Daniil Sideris

Applying reinforcement learning algorithms to multi-echelon supply chain inventory management with uncertain seasonal demands

Bachelor Thesis
in the Area Supply Chain Management
Supervised by: Matthias Lehmann

Keywords: Multi-echelon Supply Chain, Operations Research, Seasonality, Machine Learning, Reinforcement Learning

Bachelor of Science (B. Sc.) in Business Administration
Faculty of Management, Economics and Social Sciences
University of Cologne

Cologne, January 2, 2023

Contents

List of Figures

List of Tables

List of abbreviations

A2C	Advantage Actor-Critic
A3C	Asynchronous Advantage Actor Critic
ARMA	Autoregressive Moving Average
ARS	Augmented Random Search
BO	Bayesian Optimization
KL	Kullback-Leibler
MDP	Markov Decision Process
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
SAC	Soft Actor-Critic
SARMA	Autoregressive Moving Average
SB3	Stable Baselines3
TD	Temporal Difference
TRPO	Trust Region Policy Optimization
VPG	Vanilla Policy Gradient

1. Introduction

The term "Supply chain" can be explained in various ways. One way explains supply chain as a "network of organizations that are involved, through upstream and downstream linkages, in the different processes and activities that produce value in the form of products and services in the hands of the ultimate consumer" (Christopher 2005). A supply chain is organized with two or more entities or companies, which are connected by material, financial and information flows. The entities of a supply chain have different roles as they represent raw materials, components from the raw materials and the end products. The entities are connected with a logistic system and the ultimate demand is represented by the customer. It is important to highlight that the demand and requests of the entities from each other depend on the market demand (ultimate customer).

A multi-echelon supply chain network is more complex than a single-line supply chain. It's focus is on a network built from two or more echelons and therefore the materials and end products can be delivered and requested from different organizations and even flow parallel or simultaneously. The organization are located in various geographical areas and have different working hours, delivery times, transportation and financial, physical or human resources.

Since a network can be simple or complex it is critical to decide the right replenishment quantity of each product or material, choose a supplier to order from, plan the logistics and quantity of products to sell. Every supplier has to consider the costs that come with each decision.

The suppliers and the whole supply chain networks face numerous challenges and tradeoffs, which hinder smooth, cost-efficient and on-time processes. Three main challenges occur: (i) The first challenge is the problem of under-stocking, meaning an organization's availability of goods to deliver is smaller than the demand of other organizations or customers further in the supply chain. Demand variability, delays or other events can be the cause of under-stocking. For the businesses it means lost profits and opportunity costs or even less customer loyalty. (ii) Overstocking of goods forms the second problem. Overstocking means that an organization has more goods available than the market or other suppliers are demanding at the current state. The cause of this challenge is dependant on the warehouse storage resources and product durability. Too many goods in a warehouse create holding costs and some goods become unsuitable for delivery and use after a period of time. (iii) The third challenge is computational resources and high operating costs. An organization has to handle multiple products for shipment and storage, which can be delivered on various transport types to different end stages. Therefore, the calculation become complex and demanding.

Uncertainty of the different variables can be studied to optimize the reaction of a supply chain network. Uncertain demands may trigger the challenges mentioned above. This paper addresses to the topic of seasonal demands. A seasonal cycle is common in corporate demand processes and it usually repeats itself after a set amount of time (Wei et al 1990). Seasonal cycles are frequently observed in the supply chains of the food, textile, and travel sectors. A variety of reasons cause seasonality, including how a supply chain manages certain processes creating an seasonality pattern internally, and handles external elements such as the weather (Makridakis et al 1989).

Classical operations research can deal with this challenges to find an optimization of the inventory and network management. But is limited to the use cases, and is itself challenging in adaptation to changes of demand, organizations operating in the the supply chain network and other events. Data-driven and learning-based methods on the other hand tend to explore in a larger space and adapt to changes in the implemented or simulated system (Sultana et al 2020). Reinforcement learning has recently been used to address a number of difficult issues in a variety of areas, including robots, video games, health care, finance, transportation systems and industry 4.0 (Li, 2017). Nevertheless, in comparison with data-driven and learning-based methods, as for example supervised deep learning, deep reinforcement learning is not often applied to inventory management problems. However, applications of reinforcement learning show significant results and tent to be applicable in challenging situations (Boute et al 2021).

In this paper we will apply reinforcement learning algorithms to multi-echelon supply chain inventory management with uncertain seasonal demands. The focus of this work and the research question investigates the various algorithms and their performance on operations research problems with seasonality. Algorithms used in this work cover: Trust Region Policy Optimization (TRPO), Advantage Actor-Critic (A2C), Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC) and Augmented Random Search (ARS). The algorithms are applied to a simulation model of a multi-echelon network.

The structure of this paper is as follows. In the first part of this study we will review the background and related work on supply chain models, seasonality in operations research and reinforcement learning methods related to seasonality. Afterwards, the problem and simulation model for this paper is presented. Another point will be solution methodology explaining the used reinforcement learning algorithms and the formulation of the empirical study as an reinforcement learning problem. In section 5 we will introduce the results of the study. The last part covers the discussion and ends with a conclusion of this work.

2. Literature overview

Before applying and presenting the problem, methodology and results, we view the background and related literature of supply chain and reinforcement learning. Literature is classified into (i) supply chain modelling, (ii) seasonality in inventory problems and (iii) applications of reinforcement learning in the area of supply chain management.

2.1. Supply chain modelling

The topic of supply chain optimization has various approaches. In the field of inventory management, the majority of the literature can be divided into two categories. A single supply chain node with multiple products is the focus of the first series of studies. For instance, the work of Smith et all (2000) provide an probabilistic demand model for substitute products. Another case is the work of Caro et al (2010) showing a probabilistic model including one product of different sizes in a single store. Later on Caro et al (2010) apply the model to stores of the spain-based retailer Zara. Their research has resulted in an additional revenue increase of 275 Million US-Dollars (USD).

The second category has it's focus on multi-echelon supply chain for a single product. For example, the study of support models at Hewlett-Packard Company (Lee et al 1993) or the mathematical inventory model (Nahmias et al 1993) and the two-echelon retailer supply chain influenced by partial lost sales (Nahmias et al 1994).

Generally, when we speak about Reinforcement learning methods for multi-echelon supply chains, they are often based on Sterman's et al (1989) MIT beer distribution game. With the help of local data, Sterman et al created this supply chain game to investigate the bullwhip effects in a single-line, four-echelon supply chain. To develop studies on operational research, there have been various ways of implementing both categories based on the MIT beer distribution game. Another source of extended simulation models is the open-source library OR-Gym (Hubbs et al 2020) for developing reinforcement learning algorithms. The library itself is based on the reinforcement learning interface Gym from OpenAI (Brockmann et al 2016). The OR-Gym library includes different supply chain problems based on studied of others researchers, including: (i) knapsack, (ii) multi-dimensional bin packing (Balaji et al 2019), (iii) newsvendor problem, (iv) Virtual machine packing problem, (v) Vehicle Routing (Balaji et al 2019), (vi) multi-echelon supply chain re-order problem with and (vii) without backlogs, (viii) multi-echelon supply chain network problem with and (ix) without backlogs (Perez et al 2021), (x) Multi-period asset allocation problem for managing

investment decisions (Dantzig et al 1991), (xi) traveling salesman problem with bi-directional. A multi-echelon, multi-period single market supply chain model (model (viii)) is developed by Perez et al (2021). This model is presented in section 3 and forms the groundwork of this study.

2.2. Seasonality in inventory problems

Both research approaches have different focus areas. For this work the second category is the centre of attention. The second class of supply chain models deals with the so-called bullwhip effect. It is well recognized that the bullwhip effect is significantly influenced by the demand process, replenishment lead times, inventory policies, and the forecasting methodologies used (Lee et al. 1997). According to the majority of earlier studies, these factors can all be controlled, making them suitable for determining how to reduce the bullwhip impact (Duc et al. 2008). It is acknowledged, though, that the demand process, which takes place at the customer level, is unavoidable.

