European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Production, Manufacturing, Transportation and Logistics

Modelling the influence of returns for an omni-channel retailer

Joost Goedhart*, René Haijema, Renzo Akkerman

Operations Research and Logistics Group, Wageningen University, Hollandseweg 1, Wageningen 6706 KN, The Netherlands

ARTICLE INFO

Article history:
Received 9 November 2021
Accepted 16 August 2022
Available online xxx

Keywords:
Inventory
Online returns
Replenishment & rationing
Markov decision process
Deep reinforcement learning

ABSTRACT

More brick-and-mortar retailers open an online channel to increase sales. Often, they use the store to fulfil online orders and to receive returned products. The uncertain product returns however complicate the replenishment decision of a retailer. The inventory also has to be rationed over the offline and online sales channels. We therefore integrate the rationing and ordering decisions of an omni-channel retailer in a Markov Decision Process (MDP) that maximises the retailer's profit. Contrary to previous studies, we explicitly model multi-period sales-dependent returns, which is more realistic and leads to higher profit and service levels. With Value Iteration (VI) an exact solution can only be computed for relatively small-scale instances. For solving large-scale instances, we constructed a Deep Reinforcement Learning (DRL) algorithm. The different methods are compared in an extensive numerical study of small-scale instances to gain insights. The results show that the running time of VI increases exponentially in the problem size, while the running time of DRL is high but scales well. DRL has a low optimality gap but the performance drops when there is a higher level of uncertainty or if the profit trade-off between different actions is minimal. Our approach of modelling multi-period sales-dependent product returns outperforms other methods. Furthermore, based on large-scale instances, we find that increasing online returns lowers the profit and the service level in the offline channel. However, longer return windows do not influence the retailer's profit.

1. Introduction

The retail sector is changing drastically with the rise of online shopping. As customers are shopping more online, traditional brick-and-mortar stores are also changing their business strategy by opening online shopping channels. This integration of offline and online shopping channels is referred to as omni-channel retailing (Verhoef et al., 2015). Omni-channel retailing provides the customer a uniform shopping experience, in which goods can be inspected, bought, and returned through all available shopping channels. However, customers that order products online are only able to physically inspect their goods after delivery. Therefore, products that are displayed online might not satisfy customer expectations or are bought impulsively, online ordered products are often returned (Abdulla et al., 2019).

The return flow of online ordered products has become a significant issue for many retailers. For instance, an online fashion retailer reported return percentages between 13% and 45% (de Leeuw et al., 2016). As the returned products are often resalable they should be accounted for in inventory management (Radhi & Zhang, 2019). Due to the uncertainty in the quantity and timing of returned products, retailers might end up with an excessive stock if they do not consider the return flow (Bernon et al., 2016; Xu & Jackson, 2019). Even though retailers sometimes try to reduce returns (e.g., through stricter return windows or return fees), in many cases retailers are providing increasingly lenient return policies to increase customer satisfaction. Therefore, retailers need to adapt their inventory management to the growing return flow as the handling of returned products is important to reduce inventory related cost.

Customers often prefer the ability to return an online purchase in-store. Brick-and-mortar store returns are free of charge for customers and do not require repackaging, whereas via mail the customer has to deal with packaging and often has to pay a shipping fee. Additionally, the customer can get a direct replacement in the store or immediate refund (Wollenburg et al., 2018). It can also be more profitable for retailers to encourage customers to return products to the store instead of shipping via mail (Nageswaran et al., 2020). With using the brick-and-mortar store for returns they can steer the customer towards exchanging their product or towards buying another product (Tarn et al., 2003). Furthermore, the retailer can use the brick-and-mortar store to inspect returns, thus not accepting invalid or unwanted returns (de Leeuw et al., 2016). Due to such gatekeeping, returns are processed faster, which

* Corresponding author.
E-mail address: joost.goedhart@wur.nl (J. Goedhart).

https://doi.org/10.1016/j.ejor.2022.08.021

is important to sustain the value of the product as they can quickly be added back to the store inventory (Hübner et al., 2015). Allowing returns from online sales to be returned in-store is referred to as cross-channel returns (Radhi & Zhang, 2019). However, the retailer needs to account for these returns in their ordering decision as not accounting for the returns could result in unbalanced inventory positions and significant revenue loss (Chen & Bell, 2012; Hu et al., 2019).

The role of the store for an omni-channel retailer has becoming increasingly important, where the store can operate as a fulfilment centre for online orders, a pick-up point, a place to handle returns, or an information channel (Hübner et al., 2022; Mou et al., 2018). Shipping online ordered products from stores is referred to as a ship-from-store strategy (Agatz et al., 2008). The strategy has several advantages and disadvantages for the retailer. Advantages are for instance lower inventory levels, higher turnover rates, and shorter delivery distances (Bayram & Cesaret, 2021; Jalilipour Alishah et al., 2015), and examples of disadvantages are negative in-store customer experiences due to store personnel picking orders and inaccurate inventory positions. To mitigate these negative side-effects managerial studies suggest to reserve part of the brick-and-mortar store inventory for the online demand (ENC, 2016; Hobkirk, 2015). This reservation of part of the inventory is referred to as rationing the inventory across the shopping channels.

By rationing an inventory across channels, a trade-off is made. Storing products in the offline channel is more costly due to expensive shelve space in the store. However, online channels typically have a reduced profit per product due to the cost of online fulfilment and the probability of a product being returned. Little research has been conducted on the trade-off a retailer has to make between the offline and online channel in the context of using the store assets for the online channel. In this paper, we therefore study how a retailer can utilise their brick-and-mortar store to handle the online fulfilment as well as the potential returns. The objective of this study is to identify an optimal replenishment and rationing policy for an omni-channel retailer taking into account the return flow of the online ordered products.

The integration of returns in the ordering and rationing decision complicates the studied problem, as predictions for returns have to be considered in modelling the decision problem. Clearly, returns will depend on historical sales. However, keeping track of historical sales can easily make modelling approaches intractable. Therefore, historical sales data is often approximated by aggregating detailed historical sales data, so the information can still be used in some way to make better decisions. If retailers would not take into account potential returns at all, they would end up with excessive stock. Additionally, retailers can choose to manage their inventory such that the potential returns from the online sales channel are considered in setting ordering and rationing policies. For instance, the rationing decision can be used to reserve products for future in-store customers who are more profitable when inventory positions and outstanding orders are low.

This research contributes to the literature on omni-channel retailing by showing how returns affect the retailer's profit and inventory management. More specifically, we first provide a model that explicitly considers multi-period sales-dependent returns in the inventory management of an omni-channel retailer, based on a Markov Decision Process (MDP) formulation. Second, as the MDP might become too large to solve large-scale instances with value iteration (VI) to obtain an exact solution, we demonstrate how Deep Reinforcement Learning (DRL) can be used to solve the problem and obtain an approximated solution. Third, we compare the multi-period sales-dependent return MDP with other methods of modelling returns to gain insight in the importance of incorporating historical sales in decision-making. Fourth, based on our numerical results, we provide managerial insights on how an omni-channel retailer should cope with returns, as well as general insights on the use of DRL in the context of retail operations inventory management.

The remainder of this paper is structured as follows. Section 2 presents related research on return strategies for omni-channel settings and inventory management. In Section 3, we further outline the decision problem and formulate it as a MDP. Section 4 presents the implementation of the DRL algorithm to the studied problem. In Section 5, the performance of the DRL policy compared to the optimal solution is investigated for a wide range of instances on different performance measures. Additionally, the importance of including historical sales data for decision-making is investigated by comparing it with other methods of modelling returns. In Section 6 we derive managerial insights from large-scale instances. Section 7 concludes the research and discusses future research opportunities.

2. Literature review

Our work is related to the literature on omni-channel retailing as well as the literature on inventory management with return flows. Below, we briefly address related work on omni-channel retailing, with a focus on returns in an omni-channel context and the role of the store in omni-channel retailing. This is followed by a discussion of the literature on different methods for modelling return flows in inventory management.

2.1. Omni-channel retailing

The study of return management in an omni-channel context is an understudied problem (Bernon et al., 2016; Muir et al., 2019; Xu & Jackson, 2019). Hübner et al. (2022) also mention that the current body of literature does not close the gap between inventory management and returns in an omnichannel context. However, for retailers, return management is becoming increasingly important, especially due to the increase in the return flow originating from online orders. Retailers offer lenient return policies to relieve customer shopping risk and increase demand. However, different return policies have different effects. Generous return policies increase demand, whereas longer return windows and exchange leniency influence return percentages (Janakiraman et al., 2016). Ketzenberg et al. (2020) mention that lenient return policies have resulted in customers exploiting the retailers policies.

The current body of literature on return management mostly focuses on retailers operating with a single online sales channel. For an overview of this literature, we refer to the recent comprehensive review by Abdulla et al. (2019). Most modelling research around returns is about return policies, and often only focuses on single-period return windows. Abdulla et al. (2019) also conclude that significant opportunities for future research lies in analysing how operational decisions regarding returns can be made to improve retailers performance.

The strategic side of handling returns by brick-and-mortar stores has been extensively studied, where the focus is often on whether stores should be used for handling returns or not (e.g. Gao et al., 2022; Jin et al., 2020; Mandal et al., 2021). However, the operational side of handling the returned products in-store has only been studied to a limited extent (Hübner et al., 2016; Mena et al., 2016). Here, one of the main issues is the re-balancing of inventory (Bernon et al., 2016). Muir et al. (2019) and Radhi & Zhang (2019) investigate how same- and cross-channel returns influence order policies, and conclude that leveraging brick-and-mortar stores for returns improves service levels as returned products can be resold quicker. Dijkstra et al. (2019) investigate how to re-balance the cross-channel returns across the physical stores or online fulfilment center of the retailer.

2.2. Inventory management with return flows

Modelling product return flows is complicated by the interaction between inventories, sales, and returns. Return flows depend on historic sales, while current sales is limited by the inventory, which is in turn influenced by the returns. In the literature, we find two different ways to model product returns: (i) product returns are independent of demand, (ii) and product returns are dependent on demand.

By assuming that returns are independent of demand, return flows can be modelled as exogenous flows. Fleischmann et al. (2002) and Feinberg & Lewis (2005) mention that in such cases, the problem comes down to a variant of an inventory model with positive or negative net demand. However, Kiesmüller & Van der Laan (2001) show that neglecting the dependency between demand and return results in poor performance of inventory policies. Yet, Zerhouni et al. (2013) mention that ignoring the dependency between demand and return increased costs only minimal in their study, which can be attributed to the long return window they considered, damping the effect of demand fluctuations. The study setting is based on de Brito & Dekker (2003), who also mention that long return windows is an assumption that does not hold for most retail settings. Cases in which the return time is long, such as certain remanufacturing systems can neglect the dependency without much impact on performance (Fleischmann & Minner, 2004), due to the damping effect mentioned above.

Fleischmann & Kuik (2003) discuss that modelling returns dependent on demand is difficult, as such dependence spans across multiple periods. Therefore, early work on returns is focused on modelling the dependency only within the same period or across one period, as this reduces the complexity (DeCroix, 2006). However, such an assumption is often too harsh as approximating returns within at most one period results in sub-optimal policies (Benedito & Corominas, 2013). Having the returns span across multiple periods increases the modelling complexity significantly as it grows with the quantity of products sold and the number of periods considered for the return window. Benedito & Corominas (2013) consider a remanufacturing system in which the products are always returned, either when their lifetime has exceeded or are broken down during their lifetime. They do include a dependency across multiple periods and show that finding the optimal policy for a small example is computationally feasible, but for longer return windows they propose an approximated MDP where returns are independent of sales. Ambilkar et al. (2022) recently also emphasised that an interesting research direction is using machine learning techniques to solve the intractable return management problem.

2.3. Contribution

We conclude from the literature that modelling multi-period sales-dependent return flows is difficult. The complexity of modelling approaches typically grows exponentially with the length of the return window and the number of products sold in that window and thus easily becomes intractable. However, including this dependency of returns with sales is important to ensure optimal policies. Largely due to this complexity, the influence of returns on managing inventories for an omni-channel retailer is an understudied subject. Whereas most previous literature focuses on the strategic decision of whether or not to handle returns in-store, we specifically aim to study the impact of handling returns on operational decision making. We contribute to the existing literature by studying the inventory management of an omni-channel retailer that uses their store inventory to fulfil both offline and online demand. The products ordered online have a probability of being returned within a given return window, spanning over multiple selling periods. To reduce model complexity, and still capture some of the temporal dynamics, we introduce a modelling approach to address the complexity of return dependence across multiple periods.

