An aggregation-based approximate dynamic programming approach for the periodic review model with random yield

Michael A. Voelkel
Department of Supply Chain Management and Management Science, University of Cologne, D-50923 Cologne, Germany

Anna-Lena Sachs*
Department of Management Science, Lancaster University, Lancaster, United Kingdom

Ulrich W. Thonemann
Department of Supply Chain Management and Management Science, University of Cologne, D-50923 Cologne, Germany

A manufacturer places orders periodically for products that are shipped from a supplier. During transit, orders get damaged with some probability, that is, the order is subject to random yield. The manufacturer has the option to track orders to receive information on damages and to potentially place additional orders. Without tracking, the manufacturer identifies potential damages after the order has arrived. With tracking, the manufacturer is informed about the damage when it occurs and can respond to this information. We model the problem as a dynamic program with stochastic demand, tracking cost, and random yield. For small problem sizes, we provide an adjusted value iteration algorithm that finds the optimal solution. For moderate problem sizes, we propose a novel aggregation-based approximate dynamic programming (ADP) algorithm and provide solutions for instances for which it is not possible to obtain optimal solutions. For large problem sizes, we develop a heuristic that takes tracking costs into account. In a computational study, we analyze the performance of our approaches. We observe that our ADP algorithm achieves savings of up to 16% compared to existing heuristics. Our heuristic outperforms existing ones by up to 8.1%. We show that dynamic tracking reduces costs compared to tracking always or never and identify savings of up to 3.2%.

Key words: Inventory, Approximate dynamic programming, Random yield, Tracking, Value of information

---

* Corresponding author: a.sachs@lancaster.ac.uk

1. Introduction

Many companies operate large sourcing and distribution networks. They operate production facilities in different countries, receive orders from global suppliers, and serve customers globally. Such companies must take transportation risks into account when determining order or production quantities. A European pharmaceutical company that we work with, for example, sources drugs from Asia. The drugs are delivered via sea freight with a transportation time of about four weeks. The drugs must be kept within a certain temperature range during the entire journey. If the temperature leaves the range during transport, the drugs must be discarded. If the ordering process is not managed properly and does not take such supply risks into account, it is difficult to control inventory efficiently.

Temperature risks are not the only supply risks that companies face. Other risks include product handling, packaging, air flow within container, and many more (Ketzenberg et al. 2018, Sleptchenko and Johnson 2015, White and Cheong 2012).

For modeling purposes, the risks are often aggregated into yield rates, which are the ratios of usable items to items ordered, and which can be substantially below 100%. For perishable food, they range between $70\% - 80\%$ (Dobbs et al. 2011, Gustavsson et al. 2011) and for vaccines, they are about 75% (White and Cheong 2012). Random yield is also encountered in production. Samsung's curved glass production for cell phones, for example, has a yield rate of less than 50% (Sonntag and Kiesmüller 2017) and semiconductor device production processes exhibit yield rates of $50\% - 70\%$ (Gavirneni 2004). We focus on analyzing random yield in supply chains, but the approaches can also be applied to analyzing random yield in production environments.

We analyze a setting where the yield of a product is estimated using information such as time and temperature history (TTH) that can be measured and recorded throughout the voyage. For example, in shelf life models, the measured and recorded information on TTH is used to update an estimate of the expected remaining life time of a product (e.g., Gaukler et al. 2017, Ketzenberg et al. 2018). Then, a decision can be made up to which remaining lifetime a product might be sold. A similar approach would be to consider a product unsalable if the TTH shows that a predefined acceptable range has been left at least once during the voyage.

We assume in our analysis that acceptable ranges for measured characteristics are given for each product because they are either defined by the company or induced by regulatory constraints. This means that the company or (inter)national legislation specifies a policy according to which perishable products such as food, medical products, etc. are salable. These policy decisions are based on estimates at which point a product has spoiled and has to be discarded. As soon as the company finds out that a measured characteristic leaves the acceptable range, the products are considered as unsalable and the company may place a new order to satisfy demand.

If standard containers are used, the history of the measurements is accessed upon arrival of an order. Alternatively, the company may decide to use new technology that allows companies to track characteristics of an order during shipment. Smart containers provide information on measurements of temperature, humidity etc. on a regular basis or in near-real time. For example, DHL offers a service called Ocean Secure that guarantees near-real time tracking of conditions and locations of products (DHL 2013). We consider a periodic review model where companies can decide in each period whether they want to pay for the tracking service to track the order, that is, to receive time and temperature information.

If orders are untracked, the history can be observed at arrival and the salability is determined based on this history. If orders are tracked, companies can evaluate the salability earlier. They can then issue replenishment orders immediately if the order is considered unsalable. Thus, order tracking allows companies to manage their inventory more efficiently. Previous research has shown that additional information about uncertain parameters can significantly improve order decisions (e.g., Choi et al. 2008, Ketzenberg et al. 2007, 2006).

In this paper, our objective is to provide a model that allows companies to quantify the monetary savings that can be achieved by either tracking all, tracking only selected orders or not tracking at all. Companies can then use this information to weight these savings against the cost for using tracking services. This comparison requires a good understanding of all the costs involved when using and investing in a new technology. However, we do not focus on the analysis of new technology investments, but on providing a model for determining when orders should be tracked and which cost savings can potentially be achieved.

Our contribution is threefold: Firstly, we model the problem as a periodic review model with dynamic tracking and backorders under random yield and derive new structural properties of the model. Secondly, we develop three new solution approaches for the problem: We use dynamic programming to solve this problem to optimality for small problem instances. To overcome the curse of dimensionality, we propose a novel approximate dynamic programming approach that uses multiple levels of aggregation for moderate problem instances. For large problem instances, we develop a new heuristic that takes tracking decisions into account. Thirdly, we quantify the benefits of tracking random yield.

2. Literature review

Two streams of literature are relevant for our reseach, the literature on random yield and solution approaches for Markov decision processes.

An extensive overview of random yield models is provided by Yano and Lee (1995). They discuss the models that have been provided for various kinds of random yield and different kinds of supply chains. One of the earliest research on inventory models with random yield can be attributed to Karlin (1958). He considers a single period inventory system with binary order decisions, where the order yield is a random variable with a known distribution. Henig and Gerchak (1990) derive structural properties and show that there exists an upper bound for the order decision that depends on the current inventory position. Only few multi-stage models with positive lead times exist (Yano and Lee 1995). Related to our model are the models by Choi et al. (2008) and Dettenbach and Thonemann (2015). Choi et al. (2008) consider a finite horizon model with a lead time of three periods and real-time yield information. They state that simple order-up-to policies do not deliver optimal results and provide a heuristic solution which illustrates that sharing information is beneficial. Dettenbach and Thonemann (2015) consider a model with arbitrary lead times, where the yield of all or none of the orders is tracked. Tracking is often accomplished via RFID. For a literature review on RFID, we refer to Choi et al. (2008), Ngai et al. (2008) and Sarac et al. (2010). The application of RFID leads to shared information and can be utilized to generate substantial benefits (e.g., Gaukler et al. 2007, Ketzenberg et al. 2015, Lee and Özer 2007).

We extend the literature on inventory models with random yield by providing the option to track orders dynamically. Existing models assume that all or none of the orders are tracked and either solve simplified models optimally or realistic models heuristically. We consider a model in which all, none or some of the orders are tracked. We design our model as a Markov decision process. Dettenbach (2015) models this problem similarly, but within his solution approach, he considers random yield for a single fixed lead time period and assumes that demand is deterministic. We allow for random yield and stochastic demand in any lead time period, which makes classical solution approaches computationally intractable. We derive new structural properties for this problem and provide novel solution approaches that are optimal for small to moderate problem sizes and that solve large problem sizes better than existing heuristics.

One area of solution approaches for our model comprises myopic policies that optimize decisions based on costs that are a direct consequence of the decisions. They do not memorize costs or decisions per state and therefore do not learn over time. In the inventory management literature with random yield, linear inflation policies, a sub-class of myopic policies, are common (Huh and Nagarajan 2010). If the inventory position falls below a certain threshold, an order is triggered. The order size is an inflation factor multiplied with the difference between the inventory position and the threshold (e.g., Zipkin 2000, p. 393). These heuristics have proven to perform better than other heuristics for a range of problems similar to the one that we consider (e.g., Bollapragada and Morton 1999, Inderfurth and Kiesmüller 2015, Kiesmüller and Inderfurth 2018, Li et al. 2008). Ehrhardt and Taube (1987) provide one of the first linear inflation rule policies for a single period problem with random yield. They focus on the average level of replenishment but ignore its variability.

Bollapragada and Morton (1999) develop multi-period heuristics that are based on newsvendor solutions. Huh and Nagarajan (2010) show how the order threshold can be optimized for a given inflation factor. Dettenbach and Thonemann (2015) provide heuristic solution approaches that are adaptations of the heuristics by Ehrhardt and Taube (1987) and Huh and Nagarajan (2010). We contribute to this area by developing a heuristic that not only decides on the order size but also on the tracking decision depending on the order size. In contrast, while linear inflation policies are easy to apply, they are not optimal due to their myopic nature (Inderfurth and Kiesmüller 2015).

Solution approaches for Markov decision processes that obtain optimal or close-to-optimal solutions are commonly based on dynamic programming or approximate dynamic programming. Both approaches consider multiple lead time periods and are therefore not myopic. Dynamic programming yields optimal results but suffers from the curse of dimensionality, and is therefore only applicable to small problem instances. Approximate dynamic programming mainly comprises value function approximations, which are considered as the most powerful method for solving complex dynamic programs (Powell 2011, p. 235). Within the realm of value function approximations, there are different subcategories to consider. A common distinction can be made between parametric and non-parametric representations of a state and lookup tables (Powell 2011, ch. 6). Parametric models need to estimate a value function by designing a set of features that represent the value function accurately. Linear parametric models are most popular (Powell 2016). For example, Kleywegt et al. (1998) use a parametric approximation of the value function for inventory routing problems. However, the state variables of our model (inventory level and open orders) have non-linear effects on our cost function. Within non-parametric models, neural networks have received a lot of attention (Hastie et al. 2009, pp. 347-369). They are able to approximate functions of arbitrary shape. For example, Van Roy et al. (1997) apply a multi-layer neural network to two-echelon retailer inventory systems. In our problem, we found that neural networks only delivered good estimates after learning from millions of exact cost values. Further, the learning could not be transferred from one parameter setting to another. Due to that, neural networks are not suitable for solving the practical problem that we consider.

Lookup tables exhibit the greatest level of detail because they do not use feature-based functional approximations. However, they suffer from the curse of dimensionality if not used with aggregation. Lookup tables can be combined with approximate value iteration which is described in general in Bertsekas and Tsitsiklis (1996) and Powell (2011). Singh et al. (1995) discuss soft-state aggregation where states are assigned to multiple clusters with certain probabilities. Lambert et al. (2004) provide so-called macro states, which can overlap and include individual states. Actions in these macro states are not limited to macro states, but also include actions in individual states. Bertsekas and Castañon (1989) introduce an adaptive method that changes the level of aggregation during the course of the algorithm. They change the membership of a state to particular groups adaptively based on cost changes but their method still suffers from the curse of dimensionality. George et al. (2008) propose a method where multiple aggregation levels are used for which weights are solved optimally. A detailed overview of approximate dynamic programming approaches, including hierarchical ones, is given by Gosavi (2009).

We contribute to the approximate dynamic programming literature by developing a novel hierarchical state aggregation on multiple levels. Our approach assigns each state to exactly one aggregation level. Unlike existing multi-level aggregation approaches, we explore more granular levels of aggregation only for a subset of states based on information gained on a coarse aggregation level. States that are ultimately assigned to a coarse aggregation level are not explored on a more granular aggregation level, which significantly reduces the number of states.

3. Model formulation

We next formulate our model and derive important characteristics. In Section 3.1, we describe the general setting and the sequence of events. In Section 3.2, we formulate our problem as a Markov decision process and devise central theorems about this process. In Section 3.3, we derive equivalence for certain classes of states. In Section 3.4, we develop bounds for costs and optimal order quantities.

3.1. Setting

We consider a single manufacturer who places orders with a single supplier. The demand per period, D, of the product is stochastic and i.i.d. across periods. We denote the quantity of an order placed t time periods ago by O_t. Orders arrive after a deterministic lead time of λ periods. In each lead time period, orders are subject to stochastically proportional random yield. The yield rate of lead time period t ($t=1,\dots,\lambda$) is Y_t with expected value u_t. Order O_λ placed λ periods ago experiences λ random yields and we denote the number of items that arrive after these lead time periods as Q_λ. The yield rates Y_t are i.i.d. over time and can be arbitrarily distributed. This yield model is frequently used to analyze the random yield inventory problem (Choi et al. 2008, Ehrhardt and Taube 1987, Gerchak et al. 1988).

For each order, we must decide whether to track. If an order is not tracked, we receive yield information after the order has been delivered, that is, λ periods after it has been placed. When placing an order, we must decide on the order quantity and whether or not to track. While we might decide to track orders in some periods, we might decide not to track in others. In settings where it is optimal to track some but not all orders, there is no simple policy of when to decide to track. The trade-off between additional information and paying tracking cost has to be made for each state individually.

Our problem is a discrete-time stochastic control problem that employs a state space $\mathscr{S}$. We define the current state of the inventory system $x \in \mathscr{S}$ as $(I, O_1, \ldots, O_\lambda, \psi_1, \ldots, \psi_\lambda)$. A state consists of the inventory level I, the open orders O_t and the tracking decisions $\psi_t \in \{0, 1\}$, where positive values for I denote the on-hand inventory and negative values describe the backorders. The interpretation of O_t depends on ψ_t. If $\psi_t = 1$, the order is tracked and O_t denotes the number of usable items in the order that have passed through t lead time periods with random yield. If $\psi_t = 0$, the order is not tracked and O_t denotes the number of items ordered. The number of items that arrive in the current period after λ lead time periods, that is, Q_λ, can then be determined as follows: If $\psi_\lambda = 1$, O_λ denotes the number of usable items, so that $Q_\lambda = O_\lambda$. If $\psi_\lambda = 0$, O_λ denotes the number of ordered items and the number of items that arrive is $Q_\lambda = O_\lambda \prod_{t=1}^{\lambda} Y_t$ with expected value $\mathbb{E}[Q_\lambda] = O_\lambda \prod_{t=1}^{\lambda} u_t$. All notation is summarized in Appendix EC.1.

The sequence of events is as follows: At the beginning of a period, the manufacturer observes the current state of the inventory system. She decides on the number of items to order and whether or not to track this order at fixed tracking cost of τ. The order is shipped and arrives after a lead time of λ periods. Then, the order placed λ periods before the current period arrives and is stored in inventory. Demand occurs and potential backorders from previous periods as well as the demand of the current period are filled. Finally, backorder or inventory holding costs are charged.

3.2. Markov decision process formulation

Given state x, we denote the space of possible actions as $\mathscr{A}_x$. The action $a \in \mathscr{A}_x$ consists of the order quantity and tracking decision which we henceforth denote as $\hat{O}$ and $\hat{\psi}$. The expected cost for one lead time period, $c(x, a)$, consists of holding and penalty costs as well as tracking cost if we decide to track. We write

$$c(x, a) = c_{inv}(x) + \tau\hat{\psi}, \tag{1}$$

where $c_{inv}(x) = \mathbb{E}\Big[h[I - D + Q_\lambda]^+ - p[I - D + Q_\lambda]^-\Big]$ is the inventory cost and h and p are the inventory holding and penalty cost factors, respectively. The objective is to minimize expected cost for the infinite horizon model. This expected cost is the result of the minimization of the one-stage cost, $c(x, a)$, and the costs of all possible transition states of x that we denote as $\tilde{x}$.

The possible transition states $\tilde{x} \in \mathscr{S}$ are defined as $\tilde{x} = (\tilde{I}, \tilde{O}_1, \ldots, \tilde{O}_\lambda, \tilde{\psi}_1, \ldots, \tilde{\psi}_\lambda)$. The inventory level of the transition state is $\tilde{I} = I - D + Q_\lambda$. The open order $\tilde{O}_1$ in the transition state is computed as $\tilde{O}_1 = \mathbb{I}[\hat{\psi} = 1](Y_1\hat{O}) + \mathbb{I}[\hat{\psi} = 0](\hat{O})$, where $\mathbb{I}$ is the indicator function that is one if we track the order and zero otherwise. The first part of the term holds if we track the order that we place and receive yield information; the second term holds if we do not track the order that we place and do not receive yield information. Similarly, we compute the open orders in the transition state for orders placed $i \geq 2$ periods ago as $\tilde{O}_i = \mathbb{I}[\psi_{i-1} = 1](Y_i O_{i-1}) + \mathbb{I}[\psi_{i-1} = 0](O_{i-1})$. The tracking decisions are carried over from the current state to the transition state, that is $\tilde{\psi}_1 = \hat{\psi}$ and $\tilde{\psi}_i = \psi_{i-1}$ for $\lambda \geq i \geq 2$.