Having a closer look at studies considering seasonality and seasonal supply chains, they come to different findings. On the one hand, the work of Cho et al (2011) studied the bullwhip effect in seasonal supply chains. They mention an existing gap of works on the topic of seasonality and non-stationary demand process. A seasonal supply chain frequently prevents the effective application of non-seasonal demand processes. Therefore, it requires the use of a seasonal demand procedure. To investigate this problem, they implemented a non-seasonal Autoregressive Moving Average (ARMA) and a seasonal ARMA demand process using the Seasonal Autoregressive Moving Average (SARMA) demand process. The SARMA demand process uses the minimum mean-square error forecasting technique. Regarding the findings of their implementation, they show that the bullwhip effect is effected by seasonal moving coefficient and the seasonal cycle. In particular, the bullwhip effect is continuously increasing or decreasing if the lead time is less than the seasonal cycle. Additionally, the autoregressive coefficient and lead times influence the bullwhip effect.

In addition, the bullwhip effect and inventory stability in seasonal supply chains was addressed in the work of Costantino et al (2013). The analysis has been done using a simulation in a four-echelon supply chain with a moving average method on the base stock ordering policy. The foundings differ in terms of the importance of seasonality. On the one hand, ratios like average fill rate, inventory variance ratio and bullwhip effect ratio are reduced in the case of high seasonality levels. This is true especially with low demand noise. On the other hand, sensitivity of those ratios decreases with seasonal demands and high noise. Constantino

et al suggest that using ratios to quantify seasonal supply chain dynamics is inaccurate and that it is preferable to use the variance directly as the estimates for the bullwhip impact and inventory performance without dividing by the demand variation.

Although the research question of this work doesn't concern information sharing, the findings of Constantino et al demonstrate that, regardless of the degree of seasonality, forecasting and safety stock characteristics have a significant impact on supply chain performance. Additionally, the quantification of the impact of information sharing reveals that, independent of demand seasonality, all performance parameters have improved.

After reading through various research studies, it is clear that seasonality is a topic to be further investigated in the area of operation research optimization.

2.3. Reinforcement learning

In addition to the related work mentioned above, real-world operations research problems frequently involve parameter uncertainty, which is typically managed by safety margins and flexibility buffers, however this results in unused excess capacities and inventories (Seelenmeyer et al 2020). An alternate solution to this problem is machine learning, which has been used in many different challenging domains. Recent developments have expanded and benefited from their implementation, particularly when deep neural networks were used. A branch of machine learning called reinforcement learning is used to handle sequential decision-making issues with uncertainty. This problem can be expressed as a Markov Decision Process (MDP), however high dimension of the state space in the resulting model prevents it from being addressed numerically (Laumanns et al 2017). Another example are the applications of RL on two-echelon supply chain networks or serial supply chains (Kemmer et al. 2018; Hutse et al. 2019; Peng et al. 2019)

Reinforcement learning has been used for supply chain operation issues in certain works, but many of them are based on tabular RL approaches, such as Q-Learning (Giannoccaro and Pontrandolfo 2002; Chaharsooghi et al. 2008; Mortazavi et al. 2015). Some research studies based on deep reinforcement learning achieve reasonable well results. For instance, A2C outperfoms heuristic methods in the study of inventory management problem with multiple products (Sultana et al 2020).

Although, many studies implement reinforcement learning, the aspect of seasonality hasn't been addressed that often in a supply chain network. One study applies the PPO algorithm to production planning and distribution in multi-

echelon supply chains (Alves et al. 2022). The PPO is compared against a suggested linearized model. An important finding is viability of reinforcement learning given high uncertainty, such as seasonality. Another approach with consideration of seasonal demands in a two-echelon supply chain network compares Asynchronous Advantage Actor Critic (A3C), Proximal Policy Optimization, Vanilla Policy Gradient (VPG), Bayesian optimization (BO) and Oracle algorithms (Stranieri et al. 2022). PPO, in particular, has the best performance averaging bigger profits than other algorithms even if it is unable to turn a profit in the most difficult trials. Particularly as the number of warehouses rises, VPG frequently seems to converge to a local maximum that is somewhat distinct from PPO, but it still achieves respectable results. The A3C is always the quickest algorithm in the study. However, the authors noted, that unlike VPG, A3C is never the best-performing algorithm, possibly because to its increased sensitivity to the challenging job of hyperparameter tuning. The studies indicate to further research algorithms with seasonal demands and other suggested extensions.

3. Problem description

After a background and literature review of the current development supply chain models, seasonality and reinforcement learning is given, the empirical study of this paper can be formalized. Section 3 presents the supply chain model which will later be applied to our reinforcement learning algorithms.

3.1. Problem description

In this paper, the inventory management problem can be viewed as a multi-echelon, multi-period single market working with a single nonperishable product. The model for analysis is based on the work of Perez et al (2021). They purposed a simulation of a multi-echelon network and published an stochastic environment that is based upon the open-source OR-Gym Python package as mentioned in section 2.1. Key parameters were tweaked to study the use case of seasonality.

The supply chain network consists of four echelons connecting eight entities and one market (ultimate customer) as shown in the figure below (see figure 1). The first entity is the ultimate customer J_{market} followed by the retailer entity J_{retail}, two distributors J_{dist}, three production stages J_{prod} ending with two raw material nodes J_{raw}. All entities are also stored in the J variable, except the ultimate customer and raw material nodes.

The nodes can be divided into different types. The main nodes are: (i) production nodes having and inventory holding area and a manufacturing area, and (ii)

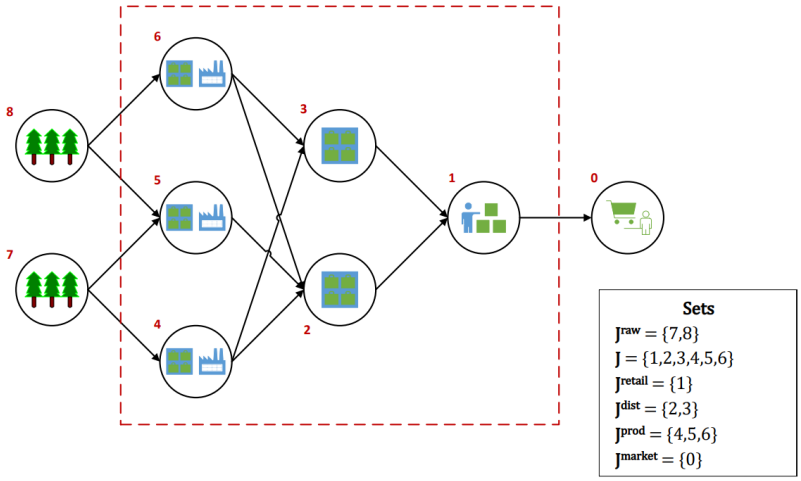

Figure 1: Supply Chain Network Schematic (Perez et al 2021)

distribution nodes with an inventory holding area only. The retail nodes can be viewed as distribution nodes. Source nodes are represented by the raw material nodes with unlimited supply of raw materials. The market (ultimate customer) is the so-called sink node and is generating an uncertain or user defined demand on their respective retailers in each period.

The production nodes as stated have an inventory holding area in which the inventory necessary to produce is kept to the respective intermediate material at that node. At each production stage the yield rations relating to the amount of material produced from one unit of inventory are defined. The specific parameters for the nodes (entities) used, can be observed in the table (see table 1). Table 2 shows the specific parameters for the edges (links or connections) between the entities.

Entity	Description	I_0	c	v	o	h
J_0	Market					
J_1	Retailer	100				0.030
J_2	Distributor	110				0.020
J_3	Distributor	80				0.015
J_4	Manufacturer	400	90	1.000	0.010	0.012
J_5	Manufacturer	350	90	1.000	0.015	0.013
J_6	Manufacturer	380	80	1.000	0.012	0.011
J_7	Raw materials					
J_8	Raw materials					

Table 1: Node specific parameters (default values)

abbr.: I_0 = initial inventory; c = production capacity; v = production yield in the range (0, 1] ; o = unit operating cost (feed-based); h = unit holding cost for excess on-hand inventory

Link	Description	L	p	g	b
1,0	Retailer-market		2.000		0.1000
2,1	Distributor-retailer	5	1.500	0.010	
3,1	Distributor-retailer	3	1.600	0.015	
4,2	Manufacturer-distributor	8	1.000	0.008	
4,3	Manufacturer-distributor	10	0.800	0.006	
5,2	Manufacturer-distributor	9	0.700	0.005	
6,2	Manufacturer-distributor	11	0.750	0.007	
6,3	Manufacturer-distributor	12	0.800	0.004	
7,4	Raw materials-manufacturer	0	0.150	0.000	
7,5	Raw materials-manufacturer	1	0.050	0.005	
8,5	Raw materials-manufacturer	2	0.070	0.002	
8,6	Raw materials-manufacturer	0	0.200	0.000	

Table 2: Edge specific parameters (default values)
abbr.: L = lead times in between adjacent nodes; p = unit price to send material between adjacent nodes (purchase price/reorder cost); b = unit backlog cost for unfulfilled market demand between adjacent retailer and market; g = unit holding cost for pipeline inventory on a specified edge

The additional aspect of production at each node comes from the node's production capacity and the available inventory. Lead times between neighbor nodes exist and are associated with the edges connecting them.

