Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Contextual reinforcement learning for supply chain management

Alex Batsis [a], Spyridon Samothrakis [b,*]

[a] Independent Researcher
[b] Institute for Analytics and Data Science, University of Essex, Wivenhoe Park, United Kingdom

ARTICLE INFO

Keywords:
Reinforcement learning
Supply chains

ABSTRACT

Efficient generalisation in supply chain inventory management is challenging due to a potential mismatch between the model optimised and objective reality. It is hard to know how the real world is configured and, thus, hard to train an agent optimally for it. We address this problem by combining offline training and online adaptation. Agents were trained offline using data from all possible environmental configurations, termed contexts. During an online adaptation phase, agents search for the context maximising rewards. Agents adapted online rapidly and achieved performance close to knowing the context a-priori. In particular, they acted optimally without inferring the correct context, but by finding a suitable one for reward maximisation. By enabling agents to leverage off-line training and online adaptation, we improve their efficiency and effectiveness in unknown environments. The methodology has broader potential applications and contributes to making RL algorithms useful in practical scenarios. We have released the code for this paper under https://github.com/abatsis/supply_chain_few_shot_RL.

0. Introduction

One of the main challenges in Reinforcement Learning (RL) is transferring agents from virtual environments to real-world settings. While we have achieved significant success in training agents in virtual environments using generic algorithms that require little or no tweaking, the training and test environments must be the same. However, in real-world applications, such as self-driving cars, manufacturing, agriculture, or logistics, it is highly unlikely that the test environment is exactly identical to the training environment. This is also particularly true in supply chain management (Forrester, 1961), where the exact environment in which one operates must be discovered – it is highly unrealistic to expect it to be known a priori.

Past efforts to address this challenge have focused on "automatic domain randomisation" (Peng, Andrychowicz, Zaremba, & Abbeel, 2018). This approach involves training an agent with different configurations of the same problem offline, enabling them to be robust to perturbations. For example, agents can be trained using various friction values in a simulator (Akkaya et al., 2019) and then deployed with a policy that is indifferent to/can handle the actual values of real world environments. Different values of a parameter (termed "context" in our discussion, synonymous with domain) induce regularisation and robustness in the perturbations one will surely encounter when deploying in the real world. Usually, the parameters that make up the context could more or less be treated as a form of noise/uncertainty. Recently, however, such setups have been reappraised (Benjamins et al., 2023), and their importance elevated so as to make context the primary focus of research.

The rationale for this paper is that domain randomisation is limited by the fact that it treats all background/latent variables as a form of noise. For example, in the case of a robotic hand, the method would only work if one can treat the friction coefficient (an example of a latent domain/context variable) as noise and still learn enough to act optimally. However, if the materials used in different robots have significantly different friction coefficient values, it would be impossible to find a single policy that works well in all scenarios. Instead, a method that incorporates memory and adapts online to different reward-state structures is necessary (Pisheh Var, Fairbank, & Samothrakis, 2023); this would be equivalent to discovering and making use of *context* (Benjamins et al., 2023), a concept very similar to a domain (i.e. a set of hidden/latent variables), but with the added caveat that it is usually not treated as noise. One would have to incorporate an agent with memory (e.g. an LSTM) and hope it learns internal latent states that represent different domains/contexts and learn to make the link between context and behaviour during training.

Methods that can handle the above scenario are challenging because a single controller must be learnt so as to accommodate all different contexts, something close to meta-learning (Li & Malik, 2016). In this paper, we propose an alternative approach. We first train individual agents who specialise in a specific context. Then, we train an agent that

* Correspondence to: Institute for Analytics and Data Science, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, Essex, United Kingdom.
E-mail addresses: abatsis@mail.com (A. Batsis), ssamot@essex.ac.uk (S. Samothrakis).

https://doi.org/10.1016/j.eswa.2024.123541
Received 29 August 2023; Received in revised form 26 January 2024; Accepted 18 February 2024
Available online 27 February 2024

incorporates context as an input, making it an integral part of the state space. We then distill all these agents into a single agent that treats the context as an unknown variable and performs a black-box search over contexts. We evaluate our method in the task of optimising supply chains, where the exact configuration is unknown to the agent, providing a more realistic scenario compared to assuming precise knowledge of real-world dynamics. We emphasise that the main novelty of our work is not the distilled agent itself, but the online adaptation of the context input. We suggest that this method outperforms the initial domain-specific models and can maintain excellent performance after dramatic changes in the distribution of inputs that the distilled agent, fed the latent context as input, cannot handle. It is also worth clarifying that the aim of this adaptation process is not to infer the latent context but to use it as best as possible to help the agent guide action. The rest of the paper is organised as follows: Section 1 provides the necessary background. Section 2 describes the methods used. Section 3 presents the collected results. Section 4 discusses the results and experiments. Finally, Section 5 concludes the paper, summarises the findings, and explores potential future work.

1. Background

1.1. Reinforcement learning

The aim in reinforcement learning is to create agents that act optimally in regimes of rewards and punishment (Sutton & Barto, 2018). In practise this often translates into agents that can attack Markov decision processes (MDPs) (Puterman, 1990), choosing actions in a way that optimises cumulative reward. Typically, such a model learns by interacting with the MDP in a stochastic way so that it can efficiently explore the space of possible actions.

Definition 1. A Markov decision process is a tuple (S, A, R, T, p) where S is a set representing the state space, A is the set of actions, $R : S \times A \times S \mapsto \mathbb{R}$ is a reward function and $T(s_{t+1}|s_t, a)$ represent the probability of transitioning to state s_{t+1} from state s_t given action a. Finally, p is the distribution of the initial state s_0.

Markov decision processes can be generalised to partially observable Markov decision processes (S, A, O, R, T, ϕ, p) where O represents the space of observations to which the agent has access and $\phi : S \mapsto O$ gives the observation the agent sees given the current state in S.

Definition 2. Given an MDP a policy π is a stochastic function such that $\pi(a|s)$ represents the probability that the agent takes action a given state s. An MDP together with a policy π is associated with the respective history space
$H = \{(s_1, a_1, r_1, s_2, a_2, r_2, \dots) \in \mathbb{R}^{\mathbb{N}} : r_i = R(s_i, a_i, s_{i+1})\}$. From Caratheodory's extension theorem the equation

$$\mu_\pi([S_1, A_1, R_1, \dots, S_n, A_n, R_n]) =$$
$$\prod_{i=1}^{n} T(s_{i+1} \in S_{i+1}|s_i \in S_I, a_i \in A_i)\pi(a_i \in A_i|s_i \in S_i),$$

where

$$[S_1, A_1, R_1, \dots, S_n, A_n, R_n] =$$
$$\{h \in H : (h_{s_i}, h_{a_i}, h_{r_i}) \in S_i \times A_i \times R_i \text{ for } i \in \{1, \dots, n\}\}$$

and S_i, A_i, R_i are intervals, defines a unique probability measure μ_π on H.

