model predictive control [29], while robust control under stochastic disturbances is handled using techniques such as H_∞ [11]. The key takeaway from these techniques is that they address one complex aspect of the problem in isolation, focusing on one of (i) stability and robustness, (ii) high dimensionality of state space, or (iii) system identification. By contrast, we require an approach that will handle all these aspects simultaneously. We therefore turn to data-driven methods.

Data-driven control approaches: There exists a large volume of existing literature on adaptive control [1, 19], 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) [4, 38] 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 [46], 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 [15, 52]. These studies use non-linear 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) [37].

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 [40], 3D games [18], and robotics [12]. 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.

RL techniques: Value based methods in reinforcement learning are easy to use [30, 54], while Advantage Actor Critic (A2C) [25] and Deep Deterministic Policy Gradients (DDPG) [27] have been shown to work well on continuous action spaces. Trust Region Policy Optimization [41] and Proximal Policy Optimization [42] have also proven effective for optimal control using RL, but these are

on-policy computationally expensive algorithms and are difficult to apply where episodes are not naturally finite-horizon. Branching DQN [50] is able to handle multiple actions, but has a significant growth in complexity with the size of the state space. Recent work on reducing the size of the action space by learning representations of actions [8] is interesting, but it is not clear how to transfer learning to instances of different scale. We do this by computing individual actions separately (Section 4).

Inventory management: Supply chain control problems have been extensively studied in operations research literature. Most of the literature considers a two-echelon inventory management problem with varying degree of complexity [26, 33, 34]. One major challenge that prior work fails to address is handling multiple products simultaneously. Instances at relatively small scale are solved as joint assortment-stocking problems using mixed-integer linear programming [7, 49] and related techniques such as branch-and-cut [9]. However, these techniques have large computation times and are limited to problems with a handful of product types (fewer than 10) and short time horizons. Implementations at practical scales typically operate with simple heuristics such as threshold-based policies [10] or formulae based on demand assumptions [6, 48].

Adaptive critic [45] and reinforcement learning [14, 20, 31] approaches are also reported in literature, but again tend to focus on single-product problems. Our prior work on a simpler version of the problem [2] 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.

Related applications: In system dynamics, there is significant work in the computation of torque commands for robotic applications [23, 24, 39, 51]. A number of these methods are model-based [32], because of the availability of accurate dynamic models of the robots. The curse of dimensionality often seen in these problems is even more acute in the current context, because of the much higher degrees of freedom. A recent approach for exploration in large state-action spaces is learning by demonstration [35]. However, this too requires an expert policy for imitation learning. Transportation problems [3, 22] sometimes tackle the scalability issue by dividing the global decision-making problem into smaller pieces, with both local and global performance affecting the reward. We use a similar approach in this work.

3 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.

Multivariable dynamical system control: 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 are 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 $\mathbf{x}(\mathbf{0})$ is arbitrarily specified. The control input is provided at discrete time intervals t. Without loss of generality, let us assume that $t \in \mathbb{I}^+$, 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 this as the computation of the optimal control vector $\mathbf{u}^*(\mathbf{t})$ where,

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

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 are unknown but the control input matrix B is known, and both F and B could in general be time dependent. 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 specialize to the inventory management scenario.

Reinforcement learning formulation: 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 $\mathbf{f}(\mathbf{x}) : \mathbb{R}^{\mathbf{p}} \rightarrow \mathbb{R}^{\mathbf{m}\times\mathbf{p}}$, $\mathcal{A}$ 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 $R(n)$,

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

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 $\mathbf{u}(t)$ makes 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 (i) a feature set $\mathbf{f}(\mathbf{x})$ as the input, and (ii) a mechanism (such as simulation) for rolling out the effect of actions on the states in a closed-loop fashion.

Inventory management as a control problem: We instantiate the generic system dynamics with a multi-product inventory management scenario, illustrated in Figure 2. As mentioned before, this is the last step of the retail supply chain from Figure 1. Our goal is to maintain sufficient inventory levels of all products to ensure availability for sale, while simultaneously minimizing 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. 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.

