On Computational Procedures for Optimising an Omni-Channel Inventory Control Model

Joost Goedhart[1] and Eligius M. T. Hendrix[2(✉)]

[1] Operations Research and Logistics, Wageningen University, 6706 KN Wageningen, The Netherlands
joost.goedhart@wur.nl
[2] Computer Architecture, Universidad de Málaga, 29071 Málaga, Spain
eligius@uma.es

Abstract. Dynamic programming (DP) and specifically Markov Decision Problems (MDP) are often seen in inventory control as a theoretical path towards optimal policies, which are (often) not tractable due to the curse of dimensionality. A careful bounding of decision and state space and use of resources may provide the optimal policy for realistic instances despite the dimensionality of the problem. We will illustrate this process for an omni-channel inventory control model where the first dimension problem is to keep track of the outstanding ordered quantities and the second dimension is to keep track of items sold online that can be returned.

Keywords: Inventory control · Markov decision problems · Stochastic processes · Value iteration · Omni-channel retailing

1 Introduction

Inventory control is a dynamic process. Therefore, dynamic programming has been considered an appropriate technique to derive so-called optimal order policies, see [6]. An illustration of how to implement Value Iteration (VI) is given in [4] for small one and two-dimensional Markov Decision Problem (MDP) cases from perishable inventory control to derive a stationary order policy. One of the early detected challenges is the introduction of lead-time in inventory models, as one should keep track of the pipeline of already ordered quantities [7]. The larger the lead time, the larger the state space which makes the problem intractable. We will illustrate the resulting challenge for a so-called omni-channel retailer model, where a retailer uses multiple channels (online and in-store) to fulfil consumer demand [3]. In such inventory models, inventory is hold to fulfil both in-store demand as well as online orders. A rationing decision is involved that allocates inventory to either online or in-store consumers. As the rationing decision is

This paper has been supported by The Spanish Ministry (RTI2018-095993-B-I00) in part financed by the European Regional Development Fund (ERDF).

O. Gervasi et al. (Eds.): ICCSA 2022 Workshops, LNCS 13378, pp. 29–42, 2022.
https://doi.org/10.1007/978-3-031-10562-3_3

dependent on the inventory level and outstanding orders, the complexity of the action is related to the state space dimension. Additionally, in practice online sales go along with return streams. Modelling this aspect implies an increase in the state space, as we have to keep track of the items sold online which have not been returned yet. The state space increases with the number of days we keep track of unreturned items, as products might be returned the next day or after a longer period.

Our research question is how to implement the corresponding dynamic programming approach and to find the limits of the models that can be solved to optimality within a reasonable computing time and memory usage. We demarcate our question to the situation of an omni-channel retailer confronted with a discrete finite demand for both channels, lost sales and inventory holding cost, ordering cost, fulfilment cost, and a profit margin for both channels.

To investigate the question, we formulate a dynamic model and illustrate its solution procedure via VI. We focus on the computational effort to find the optimal solution and on demarcation of the state space. This paper is organised as follows. We first sketch the concept of deriving the optimal policy by VI in Sect. 2. Section 3 then introduces and investigates several versions of the omni-channel model. Section 4 summarises our findings.

2 The Procedure of VI

To consider an inventory control problem as a Markov Decision Problem (MDP), we should define the state space, decision space, transition probabilities and contributions. For inventory control, this requires to have a good vision on the sequence of events. In our omni-channel case, at the end of the day, after receiving (or not) an outstanding order, the retailer decides on the next order quantity Q and the rationing R. This decision depends on the inventory state I and outstanding order states q symbolised by state vector S; a stationary policy is described by $(Q(S), R(S))$. The rationing R is given as the number of items assigned to the sales in the shop and exposed there, thus R items are stored in-store and $I - R$ items are stored in the backroom from where they can be sold online. The challenge we focus on is that the order quantity is received after a lead time of L periods, requiring to put the outstanding orders in the state space of S, as a vector $q = (q_1, .., q_\ell, \ldots, q_L)$. The inventory dynamics is following the equation