3.2. Event series

The following series of events occur at the beginning of each time period. In this paper the option with backlogs for the unfulfilled sales is considered (Option 4a).

1. At the beginning of the time period producers, distributors, and retailers who make up the main network nodes, order supplies from their respective suppliers. Orders for replenishment are fulfilled in accordance with the various suppliers' available production capacity and feedstock inventories. In order for replenishment orders to never exceed what the suppliers can provide to each node, it is assumed that the supply network is centralized.

2. In the second stage the available orders of inventory replenishment are shipped through the main network nodes (with consideration of the respective lead times). The lead times involve both times for production and transportation.

3. The next step is at the retail node, where the customer demand occurs for the single-product. The demand is covered depending on the inventory of the retailers that is available at that stage.

4. It is challenging to fulfill all sales, so the retailers face two options of dealing with unfulfilled sales:
    a) Sales that aren't completed cause backlogs at a cost. In the following time period the backlogged sales take priority.
    b) Sales that aren't completed are considered lost and a penalty in form of a goodwill loss is charged

5. Continuing on each node pays a holding cost to keep excess inventory. Holding capacity limits of the inventory are not included in the model, but they can simply be added if necessary. The inventory management issue raised here is capacitated, meaning that each production node's ability of manufacturing is constrained by its own inventory of feedstock as well as its own production capacity. The only inventories stored at the nodes of the supply network are feedstock inventories because the model is represented as a make-to-order system. Upon request, all of the product inventory is sent right away to the downstream nodes, where the products become feedstock inventory to those nodes (or simply inventory for distributors and retailers). There is another holding (e.g., transportation) cost at the pipeline inventory (in-transit inventory).

6. Inventory that was not shipped or sold at the end of the last time period is considered lost and has no salvage value.

3.3. Key variables

This subsection deals with the key variables used in the model. For future research and contributions the same notations of the variables from the work of Perez et al (2021) have been used. The main variables can be observed in table 3. The primary variables are all non-negative and continuous.

Variable	Description
$a_{t,j,k}$	The reorder quantity requested to supplier node j by node k at the beginning of period t (the amount of material sent from node j to node k)
$S^d_{t,j,k}$	The amount retailer j sells to market k in period $t-1$. Note: Retail sales are indexed at the next period since these occur after demand in the current period is realized.
$S^o_{t,j}$	The on-hand inventory at node j just prior to when the demand is realized in period t.
$S^p_{t,j,k}$	The in-transit (pipeline) inventory between node j and node k just prior to when the demand is realized in period t
$u_{t,j,k}$	The unfulfilled demand at retailer j associated with market k in period $t-1$. Note: indexing is also shifted since any unfulfilled demand occurs after the uncertain demand is realized.
$R_{t,j}$	The profit (reward) in node j for period t.

Table 3: Main variables in the supply chain network model (Perez et al 2021)

3.4. Target function

The model purposed by Perez et al (2021) has a target function. The goal of the inventory management optimization is to increase the supply network's time-averaged projected profit (variable R, see equation 1). The assumption of a single retailer-market link is made with a single demand in every period. The sum of the profits in the main network nodes are calculated from the profits in each period ($R_1 = \sum_{j \in J} R_{1,j}$ and $R_t = \sum_{j \in J} R_{t,j}(\xi_t)\ \forall\ t \in T$). The main network nodes include the production/manufacturing, distribution, and retail nodes. The variable ξ_t provides the uncertain parameter vector which is connected to the demand in period t. $\boldsymbol{\xi_t}$ describes a specific realization of the parameter. The series of unknown parameters from period t to $t^{'}$ is represented as $\boldsymbol{\xi_{[t,t']}}$, with $\xi_{[t,t]}$ being a particular realization of that sequence (note: $\xi_{[1,1]}$ corresponds to deterministic stage 1). A modification of the demand parameter with a function is explained in section 3.8.

$$maxR = \tfrac{1}{|T|} * \Big(R_1 + E_{\xi_{[2,|T|]}|\xi_{[1,1]}}[maxR_2(\xi_2) + ... + E_{\xi_{[|T|,|T|]}|\xi_{[1,|T|-1]}}[maxR_{|T|}(\xi_{|T|})]]\Big) \quad (1)$$

3.5. Network profit

To calculate the profit $R_{t,j}$ in period t at node j we have to subtract the procurement costs $PC_{t,j}$, operating costs $OC_{t,j}$, unfulfilled demand penalties $UP_{t,j}$ and inventory holding costs $HC_{t,j}$ from the sales revenue $SR_{t,j}$ (see equation 2a and 2b). Since distribution nodes do not engage in manufacturing, the operational costs do not apply to them because they are associated with production expenses. Additionally, since all demands made of the inter-network must be attainable, there is no unmet demand at non-retail nodes.

$$R_{1,j} = SR_{1,j} - PC_{1,j} - OC_{1,j} - UP_{1,j} - HC_{1,j} \quad \forall j \in J \quad (2a)$$

$$R_{t,j}(\xi_t) = SR_{t,j}(\xi_t) - PC_{t,j}(\xi_t) - OC_{t,j}(\xi_t) - UP_{t,j}(\xi_t) - HC_{t,j}(\xi_t) \quad \forall t \in \{2, ..., |T|\}, j \in J \quad (2b)$$

The equations for the different node profit parameters are shown below for the sales revenue $SR_{t,j}$ (see equations 3a, 3b and 3c), procurement costs $PC_{t,j}$ (see equations 4a and 4b), operating costs $OC_{t,j}$ (see equations 5a and 5b), unfulfilled demand penalties $UP_{t,j}$ (see equation 6) and inventory holding costs $HC_{t,j}$ (see equations 7a and 7b).

The parameters $p_{j,k}$, $b_{j,k}$, and $g_{j,k}$ in the equations below stand for the material unit price, unfulfilled unit demand penalty and the unit material pipeline holding cost (transportation cost), respectively, for the link connecting nodes j and k.

Unit operating cost, production yield and on-hand inventory holding cost at node j are stored in parameters 0_j, v_j (0 to 1 range) and h_j. The predecessors and successors of node j are represented by the sets J_j^{in} and J_j^{out}, respectively.

$$SR_{1,j} = \sum_{k \in J_i^{out}} p_{j,k} * a_{1,j,k} \quad \forall j \in J^{prod} \cup J^{dist} \tag{3a}$$

$$SR_{t,j}(\xi_t) = \sum_{k \in J_i^{out}} p_{j,k} * a_{t,j,k}(\xi_t) \quad \forall t \in \{2, ..., |T|\}, j \in J^{prod} \cup J^{dist} \tag{3b}$$

$$SR_{t,j}(\xi_t) = \sum_{k \in J_i^{out}} p_{j,k} * S^d_{t,j,k}(\xi_t) \quad \forall t \in \{2, ..., |T|\}, j \in J^{retail} \tag{3c}$$

$$PC_{1,j} = \sum_{k \in J_i^{in}} p_{j,k} * a_{1,j,k} \quad \forall j \in J \tag{4a}$$

$$PC_{t,j}(\xi_t) = \sum_{k \in J_i^{in}} p_{j,k} * a_{t,j,k}(\xi_t) \quad \forall t \in \{2, ..., |T|\}, j \in J \tag{4b}$$

$$OC_{1,j} = \frac{o_j}{v_j} \sum_{k \in J_i^{out}} a_{1,j,k} \quad \forall j \in J^{prod} \tag{5a}$$

$$OC_{t,j}(\xi_t) = \frac{o_j}{v_j} \sum_{k \in J_i^{out}} a_{t,j,k}(\xi_t) \quad \forall t \in \{2, ..., |T|\}, j \in J^{prod} \tag{5b}$$

$$UP_{t,j}(\xi_t) = \sum_{k \in J_i^{out}} b_{j,k} * u_{t,j,k}(\xi_t) \quad \forall t \in \{2, ..., |T|\}, j \in J^{retail} \tag{6}$$

$$HC_{1,j} = h_j * S^o_{1,j} \sum_{k \in J_i^{in}} g_{k,j} * S^p_{1,k,j} \quad \forall j \in J \tag{7a}$$

$$HC_{t,j}(\xi_t) = h_j * S^o_{t,j}(\xi_t) \sum_{k \in J_i^{in}} S^p_{t,k,j}(\xi_t) \quad \forall t \in \{2, ..., |T|\}, j \in J \tag{7b}$$

