doi:10.1016/j.ejor.2022.02.005 • www.stefancreemers.be • info@stefancreemers.be

The joint replenishment problem: Optimal policy and exact evaluation method

Stefan Creemers
Robert Boute

Abstract - **We propose a new method to evaluate any stationary joint replenishment policy under compound Poisson demand. The method makes use of an embedded Markov chain that only considers the state of the system after an order is placed. The resulting state space reduction allows exact analysis of instances that until now could only be evaluated using approximation procedures. In addition, the size of the state space is not affected if we include nonzero lead times, backlog, and lost sales. We characterize the optimal joint replenishment policy, and use these characteristics to develop a greedy-optimal algorithm that generalizes the can-order policy, a well-known family in the class of joint replenishment policies. We numerically show that this generalized can-order policy only marginally improves the best conventional can-order policy. For sizeable systems with multiple items, the latter can now be found using our exact embedded Markov-chain method. Finally, we use our method to improve and extend the well-known decomposition approach.**

Keywords - **inventory, joint replenishment, can-order policy, embedded Markov chain**

1 Introduction and Literature Review

The Joint Replenishment Problem (JRP) arises in environments where a group of items share a common resource, like a production line or transportation mode. In this context, there is an opportunity to exploit economies of scale by coordinating the replenishments for the different items, and to share the fixed cost associated with a replenishment. In a manufacturing setting, this fixed cost may be a setup cost in a facility that produces multiple items. In a distribution setting where multiple items can be shipped together, an order cost per shipment is incurred, independent of the number of different items in the shipment, and there are additional item-dependent costs for picking and processing (if items are delivered to geographically separated locations, the minor order cost may also reflect the cost of last-mile delivery). An order cannot be placed unless the *major* setup or order cost is also incurred. Once the major cost is incurred, any item can be ordered by simply incurring its *minor* setup or order cost. In most cases, the major and minor costs are large enough to benefit from consolidation of some, but not all, orders. Joint replenishment is relevant in retail operations, where multiple products are jointly replenishment using the same transport mode. Joint replenishment recently also gained momentum in the context of co-loading or collaborative shipping, where sharing transportation capacity between companies has shown large potential to increase truck fill rates and reduce CO_2 emissions. By bundling shipments with other partners or backhauling empty truckloads, available space in truck hauls for one company can be used to transport shipments for other companies (Creemers et al., 2017). This requires synchronisation of orders, which is in turn facilitated by a joint replenishment policy (Padilla Tinoco et al., 2017).

The goal of the JRP is to identify an order policy that minimizes inventory and order costs. Unfortunately, the optimal policy has no simple form: Ignall (1969) showed that the order quantity of the item triggering the order depends on the other item's inventory positions. The optimal joint replenishment policy can, theoretically, be found by solving a Markov-decision model. As the size of the state and the action space grow exponentially with the number of different items, however, it is impossible to solve the model for more than a few items. Even for two items, the optimal policy can only be found for problems of limited size.

The intractability of the JRP inspired the development of various heuristic policies (see e.g., Goyal and Satir (1998) and Khouja and Goyal (2008) for an overview). A well-performing class of joint replenishment policies for stochastic demand is the family of *can-order policies* (Balintfy 1964). Under the regime of a can-order policy, each item i is controlled by three parameters $s_i < c_i \leq S_i$. When item i's inventory position reaches its re-order level s_i, a replenishment order is triggered. At that time, any item $j \neq i$, with inventory position at or below its can-order level c_j, is also included in the order. Any order placed raises the inventory position up to its respective base-stock level S_i or S_j. Although not optimal, can-order policies retain the advantages and flexibility of individual ordering, yet may reduce order costs by permitting joint ordering.

