Informatik Spektrum (2023) 46:240–251
https://doi.org/10.1007/s00287-023-01556-6

HAUPTBEITRAG

Fully dynamic reorder policies with deep reinforcement learning for multi-echelon inventory management

Patric Hammler[1,2] · Nicolas Riesterer[1] · Torsten Braun[2]

Accepted: 4 November 2023 / Published online: 18 December 2023

Abstract
The operation of inventory systems plays an important role in the success of manufacturing companies, making it a highly relevant domain for optimization. In particular, the domain lends itself to being approached via Deep Reinforcement Learning (DRL) models due to it requiring sequential reorder decisions based on uncertainty to minimize cost. In this paper, we evaluate state-of-the-art optimization approaches to determine whether Deep Reinforcement Learning can be applied to the multi-echelon inventory optimization (MEIO) framework in a practically feasible manner to generate fully dynamic reorder policies. We investigate how it performs in comparison to an optimized static reorder policy, how robust it is when it comes to structural changes in the environment, and whether the use of DRL is safe in terms of risk in real-world applications. Our results show promising performance for DRL with potential for improvement in terms of minimizing risky behavior.

Introduction

The operation of inventory systems is a major cost driver for manufacturing companies around the world [1]. As such, their optimization is one of the key objectives for operations departments and impacts both operational cost and revenue [2].

From an optimization perspective, the optimization challenge can be understood as the task of finding an optimal policy of when to order which amount of stock in a warehouse to keep the cost of holding inventory as low as possible while making sure that incoming demand can be satisfied [3]. Inventory optimization is non-trivial because it needs to handle intricate dependencies between many cost components such as the cost of inventory keeping, reordering, and logistics as well as uncertainties inherent to the operational environment such as demand and lead time, i.e., the latency between orders and incoming shipments [4].

The literature distinguishes between two perspectives on this challenge. First, Single-Echelon Inventory Optimization (SEIO) [5] assumes independence between warehouses. By considering its demand requirements, every warehouse determines the ideal time and quantity for restocking without taking dependencies and interactions with other warehouses or distribution centers into consideration. Second, Multi-Echelon Inventory Optimization (MEIO) [6] assumes a holistic perspective in which the ideal policies are determined jointly by explicitly taking into consideration interrelationships between warehouses. According to [6], holistic optimization prevents inventory systems from making egoistic decisions at the cost of neighboring inventory systems, thus preventing solutions that may be optimal from a local perspective but not globally. However, this comes at the cost of complexity. SEIO is computationally less complex and does not require access to data from neighboring inventory systems. In sum, MEIO is the theoretically more general approach that offers the most potential for reaching optimality if computational challenges can be overcome.

The result of inventory optimization is a policy that defines the decision-making process for reordering. This process can be expressed as a parameterized policy [7] such as an RQ policy that consists of a tuple (r, Q) where r denotes the reorder point, i.e., a threshold level of Inventory on Hand (IOH) at which to order Q amount of stock. It is important to note that parameterized policies usually consider the values of r and Q as being static, which implies that they do not change naturally based on developments in the supply chain. As a result, the adoption of static param-

✉ Torsten Braun
torsten.braun@unibe.ch

[1] F. Hoffmann-La Roche, Basel, Switzerland

[2] Universität Bern, Bern, Switzerland

eterized policies neglects the risk of challenging situations that require ad-hoc coping strategies [8]. For example, logistics challenges and environmental effects can manifest in terms of limitations of manufacturing capacity (e.g., caused by scarcity of raw materials), extended distribution times, or a transitory increase in demand, possibly causing unexpected demand spikes to travel up the supply chain network, which is known as the Forrester effect [9] and therefore an unwanted drop in cost efficiency. One possibility to circumvent this issue is by relying on dynamic policies, which aim at incorporating a fundamental understanding of the dynamics of the real-world supply chain [8]. As a result, dynamic policies can adapt to the current inventory system's situation based on expectations about the future and therefore determine the best course of action for reacting to various circumstances. However, these advantages come at the cost of higher computational complexity and lacking interpretability.

The differences between parameterized and dynamic policies are illustrated in Fig. 1. The regular operation causes IOH to decrease over time to satisfy incoming demand. In parameterized policies, when crossing the threshold level of the static reorder point r, a reorder of volume Q is triggered (blue arrows). This behavior is maintained over time. In contrast, the dynamic policy relies on dynamic reorder points computed from the current IOH and expectations about the future (orange arrows).

Solving the inventory management challenge is computationally challenging because the large number of influential and partially stochastic factors such as demand and lead-time result in a high dimensional state space and make solutions analytically intractable [3]. In consequence, most proposed methods assume small-scale or highly simplified supply chains that have little relevance for real word applications [10]. It is largely unknown how well they perform when scaled to environments that mimic the complexities of the real world. Furthermore, optimization approaches in the field of inventory management are often evaluated in terms of their mean performance, which does not reflect the highly relevant perspectives for real-world applicability such as risk assessment by considering the worst-case performance.

In this article, the applicability and scalability characteristics of different state-of-the-art approaches for MEIO are evaluated. In particular, we focus on comparing dynamic parameterized policies obtained via Deep Reinforcement Learning (DRL) [11] to static parameterized policies obtained via traditional statistical optimization. Thus, our results help to shed light on the real-world applicability of state-of-the-art MEIO approaches, which, to date, have often remained elusive due to the prevailing focus on small-scale theoretically motivated experimental domains instead of business-relevant use cases in methodological research.

The following text is structured as follows. Section 2 provides a general introduction to the related work regarding supply chain optimization. In Section 3 we introduce our evaluative method along with details about the environments and models we apply. The results of the analysis are presented and interpreted in Section 4. Finally, in Section 5 we provide a general conclusion for our work and discuss its implications in terms of real-world applicability.