The costs of the transition states are discounted by a factor $0 < \gamma < 1$ and weighted by the transition probability, $p_a(x, \tilde{x})$, that depends on the action a and external events like demand and yield. We formulate the model as a discounted Markov decision process (Bellman 1957). The optimal infinite horizon cost, $J(x)$, can then be written as

$$J(x) = \min_{a \in \mathscr{A}_x} c(x,a) + \gamma \sum_{\tilde{x} \in \mathscr{S}} p_a(x, \tilde{x}) J(\tilde{x}). \tag{2}$$

The transition probability $p_a(x, \tilde{x})$ is 0 for states $\tilde{x}$ that are no transition states of x. It consists of the transition probabilities of the inventory level $p_a^{inv}(x, \tilde{x})$ and open orders $p_a^{ord}(x, \tilde{x})$ as well as the transition function of tracking decisions $p_a^{tra}(x, \tilde{x})$.

These three terms are independent of one another. p_a^{inv} denotes the probability for transferring inventory from the current state x to transition state $\tilde{x}$. It only depends on the inventory level of the current state x and transition state $\tilde{x}$, demand of the current period, the order that we placed λ periods ago and its tracking status, that is, p_a^{inv} depends on I, $\tilde{I}$, D, O_λ and ψ_λ. p_a^{ord} is the transition probability of open orders and depends on the open orders O_i of states x and $\tilde{x}$ with $1 \leq i < \lambda$. The reason that p_a^{ord} does not depend on O_λ is that this state variable does not result in a change of open orders in the transition state but only influences its inventory level $\tilde{I}$. Since p_a^{inv} and p_a^{ord} use different state variables and as state variables are independent of one another, these two probabilities are independent of each other as well. p_a^{tra} is deterministic and returns 1 if $\tilde{x}$ is a possible transition state of x with respect to the tracking decisions and 0 otherwise. We therefore write $p_a(x, \tilde{x})$ as

$$p_a(x, \tilde{x}) = p_a^{inv}(x, \tilde{x}) \cdot p_a^{ord}(x, \tilde{x}) \cdot p_a^{tra}(x, \tilde{x}).$$

The inventory level transition probability is defined as

$$p_a^{inv}(x, \tilde{x}) = \mathbb{P}(\tilde{I} = I - D + Q_\lambda). \tag{3}$$

This is to say, it equals the probability that the increase in inventory, $\tilde{I} - I$, is equal to the difference between the delivery quantity and the demand, $Q_\lambda - D$. The term $Q_\lambda - D$ is stochastic and its distribution is given by the convolution of Q_λ and $-D$. Note that for $\psi_\lambda = 1$, Q_λ is deterministic and the distribution of $Q_\lambda - D$ reduces to the distribution of $-D$ shifted by Q_λ.

To evaluate the order quantity transition probability, its tracking status needs to be considered. If an order is not tracked, no new information about its yield is revealed and the order quantity transfers to the transition state without change. If an order is tracked, we have information on its yield and the updated order quantity is transferred to the transition state. We compute the order quantity transition probability as

$$p_a^{ord}(x,\tilde{x}) = \prod_{i=0}^{\lambda-1}(1-\psi_i)p_a^{ord,unt}(x,\tilde{x},i) + \psi_i p_a^{ord,tra}(x,\tilde{x},i), \tag{4}$$

where $O_0 := \hat{O}, \psi_0 := \hat{\psi}$. The term consists of the transition probabilities of untracked open orders $p_a^{ord,unt}(x,\tilde{x},i)$ and tracked open orders $p_a^{ord,tra}(x,\tilde{x},i)$. Note that this distinction is important since we only receive information on yield if an order is tracked. The probabilities for untracked and tracked open orders are defined as

$$p_a^{ord,unt}(x,\tilde{x},i) = \mathbb{I}[\tilde{O}_{i+1} = O_i]$$

and

$$p_a^{ord,tra}(x,\tilde{x},i) = \mathbb{P}(Y_{i+1}\cdot O_i = \tilde{O}_{i+1}).$$

Each open order is either tracked ($\psi_i = 1$) or untracked ($\psi_i = 0$). In case that the order is untracked, $p_a^{ord,unt}$ is evaluated. $p_a^{ord,unt}$ returns 1 if the open order quantity is equal to the open order quantity of the transition state because without tracking, we receive no information on the quality of the order. In case that the order is tracked, $p_a^{ord,tra}$ is evaluated. Recall that Y_i is a random variable that describes the realized yield rate between 0 and 1 of one lead time period with expected value u_i. The transition of inventory is stochastic because tracking allows to update the order quantity. The probability that the order quantity transfers to the transition state equals the probability that the yield Y_i corresponds to the change of the order quantity. For example, if Y_i is Bernoulli-distributed, its value is either 1 or 0 and O_i either spoils so that $\tilde{O}_{i+1}=0$ or not so that $\tilde{O}_{i+1} = O_i$. Since an order is either tracked or untracked, it is never the case that both $p_a^{ord,unt}$ and $p_a^{ord,tra}$ are greater than 0 at the same time.

The transition function for tracking decisions is given by

$$p_a^{tra}(x,\tilde{x}) = \prod_{i=0}^{\lambda-1}\mathbb{I}[\tilde{\psi}_{i+1} = \psi_i]. \tag{5}$$

p_a^{tra} returns 1 if the tracking decisions for all orders are the same in the transition state shifted by one period. If one tracking decision is not the same, p_a^{tra} returns 0 because then $\tilde{x}$ cannot be a transition state of x.

In the following, we denote $J_t(x)$ as the finite horizon version of the infinite horizon cost function $J(x)$. It is defined as $J_t(x) = \min_{a\in\mathscr{A}_x} c(x,a) + \gamma\sum_{\tilde{x}\in\mathscr{S}} p_a(x,\tilde{x})J_{t+1}(\tilde{x})$. It corresponds to $J(x)$ except that it is time-dependent and that there is a terminal cost function $J_T(x) = 0$ after T periods. $J_t(x)$ is useful to prove convergence properties of $J(x)$. Lemma 1 shows that $\mathscr{A}$ is a compact set. This lemma is necessary to prove uniform convergence of $J_t(x)$ to $J(x)$. All proofs are included in the appendix.

Lemma 1. *$\mathscr{A}_x$ is a compact set for each $x \in \mathscr{S}$.*

Lemma 2 states that $J_t(x)$ converges uniformly to $J(x)$. This means that the finite horizon costs converge to the infinite horizon costs when the time horizon goes to infinity.

Lemma 2. *$J_t(x)$ converges uniformly (and absolutely) to $J(x)$ in $\mathscr{S}$.*

This observation is crucial to prove Theorem 1, which states that the limit function satisfies the Bellman Equation and that for each state $x \in \mathscr{S}$, the minimum is obtained by some $a \in \mathscr{A}_x$.

Theorem 1. *$\lim_{T\to\infty} J_t(x)$ satisfies Equation (2) for each $x \in \mathscr{S}$.*

Finally we establish in Theorem 2 that a stationary policy exists. This shows that our policies do not need to regard the time period of the state space in the infinite horizon model.

Theorem 2. *For each $x \in \mathscr{S}$, there exists a stationary optimal policy with cost function $J(x)$.*

These findings are mandatory for designing optimal and approximate solution algorithms. We continue by analyzing structural properties of this Markov decision process.

3.3. Equivalent states

The state space includes some equivalent states, that is, states with identical one-period costs and identical follow-up states. We can aggregate such states to reduce the state space, which saves memory and reduces runtime. In the following, we present two classes of equivalent states.

If a tracked order placed i periods ago spoils, O_i becomes 0 while ψ_i remains 1. There is no difference between a state with such a spoiled order and a state with an untracked order with quantity 0 that is otherwise equal. The tracking fee τ is charged when the order is placed and tracking an empty order has no effect on future inventory levels. When a tracked order O_i is spoiled, we can set ψ_i to 0 and treat this order as if it had never been tracked. This finding is formulated in Proposition 1.

Proposition 1. *Let state x^1 be defined as $(I, O_1, \ldots, O_\lambda, \psi_1^1, \ldots, \psi_\lambda^1)$ and let state x^2 be defined as $(I, O_1, \ldots, O_\lambda, \psi_1^2, \ldots, \psi_\lambda^2)$, so that they share the same inventory/backorder level and open order quantities. x^1 and x^2 are equivalent with respect to one-period costs, optimal decisions and the distribution of follow-up states if for all $1 \le i \le \lambda$, it holds that $\psi_i^2 = \psi_i^1$ for $O_i > 0$ and $\psi_i^2 \in \{0,1\}$ for $O_i = 0$.*

Proposition 1 is particularly useful for situations in which order quantities are frequently equal to 0. It is also beneficial for settings with low yield rates, which lead to frequently spoiled orders. When yield rates are high and orders are placed in most or all periods, the benefits of the aggregation are small.

We can also aggregate states of tracked orders that arrive at the end of the current period, O_λ. If the order has been tracked, there is no uncertainty and we add the order quantity to our inventory level. We describe this characteristic in Proposition 2.

PROPOSITION 2. *Let state x^1 be defined as $(I^1, O_1, \dots, O_{\lambda-1}, O^1_\lambda, \psi_1, \dots, \psi_\lambda)$ and let state x^2 be defined as $(I^2, O_1, \dots, O_{\lambda-1}, O^2_\lambda, \psi_1, \dots, \psi_\lambda)$, so that they share the same tracking decisions for $1 \leq i \leq \lambda$ and open order quantities for $1 \leq i \leq \lambda - 1$. States x^1 and x^2 are equivalent with respect to one-period costs, optimal decisions and the distribution of follow-up states for any O^1_λ if $\psi_\lambda = 1$, $O^2_\lambda = 0$ and $I^2 = I^1 + O^1_\lambda$.*

Proposition 2 effectively reduces the state space by one dimension if the order placed λ periods ago is tracked. In this case, the arriving quantity Q_λ is equal to O_λ and we can add it to the current inventory level. This aggregation is not feasible if the order is not tracked.

3.4. Bounds

In the following, we refer to the solution of the Bellman Equation $J(x)$ defined in Equation (2) as cost of a state x under the optimal policy. For our solution approaches presented in Section 4, it is useful to specify lower bounds on the cost of a state under the optimal policy, and lower bounds on the optimal order quantity in a state. Lower bounds on the cost of a state under the optimal policy are useful because they serve as starting cost-to-go for our solution approaches. The solution approaches are based on value iteration, where the optimal costs are reached from below. We can always use 0 as initial cost-to-go, but higher initial values shorten the runtimes.

Lower bounds on the optimal order quantity reduce the action space and therefore limit the realm of states visited within a learning algorithm. This limitation serves two purposes: Firstly, it reduces the runtimes. Secondly, it is necessary for convergence proofs of simulation-based learning algorithms, which we present in Section 4.

In this section, we first derive two lower bounds on the optimal cost of a state, where for each state we choose the larger one as initial cost-to-go for our algorithms. We then continue with describing a lower bound on the optimal order quantity of a state.

3.4.1. Lower bounds on optimal cost of a state

Let $(x_t)_{t\in\mathbb{N}}$ be a sequence of states and let $(a_t)_{t\in\mathbb{N}}$ be a sequence of actions made in these states, where t denotes the time period. According to Theorem 1, we can write $J(x_1)$ for an arbitrary state x_1 also as $\lim_{T\to\infty} J_t(x_1)$. We can also write $J_t(x_1)$ as a minimization of a sum of costs. It follows that

$$J(x_1) = \lim_{T\to\infty} J_t(x_1) = \lim_{T\to\infty} \min_{a_1,\dots,a_T} \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} c(x_t, a_t)\right].$$

The order decision in period 1 affects inventory levels of periods $1+\lambda$ and later. In periods $1,\ldots,\lambda$, inventory levels depend on the order pipeline that is entirely stored in our state. Based on this observation, we can formulate a first lower bound, $\underline{J}(x_1)$, on our cost function that does not depend on the order decision in period 1:

Proposition 3. *For any state $x_1 \in \mathscr{S}$,*

$$\underline{J}(x_1) = \mathbb{E}\left[\sum_{t=1}^{\lambda-1} \gamma^{t-1} c_{inv}(x_t)\right] \leq \lim_{T\to\infty} J_t(x_1) = J(x_1).$$

This bound consists of the expected inventory costs of the first λ periods. In scenarios with high penalty cost factors and low yield rates, it provides a close lower bound on the cost of a state under the optimal policy.

The periodic review model with deterministic demand and perfect yield provides costs for each state that are a second possible lower bound to the costs in our dynamic tracking model. It follows that for each state $x \in \mathscr{S}$, $\underline{\underline{J}}(x) \leq J(x)$, where $\underline{\underline{J}}(x)$ describes the optimal costs in the deterministic periodic review model with perfect yield. The optimal decision as well as the optimal costs for this setting can be calculated using a recursive function that is shown in Appendix EC.9.

The idea of this function is that there is an absorbing state in the deterministic case. When we reach this state, the infinite horizon discounted costs can be calculated with a closed-form expression.

Since it is not predetermined whether $\underline{J}(x)$ or $\underline{\underline{J}}(x)$ is lower for a specific state x, we calculate both values and choose the larger one whenever we need to find a starting cost-to-go value for a state.

3.4.2. Lower bound on order quantity

The optimal order decision of the stochastic periodic review model with perfect yield is a lower bound for the optimal order decision of our dynamic tracking program if we decide not to track. This finding is shown in general by Henig and Gerchak (1990) in Theorem 7. For all states $x \in \mathscr{S}$ and if we decide not to track, the optimal order decision, $\hat{O}^*$, is greater than or equal to the optimal order decision, $\underline{\hat{O}}$, in the stochastic demand, perfect yield scenario. $\underline{\hat{O}}$ can be calculated as $F^{-1}_{D_{\lambda+1}}\left(\frac{p}{p+h}\right) - I - \sum_{i=1}^{\lambda} O_i$, where $F^{-1}_{D_{\lambda+1}}$ is the inverse cumulative distribution function of the demand over the order lead time plus one period.

In the case that we decide to track, $\underline{\hat{O}}$ is not always a lower bound on the order quantity because tracking cost occurs, so that the cost function is only convex for $\hat{O} > 0$. It follows that an order quantity of 0 might be more profitable than $\underline{\hat{O}}$. Thus, for a tracked order, $\hat{O}^*(x) \in \{0\} \cup [\underline{\hat{O}}, \infty)$.

However, we can define a condition for which $\underline{\hat{O}}$ is a lower bound on the order quantity if we decide to track because ordering at least one tracked item is profitable:

PROPOSITION 4. *For any state $x = (I, O_1, \ldots, O_\lambda, \psi_1, \ldots, \psi_\lambda)$, if we decide to track, the optimal order quantity $\hat{O}^*$ is greater than or equal to $\underline{\hat{O}}$ if $\tau < p\prod_{i=1}^{\lambda} u_i$ and $I_1 + O_1 + \cdots + O_\lambda + 1 \leq 0$.*

Intuitively, Proposition 4 shows that if our backorder level is so high that increasing our order quantity from 0 to 1 almost surely does not lead to overage, our expected cost reduction after λ periods is well-defined by the constant expression $p\prod_{i=1}^{\lambda} u_i$. As long as this cost reduction exceeds the tracking cost rate, it is worthwhile to order a positive quantity and we can apply the lower bound on the order quantity $\underline{\hat{O}}$.

4. Solution approaches

In Section 4.1, we present the value iteration approach for the problem formulated in the last section and show that it is provably optimal. Since runtime and memory requirements grow exponentially in λ with this method, it only works for small problem instances. In Section 4.2, we present a novel approximate dynamic programming approach that is capable of solving moderate instances of our problem setting. In Section 4.3, we illustrate a heuristical solution that works for large instances.

In the following, we assume that the demand, order quantities and therefore the state space are discrete.

4.1. Optimal solution approach

Our optimal solution approach is an enhanced synchronous value iteration algorithm that works with a finite and discrete state space and converges to the optimum (Bertsekas and Tsitsiklis 1996, pp. 25-26). In Proposition 5, we show that there is a unique essential class of states in which all states communicate. When applying the stationary optimal policy, no state outside this essential class is visited.

PROPOSITION 5. *For any discrete distribution of the demand D with $\mathbb{P}(D = i) > 0$ for $\forall i > 0$ and i.i.d. yield rates Y_j with $\mathbb{P}(Y_j = 1) > 0$, $1 \leq j \leq \lambda$, the state space $\mathscr{S}$ has a unique essential class and therefore a unique stationary distribution.*

Without loss of generality, we set bounds for the state space that are not binding for the essential class of states, so that we can apply synchronous value iteration. Let C^{syn} be the cost-to-go vector covering all states and let $C^{syn}(x)$ be the costs-to-go for state x that are currently stored in the vector. We introduce the operator R that chooses the optimal action for one state and updates its cost such that it is minimized given the current cost vector:

$$(RC^{syn})(x) = \min_{a \in \mathscr{A}_x} c(x, a) + \gamma \sum_{\tilde{x} \in \mathscr{S}} p_a(x, \tilde{x}) C^{syn}(\tilde{x}), \forall x \in \mathscr{S}. \tag{6}$$

We can then write the system of equations that belong to our setting as $RC^{syn} = C^{syn}$. Bertsekas and Tsitsiklis (1996, pp. 37-41) show that R is a contraction. It follows that choosing some starting solution for C^{syn} and applying R infinitely often leads to optimal cost $J(x)$ for each state.