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 are thus $F(\mathbf{x}(z)) = A\,\mathbf{x}(z)$, where matrix A is a constant diagonal matrix with entries $-a_i$. Higher magnitudes 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,

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

The inventory levels x_i and control inputs u_i are assumed to be continuous variables, which is accurate when the products are in liquid or gas form (for example, oil levels replenished periodically by tankers). 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 is $(-a_i\,x_i - W_i)$. Integrating (5) gives the inventory at the end of the time period,

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

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 normalized to the range $[0, 1]$, the set of applicable constraints is as follows.

$$\mathbf{0} \leq \mathbf{x}(t) \leq \mathbf{1} \tag{7}$$
$$\mathbf{0} \leq \mathbf{u}(t) \leq \mathbf{1} \tag{8}$$
$$\mathbf{0} \leq \mathbf{x}(t)^- + \mathbf{u}(t) \leq \mathbf{1} \tag{9}$$
$$\mathbf{v}^T\,\mathbf{u}(t) \leq v_{\max} \tag{10}$$
$$\mathbf{c}^T\,\mathbf{u}(t) \leq c_{\max} \tag{11}$$

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 (i) the reduction of inventory for some products to 0, commonly known as out-of-stock, and (ii) 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 95$^{\text{th}}$ to the 5$^{\text{th}}$ percentile (denoted by $\Delta\mathbf{x}(t)_{.05}^{.95}$) across all products. The objective (reward) to be maximised is defined in (12). Since no control intervention is allowed between two time intervals, all

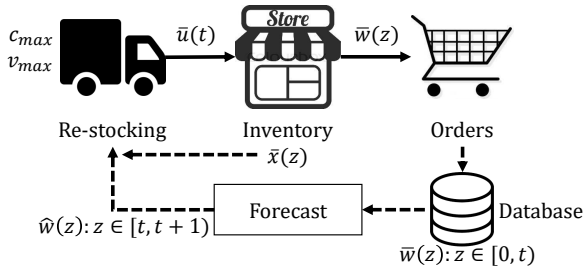

Figure 2: The inventory management problem.

[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 $\mathbf{u}(t)$.

the terms can be considered to be aggregate values received at time t. The objective is defined by,

$$R(t) = 1 - \underbrace{\frac{p_{\text{empty}}(t)}{p}}_{\text{Out of stock}} - \underbrace{\frac{\sum_i q_{\text{waste},i}(t)}{p}}_{\text{Wastage}} - \underbrace{\Delta \mathbf{x}(t)_{.05}^{.95}}_{\text{Percentile spread}}, \quad (12)$$

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 $[t-1, t)$. Since the maximum value of $p_{\text{empty}}(t)$ 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 $-2 \leq R(t) \leq 1$. 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 maximize the discounted sum of this reward as per (4), at each time step t, given the dynamics (5) and the constraints (7)-(11).

4 METHODOLOGY

Before describing the RL approach, 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,

$$\hat{w}_i(z) = \frac{\int_{t-T}^{t} w_i(z')\mathrm{d}z'}{T}, \forall\, z \in [t, t+1)\, i \in \{1, \ldots, p\}. \quad (13)$$

There are several more sophisticated forecasting algorithms available in literature, but these are not the focus of this paper. Note that all the competing algorithms tested in Sec. 5 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 (13).

The primary challenges in applying RL to this problem are (i) the large number of products p, (ii) handling the shared capacity constraints (10)-(11), and (iii) 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.

Rewards: The key challenges with computation of individual elements of $\mathbf{u}$ are (i) ensuring that the system-level constraints (10)-(11) are met, and (ii) 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.

$$\begin{aligned} R_i(t) =& 1 - b_{i,\text{empty}}(t) - q_{\text{waste},i}(t) \\ & - \Delta \mathbf{x}(t)_{.05}^{.95} - \alpha\, \max(\rho - 1, 0), \end{aligned} \quad (14)$$

where $b_{i,\text{empty}}(t)$ is a binary variable indicating whether inventory i dropped to 0 in the current time period, α is a constant parameter,

Table 1: Per-product features used in the RL framework.