The control mechanism of the can-order policy is rather simple, but the interaction between the items makes finding the optimal can-order policy parameters intractable for more than a few items when using traditional Markov-chain analysis. Several methods have been suggested for computing "good" can-order policies, the most famous being the decomposition approach. This method decomposes the N-item problem into N independent single item problems with normal (at full order cost) and discounted (at only minor order cost) replenishment opportunities. Discounted opportunities for item i occur when item $j \neq i$ reaches its re-order point and places an order. Assuming that the discount opportunity process for item i is Poisson and independent of item i (as in Silver (1974) and Federgruen et al. (1984)), Zheng (1994) shows that the can-order policy is indeed optimal for item i, but it is not necessarily the cost-minimizing policy across all items (Ignall 1969).

The assumption of exponential times between discounted replenishment opportunities is only reasonable, however, if the number of items N is large, and the ratio of major to minor order cost is low; in those cases the individual replenishments get more sparse. Moreover, as the decomposition approach makes use of an approximate cost function to optimize the parameters of the can-order policy, its cost performance, evaluated by simulation, tends to be worse than the best can-order policy (van Eijs 1994). Schultz and Johansen (1999) improve by showing that the interarrival times of discounts fit an Erlang, rather than an exponential, distribution. Melchiors (2002) suggests to compensate the item triggering the order for the other items joining the order opportunity. Their resulting can-order policies perform slightly better.

Comparative studies have pointed to the poor performance of the can-order policies, compared to periodic joint replenishment policies (Atkins and Iyogun (1988), Pantumsinchai (1992), and Viswanathan (1997, 2007)) when the major set-up cost is high. This has been contested, however, as this is due to the poor performance of the approximate cost evaluation method of the decomposition approach rather than due to the can-order policy itself (van Eijs 1994). Schultz and Johansen (1999) and Melchiors (2002) also show that can-order policies still outperform periodic policies for a large instance of 12 items introduced by Atkins and Iyogun (1988).

Padilla Tinoco et al. (2017) characterize the replenishment cycle under a can-order policy by a Markov process. This allows an exact cost evaluation. Assuming zero lead times and two items, the infinitesimal generator of the corresponding continuous-time Markov chain has state space dimension $S_1S_2 \times S_1S_2$, with S_1 and S_2 the respective base-stock levels of both items. For multiple items and large values of S_i, it becomes computationally intractable to determine the steady-state distribution of this Markov chain, and a traditional Markov-chain analysis is elusive. Kayış et al. (2008) model the inventory system as a semi-Markov decision process, and propose a simple enumeration algorithm to determine the optimal parameters of the can-order policy. By restricting their attention to only two items, the problem is still tractable. Using numerical examples, they quantify the errors introduced by the decomposition method. Feng et al. (2015) consider the JRP with correlated demand. They model the problem as a Markov decision process and determine the optimal policy using policy iteration for small problems with a few items. In order to capture the correlated demand, they propose an extension of the can-order policy; the (s, c, d, S) policy, where d_i is the smallest order-up-to level of item i if it is included in the order. They propose a linear interpolation to determine the order-up-to level in $[d_i, S_i]$ that is used if a given set of items joins the order. As such, depending on the set of items that join the order, the (s, c, d, S) policy allows to order less than the order-up-to level.

This article contributes to the JRP literature as follows. We present a new embedded Markov-chain model to evaluate any stationary joint replenishment policy under compound Poisson demand. The method allows to evaluate sizeable instances due to the substantial reduction of the state space when compared to a traditional Markov-chain approach. Whereas the current state of the art in the literature restricts the exact analysis to small problems with small maximum inventory, our method allows a numerical analysis for sizeable instances with multiple items. Moreover, our method can incorporate backlog, lost sales, compound demands, and nonzero lead times without increasing the dimensions of the state space. This contrasts to a traditional Markov-chain analysis, where the size of the state space grows rapidly with any of these model extensions. Our exact and efficient solution procedure is a unique methodological contribution that is novel to the inventory management (and by extension the joint replenishment) literature. Alternative exact methods rely on conventional Markov-chain analysis to evaluate inventory policies, combined by a search procedure to optimize the policy parameters, or relies on Markov decision processes to identify the optimal policy. These solution methods are constrained as they can only be used for small-scale problems due to the triple curse of dimensionality. Thanks to the substantial reduction in state space, our method allows an exact analysis for instances that until now could only be evaluated using approximation procedures.