3.6. Further parameters

The equations above (2a)-(7b) involve other parameters, which are calculated during the simulation. The exact equations can be found in the appendix and are listed as follows. Every node includes the on-hand inventory, which is updated with accordance to the material balances that account for incoming and outgoing material (see appendix equations 1a through 2c). Updates to the inventory levels of every node is done by addition of the incoming inventory and subtraction of any outgoing inventory to previously recorded inventory levels at the nodes, respectively. $S^o_{0,j}$ is the parameter representing the initial inventory at node j. Pipeline inventory arriving at node j from node k in period t is stored in the variable $a'_{t,k,j}$ (see appendix equation 3). To the downstream nodes $a_{t,j,k}$ or to the market at the retailer node $S^d_{t,j,k}$ the outgoing inventory is transferred or

sold, respectively. Adjustments to the sales quantities for production yields v_j are made at the production nodes. The production yields v_j are set to 1 for the distribution nodes.

The balances of the pipeline inventories at each arc are given by equations (4a) through (4c) in the appendix. As for the on-hand inventory, pipeline inventories are updated through the subtraction of delivered inventory to the downstream and the addition of new inventory requests to the recorded levels of previous pipeline inventory. The assumption is made that no inventory exists in the pipeline at $t = 0$.

Other equations include the replenishment orders of different types of nodes. Equations (5a) and (5b) in the appendix show the downstream replenishment requests limitations by the production capacity c_j. Feedstock inventory at the productions nodes are another type of limitations of the requests. The requests here are moved with v_j yield (see appendix equations 6a and 6b). The maximum limits on any downstream replenishment requests are defined by the available inventory at the distribution nodes since distribution-only nodes do not have manufacturing areas. Because of that, v_j is set to 1 in Equations (6a) and (6b) in the appendix. These sets of restrictions make sure that the requested quantities for reorders are always feasible, which implies that they can be sold and sent during the current time episode.

When the demand is realized, the retailer node sells the amount of goods that is currently in stock, as indicated by equations (7a) and (7b) in the appendix. This covers the inventory at the beginning of the period along with any reorder quantities that arrive at the start of the period (before the markets open). Equation (9a) in the appendix illustrates that sales at retailer nodes do not exceed market demand $d_{t,j,k}(\xi_t)$

Equation (8b) in the appendix illustrates what happens if backlogging is permitted (this option is used in this study): Any previously backlogged orders are added to the market demand. When counting unfilled orders as lost sales, u is eliminated from Equation (see appendix equation 8b).

Unmet demand at the retailer is the difference between current-period market demand and actual retail sales (see appendix equations 9a and 9b) The u term on the right side of Equation (9b) in the appendix is eliminated if the network operates in the lost sales mode.

3.7. Coefficients domains

The domains of the coefficients are listed in equations 8a through 11.

$$R_{1,j} \in \mathbb{R}^1 \quad \forall j \in J \tag{8a}$$

$$R_{t,j}(\xi_t) \in \mathbb{R} \quad \forall t \in \{2, ..., |T|\},\ j \in J \tag{8b}$$

$$S^0_{1,j} \geq 0 \quad \forall j \in J \tag{9a}$$

$$S^0_{t,j}(\xi_t) \geq 0 \quad \forall t \in \{2, ..., |T|\},\ j \in J \tag{9b}$$

$$a_{1,k,j},\ S^p_{1,k,j} \geq 0 \quad \forall j \in J, k \in J^{in}_j \tag{10a}$$

$$a_{t,k,j}(\xi_t),\ S^p_{t,k,j}(\xi_t) \geq 0 \quad \forall t \in \{2, ..., |T|\}, j \in J, k \in J^{in}_j \tag{10b}$$

$$S^d_{t,k,j}(\xi_t),\ u_{t,j,k}(\xi_t) \geq 0 \quad \forall t \in \{2, ..., |T|\}, j \in J^{retail}, k \in J^{out}_j \tag{11}$$

3.8. Seasonal demands function

The key point and the focus of the research question hasn't been discussed yet. The attention is on seasonality in a multi-echelon supply chain network. By default Perez et al (2021) defines the market (ultimate customer) demand as a Poisson distribution with a mean μ of 20. The environment can use general statistical distributions applied with the SciPy Python library. Another possibility is to use a list of user specified demands for each period. To integrate a seasonal behavior of the demand to the environment a function has been defined (see Equation 12). The seasonal market demand $d_{t,j,k}(x)$ is simulated as a sinusoidal function with an amplitude of 6, period of $\frac{2*\pi}{0.5} \approx 12.5663$, horizontal shift of 1.7 and an vertical shift of 20. The parameters for the x-axis are the months and time period in the model. The Episodic length is 24, which is equal to a simulation of 24 months.

$$d_{t,j,k}(x) = 6 * \sin(0.5 * x + 1.7) + 20 \tag{12}$$

Although, the function provides concrete defined market demands to the environment, the parameter and assumption of uncertainty is still valid. For the simulation and reinforcement learning algorithms the demand parameters are unknown and given directly.

4. Solution Methodology

The empirical study has been presented and explained. With this in mind, the following section will specify the solution methodology which is applied to solve the network inventory optimization problem with seasonal demands. In section 4.1, the basics of reinforcement learning and Markov decision process are described. Section 4.2 deals with the algorithms used in the reinforcement learning simulation such as Trust Region Policy Optimization, Advantage Actor-Critic, Proximal Policy Optimization, Soft Actor-Critic and Augmented Random Search. The network inventory simulation model is presented from a reinforcement learning perspective in section 4.3, where the MDP and state space, as well as the action space is formulated. The theory described in this section is primarily based on Sutton and Barto (2018) and the works related to the algorithms.

4.1. Reinforcement learning

Reinforcement learning takes a seat in the world of machine learning alongside with supervised learning and unsupervised learning. Supervised and unsupervised machine learning techniques use labeled and unlabeled data as input to predict outcomes or find hidden patterns as clusters or anomalies, respectively. In contrast, reinforcement learning has no supervisor (data) as a guide in the training process. The "data" in RL is provided through interactions in a (real-world) environment via so-called "trial-and-error"-methods.

In general, reinforcement learning is based on an agent who interacts with an environment over multiple time episodes to maximize the cumulative sum or rewards that the agent receives. This idea can be formulated as Markov decision process (see figure 2).

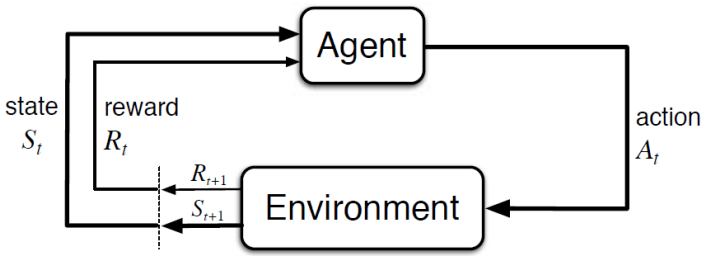

Figure 2: Agent-Environment interaction in an Markov decision process

The Markov decision process works as follows: The agent is selecting an action A_t and interacts with the environment in the time step t at the environment state S_t. Before the next time step $t+1$ the agent receives a numerical reward r_{t+1} and proceed to the new state S_{t+1}. The reward can be viewed as feedback from

the environment during the training and criterion for selecting action A_t.

Algorithms for reinforcement learning might be model-based or model-free, as well as based on value learning or policy gradient methods. In this paper we deal with model-free algorithms. In the model-based approach an model of the environment is given with the known probability distributions over various states. A model can also be built often with an approximation. On the other hand, model-free approach doesn't require any given or created model. "Experience" with trial-and-error from the training is collected and an optimal policy derived.