We ensure that the algorithm terminates in finite time by combining this operator with MacQueen extrapolation (MacQueen 1966). The algorithm computes upper and lower bounds on the cost for each state in each iteration. When the largest distance between the bounds is beneath some small value ϵ, the algorithm stops. Knowing that there exists an optimal stationary policy from Theorem 2, we conclude Corollary 1:

COROLLARY 1. *The value iteration algorithm combined with MacQueen extrapolation yields the optimal stationary policy for a sufficiently small ϵ.*

Since we know from Proposition 5 that a unique stationary distribution exists, we can determine steady state probabilities $p(x), x \in \mathscr{S}$, for example, by using power iteration. We can then determine the weighted average costs of the state space under the optimal policy by $\sum_{x\in\mathscr{S}} p(x)J(x)$.

Synchronous value iteration visits all states of the state space. In case of a finite state space and a limitation of the range of inventory levels to $I \in \mathscr{I}$, the number of states is $|\mathscr{I}| \cdot |\mathscr{A}|^{\lambda}$, when we assume that the action space, $\mathscr{A}$, is the same for each state. This exponential growth of the state space in λ is commonly referred to as the curse of dimensionality. Due to the necessity to iterate the entire state space, it is computationally intractable to solve large instances of our setting.

4.2. Approximate dynamic programming solution approach

We develop a novel approximate dynamic programming approach to solve larger problem instances based on the asynchronous value iteration algorithm (Bertsekas and Tsitsiklis 1996). We introduce an aggregation technique where states with similar costs are aggregated on multiple levels. These levels are organized according to an aggregation hierarchy. Aggregations on higher levels cover more states than aggregations on lower levels but exhibit a higher cost estimation error. We assign each state to exactly one level to trade off cost estimation errors against the size of the state space.

In Section 4.2.1, we present the asynchronous value iteration algorithm without aggregation that serves as the basis of our hierarchical approximate dynamic programming approach. In Section 4.2.2, we delineate the aggregation functions. In Section 4.2.3, we demonstrate how we can use these aggregation functions to adapt asynchronous value iteration to solve exactly one aggregation level. In Section 4.2.4, we relax the number of aggregation levels and develop rules to assign aggregation levels to particular states and present our complete hierarchical approximate dynamic programming algorithm.

4.2.1. Asynchronous value iteration

The asynchronous value iteration algorithm updates only one state per iteration according to a greedy policy (Bertsekas and Tsitsiklis 1996, pp. 237-245). Let C^{asy} be the cost-to-go vector for our entire state space under asynchronous value iteration and let $C^{asy}(x)$ be the cost-to-go of state

x that are currently stored in the vector. Starting at an arbitrary state $x_k \in \mathscr{S}$ at iteration k, we perform a greedy policy to obtain the best decision given our current cost information of the state space. We update our state space after each iteration according to

$$C^{asy}(x_k) \leftarrow \min_{a \in \mathscr{A}_{x_k}} c(x_k, a) + \gamma \sum_{\tilde{x}_k \in \mathscr{S}} p_a(x_k, \tilde{x}_k) C^{asy}(\tilde{x}_k),$$

where the right hand-side corresponds to Equation (6). We then move to the follow-up state x_{k+1}. It depends on the decision made and external events like demand and yields for the particular orders. We choose the optimal decision made in the minimization step and then choose x_{k+1} randomly as follow-up state $\tilde{x}_k$ according to the probabilities $p_a(x_k, \tilde{x}_k)$. Then, the next iteration begins and the process repeats.

We must initialize C^{asy} such that $C^{asy}(x) \leq J(x)$ for each $x \in \mathscr{S}$. A simple starting point would be 0. Since the rate of convergence depends on how close the initial cost of a state is to the optimal cost, an increased initial cost can reduce the runtime of our algorithm. Therefore, we initially set $C^{asy}(x)$ to the maximum of the lower bound of Proposition 3, $\underline{J}(x)$, and the optimal cost from the periodic review model with deterministic demand, $\underline{\underline{J}}(x)$. The state trajectory resulting from our algorithm starts with many potential states and converges to states belonging to the optimal decisions. During the learning process, the cost-to-go vector C^{asy} converges to the optimal costs given by $J(x)$ for all states resulting from the optimal decision (Bertsekas and Tsitsiklis 1996, pp. 237-245). Without loss of precision, we apply the order quantity bounds from Theorem 7 of Henig and Gerchak (1990) and our Proposition 4 to further reduce runtimes.

The asynchronous value iteration algorithm with bounds is depicted in Appendix EC.12. It has a finite search space and converges as stated in Corollary 2.

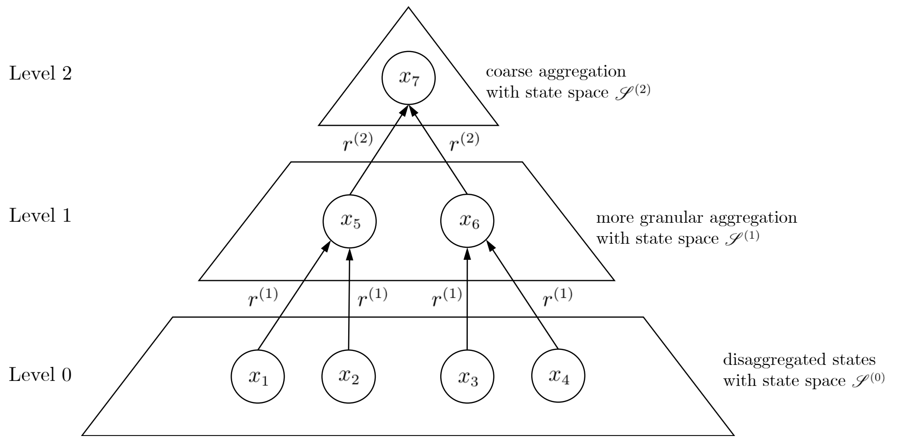
Corollary 2. *The asynchronous value iteration algorithm with bounds leads to a finite optimal solution after an infinite number of iterations that is obtained using a stationary optimal policy.*

We use policy evaluation to find a termination criterion by measuring the average cost associated with the current policy. We then observe the trend of average costs over time and find a reasonable number of iterations after which to terminate.

4.2.2. Aggregation functions

Figure 1 shows an example of an aggregation pyramid for three hierarchical levels. Level 0 contains only fully disaggregated states. Level 1 is built by aggregating similar states of level 0. Level 2 is coarser and is created by aggregating states of level 1. We denote the state spaces consisting of states from levels 0, 1 and 2 as $\mathscr{S}^{(0)}$, $\mathscr{S}^{(1)}$ and $\mathscr{S}^{(2)}$, respectively. In this example, function $r^{(1)}$ transforms a disaggregated state of level 0 into an aggregated state of level 1, and function $r^{(2)}$ transforms a state of level 1 into an even more aggregated state of level 2.

Figure 1 **Example with three aggregation levels, aggregation functions $r^{(\cdot)}$ and associated state spaces $\mathscr{S}^{(\cdot)}$ where higher levels are coarser**


Level 2
x_7
coarse aggregation with state space $\mathscr{S}^{(2)}$
$r^{(2)}$
$r^{(2)}$
Level 1
x_5
x_6
more granular aggregation with state space $\mathscr{S}^{(1)}$
$r^{(1)}$
$r^{(1)}$
$r^{(1)}$
$r^{(1)}$
Level 0
x_1
x_2
x_3
x_4
disaggregated states with state space $\mathscr{S}^{(0)}$

Formally, for each level of the aggregation hierarchy, we define aggregated state spaces $\mathscr{S}^{(g)}$, where we denote their states as $x^{(g)} \in \mathscr{S}^{(g)}$ with $x^{(g)} = (g, I, O_1, \cdots, O_\lambda, \psi_1, \cdots, \psi_\lambda), g \in \mathscr{G}$. $\mathscr{G}$ is the index set of aggregation levels with $|\mathscr{S}^{(k)}| < |\mathscr{S}^{(l)}|$ for $k > l, k \in \mathscr{G}, l \in \mathscr{G}$. For notational convenience, we denote the disaggregated state space as $\mathscr{S}^{(0)}$ and a state of this state space as $x^{(0)} \in \mathscr{S}^{(0)}$. $\mathscr{S}^{(0)}$ is equivalent to $\mathscr{S}$ that we have utilized so far and only differs in the additional constant $g = 0$ as the first state variable. We generalize aggregation functions $r^{(g)} : \mathscr{S}^{(f)} \to \mathscr{S}^{(g)}$ with $f, g \in \mathscr{G}, f < g$ for a fixed f per $r^{(g)}$. Setting f allows us to choose the aggregation level from which we take states that we aggregate into level g. Choosing a smaller f increases the precision of the aggregation while it increases the complexity of the calculation. The example in Figure 1 corresponds to $f = \max\{g-1, 0\}$.

Since the total number of disaggregated states grows exponentially in λ according to $|\mathscr{I}| \cdot |\mathscr{A}|^\lambda$, we develop aggregation functions that counteract the growth by an exponential state reduction in λ. We eliminate one dimension of open orders per aggregation level by setting the corresponding open order state variable to 0. This is a common aggregation technique (e.g., George et al. 2008, Mes et al. 2011, Simão et al. 2009). As this changes the cost of the state, we adjust the value of O_λ such that the change in cost is minimized. The result is an aggregated state with minimal cost difference compared to the original, disaggregated state.

For example, we denote disaggregated state $x^{(0)}$ as $(0, I^{(0)}, O_1^{(0)}, O_2^{(0)}, \cdots, O_\lambda^{(0)}, \psi_1^{(0)}, \psi_2^{(0)}, \cdots, \psi_\lambda^{(0)})$ and $x^{(1)}$, the aggregated state of level 1, as $(1, I^{(1)}, O_1^{(1)}, O_2^{(1)}, \cdots, O_\lambda^{(1)}, \psi_1^{(1)}, \psi_2^{(1)}, \cdots, \psi_\lambda^{(1)})$. By aggregating, we set one open order state variable to 0, that is, $O_{\lambda-1}^{(1)} = 0$. We set $O_\lambda^{(1)}$ to $\zeta^{(1)}$ which is a value that we choose to minimize the cost difference between $x^{(0)}$ and $x^{(1)}$. All other state variables stay the same, that is, $I^{(1)} = I^{(0)}$, $O_i^{(1)} = O_i^{(0)}$ for $i < \lambda - 1$ and $\psi_i^{(1)} = \psi_i^{(0)}$ for $1 \leq i \leq \lambda$. While we could also set different state variables to 0 and $\zeta^{(1)}$, our computational studies revealed that the choice of $O_\lambda^{(1)}$ and $O_{\lambda-1}^{(1)}$ yields the best results.

Table 1 Source and target aggregation states for $f = \max\{g-2, 0\}$

Source aggregation state, $x^{(f)} \in \mathscr{S}^{(f)}$	Target aggregation state, $x^{(g)} \in \mathscr{S}^{(g)}$
$(0, I, O_1, \cdots, O_\lambda, \psi_1, \cdots, \psi_\lambda)$	$(1, I, O_1, \cdots, O_{\lambda-2}, 0, \zeta^{(1)}, \psi_1, \cdots, \psi_\lambda)$
$(0, I, O_1, \cdots, O_\lambda, \psi_1, \cdots, \psi_\lambda)$	$(2, I, O_1, \cdots, O_{\lambda-3}, 0, 0, \zeta^{(2)}, \psi_1, \cdots, \psi_\lambda)$
$(1, I, O_1, \cdots, O_{\lambda-2}, 0, \zeta^{(1)}, \psi_1, \cdots, \psi_\lambda)$	$(3, I, O_1, \cdots, O_{\lambda-4}, 0, 0, 0, \zeta^{(3)}, \psi_1, \cdots, \psi_\lambda)$
$(2, I, O_1, \cdots, O_{\lambda-3}, 0, 0, \zeta^{(2)}, \psi_1, \cdots, \psi_\lambda)$	$(4, I, O_1, \cdots, O_{\lambda-5}, 0, 0, 0, 0, \zeta^{(4)}, \psi_1, \cdots, \psi_\lambda)$
$(3, I, O_1, \cdots, O_{\lambda-4}, 0, 0, 0, \zeta^{(3)}, \psi_1, \cdots, \psi_\lambda)$	$(5, I, O_1, \cdots, O_{\lambda-6}, 0, 0, 0, 0, 0, \zeta^{(5)}, \psi_1, \cdots, \psi_\lambda)$

Aggregated states of higher levels can be found in the same way. We denote $x^{(g)}$ as $(g, I^{(g)}, O_1^{(g)}, O_2^{(g)}, \cdots, O_\lambda^{(g)}, \psi_1^{(g)}, \psi_2^{(g)}, \cdots, \psi_\lambda^{(g)})$. By aggregating, we set g state variables of open orders to zero, that is, $O_i^{(g)} = 0$ for $\lambda - g \leq i \leq \lambda - 1$. We set $O_\lambda^{(g)}$ to $\zeta^{(g)}$, which again is a value that we choose to minimize the cost difference. All other state variables stay the same.

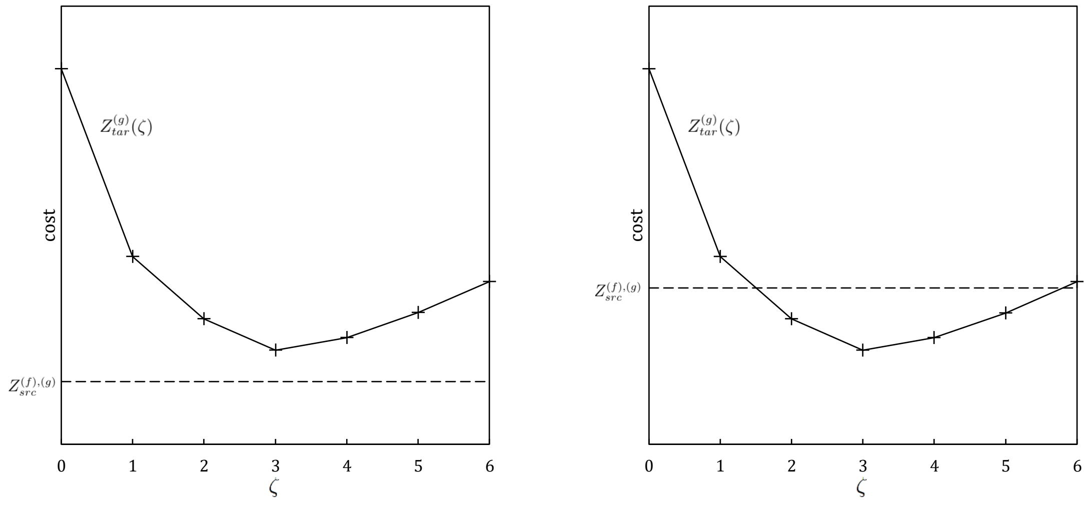
For any target aggregation level g, we may choose a source aggregation level that we denote as f with $0 \leq f < g$. We then seek to minimize the cost difference between $x^{(g)}$ and $x^{(f)}$ by setting $\zeta^{(g)}$. For example, if we choose f as the maximum of $g-2$ and 0, we come up with source and target aggregation states as depicted in Table 1, where we have left out the upper indices for unreplaced state variables that do not change from source to target aggregation state.

To optimize $\zeta^{(g)}$, we need to estimate the cost difference between $x^{(g)}$ and $x^{(f)}$. The exact cost difference is given by $|J(x^{(g)}) - J(x^{(f)})|$. During our value iteration algorithm, the optimal costs resulting from J are unknown. However, we know from Proposition 3 that up to λ states do not depend on the action except for the tracking decision. We approximate the cost difference by estimating $J(x^{(g)})$ and $J(x^{(f)})$ similarly to Proposition 3: J consists of an infinite sum of one-stage costs where all summands correspond to lead time periods. For target aggregation level g, we only regard the inventory costs of $g+1$ lead time periods because aggregation level g can be applied for $\lambda \geq g+1$. In this way, we approximate the cost difference myopically. Myopic policies that are based on cost estimates of few periods are frequently used in the approximate dynamic programming literature (e.g., Abdulwahab and Wahab 2014, Fang et al. 2013, Sauré et al. 2015). We use cost estimates not for a myopic policy itself, but for estimating the cost difference.