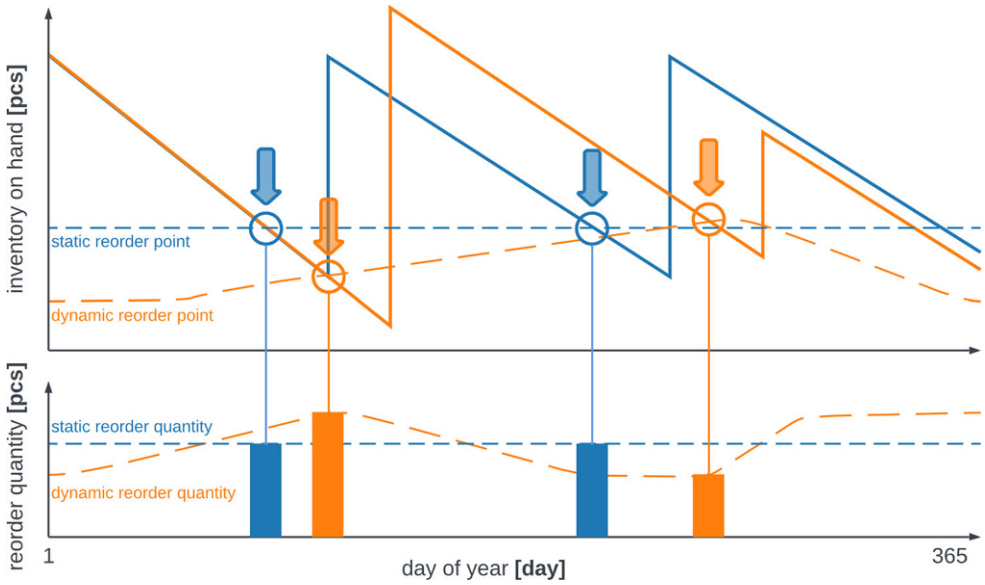

Fig. 1 The difference between a static and a dynamic reorder policy. (The *top* and *bottom* plots show the inventory on hand [*top*] and reorder quantities Q [*bottom*] on the y-axes over time in days on the x-axes. While the static reorder policy remains the same for any situation, the dynamic policy has the flexibility to adapt the strategy according to the supply chain's global state)

Background

MEIO as a Markov decision process

Mathematically, MEIOs can be expressed as Markov Decision Processes (MDP), i.e., a four-tuple ‹S,A,T,R› [12]. S denotes the state space, i.e., the set of situations s ∈ S the supply chain can be in. A is the set of actions used to control the state. In the MEIO domain, an action a ∈ A is defined on a per-warehouse level and denotes the reorder decision, i.e., whether to reorder and how much to reorder. By applying action a in state s, a transition to a new state s' is performed based on the transition probability T(s,a,s'). Finally, after transitioning between states, the reward signal R(s,a,s') is a metric for assessing the quality of an action.

Intuitively, the MDP for the MEIO problem can be interpreted as follows: The state describes the current situation in the warehouses via information such as current IOH levels, reordered but not yet delivered open order quantities, and backlogged quantities for unserved demand. The state is influenced by controllable and uncontrollable processes. *Controllable processes* such as the reorder decisions allow the warehouses to make reorders and increase the IOH level. *Uncontrollable processes* such as customer demand, the lead time, and specific warehouse properties such as the maximum inventory capacity are defined as part of the environment's dynamics. The lead time resembles the time to transport inventory from one warehouse to another warehouse. Finally, the reward structure assesses the holistic operational cost of the inventory system network.

Policies for MEIO MDPs

The goal of MEIO is to find a strategy that provides an action for any given state such that the application of the policy actions leads to minimal expected cost. This strategy is commonly termed the optimal policy for the MDP [12].

Formally, a policy π is a function $\pi: S \rightarrow A$ that maps from a state $s \in S$ to an action $a \in A$. Applied to the domain of MEIO, this implies that a policy function takes the current state of the supply chain into consideration in order to make a decision about whether and how much to reorder. In doing so, the policy has a direct effect on future states, because IOH levels will either replenish or continue to be reduced by incoming demand.

The computational challenge of MEIO is to develop efficient techniques capable of finding optimal policies. Due to the inherent stochasticity as well as the large state spaces, finding optimal analytical solutions is impossible for most practical MEIO MDPs [3]. To reduce the complexity, methods for finding simplified representations of state and action spaces have been proposed [11]. As an example for a simplified action space, the (r, Q) policy, also denoted as reorder-point lot-size policy [13], is a simple parameterized reorder policy that is based on a reorder point r and reorder quantity Q. When IOH levels fall below the reorder point r a new order of quantity Q is triggered. On the one hand, this trivial policy reduces the MEIO problem to the task of finding two static and optimal parameters for each warehouse. On the other hand, it also simplifies the observation space, as it only takes current IOH levels into consideration for making reorder decisions.

While static parameterized reorder policies are very intuitive and easy to interpret, their weakness lies in their static behavior even when situational behavior would be required. One example scenario is a distribution warehouse with a low IOH level that supplies two retail warehouses. A static policy would insist on order quantity Q in both retail warehouses. The first order would use up all inventory in the distribution warehouse thus causing the second order to not be fulfilled at all. A dynamic approach could reduce the first order to ensure the demand of both retail warehouses being fulfilled and stock-out being avoided.

Some recent research efforts proposed dynamic reorder policies providing the agent with various degrees of freedom and enabling it to base its decisions conditioned on the global state of the supply chain network. Some of them fix the reorder time but provide flexibility towards the reorder quantity [11], others provide the agent with a restricted number of reorder quantity options to select from [11], and even others force the agent to reorder on a daily basis [8]. Full flexibility is provided when the agent has the freedom to make unrestricted decisions regarding order time and quantity. Research efforts in this area are rare and require special models and optimization techniques, some of which will be discussed in the following sections.

Stochastic optimization

The most trivial option to find the optimal reorder parameters is to interpret the stochastic problem as a deterministic problem using averaged demand and lead times, thus ignoring the stochastic characteristics of the problem. This approach results in a trivial objective function but neglects the variability of the stochastic processes which may cause damage in the event of unlikely but not impossible irregularities like unexpected, very high demand, or long lead times.

Another option is to apply stochastic optimization approaches such as the simulation-based optimization framework proposed in [14]. The control variable is resembled by a set of reorder points and quantities for each inventory system in the supply chain. The target variable consists of the expected total cost to be minimized and the expected fill rates to be respected. The total cost consists of holding and reordering costs, whereas the fill rate denotes the ratio

of immediately fulfilled demand compared to the amount of backordered supplies due to stock-out. In the first step, the expected total cost and the expected fill rates are estimated for the actual control variable and some closely surrounding control variables with a Monte Carlo simulation. In order to do this, the respective control variable is applied to an episode consisting of 365 time steps resembling 1 year of operation.

The simulation considers all influential factors such as stochastic demand, stochastic lead time, and warehouse capacities. Subsequently, a linear model of the objective function is formed by linear regression, modeling the influence of the control variable on the target variables. Next, the control variables are optimized based on a cutting-plane algorithm applied to the linearized objective function. The goal of this cutting-plain algorithm is to minimize the objective function by respecting the constraints in terms of fill rates. Statistical hypothesis tests are applied to verify, whether new reorder parameters hold the predefined service level constraints and provide an improvement in terms of cost compared to the parameters from the previous iteration. This procedure is continued until termination conditions such as improvement hypothesis tests fail, or Karush-Kuhn-Tucker conditions are fulfilled.