4.2. Algorithms

Another key aspect is the distinction between value-based algorithms and policy optimization. In value-based algorithms (also called Q-learning) an action-value function $Q^*(s, \alpha)$ is learnt with an approximator $Q_\theta(s, \alpha)$. The optimal policy can be derived directly from the value function. On the contrary, in policy-based methods the agent is learning directly with a policy function, which maps state to action. No value function is used for determining the policy. A policy π has a parameter θ and represents a probability distribution of actions (see equation 13).

$$\pi_\theta(\alpha|s) = P[\alpha|s] \tag{13}$$

Through gradient ascent on the performance target $J(\pi_\theta)$, policy optimization methods can directly optimize the parameter θ, or indirectly by maximizing local approximations of $J(\pi_\theta)$ (see equation 14).

$$J(\theta) = E_{\pi\theta}[\sum \gamma r] \quad with\ discount\ factor\ \gamma \tag{14}$$

The first algorithm applied to the network inventory problem is the Trust Region Policy Optimization algorithm (Schulman et al 2015). TRPO is an on-policy algorithm. TRPO updates policies by taking the largest step possible to improve performance with consideration of the limitation on how close the new and old polices can be. The restriction is defined in terms of Kullback-Leibler Divergence (KL-Divergence), which is a measure of the separation ("distance") between probability distributions. Because the updated policy is kept in a trust region close to the current policy, this algorithm avoids noticeable performance drops when compared to traditional policy gradient methods. Bellow the pseudocode of the TRPO is presented (see algorithm 1).

Algorithm 1: Trust Region Policy Optimization (TRPO)

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: Hyperparameters: KL-divergence limit δ, backtracking coefficient α, maximum number of backtracking steps K
3: **for** $k = 0, 1, 2, ...$ **do**
4: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
5: Compute rewards-to-go $\hat{R}_t$.
6: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
7: Estimate policy gradient as

$$\hat{g_k} = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \bigtriangledown_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} \hat{A}_t.$$

8: Use the conjugate gradient algorithm to compute

$$\hat{x}_k \approx \hat{H}_k^{-1} \hat{g}_k,$$

where $\hat{H}_k$ is the Hessian of the sample average KL-divergence
9: Update the policy by backtracking line search with

$$\theta_{k+1} = \theta_k + \alpha^j \sqrt{\frac{2\delta}{\hat{x}_k^T \hat{H}_k \hat{x}_k}} \hat{x}_k,$$

where $j \in \{0, 1, 2, ...K\}$ is the smallest value which improves the sample loss and satisfies the sample KL-divergence constraint.
10: Fit value function by regression on the mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.
11: **end for**

The next algorithm applied to the network inventory problem is the N-step Advantage Actor-Critic algorithm (Mnih et al 2016). The A2C algorithm combines Policy Based and Value Based Reinforcement Learning methods. A specific problem is solved by the algorithm, which comprises of two networks (the actor and the critic) cooperating. In its most basic form, the Advantage Function determines the agent's temporal difference (TD) Error or Prediction Error. At each time step the actor network selects an action, and the critic network evaluates the quality or Q-value of the input state. As the critic network gains knowledge about whether states are better or worse, the actor uses this knowledge to instruct the agent to seek out good states and avoid bad states. The pseudocode of the A2C is presented further on (see algorithm 2).

Algorithm 2: N-step Advantage Actor-Critic (A2C)

procedure N-step Advantage Actor-Critic
  Start with policy model π_θ and value model V_ω
  repeat:
    Generate an episode $S_0, A_0, r_0, ..., S_{T-1}, A_{T-1}, r_{T-1}$ following $\pi_\theta(\cdot)$
    for t from $T-1$ to 0:
      $V_{end} = 0$ if $(t + N \geq T)$ else $V_\omega(s_{t+N})$
      $R_t = \gamma^N V_{end} + \Sigma_{k=0}^{N-1} \gamma^k (r_{t+k}$ if $(t + k < T)$ else 0)
    $L_\theta = \frac{1}{T} \Sigma_{t=0}^{T-1} (R_t - V_\omega(S_t)) \log \pi_\theta(A_t|S_t)$
    $L_\omega = \frac{1}{T} \Sigma_{t=0}^{T-1} (R_t - V_\omega(S_t))^2$
    Optimize π_θ using $\nabla L(\theta)$
    Optimize V_ω using $\nabla L(\omega)$
end procedure

Proximal policy optimization (Schulman et al 2017) is a model-free, on-policy, policy gradient reinforcement learning method. The PPO algorithm is a combination of TRPO, because of using a trust region to improve the actor, and A2C, as having multiple workers. The main concept is that the new policy should be reasonably close to the old policy following an update. To prevent an excessively large update, PPO uses clipping. PPO with clipping is presented in the pseudocde diagram (see algorithm 3).

Algorithm 3: Proximal Policy Optimization (PPO-Clip)

Input: initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, ...$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_t$.
  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{0_k}}(s_t, a_t),\ g(\varepsilon, A^{\pi_{0_k}}(s_t, a_t)) \right),$$

  typically via stochastic gradient ascent with Adam.
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

  typically via some gradient descent algorithm.
end for

The Soft Actor-Critic (Haarnoja et al 2018) algorithm is an algorithm that is simultaneously learning a policy π_θ and two Q-functions Q_{ϕ_1}, Q_{ϕ_2}. Currently,

there are two variants of SAC that are widely used: (i) using a fixed entropy regularization coefficient α, and (ii) enforcing an entropy restriction by varying α throughout the training. Entropy regularization is a key component of SAC. Entropy is a measure of randomness in the policy, and it is trained to maximize a trade-off between expected return and entropy. Although the entropy-constrained variation is typically favoured by practitioners, in this work the version with a set entropy regularization coefficient is used for simplicity (see algorithm 4).

Algorithm 4: Soft Actor-Critic (SAC)

Input: initial policy parameters θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\phi_{targ,1} \leftarrow \phi_1$, $\phi_{targ,2} \leftarrow \phi_2$
repeat
  Observe state s and select action $a \sim \pi_\theta(\cdot|s)$
  Execute a in the environment
  Observe next state $s^{'}$, reward r, and done signal d to indicate whether $s^{'}$ is terminal
  Store $(s, a, r, s^{'}, d)$ in replay buffer $\mathcal{D}$
  if $s^{'}$ is terminal, reset environment state
  if it's time to update **then**
    for j in range(however many updates) **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s^{'}, d)\}$ from $\mathcal{D}$
      Compute targets for the Q-functions:

$$y(r, s^{'}, d) = r + \gamma(1 - d)\left(\min_{i=1,2} Q_{\phi_{targ,i}}(s^{'}, \tilde{\text{a}}^{'}) - \alpha \log \pi_\theta(\tilde{\text{a}}^{'}|s^{'})\right), \;\; \tilde{\text{a}}^{'} \sim \pi_\theta(\cdot|s^{'})$$

      Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s^{'},d) \in B} (Q_{\phi_i}(s, a) - y(r, s^{'}, d))^2 \quad \text{for } i = 1, 2$$

      Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left(\min_{i=1,2} Q_{\phi_i}(s^{'}, \tilde{\text{a}}^{'}_\theta(s)) - \alpha \log \pi_\theta(\tilde{\text{a}}^{'}_\theta(s)|s^{'})\right),$$

      where $\tilde{\text{a}}_\theta(s)$ is a sample from $\pi_\theta(\cdot|s)$ which is differentiable wrt θ via the reparametrization trick.
      Update target networks with

$$\phi_{targ,i} \leftarrow \rho \phi_{targ,i} + (1 - \rho)\phi_i \quad \text{for } i = 1, 2$$

    end for
  end if
until convergence

Last but not least, the Augmented random search algorithm (Mania et al 2018) is a model-free reinforcement learning, and a modified basic random search (BRS) algorithm (Flaxman et al 2005). The ARS algorithm tends to explore in the policy space, which means that the focus is on analysis on the whole action taken that ended as a positive reward in order to establish and reinforce the consecutive policy. Every action that resulted in the lowest rewards is discarded in the policy space. ARS is based on the method of finite difference, which measures the difference between two rewards that are moving in the opposite direction after applying minor perturbations (small random values) to the weighted inputs. The authors of the ARS algorithm purposed four versions explaining the execution and their advantages and disadvantages. As for the previous algorithms, the pseudocode for ARS can be viewed below (see algorithm 5).