3. Markov decision process

We study the setting of an omni-channel retailer with a physical store and an online channel. The retailer has two decisions to be made: an ordering decision and a rationing decision. We consider a setting in which a period consists of T sub-periods. We assume the retailer places an order at the beginning of a period, which is replenished after ℓ sub-periods. This could for instance reflect a weekly ($T = 7$) ordering decision with a delivery after two days ($\ell = 2$). This ordering decision q occurs at the beginning of the first sub-period. Additionally, at the beginning of each sub-period, a rationing decision a for the two channels is made, deciding on the inventory levels that are reserved for each channel. These inventories are used to fulfil the demand of the channels, and no substitution between channels takes place. When a customer faces a stock-out in their preferred channel, the demand is assumed to be lost which is a common assumption in retail environments. The customer has a limited return window in which they can return the product. Throughout the sub-periods, products that were sold in previous periods in the online channel can be returned.

At the end of each sub-period, the remaining inventories of the two channels, together with the products returned during that sub-period, are used for fulfilment the next day. The retailer keeps track of the products sold in the online channel, as these can lead to returns. We assume that products sold through the offline channel are not returned, as product returns are mostly a characteristic of online sales.

3.1. Modelling approach

The problem described above can be formulated as an MDP. As the problem consists of decisions and state transitions on different time intervals, we formulate it as a Hierarchical Markov Decision Process as described in Kristensen (1988). Describing the studied problem as a hierarchical MDP allows for convenient notation as action spaces and state transitions will not be dependent on the sub-period. The hierarchy of decisions in the MDP is illustrated in Fig. 1. At level I of the MDP the replenishment decision is made, and at level II of the MDP the rationing decision is made in each sub-period. The selling process and possible returns are also included in level II for each sub-period.

The objective of the MDP is to maximise profit, consisting of the sales revenue and the different costs. These costs include holding and handling costs for the individual channels, online fulfilment costs, costs of handling returns, and costs of the product. In an omni-channel setting, the price of the product is equal in both channels. We keep track of the number of products sold online each sub-period however, aggregate them on the time scale of level I to reduce complexity. As the retailer has a maximum time window for customer to return their products, we only keep track of products that are sold within M periods. Furthermore, we assume that products are not returned within the period the product is sold as customers do not instantly return a product after delivery.

At level I of the MDP, the state S consists of the inventory position I and the quantity of unreturned products sold in the online channel up to M periods ago $\boldsymbol{R} = (R_1, \ldots, R_M)$, together forming the state $S = (I, \boldsymbol{R})$. We use the bold notation to indicate it as a vector. At this level, the replenishment decision q is made. At level II of the MDP, the state s consists of the inventory position I, the number of products sold in the online channel so far this sub-period R_0, unreturned products from the previous periods $\boldsymbol{R}$, the

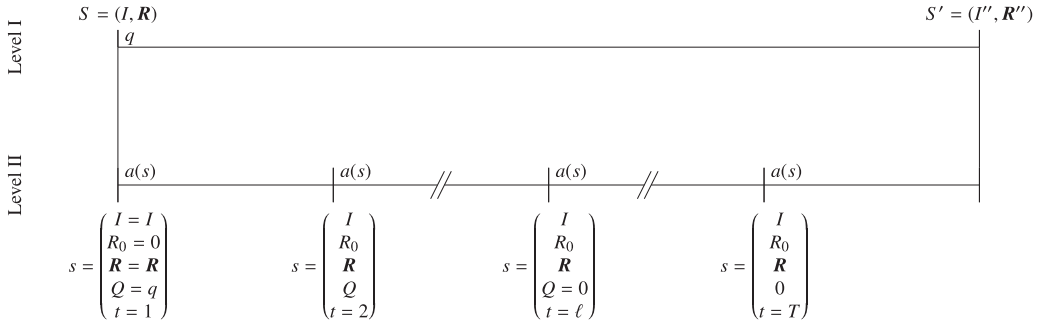

Fig. 1. Visualisation of the states and actions in one period.

outstanding replenishment order Q, and the sub-period t. At this level, the rationing decision a is made each sub-period, where a is the number of products stored in the offline channel. At the beginning of the first sub-period the information of the state at level I is used to construct the state at level II: $s = (I, R_0, \boldsymbol{R}, Q, t)$. The inventory state I and $\boldsymbol{R}$ are identical from the state at level I. For the first sub-period R_0 is set to zero, and Q is set to the replenishment decision of level I q. For all subsequent sub-periods, the number of online products sold in the previous sub-periods is added to R_0.

When the replenishment occurs at sub-period $t = \ell$, the outstanding order quantity is set to zero for the rest of the period, i.e. $s = (I, R_0, \boldsymbol{R}, Q = 0, t = \ell)$. At the end of all sub-periods, the total online sales of the period is given by R'_0, and the remaining number of unreturned products sold in the channel m periods ago by R'_m. This state information of level II is passed back to level I to form the state $S' = \left(I'', \boldsymbol{R}''\right)$. Where I'' is thus the closing inventory of the period, and $\boldsymbol{R}''$ is the sold products in the online channel that are not returned, as these are from the previous period they are given as $R''_1 = R'_0, R''_2 = R'_1, \ldots, R''_M = R'_{M-1}$.

The uncertainty in the MDP comes from both the demand and the product returns. The demand is modelled according to a discrete distribution, with $P_i(d_i)$ indicating the probability that demand of channel i is d_i with $i \in \{1 = \text{offline}, 2 = \text{online}\}$ and $d_i \in \{0, 1, \ldots, D_i\}$. D_i indicates the maximum possible demand of the channel. The returns are modelled according to M Binomial distributions given as $B(r_m)$ with parameters ρ and R_m, where ρ is the probability a single product is returned, and R_m is the pool of returnable items, which is the total sales of m periods ago that has not been returned yet. As we only assume products are returned that are sold between 1 and M periods ago, $m \in \mathcal{M}$ with $\mathcal{M} = \{1, 2, \ldots, M\}$. We assume that the return probability is independent of the period in which the product is sold. If there would be a reason to model these probabilities differently, ρ could be made dependent on m as the binomial distributions of unreturned products sold in period m are independent.

3.2. Model

3.2.1. State space

The state space of the inventory state at level I is given by $I \in \mathcal{I}$ with $\mathcal{I} = \{0, 1, \ldots, (T + \ell) \cdot \sum_i D_i + M \cdot T \cdot D_2\}$. The inventory state is limited by the maximum expected demand and the returns of the last M periods. We assume the retailer will never order more than the maximum expected demand over the period plus lead time. However, products sold online over the last M periods ago could theoretically all be returned and added to the inventory. The maximum inventory level is therefore given by $(T + \ell) \cdot \sum_i D_i + M \cdot T \cdot D_2$. The state space of the unreturned product sold m periods ago is limited by the maximum amount of expected online sales in a period: $\boldsymbol{R} \in \mathcal{R}$ with $\mathcal{R} = \{0, 1, \ldots, T \cdot D_2\}$.

The state space of the inventory at level II is equal to the state space at level I, $I \in \mathcal{I}$. The state space of the order quantity is limited by the maximum expected demand of the review period minus the inventory position: $Q \in \mathcal{Q}(I)$ with $\mathcal{Q}(I) = \{0, 1, \ldots, T \cdot \sum_i D_i - I\}$. The state space of the sold product m periods ago is equal to that at level I, $\boldsymbol{R} \in \mathcal{R}$.

The state space of the products sold in the current period is dependent on the sub-period, as it is limited by the maximum amount of expected online sales per sub-period: $R_0 \in \mathcal{P}(t)$ with $\mathcal{P}(t) \in \{0, 1, \ldots, t \cdot D_2\}$.

3.2.2. Action and action space

At level I of the MDP, the order quantity q is determined at the beginning of the period, and is limited by the order quantity state: $q \in \mathcal{Q}(I) = \{0, 1, \ldots, T \cdot \sum_i D_i - I\}$.

At level II of the MDP, a rationing decision a is made every sub-period t. This rationing decision decides how many products are stored in the channels. The action space of the rationing decision is dependent on the current inventory position: $a \in \mathcal{A}(I)$ with $\mathcal{A}(I) = \{0, 1, \ldots, I\}$. The rationing decision is made at the beginning of the sub-period and is not revised throughout the sub-period.

3.2.3. State transitions

At level I of the MDP the state transitions every period, while at level II of the MDP the state transitions every sub-period. At level II of the MDP the state transition from $s = (I, R_0, \boldsymbol{R}, Q, t)$ to $s' = (I', R'_0, \boldsymbol{R}', Q', t + 1)$ occurs. The transition of the inventory I to I' is dependent on the demand, rationing decision, returns, replenishment, and the sub-period:

$$I' = (a - d_1)^+ + ((I - a) - d_2)^+ + \sum_{m=1}^{M} r_m + \delta(t = \ell) \cdot Q \tag{1}$$

where $x^+ = \max(0, x)$ and $\delta(x)$ denotes the Kronecker delta, which gives the value 1 if $x = \text{True}$, otherwise 0. The first term of the equation refers to the demand satisfied in the offline channel, in which the inventory level is a and the demand occurring d_1. The second term refers to the demand satisfied in the online channel, in which the inventory level is $I - a$ and the demand occurring d_2. The third term of the equation refers to the returned products, which is the sum of the products returned that are sold between 1 and M periods ago. The last term refers to the replenishment, which occurs at $t = \ell$, which is at the beginning of sub-period ℓ.

The transition of the number of products sold online in the current period R_0 to R'_0 is dependent on the number of product sold online the previous sub-period and the products sold online in the current sub-period:

$$R'_0 = R_0 + \min(d_2, I - a) \tag{2}$$

where the product sold online in the current sub-period are added to those where already sold in the subsequent sub-period. The transition of unreturned products m periods ago, R_m to the next state, R'_m is dependent on the quantity of products being returned

r_m. Only unreturned products sold more than 1 period ago can be returned, with a maximum of M periods.

$$R'_m = R_m - r_m \qquad \forall m \in \mathcal{M} \tag{3}$$

The transition of the replenishment quantity Q to Q' depends on the sub-period. If the replenishment has yet to occur, Q remains constant, otherwise it is set to zero:

$$Q' = \begin{cases} Q & \text{if } t < \ell \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

3.3. Bellman equation and value iteration

The objective of the MDP is to find the optimal policy that maximises the long-term discounted profit. The expected future discounted profit when following an optimal policy over n consecutive sub-periods when starting in state s is defined as $v_n(s)$. To obtain the optimal policy by value iteration one starts setting $v_0(s) = 0$ for all states s. Next, one computes for all $s : v_1(s) = \max_{a \in \mathcal{A}(I)} \{\mathbb{E}R_{\text{II}}(s,a)\}$, and continues by computing v_2, v_3, etc. using the recursive Bellman equation in equation (8).

$$v_n(s) = \begin{cases} \max\limits_{q \in \mathcal{Q}(I)} \left\{ \mathbb{E}R_{\text{I}}(s,q) + \max\limits_{a \in \mathcal{A}(I)} \left\{ \mathbb{E}R_{\text{II}}(s,a) \right.\right. \\ \qquad \left.\left. + \gamma \sum\limits_{d_1=0}^{D_1} \sum\limits_{d_2=0}^{D_2} \sum\limits_{m=1}^{M} \sum\limits_{r_m=0}^{R_m} P_1(d_1) P_2(d_2) B(r_m) \cdot v_{n-1}(s') \right\} \right\} & \text{if } n \bmod T = 0 \\ \max\limits_{a \in \mathcal{A}(I)} \left\{ \mathbb{E}R_{\text{II}}(s,a) + \gamma \sum\limits_{d_1=0}^{D_1} \sum\limits_{d_2=0}^{D_2} \sum\limits_{m=1}^{M} \sum\limits_{r_m=0}^{R_m} P_1(d_1) P_2(d_2) B(r_m) \cdot v_{n-1}(s') \right\} & \text{otherwise.} \end{cases} \tag{8}$$

At level I of the MDP the state only transitions at the end of the period, where the information of the state at level II is obtained to form the new state of level I: $S' = (I'', \boldsymbol{R}'')$. I'' is the closing inventory of the period, which is obtained after the last inventory state transition of level II of the MDP when $t = T$.

The state transition of the number of unreturned products for level I is obtained after the last state transition of R_0 and $\boldsymbol{R}$ from level II of the MDP when $t = T$. As we keep track of the periods ago a products was sold and the period has ended, unreturned products that were returned $m-1$ periods ago are now m periods old. Unreturned products older than M periods are not eligible to be returned anymore, thus we do not keep track of these items.

$$R''_m = R'_{m-1} \quad \forall m \in \mathcal{M} \tag{5}$$

3.2.4. Expected immediate reward

The goal of the MDP is to maximise its profit, which consists of the revenue, and several costs. The expected immediate reward at level I of the MDP only consists of the ordering cost and is given as follows:

$$\mathbb{E}R_{\text{I}}(S,q) = -q \cdot c_p \tag{6}$$

In which c_p is the procurement cost of the product. The expected immediate reward per sub-period at level II of the MDP consists of revenue sold from selling products, online fulfilment cost, holding costs and handling costs of returns:

$$\mathbb{E}R_{\text{II}}(s,a) = \begin{cases} p\left(\sum\limits_{d<a} d \cdot P_1(d) + \sum\limits_{d \geq a} a \cdot P_1(d) \right) & (7.1) \\ +(p - c_u)\left(\sum\limits_{d<I-a} d \cdot P_2(d) + \sum\limits_{d \geq I-a} (I-a) \cdot P_2(d) \right) & (7.2) \\ -(c_{h1} \cdot a + c_{h2} \cdot (I-a)) & (7.3) \\ -(p + c_r) \sum\limits_{m=1}^{M} R_m \cdot \rho & (7.4) \end{cases}$$

The first term (7.1) of the expected immediate reward consists of the revenue of the offline channel given price p. The same sales price is applied for the online channel, however, a shipment cost of c_u is incurred as seen in the second term (7.2). As customers are expecting free shipping, the retailer has to incur the cost. The third term (7.3) is the holding cost of the offline and online channel, c_{h1} and c_{h2} per product respectively. The last term (7.4) is the cost from expected returns of M periods ago, $p + c_r$ is the handling cost of a return plus the price of the product, as customer receive full return payment.