The simulation-based approach incorporates a fundamental understanding and handling of the variability in the domain and results in optimal but static (r, Q) values for each inventory system considered. Further information can be found in [14].

Deep reinforcement learning

In order to enable dynamic reorder policies, a model is required that is capable of interpreting a high-dimensional state vector and mapping it on the corresponding optimal control vector resembling the reorder decisions [15]. Furthermore, a learning technique must be applied, that can assess the characteristics of the MDP introduced in the subsections “MEIO as a Markov Decision Process” and “Policies for MEIO MDPs”. DRL is a suitable candidate for this application as it combines both capabilities. Firstly, it leverages deep neural networks as function approximators, which are known for their high representative capacity and the capability of assessing patterns and *n*-order dependencies of unstructured input data [15]. Secondly, DRL is a trial-and-error learning technique that learns optimal policies by exploring MDP environments.

A wide range of DRL algorithms have been introduced recently, of which we use a selection of state-of-the-art approaches in this article. Commonly, DRL algorithms are categorized into on-policy and off-policy approaches. On-policy approaches learn from experience directly generated from the applied policy, whereas off-policy approaches learn from experience generated from a different policy (e.g., to improve exploration) [16].

Synchronous Advantage Actor Critic (A2C) [17] is an on-policy approach consisting of one global model and several worker models collecting independent experiences and synchronously sending gradients to the global model after a fixed number of interactions with their corresponding local environment. The fact that the experience of local workers is independent through separate environments eliminates the need for storing experience for later resampling in a replay buffer. Proximal Policy Optimization (PPO) [18] is another on-policy approach that addresses the brittleness of the objective function by constraining the potential update steps. Deep Deterministic Policy Gradient (DDPG) [19] is a deterministic off-policy approach that learns a deterministic policy aiming to increase sample efficiency by storing past experiences in a replay buffer and reusing them for learning purposes. Twin Delayed DDPG (TD3) [20] is an advancement of DDPG and addresses the performance brittleness with regard to the hyperparameters of DDPG.

Recent efforts also applied DRL to MEIO. Harsha et al. [7] propose a novel DRL approach named PARL and apply it to a stochastic demand and deterministic lead time inventory system network of different scales and compare the resulting policy with the policies found by various state-of-the-art DRL and non-DRL approaches. Gijsbrechts et al. [11] apply DRL to a MEIO challenge and thus partly enabled situational control. Whereas the order intervals are periodic and thus still fixed, the order quantities are conditioned on the global state of the supply chain.

Other semi-dynamic DRL-based reorder policies have been examined. Sultana et al. [10] apply a hierarchical multi-agent reinforcement learning approach to a multi-node and multi-product supply chain network in which the agents have the choice between multiple reorder quantity options. Perez et al. [8] provide the agent with even more flexibility and allow any reorder quantity in any time step. A DRL approach learns a deterministic policy conditioning on the global supply chain state and controlling the reorder quantities for each inventory system instance in the network. While this setup theoretically provides the DRL agent with the most flexibility, the evaluation results suggest that the policy learned is forced to apply reorders on a daily basis and is thus not feasible for business scenarios in which daily reorders are not optimal with respect to transportation cost.

Method

Objective

The core objective of this work is to investigate the general applicability of two different policy types on the problem of MEIO in a real-world setting: A static parameterized reorder policy optimized via the simulation-based framework introduced above in the section "Stochastic Optimization" and fully dynamic reorder policies learned by DRL agents. As general applicability has many facets to it, we focus on a variety of performance metrics: (1) The performance implied by the mean yearly cost over several years of simulation; (2) the algorithm robustness evaluating variation of performance across multiple optimization runs; (3) the algorithm adaptability investigating whether algorithms still perform well when there are structural changes to the environment; (4) the algorithm scalability characteristic quantifying the ability to scale up to large warehouse networks; and (5) the algorithm risk characteristic measuring the worst performance over a large number of simulated years.

Environment

The supply chain schemes considered in this work consist of three distinct types of nodes: *factory nodes, distribution warehouses*, and *retail warehouses*. Factory nodes supply the distribution warehouses, which, in turn, supply the retail warehouses. The exchange of inventory is always directed from the upper hierarchical level to the lower hierarchical level with no horizontal exchange, e.g., from one retail warehouse to another retail warehouse.

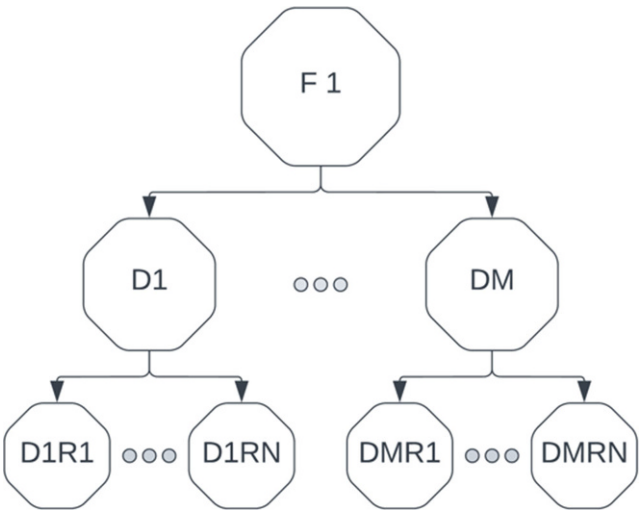

Fig. 2 The supply chain scheme structure. (The root node F1 is a single factory node supplying M distribution warehouses D1, D2, ..., DM. Each distribution warehouse supplies N retail warehouses D1R1, D1R2, ..., D1RN to DMR1, DMR2, ..., DMRN)

To simplify terminology and provide a structured analytical approach, we restrict ourselves to regular (M, N) schemes, where M denotes the number of distribution warehouses and N denotes the number of retail warehouses connected to each distribution warehouse (see Fig. 2). This implies that each (M, N) scheme consists of a total of $M \cdot N + 1$ nodes. In our analysis, we include five supply chain schemes: (1, 2), (1, 5), (1, 11), (2, 2), and (4, 2). This set of schemes provides suitable coverage of the supply chain space to gain insight into the effects of scale on algorithm performance.