$$I_{new} = (R - d_1)^+ + (I - R - d_2)^+ + q_L, \tag{1}$$

with $x^+ := \max\{x, 0\}$, where d_1 and d_2 represent the realisation of demand in the shop and of demand online respectively. The dynamics of the pipeline inventory is

$$q_\ell = q_{\ell-1}, \ell = 2, \ldots, L \tag{2}$$

and $q_1 = Q$. In more abstract terms, a transformation from one state to the other is given by a function T, transforming the state $S = T(I, q, Q, R, d_1, d_2)$ depending on current state, decisions and realisation of demand. Notice that we

implicitly think in time steps of one day instead of in terms of continuous time, which is also reasonable for a retailer situation. Moreover, we think of the state space $\mathcal{S}$ as finite and discrete by bounding the inventory by a maximum $\overline{I}$ and also the order quantity by $\overline{Q}$. This means that one can map the state space in an array s, where an element s_j represents a state value for all state variables.

For the optimisation of the inventory control, we have a cost function $C(Q, R, I)$ which depends on the procurement cost and inventory holding cost. Moreover, there is an expected gain $G(I, R)$ of the current state (I, R). This defines the expected profit where the margins for selling online and from the shop differ.

2.1 MDP Background

Under certain circumstances, optimal inventory control can be characterised by the MDP theory, introduced originally by [1]. For the optimal policy (Q, R), there exists a so-called value function $v(S)$ and a scalar π to be interpreted as the expected daily profit such that

$$\forall S \in \mathcal{S}, v(S) + \pi = \max_{Q,R} \left(G(I, R) - C(I, Q, R) + E(v(T(S, Q, R, \mathbf{d}_1, \mathbf{d}_2))) \right),$$

where E is the expectation over the stochastic demand $\mathbf{d}_1$ and $\mathbf{d}_2$, which we take as independent events. Working with a discrete finite distribution $p_{1k} = P(\mathbf{d}_1 = k)$ and $p_{2m} = P(\mathbf{d}_2 = m)$, we can see the expected valuation of the future in a discrete way, see [5]. Consider all states in an array $s := (s_0, s_1, \ldots, s_{N-1})$ and define the expected revenue as

$$r(s_i, Q, R) := G(s_i, R) - C(s_i, Q, R),$$

then the Bellman equation comes down to the existence of an array V and constant π such that

$$\forall i,\ V_i + \pi = \max_{Q,R} \left(r(s_i, Q, R) + \textstyle\sum_{k,m} p_{1k} p_{2m} V_j \right),$$

with V_j interpreted as the value of $v(T(s_i, Q, R, d_{1k}, d_{2m}))$. If we now map the optimal control rule $(Q(S), R(S))$ in an array, then the best thing to do in situation s_i is given by

$$(Q_i, R_i) = \operatorname*{argmax}_{Q,R} \left(r(s_i, Q, R) + \textstyle\sum_{k,m} p_{1k} p_{2m} V_j \right), \tag{3}$$

with $V_j = v(T(s_i, Q, R, d_{1k}, d_{2m}))$. In theory, the equations imply a fixed point relation around the expected gain π. How to derive the optimal policy now in a computational way? How are we going to be bothered by the curse of dimensionality?

2.2 Value Iteration Procedure

Following the fixed point idea with respect to value π, we will follow a procedure called Value Iteration (VI). From a computational point of view it is sufficient

Algorithm 1. Pseudocode of VI for inventory control

1: Set array elements V_i to $r(s_i, 0, 0)$ for $i = 0, \ldots, N-1$
2: **repeat**
3: Copy vector V into vector W
4: **for** $i = 0, \ldots, N-1$ **do**
5: **for** Feasible values of Q, R **do**
6: **for** all demand realisations d_{1k}, d_{2m} **do**
7: Get W_j with $s_j = T(s_i, Q, R, d_{1k}, d_{2m})$
8: $V_j = \max_{Q,R}[r(s_i, Q, R) + \sum_{k,m} p_{1k} p_{2m} W_j]$
9: **until** $\max_j(V_j - W_j) - \min_j(V_j - W_j) < \varepsilon$