We define the inventory cost for end of period inventory z as $c_{ein}(z) = h[z]^+ - p[z]^-$. We denote the approximated cost of the source aggregation state $x^{(f)}$ as $Z_{src}^{(f),(g)}$ and write it as

$$Z_{src}^{(f),(g)} = \sum_{k=1}^{g+1} \gamma^{k-1} \mathbb{E}[c_{ein}(I^{(f)} - \sum_{l=1}^{k} D_l + \sum_{l=1}^{k} \mathbb{I}[\psi_{\lambda-l+1}^{(f)} = 0](O_{\lambda-l+1}^{(f)} \sum_{m=1}^{\lambda} Y_{m,l}) \\ + \mathbb{I}[\psi_{\lambda-l+1}^{(f)} = 1](O_{\lambda-l+1}^{(f)} \sum_{m=\lambda-l+2}^{\lambda} Y_{m,l})].$$

Figure 2 Conceptual example of $Z_{tar}^{(g)}(\zeta)$ and two possible $Z_{src}^{(f),(g)}$ locations



(a) $Z_{src}^{(f),(g)}$ below $Z_{tar}^{(g)}$ for any ζ (b) $Z_{src}^{(f),(g)}$ above minimum of $Z_{tar}^{(g)}$

D_l are i.i.d. random variables, which all exhibit the same distribution as demand D. $Y_{m,l}$ are i.i.d. random variables that all exhibit the same distribution as random yield rate Y_m, that is, for the random yield rate of lead time period m.

We denote the approximated cost of the target aggregation state $x^{(g)}$ as $Z_{tar}^{(g)}$ and write it as

$$Z_{tar}^{(g)}(\zeta) = \sum_{k=1}^{g+1} \gamma^{k-1} \mathbb{E}[c_{ein}(I^{(g)} - \sum_{l=1}^{k} D_l + \mathbb{I}[\psi_\lambda^{(g)} = 0](\zeta \sum_{n=1}^{\lambda} Y_n) + \mathbb{I}[\psi_\lambda^{(g)} = 1]\zeta)].$$

Note that we have replaced $O_\lambda^{(g)}$ with parameter ζ. Given these cost approximations, we can find the optimal value $\zeta^{(g)}$ by solving

$$\zeta^{(g)} = \arg\min_{\zeta} |Z_{src}^{(f),(g)} - Z_{tar}^{(g)}(\zeta)|. \tag{7}$$

Proposition 6 states that $Z_{tar}^{(g)}(\zeta)$ is convex in ζ. We can utilize this property to solve Equation (7) for ζ efficiently.

Proposition 6. *$Z_{tar}^{(g)}(\zeta)$ is convex in ζ.*

Figure 2 depicts the potential values of $Z_{tar}^{(g)}$ relative to $Z_{src}^{(f),(g)}$. Figure 2a shows the case where $Z_{src}^{(f),(g)}$ is smaller than $Z_{tar}^{(g)}(\zeta)$ for any ζ. In the example, we obtain the minimal difference for $\zeta = 3$. Figure 2b shows the case that $Z_{src}^{(f),(g)}(\zeta)$ lies above the minimum of $Z_{tar}^{(g)}$. In this case, there are up to four ζ candidates with minimal distance because $\zeta^{(g)}$ needs to be integer. In this example, the candidates are 1, 2, 5, and 6. We choose the one with the minimal difference.

In case of multiple optimal solutions, we select the one with the lowest ζ, so that we minimize the number of resulting states in the first aggregation level.

We denote the aggregation function that takes $x^{(f)}$ as input and yields the aggregated state $x^{(g)}$ by solving Equation 7 as $r^{(g)}$ with $x^{(g)} = r^{(g)}(x^{(f)})$.

4.2.3. Learning process for one aggregation level

We start by describing the learning process for the case that each disaggregated state is assigned to exactly one aggregation level $g \in \mathscr{G}^+$, where we define $\mathscr{G}^+ = \mathscr{G} \setminus \{0\}$. The structure of the algorithm corresponds to that of Section 4.2.1. In each iteration k, we move from one disaggregated state $x_k^{(0)} \in \mathscr{S}^{(0)}$ to another by sampling random yields and random demand. Since we utilize state aggregations, we do not store costs and decisions of $x_k^{(0)}$, but the average costs and aggregated decision of the aggregated state obtained by $r^{(g)}(x_k^{(0)})$, to which multiple disaggregated states contribute. Let C^{ag1} be the cost-to-go vector for the learning process and let $C^{ag1}(r^{(g)}(x_k^{(0)}))$ be the cost-to-go for state $x_k^{(0)}$ aggregated on level g. We learn the optimal decision with

$$a_k^*(r^{(g)}(x_k^{(0)})) \leftarrow \arg\min_{a_k} c(x_k^{(0)}, a_k) + \gamma \sum_{\tilde{x}_k^{(0)} \in \mathscr{S}^{(0)}} p_{a_k}(x_k^{(0)}, \tilde{x}_k^{(0)}) C^{ag1}(r^{(g)}(\tilde{x}_k^{(0)})), \tag{8}$$

where $a_k \in \mathscr{A}_{x_k^{(0)}}$. We update the weighted cost average for the aggregated state obtained via $r^{(g)}(x_k^{(0)})$ by smoothing according to

$$\begin{aligned} C^{ag1}(r^{(g)}(x_k^{(0)})) \leftarrow & \alpha\big[c(x_k^{(0)}, a_k^*(r^{(g)}(x_k^{(0)}))) + \gamma \sum_{\tilde{x}_k^{(0)} \in \mathscr{S}^{(0)}} p_{a_k^*(r^{(g)}(x_k^{(0)}))}(x_k^{(0)}, \tilde{x}_k^{(0)}) C^{ag1}(r^{(g)}(\tilde{x}_k^{(0)}))\big] + \\ & (1-\alpha) C^{ag1}(r^{(g)}(x_k^{(0)})), \end{aligned} \tag{9}$$

where α is referred to as the step size (Powell 2011, p. 245).

Next, we demonstrate how we use multiple aggregation levels to assign seldom visited states to coarse aggregation levels and frequently visited states to more granular aggregation levels.

4.2.4. Learning process for hierarchy of aggregation levels

We define $\hat{g}$ as $|\mathscr{G}^+|$, that is, the highest possible aggregation level under lead time λ. A state on this level can be described as $x^{(\hat{g})} = (\hat{g}, I, 0, \cdots, 0, \zeta^{(\hat{g})}, \psi_1, \cdots, \psi_\lambda)$, that is, a state where all but one original open order state variables are set to 0 and the one open order state variable is replaced by the value $\zeta^{(\hat{g})}$. Only considering the highest aggregation level leads to fast convergence, albeit only on a coarse level. To trade off convergence time with precision, we introduce fixed weights $w^{(g)}$ that define how many probablity-weighted states we keep at each aggregation level. For example, for $\lambda = 3$, let $w^{(1)} = w^{(2)} = 30\%$. Then, we have two aggregation levels with associated state spaces $\mathscr{S}^{(1)}$ and $\mathscr{S}^{(2)}$. The states with the smallest visit frequencies that account for 30% of the total probability mass of all states are assigned to aggregation level 2. Of the remaining 70%, the states with the smallest visit frequencies that account for 30% of the remaining probability mass are assigned to aggregation level 1. We keep $70\% \cdot 70\% = 49\%$ of all states at a disaggregated level.

Let $e_k^{(g)}: \mathscr{S}^{(g)} \to \{0,1\}$ for all $g \in \mathscr{G}^+$ and for all $k \in \mathbb{N}$ be the family of functions that yield whether a specific aggregated state is utilized or not in iteration k, that is, whether the corresponding aggregated state is enabled. If an aggregated state is disabled, we utilize a lower, more granular aggregation level.

At the beginning of our algorithm, we assign all states to the highest possible aggregation level and conduct a learning according to the modified algorithm described in Section 4.2.3. That is,

$$e_0^{(g)}(x) = \mathbb{I}[g = \hat{g}], \forall g \in \mathscr{G}^+, \forall x \in \mathscr{S}^{(g)}.$$

Starting at $i = \hat{g}$, after a number of iterations, $M^{(i)}$, we enable the states of state space i with the least visits, such that their probability mass is $w^{(g)}$. We disable all other states on this aggregation level. We achieve this efficiently with the following technique.

For all $1 \le i \le \hat{g}$, at iteration $k = M^{(i)}$, we determine a visit frequency threshold, $\bar{\beta}_k^{(i)}$. The states that have a visit frequency below this threshold have a total state probability mass of at most $w^{(i)}$. We formally define the threshold as

$$\bar{\beta}_k^{(i)} = \max \beta, \text{ s.t.} \sum_{x \in \mathscr{S}^{(i)}} \frac{\mathbb{I}[q_k(x) \le \beta] q_k(x)}{\sum_{y \in \mathscr{S}^{(i)}} q_k(y)} \le w^{(i)},$$

where $q_k(x)$ is the frequency of visiting some state x after k iterations. We collect the corresponding states in the set $S_k^{(i)} = \{ x \in \mathscr{S}^{(i)} \mid q_k(x) \le \bar{\beta}_k^{(i)} \}$ and set $e_k^{(g)}(x) = \mathbb{I}[x \in S_k^{(i)}]$ for all $x \in \mathscr{S}^{(i)}$.

Next, we utilize $e_k^{(g)}$ to choose a specific aggregation level. For a state $x_k^{(0)} \in \mathscr{S}^{(0)}$ in iteration k, we choose the highest aggregation level for which $e_k^{(g)}$ is 1. We denote the corresponding aggregation state by $x_k^{(g^*)}$. We recursively define

$$r(x_k^{(0)}, g) = \begin{cases} r^{(g)}(x_k^{(0)}), & \text{for } e_k^{(g)}(r^{(g)}(x_k^{(0)})) = 1 \\ r(x_k^{(0)}, g-1), & \text{else} \end{cases} \tag{10}$$

for $g \in \mathscr{G}^+$ and $r(x_k^{(0)}, 0) = x_k^{(0)}$, so that $x_k^{(g^*)} = r(x_k^{(0)}, \hat{g})$. We henceforth write $r(x_k^{(0)}, \hat{g})$ as $r(x_k^{(0)})$. Equation (10) uses the aggregation functions defined in Section 4.2.2 to obtain the aggregated state for $e_k^{(g)}(r^{(g)}(x_k^{(0)})) = 1$. If $e_k^{(g)}(r^{(g)}(x_k^{(0)}))$ is 0 for all $g \in \mathscr{G}^+$, then $r(x_k^{(0)})$ yields the disaggregated state $x_k^{(0)}$. We obtain the new cost and updating equations by replacing $r^{(g)}(\cdot)$ with $r(\cdot)$ in Equations (8) and (9).

Let C^{agr} be the cost-to-go vector for the learning process and let $C^{agr}(r(x_k^{(0)}))$ be the cost-to-go for state $x_k^{(0)}$ aggregated on the highest enabled level g^*. We learn the optimal decision with

$$a_k^*(r(x_k^{(0)})) \leftarrow \min_{a_k} c(x_k^{(0)}, a_k) + \gamma \sum_{\tilde{x}_k^{(0)} \in \mathscr{S}^{(0)}} p_{a_k}(x_k^{(0)}, \tilde{x}_k^{(0)}) C^{agr}(r(\tilde{x}_k^{(0)})),$$

where $a_k \in \mathscr{A}_{x_k^{(0)}}$. We update the weighted cost average for the aggregated state obtained via $r(x_k^{(0)})$ by smoothing according to

$$C^{agr}(r(x_k^{(0)})) \leftarrow \alpha\big[c(x_k^{(0)}, a_k^*(r(x_k^{(0)}))) + \gamma \sum_{\tilde{x}_k^{(0)} \in \mathscr{S}^{(0)}} p_{a_k^*(r(x^{(0)}))}(x_k^{(0)}, \tilde{x}_k^{(0)}) C^{agr}(r(\tilde{x}_k^{(0)}))\big] + (1-\alpha) C^{agr}(r(x_k^{(0)})).$$

This concludes the hierarchical approximate dynamic programming algorithm. It is fully depicted in Appendix EC.14.

Without aggregation, our approach converges to the optimal solution according to the asynchronous value iteration algorithm as formulated in Corollary 2.

4.3. Heuristic solution approaches

A well-known class of heuristics for periodic review settings with random yield is based on modified base-stock policies where the order size is inflated by a factor based on the yield rate. These policies are called linear inflation policies. The general idea behind these heuristics is to order more than under the standard base-stock policy because fewer ordered items arrive. The order quantity is $\iota(\theta - \mathbb{E}[IP])$ for $\mathbb{E}[IP] < \theta$ or 0 otherwise, where $\mathbb{E}[IP]$ is the expected inventory position, θ is the order threshold value and ι is the inflation factor. The expected inventory position can be calculated as

$$\mathbb{E}[IP] = I + \sum_{i=1}^{\lambda} \mathbb{I}[\psi_i = 0](O_i \prod_{j=1}^{\lambda} u_j) + \mathbb{I}[\psi_i = 1](O_i \prod_{j=i+1}^{\lambda} u_j).$$

For untracked orders, we expect the yield of the orders to be equal to the ordered items multiplied by the average yield rate over all lead time periods. For tracked orders, each order quantity has already passed a number of lead time periods and we only need to apply the average yield rate for the remaining number of lead time periods.

The choice of θ and ι depends on the heuristic. Bollapragada and Morton (1999) introduce the MULT heuristic, where θ is calculated as $F_{\lambda+1}^{-1}(p/(p+h))$ and ι as $\prod_{i=1}^{\lambda} u_i^{-1}$. The idea is to inflate the order size by the expected relative loss through random yield during the lead time.

For the case of $\lambda = 0$, Huh and Nagarajan (2010) develop an improved heuristic, where θ is optimized for a given ι. Dettenbach and Thonemann (2015) adjust it for a case with positive lead time. They propose an inflation factor of

$$\iota = \frac{1}{2}\left(\frac{1}{\prod_{i=1}^{\lambda} u_i} + \sup\left\{n : \mathbb{E}\left[\mathbb{I}\left[\frac{1}{n} \le \prod_{i=1}^{\lambda} Y_i\right] \cdot \prod_{i=1}^{\lambda} Y_i\right] \le \frac{p}{p+h} \cdot \prod_{i=1}^{\lambda} u_i\right\}\right)$$

that takes the yield rate distribution into account. Note that it simplifies to the inflation factor of the MULT heuristic if the yield rate is deterministic. They set

$$\theta = \inf\left\{\hat{\theta} : \frac{1}{T}\sum_{t=1}^{T} \mathbb{P}\Big(I_{t+1}^{(0,\iota)} + \hat{\theta} \le 0\Big) \le \frac{p}{p+h}\right\},$$

where $I_{t+1}^{(0,\iota)}$ denote the simulated inventory levels for $\theta = 0$. θ can be determined by simulation as described by Huh and Nagarajan (2010).

These existing heuristics only optimize the order decision and can handle the cases when we track always or never. We present a heuristic that chooses the order quantity according to the heuristic of Huh and Nagarajan (2010), extends it to the positive lead time setting and allows for dynamic tracking based on a threshold.

We define $\hat{\psi}$ by $\hat{\psi} = \mathbb{I}[\xi \mathbb{E}[IP] < \xi\mu]$. The parameter μ is a threshold value. The parameter ξ can be -1 or 1 and therefore determines whether an order is tracked for a positive lead time above or below the threshold. For fixed values of ξ and μ, we apply the heuristic of Huh and Nagarajan (2010) and determine the cost. We consider a range of values for ξ and μ and choose the solution with the minimal cost. Thus, our heuristic is always at least as good as tracking always or never.

5. Computational results

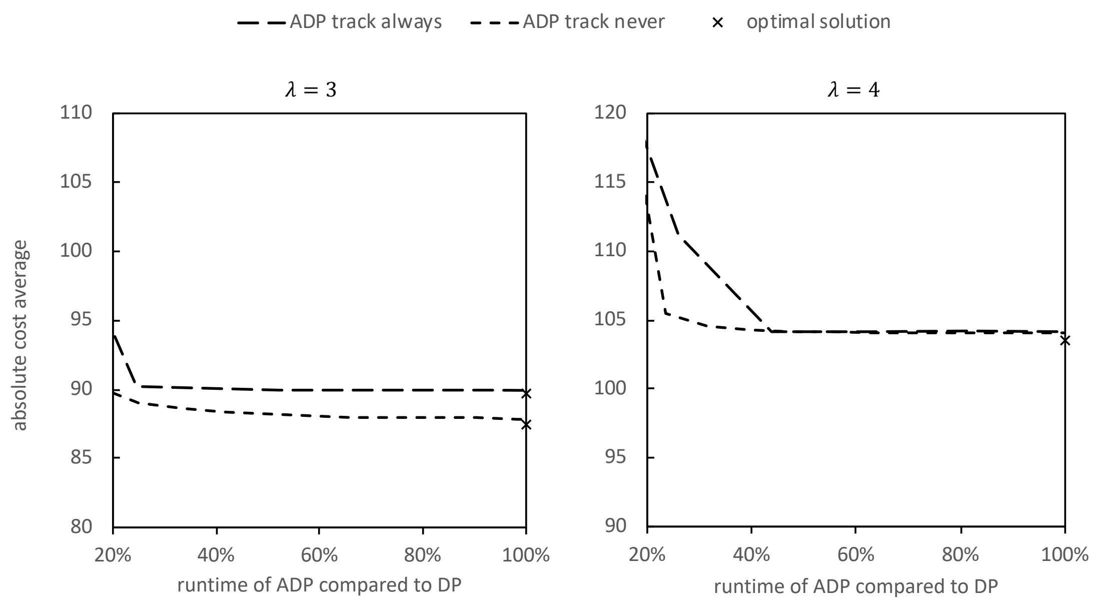
We evaluate the performance of our approaches numerically. We implement all approaches with C++ and conduct our numerical studies on hardware with 24 GB memory and CPUs with six 2.66 GHz cores (Xeon X5650).