Notation	Explanation
$x_i(t)$	Current inventory level
$\hat{W}_i(t)$	Forecast aggregate orders in $[t, t+1)$
σ_i	Historical std. dev. in forecast errors for i
v_i	Unit volume
c_i	Unit weight
l_i	Shelf life
$\mathbf{v}^T\, \hat{\mathbf{W}}(t)$	Total volume of forecast for all products
$\mathbf{c}^T\, \hat{\mathbf{W}}(t)$	Total weight of forecast for all products

and ρ is the ratio of total volume or weight requested by the RL agent to the available capacity. We formally define this as,

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

Equation (14) 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) do not exceed the available capacity ($\rho \leq 1$), then the average value of (14) 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 (14) are common to all products at time t.

States and actions: 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 four inputs are product meta-data, relating to either long-term or constant behaviour. The quantity σ_i is the standard deviation of forecast errors for that product, computed using historical data. Similarly, the shelf life l_i is the normalized 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.

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 $\mathbf{u}(t)$, accoring to the workflow shown in Figure 3.

Neural network architecture: The computation of $\mathbf{u}_i(t)$ is carried out using advantage actor critic (A2C) [25] as well as Deep Q-Network (DQN) [30]. This description covers A2C, with the understanding that DQN is similar but without a separate actor and critic. The critic network accepts the 8 features from Table 1 as input, contains one hidden layer with 4 *tanh* neurons, and produces a scalar value output. The network is trained using stochastic gradient descent with a learning rate of 0.025, momentum of 0.8, a batch size of 32 samples, and mean squared loss with respect to the TD(0) error. The actor network processes the same 8 inputs as the critic,

but its output is a probability distribution over a user-defined set of quantized actions between 0 and 1. In general, the output layer could be any set of n quantized action values. Because of the larger output size, the actor network has two hidden layers with $2n$ neurons each. The hidden layers have *tanh* activation while the output layer has *relu* activation. Its training methodology is mildly modified from standard A2C, as described below.

Modified training approach: The TD(0) discounted reward is used to compute the advantage δ_i for product i with respect to the value estimate. Intuitively, a positive δ_i should encourage the actor to increase the probability of choosing the same action again. However, a subtle difference exists between the outputs of a typical discrete choice problem (such as classification) and the current problem. In our case, the outputs map to specific scalar values, and neighbouring outputs correspond to *similar* actions. Therefore, a loss function such as cross-entropy is not used here, and we also forgo the usual policy-gradient based training methodology for the actor. Instead, we use a mean square loss with respect to an adjusted probability distribution as follows.

Let us assume that the chosen output for a product i is j with an activation $a(j)$, and the realized actor delta from the TD(0) trace is δ_i. Then the target value of each action $j^* \in \{1, \ldots, n\}$ is set to $\left(a(j^*) + \frac{\delta_i}{q\,(|j-j^*|+1)}\right)$. The target vector is then re-normalized to sum up to 1, and the actor is trained using a mean square loss compared to this updated and smoothed vector. The action corresponding to each product in each time period is stored as a unique sample in an experience buffer, with batch training after every 32 time periods of $32p$ samples. The combined vector of desired control actions is denoted by $\mathbf{u}$. Since the output actions are rectified linear units (*relu*), these actions need not satisfy constraints (8)-(11). Therefore, a constrained version of the control action is calculated in two steps. First, $\mathbf{u}_i(t)$ is clipped to a maximum of $(1 - \mathbf{x}_i(t))$, in order to satisfy (8) and (9). Second, constraints (10) and (11) are enforced by proportionally reducing the desired quantities,

$$\mathbf{u}_{con}(t) = \mathbf{u}(t) \cdot \min\left(1,\ \frac{v_{\max}}{\mathbf{v}^T\,\mathbf{u}(t)},\ \frac{c_{\max}}{\mathbf{c}^T\,\mathbf{u}(t)}\right). \tag{15}$$

The capacity exceedance penalty according to ρ ensures that this reduction to feasible values is part of the training rewards for the RL algorithm, and is rarely required after training.