Algorithm 5: Augmented Random Search (ARS):
four versions **V1, V1-t, V2** and **V2-t**

1: **Hyperparameters:** step-size α, number of directions per iteration N, number of top-performing directions to use b ($b \leq N$ is allowed only for **V1-t** and **V2-t**), standard deviation of the exploration noise v
2: **Initialize:** $M_0 = \mathbf{0} \in \mathbb{R}^{p \times n}$, $\mu_0 = \mathbf{0} \in \mathbb{R}^n$, and $\Sigma_0 = \mathbf{I}_n \in \mathbb{R}^{n \times n}$, $j = 0$
3: **while** ending condition not satisfied **do**
4: sample $\delta_1, \delta_2, ..., \delta_N$ in $\mathbb{R}^{n \times n}$ with i.i.d. standard normal entries
5: Collect $2N$ rollouts of H and their corresponding rewards using the $2N$ policies

$$\textbf{V1:} \begin{cases} \pi_{j,k,+}(x) = (M_j + \nu\delta_k)x \\ \pi_{j,k,-}(x) = (M_j - \nu\delta_k)x \end{cases}$$

$$\textbf{V2:} \begin{cases} \pi_{j,k,+}(x) = (M_j + \nu\delta_k)\ \mathrm{diag}(\Sigma_j)^{-1/2}(x - \mu_j) \\ \pi_{j,k,-}(x) = (M_j - \nu\delta_k)\ \mathrm{diag}(\Sigma_j)^{-1/2}(x - \mu_j) \end{cases}$$

for $k \in \{1, 2, ..., N\}$
6: Sort the directions δ_k by $\max\{r(\pi_{j,k,+}), (\pi_{j,k,-})\}$, denote by $\delta_{(k)}$ the k-th largest direction, and by $\pi_{j,(k),+}$ and $\pi_{j,(k),-}$ the corresponding policies
7: Make the update step:

$$M_{j+1} = M_j + \frac{\alpha}{b\sigma_R} \sum_{k-1}^{b} \left[r(\pi_{j,(k),+}) - r(\pi_{j,(k),-}) \right] \delta_{(k)},$$

where σ_R is the standard deviation of the $2b$ rewards used in the update step
8: **V2** : Set μ_{j+1}, Σ_{j+1} to be the mean an covariance of the $2NH(j+1)$ states encountered from the start of training.[2]
9: $j \leftarrow j + 1$
10: **end while**

4.3. Model implementation

In order to implement the reinforcement learning algorithms in the model, the network problem is formulated as a MDP - as stated in section 4.1 - a stochastic, sequential decision making problem. The agent determines and selects an action a_t based on the current state of the environment (S_t) at each time period t. Following the realization of the random variables and the chosen action, the simulation proceeds to the next state (S_{t+1}) and then returns the corresponding reward (R_t), which is the profit function from the equation 2a averaged over all nodes $j(R_t = \sigma_{j \in J} R_{t,j})$.

The state is represented by a vector that includes entries for the current demand at the retail node, each node's inventory levels in the network, and the inventory located in the pipelines along each edge in the network. The notation for the state would be as follows $S_t = \{d_{t,j,k}, S^o_{t,j}, S^p_{t,k',j} | j \in J, k \in J^{out}_j, k' \in J^{in}_j\}$. The parameters can be referred from section 3. In the same way the action space at each time period is formulated by a vector with all of the reorder quantities in the supply chain network ($a_t = \{a_{t,k,j} | j \in J, k \in J^{in}_j\}$).

5. Results

This section shows the results of the performance and evaluation of the reinforcement learning algorithms. To compare the methods, the experiments were performed using Python 3.9.13 on a computer with a 3.7 MHz x 6-core processor, 16 GB of RAM, a 6 GB GPU and Windows 11 version 22H2. All algorithms were applied using the Python Library Stable Baselines3 (Raffin et al. 2021a) as well as the Stable Baselines3 Contrib Linrary (Raffin et al. 2021b) containing experimental RL code and the latest algorithms. The training process was performed four times with different seeds and had an training time of four hours (240 Minutes) for each algorithm.

The performance results are summarized in table 4. The Columns show the different algorithms. The rows represent the performance criteria, such as mean profit (reward) at the end of the training process, the performance ratio between algorithm with the highest profit and the remaining algorithms, the total timesteps taken and mean profit (reward) during the training. The statistics are rounded to ten thousandths (4 digits).

In figure 3 we can see the different training processes for the algorithms TRPO, A2C, PPO and ARS. The x-axis corresponds to the training time given in minutes. The y-axis shows the mean episodic profit (reward) during the training process averaged over 100 episodes.

Training set	Performance measure	TRPO	A2C	PPO	SAC	ARS
1	Final Mean profit	618.4371	225.9229	610.0687	-5904.6112	642.9306
	Performance ratio	1.0396	2.8458	1.0539	-0.1089	1.0000
	Total timesteps	780288	753000	765952	333696	793344
	Mean training profit	474.8069	89.3434	165.8497	-5254.0632	468.9429
2	Final Mean profit	615.3205	528.2493	603.0623	-8693.5734	653.7538
	Performance ratio	1.0625	1.2376	1.0841	-0.0752	1.0000
	Total timesteps	716800	665500	675840	180672	721920
	Mean training profit	387.4738	169.6727	47.3073	-5430.2779	461.5172
3	Final Mean profit	612.5158	611.3738	608.9814	-6903.3306	645.6006
	Performance ratio	1.0540	1.0560	1.0601	-0.0935	1.0000
	Total timesteps	761856	1152000	761856	328896	771840
	Mean training profit	416.6490	345.1026	145.1410	-6603.5427	492.4325
4	Final Mean profit	614.5322	597.0429	611.7216	-7041.905	638.5094
	Performance ratio	1.0390	1.0695	1.0438	-0.0907	1.0000
	Total timesteps	776192	754000	759808	334368	778368
	Mean training profit	439.0174	432.0617	161.3811	-6163.9258	442.9504

Table 4: Performance comparison of different training processes

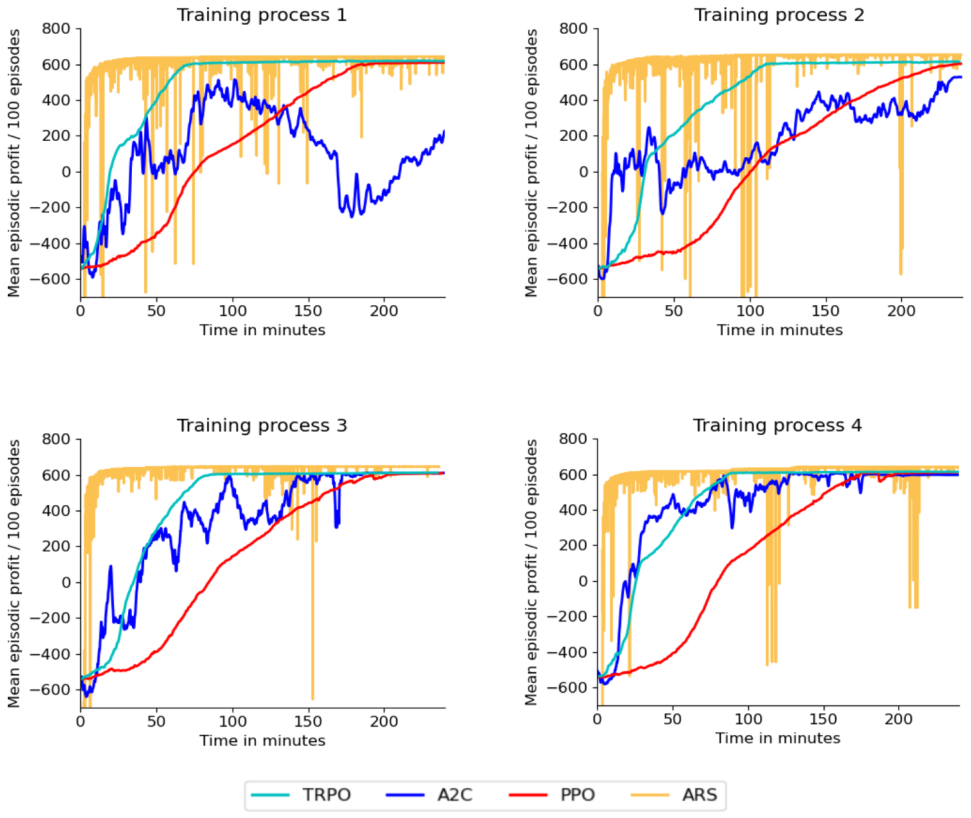

Figure 3: Comparison of training processes via mean episodic profit averaged over 100 episodes during four hours of training

The identical visualisation structure is done for the SAC algorithm, but shown in figure 4. The reason for a separate figure for the SAC algorithm is the different numerical range of the y-axis. Consequently, all four trainings of SAC are summarized in this figure.