In Section 5.1, we present optimal and approximate dynamic programming solutions for the cases of tracking never and always and $\lambda \leq 4$. In Section 5.2, we analyze the value of tracking always and the value of tracking dynamically for $\lambda \leq 4$. In Section 5.3, we compare the performance of our approximate dynamic programming approach to heuristical solutions for $\lambda = 5$ and $\lambda = 6$. We conclude in Section 5.4 by presenting the values of tracking always and dynamically for instances of $\lambda \geq 5$.

5.1. ADP versus optimal results

According to a European pharmaceutical company that we work with, yield rates between 0.9 and 0.98 per period are common in practice. We choose the same value u for all expected yield rates u_i of lead time periods i with $1 \leq i \leq \lambda$. We examine settings with u values of 0.9, 0.94 and 0.98. For D, we choose a Poisson distribution with parameter value 2 that is truncated at $D = 6$, and allocate the excess probability mass to $D = 6$, that is, we set $P(D = 6) \leftarrow P(D \geq 6)$. The truncation reduces the computational effort significantly and without truncation $P(D \leq 6)$ entails 99.5% of all demand values. We normalize the holding cost factor at 1 and use critical ratios of 0.9, 0.95 and 0.99 with corresponding penalty cost factors. For the tracking cost parameter τ, we consider values such that we can observe the area that contains the intersection of the costs of tracking always and never. For tracking costs smaller than the tracking cost corresponding to the intersection, tracking always yields the lowest total cost. For higher tracking costs, the optimal decision is to track never and the total cost is constant. Note that these values must be interpreted relative to the normalized

Figure 3 Cost development of ADP over time relative to optimal solutions for $\lambda = 3$ and $\lambda = 4$



holding cost factor rather than the absolute cost values. The discount factor γ is 0.9, which is a reasonable trade-off between convergence speed and sufficient future state coverage.

For the optimal result calculation, we use the loss-free synchronous value iteration algorithm as described in Section 4.1. It exploits our structural findings. We choose ϵ as 10^{-9}.

For our hierarchical approximate dynamic programming approach, we use $f = \max\{g-2, 0\}$. Choosing a smaller f leads to computational effort that slows down our algorithm. Choosing a higher f results in a loss in precision. We choose the same weights w_i for all aggregation levels i and set a value M, so that $M^{(i)} = (\lambda - i)M$. For $\lambda = 2$ and $\lambda = 3$, we select $M = 2 \cdot 10^6$ with $w_i = 0.0005$ for tracking never and $w_i = 0.001$ for tracking always. For $\lambda = 4$, we set $M = 1 \cdot 10^8$ and $w_i = 0.001$ for all tracking policies.

All results are evaluated using policy evaluation with 80 runs and 10^7 iterations per run, where we skip the first 1000 iterations per run. The 95% confidence interval for these runs has a spread of $\pm 0.2\%$ on average over all ADP scenarios.

The total costs of the optimal and approximate solutions are shown in Table EC.1 in Section EC.15 of the online appendix. Over all 162 scenarios, we observe an average error of the approximate solutions of 0.21 %. The runtimes for $\lambda = 2$ are below one CPU minute. For $\lambda = 3$, the optimal solutions take 69 CPU minutes on average. For $\lambda = 4$, the optimal solutions need 64 CPU hours on average.

The approximate dynamic programming approach leads to faster results while barely losing precision. This is illustrated in Figure 3 where the curves represent the interim results over time of the approximate dynamic programming approach for $\lambda = 3$ and $\lambda = 4$. Note that we omitted the case of $\lambda = 2$ due to the short and similar runtime of both the optimal and approximate dynamic programming approach. The optimal solutions for the scenarios of tracking never and always are marked with the "x" symbols that are closest to the curves. The time axes are scaled as CPU minutes of the approximate dynamic program divided by the CPU minutes of the optimal dynamic program. After approximately 44%, average costs have a difference of below 2% towards the optimal solution. Note that we could reduce the disaggregation iteration numbers $M^{(i)}$ or increase the aggregation level weights w_i to achieve much faster convergence with a higher error. The opposite changes would increase convergence time and increase precision of the results.

It is also wortwhile to note that the total number of states in the approximate dynamic programming approach is considerably lower than in the optimal dynamic programming approach. The ratio of used states including aggregated states and the number of states necessary for the optimal dynamic programming approach amount to 17.6% and 2.0% on average for tracking never and always, respectively. Tracking never has a lower state reduction benefit due to structural properties for tracked orders presented in Sections 3.3 and 3.4. The ratio decreases with increasing λ. Averaging over tracking methods, the ratio ranges from 5.2% at $\lambda = 2$ to 2.2% at $\lambda = 4$.

5.2. Value of tracking and dynamic tracking for small instances

For lead times λ of 3 and 4, the optimal costs for tracking never, always and dynamically are shown in Figure EC.1 in Section EC.16 of the online appendix with tracking costs on the x-axis. We average all values over critical ratios of 0.9, 0.95 and 0.99.

We define the value of tracking as the cost difference between always and never tracking, or 0 if negative. The value increases with decreasing tracking costs and yield rate. For $\lambda = 2$, the average value of tracking amounts to 1.2%, with the maximum being 10.9%. For $\lambda = 3$, the average value of tracking amounts to 1.5% with the maximum being 12.6%. For $\lambda = 4$, the average value of tracking amounts to 1.7%. The largest value is 13.9%.

We measure the value of dynamic tracking as the difference between tracking dynamically and the better of tracking always or never for the same parameters. It is apparent that tracking dynamically exhibits higher values for lower yield rates. The lead time has only a small effect. In all of our computations, the value of dynamic tracking is highest when the cost curves of always and never tracking coincide. We then obtain cost savings of up to 3.0% for $\lambda = 2$, up to 3.2% for $\lambda = 3$ and up to 2.8% for $\lambda = 4$.

5.3. ADP versus heuristics

For $\lambda = 5$ and $\lambda = 6$, the benefit of the ADP for the cases of tracking never and always is shown in Figure EC.2 in the online appendix. For $\lambda = 5$, we choose $M = 3 \cdot 10^8$ and $w_i = 0.01$. For $\lambda = 6$, we set $M = 3 \cdot 10^8$ and $w_i = 0.01$ for tracking never and $w_i = 0.001$ for tracking always. Compared to the heuristic of Huh and Nagarajan (2010), the ADP achieves average cost savings of 9.2% and maximum cost savings of 16.0% for $\lambda = 5$. The maximum cost savings are reached for $\tau = 0$, $p = 19$ and $u = 0.9$. For $\lambda = 6$, the ADP achieves average cost savings of 1.5% and maximum cost savings of 6.6%. The maximum cost savings occur at $\tau = 1.5$, $u = 0.94$ and $p = 19$. Like for $\lambda = 3$ or $\lambda = 4$, the maximum cost savings occur when the costs of tracking always and never are closest to each other. The ADP is superior to the heuristics for almost all cases. Only for $u = 0.98$ and tracking always, the heuristic shows some benefits compared to the ADP.

For $\lambda = 5$, the runtimes of the ADP amount to 37.8 CPU days on average. For $\lambda = 6$, the runtimes of the ADP are 47.5 CPU days on average.

5.4. Value of tracking and dynamic tracking for moderate and large instances

For instances of $\lambda = 5$, we compare our approximate dynamic programming solutions for tracking always, never and dynamically. We set $M = 3 \cdot 10^8$ and $w_i = 0.001$. The average value of tracking amounts to 3.4%. The largest value is 15.3%, which is achieved for $u = 0.9$, $CR = 0.9$ and tracking cost of $\tau = 0$. The value of tracking dynamically is 0.4% on average, the largest value is 1.7%. The results are shown in Figure EC.3 in the online appendix. For $\lambda = 5$, the ADP for dynamic tracking has an average run time of 69.3 CPU days.

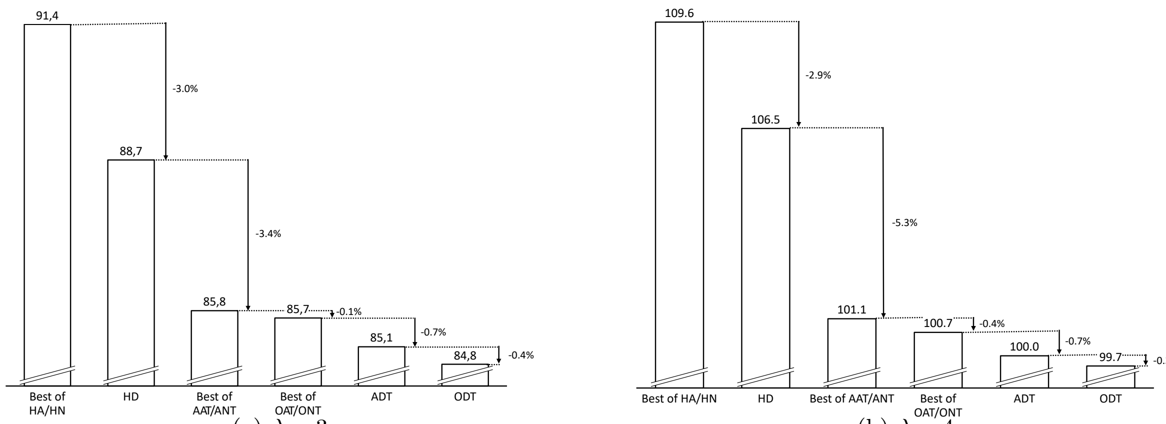
For instances of $\lambda \geq 6$, we compare the heuristic of Huh and Nagarajan (2010) with our improved dynamic heuristic. The results are shown in Figure EC.4 in the online appendix. As for smaller instances, the benefit of tracking dynamically is largest when the costs for tracking always and never coincide. We tested 9 combinations of critical ratios and penalty costs. Taking the tracking costs where the costs of tracking always and never are closest to each other, we get an average of 6.0% cost savings for each $\lambda = 6$, $\lambda = 7$ and $\lambda = 8$. The nine cost saving values spread from 3.5% to 7.8% for $\lambda = 6$, from 3.9% to 7.8% for $\lambda = 7$ and from 4.6% to 8.1% for $\lambda = 8$.

It is interesting to see that the value of tracking dynamically is lower for ADP instances than for the heuristics. This implies that tracking dynamically leads to even more substantial cost savings when order decisions are not optimal.

5.5. Summary of results

We compare the resulting average costs for all the methods we discussed, namely, optimal solution, ADP and heuristics, and for all the tracking possibilities which are: always, dynamic and never tracking. To provide an overview of the cost savings for all the methods, we compare them at the example of lead times $\lambda = 3$ and $\lambda = 4$ because it is possible to compute an optimal solution as a base case for these lead times. We give an overview of the cost savings in Figure 4. We denote the heuristic of Huh and Nagarajan (2010) for tracking always and never by HA and HN, respectively.

Figure 4 Average costs for best of always/never track heuristic, dynamic heuristic, best of ADP for always/never track, best of optimal DP for always/never track and ADP/DP for dynamic tracking



(a) $\lambda = 3$ (b) $\lambda = 4$

We label our novel dynamic heuristic HD. We abbreviate the costs for tracking always, never and dynamically under the ADP approach with AAT, ANT, and ADT, respectively. We name the costs for tracking always, never and dynamically under the optimal solution approach as OAT, ONT, and ODT, respectively.

The results show that we can achieve a solution that is very close to optimality (less than 0.4%) with ADP with dynamic tracking. The ADP solutions for tracking always or never are also close to the optimal solutions for tracking always or never, with a difference of less than 0.4% on average. Using existing heuristics results in much faster computation times, but also leads to inventory policies that are on average between 6.7% to 8.8% more costly than optimal ones that track always or never. Our novel heuristic is able to capture parts of the cost advantage by achieving a solution that incurs up to 3% less costs on average than existing heuristics.

6. Conclusion

In this paper, we analyze a stochastic demand periodic review model with backordering, positive lead times, random yield and the possibility to track always, never or dynamically under fixed tracking cost per order. We prove that the cost function satisfies the Bellman Equation and create an optimal solution algorithm based on value iteration combined with MacQueen extrapolation for small to medium sized instances. We further show the equivalence of certain states and derive lower bounds on the optimal order quantity and on the optimal cost of a state. For larger instances, we construct an approximate dynamic programming algorithm that makes extensive use of state aggregation and disaggregation. For yet larger instances, we provide a novel heuristic that makes a tracking decision based on comparing a threshold with the expected inventory position.

Companies can use one of our solution approaches to determine optimal or close-to-optimal ordering and tracking policies and to determine inventory cost savings for their particular setting.

If they currently do not track at all, they can determine what the cost savings of tracking might be and whether investing in tracking technologies like smart containers is worthwhile. If they currently track all orders, they can identify the cost savings that are associated with tracking only particular orders, for example, by using near-real time tracking services only for a subset of their orders.

In our models, we assume that time and temperature history is used to determine whether an order is salable. We assume a Bernoulli distributed yield rate per period where all or none of the ordered items spoil. Models with binomially distributed yield, where a subset of ordered items could spoil, have yet to be solved for medium to large instances. Our hierarchical approximate dynamic programming approach utilizes bounds on order sizes and the cost of a state. Developing potential bounds and adapting our approach to binomially distributed yield might be a promising subject to forthcoming studies.

Our approach uses aggregations of open orders based on the proximity of costs for a myopic time horizon. It might be interesting to investigate other metrics. Our disaggregation logic fixes the aggregated states of one aggregation level once during the whole algorithm. For future research, it might be worthwhile to check whether a feedback mechanism to disaggregate further states of already established aggregation levels could improve results.

References

Abdulwahab U, Wahab M (2014) Approximate dynamic programming modeling for a typical blood platelet bank. *Computers & Industrial Engineering* 78:259–270.

Bellman R (1957) *Dynamic Programming* (Princeton University Press).

Bertsekas DP, Castañon DA (1989) Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Aut. Control (34)* 34(6):589–598.

Bertsekas DP, Tsitsiklis JN (1996) *Neuro-dynamic programming* (Athena Scientific).

Bollapragada S, Morton TE (1999) Myopic heuristics for the random yield problem. *Operations Research* 47(5):713–722.

Choi HcP, Blocher JD, Gavirneni S (2008) Value of sharing production yield information in a serial supply chain. *Production and Operations Management* 17(6):614–625.

Dettenbach M (2015) *The value of supply chain visibility when yield is random* (Logos Verlag Berlin GmbH).

Dettenbach M, Thonemann UW (2015) The value of real time yield information in multi-stage inventory systems - Exact and heuristic approaches. *European Journal of Operational Research* 240(1):72–83.

DHL (2013) DHL ocean secure: Increasing your visibility and control. URL `https://www.dhl.com/content/dam/downloads/g0/logistics/brochures/ocean_freight/dhl_ocean_secure_nov_2013.pdf`, accessed at 03.05.2019.

Dobbs R, Oppenheim J, Thompson F, Brinkman M, Zornes M (2011) Resource revolution: Meeting the world's energy, materials, food, and water needs. *McKinsey Global Institute, McKinsey & Company* .

Ehrhardt R, Taube L (1987) An inventory model with random replenishment quantities. *International Journal of Production Research* 25(12):1795–1803.

Fang J, Zhao L, Fransoo JC, Woensel TV (2013) Sourcing strategies in supply risk management: An approximate dynamic programming approach. *Computers & Operations Research* 40(5):1371–1382.

Gaukler G, Ketzenberg M, Salin V (2017) Establishing dynamic expiration dates for perishables: An application of rfid and sensor technology. *International Journal of Production Economics* 193:617–632.

Gaukler GM, Seifert RW, Hausman WH (2007) Item-level RFID in the retail supply chain. *Production and Operations Management* 16(1):65–76.

Gavirneni S (2004) Supply chain management at a chip tester manufacturer. *The Practice of Supply Chain Management: Where Theory and Application Converge*, volume 62 of *International Series in Operations Research & Management Science*, 277–291 (Springer US).

George A, Powell WB, Kulkarni SR (2008) Value function approximation using multiple aggregation for multiattribute resource management. *Journal of Machine Learning Research* 9:2079–2111.

Gerchak Y, Vickson RG, Parlar M (1988) Periodic review production models with variable yield and uncertain demand. *IIE Transactions* 20(2):144–150.

Gosavi A (2009) Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing* 21(2):178–192.

Gustavsson J, Cederberg C, Sonesson U, Van Otterdijk R, Meybeck A (2011) *Global food losses and food waste* (FAO Rome).