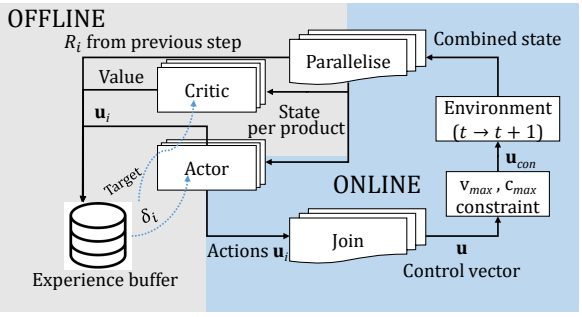

Figure 3: Workflow of RL computation and training, shown for A2C [25]. The left half of the figure operates offline, for training. Online decisions require only the right half.

5 EXPERIMENTS AND RESULTS

We describe experimental results on using the RL approach and other baselines, on data derived from a public source [21]. The data sets were prepared as follows.

5.1 Data for experiments

We use a public data set for brick and mortar stores [21] 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.

5.2 Baseline algorithms

The two RL approaches that we use are DQN and a slightly modified A2C (which we call A2C_mod), as described in Section 4. For comparison, we also use A2C without modification and a categorical cross-entropy loss (called A2C_cat), DDPG [27] with a single continuous action output, and a standard heuristic from supply chain literature, based on proportional control [37].

Heuristic based on proportional control: We use a modified version of s-policy [34], a standard heuristic in 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,

$$\mathbf{u}_{pr}(t) = \max\left[0, \mathbf{x}^* + \hat{\mathbf{W}}(t) - \mathbf{x}(t)^-\right]. \tag{16}$$

The desired action $\mathbf{u}_{pr}(t)$ according to (16) satisfies constraints (8) and (9), since $\mathbf{0} \le \mathbf{x}^* \le \mathbf{1}$. 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 (15).

Other RL approaches: We use two types of related RL approaches in addition to A2C_mod and DQN, for performance comparison. In order to ensure fairness, we retain the same input and output schema and sales forecast values for all algorithms. The

closest related approach is to use vanilla A2C with categorical cross-entropy loss. Second, we implement DDPG [27] which is based on the A2C framework and has continuous action output. We provide the same state as provided in the A2C method to the actor, which outputs replenishment quantities for each product. The weights of actor and critic are shared between all products for fairness. During training, product indices are randomly shuffled to negate any ordering biases.

5.3 Results

We ran each algorithm for 600 training episodes, each containing the 900 time steps in training data. Various initial inventory levels were tried, and the results were largely invariant (the effect of initial state quickly diminishes in a 900-step episode). The training progress of A2C_mod is shown in Figure 4. Of key interest are the curves for 'business reward', which is the overall system reward based on (12), and the 'internal reward', which is the average per product as per (14). As training progresses, the algorithm learns to minimise the difference between the two curves, by minimising capacity overshoot (based on ρ). It also learns to reduce the average inventory levels so as to minimise wastage, while keeping them above the stock-out level θ.

Training comparison: The average rewards as per (12) for all competing algorithms over the course of training (first 900 time periods) on the 220-product data set, are shown in Figure 5. The results on the independent test data set (last 496 time periods) are compiled in Table 2. A2C_mod has the fastest growth in reward, while DQN does the best over 600 episodes. The heuristic has no training, and its performance is thus flat. A2C_cat differs only in the loss function from A2C_mod, but the performance gap is persistent. It also retains a high degree of variance towards the end of training. DDPG converges to the lowest reward on both training and test data in 600 episodes. We observed that DDPG also converges to a similar reward (0.69) if trained for 7500 episodes (12 times the training in Figure 5). The slow nature of convergence may be due to the greater degree of freedom, since it has a continuous output in contrast to the discrete choices available for A2C and DQN.