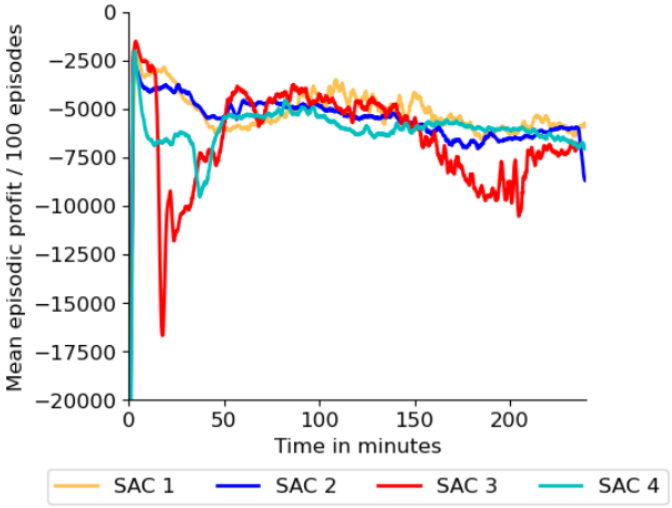

Figure 4: SAC performance comparison via mean episodic profit averaged over 100 episodes during four hours of training

6. Discussion

Following the results presented in section 5, we can discuss the findings. Overall, the reinforcement learning algorithms in all four scenarios have achieved nearly optimal results. In particular, the algorithms TRPO, PPO and ARS achieve a network profit (reward) over 600 at the end of the training phase. The A2C algoirthm demonstrates a network profit over 600 in the third scenario only. In comparison, the second and fourth training session of A2C realize a network profit of 528.2493 and 597.0429 and is therefore a positive and acceptable outcome of the simulation in terms of seasonality. Lower is the value of the first application of A2C with a reward of 225.9229.

The SAC algorithm has lead to the weakest results. As mentioned in the results section, the graphs for all four trainings of the SAC algorithm have been put in a separate figure (see figure 3). All four versions of SAC have a negative network profit at the end of the training process, this being -5904.6112, -8693.5734, -6903.3306 and -7041.905. In addition, the figure shows, that in the global maximum the SAC algorithm reaches a network profit of -2081.5722, -2317.6767, -1520.2705 and -2020.9177 in the four scenarios.

In context of the training process itself, we see different patterns. TRPO and PPO tend to have smoother reward curves in comparison to A2C, SAC and ARS.

Furthermore, PPO and ARS tend to maximization of the network profit under seasonality faster than the other alogirthms which can be observed on the graphs in figure 3. Total timesteps measure the steps taken in the environment and indicate that ARS achieves the best results, as well as takes more steps during the training process. The opposite is valid for the SAC algorithm with 333696, 180672, 328896 and 334368 stesp in the environment and is the slowest learning performance. The mean of the training process shows another measure to give a sense of the performances. Algorithms with similar means of the training are more consistent in general (TRPO, PPO, ARS). In the same way, deviations between the average training values indicate varied performance.

On the whole, ARS has the best performance in all four scenarios with the highest network profit and most steps taken. From the graph it is clear that ARS achieves approximately the maximum reward in the shortest time period in the first 30 minutes with some volatility during the rest of the training. Because of that ARS is used as the base for the performance ratio measure - an alternative estimate of the algorithm's efficiency.

On average, each training run of the algorithms took less than 250 minutes. Nevertheless, the application of reinforcement learning models has a very quick execution time once they have been trained. The reinforcement learning algorithms will instantly deliver the decisions for the following time step after receiving the supply chain's current state for the model. Thus, even if the training process may take a while to complete, it may be applied quickly, which is a huge benefit for potential real-time decision-making and operations research problems, especially in terms of seasonality.

7. Conclusion

This work contributes to the application of reinforcement learning algorithms in supply chain networks with demand seasonality. A simulation of a multi-echelon network with 24-period single market and a single nonperishable product was the baseline. Deep reinforcement learning algorithms such as TRPO, A2C, PPO, SAC and ARS were applied, and their performance compared during four training runs. ARS has shown the highest and fastest performance, whereas SAC showed the weakest achievements. However, TRPO, A2C and PPO reach high network profits (rewards) at the end of the training process and are as capable of optimizing the network profit in terms of seasonality.

Future extensions are needed to evaluate the performance of reinforcement learning algorithms under seasonality. This work can be improved and expanded in a variety of ways. First of all, the algorithms used in this work can be studied

further, by considering the effects of additional configurations to the environment, including more levels of supply chain echelons, changing the amount of entities or tweaking parameters and the seasonality function. Furthermore, a wider variety of different deep reinforcement learning algorithms should be tested to gain a better understanding of their performance. Alternatively, hyperparameter tuning of the algorithms can be investigated to study whether the performance improves and achieves different results.

In conclusion, this study shows, that the reinforcement learning algorithms used are a particularly helpful method for addressing these optimization difficulties and may be practically applied easily while still outperforming existing state-of-the-art methods. However, the practical applications of deep reinforcement learning methods still need to be researched.

A. Appendix

Following equations (A.1a through A.9b) were mentioned in section 3.6 and provide additional information about the parameters:

$$S^o_{1,j} = S^o_{0,j} + \sum_{k \in J^{in}_j} a^{'}_{1,k,j} - \frac{1}{v_j} * \sum_{k \in J^{out}_j} a_{1,j,k} \quad \forall j \in J^{prod} \cup J^{dist} \tag{A.1a}$$

$$S^o_{2,j}(\xi_2) = S^o_{1,j} + \sum_{k \in J^{in}_j} a^{'}_{2,k,j} - \frac{1}{v_j} * \sum_{k \in J^{out}_j} a_{2,j,k}(\xi_2) \quad \forall j \in J^{prod} \cup J^{dist} \tag{A.1b}$$

$$S^o_{t,j}(\xi_t) = S^o_{t-1,j}(\xi_{t-1}) + \sum_{k \in J^{in}_j} a^{'}_{t,k,j} - \frac{1}{v_j} * \sum_{k \in J^{out}_j} a_{t,j,k}(\xi_t) \quad t \in \{3, ..., |T|\} \quad \forall j \in J^{prod} \cup J^{dist} \tag{A.1c}$$

$$S^o_{1,j} = S^o_{0,j} + \sum_{k \in J^{in}_j} a^{'}_{1,k,j} \quad \forall j \in J^{retail} \tag{A.2a}$$

$$S^o_{2,j}(\xi_2) = S^o_{1,j} + \sum_{k \in J^{in}_j} a^{'}_{2,k,j} - \sum_{k \in J^{out}_j} S^d_{2,j,k}(\xi_2) \quad \forall j \in J^{retail} \tag{A.2b}$$

$$S^o_{t,j}(\xi_t) = S^o_{t-1,j}(\xi_{t-1}) + \sum_{k \in J^{in}_j} a^{'}_{t,k,j} - \sum_{k \in J^{out}_j} S^d_{t,j,k}(\xi_t) \quad \forall t \in \{3, ..., |T|\}, j \in J^{retail} \tag{A.2c}$$

$$a^{'}_{t,k,j} \begin{cases} 0, & \text{if } t - L_{k,j} < 1 \\ a_{1,k,j}, & \text{if } t - L_{k,j} = 1 \\ a_{t-L_{k,j},k,j}(\xi_{t-L_{k,j}}), & \text{if } t - L_{k,j} > 1 \end{cases} \quad \forall t \in T, j \in J^{retail}, k \in J^{in}_j \tag{A.3}$$

$$S^p_{1,k,j} = -a^{'}_{1,k,j} + a_{1,k,j} \quad \forall j \in J, k \in J^{in}_j \tag{A.4a}$$

$$S^p_{2,k,j}(\xi_2) = S^p_{1,k,j} - a^{'}_{2,k,j} + a_{2,k,j}(\xi_2) \quad \forall j \in J, k \in J^{in}_j \tag{A.4b}$$

$$S^p_{t,k,j}(\xi_t) = S^p_{t-1,k,j}(\xi_{t-1}) - a^{'}_{t,k,j} + a_{t,k,j}(\xi_t) \quad \forall t \in \{3, ..., |T|\}, j \in J, k \in j^{in}_j \tag{A.4c}$$