Here, γ is the discount factor that is included to ensure a fair comparison between VI and the solution technique of DRL, which requires discounting of future rewards. The first equation in (8), where n is a multiple of T, relates to the ordering and rationing decision, which both happen at the beginning of the period consecutively. The second equation relates to the other time periods, in which only the rationing decision is involved.

Let n be the iteration counter. The Markov process is unichain and aperiodic at level I, and unichain but periodic at level II with period T. Hence the difference $v_n - v_{n-T}$ converges for all states to the same value, as n moves to infinity. A discounting factor γ is included for a fair comparison with DRL, which requires discounting for stabler training as future rewards might be uncorrelated in DRL. For $\gamma < 1$, the difference $v_n - v_{n-T}$ converges to zero, as n tends to infinity. If $\|v_n - v_{n-T}\|$ is smaller than $\varepsilon(1-\gamma)/\gamma$, the value iteration stops (we specify $\varepsilon = 0.1$). For details on the speed and conditions for convergence we refer to Puterman (1994). As at level II the problem is periodic with period T, one full iteration consists of T sub-periods.

The (nearly) optimal strategy for the two levels of the MDP can be obtained from the results of the value matrices. First the optimal replenishment policy $q^*(s)$ at $t = 1$ for level I is found by:

$$q^*(s) = \arg\max_{q \in \mathcal{Q}(I)} \left\{ \mathbb{E}R_{\text{I}}(s,q) + \max_{a \in \mathcal{A}(I)} \left\{ \mathbb{E}R_{\text{II}}(s,a) \right.\right. \\ \left.\left. + \gamma \sum_{d_1=0}^{D_1} \sum_{d_2=0}^{D_2} \sum_{m=1}^{M} \sum_{r_m=0}^{R_m} P_1(d_1) P_2(d_2) B(r_m) \cdot v_{n-1}(s') \right\} \right\} \tag{9}$$

Second, the optimal rationing decision $a^*(s)$ for all states in $t = 1, \ldots, T$ for level II is found by:

$$a^*(s) = \arg\max_{a \in \mathcal{A}(I)} \left\{ \mathbb{E}R_{\text{II}}(s,a) \right. \\ \left. + \gamma \sum_{d_1=0}^{D_1} \sum_{d_2=0}^{D_2} \sum_{m=1}^{M} \sum_{r_m=0}^{R_m} P_1(d_1) P_2(d_2) B(r_m) \cdot v_{n-1}(s') \right\} \tag{10}$$

4. Implementation of deep reinforcement learning algorithm: proximal policy optimisation

For small-scale settings the Bellman equations can be solved to optimality within reasonable computation time and RAM usage.

However, for large-scale settings, the action and state space becomes increasingly large resulting in infeasible computational time or memory limitations. DRL is considered to be useful to circumvent the curse of dimensionality, as it can be used to develop near-optimal solutions that could not be obtained using conventional approaches (Boute et al., 2022). In DRL, many different algorithms are used. They all follow the general structure of an agent interacting with an environment and collecting experience to find the best policy. This policy is found by using the collected experience to train an approximation of the state values and policy, in which the approximations often take the form of a neural network.

Actor-critic methods are reinforcement learning algorithms in which both policy gradient and value estimation are applied. The actor and the critic are the two interrelated components making up the method. The critic provides an estimation of the expected future discounted profit, which is used by the actor to apply gradient descent on the policy. Asynchronous Advantage Actor-Critic (A3C) is a well-known actor-critic method that has proven to perform well for various inventory management problems (Gijsbrechts et al., 2022). However, the A3C algorithm has the disadvantage of needing a lot of training data and extensive parameter tuning. In order for the actor to converge, it needs to have an almost infinite amount of training data. Recently, Vanvuchelen et al. (2020) have shown that another actor-critic method called Proximal Policy Optimisation (PPO) also performs well in an inventory management setting and that it is less sensitive to the disadvantages of A3C. PPO is praised for its sample efficiency and ease of implementation (Schulman et al., 2017).

The PPO algorithm trains the actor and the critic, which are both represented by a neural network. The input of these neural networks is the state of the system. The output for the actor is the action and the output for the critic is an estimation of the expected future discounted profits for the input state. As the hierarchical MDP discussed in this paper has two levels, we use two actor-critic sets and four neural networks in this paper. The advantage of having a separate actor-critic set for each action is that it decreases the complexity of training the actor neural network. This could improve training, as the neural networks only have to approximate one action. The architecture of the neural networks is identical for all four networks, except for the output layer.

The neural networks consists of two fully connected layers with width 256, all with a tanh activation function and a bias. The tanh activation function has the advantage of non-linearity and has been proven to be better at quickly finding local (or global) minima as the derivatives can be large (LeCun et al., 2012). The four neural networks have different applications, and therefore the parameters of the neural network, which are the weights and bias, are different. The parameters of the actor and critic neural network are given by θ_h and ϕ_h respectively, where the level of the MDP is given by $h \in \{1 = \text{level I}, 2 = \text{level II}\}$.

For the actor, the last layer is a softmax activation function, which provides a probability distribution over the action space denoted by $\pi(\cdot|S, \theta)$ with S being the input state. Therefore, the size of the last layer is equal to the action space of the level. From the actor we can get the best action by taking the action with highest probability. For the critic, the size of the last layer is one, as the output is the estimation of the expected future profit for state S is denoted by $V(S, \phi)$.

4.1. PPO algorithm

To update the neural networks, the PPO algorithm needs to collect information about the environment. This is performed by sampling the studied problem and storing the visited states, actions, and related profits during the sampling. Fig. 2 visualises the implementation of the PPO algorithm with the sampling, updating of the neural network, and the evaluation of the policy found. We sample for B periods and thus $T \cdot B$ sub-periods, and store the visited states, taken actions, and received profits. As our MDP has two levels, the storing of the training data differs and we store them separately. The algorithm starts with initialising the neural network parameters randomly and setting the training iteration counter (κ) to one.

4.2. Sampling

The sampling is started by drawing a random state S_1, setting the sub-period $t = 1$, and setting the sample points K and k to one. The state S_K is used in the actor and critic of level I to get the ordering action q_K and expected future profit $V(S_K, \phi_1)$. The ordering action q_K is obtained by randomly sampling from the probability distribution of the ordering action space, $q_K \sim \pi(\cdot|S_K, \theta_1)$. The ordering action is put into environment level I to get the state s_k, which is used by the actor and critic of level II. Additionally, from environment level I we collect the profit E_K when taking action q_K when in state S_k, this information combined with $V(S_K, \phi_1)$ is stored in the training data of level I. From the actor and critic of level II a rationing action is drawn from the probability distribution $a_k \sim \pi(\cdot|s_k, \theta_2)$ and expected future profit $V(s_k, \phi_2)$ is gained, where the rationing action is put in environment level II to get the reward e_k, which together with the state and action is stored in training data of level II. The demand and returns that occur in the environment level II are randomly drawn from their respective distributions. Furthermore, the next state s_{k+1} is collected from environment level II. This next state is used as input for the neural network of level II if the sub-period has not ended yet, which is determined by $t \leq T$, else this state is used as input for the neural network of level I. Every time a sample is drawn from environment level II, both the sample point k and sub-period t are increased by one. When the sub-period has ended, the state s_k is used to construct state S_{K+1}, t is set to one indicating we are at the beginning of the period again, and the sample point K is increased by one. When $K = B$ and $k = T \cdot B$ the sampling is done and the neural networks are updated.

4.3. Update neural networks

First, with the collected training data the advantage and discounted profit are calculated. Secondly, the training data, advantage, and discounted profit are shuffled and split into η mini-batches to calculate the average loss over each mini-batch. Each mini-batch consecutively updates the neural networks once. For each mini-batch the loss is minimised by updating the neural networks parameters θ_h and ϕ_h to θ'_h and ϕ'_h. The procedure of calculating the loss and how the parameters of the neural network are updated with respective to the loss is given in Appendix A.

The process of splitting the training data, advantage, and discounted profit and updating the neural networks is called an epoch. As we are using the collected training data, advantage, and discounted profits multiple times to update the neural networks a total of u epochs are performed. In addition, in every epoch the training data is shuffled, to create new configuration of mini-batches in each epoch. This ensures that the updates of the neural networks are not over-fitted to the training data. Thus, the parameters of the neural networks are updated $\eta \cdot u$ times, were each sample point in the training data is used u times.

4.4. Evaluation

Once the neural networks are trained u epochs, the training iteration counter κ is updated by one. After updating the neural networks for $\kappa_{\max}$ iterations, where each training iteration consists

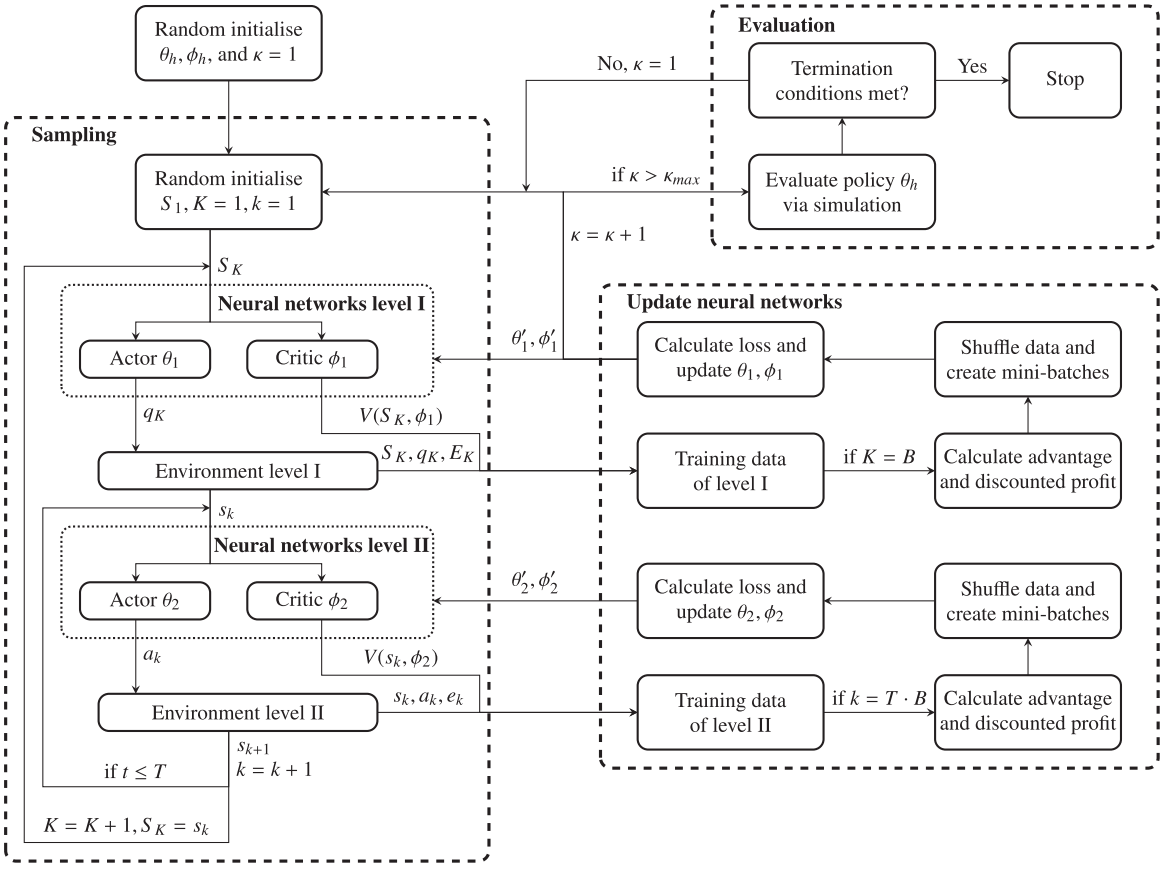

Fig. 2. PPO algorithm flow diagram.

of B periods, the resulting policy is evaluated. We follow a similar procedure as Vanvuchelen et al. (2020). The policy is simulated for 1,000,000 periods considering a small warm-up period. During the sampling the action to be taken is randomly drawn from the output of the actor, which is a probability distribution over the actions, whereas in the evaluation we take the action with the highest probability as this is considered to be the best found action for the given state. Thus for the ordering action we take $q = \arg\max(\pi(\cdot|S, \theta_1))$ and for the rationing action $a = \arg\max(\pi(\cdot|s, \theta_2))$.