Transfer learning: The key advantage of the parallelised decision computation described in Section 4, is that the models are able to handle a change in the number of products without retraining. Figure 6 shows the rewards within a single episode for the heuristic and for A2C_mod trained on 220 products, for an instance where only 200 products are active. The truck volume and weight capacity have been proportionally scaled. We observe that A2C_mod is able to generate a consistently higher reward

Table 2: Training and test performance on 220 products.

Model	Training	Testing
A2C_mod (600 episodes)	0.709	0.723
DQN (600 episodes)	0.757	0.737
A2C_cat (600 episodes)	0.671	0.718
DDPG **(7500 episodes)**	0.692	0.702
Heuristic (no training)	0.63	0.607

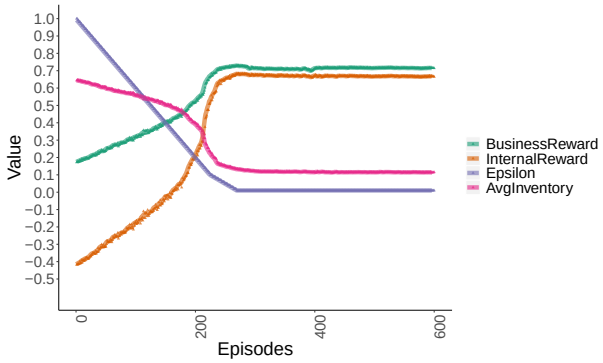

Figure 4: A2C_mod during training on 220 products.

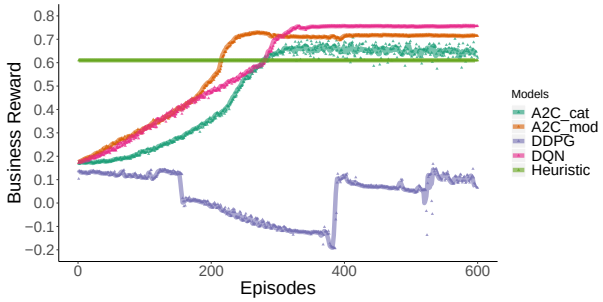

Figure 5: Training comparison for all algorithms.