$$\sum_{k \in J^{out}_j} a_{1,j,k} \leq c_j \quad \forall j \in J^{prod} \tag{A.5a}$$

$$\sum_{k \in J^{out}_j} a_{t,j,k}(\xi_t) \leq c_j \quad \forall t \in \{2, ..., |T|\}, j \in J^{prod} \tag{A.5b}$$

$$\sum_{k \in J^{out}_j} a_{1,j,k} \leq S^o_{1,j} * v_j \quad \forall j \in J^{prod} \cup J^{dist} \tag{A.6a}$$

$$\sum_{k \in J_j^{out}} a_{t,j,k}(\xi_t) \leq S^o_{t,j}(\xi_t) * v_j \quad \forall t \in \{2, ..., |T|\}, j \in J^{prod} \cup J^{dist} \tag{A.6b}$$

$$\sum_{k \in J_j^{out}} S^d_{2,j,k}(\xi_2) \leq S^o_{1,j} \quad \forall j \in J^{retail} \tag{A.7a}$$

$$\sum_{k \in J_j^{out}} S^d_{t,j,k}(\xi_t) \leq S^o_{t-1,j}(\xi_{t-1}) \quad \forall t \in \{3, ..., |T|\}, j \in J^{retail} \tag{A.7b}$$

$$S^d_{2,j,k}(\xi_2) \leq d_{2,j,k}(\xi_2) \quad \forall j \in J_{retail}, k \in J_j^{out} \tag{A.8a}$$

$$S^d_{t,j,k}(\xi_t) \leq d_{t,j,k}(\xi_t) + u_{t-1,j,k}(\xi_{t-1}) \quad \forall t \in \{3, ..., |T|\}, j \in J_{retail}, k \in J_j^{out} \tag{A.8b}$$

$$u_{2,j,k}(\xi_2) = d_{2,j,k}(\xi_2) - S^d_{2,j,k}(\xi_2) \quad \forall j \in J^{retail}, k \in J_j^{market} \tag{A.9a}$$

$$u_{t,j,k}(\xi_t) = d_{t,j,k}(\xi_t) + u_{t-1,j,k}(\xi_{t-1}) - S^d_{t,j,k}(\xi_t) \quad \forall t \in \{3, ..., |T|\}, j \in J^{retail}, k \in J_j^{market} \tag{A.9b}$$

B. References

[1] Alves, J., Mateus, G. 2022. Multi-echelon Supply Chains with Uncertain Seasonal Demands and Lead Times Using Deep Reinforcement Learning. *arXiv:2201.04651v1 [cs.LG].*

[2] Boute, R.N., Gijsbrechts, J., Jaarsveld, W., Vanvuchelen, N. 2021. Deep reinforcement learning for inventory control: A roadmap. *European Journal of Operational Research* 298(2), 401-412

[3] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W. 2016. OpenAI Gym. *arXiv:1606.01540v1 [cs.LG]*

[4] Caro F., Gallien J. 2010. Inventory management of a fast-fashion retail network. *Operations Research* 58(2), 257-273.

[5] Cho, D.W., Lee, Y.H. 2011. Bullwhip effect measure in a seasonal supply chain. *Journal of Intelligent Manufacturing* 23, 2295-2305.

[6] Costantino, F., Giulio, D.G., Shaban, A., Tronci, M. 2013. Exploring the Bullwhip Effect and Inventory Stability in a Seasonal Supply Chain. *International Journal of Engineering Business Management*, 5. 5-23.

[7] Duc, T. T. H., Luong, H. T., Kim, Y. D. 2008. A measure of bullwhip effect in supply chains with a mixed autoregressive-moving average demand process. *European Journal of Operational Research* 187(1), 243-256.

[8] Flaxman, A.D., Kalai, A.T., McMahan, H.B. 2005. Online convex optimization in the bandit setting: gradient descent without a gradient *arXiv:cs/0408007v1 [cs.LG].*

[9] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *Proceedings of the 35th International Conference on Machine Learning* PMLR 80, 1861-1870.

[10] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu H., Gupta, A., Abbeel, P., Levine, S. 2019. Soft Actor-Critic Algorithms and Applications. *arXiv:1812.05905v2 [cs.LG].*

[11] Hubbs, C.D., Perez, H.D., Sarwar, O., Sahinidis, N.V., Grossmann, I.E., Wassick, J.M. 2020. OR-Gym: A Reinforcement Learning Library for Operations Research Problems. *arXiv 2020, arXiv:2008.06319v2.*

[12] Hutse V. 2019. Reinforcement Learning for Inventory Optimisation in multi-echelon supply chains. *Master's Dissertation in degree of Master in Business Engineering: Data Analytics*. Ghent University.

[13] Kemmer, L., von Kleist, H., Rochebouet, D., Tziortziotis, N., Read, J. 2018. Reinforcement learning for supply chain optimization. *European Workshop on Reinforcement Learning* 14.

[14] Laumanns, M., Woerner, S. 2017. Multi-echelon supply chain optimization: Methods and application examples. Povoa, A.P.B., Corominas, A., de Miranda, J.L. *Optimization and Decision Support Systems for Supply Chains*. Springer International Publishing, Cham, 131-138.

[15] Lee, H.L., Billington, C. 1993. Material management in decentralized supply chains. *Operations research* 41(5), 835-847.

[16] Lee, H. L., Padmanabhan, V., Whang, S. G. 1997. The bullwhip effect in Supply chains. *Sloan Management Review* 38(3), 93-102.

[17] Makridakis, S., Wheelwright, S. C. 1989. Forecasting methods for management. *Singapore: Wiley, Cypriot filax*, 614

[18] Mania, H., Guy, A., Recht, B. 2018. Simple random search provides a competitive approach to reinforcement learning. *arXiv:1803.07055v1 [cs.LG]*.

[19] Matyas, J. 1965. Random optimization. Automation and Remote control. *Avtomatika i Telemekhanika* 26(2), 246-253.

[20] Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K. 2016. Asynchronous Methods for Deep Reinforcement Learning. *Proceedings of The 33rd International Conference on Machine Learning* PMLR 48, 1928-1937

[21] Nahmias, S., Smith, S.A. 1993. Mathematical Models of Retailer Inventory Systems: A Review. *Springer US, Boston, MA*, 249-278.

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

[23] Peng, Z., Zhang, Y., Feng, Y., Zhang, T., Wu, Z., Su, H. 2019. Deep reinforcement learning approach for capacitated supply chain optimization under demand uncertainty. *Chinese Automation Congress (CAC)*. 3512-3517.

[24] Perez, H.D., Hubbs, C.D., Li, C., Grossmann, I.E. 2021. Algorithmic Approaches to Inventory Management Optimization. *MDPI journal processes*

[25] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N. 2021a. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 1-8.

[26] Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A. 2021b. Stable-Baselines3 - Contrib (SB3-Contrib) version 1.6.2. Available online at https://github.com/Stable-Baselines-Team/stable-baselines3-contrib, checked on 25/10/2022.

[27] Schulman, J., Levine, S., Moritz, P., Jordan, M., Abbeel, P. 2015, Trust Region Policy Optimization. *Proceedings of the 32nd International Conference on Machine Learning* PMLR 37, 1889-1897.

[28] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. 2017. Proximal Policy Optimization Algorithms. *arXiv:1707.06347v2 [cs.LG].*

[29] Seelenmeyer, S., Kissler, A. 2020. Multimodal logistics - application of optimization methods and future usage of artificial intelligence. *CO@Work 2020 Summer School, SAP SE.*

[30] Smith S.A., Agrawal N. 2000. Management of multi-item retail inventory systems with demand substitution. *Operations Research* 48(1), 50-64.

[31] Stranieri, F., Stella, F. 2022. A Deep Reinforcement Learning Approach to Supply Chain Inventory Management. *European Workshop on Reinforcement Learning* 15.

[32] Sultana, N.N., Meisheri, H., Baniwal, V., Nath, S., Ravindran, B., Khadilkar, H. 2020. Reinforcement Learning for Multi-Product Multi-Node Inventory Management in Supply Chains Handling warehouses, trucks, and stores with finite capacity. *arXiv:2006.04037 [cs.LG].*

[33] Sutton, R.S., Barto, A.G. 2018. *Reinforcement Learning: An Introduction.* 2nd ed. Westchester Publishing Services. United States of America.

[34] Wei, W. W. S. 1990. Time series analysis: Univariate and multivariate methods. *Redwood City: Addison-Wesley.*