The cumulative reward $\mathcal{R} : H \mapsto \mathbb{R}$ is defined as

$$\mathcal{R}(h) = \sum_i \gamma^i h_{r_i}.$$

The discount factor γ is used to ensure that the sum above is finite, values closer to 1 favouring high rewards in the long term. The aim of reinforcement learning is to learn a police π that maximises $\mathbb{E}_{\mu_\pi}(\mathcal{R})$. In practise, it is often assumed that specific states in S are terminating, indicating the end of an episode; alternatively, an episode is defined as a fixed number of steps. In this case, the accumulative reward is defined as the sum of all steps in an episode. In the latter case, the history space H is approximated by $\{(s_1, a_1, r_1, \dots, s_n, a_n, r_n) \in \mathbb{R}^n : r_i = R(s_i, a_i, s_{i+1})\}$ where n is the size of an episode. Finally, for future reference, we define one more useful notion, which is the value function $V_\pi : S \to \mathbb{R}$ defined by

$$V_\pi(s) = \mathbb{E}_{\mu_\pi}\left[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s\right],$$

which is the expected return for starting at state s. The value function intuitively reflects how good it is to be at a given state s.

1.2. Reinforcement learning and supply chains

There exists an extensive body of literature on the use of reinforcement learning in supply chain management. This discussion aims to provide a concise overview of this literature. In a comprehensive study by Gijsbrechts, Boute, Van Mieghem, and Zhang (2022), various applications of reinforcement learning were evaluated in the context of supply chain challenges, including lost sales, dual sourcing, and multi-echelon problems. The investigation demonstrated that the use of reinforcement learning, particularly the A3C model, can effectively match the performance of cutting-edge model-specific heuristics. De Moor, Gijsbrechts, and Boute (2022) introduced the implementation of potential-based reward shaping in a Deep Q-Network (DQN) for policy transfer. By incorporating insights from well-established heuristics, such as base stock and BSP-low-EW policies, the researchers addressed the perishable inventory problem. Their methodology achieved training stability, improved performance, and made training fast. In particular, the study revealed that the shaped DQN outperforms the teacher policy in scenarios where the latter outperforms the unshaped DQN. The investigation carried out by Stranieri and Stella (2022) focused on the specifics of divergent management of the two-echelon supply chain, accommodating varying unit quantities at each level. To tackle this challenge, the researchers developed an environment tailored to the problem and designed a dedicated library for its resolution using deep reinforcement learning techniques. The experimentation encompassed A3C, VPG, and PPO algorithms, with results that demonstrate that these algorithms achieved nearly optimal solutions. Notably, PPO demonstrated superior adaptability and overall performance. Addressing the complexities of multiproduct and multistore supply chains with short product life cycles, Demizu, Fukazawa, and Morita (2023) presented a methodology tested on the example of new smartphones, where one has very limited data on novel products. The proposed approach involves an offline learning phase for a demand forecasting model and a random shooting model for real-time management. Through empirical validation using real-world data, the researchers established the superiority of their approach over existing methods.

1.3. Multi-echelon inventory management

The problem of multi-echelon inventory management was first formalised by Glasserman and Tayur (1995). A supply chain starts from a retailer which orders products from a supplier that produces and ships products to the retailer. The supplier may have its own supplier, and so on until we reach the party that sources raw materials. We will refer to the different parties described above as levels. The inventory management problem is to optimise the way orders to the suppliers are placed to optimise profit given prices, inventory holding costs, shipping costs, lead times, and production capacity. Our inventory management modelling is based on the work of Hubbs et al. (2020). In order to be as comprehensive as possible, we describe the framework here.

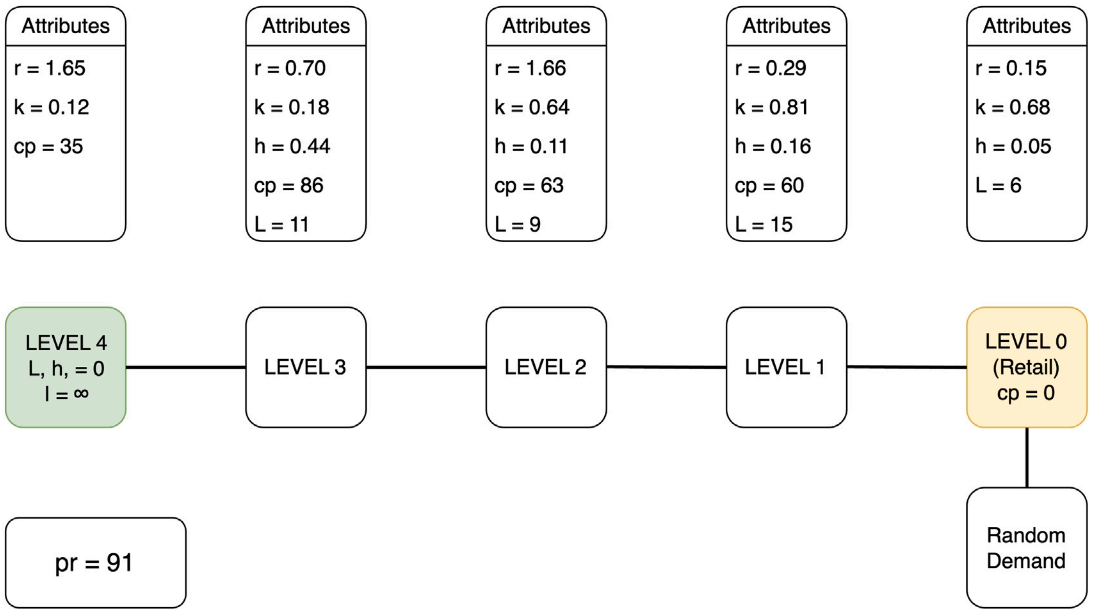


Fig. 1. Example of a supply chain configuration.

Table 1
Element V.

Vector variable	Interpretation
I_0	Initial inventory
r	Replenishment costs per unit
k	Unfulfilled orders costs
h	Holding costs per unit
cp	Production capacity
L	Lead times
pr	Unit price for final product

Definition 3. Let $V = (I_0, r, k, h, cp, L, pr) \in (\mathbb{R}^{m+1})^6 \times \mathbb{R}$, $\tau \in \mathbb{N}$ be a number representing the number of steps in an episode (also called number of periods) and D a distribution generating random demand. Also assume that $cp(0) = 0$, $L(m) = 0$, $I_0(m) = \infty$ and $h(m) = 0$. Then we will say that the tuple (V, τ, D) is a supply chain configuration of size m.

The semantic interpretation of the element V is given in the Table 1, the variables are vectors as they hold information for all $m + 1$ levels of the supply chain. See also Fig. 1. Given a supply chain configuration we can model the inventory management task as a MDP as follows.