to think in copying the last valuation in an array W and to determine the new valuation array V as sketched in Algorithm 1 up to convergence takes place of what is called the span, i.e. the gap between maximum and minimum difference of the two arrays. The convergence is slow when state values that have a low probability of occurrence are included. The challenge is, that this is known in the end, by simulating the optimal policy or alternatively deriving the stationary state probabilities of the Markov Chain. In practice, this leads to ideas of adaptive Dynamic Programming, when we adapt the state space depending on the outcomes of the optimisation.

A large practical computational gain is to compute the expected revenue $r(s_i, Q, R)$ to be calculated beforehand as sketched in Fig. 1, so not to run over all demand realisations during the VI loop. Another important concept is to define, before the iteration process the transition matrix. Instead of doing a loop over possible outcomes of demand as sketched by lines 6 and 7 in Algorithm 1, we construct a matrix $M(Q, R)$, where entrances $M_{ij}(Q, R)$ capture the probability to go from state s_i to state s_j when performing action (Q, R). This means, we can replace part of the evaluation of lines $6-8$ in Algorithm 1 by a matrix-vector multiplication multiplying M with array W calling compiled code.

3 Cases

We consider an omni-channel retailer case who sells in both channels for a margin of $m = 45$, where online shipping adds an additional 10 cost per unit. Order cost is $k = 33$ and storing in the shop requires holding cost of $h_1 = 1$ and the backroom $h_2 = 0.5$ per night. The margin and costs define the expected revenue $r(S, Q, R)$. The demand is assumed to be Poisson distributed, but truncated above at the 0.999 quantile in order to have a bounded space on demand. In our base case, we take a mean demand for an item in-store of $\mu_1 = 6$ and online of $\mu_2 = 2$ per day. Based on the probability mass distribution of demand, we can construct a transition matrix which also depends on the rationing decision R following dynamics (1). The order quantity decision Q affects the dynamics of the order pipeline (2), which does not depend on random events.