We also characterize the optimal policy, and use our findings to develop a greedy-optimal algorithm that generalizes the can-order policy. Using a set of numerical examples, we show that the generalized can-order policy is only marginally better than the best conventional can-order policy. This implies that the conventional can-order policy, when implemented with its optimal parameter set, performs very close to the optimal solution. Using our embedded Markov-chain method, this optimal parameter set can now be found for sizeable instances with multiple items.

We finally use our method to improve and extend the conventional decomposition approach that is used to optimize the can-order policy for multiple items. The conventional decomposition approach decomposes the N-item problem into N single-item problems that are solved approximately. Our method allows the exact evaluation of single-item problems, as well as the exact analysis of multi-item problems. We show how the extension of the conventional decomposition approach into smaller multi-item problems produces can-order policies that are closer to the best can-order policy.

2 Problem description and formulation

We consider a multi-item environment where each item $i : i \in \mathbf{N} = \{1, 2, \ldots, N\}$ faces a Poisson demand process with rate parameter λ_i. A demand for item i may trigger a replenishment order of item i at cost $K + k_i$. If an order is placed, all items $j \in \mathbf{N} \setminus \{i\}$ may also be included in the same replenishment at minor order cost k_j. Per unit inventory, a holding cost h_i is incurred per unit of time for all $i \in \mathbf{N}$. For now, we assume orders are instantaneously received in inventory and inventories are non-negative. We refer to this setting as the base case. Extensions to the base case, including backlog, lost sales, compound demands, and nonzero lead times, are outlined in Section 7.

Key to our method is the definition of two sets of states: (1) a set of X *trigger* states $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_X\}$, that define the inventory positions of each item at the instant a new replenishment order is placed or *triggered*, and (2) a set of Y *initial* states $\mathcal{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_Y\}$, that define the inventory positions of each item after an order is placed. Trigger state $x \in \{1, 2, ..., X\}$ is defined by the inventory position of each item i in trigger state x, $\mathbf{X}_x = \{I_{x1}, I_{x2}, \ldots, I_{xN}\}$. Initial state $y \in \{1, 2, ..., Y\}$ is defined by the inventory position of each item i in initial state y, $\mathbf{Y}_y = \{I_{y1}, I_{y2}, \ldots, I_{yN}\}$. We set up a Discrete-Time Markov Chain (DTMC) that only considers transitions between initial states.

A policy $\mathcal{P}$ defines the order decisions in each of the trigger states. More formally, policy $\mathcal{P}$ may be seen as a mapping function $\mathcal{P} : \mathcal{X} \mapsto \mathcal{Y}$ that maps each trigger state in $\mathcal{X}$ to an initial state in $\mathcal{Y}$. In short, we denote $\mathcal{P}(x) = y$. Any admissible replenishment policy can be defined using $\mathcal{P}$. For instance, a can-order policy with re-order levels $\mathbf{s} = \{s_1, s_2, \ldots, s_N\}$, can-order levels $\mathbf{c} = \{c_1, c_2, \ldots, c_N\}$, and order-up-to levels $\mathbf{S} = \{S_1, S_2, \ldots, S_N\}$ is defined by:

$$\begin{aligned} I_{\mathcal{P}(x)i} &= S_i \quad \text{if } I_{xi} \leq c_i, \quad \forall i \in \mathbf{N}, x \in \mathcal{X}, \\ I_{\mathcal{P}(x)i} &= I_{xi} \quad \text{otherwise}, \quad \forall i \in \mathbf{N}, x \in \mathcal{X}, \end{aligned}$$

with $\mathcal{X}$ the set of all trigger states for which the inventory of one item $i : i \in \mathbf{N}$ has inventory position $I_{xi} = s_i$, and all other items $j \neq i$ have inventory position $I_{xj} > s_j$.