Definition 4. Let (V, τ, D), with $V = (I_0, r, k, h, cp, L, pr)$, be a supply chain configuration of size m. The respective MDP, which we will call the supply chain environment, is (S, A, R, T, p) where $S = \mathbb{R}^{m+m*max(L)}$. An element of S is to be understood as m numbers I^i describing the inventory at level i plus $m * max(L)$ numbers O^i_t describing the orders accepted at level i during step t for the last $max(L)$ steps and all m levels. We set $A = \mathbb{R}^m$ to be interpreted as the set of vectors $(a^1_t, \dots, a^m_t)$ describing the orders placed by each level at step t. The transitions T are described by the equations below,

$$I^i_{t+1} = I^i_t + O^i_{t-L(i)} - S^i_t, \quad i \in \{0, \dots, m-1\}$$

$$O^i_{t+1} = \min(cp(i+1), I^{i+1}_t, a^i_t), \quad i \in \{0, \dots, m-1\}$$

$$S^i_t = \begin{cases} O^{i-1}_t, & i > 0 \\ \min(I^0_t + O^0_{t-L(i)}, D_t), & i = 0 \end{cases} \quad i \in \{0, \dots, m\}$$

where D_t is the random demand at time t following the distribution D. Reward R is given by

$$U^i_t = \begin{cases} a^{i-1}_t - S^i_t, & i > 0 \\ D_t - S^i_t, & i = 0 \end{cases}$$

$$R^i_t = prS^i_t - r(i)O^i_t - k(i)U^i_t - h(i)I^i_{t+1}, \quad i < m$$

$$R_t = \sum_{i=0}^{m-1} R_i + (pr - r(m))S(m).$$

The distribution p of the initial state is the Dirac measure determined by I_0.

We will use the terms supply chain configuration and supply chain environment interchangeably as they uniquely determine each other. For more details see Hubbs et al. (2020).

1.4. Zero-shot and few-shot generalisation

The main idea of zero-shot generalisation is to build a reinforcement learning model that works on a family of MDPs parameterised by a variable representing context as part of what defines each MDP. This allows the model to act in environments that it has never seen before or be robust in small changes of the same environment. This is particularly useful in real-world applications where environments can be significantly different from those used for training. For an extensive survey on the subject, see Kirk, Zhang, Grefenstette, and Rocktäschel (2023). In this project, the context will be the supply chain configuration, and our aim is to build a model that can work for any such configuration when that is given as an input to the model, without any additional training required. To formalise this, we give the definition of a contextual Markov decision process (CMDP) as presented in Hallak, Di Castro, and Mannor (2015). For simplicity we do not cover the partial observably case as this is beyond the scope of this paper.

Definition 5. A Contextual Markov Decision Process (CMDP) $\mathcal{M}$ is a tuple $(S', A, R, T, C, p(s'|c), p(c))$ where S' stands for the underlying state space, C is the context space, and A, R and T are the same as in definition of MDP. Finally $p(c)$ is a distribution among different context elements and $p(s'|c)$ is the distribution of the initial state in the respective MDP. A contextual Markov decision process will be identified with the MDP $(S' \times C, A, R, T, p(c)p(s'|c))$ which precisely determines the types of A, R and T. The sets $S_c = S' \times \{c\}$ are assumed to be invariant in the sense that

$$c_1 \neq c_2 \implies T((s', c_2), a|(s, c_1)) = 0.$$

We will also need a notion of restricting to finite subsets $C_s \subseteq C$.

Definition 6. Let $\mathcal{M}$ be a CMDP $(S', A, R, T, C, p(s'|c), p(c))$ and $C_s \subseteq C$ be finite. We denote by $\mathcal{M}|_C$ the contextual MDP $(S', A, R, T, C_s, p_s(s'|c), p_s(c))$ where

$$p_s(c) = \left(\sum_{x \in C_s} p(x)\right)^{-1} p(c), \qquad c \in C_s.$$

Also, for $c_0 \in C$ we set $\mathcal{M}_{c_0}$ to be the MDP $(S' \times \{c_0\}, A, R_{c_0}, T_{c_0}, p(s|c))$ where R_{c_0}, T_{c_0} are the restrictions required of R and T respectively.

We can now formulate the zero-shot generalisation problem for a CMDP $\mathcal{M}$. We have finite samples $C_{\text{train}}, C_{\text{test}} \subseteq C$ and set p_{test} to be the context distribution of $\mathcal{M}|_{C_{\text{test}}}$. The aim is to build a policy π for $\mathcal{M}$ that maximises

$$\mathbb{E}_{c \sim p_{\text{test}}(c)}(\mathbb{E}_{\mu_\pi}(\mathcal{R}_c)),$$

where $\mathcal{R}_c$ is the cumulative reward corresponding to MDP $\mathcal{M}_c$, assuming that we have access only to $\mathcal{M}|_{C_{\text{train}}}$ during the training process.

1.5. Proximal policy optimisation

Training a policy neural network in reinforcement learning comes with additional challenges compared to supervised learning. We do not have a static high-quality diverse dataset beforehand to use for learning. We acquire data through sessions of interaction with the environment (consisting of multiple episodes), but the data we get this way have a very narrow scope for each session and at the early stages of learning are not particularly meaningful. The general strategy for reinforcement learning algorithms is based on a loop of interacting with the environment to collect data, and then only partially optimise the policy before we go back to the environment to collect more data through stochastically acting on the environment based on the new updated policy. The reason why randomness is involved in choice of actions is that we can make sure we keep exploring new ways of acting on the environment. In vanilla policy gradients (see Mnih et al., 2016) the direction of the policy optimisation is determined by the objective function

$$\mathbb{E}_{\mu_\pi}[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t],$$

where $\hat{A}_t$ is an estimator of the advantage function

$$\hat{A}_t = V_{\pi_{\text{current}}}(s_t) - V_{\pi_{\text{previous}}}(s_t).$$

The downside of the vanilla policy gradient is that it is highly unstable, as the data we collect during a policy update iteration can lead to the destruction of an effective policy and throw our model at an area far away from the optimal policy. To deal with this problem, the authors of Schulman, Levine, Abbeel, Jordan, and Moritz (2015) came up with the trust region policy optimization (TRO). The main idea of TRO is to use the following objective function.

$$\mathbb{E}_{\mu_\pi}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t\right],$$

constrained to

$$\mathbb{E}_{\mu_\pi}\left[\text{KL}(\pi_{\theta_{\text{old}}}(\cdot, s_t), \pi_\theta(\cdot, s_t))\right] < \delta,$$

where KL stands for the relative entropy of the distribution $\pi_{\theta_{\text{old}}}(\cdot, s_t)$ with respect to the distribution $\pi_\theta(\cdot, s_t)$. Relative entropy is a generalisation of Shannon's entropy that can be used as a distance function between probability measures. It should be noted that it is not a metric and in particular it is not symmetric. Bounding KL acts as a constraint that protects from dramatic changes during policy update. Proximal policy optimisation algorithm is an improvement of TRO based on a computationally simpler objective function. The benefits of PPO is that it is easy to implement, it is more stable in terms of hyper-parameter tuning, and it is faster than TRO without losing precision. The most important term in the objective function that PPO uses is

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_{\mu_\pi}[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)],$$

where

$$\text{clip}(x) = \begin{cases} 1-\epsilon, & x <= 1-\epsilon \\ x, & x \in (1-\epsilon, 1+\epsilon) \\ 1+\epsilon, & \text{else} \end{cases}$$

PPO (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) uses the function L^{CLIP} to achieve stability during policy update, which is much simpler than the more computationally expensive KL distance used in TRO. The final objective function also includes a term that penalises low entropy for the action distributions $\pi(\cdot, s)$ and ensures that the exploration of actions is efficient.