In our evaluation, training and validation episodes both consist of 365 discrete time steps reflecting 1 day of operations, each. Thus, training and validation episodes stretch over 1 year of simulation, each. The key metric used to assess performance is the total annual cost resulting from the daily interaction between the environment and the policy. The transfer of goods between the supply chain nodes is modeled based on statistical assumptions. Customer demand follows a normal distribution as in [11, 21, 22], and affects the retail warehouses. To satisfy demand, retail warehouses need to periodically replenish their inventory by ordering their corresponding upstream distribution warehouse. The time between reordering and replenishment is denoted as lead time and follows a stochastic distribution as in [22]. If the number of supplies requested from the distribution level exceeds the amount of available inventory, the order quantity is reduced to the amount of available stock. To prevent this, distribution warehouses need to renew their inventory from factory nodes, which, in turn, are assumed to have unlimited stock allowing them to serve orders from distribution warehouses at any time.

Cost structure

The total cost c_{total} of inventory management operations is defined based on four distinct subtypes of cost for each inventory system i: Holding inventory without using it for sales causes storage costs and ties up capital that cannot be used for other investments. This is referred to as holding cost $c_{i,h}$. One option to keep holding costs down is to reduce the duration between reorders. However, reordering causes reorder cost $c_{i,r}$. In the event of late reordering or stock-out in the upstream inventory system, stock-outs on the retail inventory level can occur, potentially resulting in lost sales and reputational damage for the company. These costs are referred to as shortage cost $c_{i,s}$. Reordering too much, on the other hand, leads to an IOH level exceeding inventory system capacities and with this to overload cost $c_{i,o}$, requiring further processing steps up to and including disposal.

To quantify holding, reorder, shortage, and overload costs, the cost follows the following structure:

$$c_{i,h}(t) = v_{i,h} \max(0, \text{ioh}_i(t)) \quad (1)$$

$$c_{i,r}(t) = \begin{cases} 0, \text{if nothing reordered} \\ \max\left(c_{\text{mrc}}, v_{i,r} * q_i(t)\right), \text{otherwise} \end{cases} \quad (2)$$

$$c_{i,s}(t) = v_{i,s} * n_{i,ls} \quad (3)$$

$$c_{i,o}(t) = v_{i,o} * \max(\text{ioh}_i(t) - \text{cap}_i, 0) \quad (4)$$

$$c_{\text{total}}(t) = \sum_i c_{i,r}(t) + c_{i,h}(t) + c_{i,s}(t) + c_{i,o}(t) \quad (5)$$

Holding cost is either zero in the event of a stock-out or correlates with the IOH level. Inverse to this is the shortage cost, which is either zero if there is sufficient stock or increases as soon as lost sales occur due to stock-out. $n_{i,ls}$ is the number of lost sales and depends on the customer's waiting time, whereby it is assumed that the higher the waiting time, the greater the likelihood that a customer will withdraw from the purchase. If the IOH level exceeds the demand, sales are performed immediately with a waiting time of 0 days. The probability increases linearly, with 0 and 5 days leading to 0 and 100% probability of a lost sale, respectively. Reorder cost occurs only in the event of a reorder and depends on the reorder quantity, whereby the reorder cost has a lower bound of cmrc to discourage little reorder quantities. Overload cost depends on the quantity exceeding the capacity of the inventory system. Each cost type is connected to a parameter v embodying different meanings: $v_{i,h}$ denotes the holding cost and refers to the capital tie-up and storage cost per day and per item hold. $v_{i,r}$ denotes reorder cost per item reordered. $v_{i,o}$ denotes overload cost per item disposed. $v_{i,s}$ denotes shortage cost per sales lost.

Table 1 Environment parameterization used in the experiments

Variable	Unit	Value
Factory		
Inventory on hand (IOH)	Pcs	∞
Capacity (cap_i)	Pcs	∞
Overall cost	USD	0
Distribution Warehouses		
Capacity (cap_i)	Pcs	1,000,000
Lead time distribution	–	Normally distributed
Lead time exp	Days	2
Lead time std	Days	1
Min. reorder cost (c_{mrc})	USD	1000
Reorder cost coefficient ($v_{i,r}$)	USD	0
Shortage cost coefficient ($v_{i,s}$)	USD	0
Holding cost coefficient ($v_{i,h}$)	USD	8.50
Overload cost coefficient ($v_{i,o}$)	USD	120
Retail warehouses		
Capacity (cap_i)	Pcs	100,000
Demand distribution	–	Normally distributed
Daily demand exp	Days	3300
Daily demand var	Days	100
Lead time distribution	–	Normally distributed
Lead time exp	Days	2
Lead time std	Days	1
Min. reorder cost (c_{mrc})	USD	5000
Reorder cost coefficient ($v_{i,r}$)	USD	0.5
Shortage cost coefficient ($v_{i,s}$)	USD	10
Holding cost coefficient ($v_{i,h}$)	USD	0.1
Overload cost coefficient (vi,o)	USD	0.1
Max. backlog duration	Days	7

exp mean of the normal distribution, *std* standard deviation of the normal distribution, *var* variance of the normal distribution

A summary of the parameters defining the statistical distributions of the environment and cost functions can be found in Table 1.

Action and state space

The design of the action and the observation space depends on the policy type. In the example of a *static reorder policy*, the action space is constrained to the static reorder points and quantities for each inventory system in the network. The action does not condition on the situation of the warehouse and remains constant for the full period. In contrast, the *dynamic reorder policies* learned by the DRL agents are conditioned on the situation within the inventory systems. States are mathematically formalized as vectors including the current IOH, the number of reordered supplies of the oldest open order, the number of days since the oldest open order was triggered, and the number of reordered supplies of all open orders for all inventory systems in the multi-echelon network. Every day, the state is mapped to an action consisting of a dynamic reorder point and quantity for all warehouses. Because of this, the action is dynamically changing with changes in the environment's state, and the corresponding policy is thus considered dynamic.

Models and evaluation setting

We apply SimBased, A2C, PPO, DDPG, and TD3 to the respective environments. For the DRL approaches we rely on the implementations from Stable Baselines3 [23]. These are outlined in the section "Deep Reinforcement Learning".

To guarantee that each of the approaches work in its best condition, we apply extensive hyperparameter tuning using Optuna [24] to the approaches on the (1, 2) scheme.

To perform the validation, we consider the average yearly cost of 200 independent years of simulation in a repeated random sampling setup to measure the model performance during training. Ultimately, we evaluate the metrics of interest on the simulation results.

Evaluation criteria

Applicability and scalability are broad terms and are made up of many different properties. To generate a comprehensive understanding, several analyses are performed.