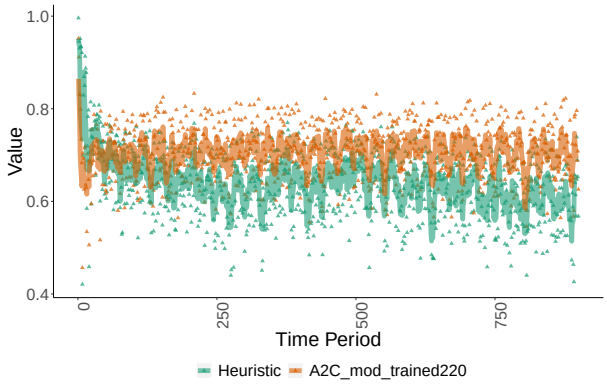

Figure 6: Rewards across 900 DMs over 200 Products

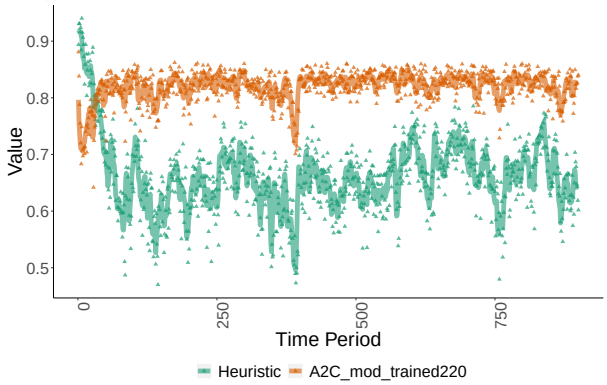

Figure 7: Rewards across 900 DMs over 100 Products

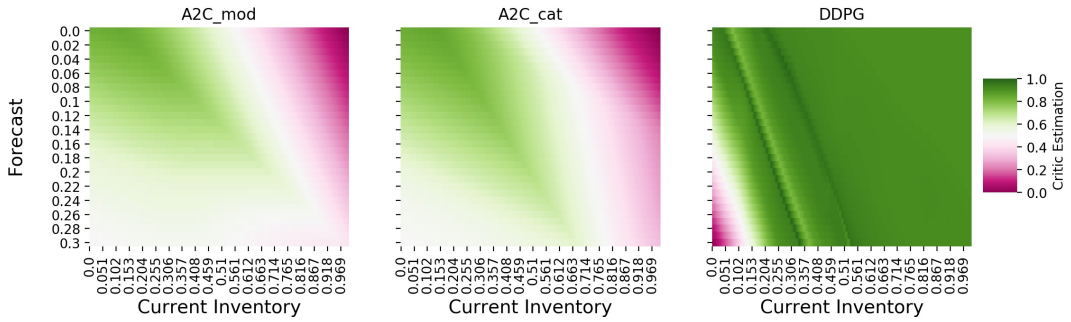

Figure 8: Critic estimate as a function of inventory and forecast.

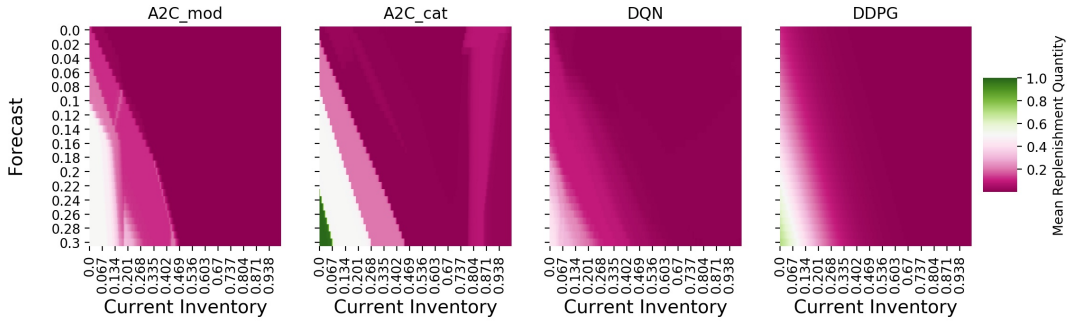

Figure 9: Mean replenishment quantity across products as a function of inventory and forecast.

than the heuristic. The difference is even clearer in Figure 7, where the completely independent set of 100 products (second data set) is being replenished. These results demonstrate the model's capability to handle the variable-dimensional aspect of the inventory management problem.

Quantifying value learning: Value predictions produced by the critic networks for A2C_mod, A2C_cat, and DDPG are shown in Figure 8. The DDPG outputs are generated after 7500 training episodes. The two A2C-based methods have broadly similar value estimates, with the highest values seen for low forecast and moderately low inventory levels. The worst values for A2C are near the top right, where forecast is low while inventory is high. There is no action available to address this problem (cannot artificially reduce inventory levels), which leads to high wastage. The lowest values for DDPG are for low inventory and high forecast (bottom left). This is also an undesirable state, but it can be fixed by a large replenishment action. Note that the DDPG value is based on both state and action, while the A2C values are based only on state.

Quantifying policy learning: Figure 9 compares the requested replenishment action as a function of two features (current inventory and forecast), averaging over all other feature values. We note that A2C_cat has sharp edges in the policy, indicating that the actions are based primarily on inventory and forecast. The other three approaches have broadly similar policies, with DQN having the smoothest variation. This could be the reason for its advantage over all other methods in training as well as test data.

6 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.

REFERENCES

[1] Karl J Åström and Björn Wittenmark. 2013. *Adaptive control.* Courier Corporation.

[2] Souvik Barat, Harshad Khadilkar, Hardik Meisheri, Vinay Kulkarni, Vinita Baniwal, Prashant Kumar, and Monika Gajrani. 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, 1802–1804.

[3] Ana LC Bazzan and Franziska Klügl. 2013. Introduction to intelligent systems in traffic and transportation. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 7, 3 (2013), 1–137.

[4] Dimitri P Bertsekas. 2005. *Dynamic programming and optimal control, Chapter 6.* Vol. 1. Athena scientific Belmont, MA.