Hastie T, Tibshirani R, Friedman J (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics (Springer).

Henig M, Gerchak Y (1990) The structure of periodic review policies in the presence of random yield. *Operations Research* 38(4):634–643.

Heyman DP, Sobel MJ (1982) *Stochastic models in operations research: Stochastic optimization*, volume 2 (Courier Corporation).

Huh WT, Nagarajan M (2010) Technical note – Linear inflation rules for the random yield problem: Analysis and computations. *Operations Research* 58(1):244–251.

Inderfurth K, Kiesmüller GP (2015) Exact and heuristic linear-inflation policies for an inventory model with random yield and arbitrary lead times. *European Journal of Operational Research* 245(1):109–120.

Karlin S (1958) One stage inventory models with uncertainty. K Arrow SK, Scarf H, eds., *Studies in the Mathematical Theory of Inventory and Production*, chapter 8, 109–134 (Stanford, CA: Stanford University Press).

Ketzenberg M, Bloemhof J, Gaukler G (2015) Managing perishables with time and temperature history. *Production and Operations Management* 24(1):54–70.

Ketzenberg M, Gaukler G, Salin V (2018) Expiration dates and order quantities for perishables. *European Journal of Operational Research* 266(2):569–584.

Ketzenberg ME, Rosenzweig ED, Marucheck AE, Metters RD (2007) A framework for the value of information in inventory replenishment. *European Journal of Operational Research* 182(3):1230–1250.

Ketzenberg ME, Van Der Laan E, Teunter RH (2006) Value of information in closed loop supply chains. *Production and Operations Management* 15(3):393–406.

Kiesmüller G, Inderfurth K (2018) Approaches for periodic inventory control under random production yield and fixed setup cost. *OR Spectrum* 40(2):449–477.

Kleywegt AJ, Nori VS, Savelsbergh MW (1998) A computational approach for the inventory routing problem. *Proc. Triennial Sympos. Transportation Anal.(Tristan III), San Juan, Puerto Rico* .

Lambert TJ, Epelman MA, Smith RL (2004) Aggregation in stochastic dynamic programming. Technical report, Department of Industrial and Operations Engineering, Ann Arbor, MI.

Lee H, Özer Ö (2007) Unlocking the value of RFID. *Production and Operations Management* 16(1):40–64.

Li Q, Xu H, Zheng S (2008) Periodic-review inventory systems with random yield and demand: Bounds and heuristics. *IIE Transactions* 40(4):434–444.

MacQueen J (1966) A modifed dynamic programming method for Markovian decision problems. *Journal of Mathematical Analysis and Applications* 14(I):38–43.

Mes MRK, Powell WB, Frazier PI (2011) Hierarchical knowledge gradient for sequential sampling. *Journal of Machine Learning Research* 12:2931–2974.

Ngai EWT, Moon KKL, Riggins FJ, Yi CY (2008) RFID research: An academic literature review (1995-2005) and future research directions. *International Journal of Production Economics* 112(2):510–520.

Powell WB (2011) *Approximate dynamic programming: Solving the curses of dimensionality*, volume 842 of *Wiley Series in Probability and Statistics* (John Wiley & Sons), 2nd edition.

Powell WB (2016) Perspectives of approximate dynamic programming. *Annals of Operations Research* 241(1-2):319–356.

Sarac A, Absi N, Dauzère-Pérès S (2010) A literature review on the impact of RFID technologies on supply chain management. *International Journal of Production Economics* 128(1):77–95.

Sauré A, Patrick J, Puterman ML (2015) Simulation-based approximate policy iteration with generalized logistic functions. *INFORMS Journal on Computing* 27(3):579–595.

Simão HP, Day J, George AP, Gifford T, Nienow J, Powell WB (2009) An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science* 43(2):178–197.

Singh SP, Jaakkola T, Jordan MI (1995) Reinforcement learning with soft state aggregation. *Advances in neural information processing systems*, 361–368.

Sleptchenko A, Johnson ME (2015) Maintaining secure and reliable distributed control systems. *INFORMS Journal on Computing* 27(1):103–117.

Sonntag D, Kiesmüller GP (2017) The influence of quality inspections on the optimal safety stock level. *Production and Operations Management* 26(7):1284–1298.

Van Roy B, Bertsekas DP, Lee Y, Tsitsiklis JN (1997) A neuro-dynamic programming approach to retailer inventory management. *Proceedings of the 36th IEEE Conference on Decision and Control*, volume 4, 4052–4057 (IEEE).

White CC, Cheong T (2012) In-transit perishable product inspection. *Transportation Research Part E: Logistics and Transportation Review* 48(1):310–330.

Yano CA, Lee HA (1995) Lot sizing with random yield: A review. *Operations Research* 43(2):311–334.

Zipkin PH (2000) *Foundations of inventory management* (Boston: McGraw-Hill).

Electronic Companion

EC.1. List of symbols

a_x	action in state x defined as pair $(\hat{O}_x, \hat{\psi}_x)$
α	step size for asynchronous value iteration
$\mathscr{A}$	action space of all possible actions if action space is independent of the state
$\mathscr{A}_x$	action space of all possible actions available in state x
$\bar{\beta}^{(i)}$	maximum visit frequency threshold for state aggregation of level i to be used
$c(x,a)$	one-stage cost for state x given action a
$c_{ein}(z)$	one-stage holding/penalty cost given end of period inventory level z
$c_{inv}(x)$	one-stage holding/penalty cost given state x
C^{asy}	costs-to-go of asynchronous value iteration algorithm
C^{syn}	costs-to-go of synchronous value iteration algorithm
C^{ag1}	costs-to-go for optimal action of learning process with one aggregation level
C^{agr}	costs-to-go for optimal action of learning process with multiple aggregation levels
D	demand distribution
$\Delta^{(g)}$	cost difference of state on aggregation level g towards less aggregated state
$e_k^{(g)}(x_k)$	enablement of state x_k in aggregation level g in iteration k
ϵ	small threshold value used to determine convergence
f	source aggregation level to use for aggregation functions
F_D^{-1}	inverse cumulative distribution function of demand distribution D
γ	discount factor within interval $(0,1)$
$\mathscr{G}$	index set of aggregation levels
$\mathscr{G}^+$	index set of aggregation levels greater 0
h	holding cost factor
I	inventory level if positive, backorder level if negative
IP	inventory position
ι	inflation factor for linear inflation policies
$\mathscr{I}$	space of possible inventory / backorder levels
$\mathbb{I}[\cdot]$	indicator function that returns 1 if expression in brackets is true and 0 else
$J(x)$	minimal expected cost for state x in infinite horizon setting
$J_t(x)$	minimal expected cost starting from state x from period t to T
$\underline{J}(x)$	lower bound on optimal cost for state x based on myopic cost estimate
$\underline{\underline{J}}(x)$	lower bound on optimal cost for state x based on perfect yield problem

λ	lead time
$M^{(g)}$	iterations for aggregation level g after which aggregations are enabled/disabled
μ	threshold value to determine tracking decision in dynamic heuristic
$\hat{O}$	order decision in current period
$\hat{O}^*$	optimal order decision in current period
$\underline{\hat{O}}$	lower bound on optimal order size for untracked orders
O_t	order quantity placed t time periods ago
p	penalty cost factor
$p(x)$	steady state probability for state x
$p_a(x,\tilde{x})$	transition probability from state x to transition state $\tilde{x}$ under action a
$p_a^{inv}(x,\tilde{x})$	transition probability of inventory from state x to transition state $\tilde{x}$
$p_a^{tra}(x,\tilde{x})$	transition probability of tracking decisions from state x to transition state $\tilde{x}$
$p_a^{ord}(x,\tilde{x})$	transition probability of order quantity placed i periods ago to transition state $\tilde{x}$
$p_a^{ord,unt}(x,\tilde{x},i)$	transition probability of order quantity placed i periods ago if untracked
$p_a^{ord,tra}(x,\tilde{x},i)$	transition probability of order quantity placed i periods ago if tracked
$\hat{\psi}$	tracking decision in current period
ψ_t	tracking decision for order placed t time periods ago
$q(x)$	frequency of visiting state x within asynchronous value iteration algorithm
Q_t	(random) yield of order placed t time periods ago
$r(x)$	aggregated state of highest enabled level for state x
$r^{(g)}(x)$	aggregated state of level g for state x
R	operator that chooses optimal action and updates costs accordingly
$\mathscr{S}$	state space defined as $(I, O_1, \cdots, O_\lambda, \psi_1, \cdots, \psi_\lambda)$
$\mathscr{S}^{(g)}$	state space of aggregation level g defined as $(g, I, O_1, \cdots, O_\lambda, \psi_1, \cdots, \psi_\lambda)$
T	number of periods regarded for the finite horizon cost function
τ	tracking cost
θ	order threshold value for linear inflation policies
u_t	expected value of random yield rate Y_t
$w^{(g)}$	probability mass of states to aggregate on aggregation level g
x	state with lower indices for time or iteration, upper indices for equivalent states and upper indices in parentheses for aggregated states
ξ	1 or -1 for tracking if IP is above or below threshold in dynamic heuristic
Y_t	random yield rate for lead time period t
$Z_{src}^{(f),(g)}$	myopic state cost of state on aggregation level $f < g$
$Z_{tar}^{(g)}$	myopic state cost of state on aggregation level g
$\zeta^{(g)}$	State variable replacing O_λ in aggregation level g

EC.2. Proof of Lemma 1

Say we have a stationary policy with $\hat{O}=0$ for each $x \in \mathscr{S}$. We start at some arbitrary state with inventory level I_0 and we let D_t be the cumulative demand from period 1 to t. We then have one-period cost for period t of $\mathbb{E}\Big[h[I_0-D_t]^+ - p[I_0-D_t]^-\Big] \le \mathbb{E}\Big[(h+p)|I_0-D_t|\Big] \le (h+p)\mathbb{E}\Big[|I_0|+D_t\Big] = (h+p)(|I_0|+\mu t)$ with μ being the mean of the one-period demand. Therefore, the infinite horizon cost for this policy is bounded by $\sum_{t=1}^{\infty} \gamma^{t-1}(h+p)(|I_0|+\mu t) = (h+p)\Big(\frac{|I_0|}{1-\gamma} + \frac{\mu}{(1-\gamma)^2}\Big) < \infty$.

$V_t(x)$ is convex (Dettenbach 2015, Theorem 4-1) for $\hat{\psi}=0$ and $\hat{\psi}=1$ while we do not track in the case that we do not order. With $\hat{O}$ going to infinity, the period cost and therefore the total cost will go to infinity. As a consequence, the search space of $\hat{O}$ is restricted to values with costs smaller than or equal to the cost associated with $\hat{O}=0$ which holds both for $\hat{\psi}=0$ and $\hat{\psi}=1$. $\square$

EC.3. Proof of Lemma 2

$J_t(x), \forall x \in \mathscr{S}$ converges uniformly to $J(x)$ when a) $|\gamma^n \min_{a \in \mathscr{A}_{x_n}} c(x_n,a)| \le \bar{c}_n$ and b) $\sum_{n=0}^{\infty} \bar{c}_n < \infty$ for each $n \in [t,T]$ (Heyman and Sobel 1982, Proposition A-5).

Condition (a) means that there exists an upper bound $\bar{c}_n$ for each period n from t to T that the one-period cost never exceeds. We have shown that a policy with this property exists with $\hat{O}=0$ for each $x \in \mathscr{S}$ in Lemma 1. Since we seek the optimal decision, policies with higher costs are discarded. It follows that the costs associated with the policy from Lemma 1 are satisfactory for $\bar{c}_n$.

Condition (b) is satisfied because, according to Lemma 1, there exists an undiscounted upper bound for each period n that the undiscounted one-period cost function never exceeds. It follows that we can discount both the one-period cost function and the upper bound by γ^n. The upper bound $\bar{c}_n$ then contains γ^n as a factor. $\lim_{n\to\infty} \gamma^n = 0$ because $\gamma < 1$. As a consequence, condition b) follows. $\square$

EC.4. Proof of Theorem 1

$\lim_{T\to\infty} J_t(x)$ satisfies Equation (2) if Theorem 8-14 of Heyman and Sobel (1982) is fulfilled.

According to condition (a), there exists a limit function for each x, which is the case (Dettenbach 2015, Theorem 4-2).

Condition (b) requires the reward function to be strictly non-negative, which is the case for $c(x,a)$.

Conditions (c) and (d) are merely necessary to show that $J_t(x)$ converges uniformly to $J(x)$ (Heyman and Sobel 1982, p. 419). We have shown the uniform convergence by Lemma 2 which completes the proof. $\square$

EC.5. Proof of Theorem 2

An optimal stationary policy exists both for $\hat{\psi}=0$ and $\hat{\psi}=1$ if the conditions of Theorem 8-15 of Heyman and Sobel (1982) are met. The right hand side of Equation (2), $c(x,a)+\gamma\sum_{\tilde{x}}p_a(x,\tilde{x})J(\tilde{x})$, is convex in x and a. It is convex in $I,O_1,\cdots,O_\lambda,\psi_1,\cdots,\psi_\lambda$ and the order quantity $\hat{O}$ for a fixed $\hat{\psi}$ (Dettenbach 2015, Theorem 4-1). We have shown that the conditions for Theorem 8-14 for a fixed $\hat{\psi}$ are met in the proof of Theorem 1. Convexity together with the conditions of Theorem 8-14 meet the conditions of Theorem 8-15 for a fixed $\hat{\psi}$. Therefore, for each state, there exist stationary optimal policies for $\hat{\psi}=0$ and $\hat{\psi}=1$ and the minimization merely selects the better one. It follows that an optimal stationary policy for our dynamic tracking problem exists as well and $J(x)$ is its cost function. □

EC.6. Proof of Proposition 1

Two states can be viewed as one state without loss of precision if and only if they exhibit the same one-stage costs and lead to the same set of follow-up states with equal probabilities for the same order and tracking decisions. Let $x^1=(I,O_1,\ldots,O_\lambda,\psi_1^1,\ldots,\psi_\lambda^1)$ and $x^2=(I,O_1,\ldots,O_\lambda,\psi_1^2,\ldots,\psi_\lambda^2)$, where $\psi_1^1=\psi_1^2$ for $O_i>0$ but ψ_1^1 and ψ_1^2 might differ for $O_i=0$. Clearly, for a fixed order and tracking decision, the tracking costs for x^1 and x^2 are the same and $c_{inv}(x^1)=c_{inv}(x^2)$. The equivalence of follow-up states can be shown for each O_k and ψ_k. Let $k=\lambda$, then $p_a^{inv}(\cdot)$ is the same for $\psi_\lambda=0$ and $\psi_\lambda=1$ according to Equation (3). p_a^{tra} and p_a^{ord} are unaffected by ψ_λ after Equations (4) and (5). Since neither one-period costs nor transition probabilities change with ψ_λ, the proposition follows for $k=\lambda$.