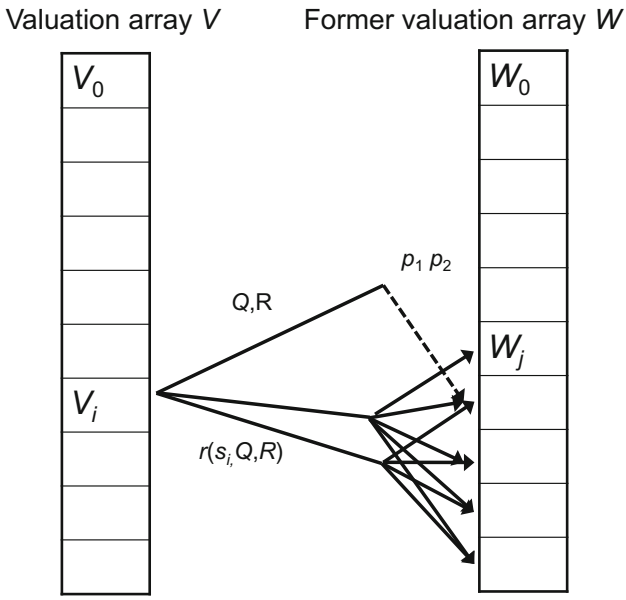

Fig. 1. Data dependence in value iteration

One of the first questions to ask ourselves when implementing a VI is how to bound the state and decision space. In the described model, the maximum inventory $\overline{I}$ does not deviate too much from the maximum order quantity $\overline{Q}$. For the latter, the so-called Economic Order Quantity (EOQ) (see e.g. [6]) is relevant for minimising order cost. In our case, we have to take into account that the rationing to the more expensive channel will be limited up to what is profitable, so the holding costs mainly depend on the back room holding cost h_2. Following general lines, we come to an EOQ = 32, which coincides with a replenishment cycle of 4 d. However, demand is stochastic, so a safety stock applies weighting the lost sales of the total demand during the cycle and the expected inventory cost. Therefore, we estimate using the EOQ model and a safety stock estimation that $\overline{I} = \overline{Q} = 45$, is a reasonable upper bound of the state space with respect to the inventory level. Without such an analysis, one depends on a numerical procedure and observe which values will be attained by following the derived optimal policy. We should keep in mind that including state values that have a low probability of occurrence, will not only increase the solution time in a polynomial way, but will also increase the number of iterations necessary for convergence. In our first experiment, we will illustrate this feature. Notice that the rationing decision is always bounded by $R \leq I$.

3.1 First Case, Lead Time $L = 1$

The complexity should be low, if the pipeline of outstanding orders caused by the lead time is low. In our case of a lead time of $L = 1$, one can derive that in fact it is sufficient to capture the state variable S only by inventory I, as $q_l = Q$ can be handled by optimising simultaneously over order quantity Q and rationing R. This means that the ordered quantity does not appear as state variable, but can directly be included in the state transition of inventory level I at the end of the day.

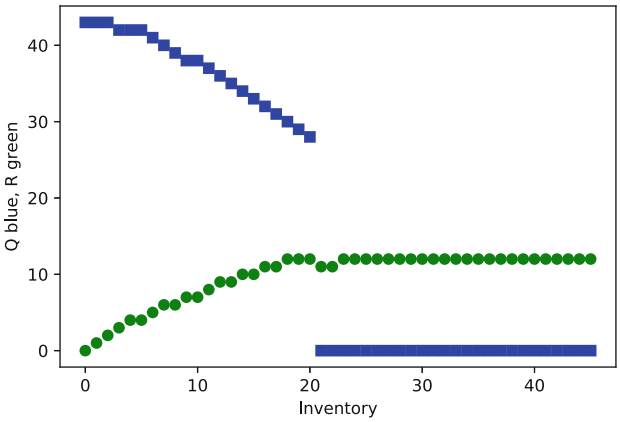

Fig. 2. Optimal order quantity $Q(I)$ and rationing $R(I)$ as function of inventory

The code we developed in python, first calculates the revenue r and a transition matrix $M(Q, R)$ to exploit vector-matrix multiplications and computing the maximum (best decision) using libraries. Following the sketched procedure with a termination criterion of $\varepsilon = 0.1$, the VI converges after 36 iterations and 2 s providing the optimal policies as depicted in Fig. 2. Observing the decisions, we can see that the rationing R is bounded by balancing expected lost sales versus the inventory holding costs and limited to $\overline{R} = 12$; see proof in [3]. The optimal order quantity Q is higher than the EOQ, but prevents putting an order from inventory levels higher than about $I = 20$. For both decisions, one could exploit monotonicity considerations; one can observe that if $Q(I) = 0$, then $Q(I+1) = 0$ and if $R(I) = \overline{R}$, then $R(I+1) = \overline{R}$. We did not make use of such considerations, as we are following matrix multiplication for all potential decisions rather than a do-loop.

We simulated the dynamics of the model for $T = 500,000$ days providing a daily profit of 310. We did an additional experiment varying the size of the state space using $\overline{I} = 50, 100, 150, 200$. In order to inspect the state space, we measured the occurrence of each inventory level as a frequency as depicted in Fig. 3. From the VI follows that the maximum amount to be ordered is $\overline{Q} = 42$.

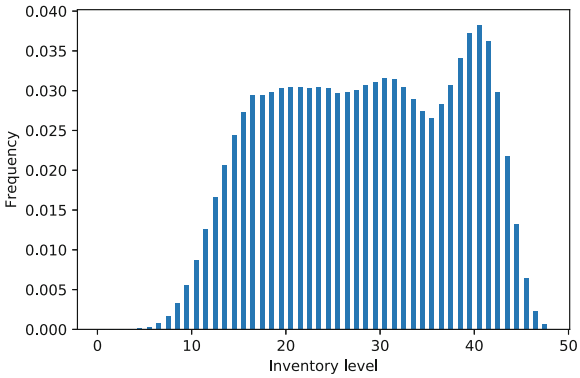

Fig. 3. Frequency of occurrence of inventory level following the optimal decisions

This means that basically high values of the inventory level do not occur, as it does not make sense to order each day a high amount. This is illustrated in Fig. 3. As we measure the inventory level after the order has been delivered, there is no surprise that around $I = 40$, we have a high probability of occurrence. We are more surprised by the equal occurrence of values in the range $I \in [18, 32]$.