1.6. CMA-ES

Covariance matrix adaptation evolution strategy (CMA-ES) is a stochastic evolutionary optimisation algorithm for non-convex or non-linear continuous optimisation problems. It has been proven exceptionally successful in a wide variety of practical applications (Hansen, Auger, Ros, Finck, & Pošík, 2010). In this section notation of the form c_i stands for constants; for simplicity, we will not expand on how these constants are determined here. We will also assume that the recombination weights are trivial. To summarise CMA-ES, assumes that $f : \mathbb{R}^n \to \mathbb{R}$ is the given function we want to minimise. It starts with a multivariate normal distribution on the domain of f of the form

$$m + \sigma\mathcal{N}(0, C)$$

CMA-ES samples λ number of points $x_1, \dots, x_\lambda$ from the distribution above, sort them according to f and select the l points of the lowest f value. The numbers λ and l are constants and called the population size and parent number respectively. We will denote by $x_{1:\lambda}, \dots, x_{\lambda:\lambda}$ the sequence after sorting. That is, there is a permutation π such that

$$(x_{1:\lambda}, \dots, x_{\lambda:\lambda}) = (x_{\pi(1)}, \dots, x_{\pi(\lambda)})$$

and

$$f(x_{1:\lambda}) \leq \dots \leq f(x_{\lambda:\lambda})$$

It then use the selected points $x_{1:\lambda}, \dots, x_{l:\lambda}$ to inform and adapt the sampling distribution and start over by generating a new sample $x_1, \dots, x_\lambda$ from the updated distribution. From generation κ to generation $\kappa + 1$ the mean parameter of the normal distribution is updated by

$$m_{\kappa+1} = \sum_{i=1}^{l} x_{i:\lambda}$$

For the adaptation of the covariance matrix C there are two different strategies to consider. The first one is to use the empirical covariance matrix of the selected points $x_{1:\lambda}, \dots, x_{l:\lambda}$ to update the covariance matrix according to the equation

$$C_{\kappa+1} = c_1 C_\kappa + c_2 \sum_{i=1}^{l} \frac{x_{i:\lambda} - m_\kappa}{\sigma_\kappa}\left(\frac{x_{i:\lambda} - m_\kappa}{\sigma_\kappa}\right)^\intercal$$

The second strategy is cumulation control for the implementation of which we keep track of the path vector p_c^κ. Here, it updates the covariance matrix to favour a single direction, which is the one defined by the mean step. It does that by updating as follows, we first update the path control vector,

$$p_c^{\kappa+1} = c_3 p_c^{\kappa} + c_4 \frac{m_{\kappa+1} - m_{\kappa}}{\sigma_{\kappa}}$$

and the covariance matrix is updated by

$$C_{\kappa+1} = c_5 C_{\kappa} + c_6 p_c^{\kappa+1} \left(p_c^{\kappa+1}\right)^{\intercal}.$$

The term $p_c^{\kappa+1} \left(p_c^{\kappa+1}\right)^{\intercal}$ can be more intuitively understood by equation

$$p_c^{\kappa+1} \left(p_c^{\kappa+1}\right)^{\intercal} x =< x, p_c^{\kappa+1} > p_c^{\kappa+1}$$

where $\langle \cdot, \cdot \rangle$ is the inner product. The CMA-ES algorithm combines the two methods in an updating equation of the form

$$C_{\kappa+1} = c_7 C_{\kappa} + c_8 p_c^{\kappa+1} \left(p_c^{\kappa+1}\right)^{\intercal} + c_9 \sum_{i=1}^{l} \frac{x_{i:\lambda} - m_{\kappa}}{\sigma_{\kappa}} \left(\frac{x_{i:\lambda} - m_{\kappa}}{\sigma_{\kappa}}\right)^{\intercal}.$$

Finally to update the total variance σ we use the step-size control vector p_{σ}^{κ}, the following is done:

$$p_{\sigma}^{\kappa+1} = c_{10} p_{\sigma}^{\kappa} + c_{11} C_{\kappa}^{-1/2} \frac{m_{\kappa+1} - m_{\kappa}}{\sigma_{\kappa}}$$

and

$$\sigma_{\kappa+1} = \sigma_{\kappa} \exp(c_{12} p_{\sigma}^{\kappa+1} + c_{13}).$$

Here the matrix $C_{\kappa}^{-1/2}$ acts as normalisation transformation in the sense that

$$x \sim \mathcal{N}(0, C) \implies C^{-1/2} x \sim \mathcal{N}(0, \mathbf{I}).$$

2. Methods

2.1. Contextual Markov decision process

We can frame the inventory management problem in this framework by identifying finite sequences of numbers $(a_0, \ldots, a_n)$ with zero terminating sequences $(a_0, \ldots, a_n, 0, 0, \ldots)$ (computationally, we do that with zero padding). This way, all supply chain configurations share a common state space S' and action space A. The context set C will be the set of all supply chain configurations and $R((s', c), a, (s, c)) = R_c(s', c, s)$ where R_c is the reward function corresponding to the configuration c. By the definition of CMDP the case of $R((s', c'), a, (s, c))$ with $c' \neq c$ is irrelevant and can be set to anything.

2.2. Meta-learning

Meta-learning is an umbrella term that refers to methods that include a meta-learner model that learns from the outcomes of other machine learning models or "learns-to-learn". In our context, with the notation of Section 1.4, we consider a sample $C_{\text{train}} \subseteq C$ and for each $c \in C_{\text{train}}$ we consider a pre-trained model corresponding to an approximation π_c of the optimal policy for $\mathcal{M}_c$. Through further (after training) interaction of π_c with the environment $\mathcal{M}_c$ we generate data sets D_c containing data points of the form $((s, c), a) \in S \times A$, where a is an action taken by π_c given state s. We summarise in the following diagram

$$c \longrightarrow \mathcal{M}_c \longrightarrow \pi_c \longrightarrow D_c$$

The meta-learner is a neural network trained with supervised learning on the data set

$$D_{\text{train}} = \bigcup_{c \in C_{\text{train}}} D_c,$$

to predict $a = \pi_c(s)$ when (s, c) is given as input (we emphasise that the context is part of the input of the model). Similarly, we construct a data set D_{test} to evaluate the performance of our model. We note that we split D_{train} further to extract a validation set used to optimise regularisation. To calculate the policies π_c, we use the PPO algorithm.