To measure the performance, the average profit per period from the simulation is used. If the average profit per period of the best resulting policy has not changed more than 0.5% over three consecutive policy evaluations, we assume the algorithm has converged to its final policy. A maximum of 30 policy evaluations are considered, if the maximum is reached the best performing policy is assumed to be the final policy.

4.5. Implementation

4.5.1. Parameters

To implement the PPO algorithm, several parameters and input data need to be defined. The tuning of reinforcement learning parameters can be computational costly (Gijsbrechts et al., 2022), thus we started from the settings of related work such as Vanvuchelen et al. (2020). We did however adapt some of the parameters to our problem setting, and performed some experiments to improve the performance of our PPO algorithm. This was done by tuning the parameters and investigating the results for a test case. We evaluated different neural network architectures, sample periods, amount of mini-batches, learning rate, entropy regularisation, and the Huber loss constant. Table 1 describes the parameters used.

Table 1
Parameter values used in the implementation of the PPO algorithm.

Hyperparameters	Value
Depth of NN	2
Width of NN	256
Activation function	TanH
Sample periods (B)	512
Amount of mini-batches (η)	4
Number of epochs (μ)	10
Training iterations ($\kappa_{\max}$)	1000
Learning rate (α)	10^{-4}
Entropy regularisation (β_E)	10^{-5}
Clipping parameter (ϵ)	0.2
Huber loss constant (δ)	5

Our model has relatively large demand fluctuations during the period compared to related work, a larger sampling period is used to reduce uncertainty in the update of the network. Furthermore, the width of the neural network is increased, as our action space is relatively large and should not exceed the width of the neural network. The initial bias and weights of the neural network are sampled from a Normal distribution with mean 0 and standard deviation 0.5. Therefore the initial policy is a random policy.

4.5.2. Scaling of states and input data

By normalising the input data of the neural network it is expected to converge quicker to the training data (LeCun et al., 2012). The input of the neural network, which is the state, is therefore scaled so that the minimum value of the state is -5 and the largest

value of the state is 5. Additionally, the economic parameters are scaled from 0 to 1.

4.5.3. Action masking

As described in Section 3.2.2, the action space is limited by the current state for the replenishment and rationing action. To ensure the PPO algorithm does not consider invalid actions, we apply action masking using the method of Huang & Ontañón (2020). This method sets the output of invalid actions in the actor neural network before the softmax activation layer to a small negative value (in our case -10^{-8}), so that after the softmax activation layer the probability of choosing these actions becomes negligible. An additional advantage of action masking is that by masking the action space, the PPO algorithm can learn faster as it quickly learns to disregard the invalid actions.

5. Computational complexity and performance

As mentioned in Section 4, only small-scale instances can normally be solved with the Bellman equation. To compare the computational complexity and performance of PPO with VI, we develop a range of small-scale instances, consisting of a base test case and various alterations. The data set used is based on recent literature studying similar omnichannel retail settings with a few alterations to reduce the size of the problem (Bayram & Cesaret, 2021; Dijkstra et al., 2019; Goedhart et al., 2022; Li et al., 2015; Ovezmyradov & Kurata, 2019). Additionally, we investigate the effect of modelling the returns multi-period and sales-dependent by comparing it with two other methods that approximate the return flow.

5.1. Base test case and alterations

For the base test case, we assume average demand of the individual channels is assumed to be Poisson distributed with $\mu_1 = 3$ and $\mu_2 = 1$. As one needs a finite support, the distribution is right truncated at a cumulative probability of 99%, preserving the mean value using the approach by Cohen (1954) is used. Thus the maximum demand in a sub-period is $D_1 = 8$ and $D_2 = 4$.

We assume a period consists of 7 sub-periods ($T = 7$), to reflect a weekly ordering decision. Replenishment orders are placed on Monday ($t = 1$) with lead time $\ell = 2$, thus delivered on Wednesday ($t = 3$). Here both events occur at the beginning of the sub-period.

The probability of a product being returned is $\rho_p = 0.4$ in a return window $M = 2$ periods (Dijkstra et al., 2019). The probability of a product being returned in a sub-period is $\rho = 1 - (1 - \rho_p)^{1/(T \cdot M)} = 0.036$. Similar to the demand distribution, we truncate the return distribution at a cumulative return probability of 99%. Additionally, we reshape the binomial distribution according to the approach of Johnson et al. (2005) to preserve the mean value. Although a binomial distribution has a natural maximum value, we truncate the return distribution to reduce the size of the problem.

The economic parameters are based on literature studying similar settings (Bayram & Cesaret, 2021; Li et al., 2015; Ovezmyradov & Kurata, 2019). We set the price of the product at $p = 100$ and the cost of the product $c_p = 30$. The handling cost incurred for satisfying an online order is set at $c_u = 5$. The holding cost of the offline and online channel are $c_{h1} = 1$ and $c_{h2} = 0.5$, respectively. The handling cost of a returned product is $c_r = 5$. The discount factor is $\gamma = 0.99$, similarly as in Vanvuchelen et al. (2020).

The different instances are created by varying specific subsets of parameters. Table 2 presents the values for the specific parameters. The parameter values for the base case are listed in bold.

The different instances in which $\mu_2 = F(\rho)$ represent scenarios in which the online return percentages vary, but the net online sales remains constant. These instances are created to investigate how the uncertainty of returns influence the profitability of the retailer, without it being influenced by the net online sales. The online demand is calculated as follows:

$$F(\rho) = \frac{\mu_2^*(1 - \rho_p^*)}{(1 - \rho)} \tag{11}$$

Here, μ_2^* and ρ_p^* are the online demand and return percentage of the base test case. Other instances are created by halving the coefficient of variation (CV_i). For alternative values of CV_i, the demand distribution is fitted to μ_i and CV_i using the procedure described in Adan et al. (1995). All demand distributions are reshaped as right-truncated distributions (Cohen, 1954; Johnson et al., 2005; Louchard & Ward, 2015; Shah, 1966), truncating at a probability of 99% of the cumulative distribution function. In the remainder of this paper, if a instance does not specify a certain parameter value, it will be equal to their base test case setting.

Table 2
Parameter values for small-scale instances, where the base case is given in bold.

Parameters	Value
μ_1	1, 2, **3**
μ_2	**1**, $F(\rho)$
CV_i	$1/(2\sqrt{\mu_i})$, $\sqrt{\mathbf{1/\mu_i}}$
ρ_p	0.2, 0.3, **0.4**
M	1, **2**
c_r	0, **5**, 10
c_p	**30**, 50, 70
c_{h1}	0.1, 0.2, **1**
c_{h2}	0.05, 0.1, **0.5**
ℓ	1, **2**, 3

5.2. Value iteration vs proximal policy optimisation

The results presented in this section were obtained by implementing VI in Python version 3.7.2. For the neural network models of the PPO algorithm TensorFlow 2.3.0 was used. The model was run on a Personal Computer with Intel Xeon W-2133 CPU @ 3.60 gigahertz and 32 GB of RAM.

Table 3 presents the required computational time, RAM usage, and optimality gap for both algorithms, as well as the total amount of states in the system. The RAM usage is mainly dependent on the size of the value matrix (as this changes due to the ordering state becoming zero after replenishment, it is expressed as the maximum RAM usage during an iteration). The optimality gap is the difference in profit between the optimal solution derived via VI and the policy found by the PPO algorithm. The profit for the policy found by VI and PPO is obtained by simulating the resulting policies of both algorithms for 1,000,000 periods with a small warm-up period.

From Table 3, it is observed that the demand, coefficient of variation, returns, and lead time increase the state space exponentially. Especially for the return percentages the curse of dimensionality is clear, as an increase of 10% in return percentages doubles the amount of states in the model. For some return percentages, a similar amount of states can be seen. This is due to the truncation of the distribution function for the number of returns in a sub-period, as the truncation then happens at the same value. Increasing the lead time increases the number of sub-periods the ordering state is included in the state space and the maximum order quantity, therefore increasing the amount of states significantly.

It is observed that for the small-scale instances the CPU time of VI is much lower compared to the PPO algorithm. The CPU time of VI grows exponentially with the number of states and state transitions, the CPU time of the PPO algorithm is less sensitive to the dimensionality of the instance. VI solves the problem iteratively backwards via the Bellman equation, where for each possible state, the action and related probability distribution on the next states is

Table 3
Computational complexity and performance for VI and the PPO algorithm for the small-scale instances.

Instance	States (in millions)	CPU time (hours)		RAM (MB)		Optimality gap (%)
		MDP	PPO	MDP	PPO	
$\mu_1 = 3, \mu_2 = 1$[a]	205.8	17.87	85.7	4283.1	6.1	−1.64
$\mu_1 = 2, \mu_2 = 1$	153.6	12.19	104.2	3170.0	5.4	−2.20
$\mu_1 = 1, \mu_2 = 1$	108.8	7.61	146.6[b]	2219.5	4.7	−2.73
$CV_i = 1/\sqrt{\mu_i}$[a]	205.8	17.87	85.7	4283.1	6.1	−1.64
$CV_1 = 1/(2\sqrt{\mu_1})$	130.3	6.04	141.8	2674.5	5.1	−1.72
$CV_2 = 1/(2\sqrt{\mu_2})$	59.3	0.55	142.3	1275.8	5.0	−1.84
$CV_i = 1/(2\sqrt{\mu_i})$	33.7	0.24	142.0	709.1	3.9	−1.33
$\rho_p = 0.2$	43.8	12.72	128.8	521.9	5.3	−2.12
$\rho_p = 0.3$	107.1	17.48	100.3	1746.2	5.7	−2.29
$\rho_p = 0.4$[a]	205.8	17.87	85.7	4283.1	6.1	−1.64
$\rho_p = 0.5$	431.8	30.24	145.8[b]	8777.4	6.5	−2.20
$\rho_p = 0.6$	431.8	31.93	146.1[b]	8777.4	6.5	−2.03
$\rho_p = 0.2, \mu_2 = F(\rho_p)$	18.8	12.72	123.3	440.4	5.0	−2.12
$\rho_p = 0.3, \mu_2 = F(\rho_p)$	80.3	17.48	149.2[b]	1741.2	5.7	−2.29
$\rho_p = 0.4, \mu_2 = F(\rho_p)$[a]	205.8	17.87	85.7	4283.1	6.1	−1.64
$\rho_p = 0.5, \mu_2 = F(\rho_p)$	431.8	30.24	123.3	8777.4	5.0	−2.20
$\rho_p = 0.6, \mu_2 = F(\rho_p)$	903.4	31.93	123.3	18131.9	5.0	−2.03
$M = 1$	10.5	0.85	147.0[b]	149.6	5.7	−2.33
$M = 2$[a]	205.8	17.87	85.7	4283.1	6.1	−1.64
$c_r = 0$	205.8	22.63	145.6[b]	4283.1	6.1	−2.11
$c_r = 5$[a]	205.8	17.87	85.7	4283.1	6.1	−1.64
$c_r = 10$	205.8	22.78	148.6[b]	4283.1	6.1	−2.44
$c_p = 30$[a]	205.8	17.87	85.7	4283.1	6.1	−1.64
$c_p = 50$	205.8	23.41	144.4[b]	4283.1	6.1	−2.04
$c_p = 70$	205.8	24.33	126.4	4283.1	6.1	−0.89
$c_{h1} = 0.1, c_{h2} = 0.05$	205.8	23.96	125.0	4283.1	6.1	−1.97
$c_{h1} = 0.2, c_{h2} = 0.1$	205.8	24.40	126.9	4283.1	6.1	−0.81
$c_{h1} = 1, c_{h2} = 0.5$[a]	205.8	17.87	85.7	4283.1	6.1	−1.64
$\ell = 1$	63.3	3.30	135.9	3325.7	5.4	−2.40
$\ell = 2$[a]	205.8	17.87	85.7	4283.1	6.1	−1.64
$\ell = 3$	493.3	58.63	137.9	5360.8	6.9	−2.34
Average	214.9	18.39	126.9	4251.3	5.6	−1.95

[a] Base test case with $\mu_1 = 3, \mu_2 = 1, CV_i = 1/\sqrt{\mu_i}, \rho_p = 0.4, M = 2, c_r = 5, c_p = 30, c_{h1} = 1, c_{h2} = 0.5, \ell = 2$.
[b] Instances in which the policy did not converge within 30 iterations.

calculated and used to update the state value. Therefore, the CPU time of VI does not solely depend on the number of states but also on the number of state transitions and the action space resulting in larger computational time.