Table 1. Computational time varying $\overline{I}$

Upper bound $\overline{I}$	50	100	150	200
Time seconds	3	11	30	61

Varying state limit value $\overline{I} = 50, 100, 150, 200$, one would expect a computational time linear increase in this case. As shown in Table 1, this is not the case. This illustrates, that a strategy could be to start with state boundaries that are too small, observe and possibly simulate the system, and extend the boundaries gradually rather than starting with a space which is too big requiring an enormous computational time. Notice that limits can also be taken lower exploiting extrapolation of the value function. In our example, high values of inventory lead to an optimal policy of not to order, i.e. $Q = 0$. This means that the procedure requires a valuation of high inventory levels, but does not have to evaluate what is the optimal policy for those cases. For the valuation, extrapolation can be used.

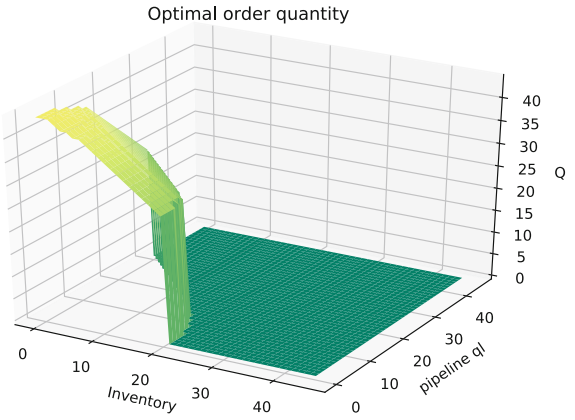

Fig. 4. Optimal order quantity $Q(I, q_2)$ as function of the (pipeline) inventory

3.2 Extending Lead Time $L = 2, 3$

With a longer lead time, we have to take the pipeline inventory into account which has been ordered L days before. For the optimal order quantity, this has quite some impact as sketched in Fig. 4. The rationing decision is less sensitive to the pipeline inventory. Only when inventory levels are low, the pipeline of outstanding orders may influence the ration decision. This is sketched in Fig. 5; only for very low inventory levels the ration is influenced by the order quantity two days ago.

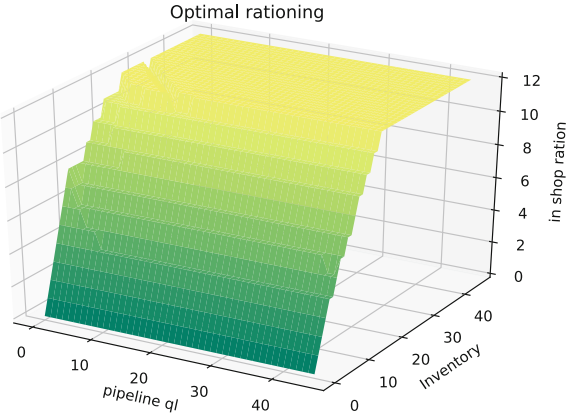

Fig. 5. Optimal ration $R(I, q_2)$ as function of the (pipeline) inventory

To obtain the optimal decision rule, the number of state values has increased to 2,116 and the VI requires 35 iterations to convergence in 65 s in our implementation. The increase in computational time compared to the case $L = 1$ is less than proportional to the number of state values.