Table 2
Distributions used to generate random supply chain configurations.

Parameter	LB	UB	Distribution
Initial inventory	10	100	discrete
Replenishment cost	0	2	continuous
Unfulfilled cost	0	1	continuous
Excess holding cost	0	0.5	continuous
Production capacities	10	100	discrete
Lead times	1	20	discrete
Final product price	2	100	discrete

2.3. Data generation

We randomly generate supply chain configurations to obtain finite samples C_{train} and C_{test} of C. The discount factor is set to 0.97 and the costumer demand distribution is set to be the uniform distribution on the set $\{10, \ldots, 99\}$ for all configurations. In order to test unpredictable changes that can occur in a real-life use case, we also generate a test set C_{skewed} where the costumer demand distribution for each configuration is Poisson with expected value 70. We set $C_{\text{test\_all}} = C_{\text{test}} \cup C_{\text{skewed}}$. The size of the supply chain m is uniformly chosen from the set $\{3, \ldots, 8\}$ while the number of periods is uniformly chosen from the set $\{10, \ldots, 99\}$. The other parameters are randomly generated with continuous or discrete uniform distribution of support $[\text{LB}, \text{UB}]$ or $[\text{LB}, \text{UB}) \cap \mathbb{Z}$ respectively, as shown in Table 2.

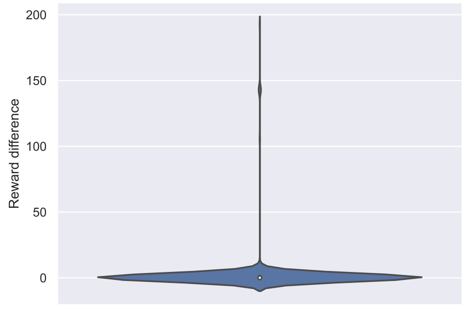
For each of these environments $c \in C_{\text{train}} \cup C_{\text{test\_all}}$ we train a PPO model π_c and after training is complete, we evaluate the PPO model on that same environment producing the datasets D_{train}, D_{test} and D_{skewed} described in the previous subsection. So each data point consists of the state of the supply chain, the action taken by the model, and the configuration of the environment. We also save the respective reward for each action taken by π_c for future comparisons.

2.4. Model

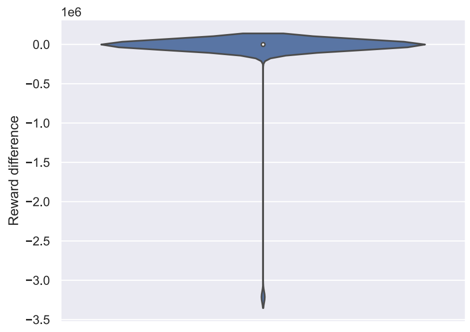
We use a neural network model which is an ensemble (Dietterich, 2000) of sequential models with skip connections (Orhan & Pitkow, 2018). The benefits of using both methods in order to stabilise neural networks have been well known in the literature for a while. We used a random sample of 10% of the training data as a validation set to experiment with the model architecture. For each data point $((s, c), a)$ the input data for the model consists of the numbers that describe the state $s \in S_c$ of the supply chain environment and the configuration of the environment itself $c \in C$. The output data are the numbers that describe the respective action $a \in A$ taken by the PPO model π_c that was trained for that specific environment. Since the various components of the input data as well as the output data have different sizes for different environments, we standardise the form of the data by zero padding. To do that, we perform an analysis of the different data sizes in different environments $c \in C_{\text{train}}$.

2.5. Evaluation

We perform two experiments to evaluate the meta-learner model. In the first experiment, we evaluate the meta-learner for each environment in $c \in C_{\text{test\_all}}$, providing the environment configuration c to the meta-learner model as the context input. The results are then compared with the performance of the respective PPO model π_c, which was monitored and saved during the evaluation stage of the data generation process. The second experiment is a hidden context evaluation where the supply chain configuration $c_{\text{true}} \in C_{\text{test\_all}}$ is unknown to the model. The environment guess, given as context input $c_{\text{input}} \in C$ to the model, is optimised online as the model interacts with the environment using the CMA-ES optimisation algorithm. More precisely, we perform CMA-ES optimisation to maximise the stochastic function $\mathcal{R}_{1\text{epEval}:c_{\text{true}}}(c_{\text{input}})$

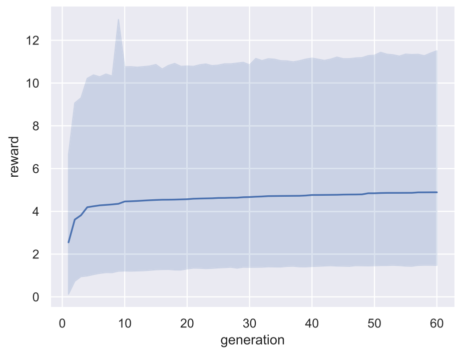


(a) Difference of meta-learner rewards collected during the evaluation process of the experiment 1 minus rewards given by the PPO models during the data generation phase per $c \in C_{\text{test}}$.

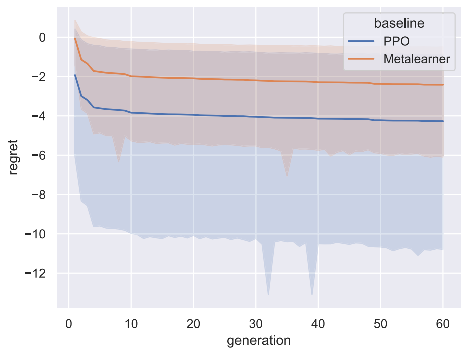


(b) Difference of meta-learner rewards collected during the evaluation process of the experiment 1 minus rewards given by the PPO models during the data generation phase per $c \in C_{\text{skewed}}$.

Fig. 2. Results comparisons between the learners and the metalearner.

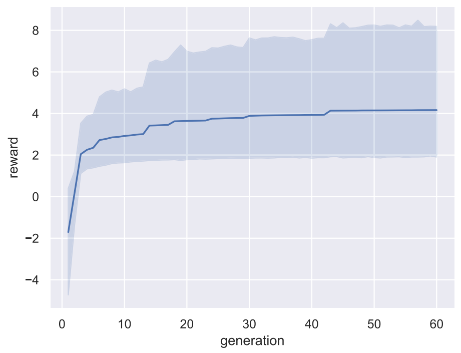


(a) Online rewards of the meta-learner per CMA-ES generation in experiment 2, averaged over all different $c \in C_{\text{test}}$. Each generation is 10 function evaluations.