The PPO algorithm is less sensitive to the dimensionality of the problem as it approximates the state values and actions by a neural network. The results of the simulation are used to update the neural network, which generalises across similar states. As the PPO algorithm does not need to visit each state to solve the problem, the CPU time does not increase with the number of states. However, as the PPO algorithm is less effective in finding the optimal action for each state compared to VI, the computational time is often larger. The PPO algorithm uses simulation and random initial weights of the neural network, varying the computational time used for training per instance. Furthermore, the visited states, rewards, and actions are randomly simulated, influencing the training as this information is used to update the neural network.

From Table 3 it can be concluded that the RAM usage of the MDP increases with the size of the instance. The RAM usage of the MDP is less sensitive to the demand and lead time, as these increase the size of the value matrix to a limited extent. The RAM usage is more sensitive to the return distribution, as this influences the state space of multiple states. The RAM usage of the PPO algorithm is less sensitive to parameter values as the only information it needs to store are the parameters of the neural network and the samples of the simulation. These are less dependent of the size of the problem.

From Table 3 it can be observed that the largest optimality gap is found for the instance of $\mu_1 = 1, \mu_2 = 1$. Here, the PPO algorithm yields 2.73% less profit than the optimal solution. The PPO algorithm performs best for the instance of $c_{h1} = 0.2, c_{h2} = 0.1$, in which the profit gap is −0.81%.

When looking at the underlying cost and how the found policy of the PPO algorithm differs from the optimal policy (see also Table B.1 in Appendix B and Table C.1 in Appendix C), it is observed that instances in which the PPO algorithm has difficulty in finding the optimal policy is when around the optimal solution the profit difference is minimal. Finding the optimal action is then difficult as performing non-optimal minimally influences the profit. This is observed as the rationing decision at high inventory level differs the most between the two policies. At a high inventory level the rationing decision has less influence on the profit as there is no competition between the channels for products. Therefore, at high inventory levels the rationing decision is more focused on cost minimisation. Additionally, when the action state space is large the PPO algorithm has more decisions to evaluate, increasing the difficulty of finding the optimal action. Consequently, at low inventory levels, the PPO algorithm can more easily identify the optimal rationing decision as the action space is limited by the inventory level and the channels compete for products, thus the rationing decision also influences the profit more.

For the instances with decreasing demands, we notice that the optimality gap is growing. A similar trend can be observed for increasing coefficients of variation in demand. This suggests that

Table 4
Computational complexity and performance for the different modelling approaches.

	CPU time (hours)			RAM usage (MB)			Optimality gap (%)	
Instance	Multi R	Agg. R	Ind. R	Multi R	Agg. R	Ind. R	Agg. R	Ind. R
$\mu_1 = 3, \mu_2 = 1$[a]	17.87	1.82	0.23	4283.1	199.7	53.0	−1.03	−8.88
$\mu_1 = 2, \mu_2 = 1$	12.19	1.13	0.37	3170.0	124.1	31.0	−1.57	−14.87
$\mu_1 = 1, \mu_2 = 1$	7.61	0.58	0.85	2219.5	69.6	15.9	−3.04	−33.44
$CV_i = 1/(\sqrt{\mu_i})$[a]	17.87	1.82	0.23	4283.1	199.7	53.0	−1.03	−8.88
$CV_1 = 1/(2\sqrt{\mu_1})$	6.04	0.61	0.20	2674.5	94.5	26.6	−1.11	−15.56
$CV_2 = 1/(2\sqrt{\mu_2})$	0.55	0.72	0.17	1275.8	127.3	22.2	−0.97	−3.23
$CV_i = 1/(2\sqrt{\mu_i})$	0.24	0.31	0.13	709.1	57.4	10.7	−1.04	−6.38
$\rho_p = 0.2$	12.72	0.49	0.17	521.9	73.7	42.1	−0.03	−2.99
$\rho_p = 0.3$	17.48	1.31	0.20	1746.2	129.2	42.1	−0.88	−5.00
$\rho_p = 0.4$[a]	17.87	1.82	0.23	4283.1	199.7	53.0	−1.03	−8.88
$\rho_p = 0.5$	30.24	2.64	0.33	8777.4	464.6	70.7	−0.78	−19.27
$\rho_p = 0.6$	31.93	2.71	0.52	8777.4	613.1	78.2	−0.37	−36.93
$\rho_p = 0.2, \mu_2 = F(\rho_p)$	2.59	0.40	0.15	440.4	40.5	12.8	−0.17	−3.12
$\rho_p = 0.3, \mu_2 = F(\rho_p)$	17.74	1.25	0.20	1741.2	105.5	38.0	−0.74	−6.79
$\rho_p = 0.4, \mu_2 = F(\rho_p)$[a]	17.87	1.82	0.23	4283.1	199.7	53.0	−1.03	−8.88
$\rho_p = 0.5, \mu_2 = F(\rho_p)$	28.70	3.11	0.30	8777.4	464.6	70.7	−1.00	−17.07
$\rho_p = 0.6, \mu_2 = F(\rho_p)$	83.63	5.40	0.78	18131.9	992.8	148.5	−0.73	−57.78
$\rho_w = 1$	0.85	0.50	0.23	149.5	73.7	27.3	−1.40	−10.73
$\rho_w = 2$[a]	17.87	1.82	0.23	4283.1	199.7	53.0	−1.03	−8.88
$c_r = 0$	22.63	1.55	0.23	4283.1	199.7	53.0	−0.76	−8.83
$c_r = 5$[a]	17.87	1.82	0.23	4283.1	199.7	53.0	−1.03	−8.88
$c_r = 10$	22.78	1.61	0.23	4283.1	199.7	53.0	−1.39	−8.97
$c_p = 30$[a]	17.87	1.82	0.23	4283.1	199.7	53.0	−1.03	−8.88
$c_p = 50$	23.41	2.19	0.25	4283.1	199.7	53.0	−1.78	−14.55
$c_p = 70$	24.33	1.96	0.25	4283.1	199.7	53.0	−2.51	−29.22
$c_{h1} = 0.1, c_{h2} = 0.05$	24.40	1.83	0.23	4283.1	199.7	53.0	−0.80	−5.69
$c_{h1} = 0.2, c_{h2} = 0.1$	23.96	1.86	0.23	4283.1	199.7	53.0	−0.83	−6.24
$c_{h1} = 1, c_{h2} = 0.5$[a]	17.87	1.82	0.23	4283.1	199.7	53.0	−1.03	−8.88
$\ell = 1$	3.30	0.82	0.22	3325.7	152.2	37.6	−0.97	−7.20
$\ell = 2$[a]	17.87	1.82	0.23	4283.1	199.7	53.0	−1.03	−8.88
$\ell = 3$	58.63	3.59	0.27	5360.8	253.6	71.0	−1.04	−12.40
Average	20.60	1.67	0.29	4251.32	227.58	48.54	−1.08	−14.57

[a] Base test case with: $\mu_1 = 3, \mu_2 = 1, CV_i = 1/\sqrt{\mu_i}, \rho_p = 0.4, M = 2, c_r = 5, c_p = 30, c_{h1} = 2, c_{h2} = 1, \ell = 2$.

having a higher dispersion of the demand distribution negatively influences the training of the PPO algorithm. With a higher coefficient of variation, the rewards that the PPO algorithm experiences during sampling are more dispersed as this is partly driven by demand. To better distinguish good actions from bad actions, more training data is needed to mitigate the influence of the larger dispersed demand distribution.

Overall, the PPO algorithm shows good performance, with a profit optimality gap of around −2.0%. The advantage of the PPO algorithm is that it is not influenced much by the dimensions of the studied problem, as CPU time and RAM usage are relatively stable. The PPO algorithm does however experience difficulties in finding the optimal policy when the uncertainty increases. Furthermore, the rewards influence the training, as these influence the trade-off the PPO algorithm makes between different actions. As the training of a reinforcement learning algorithm is influenced by the received rewards, and one can manipulate them to find better policies, this is often referred to as reward shaping in the reinforcement learning literature.

5.3. Modelling returns

Current literature mentions several methods to model returns. In this paper, we explicitly model the returns as multi-period and sales-dependent (which we refer to as Multi R in Table 4). To investigate how much influence this approach has on the retailer's inventory management, we compare it with two existing modelling approaches: (1) sales-independent returns (Ind. R) and (2) sales-aggregated returns (Agg. R).

Modelling the returns independent of sales, as we do in Ind. R, results in returns being an exogenous flow, similar to having a negative demand (see e.g. Feinberg & Lewis, 2005). We approximate the return flow as a negative demand with a Binomial distribution with ρ as the probability a single product is returned, and average online sales over the return window ($\mu_2 \cdot M \cdot T$) as the pool of returnable items. With regards to our MDP, the states R_0 and $\boldsymbol{R}$ are not relevant anymore when approximating the returns as independent of sales and are therefore omitted.

For the approach of Agg. R, we model the returns by aggregating the state $\boldsymbol{R}$ into a single state R and omit R_0. The state R now approximates the total online sales in the past M periods. However, as the expiration date of whether a products can be returned is not included in the state it needs to be approximated. We therefore define the state transition of R to R' as follows:

$$R' = (R - r - h + \min(d_2, I - a))^+ \tag{12}$$

Here, r represents the number of products being returned (determined by a Binomial distribution with ρ as the probability a single product is returned), R is the pool of returnable items, h is the approximated online sales that cannot be returned anymore, and the last term is the online sales of the current sub-period. We approximate h using a Poisson distribution with an average value of $\mu_2 \cdot (1 - \rho_p)$ as this the number of products that on average are not returned.

Table 5
Parameter values for large-scale instances, where the base case is given in bold.

Parameters	Value
μ_1	2, 4, **6**
μ_2	**2**, 4, 6, $F(\rho_p)$
ρ_p	0.3, **0.4**, 0.5, 0.6
M	**2**, 3, 4, 5
ℓ	1, **2**, 3, 4

We reformulate the MDP for both methods of modelling returns and use VI to obtain the policies. Similar to Section 5.2, we compare the different methods of modelling returns on the optimality gap, the required computational time, and the RAM usage. The profit is calculated by simulation the resulting policies in the MDP with the multi-period sales-dependent returns to investigate the optimality gap. The results are presented in Table 4.

The advantage of modelling the returns independent from demand (or past sales) is clearly seen in the CPU time. Modelling the returns based on an aggregated number seems to be more influenced by the increased state spaces, but the CPU time still remains low compared to the original MDP. Similar trends can be observed for the RAM usage. It can be concluded that there is a clear trade-off: exact modelling results in a better performing policy at the cost of computational complexity.

From Table 4, it is observed that the sales-aggregated return method shows a low optimality gap, between −3.04% and −0.03%. When online sales become more dominant a larger optimality gap is observed and with lower return percentages the optimality gap decreases. The sales-independent return method shows larger optimality gaps, as high as −60%. Here, the optimality gap increases when the returns are becoming more dominant, such as in scenarios with larger shares of online sales or higher return percentages. Additionally, for both methods the optimality gap increases with higher cost of the product.

Overall, the approach of aggregating the sales still shows relatively good performance, as the optimality gap is lower compared to the policies found by the PPO algorithm. Both methods show good performance and have the potential to be used for large-scale instances. We further elaborate on the performance of the aggregated approach and the PPO algorithm for large-scale instances in the next section.

6. Large-scale instances

In many settings in the literature and in practice, demand volumes, lead times, return windows, and return percentages are larger than in the small-scale settings discussed above. In such settings, the state space of the MDP gets too large to obtain an exact solution. For instance, for the presented large-scale base case, the amount of states would be just over one billion, with a RAM usage of around 16 GB, and an estimated CPU time of 300 hours if solved with VI. Therefore, we use the PPO algorithm with the multi-period sales-dependent returns and the MDP in which the sales are aggregated to solve and analyse large-scale instances. From these large-scale instances relevant managerial insights can be derived as they better reflect current practices. The cost parameters and coefficient of variation of the small-scale instances are also used for the large-scale instances, but parameters that influence the state space of the MDP are increased. In Table 5 the different parameter values for the large-scale instances are presented.

We evaluate the different instances on different performance indicators related to profit and service level. We focus on instances in which demand, return percentages and windows, and lead time are altered as these instances give us insight in how uncertainty influences omni-channel retailer's performance. We investigate the profit and the alpha service level, which is an important performance indicator that tells us the fraction of time one ends a day ends with products still in stock. Most important is the cycle service level (i.e. the alpha service level just before replenishment). Table 6 gives the profit and cycle service level for our large-scale instances. These results were obtained by simulating the policy found by the PPO algorithm and from VI of the sales-aggregated return MDP with the same procedure as described in Section 5.2.

For almost all the small-scale instances presented in Section 5, the policy resulting from the sales-aggregated return MDP showed a lower optimality gap than the policy of the PPO algorithm. Table 6 shows that for large-scale instances, however, the policy of the PPO algorithm has a higher profit than the policy of the sales-aggregated return MDP. The cycle service level is higher for the policy of the PPO algorithm across almost all instances and channels, indicating that more demand is fulfilled.