For $k<\lambda$, ψ_k has no effect on p_a^{inv} by Equation (3). Equation (5) states that ψ_k is merely transferred to the follow-up state. It follows that if state equivalence applies to $k=\lambda$, it also applies to $k<\lambda$ with respect to p_a^{tra}. According to Equation (4), the order decisions transfer deterministically to the follow-up state if $O_k=0$ for both $\psi_k=0$ and $\psi_k=1$. This completes the proof. □

EC.7. Proof of Proposition 2

As per Equations (4) and (5), transition probabilities of the order and tracking decisions do not depend on O_λ. Our definition of O_λ says that the order arrives with certainty if tracked. As a consequence, if $\psi_\lambda=1$, we can write $I^*=I+O_\lambda$ as deterministic variable and replace I with I^* and O_λ with 0 in Equations (1) and (3) without loss of generality and Proposition 2 follows. □

EC.8. Proof of Proposition 3

If we observe only λ terms of $J(x_t)$, it follows that

$$\begin{aligned}
J(x_t) &= \min_{a_t \in \mathscr{A}_{x_t}} c(x_t, a_t) + \gamma \sum_{x_{t+1} \in \mathscr{S}} p_{a_t}(x_t, x_{t+1}) J(x_{t+1}) \\
&\le \min_{a_t \in \mathscr{A}_{x_t}} c(x_t, a_t) + \gamma \sum_{x_{t+1} \in \mathscr{S}} p_{a_t}(x_t, x_{t+1}) \min_{a_{t+1} \in \mathscr{A}_{x_{t+1}}} c(x_{t+1}, a_{t+1}) \\
&+ \ldots + \gamma^{\lambda-1} \sum_{x_{t+\lambda-1} \in \mathscr{S}} p_{a_t}(x_t, x_{t+1}) \ldots p_{a_{t+\lambda-2}}(x_{t+\lambda-2}, x_{t+\lambda-1}) \min_{a_{t+\lambda-1}} c(x_{t+\lambda-1}, a_{t+\lambda-1}) \\
&\le c_{inv}(x_t) + \gamma \sum_{x_{t+1} \in \mathscr{S}} p^{inv}(x_t, x_{t+1}) c_{inv}(x_{t+1}) \\
&+ \ldots + \gamma^{\lambda-1} \sum_{x_{t+\lambda-1} \in \mathscr{S}} p^{inv}(x_t, x_{t+1}) \ldots p^{inv}(x_{t+\lambda-2}, x_{t+\lambda-1}) c_{inv}(x_{t+\lambda-1}).
\end{aligned}$$

The first inequality follows from the fact that we only view λ periods on the right-hand-side. The second inequality applies because we only view inventory costs and remove tracking costs. Since inventory costs only depend on the inventory level, we only focus on the inventory level transition probability p^{inv}, where we omit a in the notation because p_a^{inv} is for these instances independent of any action. States $x_{t+1}, \ldots, x_{t+\lambda-1}$ are therefore only known with respect to their inventory level, which suffices to calculate inventory costs. As actions placed in some period n only affect inventory costs in period $n+\lambda$, we can eliminate the minimizations in periods t to $t+\lambda-1$. The cost bound follows. □

EC.9. Deterministic period review cost

```
Require: deterministic demand $d$
 1: fix convergence rate $\epsilon$
 2: function $\underline{\underline{J}}(x = (I, O_1, \ldots, O_\lambda, \psi_1, \ldots, \psi_\lambda) \in \mathscr{S}, n = 1)$
 3:     $\hat{O} \leftarrow (\lambda+1)d - I - \sum_i O_i$
 4:     $\hat{O} \leftarrow [\min\{\hat{O}, O^{max}\}]^+$
 5:     $I \leftarrow I + O_\lambda - d$
 6:     $\Delta c \leftarrow \gamma(h[I]^+ + p[-I]^+)$
 7:     if $\Delta c < \epsilon$ and $n > \lambda + 1$ then
 8:         return $c + \Delta c$
 9:     for $i = \lambda - 1$ to 0 do
10:         $O_i \leftarrow O_{i-1}$
11:     $O_1 \leftarrow \hat{O}$
12:     return $\underline{\underline{J}}(x, n+1)$
```

EC.10. Proof of Proposition 4

Let I_λ be the the inventory level after λ periods, that is the inventory level that is affected by the order that we place now. If $\mathbb{P}(I_\lambda+1>0)=0$, we know that one additional item ordered will reduce the expected underage cost for period λ by the expected arrival of this additional item because there is no chance that inventory holding costs apply. Thus, the expected inventory cost will be decreased by $\Delta c = p\prod_{i=1}^{\lambda} u_i$. As long as Δc exceeds τ, it will be beneficial to change the order quantity from 0 to 1. Since $I_\lambda+1 \leq I_1+O_1+\ldots+O_\lambda+\hat{O}+1$, $\mathbb{P}(I_\lambda+1>0)=0$ holds for $\hat{O}\leq -I_1-O_1-\ldots-O_\lambda-1$. In particular, for $\hat{O}=0$, $\mathbb{P}(I_\lambda+1>0)=0$ holds for $I_1+O_1+\ldots+O_\lambda+1\leq 0$. If this condition is met, so that it is beneficial to order one instead of no items, the order decision is within the convex area of the cost function. Then, Theorem 7 of Henig and Gerchak (1990) applies and we should order according to $\hat{O}\geq \underline{\hat{O}}(x)$. $\square$

EC.11. Proof of Proposition 5

Applying the optimal stationary policy, I has a certain range. If I is sufficiently small, an optimal policy will lead to a positive order quantity according to Theorem 7 of Henig and Gerchak (1990) or Proposition 4. This order quantity will arrive after λ periods with probability greater zero due to $\mathbb{P}(\hat{O}\prod_{i=1}^{\lambda} u_i>0)>0$, so that it is possible to increase the inventory level no matter how low I is. On the other hand, there clearly exists an inventory level that is high enough so that the optimal policy is to order nothing. Since $\mathbb{P}(D>0)>0$, it is also possible to reach states with lower inventory levels. Thus, we have shown that states within a range of inventory levels communicate with one another and that it is not possible to split this range up to create more than two communicating classes.

There could still exist more than one essential class with respect to order decision and tracking decision state variables. We disprove this option by showing next that for any two essential states with the same inventory level $x_1=(I,O_1^1,\ldots,O_\lambda^1,\psi_1^1,\ldots,\psi_\lambda^1), x_2=(I,O_1^2,\ldots,O_\lambda^2,\psi_1^2,\ldots,\psi_\lambda^2)$, it is always possible to find a trajectory connecting these two states.

We evince that x_1 and x_2 communicate with each other starting with $\psi_1=\ldots=\psi_\lambda=0$ by induction on λ. We start with $\lambda=1$. Let $x_1=(I,O_1^1,0)$ and $x_2=(I,O_1^2,0)$. Then, with probability greater zero, $x_3=(I-d_1,O_0^1,0)$ is a direct follow-up state of x_1 and $x_4=(I-d_2,O_0^2,0)$ is a direct follow-up state of x_2 because yields might be 0 and d_1 and d_2 are arbitrary demand values. It is now easy to see that there exists at least one combination of d_1 and d_2 such that the same order decision, $\hat{O}$, will be made. Then, with possibility greater zero $x_5=(I-n,\hat{O},0)$ is a common direct follow-up state of both x_3 and x_4 for some n. Since x_1 and x_2 are essential by definition and lead to x_5, x_5 is also essential and, thus, x_5 communicates both with x_1 and x_2. As communication is transitive, x_1 and x_2 communicate, too.

For $\lambda = 2$, let $x_1 = (I, O_1^1, O_2^1, 0, 0)$ and $x_2 = (I, O_1^2, O_2^2, 0, 0)$. Applying the same logic as in the last paragraph, the follow-up states of x_1 and x_2 could be $x_3 = (I - d_1, O_0^1, O_1^1, 0, 0)$ and $x_4 = (I - d_2, O_0^2, O_1^2, 0, 0)$, respectively, that lead to the same decision $\hat{O}$. Then, the follow-up states of x_3 and x_4 could be $x_5 = (I - d_1 - d_3, \hat{O}, O_0^1, 0, 0)$ and $x_6 = (I - d_2 - d_4, \hat{O}, O_0^2, 0, 0)$. Since the second state variable is the same for both x_5 and x_6, we can now apply the same logic as for $\lambda = 1$ and find a common follow-up state. For $\lambda > 2$, we can apply the same process and always find for two essential states x_1 and x_2 common follow-up states, so that communication is shown.

For tracked orders, we utilize a similar process: If O_λ is tracked, Proposition 2 says that the order can directly be added to the inventory level. Then, our reasoning for communicating states with respect to inventory levels applies.

If an earlier order is tracked, we can apply the same logic as for untracked orders. The only difference is that orders may spoil, leading to 0 for the order quantity state variable in the follow-up state. Then, it follows straightforwardly that a common follow-up state exists because, no matter the order quantity, a tracked order of any size can spoil. This completes the proof and Proposition 5 follows. □

EC.12. Approximate value iteration enhanced by structural findings

The following function describes the algorithm with parameter $\bar{\psi}$ that describes the possible tracking options (track always, track never, track dynamically). C^* describes the optimal average costs-to-go for the entire state space and is updated each E iterations if a better policy via policy evaluation is found. P describes the function for policy evaluation. It yields the average costs C given the actions $\hat{O}(x_k), \hat{\psi}(x_k)$ for all states x_k. We denote the policies for all states as $\hat{O}$ and $\hat{\psi}$. The optimal policy is called (O^*, ψ^*) and returned.

```
1: **function** ADP($\bar{\psi} \in \{\{0\},\{1\},\{0,1\}\}$)
2:     $C^{asy}(x) \leftarrow \max\{\underline{J}(x), \underline{\underline{J}}(x)\}, \forall x \in \mathscr{S}$
3:     Set $k = 1$, $x_1 = (0, \cdots, 0)$, $C^* = +\infty$
4:     **do**
5:         Set $O_{min}(\psi) = \underline{\hat{O}}(x_k)$ for $\psi = 0$ and for $\psi = 1$ with $\tau > p\prod_{i=1}^{\lambda} u_i$ and with $I + \sum_{i=1}^{\lambda} O_i + 1 \leq 0$, or 0 else
6:         $$(\hat{O}(x_k), \hat{\psi}(x_k)) \leftarrow \min_{\hat{\psi} \in \bar{\psi}, \hat{O} \geq O_{min}(\hat{\psi})} \tau\hat{\psi} + \gamma \sum_{x_{k+1}} p_{\hat{O},\hat{\psi}}(x_k, x_{k+1}) C^{asy}(x_{k+1})$$
7:         $$C^{asy}(x_k) \leftarrow c_{inv}(x_k) + \tau\hat{\psi} + \gamma \sum_{x_{k+1}} p_{\hat{O},\hat{\psi}}(x_k, x_{k+1}) C^{asy}(x_{k+1})$$
8:         **if** $k \ mod \ E = 0$ **then**
```

```
 9:            $C \leftarrow P((\hat{O},\hat{\psi}))$
10:            **if** $C < C^*$ **then**
11:                $C^* \leftarrow C$
12:                $(O^*,\psi^*) \leftarrow (\hat{O},\hat{\psi})$
13:        Sample $\tilde{d}$ from $D$
14:        $O_0 \leftarrow \hat{O}, \psi_0 \leftarrow \hat{\psi}$
15:        **for** $i=0$ to $\lambda-1$ **do**
16:            **if** $\psi_i = 1$ **then**
17:                Sample $\tilde{O}_{i+1}$ with $\mathbb{E}[\tilde{O}_{i+1}] = u_i O_i$
18:            **else**
19:                $\tilde{O}_{i+1} \leftarrow O_i$
20:        **if** $\psi_\lambda = 1$ **then**
21:            $\tilde{O}_{\lambda+1} \leftarrow O_\lambda$
22:        **else**
23:            Sample $\tilde{O}_{\lambda+1}$ with $\mathbb{E}[\tilde{O}_{\lambda+1}] = O_\lambda \prod_{i=1}^{\lambda} u_i$
24:        $x_{k+1} \leftarrow (I - \tilde{d} + \tilde{O}_{\lambda+1}, \tilde{O}_1, \ldots, \tilde{O}_\lambda, \psi_0, \ldots, \psi_{\lambda-1})$
25:        $k \leftarrow k+1$
26:    **while** $k \leq K$
27:    **if** $\bar{\psi} = \{0,1\}$ and $\min\{P(ADP(\{0\})), P(ADP(\{1\}))\} < P((O^*,\psi^*))$ **then**
28:        **return** $\arg\min_{ADP(\cdot)}\{P(ADP(\{0\})), P(ADP(\{1\}))\}$
29:    **else**
30:        **return** $(O^*,\psi^*)$
```

EC.13. Proof of Proposition 6

$c_{ein}(\zeta)$ is convex in ζ because it is linearly decreasing for $\zeta \leq 0$ and linearly increasing for $\zeta \geq 0$. Then, $\mathbb{E}[c_{ein}(I^{(g)} - \sum_{l=1}^{k} D_l + \mathbb{I}[\psi_\lambda^{(g)} = 0](\zeta \sum_{n=1}^{\lambda} Y_n) + \mathbb{I}[\psi_\lambda^{(g)} = 1]\zeta)]$ is also convex in ζ (Heyman and Sobel 1982, Proposition B-2). Since γ^n is positive for all $n \in \mathbb{R}$ and the sum of convex functions is convex, $Z_{tar}^{(g)}$ is convex. $\square$

EC.14. Hierarchical approximate dynamic programming algorithm

Variables for the following algorithm have the same meaning as in Appendix EC.12 if not stated otherwise.

```
1: **function** HADP($\bar{\psi} \in \{\{0\},\{1\},\{0,1\}\}$)
2:     $C^{agr}(x) \leftarrow \max\{\underline{J}(x), \underline{\underline{J}}(x)\}, \forall x \in \mathscr{S}^{(0)}$
3:     $\beta_x \leftarrow 0, \forall g \in \mathscr{G}^+, x \in \mathscr{S}^{(g)}$
```

```
4:    Set $k=1$, $x_1=(0,\cdots,0)$, $C^*=+\infty$
5:    $e_0^{(g)}(x) \leftarrow \mathbb{I}[g=|\mathscr{G}|-1], \forall g \in \mathscr{G}^+, \forall x \in \mathscr{S}^{(g)}$
6:    do
7:        Set $O_{min}(\psi)=\underline{\hat{O}}(x_k)$ for $\psi=0$ and for $\psi=1$ with $\tau > p\prod_{i=1}^{\lambda} u_i$ and with $I+\sum_{i=1}^{\lambda} O_i+1 \le 0$, or 0 else
8:        $$(\hat{O},\hat{\psi}) \leftarrow \min_{\hat{\psi}\in\bar{\psi},\hat{O}\ge O_{min}(\hat{\psi})} \tau\hat{\psi}+\gamma\sum_{\tilde{x_k}} p_{\hat{O},\hat{\psi}}(x_k,\tilde{x}_k)C^{agr}(r(\tilde{x}_k))$$
9:        $a_k=(\hat{O},\hat{\psi})$
10:       $C^{agr}(r(x_k)) \leftarrow \alpha\big[c_{inv}(x_k)+\tau\hat{\psi}+\gamma\sum_{\tilde{x}_k} p_{\hat{O},\hat{\psi}}(x_k,\tilde{x}_k)C^{agr}(r(\tilde{x}_k))\big]+(1-\alpha)C^{agr}(r(x_k))$
11:       if $k$ mod $E=0$ then
12:           $C \leftarrow P((\hat{O},\hat{\psi}))$
13:           if $C<C^*$ then
14:               $C^* \leftarrow C$
15:               $(O^*,\psi^*) \leftarrow (\hat{O},\hat{\psi})$
16:       for $i=1$ to $|\mathscr{G}^+|$ do
17:           if $k=M^{(i)}$ then
18:               $\bar{\beta}^{(i)} \leftarrow \max\beta$, s.t. $\sum_{x\in\mathscr{S}^{(i)}} \mathbb{I}[q(x)\le\beta]q(x)/(\sum_{y\in\mathscr{S}^{(i)}} q(y)) \le w^{(i)}$
19:               $S^{(i)} \leftarrow \{\, x\in\mathscr{S}^{(i)} \mid q(x)\le\bar{\beta}^{(i)} \,\}$
20:               $e^{(i)}(x) \leftarrow \mathbb{I}[x\in S^{(i)}], \forall x\in\mathscr{S}^{(i)}$
21:       Sample $\tilde{d}$ from $D$
22:       $O_0 \leftarrow \hat{O}, \psi_0 \leftarrow \hat{\psi}$
23:       for $i=0$ to $\lambda-1$ do
24:           if $\psi_i=1$ then
25:               Sample $\tilde{O}_{i+1}$ with $\mathbb{E}[\tilde{O}_{i+1}]=u_iO_i$
26:           else
27:               $\tilde{O}_{i+1} \leftarrow O_i$
28:       if $\psi_\lambda=1$ then
29:           $\tilde{O}_{\lambda+1} \leftarrow O_\lambda$
30:       else
31:           Sample $\tilde{O}_{\lambda+1}$ with $\mathbb{E}[\tilde{O}_{\lambda+1}]=O_\lambda\prod_{i=1}^{\lambda} u_i$
32:       $x_{k+1} \leftarrow (I-\tilde{d}+\tilde{O}_{\lambda+1},\tilde{O}_1,\ldots,\tilde{O}_\lambda,\psi_0,\ldots,\psi_{\lambda-1})$
33:       $k \leftarrow k+1$
34:   while $k\le K$
```

```
35:     **if** $\bar{\psi} = \{0,1\}$ and $\min\{P(HADP(\{0\})), P(HADP(\{1\}))\} < P((O^*, \psi^*))$ **then**
36:         **return** $\arg\min_{HADP(\cdot)}\{P(HADP(\{0\})), P(HADP(\{1\}))\}$
37:     **else**
38:         **return** $(O^*, \psi^*)$
```

EC.15. Tables

Table EC.1 Optimal costs, costs from ADP approach and error for never/always tracking