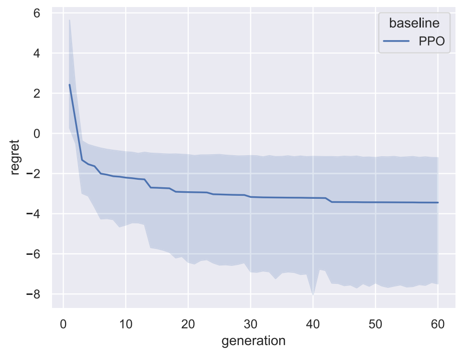


(b) Regret of the meta-learner rewards (against true environment configuration and PPO performance) per CMA-ES generation in experiment 2, averaged over all different $c \in C_{\text{test}}$.

Fig. 3. Results comparisons between the learners and the metalearner.



(a) Online rewards of the meta-learner per CMA-ES generation in experiment 2, averaged over all different $c \in C_{\text{skewed}}$. Each generation is 10 function evaluations.



(b) Regret of the meta-learner rewards against PPO per CMA-ES generation in experiment 2, averaged over all different $c \in C_{\text{skewed}}$.

Fig. 4. Results comparisons between the learners and the metalearner.

that maps $c_{\text{input}} \in C$ to the cumulative reward obtained when the meta-learner interacts with the environment $\mathcal{M}_{c_{\text{true}}}$ for exactly one episode given c_{input} as context input. We then compute the regret of the meta-learner rewards (collected during the context optimisation process) compared both to the performance of the respective PPO model π_c and the meta-learner when c_{true} is given as context input, call them R_{PPO} and R_{ML} respectively. To be precise $R_{\text{PPO}} = \text{reward} - \text{ARP}(c)$ and $R_{\text{ML}} = \text{reward} - \text{ARM}(c)$ where $\text{ARP}(c)$ is the average reward collected from the interaction of π_c with $\mathcal{M}_c$ during the data generation process of D_c and $\text{ARM}(c)$ is the average reward collected from the interaction of the meta-learner with $\mathcal{M}_c$ during the evaluation process of experiment 1.

2.6. Model explanation

Model explainability of the meta-learner can be useful in providing useful insights relevant to supply chain management. We visualise the behaviour of the meta-learner model by fitting decision trees to the data generated during the evaluation experiments and extracting visualisations out of these trees. For visualising the decision trees we used the python library dtreeviz (Parr, Lapusan, & GroverSmith, 2024). In particular Fig. 7 corresponds to the data generated in experiment 1 where the meta-learner interacts with the environments $c \in C_{\text{test}}$ by giving it the true environment configuration as context input. Fig. 8 corresponds to data generated by the meta-learner interacting with

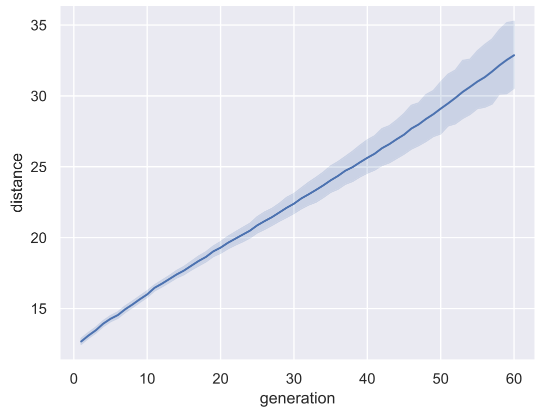


Fig. 5. Distance of the meta-learner context input $c_{\text{input}} \in C$ and the true context $c_{\text{true}} \in C_{\text{test}}$ corresponding to the supply chain environment per CMA-ES generation in experiment 2, averaged over all different $c \in C_{\text{test}}$.

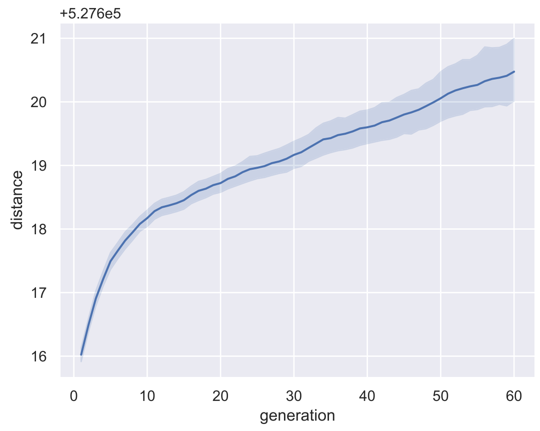


Fig. 6. Distance of the meta-learner context input $c_{\text{input}} \in C$ and the true context $c_{\text{true}} \in C_{\text{skewed}}$ corresponding to the supply chain environment per CMA-ES generation in experiment 2, averaged over all different $c \in C_{\text{skewed}}$.

environments $c \in C_{\text{test}}$ with given context input set to be the context estimation $c_{\text{optimised}}$ inferred after the CMA-ES optimisation process has taken place in experiment 2. We could not produce a similar visualisation for the case of C_{skewed} as the decision tree approximation fails to achieve a satisfactory loss compared to agent behaviour. We attribute that to the complexity of the model behaviour in this case.

3. Results

The results of experiment 1 are presented on Table 3 and Figs. 2(a), 2(b). We can see that in C_{test} the meta-learner model has a very similar performance to the PPO models and in rare cases it significantly outperforms them. On the other hand, in C_{skewed} the meta-learner completely falls over (notice the scientific notation in Fig. 2(b)). The results presented for experiment 2 in Figs. 3, 4, 5 and 6 are grouped and averaged over C_{test} and C_{skewed}. To avoid having our results skewed by environment configurations that give rise to high rewards or having environments of low rewards under-represented, we normalise the data for each $c \in C_{\text{test\_all}}$ separately to equalise the contribution of each $c \in C_{\text{test\_all}}$. To be precise, for each test corresponding to $c \in C_{\text{test\_all}}$, the reward and/or regret are divided by the term

$$\sum_{i=1}^{n} \sum_{j=1}^{\tau_c} \gamma^j |r_{i,j}|,$$

where $r_{i,j}$ are the rewards collected during the evaluation stage of the data generation process, n is the number of episodes used for that evaluation, and τ_c is the number of steps in each episode. In Figs. 3 and 4 we can see the optimisation of meta-learner online rewards, that were collected during CMA-ES optimisation, and the regret against the corresponding PPO model and the meta-learner with true context given. We see that regret drops significantly below zero, which roughly translates to performance being better vs known context and PPO. We emphasise that this is true even in the case of C_{skewed}, unlike experiment 1. During the same process we monitor the distance of the context estimation $c_{\text{input}} \in C$ from the true context $c_{\text{true}} \in C_{\text{test\_all}}$ corresponding to the supply chain configuration. Distance was measured with the Euclidean norm on normalised versions of the number vectors describing the respective context $c \in C$. We see in Fig. 5 that in the case of C_{test} the optimised context input c_{input} clearly does not converge to the true configuration of the supply chain environment c_{true} but stays relatively close to it. In the case of C_{skewed} it seems that c_{true} quickly becomes irrelevant and c_{input} diverges far away from it.