From Table 6 it is also observed that if the average demand shifts from the offline channel to the online channel, the profit decreases. This is mostly the result of the increase in return cost and lower net sales: if more products are ordered online, the return flow will be higher. Although the profit decreases with more online sales, the policies found differ on the cycle service level. Both policies have higher cycle service level for the offline channel when offline sales decrease. However, where the policy of the PPO algorithm decreases the service level for the online channel with higher online sales, the policy of the sales-aggregated return MDP increases their service level. Overall, the policy of the PPO algorithm has a higher service level across all instances.

The return percentage negatively influences the profit, as more products are returned the retailer has higher return costs. Although the retailer orders less products in instances with higher return percentages, the cycle service level is one of the highest for instances with the highest return percentage. As the retailer has less influence on their inventory position, they might have excessive stock before replenishment arrives. For the instances in which the net online sales remains constant, an opposite trend is observed for the online channel. As the return percentage and online demand increases, the cycle service level decreases except for the offline channel with the policy found by the sales-aggregated return MDP. The profit decreases more in these instances, due to higher return flow, thus the retailer orders less. The retailer prefers to store products in the online channel as the holding cost is lower and make them available later during the sub-period for the offline channel if needed. However, as the return flow is too high, storing them in the online channel becomes less profitable as they might be sold to online customers. Therefore, the retailer orders less products for all channels and will face a stock-out more frequently. For the policy found by the sales-aggregated return MDP this also happens, however the rationing action differs where more products are stored in the offline channel with increasing online sales.

When investigating longer return windows it is observed that the profit is not much influenced. Although the inventory cost is slightly increased, they are negligible indicating that the return window has little effect on the profit for the retailer. However, the return window does influence the cycle service level. When the return window is increased, the cycle service level also increases. When the return window is increased, the probability of a product being returned in a sub-period is decreased. Therefore, the return distribution becomes less uncertain thus the retailer has more control on their inventory levels. Therefore they can better satisfy demand without ending up with excessive or little stock and thus improve their service level without high costs.

With an increase in lead time the profit does not change much, and for the policy of the sales-aggregated return MDP the cycle

Table 6
Profit and cycle service level for large-scale instances.

	Profit		Cycle service level			
			Offline		Online	
Instance	PPO	Agg. R	PPO	Agg. R	PPO	Agg. R
$\mu_1 = 6, \mu_2 = 2$[a]	3130.28	3061.15	0.982	0.969	0.950	0.806
$\mu_1 = 4, \mu_2 = 4$	2662.31	2596.45	0.987	0.986	0.945	0.832
$\mu_1 = 2, \mu_2 = 6$	2226.48	2131.20	0.990	0.986	0.936	0.845
$\rho_p = 0.3$	3241.45	3170.67	0.979	0.970	0.945	0.852
$\rho_p = 0.4$[a]	3130.28	3061.15	0.982	0.969	0.950	0.806
$\rho_p = 0.5$	3033.68	2967.38	0.988	0.970	0.955	0.800
$\rho_p = 0.6$	2934.70	2877.55	0.993	0.974	0.958	0.825
$\rho_p = 0.3, \mu_2 = F(\rho_p)$	3102.50	3081.22	0.978	0.967	0.945	0.833
$\rho_p = 0.4, \mu_2 = F(\rho_p)$[a]	3130.28	3061.15	0.982	0.969	0.950	0.806
$\rho_p = 0.5, \mu_2 = F(\rho_p)$	3081.73	3038.53	0.982	0.973	0.945	0.796
$\rho_p = 0.6, \mu_2 = F(\rho_p)$	3026.83	3004.43	0.977	0.980	0.903	0.812
$\rho_w = 2$[a]	3130.28	3061.15	0.982	0.969	0.950	0.806
$\rho_w = 3$	3134.32	3048.15	0.986	0.968	0.952	0.829
$\rho_w = 4$	3136.88	3042.26	0.968	0.965	0.959	0.850
$\rho_w = 5$	3138.79	3030.63	0.991	0.966	0.964	0.856
$l = 1$	3137.64	3065.18	0.981	0.969	0.953	0.806
$l = 2$[a]	3130.28	3061.15	0.982	0.969	0.950	0.806
$l = 3$	3127.76	3058.87	0.963	0.968	0.946	0.800
$l = 4$	3112.90	3055.85	0.944	0.968	0.907	0.797
Average	3015.22	2948.64	0.979	0.972	0.944	0.823

[a] Base test case with: $\mu_1 = 6$, $\mu_2 = 2$, $CV_i = 1/\sqrt{\mu_i}$, $\rho = 0.4$, $M = 2$, $c_r = 5$, $c_p = 30$, $c_{h1} = 1, c_{h2} = 0.5$, $l = 2$.

service level remains constant. However, for the policy found by the PPO algorithm the cycle service level does decrease. The policy found by the PPO algorithm has more difficulty with the uncertainty of demand during the lead time resulting in higher chance of a stock-out. However, it appears the cycle service level has little effect on the profit.

From Table 6 it can be concluded that the policy of the PPO algorithm outperforms the policy of the sales-aggregated return MDP for large-scale instances on both profit and cycle service level. Furthermore, a higher online demand and return flow negatively influences the profit. The cycle service level of the offline channel increases with higher return percentage, unless the online demand also increases as the retailer compensates for the decrease in profit margins by ordering less products thus negatively influencing the offline channel. However, the retailer can compensate for the lower inventory by preferring the offline channel via the rationing action. Furthermore, a longer return window or lead time has limited influence on the profit but does influence the cycle service level.

7. Discussion and conclusion

Omni-channel retailers are experiencing an increase in returns originating from the growth of online sales. This return flow is becoming a significant issue for their inventory management because the uncertainty of whether a product is being returned can result in excessive stock. Also, several studies suggest to use the store for the fulfilment of online orders and the handling of returns. This has the advantage of leveraging the assets of the offline channel for the online channel.

In this paper, we therefore study the problem of a retailer who replenishes and rations their store inventory across an offline and online channel, as well as integrates returns in managing their inventory. The retailer has to make a trade-off between serving in-store and online customers, where the online sales might lead to returns. This paper contributes to the academic literature by providing a model for the inventory management of an omni-channel retailer with multi-period sales-dependent returns. In contrast to previous work, our model considers the returns dependent on sales over multiple periods. The resulting MDP can be solved with value iteration for small-scale instances resulting in exact solutions. However, the complexity of the model grows for larger demands and longer return windows and becomes unsolvable. Therefore, we investigate alternative solution methods such as DRL and approximating the returns.

As a DRL algorithm, PPO is used, as it has been proven to perform well in inventory management settings. The PPO algorithm is able to provide solutions within reasonable optimality gaps. Although VI shows a much lower running time for small instances, it increases exponential in the problem size. The PPO algorithm shows a high computation time for small instances but scales much better to larger problems. The PPO algorithm leaves an optimality gap of about 2.0% for all small-scale instances. Instances in which there is a higher level of uncertainty negatively influence the training of the PPO algorithm. Furthermore, the PPO algorithm has trouble finding the optimal action when the profit trade-off is minimal between the optimal and non-optimal actions. We conclude that the PPO algorithm is useful in environment with relatively low uncertainty and when the cost differences around the optimal solution are not too small.

When investigating different methods to model and approximate the return flow it was found that modelling the returns independently from historical sales showed a high optimality gap, increasing with larger return flows. Furthermore, aggregating historical sales in one state outperforms the PPO algorithm for small-scale instances (with an optimality gap of around 1.0%). Approximating the returns into one state has the advantage of lower CPU time and RAM usage, and in situations with relatively low return flows, the near-optimal results suggest that the existing models might be useful for implementations in practice.

As the multi-period sales-dependent returns MDP cannot be solved with VI for more realistic retail settings, we use PPO and VI on the MDP formulated with sales-aggregated returns to investigate how return and demand uncertainty influence the omni-channel retailer's profit and service level. For large-scale instances, the PPO algorithm outperforms the policy found with VI of the sales-aggregated return MDP, therefore the PPO algorithm is pre-

ferred for finding a policy when faced with larger demand volumes. The results indicate that if customers shift from the offline to the online channel it has a negative impact on the profit for the retailer. As online demand increases, the online returns increase which negatively influence the retailer profit as it requires more handling. If the online returns are becoming too large it will also negatively influence the service level of the offline channel, as the retailer will order less products for both channels due to the low profitability per product sold online unless the retailer compensate it with their rationing strategy. Furthermore, although a longer return window does not affect the profit per product sold, it does increase the service level of both channels, therefore a longer return window might be more preferable as long as it does not result in higher return percentages.

This paper investigates how returns influence the profitability of an omni-channel retailer who uses their store for the fulfilment of online demand and returns of online products. Although the model captures most typical characteristics of omni-channel retailers, several potentially relevant factors were not taken into account, and could be directions for further research. First, we only investigate the perspective of a single store. In a multi-store context, additional research could also include which stores are allowed to fulfil online demand and handle online returns. Second, further development of DRL algorithms for inventory problems is relevant. Although the context of our studied problem does not require discounting of future rewards, the PPO algorithm needs it for stable training and convergence. Having the discount factor too close to one results in unstable training, therefore an interesting research direction would be in the development of stable undiscounted DRL algorithms. The PPO algorithm develops a well-performing policy in our study, it has difficulties learning in some instances. Several research directions can be identified to improve the training of the PPO algorithm. Transfer learning (in which the policy found of one instance is used for the learning of similar instances) might be useful in reducing training time. Also, reward shaping might be used to provide better feedback on which actions are preferred. Furthermore, behaviour cloning, in which the PPO algorithm is pre-trained with an expert policy such as a simple heuristic, might be useful. Further reductions in CPU time could also be achieved by parallelisation of the sampling of training data.

Appendix A. Loss function of PPO

Below we describe the procedure of calculating the loss for each set of samples in a mini-batch. The loss is used to update the parameters of the actor and critic neural network. The set of samples in a mini-batch for level I is denoted by $\mathcal{K}_1$ and for level II by $\mathcal{K}_2$, where $\mathcal{K}_1 = \{0, 1, \ldots, B/\eta\}$ and $\mathcal{K}_2 = \{0, 1, \ldots, T \cdot B/\eta\}$. A sample point at level I is denoted by K and at level II by k.

A1. Advantage and discounted profit

The advantage and discounted profit are calculated for each sample in the mini-batch. The advantage is an indication of how well the chosen action performs compared with the expected cost. We use this information to update our neural networks. In this paper, we use the generalised advantage estimator approach, as described in Vanvuchelen et al. (2020). The advantage of sample K of level I is calculated as follows:

$$G_K = U_K - V(S_K, \phi_1) \tag{A.1}$$

Here, G_K is the advantage of sample K of level I and U_K is the discounted profit for sample K (defined below). The advantage of sample k of level II is calculated as follows:

$$g_k = u_k - V(s_k, \phi_2) \tag{A.2}$$

Here, g_k is the advantage of sample k of level II and u_k is the discounted profit for sample k (defined below). The discounted profit of sample K of level I is calculated as follow:

$$U_K = \sum_{i=K}^{B} \gamma^{i-K} \cdot E_i + \gamma^{B-K} \cdot V(S_B, \phi_1) \tag{A.3}$$

The discounted profit is an estimation of the future profit the retailer can expect to receive at it current state. This estimation is derived from the profit obtained through the sampling, and the value of the actor of the last state. The discount factor (γ) is needed to ensure the training of the PPO algorithm is stable and leads to convergence (Wiering, 2004). The discounted profit of sample k of level II is calculated similarly:

$$u_k = \sum_{i=k}^{T \cdot B} \gamma^{i-k} \cdot e_i + \gamma^{T \cdot B - k} \cdot V(s_{T \cdot B}, \phi_2) \tag{A.4}$$

A2. Training the neural networks

With the collected training data, advantage, and discounted profit of the sampling, the gradient of the loss function with respect to the weights and bias of the neural networks can be calculated. The loss function is a predictor for the error of the neural network. By updating the weights and bias in the direction of the gradient of the loss with a step size, the loss can be minimised.

To train the neural network, three different type of loss functions are used, two for the actor network and one for the critic network. The loss functions of the actor network consist of the policy loss and an entropy loss. The policy loss trains the actor neural network so that actions that give high expected profit are preferred above actions with lower expected profits, while the entropy loss tries to encourage exploration of new actions. The policy loss for a shuffled mini-batch of level I is defined as follows:

$$\text{Policy loss level1} = -\sum_{K \in \mathcal{K}_1} \min\left(\frac{\pi\left(q_K | S_K, \theta_1'\right)}{\pi\left(q_K | S_K, \theta_1\right)} \cdot G_K, \right.$$
$$\left. \text{clip}\left(\frac{\pi\left(q_K | S_K, \theta_1'\right)}{\pi\left(q_K | S_K, \theta_1\right)}, 1-\epsilon, 1+\epsilon \right) \cdot G_K \right) \tag{A.5}$$

Here, $\frac{\pi\left(q_K | S_K, \theta_1'\right)}{\pi\left(q_K | S_K, \theta_1\right)}$ is the ratio of the probability of choosing ordering action q_K in state S_K with the new neural network parameters θ_1' and the current parameters of the neural network θ_1. The policy loss formula uses a clipping formula to limit the loss function, where $\text{clip}(x, x_{\min}, x_{\max})$ ensures that x is between the range of $x_{\min}$ and $x_{\max}$, otherwise the value is clipped to the range edges. The clipping parameter ϵ determines the value of the range edges.