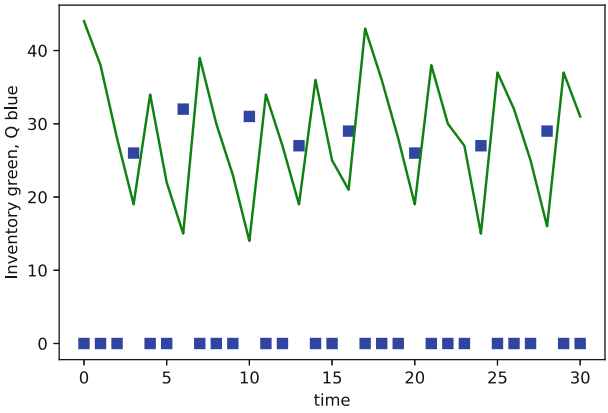

Fig. 6. Inventory development and order quantities, $L = 2$

A simulation first shows us the inventory development and order quantities for the first weeks in Fig. 6. Notice that the pipeline inventory follows the order quantity with a delay. The average daily profit of the optimal rule is 308. This is only slightly less than the situation with a lead time of 1 period (using the same pseudo random demand data).

On one hand, the lead time may provide less predictability of demand and imply higher cost. On the other hand, if we compare the distribution of inventory of a lead time of 2 in Fig. 7 with that of lead time of 1 in Fig. 3, one can observe that the physical inventory is lower in the first case, as part is pipeline inventory for which no inventory cost is incurred.

The state space when extending the lead time is $\overline{Q} = 45$ times bigger leading to a computing time which appears 30 times bigger, which is less than linear, although required memory is linearly increasing. The efficiency gain is due to pre-processing the transition matrix which includes the rationing decision and using library routines to compute the maximum profit rationing value. Notice that the transition of the pipeline inventory is straightforward.

We pushed the implementation to the largest state space we consider by having a lead time $L = 3$ providing 97,336 state values. The computational time increases nearly proportional to the increase in the state space converging in 3330 s seconds after 34 iterations. The outcome of the procedure shows that due to the longer lead-time, the inventory is increased considerably, see Fig. 9. As illustrated in Fig. 8, this also has some impact on the VI procedure. It appears that the orders are going to follow a cyclical behaviour. In that case, also the

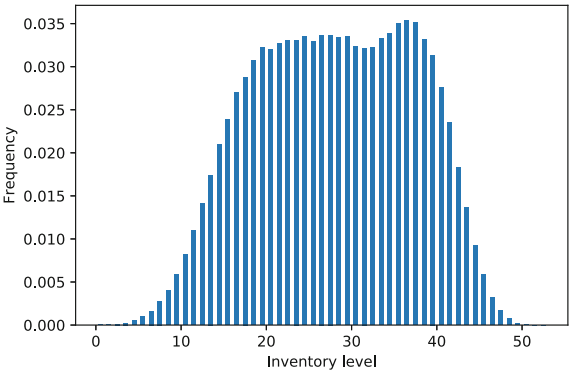

Fig. 7. Frequency inventory level, $L = 2$

convergence should be checked in another way, due to alternating states. This illustrates, that it is of utmost importance to check the results by a simulation.

3.3 Including Online Sales in the State Space

The recent decade, there has been an increased interest in the effect of returning sold online products on the logistics decision, see e.g. [2]. Our question is here, what is the effect on the state space and increasing computational time of including recent sales in the state space? To know the possible returned products, we have to keep track of the number of items sold over the last n days, where n represents the horizon in which customers may return their bought product.

In terms of the profit, the bookkeeping is getting more complicated, but also helps to decide not to sell the product online at all. On a return, we get the margin of the item back, but we loose not only the earlier mentioned shipping costs of 10 units, but also a handling cost of 5 units. This reduces the expected profit for online items drastically. Assume that we have a probability of $p = 40\%$ that an item is returned. The online sales has a profit margin of 35 if the customer keeps the item (60% probability). This should be weighted with the loss of 15 when the product is returned (40% in our example).