Table 3
In the first two columns we see meta-learner (D-PPO) rewards collected during the evaluation process of the experiment 1 for C_{test} compared to the rewards given by the PPO models during the data generation phase for D_{test}. Next two columns correspond to the same report for C_{skewed}.

	PPO	D-PPO	PPO_S	D-PPO_S
Mean	0.621	2.473	0.716	−35403
SD	0.800	14.334	0.700	319 540
Min	−2.988	−4.105	−1.089	≈-3e+06
Q1	0.956	0.853	0.995	−2960
Q2	0.997	1.031	0.999	−1309
Q3	1.006	1.187	1.001	−526
Max	3.420	193.171	2.823	−49

4. Discussion

Our findings suggest that by combining multiple smaller teacher neural networks, trained in different environments, a larger neural network can be created with overall similar performance. One can then further train this distilled neural network (the metalearner) using black-box methods over the environmental configuration and outperform the teacher agents, as well as maintain performance in cases of data drifts that can significantly reduce the performance of the distilled neural network.

Our approach is based on existing research that explores the concept of distilling knowledge from multiple sources. Previous studies have shown that combining information from diverse environments can lead to improved performance and improved adaptability in neural networks, as in GATO (Laskin et al., 2022); in our case, we have the added flexibility of not providing the context to the algorithm at any point, but adapting to it online. We consider this to be a far more realistic scenario, as in the general case real-world environmental dynamics cannot be replicated exactly inside a computer. Our method can also be linked directly to Jiang, Ke, and van Hasselt (2023), without the added POMPDP component, but also to efforts to create end-to-end differentiable simulators that can be "linked" to real world (Heiden, Millard, Coumans, & Sukhatme, 2020), with our main difference being that we do not attempt to adapt the model per se, but use selected model parameters for direct reward optimisation.

The surprising aspect of our results lies in the fact that the neural network never learns to approximate the real environment directly. Instead, it improves its performance by "imagining" a completely fictional configuration, as shown in Fig. 5. This indicates that the neural network is not dependent on accurate representation of the real world, but uses the extra parameters as it sees fit so as achieve better performance. Furthermore, and even more surprisingly, CMA-ES is able to uncover solutions that are better than those trained with knowledge of the context, hence the negative regret. This is further evidence that context variables tend to be used more like variables that can be

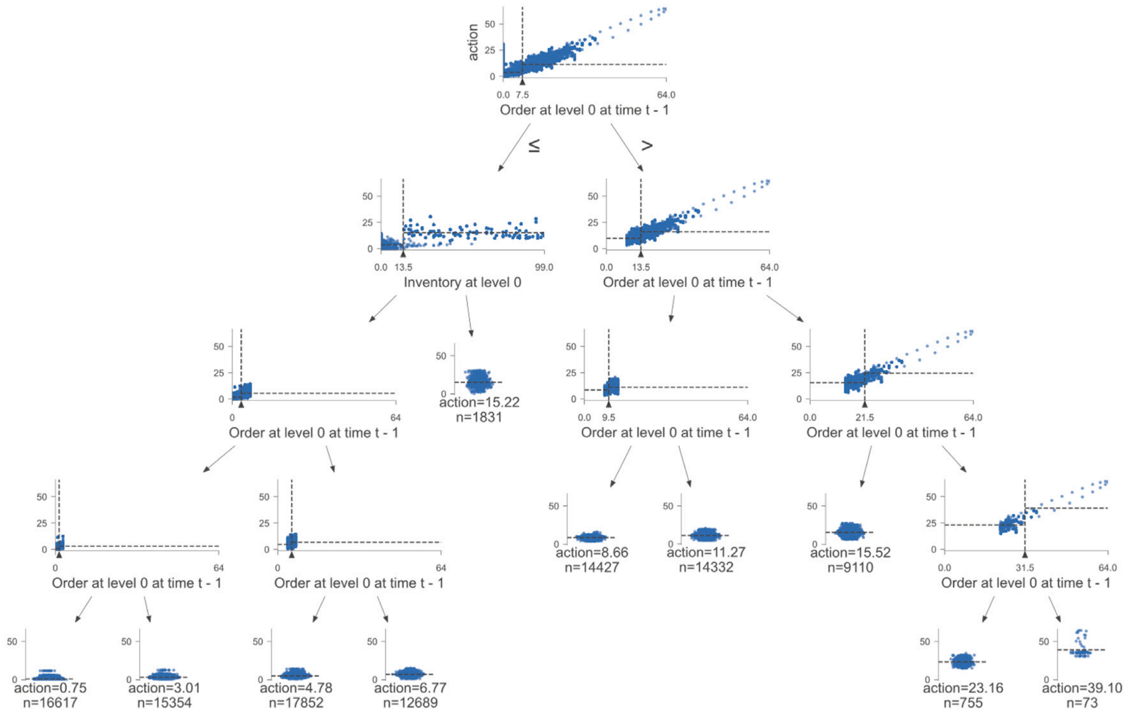


Fig. 7. Model behaviour visualisation, at the retail level, when the true supply chain configuration is given as context input for C_{test}.