[5] Samir Bouabdallah, Andre Noth, and Roland Siegwart. 2004. PID vs LQ control techniques applied to an indoor micro quadrotor. In *Proc. of The IEEE International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2451–2456.

[6] Gfirard Cachon and Marshall Fisher. 1997. Campbell soup's continuous replenishment program: evaluation and enhanced inventory decision rules. *Production and Operations Management* 6, 3 (1997), 266–276.

[7] Felipe Caro and Jérémie Gallien. 2010. Inventory management of a fast-fashion retail network. *Operations Research* 58, 2 (2010), 257–273.

[8] Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip S Thomas. 2019. Learning action representations for reinforcement learning. *arXiv preprint arXiv:1902.00183* (2019).

[9] Leandro C Coelho and Gilbert Laporte. 2014. Optimal joint replenishment, delivery and inventory management policies for perishable products. *Computers & Operations Research* 47 (2014), 42–52.

[10] Cosmin Condea, Frédéric Thiesse, and Elgar Fleisch. 2012. RFID-enabled shelf replenishment with backroom monitoring in retail stores. *Decision Support Systems* 52, 4 (2012), 839–849.

[11] John C Doyle, Keith Glover, Pramod P Khargonekar, and Bruce A Francis. 1989. State-space solutions to standard H/sub 2/and H/sub infinity/control problems. *IEEE Trans. on Automatic control* 34, 8 (1989), 831–847.

[12] Yan Duan, Marcin Andrychowicz, Bradly Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. 2017. One-Shot Imitation Learning. In *NIPS*, Vol. 31.

[13] John Fernie and Leigh Sparks. 2018. *Logistics and retail management: emerging issues and new challenges in the retail supply chain.* Kogan page publishers.

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

[15] Gregory A Godfrey and Warren B Powell. 2002. An adaptive dynamic programming algorithm for dynamic fleet management, I: Single period travel times. *Transportation Science* 36, 1 (2002), 21–39.

[16] F Golnaraghi and BC Kuo. 2010. Automatic control systems. *Complex Variables* 2 (2010), 1–1.

[17] Tore K Gustafsson and Kurt V Waller. 1983. Dynamic modeling and reaction invariant control of pH. *Chemical Engineering Science* 38, 3 (1983), 389–398.

[18] Jack Harmer, Linus Gisslen, Jorge del Val, Henrik Holst, Joakim Bergdahl, Tom Olsson, Kristoffer Sjoo, and Magnus Nordin. 2018. Imitation Learning with Concurrent Actions in 3D Games. *arXiv preprint arXiv:1803.05402* (2018).

[19] Petros A Ioannou and Jing Sun. 1996. *Robust adaptive control.* Vol. 1. PTR Prentice-Hall Upper Saddle River, NJ.

[20] Chengzhi Jiang and Zhaohan Sheng. 2009. Case-based reinforcement learning for dynamic inventory control in a multi-agent supply-chain system. *Expert Systems with Applications* 36, 3 (2009), 6520–6526.

[21] Kaggle. Retrieved 08-2018. Instacart Market Basket Analysis Data. https://www.kaggle.com/c/instacart-market-basket-analysis/data. (Retrieved 08-2018).

[22] H. Khadilkar. 2019. A Scalable Reinforcement Learning Algorithm for Scheduling Railway Lines. *IEEE Transactions on Intelligent Transportation Systems* 20, 2 (Feb 2019), 727–736.

[23] Jens Kober, J Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32, 11 (2013), 1238–1274.

[24] Nate Kohl and Peter Stone. 2004. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, Vol. 3. IEEE, 2619–2624.

[25] V Konda and J Tsitsiklis. 2000. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*. 1008–1014.

[26] Hau L Lee and Corey Billington. 1993. Material management in decentralized supply chains. *Operations research* 41, 5 (1993), 835–847.

[27] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *CoRR* abs/1509.02971 (2015).

[28] X-J Liu and CW Chan. 2006. Neuro-fuzzy generalized predictive control of boiler steam temperature. *IEEE Transactions on energy conversion* 21, 4 (2006), 900–908.