For the inventory dynamics, if we consider a fixed probability of returning bought items during the return period, the number of returned items can be modelled using a binomial distribution. The challenge is that this requires to keep track of the number of items bought j periods ago, $(r_1, \ldots, r_j, \ldots, r_n)$. For our exercise, we will consider $n = 1, 2, 3$ and therefore we also have to adapt the probability p_j that a product is returned which has been sold j days ago. We will inspect the computational consequence of adding this state variable to the VI algorithm for $n = 1, 2, 3$. The dynamics of the sold items for $n = 3$ is $r_3 = r_2 - bin(r_2, p_2)$, $r_2 = r_1 - bin(r_1, p_1)$ and $r_1 = \min(I - R, d_2)$, which should be included in the transition matrix computation. For our exercise, we can bound the state space thanks to the truncated demand to $\overline{r}_j = 7$, as that covers

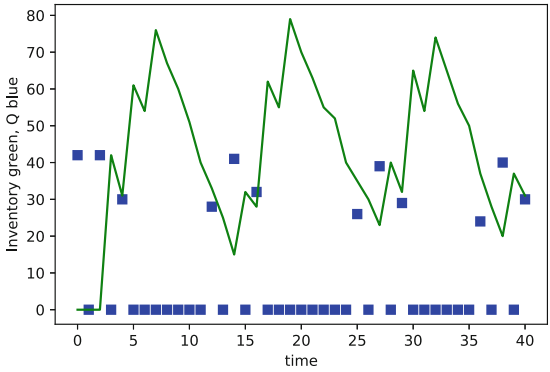

Fig. 8. Inventory development and order quantities, $L = 3$

practically 0.999 of the demand probability for d_2. This implies an increase of the state space with a factor $(\overline{r}_j + 1)^n$.

More cumbersome is now the inventory dynamics. To the dynamics of I_{new} in (1) should now be added the binomial probability distribution of (r_1, p_1), (r_2, p_2) and (r_3, p_3). Of course, this is just one way to model the return of bought products. We implemented the transition matrix, first for a case where $n = 1$ to compare the computational time and the results to that of Case 1 taking all parameters equal. The computational time for our implementation is 8 s compared to the 2 s if no returning of sold products is allowed. However, the state space is 8 times bigger. Convergence requires 36 iterations within 8 s. Again, the time efficiency is reached due to the pre-computation of expected gain and the transition matrix, where not only the probability of demand, but also that of returns is captured to reach state j from state i with decision R. As described, the transition due to the order quantity has a more deterministic character.

The optimal policy now also depends on the number of products sold one day ago. Actually, it anticipates to order less given that there are returns expected of sold products. The behaviour is sketched in Fig. 10. Due to the probability of returns of sold products, the expected profit reduces from 310 to 271 units.

Now extending the state space taking $n = 2$ days as the possibility to return the online bought product, increases the number of state values again with a factor 8. The other parameters are left constant. The algorithm now requires 35 iterations and 102 s. Our next case considered a return time of 3 d, which increases the state space by 8 again. The computational time now increases to 1030 seconds and the profit of the derived policy is basically the same to a two day return period.

The extension of the model allows us now to consider lead time increase and returns from earlier sales. We run a variant for $L = 2$ and $n = 1$ with in total $N = 16,928$ which converged after 35 iterations taking 800 s of computational time. The computation of the transition matrix entails now considering both lead

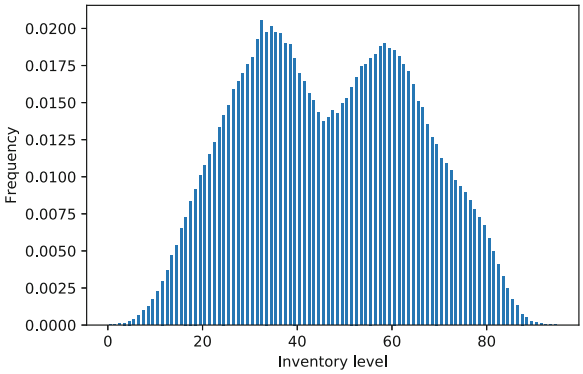

Fig. 9. Frequency inventory level, L=3

time and the probabilities of having returns. The computational time compared to the case of $L = 2$ and $n = 0$ increased more than proportional in the number of state values.