Example: *We illustrate how our method evaluates a can-order policy, and compare our method with the traditional Markov-chain approach. We note that* any *stationary policy can be modelled using this method. The motivation behind the example is its simplicity, while at the same time being capable to illustrate a number of learning points. We consider two items with Poisson demand and rate parameters $\lambda_1 = 12$ and $\lambda_2 = 16$, and cost parameters*

$h_1 = 12$, $h_2 = 23$, $K = 25$, $k_1 = 7$, and $k_2 = 21$. Assuming zero lead time, an order is triggered whenever an item depletes its inventory. We evaluate the costs of the can-order policy with $S_1 = 7$, $S_2 = 8$, $c_1 = 4$, $c_2 = 2$, and $s_1 = s_2 = 0$. Policy $\mathcal{P}$ is thus defined by:

$$\begin{aligned}
I_{\mathcal{P}(x)1} &= 7 && \textit{if } I_{x1} \leq 4, \quad \forall x \in \mathcal{X},\\
I_{\mathcal{P}(x)1} &= I_{x1} && \textit{if } I_{x1} > 4, \quad \forall x \in \mathcal{X},\\
I_{\mathcal{P}(x)2} &= 8 && \textit{if } I_{x2} \leq 2, \quad \forall x \in \mathcal{X},\\
I_{\mathcal{P}(x)2} &= I_{x2} && \textit{if } I_{x2} > 2 \quad \forall x \in \mathcal{X},
\end{aligned}$$

where $\mathcal{X}$ has 15 trigger states: $\mathbf{X}_1 = \{0, 8\}$, $\mathbf{X}_2 = \{0, 7\}$, $\mathbf{X}_3 = \{0, 6\}$, $\mathbf{X}_4 = \{0, 5\}$, $\mathbf{X}_5 = \{0, 4\}$, $\mathbf{X}_6 = \{0, 3\}$, $\mathbf{X}_7 = \{0, 2\}$, $\mathbf{X}_8 = \{0, 1\}$, $\mathbf{X}_9 = \{7, 0\}$, $\mathbf{X}_{10} = \{6, 0\}$, $\mathbf{X}_{11} = \{5, 0\}$, $\mathbf{X}_{12} = \{4, 0\}$, $\mathbf{X}_{13} = \{3, 0\}$, $\mathbf{X}_{14} = \{2, 0\}$, and $\mathbf{X}_{15} = \{1, 0\}$.

To evaluate policy $\mathcal{P}$, the traditional Markov-chain approach uses a continuous-time Markov chain (CTMC) with $S_1 \times S_2 = 56$ states. The infinitesimal generator of the CTMC is given in Table 12 in Appendix, and its steady-state distribution can be used to evaluate the cost of policy $\mathcal{P}$. Our method evaluates the steady-state distribution of a DTMC with only $Y = 8$ initial states: $\mathbf{Y}_1 = \{7, 8\}$, $\mathbf{Y}_2 = \{7, 7\}$, $\mathbf{Y}_3 = \{7, 6\}$, $\mathbf{Y}_4 = \{7, 5\}$, $\mathbf{Y}_5 = \{7, 4\}$, $\mathbf{Y}_6 = \{7, 3\}$, $\mathbf{Y}_7 = \{6, 8\}$, and $\mathbf{Y}_8 = \{5, 8\}$. This reduction in the number of states reduces the computational effort required to evaluate policy $\mathcal{P}$ significantly.