Training performance and robustness

After hyperparameter tuning, the resulting optimally parameterized approaches are applied to the same environment five times. The resulting performances are the foundation for determining each algorithm's general performance and robustness.

Algorithm Performance is measured by ranking algorithms based on their best evaluation result during training. *Robustness* further considers not just the best performance but all performances per scheme. In total, we apply the five algorithms stated above to five schemes resulting in 25 training runs per scheme. To tackle robustness, these 25 performances are rank ordered and used to compute an average rank per algorithm that represents robustness in comparison to the competing algorithms.

Best model performance

As part of training, we store the overall best-performing model based on evaluation scores for each algorithm-

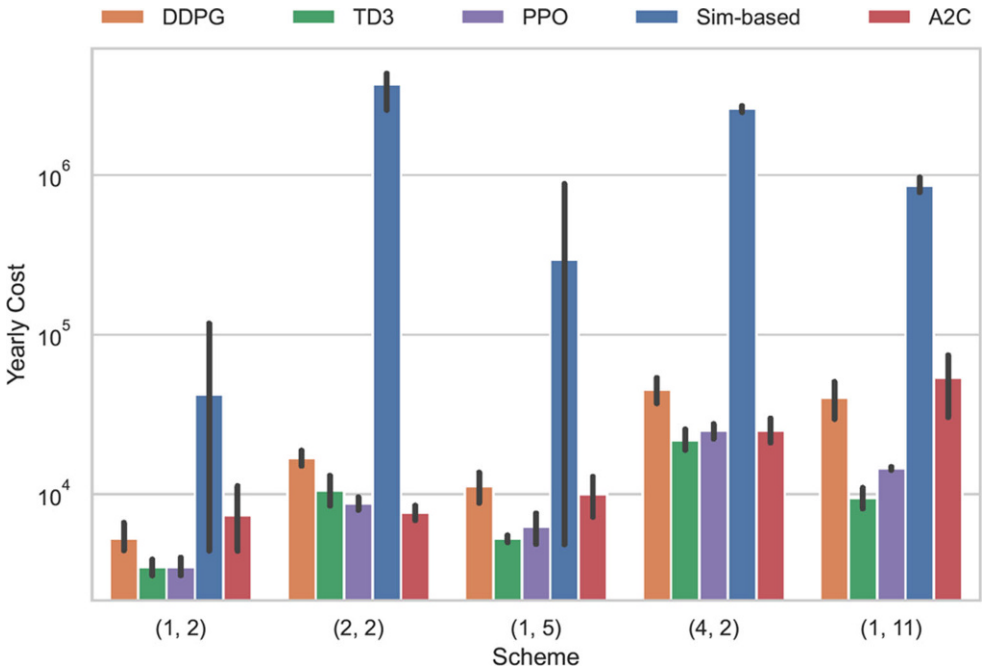

Fig. 3 Overview of algorithm performance during training. (Cost is averaged across 200 years of simulation for evaluation. *Error bars* denote the 95% confidence interval)

Table 2 Performance and robustness analysis. Performance is defined on the interval from 1 to 5 while 1 represents the algorithm with the best overall run per scheme. Robustness is defined as the mean rank for each run. Smaller values are better. Best values are highlighted in italics

	(1, 2)	(1, 5)	(1, 11)	(2, 2)	(4, 2)	Small	All
Performance							
A2C	3	4	3	*1*	3	3.5	2.8
DDPG	5	5	4	4	4	5.0	5.2
PPO	2	*1*	2	3	2	*1.5*	2.0
TD3	*1*	3	*1*	2	*1*	2.0	*1.6*
Sim-based	4	2	5	5	5	3.0	4.2
Robustness							
A2C	17.4	17.4	16.2	*4.6*	8.8	17.4	12.16
DDPG	17.0	20.0	14.8	17.8	18.0	18.5	17.52
PPO	6.6	10.0	8.0	8.0	9.8	8.3	8.48
TD3	*6.0*	*6.6*	*3.0*	11.6	*5.4*	*7.3*	*5.52*
Sim-based	18.0	11.0	23.0	23.0	23.0	14.5	19.60

A2C Synchronous Advantage Actor Critic, *DDPG* Deep Deterministic Policy Gradient, *PPO* Proximal Policy Optimization, *TD3* Twin Delayed *DDPG*

scheme pair. To analyze model performances, these model instances are applied to 1000 years of simulation from which cost data are extracted. The mean and maximum yearly costs serve as estimates for the average and worst-case performances.

Policy behaviors

We analyze the behavioral structures of the results to identify differences between regular and irregular situations the policies encounter during simulation. Taking the 1000 years of simulation from the model performance analysis above, we extract the top 50 years based on minimum total cost. In this subset of data, we identify whether substantial cost occurs at the level of distribution or retail warehouses. Inspecting the best years of simulation provides additional information about how regularly or robustly the models respond to suitable situations. On the other hand, the worst years of simulation will shed light on how well the policies can cope with irregular situations such as those featuring low starting IOH.

Scheme difficulty

Lastly, scheme difficulty is analyzed to investigate to what extent the scheme architecture and size impact the resulting cost.

Results

Performance and robustness of optimization algorithms

The yearly cost per scheme and run per optimization approach is visualized in Fig. 3. In this analysis, the main focus is identifying the best model per algorithm and scheme combination and interpreting its best validation result as algorithm performance. Furthermore, taking all runs and their corresponding best yearly average performances into account provides insight into the algorithm robustness. Table 2 emphasizes these metrics for each algorithm and scheme combination. To give a better overview of the impact of scheme size on results, we also provide summaries of the results for small schemes, which include the (1, 2) and the (1, 5) scheme, as well as a total for all schemes.

Comparing the dynamic and static policies, the results clearly show that the flexibility introduced by the DRL agents has a positive impact on both performance (mean scores of 2.9 vs. 4.2) and robustness (mean scores of 10.195 vs. 18.60). With the exception of the (1, 5) scheme where Sim-based manages to outperform the on-policy DRL algorithms A2C (by 2 for performance and 6.4 for robustness) and DDPG (by 3 for performance and 9 for robustness), it performs worst in all other scenarios.

Within the class of on-policy algorithms, the difference between A2C and PPO is only marginal. This can be explained by the fact, that both approaches are similar and with certain hyperparameter settings even identical [25].