3.4 Summary of the Numerical Results

Table 2. Results of Algorithm 1 on the cases

L	n	N	Tcomp	Profit
1	0	46	2	309
2	0	2,116	65	308
3	0	97,336	3,330	301
1	1	368	8	271
1	2	2,944	102	266
1	3	23,552	1,030	266
2	1	16,928	800	268

The VI algorithm was implemented in python exploiting the NumPy library routines. For an experiment running the cases on a desktop Intel(R) i5-2400 CPU with 8 GB ram, the computational time is reported in Table 2. Remember that all cases use the same cost parameter values and the same (pseudo-) random numbers for validating the expected profit of the resulting policy. The number of state values is represented by N, L is the lead time, n the number of days a customer can return an online bought product, Tcomp is the computational time in seconds and profit the expected daily profit estimated by the simulation.

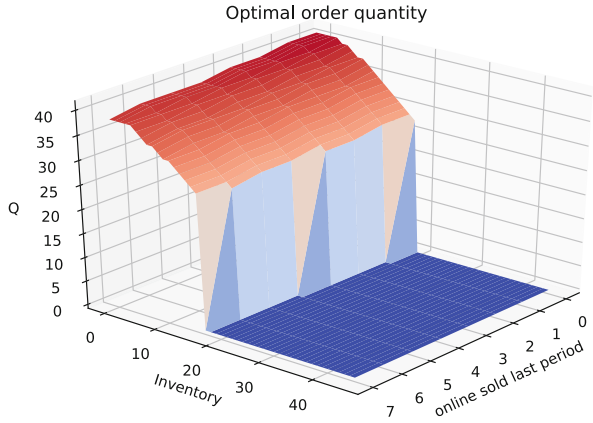

Fig. 10. Optimal order quantity, $L = 1, n = 1$

4 Conclusions

The determination of optimal inventory management policies for higher dimensional state spaces, is often considered not feasible due to an increasing complexity. The computational feasibility of using Value Iteration (VI) mainly depends on a careful bounding of the space using information on the maximum demand and order quantities. In general, the relevant state space to be considered follows from simulating the optimal policy. This means that a strategy to start with smaller space boundaries and gradually increasing them based on the simulation outcomes may help to define the smallest relevant space. In our case, we could analytically estimate the relevant maximum inventory due to revenue considerations on the optimal order quantity following the EOQ model and a safety stock estimation. This has been illustrated in the paper for several cases. The maximum sold items to bound the returns space is mainly determined by the demand of in-shop sales, although it also depends on the rationing decision. Moreover, we found that in python based implementations, one should try to make use of compiled library procedures to replace explicit do-loops.

In our illustration, we have shown that pipeline inventory can be taken into account up to a limited lead-time. An extension towards including past sales to anticipate on returned sold products allows a wider extension due to a strict bounding on the demand levels, leaving out low probability state values which hinder convergence of the VI algorithm.

References

1. Bellman, R.: A Markovian decision process. J. Math. Mech. **6**(5), 679–684 (1957)
2. Bernon, M., Cullen, J., Gorst, J.: Online retail returns management: integration within an omni-channel distribution context. Int. J. Phys. Dis. Logistics Manag. **64**(6), 584–605 (2016)

3. Goedhart, J., Haijema, R., Akkerman, R.: Inventory rationing and replenishment for an omni-channel retailer. Comput. Oper. Res. **140**, 105647 (2022)
4. Hendrix, E.M.T., Kortenhorst, C., Ortega, G.L.: On computational procedures for value iteration in inventory control. IFAC-PapersOnLine **52**(13), 1484–1489 (2019)
5. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st edn. John Wiley & Sons Inc., New York (1994)
6. Silver, E., Pyke, D., Peterson, R.: Inventory Management and Production Planning and Scheduling. Wiley (1998)
7. Zipkin, P.: Old and new methods for lost-sales inventory systems. Oper. Res. **56**(5), 1256–1263 (2008)