[29] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. 2000. Constrained model predictive control: Stability and optimality. *Automatica* 36, 6 (2000), 789–814.

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep RL. *Nature* 518, 7540 (2015), 529.

[31] Ahmad Mortazavi, Alireza Arshadi Khamseh, and Parham Azimi. 2015. Designing of an intelligent self-adaptive model for supply chain ordering management system. *Engineering Applications of Artificial Intelligence* 37 (2015), 207–220.

[32] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. 2018. Neural network dynamics for model-based deep RL with model-free fine-tuning. In *ICRA*. IEEE, 7559–7566.

[33] Steven Nahmias and Stephen A. Smith. 1993. *Mathematical Models of Retailer Inventory Systems: A Review.* Springer US, Boston, MA, 249–278. https://doi.org/10.1007/978-1-4615-3166-1_14

[34] Steven Nahmias and Stephen A Smith. 1994. Optimizing inventory levels in a two-echelon retailer system with partial lost sales. *Management Science* 40, 5 (1994), 582–596.

[35] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Overcoming exploration in reinforcement learning with demonstrations. In *ICRA*. IEEE, 6292–6299.

[36] Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. 2006. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*. Springer, 363–372.

[37] K Ogata and Y Yang. 2002. *Modern control engineering*. Vol. 4. Prentice Hall.

[38] Warren B Powell. 2007. *Approximate Dynamic Programming: Solving the curses of dimensionality*. Vol. 703. John Wiley & Sons.

[39] Warren B Powell. 2012. AI, OR and control theory: A rosetta stone for stochastic optimization. *Princeton University* (2012).

[40] Stéphane Ross and J. Andrew Bagnell. 2010. Efficient Reductions for Imitation Learning. In *Proc. of The International Conference Artificial Intelligence and Statistics (AISTATS)*.

[41] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning*. 1889–1897.

[42] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[43] Dheeraj Shah. 2020. The Six Aces to Thrive in Supply Chain 4.0. (January 2020). https://www.tcs.com/blogs/six-aces-to-thrive-in-supply-chain-4-0

[44] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295* (2016).

[45] Stephen Shervais, Thaddeus T Shannon, and George G Lendaris. 2003. Intelligent supply chain management using adaptive critic learning. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 33, 2 (2003), 235–244.

[46] Jennie Si, Andrew G Barto, Warren B Powell, and Don Wunsch. 2004. *Handbook of learning and approximate dynamic programming*. Vol. 2. John Wiley & Sons.

[47] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.

[48] Edward A Silver. 1979. A simple inventory replenishment decision rule for a linear trend in demand. *Journal of the Operational Research society* 30, 1 (1979), 71–75.

[49] Stephen A Smith and Narendra Agrawal. 2000. Management of multi-item retail inventory systems with demand substitution. *Operations Research* 48, 1 (2000), 50–64.

[50] Arash Tavakoli, Fabio Pardo, and Petar Kormushev. 2018. Action branching architectures for deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[51] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. 2010. Reinforcement learning of motor skills in high dimensions: A path integral approach. In *Robotics and Automation (ICRA)*. IEEE, 2397–2403.

[52] H Topaloglu and W Powell. 2006. Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems. *INFORMS Journal on Computing* 18, 1 (2006), 31–42.

[53] Vadim Utkin, Jürgen Guldner, and Jingxin Shi. 2009. *Sliding mode control in electro-mechanical systems.* CRC press.

[54] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning.. In *AAAI*, Vol. 2. Phoenix, AZ, 5.

[55] Richa Verma, Sarmimala Saikia, Harshad Khadilkar, Puneet Agarwal, Ashwin Srinivasan, and Gautam Shroff. 2019. An RL Framework for Container Selection and Ship Load Sequencing in Ports. In *International conf. on autonomous agents and multi agent systems*.

[56] W Zhang and T Dietterich. 1995. A reinforcement learning approach to job-shop scheduling. In *International Joint Conference on Artificial Intelligence*. Montreal, Canada.