The policy loss for a shuffled mini-batch of level II is defined as follows:

$$\text{Policy loss level II} = -\sum_{k \in \mathcal{K}_2} \min\left(\frac{\pi\left(a_k | s_k, \theta_2'\right)}{\pi\left(a_k | s_k, \theta_2\right)} \cdot g_k, \right.$$
$$\left. \text{clip}\left(\frac{\pi\left(a_k | s_k, \theta_2'\right)}{\pi\left(a_k | s_k, \theta_2\right)}, 1-\epsilon, 1+\epsilon \right) \cdot g_k \right) \tag{A.6}$$

The entropy loss for a shuffled mini-batch of level I is defined as follows:

$$\text{Entropy loss level I} = \beta_E \sum_{K \in \mathcal{K}_1} \pi\left(\cdot | S_K, \theta_1'\right) \cdot \log \pi\left(\cdot | S_K, \theta_1'\right) \tag{A.7}$$

If $\pi\left(\cdot | S_K, \theta_1'\right)$ is evenly distributed, thus each action has the same probability, the entropy will be largely negative. A deterministic policy, where one action has a high probability compared to others, results in the entropy loss to be closer to zero. The entropy

regularisation term β_E determines how much emphasis is placed on the entropy in the loss when combined with the policy loss. By minimising the entropy, the probability distribution of actions is more evenly distributed. During sampling this encourages the algorithm to explore new actions, preventing the policy from converging to a bad-performing local optimal. The entropy loss for a shuffled mini-batch of level II is defined as follows:

$$\text{Entropy loss level II} = \beta_E \sum_{k \in \mathcal{K}_2} \pi\left(\cdot|s_k, \theta_2'\right) \cdot \log \pi\left(\cdot|s_k, \theta_2'\right) \tag{A.8}$$

For the critic network a value loss function is used, which is based on the difference between the future discounted profit and the output of the value function approximation. Here the Huber loss is used as it is less sensitive towards outliers as opposed to the mean squared error loss (Huber, 1964). Due to potentially large demand fluctuations, such outliers in profit are quite likely to ocur in our problem setting, and the use of the Huber loss therefore is less influenced by these outliers. The Huber loss is defined as follows:

$$L(x) = \begin{cases} \frac{1}{2}x^2 & \text{for } |x| \leq \delta \\ \delta\left(|x| - \frac{1}{2}\delta\right) & \text{otherwise.} \end{cases} \tag{A.9}$$

Where δ is the Huber loss constant. The value loss for a shuffled mini-batch of level I is defined as follows:

$$\text{Value loss level I} = \sum_{K \in \mathcal{K}_1} L\left(V\left(S_K, \phi_1'\right) - U_K\right) \tag{A.10}$$

The value loss for a shuffled mini-batch of level II is defined as follows:

$$\text{Value loss level II} = \sum_{k \in \mathcal{K}_2} L\left(V\left(s_k, \phi_2'\right) - u_k\right) \tag{A.11}$$

From the three different losses the total average loss of a mini-batch is calculated as follow:

$$\text{Average loss level I} = \frac{\text{Value loss level I} + \text{Policy loss level I} + \text{Entropy loss level I}}{B/\eta} \tag{A.12}$$

$$\text{Average loss level II} = \frac{\text{Value loss level II} + \text{Policy loss level II} + \text{Entropy loss level II}}{T \cdot B/\eta} \tag{A.13}$$

Minimising the average loss via updating θ' and ϕ' will result in the neural networks fitting to the given training data. The neural network parameters are iteratively updated using a stochastic gradient descent with the ADAM optimiser, as this optimiser is less sensitive to parameter tuning and is overall considered to be the current best practice (Kingma & Ba, 2014). The optimiser uses the average loss of minibatch j for actor and critic network of level h to calculate the gradient of the weights and bias with respect to the average loss and perform an update step with a learning rate of α. A low learning rate results in slow convergence to a good policy, while a high learning rate might results in overshooting a good policy.

Appendix B. Optimality gap

The optimality gap of the PPO algorithm for different small-scale instances is included in Table B1.

Table B1
Optimality gap of the PPO algorithm for different instances.

	Profit			Revenue	Costs			
Instance	MDP	PPO	Gap (%)	Gap (%)	Inventory gap (%)	Ordering gap (%)	Fulfillment gap (%)	Return gap (%)
$\mu_1 = 3, \mu_2 = 1$[a]	1575.15	1549.76	−1.64	0.17	21.60	0.18	−0.02	0.04
$\mu_1 = 2, \mu_2 = 1$	1111.99	1088.10	−2.20	0.07	28.03	0.13	−0.20	−0.29
$\mu_1 = 1, \mu_2 = 1$	653.91	636.55	−2.73	0.37	28.07	0.30	0.53	0.62
$CV_i = 1/\sqrt{\mu_i}$[a]	1575.15	1549.76	−1.64	0.17	21.60	0.18	−0.02	0.04
$CV_1 = 1/(2\sqrt{3})$	1616.12	1588.86	−1.72	0.02	24.80	0.01	0.04	0.05
$CV_2 = 1/(2\sqrt{1})$	1583.14	1554.56	−1.84	−0.07	22.68	0.03	−1.00	−1.00
$CV_i = 1/(2\sqrt{\mu_i})$	1626.19	1604.78	−1.33	−0.02	20.72	−0.02	0.06	0.06
$\rho_p = 0.2$	1676.88	1642.04	−2.12	−0.06	26.41	0.01	−1.22	−1.40
$\rho_p = 0.3$	1626.03	1589.67	−2.29	−0.19	22.69	−0.13	−0.83	−0.93
$\rho_p = 0.4$[a]	1575.15	1549.76	−1.64	0.17	21.60	0.18	−0.02	0.04
$\rho_p = 0.5$	1526.03	1493.15	−2.20	0.03	26.53	0.05	−0.12	−0.12
$\rho_p = 0.6$	1476.48	1447.05	−2.03	0.25	23.45	0.12	0.96	0.98
$\rho_p = 0.2, \mu_2 = F(\rho_p)$	1590.26	1555.65	−2.22	0.07	26.79	0.14	−1.42	−1.52
$\rho_p = 0.3, \mu_2 = F(\rho_p)$	1587.48	1557.07	−1.95	−0.02	23.34	0.04	−0.78	−0.78
$\rho_p = 0.4, \mu_2 = F(\rho_p)$[a]	1575.15	1549.76	−1.64	0.17	21.60	0.18	−0.02	0.04
$\rho_p = 0.5, \mu_2 = F(\rho_p)$	1558.63	1528.31	−1.98	0.17	23.23	0.08	0.62	0.68
$\rho_p = 0.6, \mu_2 = F(\rho_p)$	1540.07	1500.50	−2.64	0.97	32.12	0.66	2.24	2.21
$M = 1$	1576.78	1539.90	−2.40	0.08	26.83	0.07	0.32	0.18
$M = 2$[a]	1575.15	1549.76	−1.64	0.17	21.60	0.18	−0.02	0.04
$c_r = 0$	1588.93	1556.06	−2.11	0.10	24.74	0.08	0.28	0
$c_r = 5$[a]	1575.15	1549.76	−1.64	0.17	21.60	0.18	−0.02	0.04
$c_r = 10$	1561.88	1524.68	−2.44	0.57	31.97	0.53	0.81	0.90
$c_p = 30$[a]	1575.15	1549.76	−1.64	0.17	21.60	0.18	−0.02	0.04
$c_p = 50$	1078.26	1056.67	−2.04	0.88	24.97	0.82	1.47	1.44
$c_p = 70$	582.58	577.41	−0.89	3.67	22.76	3.06	9.30	9.41
$c_{h1} = 0.1, c_{h2} = 0.05$	1680.98	1667.44	−0.81	−0.02	55.71	0.05	−0.56	−0.61
$c_{h1} = 0.2, c_{h2} = 0.1$	1558.63	1528.31	−1.98	0.17	23.23	0.08	0.62	0.68
$c_{h1} = 1, c_{h2} = 0.5$[a]	1575.15	1549.76	−1.64	0.17	21.60	0.18	−0.02	0.04
$l = 1$	1576.78	1539.90	−2.40	0.08	26.83	0.07	0.32	0.18
$l = 2$[a]	1575.15	1549.76	−1.64	0.17	21.60	0.18	−0.02	0.04
$l = 3$	1573.36	1537.44	−2.34	0.40	29.45	0.42	0.25	0.23
Average	1464.53	1432.82	−2.27	−0.44	23.09	−0.38	−1.09	−1.07

[a] Base test case with: $\mu_1 = 3, \mu_2 = 1, CV_i = 1/\sqrt{\mu_i}, \rho_p = 0.4, M = 2, c_r = 5, c_p = 30, c_{h1} = 1, c_{h2} = 0.5, l = 2$.

Table C1
Weighted NRMSE of actions by PPO.

	Weighted NRMSE							
	Ordering	Rationing						
Instance		Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Average
$\mu_1 = 3, \mu_2 = 1$[a]	0.08	0.29	0.21	5.99	0.44	0.52	0.53	0.31	1.18
$\mu_1 = 2, \mu_2 = 1$	0.11	0.29	0.20	2.40	0.60	0.99	0.75	0.41	0.80
$\mu_1 = 1, \mu_2 = 1$	0.20	0.29	0.30	3.53	1.51	0.91	0.59	0.55	1.09
$CV_i = 1/\sqrt{\mu_i}$[a]	0.08	0.29	0.21	5.99	0.44	0.52	0.53	0.31	1.18
$CV_1 = 1/(2\sqrt{3})$	0.09	0.61	0.41	7.05	6.26	0.48	0.64	1.02	2.35
$CV_2 = 1/(2\sqrt{1})$	0.07	0.51	0.21	4.30	1.80	0.54	0.82	0.59	1.25
$CV_i = 1/(2\sqrt{\mu_i})$	0.11	0.51	0.33	3.69	2.33	2.90	0.33	0.46	1.51
$\rho_p = 0.2$	0.10	0.39	0.29	6.97	3.39	1.21	1.16	0.60	2.00
$\rho_p = 0.3$	0.09	0.23	0.19	4.02	2.79	0.61	0.65	0.41	1.27
$\rho_p = 0.4$[a]	0.08	0.29	0.21	5.99	0.44	0.52	0.53	0.31	1.18
$\rho_p = 0.5$	0.00	0.25	0.18	6.21	0.72	0.76	0.58	0.63	1.33
$\rho_p = 0.6$	0.00	0.35	0.26	3.92	0.72	0.69	0.45	0.40	0.97
$\rho_p = 0.2, \mu_2 = F(\rho_p)$	0.09	0.42	0.30	6.49	2.35	0.62	0.84	0.46	1.64
$\rho_p = 0.3, \mu_2 = F(\rho_p)$	0.15	0.23	0.13	4.57	3.07	0.72	0.78	0.44	1.42
$\rho_p = 0.4, \mu_2 = F(\rho_p)$[a]	0.08	0.29	0.21	5.99	0.44	0.52	0.53	0.31	1.18
$\rho_p = 0.5, \mu_2 = F(\rho_p)$	0.14	0.35	0.35	4.26	0.50	0.38	0.31	0.23	0.91
$\rho_p = 0.6, \mu_2 = F(\rho_p)$	0.14	0.17	0.24	9.88	0.94	0.72	0.50	0.16	1.80
$M = 1$	0.18	0.30	0.23	7.77	2.22	0.71	0.79	0.69	0.00
$M = 2$[a]	0.08	0.29	0.21	5.99	0.44	0.52	0.53	0.31	1.185
$c_r = 0$	0.00	0.21	0.21	5.93	3.47	0.68	0.94	0.49	1.70
$c_r = 5$[a]	0.08	0.29	0.21	5.99	0.44	0.52	0.53	0.31	1.185
$c_r = 10$	0.11	0.27	0.21	7.06	0.58	0.83	1.02	0.64	1.52
$c_p = 30$[a]	0.08	0.29	0.21	5.99	0.44	0.52	0.53	0.31	1.185
$c_p = 50$	0.10	0.18	0.21	3.28	0.48	0.69	0.50	0.43	0.82
$c_p = 70$	0.12	0.15	0.13	2.52	0.35	0.28	0.18	0.29	0.56
$c_{h1} = 0.1, c_{h2} = 0.05$	0.22	0.46	0.53	4.46	0.67	1.07	1.51	1.04	1.39
$c_{h1} = 0.2, c_{h2} = 0.1$	0.31	0.68	0.22	20.33	1.37	1.37	1.38	1.09	3.78
$c_{h1} = 1, c_{h2} = 0.5$[a]	0.08	0.29	0.21	5.99	0.44	0.52	0.53	0.31	1.185
$l = 1$	0.00	0.43	6.20	7.10	1.10	0.72	0.35	0.40	2.33
$l = 2$[a]	0.08	0.29	0.21	5.99	0.44	0.52	0.53	0.31	1.185
$l = 3$	0.00	0.30	0.23	7.77	2.22	0.71	0.79	0.69	1.82
Average of all instances	1.33	0.54	0.54	6.23	2.42	1.07	0.91	0.76	1.71

[a] Base test case with: $\mu_1 = 3, \mu_2 = 1, CV_i = 1/\sqrt{\mu_i}, \rho_p = 0.4, M = 2, c_r = 5, c_p = 30, c_{h1} = 1, c_{h2} = 0.5, l = 2$.