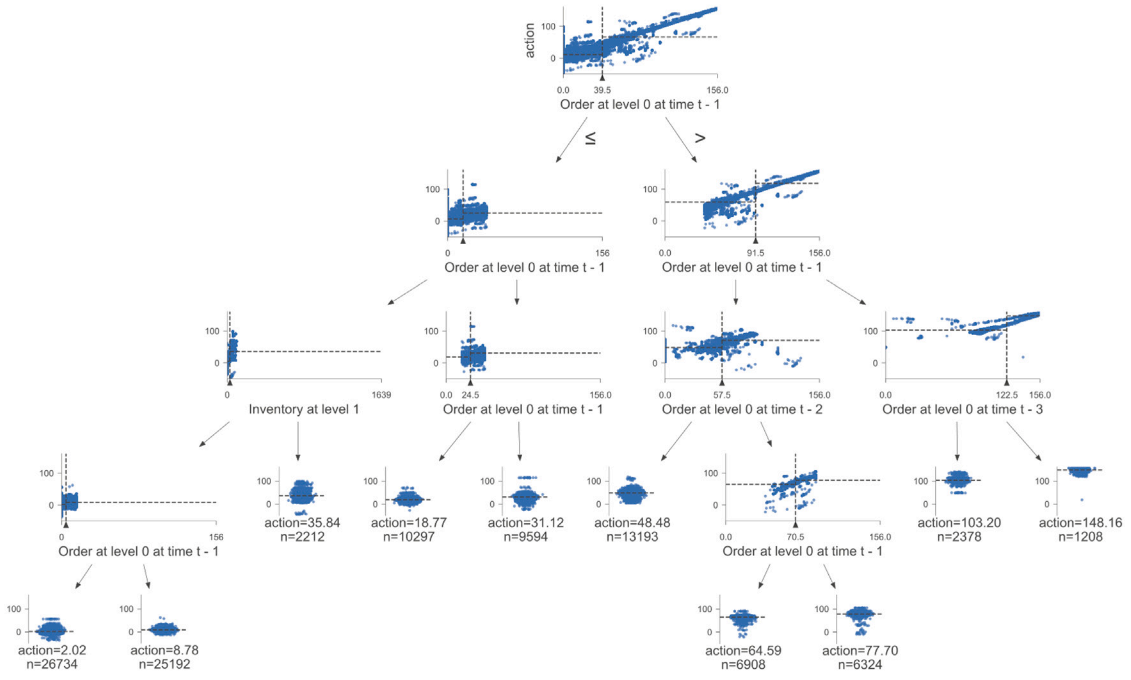


Fig. 8. Model behaviour visualisation, at the retail level, after CMA-ES adaptation for C_{test}.

optimised to help performance, rather than what their original purpose was.

If this result can be replicated across various settings, not limited to our specific setup, it will have significant implications beyond supply chains. It suggests that latent variables, which can be entirely fictive, have the potential to enable exceptional performance in a model by creating a new space over which one can search. Moreover, we anticipate that this difference will be amplified when the test model deviates further from the real mode, as the optimisation algorithm will exploit novel niches unrelated to the intended representation of the latent variables.

5. Conclusion

Our research demonstrates that it is possible to train a single agent encompassing all conceivable inventory management supply chain scenarios offline and then adapt it effectively when confronted with limited information in real-world supply chains. This is particularly significant for transitioning from virtual environments to actual real-world supply chains, where specific details of the supply chain setup are unknown and rapid adaptation is crucial. To advance this field further, it would be valuable to reduce the variables to search on-line, potentially down to just 2–3 by adding an intermediate layer between the context input and the distilled neural network. Thus, instead of searching over the original context space, one can search over a compressed representation of it. By doing so, the algorithm would only need to perform minimal online searching, especially if smoothness constraints are imposed on these variables. Another promising direction, as previously suggested, is to extend this analysis to multiple environments and determine if our findings can be replicated. If successful, it would indicate that we have indeed identified the appropriate approach for few-shot reinforcement learning.

CRediT authorship contribution statement

Alex Batsis: Conceptualization, Software, Writing – original draft. **Spyridon Samothrakis:** Writing – review & editing, Software, Conceptualization, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

Spyridon Samothrakis acknowledges the support of the Business and Local Government Data Research Centre BLG DRC (ES/S007156/1) funded by the Economic and Social Research Council (ESRC), United Kingdom.

References

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., et al. (2019). Solving rubik's cube with a robot hand. arXiv preprint arXiv:1910.07113.

Benjamins, C., Eimer, T., Schubert, F., Mohan, A., Döhler, S., Biedenkapp, A., et al. (2023). Contextualize me–the case for context in reinforcement learning. *Transactions on Machine Learning Research*.

De Moor, B. J., Gijsbrechts, J., & Boute, R. N. (2022). Reward shaping to improve the performance of deep reinforcement learning in perishable inventory management. *European Journal of Operational Research*, *301*(2), 535–545.

Demizu, T., Fukazawa, Y., & Morita, H. (2023). Inventory management of new products in retailers using model-based deep reinforcement learning. *Expert Systems with Applications*, *229*, Article 120256.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (pp. 1–15). Springer.

Forrester, J. (1961). *Industrial dynamics*. Pegasus Communications.

Gijsbrechts, J., Boute, R. N., Van Mieghem, J. A., & Zhang, D. J. (2022). Can deep reinforcement learning improve inventory management? performance on lost sales, dual-sourcing, and multi-echelon problems. *Manufacturing & Service Operations Management*, *24*(3), 1349–1368.

Glasserman, P., & Tayur, S. (1995). Sensitivity analysis for base-stock levels in multiechelon production-inventory systems. *Management Science*, *41*(2), 263–281.

Hallak, A., Di Castro, D., & Mannor, S. (2015). Contextual Markov decision processes. arXiv:1502.02259.

Hansen, N., Auger, A., Ros, R., Finck, S., & Pošík, P. (2010). Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *Proceedings of the 12th annual conference companion on genetic and evolutionary computation* (pp. 1689–1696). New York, NY, USA: Association for Computing Machinery.

Heiden, E., Millard, D., Coumans, E., & Sukhatme, G. S. (2020). Augmenting differentiable simulators with neural networks to close the Sim2Real gap. In *R: SS workshop on closing the reality gap in sim2Real transfer for robotics*.

Hubbs, C., Perez, H., Sarwar, O., Sahinidis, N., Grossmann, I., & Wassick, J. (2020). OR-gym: A reinforcement learning library for operations research problem.

Jiang, C., Ke, N. R., & van Hasselt, H. (2023). Learning how to infer partial MDPs for in-context adaptation and exploration. In *Workshop on reincarnating reinforcement learning*.

Kirk, R., Zhang, A., Grefenstette, E., & Rocktäschel, T. (2023). A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, *76*.

Laskin, M., Wang, L., Oh, J., Parisotto, E., Spencer, S., Steigerwald, R., et al. (2022). In-context reinforcement learning with algorithm distillation. In *The eleventh international conference on learning representations*.

Li, K., & Malik, J. (2016). Learning to optimize. In *International conference on learning representations*.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Harley, T., Lillicrap, T. P., et al. (2016). Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd international conference on international conference on machine learning - vol. 48* (pp. 1928–1937). JMLR.org.

Orhan, E., & Pitkow, X. (2018). Skip connections eliminate singularities. In *International conference on learning representations*.

Parr, T., Lapusan, T., & GroverSmith, P. (2024). Parrt/dtreeviz: A python library for decision tree visualization and model interpretation. GitHub.

Peng, X. B., Andrychowicz, M., Zaremba, W., & Abbeel, P. (2018). Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation* (pp. 3803–3810). IEEE.

Pisheh Var, M., Fairbank, M., & Samothrakis, S. (2023). A minimal "functionally sentient" organism trained with backpropagation through time. *Adaptive Behavior*, Article 10597123231166416.

Puterman, M. L. (1990). Markov decision processes. In *Handbooks in operations research and management science*: *vol. 2*, (pp. 331–434). Elsevier.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In F. Bach, & D. Blei (Eds.), *Proceedings of machine learning research*: *vol. 37*, *Proceedings of the 32nd international conference on machine learning* (pp. 1889–1897). Lille, France: PMLR.

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

Stranieri, F., & Stella, F. (2022). A deep reinforcement learning approach to supply chain inventory management. arXiv:2204.09603.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.