Clear differences can be found in the class of off-policy DRL algorithms, where TD3 performs well with regard to

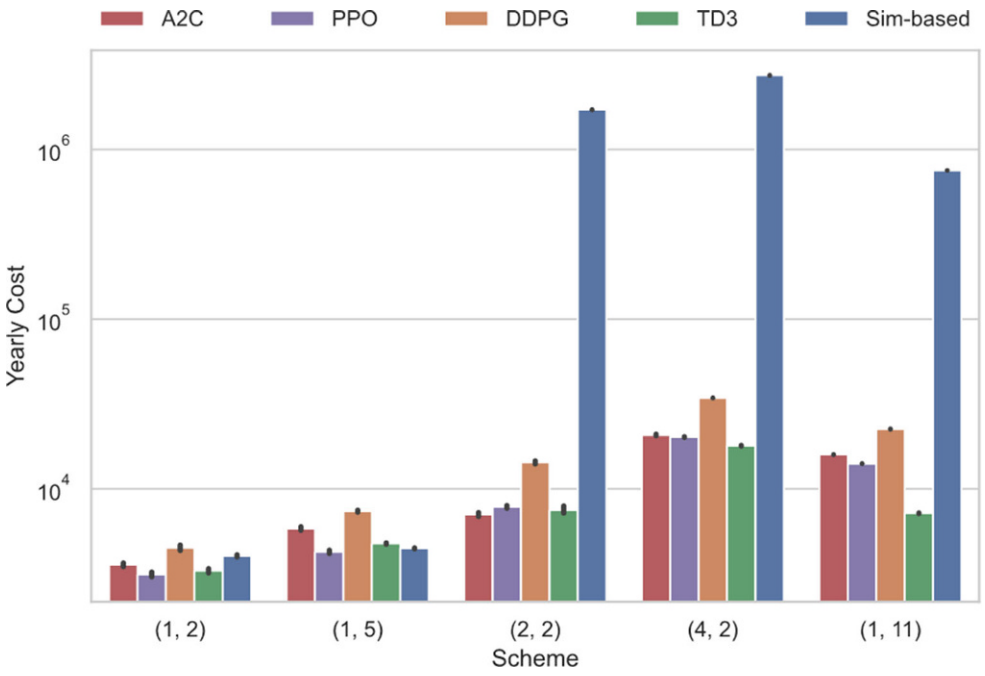

Fig. 4 Illustration of the yearly cost for the best model found by each algorithm for each scheme. (The number of simulated years is 1000, whereas the *error bars* denote the 95 certainty interval.) *A2C* Synchronous Advantage Actor Critic, *PPO* Proximal Policy Optimization, *DDPG* Deep Deterministic Policy Gradient, *TD3* Twin Delayed *DDPG*

Table 3 Overview of the mean and worst model performance per algorithm and scheme combination. Best results (average-case and worst-case) are highlighted in italics

Type	(1, 2)	(1, 5)	(1, 11)	(2, 2)	(4, 2)
A2C					
Average	3.6E + 03	5.8E + 03	1.6E + 04	*7.1E + 03*	2.1E + 04
Worst	7.9E + 03	3.4E + 04	1.8E + 04	*3.0E + 04*	5.5E + 04
DDPG					
Average	4.5E + 03	7.4E + 03	2.3E + 04	1.4E + 04	3.4E + 04
Worst	1.9E + 04	1.8E + 04	2.6E + 04	5.9E + 04	3.7E + 04
PPO					
Average	*3.1E + 03*	*4.3E + 03*	1.4E + 04	7.8E + 03	2.0E + 04
Worst	8.4E + 03	3.7E + 04	1.6E + 04	3.2E + 04	4.4E + 04
TD3					
Average	3.3E + 03	4.8E + 03	*7.2E + 03*	7.5E + 03	*1.8E + 04*
Worst	*7.1E + 03*	9.4E + 03	*1.5E + 04*	1.8E + 05	*2.4E + 04*
Sim-based					
Average	4.0E + 03	4.5E + 03	7.5E + 05	1.7E + 06	2.7E + 06
Worst	7.2E + 03	*6.4E + 03*	7.9E + 05	1.9E + 06	3.0E + 06

A2C Synchronous Advantage Actor Critic, *DDPG* Deep Deterministic Policy Gradient, *PPO* Proximal Policy Optimization, *TD3* Twin Delayed *DDPG*

performance and robustness, while DDPG performs poorly. The difference could be due to the fact that TD3 is an evolution of DDPG adding improvements to counteract the problem of hyperparameter brittleness. Structural changes to the environment, such as changing the network scheme, are adjustments that TD3 can handle much better even without reconfiguring its hyperparameters.

Putting the results from off- and on-policy algorithms together, it is difficult to identify a clear winner. In the context of performance, TD3 and PPO, the best off- and on-policy approaches, respectively, perform similarly (1.6 vs. 2.0). However, when focusing on robustness, TD3 appears to slightly outperform PPO (5.52 vs. 7.48). This suggests that the statistical properties of supply chain optimization might benefit from the off-policy approach that tries to circumvent issues of time-correlated update samples via random batching from an extensive episodic memory.

Best model fitness

From a real-world applicability perspective, the question regarding mean and worst-case cost is highly relevant. To

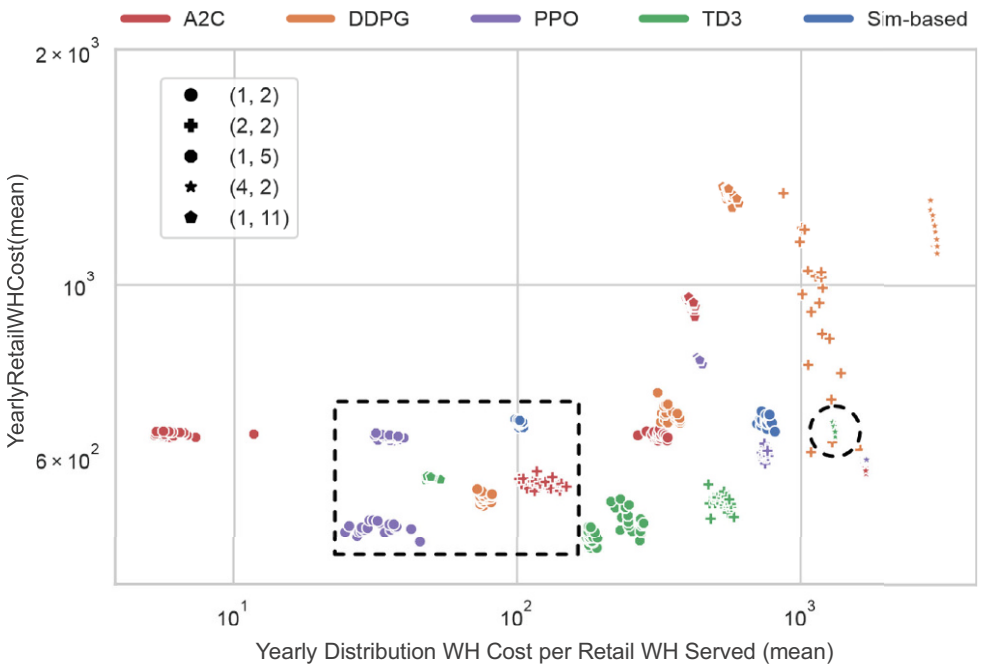

Fig. 5 The normalized cost for each scheme and algorithm combination. *A2C* Synchronous Advantage Actor Critic, *DDPG* Deep Deterministic Policy Gradient, *PPO* Proximal Policy Optimization, *TD3* Twin Delayed *DDPG*