Appendix C. Goodness of fit

To evaluate how much the policy found by the PPO algorithm resembles the optimal policy, the weighted Normalised Root Mean Square Error (NRMSE) is used. The NRMSE is a common metric for comparison of DRL algorithms (e.g. Chi et al., 2010; Rocchetta et al., 2019; Xie et al., 2019). We simulate the heuristics for J periods where the set of periods is given by $\mathcal{J} = \{0, 1, \ldots, J\}$ and calculate the weighted NRMSE as follows:

$$\text{Weighted NRMSE} = \frac{\sqrt{\frac{1}{J}\sum_{j\in\mathcal{J}}\left(\pi_j^* - \pi_j\right)^2}}{\max_{j\in\mathcal{J}}(\pi_j^*) - \min_{j\in\mathcal{J}}(\pi_j^*)} \tag{C.1}$$

Here, π_j^* is the optimal action to be taken in simulation period j and π_j is the action chosen by the PPO algorithm policy found in the same period. Normalisation of the RMSE is performed by dividing the metric with the maximum and minimum value of the optimal policy, for cases where $\max_{j\in\mathcal{J}}(\pi_j^*) - \min_{j\in\mathcal{J}}(\pi_j^*) = 0$ the term is set to 1. For the ordering action the simulation period $J = 1.000.000$ periods and for the rationing action $J = T \cdot 1.000.000$ sub-periods.

References

Abdulla, H., Ketzenberg, M., & Abbey, J. D. (2019). Taking stock of consumer returns: A review and classification of the literature. *Journal of Operations Management, 65*(6), 560–605.

Adan, I., van Eenige, M., & Resing, J. (1995). Fitting discrete distributions on the first two moments. *Probability in the Engineering and Informational Sciences, 9*(4), 623–632.

Agatz, N. A., Fleischmann, M., & Van Nunen, J. A. (2008). E-fulfillment and multi-channel distribution—Areview. *European Journal of Operational Research, 187*(2), 339–356.

Ambilkar, P., Dohale, V., Gunasekaran, A., & Bilolikar, V. (2022). Product returns management: A comprehensive review and future research agenda. *International Journal of Production Research, 60*(12), 3920–3944.

Bayram, A., & Cesaret, B. (2021). Order fulfillment policies for ship-from-store implementation in omni-channel retailing. *European Journal of Operational Research, 294*(3), 987–1002.

Benedito, E., & Corominas, A. (2013). Optimal manufacturing policy in a reverse logistic system with dependent stochastic returns and limited capacities. *International Journal of Production Research, 51*(1), 189–201.

Bernon, M., Cullen, J., & Gorst, J. (2016). Online retail returns management: Integration within an omni-channel distribution context. *International Journal of Physical Distribution and Logistics Management, 46*(6/7), 584–605.

Boute, R. N., Gijsbrechts, J., van Jaarsveld, W., & Vanvuchelen, N. (2022). Deep reinforcement learning for inventory control: A roadmap. *European Journal of Operational Research, 298*(2), 401–412.

de Brito, M. P., & Dekker, R. (2003). Modelling product returns in inventory control—exploring the validity of general assumptions. *International Journal of Production Economics, 81*, 225–241.

Chen, J., & Bell, P. C. (2012). Implementing market segmentation using full-refund and no-refund customer returns policies in a dual-channel supply chain structure. *International Journal of Production Economics, 136*(1), 56–66.

Chi, C.-Y., Tsai, R. T.-H., Lai, J.-Y., & Hsu, J. Y.-j. (2010). A reinforcement learning approach to emotion-based automatic playlist generation. In *2010 international conference on technologies and applications of artificial intelligence* (pp. 60–65). IEEE.

Cohen, A. C. (1954). Estimation of the poisson parameter from truncated samples and from censored samples. *Journal of the American Statistical Association, 49*(265), 158–168.

DeCroix, G. A. (2006). Optimal policy for a multiechelon inventory system with remanufacturing. *Operations Research, 54*(3), 532–543.
Dijkstra, A. S., Van der Heide, G., & Roodbergen, K. J. (2019). Transshipments of cross-channel returned products. *International Journal of Production Economics, 209*, 70–77.
ENC (2016). What every omni-channel retailer should know about developing a ship from store strategy. enVista Company.
Feinberg, E. A., & Lewis, M. E. (2005). Optimality of four-threshold policies in inventory systems with customer returns and borrowing/storage options. *Probability in the Engineering and Informational Sciences, 19*(1), 45–71.
Fleischmann, M., & Kuik, R. (2003). On optimal inventory control with independent stochastic item returns. *European Journal of Operational Research, 151*(1), 25–37.
Fleischmann, M., Kuik, R., & Dekker, R. (2002). Controlling inventories with stochastic item returns: A basic model. *European Journal of Operational Research, 138*(1), 63–75.
Fleischmann, M., & Minner, S. (2004). Inventory management in closed loop supply chains. In *Supply chain management and reverse logistics* (pp. 115–138). Springer.
Gao, F., Agrawal, V. V., & Cui, S. (2022). The effect of multichannel and omnichannel retailing on physical stores. *Management Science, 68*(2), 809–826.
Gijsbrechts, J., Boute, R. N., Van Mieghem, J. A., & Zhang, D. (2022). Can deep reinforcement learning improve inventory management? Performance on dual sourcing, lost sales and multi-echelon problems. *Manufacturing and Service Operations Management, 24*(3), 1349–1368.
Goedhart, J., Haijema, R., & Akkerman, R. (2022). Inventory rationing and replenishment for an omni-channel retailer. *Computers & Operations Research, 140*, 105647.
Hobkirk, I. (2015). Key distribution strategies of top omni-channel retailers. *Logistics Management, 55*(1), 48–53.
Hu, X., Wan, Z., & Murthy, N. N. (2019). Dynamic pricing of limited inventories with product returns. *Manufacturing and Service Operations Management, 21*(3), 501–518.
Huang, S., & Ontañón, S. (2020). A closer look at invalid action masking in policy gradient algorithms. arXiv preprint arXiv:2006.14171
Huber, P. J. (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics, 35*(1), 73–101.
Hübner, A., Hense, J., & Dethlefs, C. (2022). The revival of retail stores via omnichannel operations: A literature review and research framework. *European Journal of Operational Research, 302*(3), 799–818.
Hübner, A., Holzapfel, A., & Kuhn, H. (2015). Operations management in multi-channel retailing: An exploratory study. *Operations Management Research, 8*(3–4), 84–100.
Hübner, A., Holzapfel, A., & Kuhn, H. (2016). Distribution systems in omni-channel retailing. *Business Research, 9*(2), 255–296.
Jalilipour Alishah, E., Moinzadeh, K., & Zhou, Y.-P. (2015). Inventory fulfillment strategies for an omni-channel retailer. *Working paper*. University of Washington, Seattle.
Janakiraman, N., Syrdal, H. A., & Freling, R. (2016). The effect of return policy leniency on consumer purchase and return decisions: A meta-analytic review. *Journal of Retailing, 92*(2), 226–235.
Jin, D., Caliskan-Demirag, O., Chen, F. Y., & Huang, M. (2020). Omnichannel retailers' return policy strategies in the presence of competition. *International Journal of Production Economics, 225*, 107595.
Johnson, N. L., Kemp, A. W., & Kotz, S. (2005). *Univariate discrete distributions. Third edition*. John Wiley & Sons.
Ketzenberg, M. E., Abbey, J. D., Heim, G. R., & Kumar, S. (2020). Assessing customer return behaviors through data analytics. *Journal of Operations Management, 66*(6), 622–645.
Kiesmüller, G. P., & Van der Laan, E. A. (2001). An inventory model with dependent product demands and returns. *International Journal of Production Economics, 72*(1), 73–87.
Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
Kristensen, A. R. (1988). Hierarchic Markov processes and their applications in replacement models. *European Journal of Operational Research, 35*(2), 207–215.
LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade* (pp. 9–48). Springer.
de Leeuw, S., Minguela-Rata, B., Sabet, E., Boter, J., & Sigurðardóttir, R. (2016). Trade-offs in managing commercial consumer returns for online apparel retail. *International Journal of Operations & Production Management, 36*(6), 710–731.
Li, T., Zhao, X., & Xie, J. (2015). Inventory management for dual sales channels with inventory-level-dependent demand. *Journal of the Operational Research Society, 66*(3), 488–499.
Louchard, G., & Ward, M. D. (2015). The truncated geometric election algorithm: Duration of the election. *Statistics & Probability Letters, 101*, 40–48.
Mandal, P., Basu, P., & Saha, K. (2021). Forays into omnichannel: An online retailer's strategies for managing product returns. *European Journal of Operational Research, 292*(2), 633–651.
Mena, C., Bourlakis, M., Ishfaq, R., Defee, C. C., Gibson, B. J., & Raja, U. (2016). Realignment of the physical distribution process in omni-channel fulfillment. *International Journal of Physical Distribution & Logistics Management, 46*(6/7), 543–561.
Mou, S., Robb, D. J., & DeHoratius, N. (2018). Retail store operations: Literature review and research directions. *European Journal of Operational Research, 265*(2), 399–422.
Muir, W. A., Griffis, S. E., & Whipple, J. M. (2019). A simulation model of multi-echelon retail inventory with cross-channel product returns. *Journal of Business Logistics, 40*(4), 322–338.
Nageswaran, L., Cho, S.-H., & Scheller-Wolf, A. (2020). Consumer return policies in omnichannel operations. *Management Science, 66*(12), 5558–5575.
Ovezmyradov, B., & Kurata, H. (2019). Effects of customer response to fashion product stockout on holding costs, order sizes, and profitability in omnichannel retailing. *International Transactions in Operational Research, 26*(1), 200–222.
Puterman, M. L. (1994). *Markov decision processes*. John Wiley & Sons.
Radhi, M., & Zhang, G. (2019). Optimal cross-channel return policy in dual-channel retailing systems. *International Journal of Production Economics, 210*, 184–198.
Rocchetta, R., Bellani, L., Compare, M., Zio, E., & Patelli, E. (2019). A reinforcement learning framework for optimal operation and maintenance of power grids. *Applied Energy, 241*, 291–301.
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347
Shah, S. (1966). On estimating the parameter of a doubly truncated binomial distribution. *Journal of the American Statistical Association, 61*(313), 259–263.
Tarn, J. M., Razi, M. A., Wen, H. J., & Perez, A. A. (2003). E-fulfillment: The strategy and operational requirements. *Logistics Information Management, 16*(15), 350–362.
Vanvuchelen, N., Gijsbrechts, J., & Boute, R. (2020). Use of proximal policy optimization for the joint replenishment problem. *Computers in Industry, 119*, 103239.
Verhoef, P. C., Kannan, P. K., & Inman, J. J. (2015). From multi-channel retailing to omni-channel retailing: Introduction to the special issue on multi-channel retailing. *Journal of Retailing, 91*(2), 174–181.
Wiering, M. (2004). Convergence and divergence in standard and averaging reinforcement learning. In *Machine learning: ECML 2004: vol. 3201* (pp. 477–488). Springer Berlin Heidelberg.
Wollenburg, J., Holzapfel, A., Hübner, A., & Kuhn, H. (2018). Configuring retail fulfillment processes for omni-channel customer steering. *International Journal of Electronic Commerce, 22*(4), 540–575.
Xie, T., Ma, Y., & Wang, Y.-X. (2019). Towards optimal off-policy evaluation for reinforcement learning with marginalized importance sampling. arXiv preprint arXiv:1906.03393
Xu, X., & Jackson, J. E. (2019). Investigating the influential factors of return channel loyalty in omni-channel retailing. *International Journal of Production Economics, 216*, 118–132.
Zerhouni, H., Gayon, J.-P., & Frein, Y. (2013). Influence of dependency between demands and returns in a reverse logistics system. *International Journal of Production Economics, 143*(1), 62–71.