			$\lambda = 2$						$\lambda = 3$						$\lambda = 4$					
			Never Track			Always Track			Never Track			Always Track			Never Track			Always Track		
u	τ	CR	ADP	DP	Error (%)	ADP	DP	Error (%)	ADP	DP	Error (%)	ADP	DP	Error (%)	ADP	DP	Error (%)	ADP	DP	Error (%)
0.9	0	0.9	62.1	62.1	0.0	56.0	56.0	0.0	79.2	79.1	0.1	70.3	70.3	0.1	96.8	96.5	0.3	85.0	84.8	0.2
		0.95	75.5	75.5	0.1	68.6	68.5	0.1	96.0	95.8	0.1	85.8	85.7	0.1	117.2	116.9	0.3	103.7	103.3	0.3
		0.99	104.2	104.1	0.1	95.7	95.5	0.2	132.1	131.5	0.4	119.6	119.4	0.2	161.8	159.8	1.2	144.2	143.8	0.3
	1	0.9	62.1	62.1	0.0	63.9	63.9	0.0	79.2	79.1	0.1	78.5	78.5	0.1	96.8	96.5	0.3	93.8	93.6	0.2
		0.95	75.5	75.5	0.1	76.3	76.3	0.0	96.0	95.8	0.1	94.0	94.0	0.1	117.2	116.9	0.3	112.2	112.1	0.1
		0.99	104.2	104.1	0.1	103.5	103.4	0.1	132.1	131.5	0.4	128.1	128.0	0.1	161.8	159.8	1.2	152.7	152.5	0.2
	2	0.9	62.1	62.1	0.0	70.7	70.6	0.0	79.2	79.1	0.1	85.6	85.5	0.1	96.8	96.5	0.3	101.2	101.0	0.2
		0.95	75.5	75.5	0.1	82.7	82.6	0.1	96.0	95.8	0.1	100.8	100.7	0.1	117.2	116.9	0.3	119.9	119.9	0.0
		0.99	104.2	104.1	0.1	110.7	110.6	0.1	132.1	131.5	0.4	134.7	134.6	0.1	161.8	159.8	1.2	160.4	160.4	0.0
0.94	0	0.9	55.5	55.4	0.1	51.1	51.1	0.0	68.2	68.1	0.1	62.0	62.0	0.1	80.7	80.5	0.3	72.6	72.4	0.2
		0.95	67.1	67.1	0.1	61.9	61.9	0.0	82.3	82.2	0.1	75.0	75.0	0.0	97.3	97.0	0.3	87.7	87.5	0.2
		0.99	91.8	91.6	0.2	85.2	85.1	0.0	112.0	111.7	0.2	102.7	102.6	0.1	132.1	131.6	0.4	119.8	119.5	0.2
	1	0.9	55.5	55.4	0.1	58.8	58.7	0.0	68.2	68.1	0.1	70.2	70.1	0.0	80.7	80.5	0.3	81.1	80.9	0.2
		0.95	67.1	67.1	0.1	69.8	69.8	0.0	82.3	82.2	0.1	83.1	83.1	0.0	97.3	97.0	0.3	96.2	96.1	0.1
		0.99	91.8	91.6	0.2	93.2	93.1	0.1	112.0	111.7	0.2	111.0	110.9	0.1	132.1	131.6	0.4	128.3	128.0	0.2
	2	0.9	55.5	55.4	0.1	64.8	64.8	0.0	68.2	68.1	0.1	76.4	76.4	0.0	80.7	80.5	0.3	88.0	88.0	0.1
		0.95	67.1	67.1	0.1	76.4	76.4	0.0	82.3	82.2	0.1	89.9	89.9	0.0	97.3	97.0	0.3	103.4	103.3	0.1
		0.99	91.8	91.6	0.2	99.6	99.5	0.1	112.0	111.7	0.2	118.0	117.9	0.1	132.1	131.6	0.4	135.5	135.4	0.1
0.98	0	0.9	48.8	48.8	0.0	47.2	47.1	0.0	57.5	57.5	0.1	54.8	54.7	0.1	66.1	65.6	0.7	62.4	62.1	0.4
		0.95	58.5	58.5	0.0	56.6	56.6	0.0	69.2	69.0	0.4	65.8	65.8	0.1	79.3	78.6	0.8	74.6	74.3	0.4
		0.99	78.6	78.5	0.1	75.8	75.7	0.0	92.9	92.5	0.4	88.2	88.1	0.1	106.9	105.3	1.6	99.8	99.5	0.3
	1	0.9	48.8	48.8	0.0	54.5	54.5	0.0	57.5	57.5	0.1	62.4	62.4	0.0	66.1	65.6	0.7	70.3	70.1	0.2
		0.95	58.5	58.5	0.0	64.0	64.0	0.0	69.2	69.0	0.4	73.7	73.7	0.0	79.3	78.6	0.8	82.5	82.4	0.1
		0.99	78.6	78.5	0.1	83.0	83.0	0.0	92.9	92.5	0.4	96.0	95.9	0.1	106.9	105.3	1.6	107.8	107.7	0.1
	2	0.9	48.8	48.8	0.0	59.7	59.7	0.0	57.5	57.5	0.1	68.4	68.3	0.1	66.1	65.6	0.7	76.1	76.0	0.1
		0.95	58.5	58.5	0.0	69.8	69.8	0.0	69.2	69.0	0.4	79.5	79.4	0.1	79.3	78.6	0.8	89.1	89.0	0.1
		0.99	78.6	78.5	0.1	89.9	89.9	0.1	92.9	92.5	0.4	103.0	103.0	0.0	106.9	105.3	1.6	114.8	114.8	0.1
Average					0.1			0.0			0.2			0.1			0.7			0.2

EC.16. Figures

Figure EC.1 Optimal cost developments for rising tracking costs for $\lambda = 3$ and $\lambda = 4$

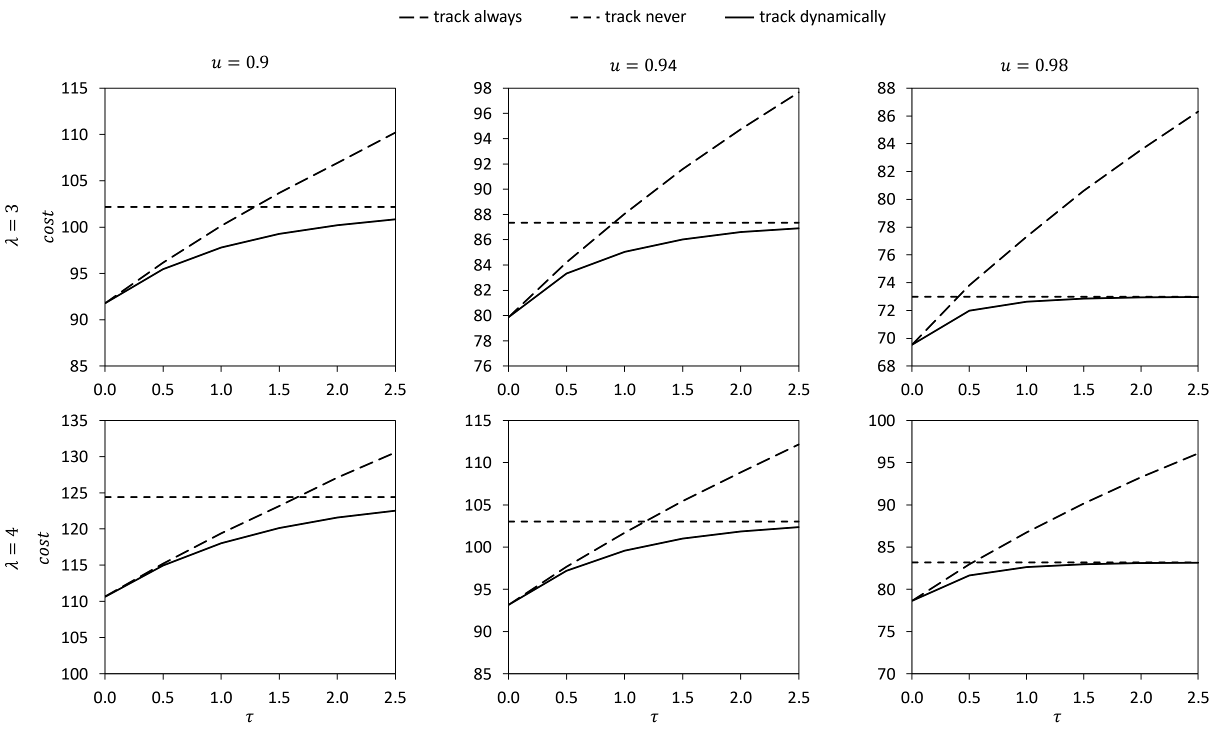


Figure EC.2 Benefit of ADP always/never track compared to Huh & Nagaran heuristic with always/never track

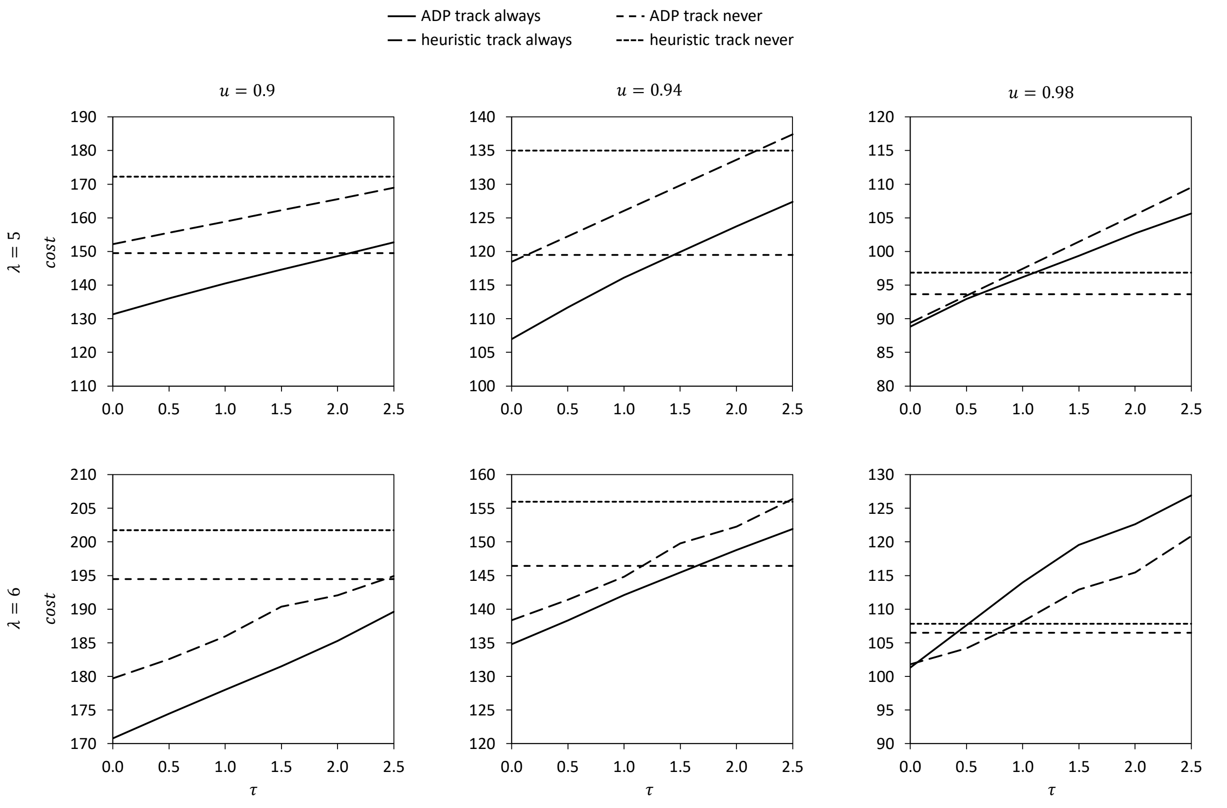


Figure EC.3 ADP cost developments for rising tracking costs for $\lambda = 5$

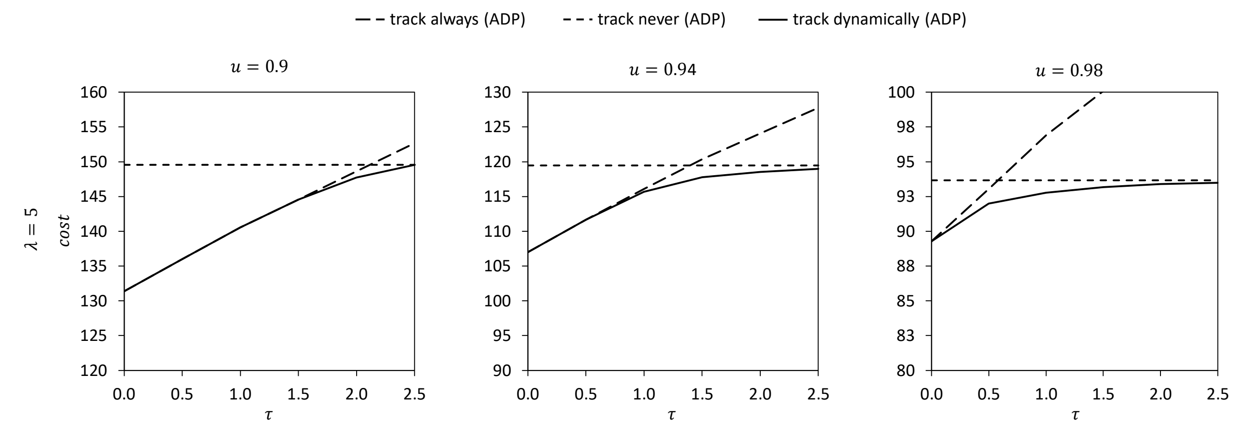
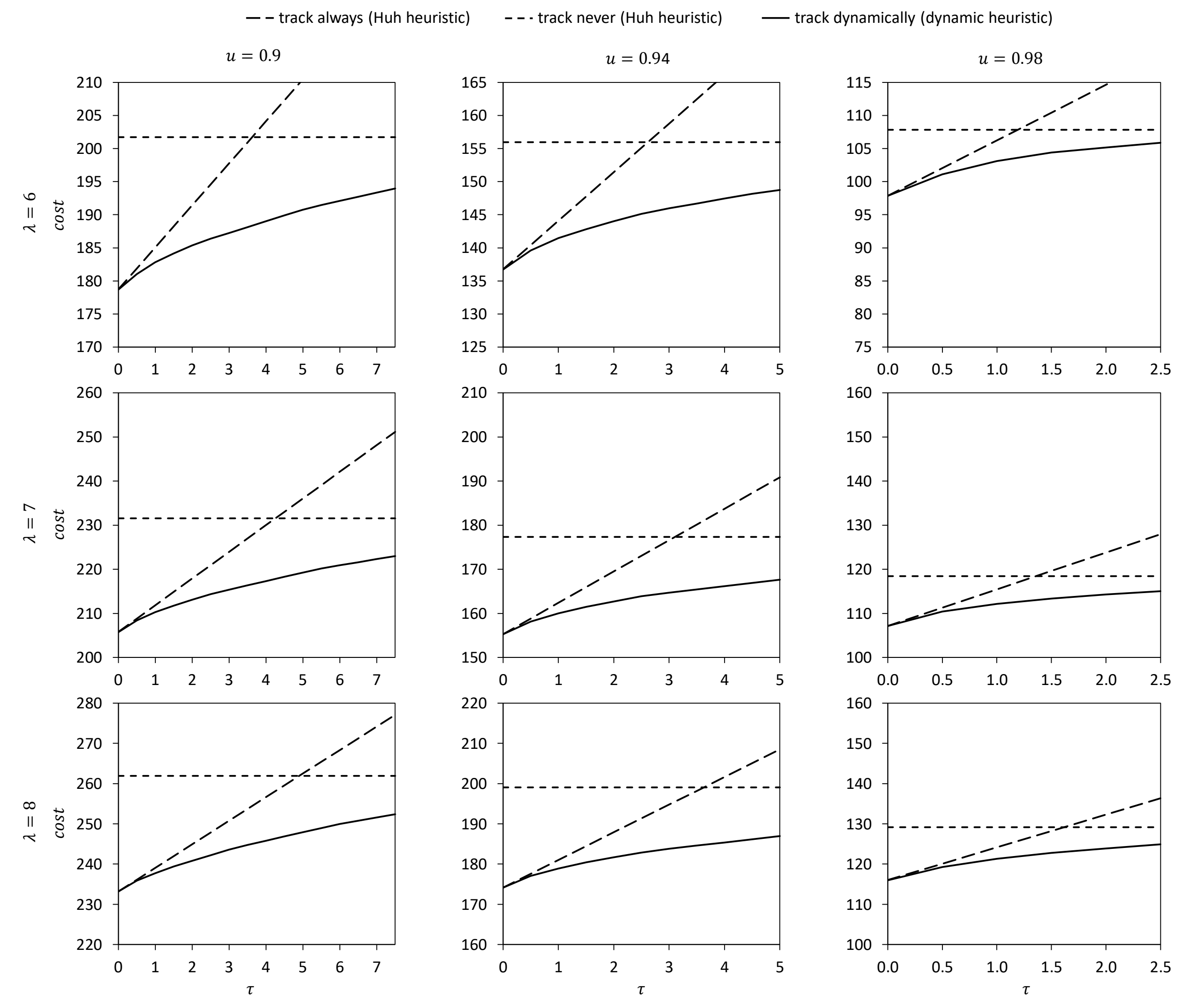


Figure EC.4 Cost developments of dynamic heuristic compared to Huh & Nagaran's heuristic for rising tracking costs for $\lambda = 6$, $\lambda = 7$ and $\lambda = 8$ averaged over critical ratios


track always (Huh heuristic)
track never (Huh heuristic)
track dynamically (dynamic heuristic)
$u = 0.9$
$u = 0.94$
$u = 0.98$
$\lambda = 6$
$\lambda = 7$
$\lambda = 8$
cost
τ