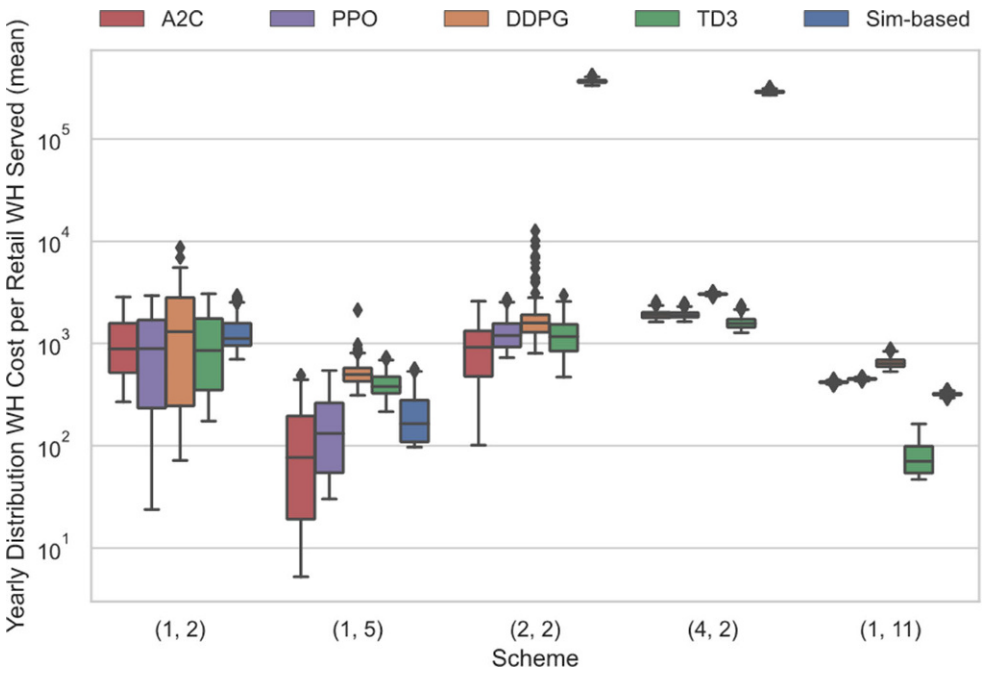

Fig. 6 Normalized mean yearly distribution warehouse cost per retail warehouse served for each algorithm and scheme combination. *A2C* Synchronous Advantage Actor Critic, *PPO* Proximal Policy Optimization, *DDPG* Deep Deterministic Policy Gradient, *TD3* Twin Delayed *DDPG*

measure the cost expectation and the reliability of an optimization approach the best model resulting from the performance and robustness evaluation (see the Section "Performance and Robustness of Optimization Algorithms") is taken for each algorithm and scheme combination and is applied to 1000 years of simulation. Consequently, the mean yearly cost and the worst yearly cost are considered and displayed in Fig. 4 and in Table 3.

The results corroborate the conclusion drawn from the algorithm performance and robustness analysis above. Again, the dynamic DRL-based policies outperform the static policy of Sim-based. The exception to this is the (1, 5) scheme in which Sim-based achieves the optimal worst-case model performance (also note that Sim-based ranks second for the [1, 2] scheme). This suggests that for small-scale environments, Sim-based might provide a practically useful trade-off between performance and safety in terms of the policy not behaving surprisingly bad.

The on-policy (A2C and PPO), as well as the off-policy approaches (TD3 and DDPG), show results that mirror the results observed in the algorithm performance and robustness analysis above. In the case of on-policy algorithms, both algorithms perform similarly well, thus emphasizing the relationship between both algorithms. For off-policy algorithms, TD3 performs better both in terms of average- and worst-case model performance, again highlighting the benefits of its extension over DDPG. In sum, however, the results again favor off-policy over on-policy approaches.

Policy analysis

The focus is now directed to the question of how the policy strategies deviate and what causes the performance differences. Figure 5 shows the trade-off between expenses on the distribution warehouse level and retail warehouse level. Each point represents the trade-off for 1 year of simulation, whereas the color and point shape denotes the algorithm and scheme.

The points are all clustered closely together, which suggests that the algorithms are developing a consistently performing strategy. It is notable that DDPG occupies the most space. Together with the finding that DDPG usually also has the worst performance, this suggests that consistent strategies correlate with low cost. Furthermore, it is striking that the best model results for each scheme are located closely together (within the dashed box) except for the (4, 2) scheme (within the dashed circle). This highlights that the increased branching from a large number of distribution warehouses causes a particularly impactful increase in complexity.

Scheme difficulty

The previous section highlighted policy differences in terms of warehouse efficiency and cost allocation trade-offs, which raises questions about their causes. One potential explanation is that the difference is caused by the higher complexity of the network architecture that does not allow for lower-cost solutions. Alternatively, the policy could diverge from the optimal solution because of increased complexity from an optimization perspective. Thus, we

analyze the scheme difficulty via the mean yearly distribution warehouse cost per retail warehouse served next. This metric of interest is compared across all schemes and the following observations can be made:

Firstly, the variability of mean cost is higher for small schemes compared to large schemes. As the observation and action space dimension and hence the optimization complexity is lower for small schemes, the variability of the cost suggests that warehouses react more flexibly in special situations and allocate cost to the warehouse level depending on the requirements of a specific situation. With the variability decreasing for large schemes, this suggests that situational decision-making collapses into a static policy for more complex schemes.

Secondly, the operational cost for distribution warehouses is lowest for the (1, 5) scheme followed by the (1, 11). With the (1, 2) scheme being the simplest scheme related to observation and action space dimension and the (1, 5) scheme outperforming the (1, 2) scheme in terms of distribution warehouse efficiency, there is evidence that the scheme architecture has an impact on the cost efficiency.

Conclusions

In this paper, the applicability of algorithms learning a dynamic reorder policy and an algorithm finding a static policy was evaluated and compared. The results show that the dynamic reorder policies outperform the static reorder policy from an algorithm performance and robustness perspective. Furthermore, having optimized the hyperparameters for each approach for the (1, 2) scheme, the DRL-based algorithms even outperform the Sim-based algorithm on larger schemes, which implies higher scheme flexibility and robustness in terms of structural environment changes. From a model analysis perspective, the results showed that the Sim-based approach works decently (Fig. 6) in two out of five schemes. In those where it was successful, the worst yearly performance shows the best or close-to-best performance, implying that static reorder policies are suboptimal with regard to average-case performance but offer high security guarantees due to their explainable and simple nature. This is underlined by TD3, which is the overall best algorithm in terms of performance and robustness but shows a poor worst-case performance on the (2, 2) scheme. These results highlight the need to add an extra layer of security when applying DRL to real-world systems. For example, one could introduce a static emergency RQ policy into dynamic DRL policies to provide basic safety guarantees.

The investigation of the potential of DRL for industry applications is still in its early phases. To provide convincing arguments for the promises of performance and flexibility made by DRL, it has to be further analyzed how close these approaches can approximate optimality in comparison to traditional optimization approaches. Similarly, the limitations with regards to explainability and interpretability due to the reliance on deep neural network models need to be overcome to better understand when DRL models need to be extended with carefully hand-crafted guardrails preventing worst-case scenarios from happening.

In summary, this work identifies the high potential for real-world application of DRL-based reorder policies. However, some limitations with regard to the reliability in terms of worst performance and the need for additional security layers in conjunction with real-world applications are uncovered, which underline the need for further research efforts in this domain.

Acknowledgements We thank Yuri Pyatkin, Meisam Asgari, and Tomasz Zdanowicz for supporting us in the implementation of the core evaluation framework underlying this work. We also thank Kürsat Petek, Igor Kulev, and Martin Kuhn for valuable discussions and assistance with the manuscript. This paper was supported by F. Hoffmann-La Roche.

Funding Open access funding provided by University of Bern

Conflict of interest P. Hammler, N. Riesterer, and T. Braun declare that they have no competing interests.

References

1. Greer BM, Theuri P (2012) Linking supply chain management superiority to multifaceted firm financial performance. J Supply Chain Manag 48(3):97–106
2. Matinheikki J, Kauppi K, Brandon-Jones A, Raaij EM (2022) Making agency theory work for supply chain relationships: a systematic review across four disciplines. Int J Oper Prod Manag 42(13):299–334
3. Kok T, Grob C, Laumanns M, Minner S, Rambau J, Schade K (2018) A typology and literature review on stochastic multi-echelon inventory models. Eur J Oper Res 269(3):955–983
4. Hammler P, Riesterer N, Mu G, Braun T (2023) Multi-echelon inventory optimization using deep reinforcement learning. In: Canci JK, Mekler P, Mu G (eds) Quantitative models in life science business: from value creation to business processes. Springer, Cham, pp 73–93 https://doi.org/10.1007/978-3-031-11814-2_5
5. Hausman WH, Erkip NK (1994) Multi-echelon vs. single-echelon inventory control policies for low-demand items. Manage Sci 40(5):597–602

6. Clark AJ, Scarf H (1960) Optimal policies for a multi-echelon inventory problem. Manage Sci 6(4):475–490
7. Harsha P, Jagmohan A, Kalagnanam JR, Quanz B, Singhvi D (2021) Math programming based reinforcement learning for multi-echelon inventory management (arXiv preprint arXiv:2112.02215)
8. Perez HD, Hubbs CD, Li C, Grossmann IE (2021) Algorithmic approaches to inventory management optimization. Processes 9(1):102
9. Metters R (1997) Quantifying the bullwhip effect in supply chains. J Oper Manag 15(2):89–100
10. Sultana NN, Meisheri H, Baniwal V, Nath S, Ravindran B, Khadilkar H (2020) Reinforcement learning for multi-product multi-node inventory management in supply chains (arXiv preprint arXiv:2006.04037)
11. Gijsbrechts J, Boute RN, Van Mieghem JA, Zhang D (2021) Can deep reinforcement learning improve inventory management? performance on dual sourcing, lost sales and multi-echelon problems. Manuf Serv Oper Manag. https://doi.org/10.1287/msom.2021.1064
12. van Otterlo M, Wiering M (2012) Reinforcement learning and markov decision processes. Reinforcement Learning. Springer, pp 3–42
13. Chen FY, Wang T, Xu TZ (2005) Integrated inventory replenishment and temporal shipment consolidation: a comparison of quantity-based and time-based models. Ann Oper Res 135(1):197–210
14. Chu Y, You F, Wassick JM, Agarwal A (2015) Simulation-based optimization framework for multi-echelon inventory systems under uncertainty. Comput Chem Eng 73:1–16. https://doi.org/10.1016/j.compchemeng.2014.10.008
15. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA (2017) Deep reinforcement learning: a brief survey. IEEE Signal Process Mag 34(6):26–38
16. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction, 2nd edn. MIT Press
17. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap TP, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning
18. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms https://doi.org/10.48550/ARXIV.1707.06347
19. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2015) Continuous control with deep reinforcement learning https://doi.org/10.48550/ARXIV.1509.02971
20. Fujimoto S, Hoof H, Meger D (2018) Addressing function approximation error in actor-critic methods. International conference on machine learning, pp 1587–1596
21. Van Roy B, Bertsekas DP, Lee Y, Tsitsiklis JN (1997) A neuro-dynamic programming approach to retailer inventory management. Proceedings of the 36th IEEE Conference on Decision and Control, vol 4. IEEE, pp 4052–4057
22. Axsäter S (2006) A simple procedure for determining order quantities under a fill rate constraint and normally distributed lead-time demand. Eur J Oper Res 174(1):480–491
23. Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M, Dormann N (2021) Stable-baselines3: reliable reinforcement learning implementations. J Mach Learn Res 22(268):1–8
24. Akiba T, Sano S, Yanase T, Ohta T, Koyama M (2019) Optuna: a nextgeneration hyperparameter optimization framework. Proceedings of the 25rd ACM SIGKDD international conference on knowledge discovery and data mining, pp 2623–2631
25. Huang S, Kanervisto A, Raffin A, Wang W, Ontañón S, Dossa RFJ (2022) A2c is a special case of ppo (arXiv preprint arXiv:2205.09123)