The transitions between initial states of our DTMC are driven by the probability to transition from initial state y to trigger state x, defined by P_{yx}, that can be obtained using the multinomial distribution:

$$P_{yx} = \begin{cases} \dfrac{(\Delta_{yx} - 1)!}{\prod_{i=1}^{N} (I_{yi} - I'_{xi})!} \displaystyle\prod_{i=1}^{N} \left(\frac{\lambda_i}{\lambda_{\mathbf{N}}}\right)^{(I_{yi} - I_{xi})} & \text{if } I_{yi} \geq I_{xi} \quad \forall i \in \mathbf{N}, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\Delta_{yx} = \sum_{i=1}^{N} (I_{yi} - I_{xi})$ and $I'_{xi} = (I_{xi} + 1)$ if item i triggers the order in trigger state x, and $I'_{xi} = I_{xi}$ otherwise (note that, if item i triggers the order in trigger state x, inventory position I_{xi} is never visited; hence the definition of I'_{xi}). The parameter $\lambda_{\mathbf{N}} = \sum_{i=1}^{N} \lambda_i$ defines the demand rate over all items. Given P_{yx} and $\mathcal{P}$, the probabilities to transition from initial state y to initial state y', denoted $\phi_{yy'}$, are then characterized by:

$$\phi_{yy'} = \sum_{x=1}^{X} P_{yx} \delta_{xy'}, \tag{2}$$

where $\delta_{xy'} = 1$ if $\mathcal{P}(x) = y'$, and $\delta_{xy'} = 0$ otherwise. The probabilities $\phi_{yy'}$ define the

DTMC with transitions from one initial state to another. Let $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_Y\}$ denote the steady-state distribution of this DTMC, with π_y denoting the steady-state probability of being in initial state y.

The expected time until the subsequent order, T_y, the expected order cost, O_y, and the expected cumulative inventory holding cost until the subsequent order is triggered, H_y, are performance measures associated with initial state y, and can be derived as follows:

$$T_y = \lambda_{\mathbf{N}}^{-1} \sum_{x=1}^{X} P_{yx} \Delta_{yx}, \tag{3}$$

$$H_y = \lambda_{\mathbf{N}}^{-1} \sum_{x=1}^{X} P_{yx} \Delta_{yx} \left(\sum_{i=1}^{N} \left(I'_{xi} + \frac{I_{yi} - I'_{xi}}{2} \right) h_i \right), \tag{4}$$

$$O_y = \sum_{x=1}^{X} P_{yx} \left(K + \sum_{i=1}^{N} k_i \gamma_{xi} \right), \tag{5}$$

where $\gamma_{xi} = 1$ if item i triggers or joins the order in trigger state x, and $\gamma_{xi} = 0$ otherwise. The cost of policy $\mathcal{P}$, denoted $C(\mathcal{P})$, is then given by:

$$C(\mathcal{P}) = \frac{\sum\limits_{y=1}^{Y} \pi_y \left(O_y + H_y\right)}{\sum\limits_{y=1}^{Y} \pi_y T_y}. \tag{6}$$

Similar to the traditional Markov-chain approach, the calculation of the steady-state distribution $\boldsymbol{\pi}$ is the computational bottleneck to evaluate $C(\mathcal{P})$. It requires the inversion of a matrix whose size is determined by the number of states in the Markov chain. For instance, if Gauss-Jordan elimination is used to invert the matrix, the complexity is $\mathcal{O}(n^3)$, with n denoting the number of states of the Markov chain. A traditional Markov-chain approach considers all states, including the transient states when transitioning from one initial state to another. Our method, in contrast, only considers initial states: states in which you end up after an order is placed. As a result, our method dominates a traditional Markov-chain approach, and never requires more states than a traditional Markov-chain approach.

The reduction in the number of states can be significant. A can-order policy, for instance, with re-order levels $\mathbf{s} = \{s_1, s_2, \ldots, s_N\}$, can-order levels $\mathbf{c} = \{c_1, c_2, \ldots, c_N\}$, and base-stock levels $\mathbf{S} = \{S_1, S_2, \ldots, S_N\}$ has $X = \sum_{i=1}^{N} \prod_{j \in \mathbf{N} \setminus \{i\}} (S_j - s_j)$ trigger states and $Y = \sum_{i=1}^{N} \prod_{j \in \mathbf{N} \setminus \{i\}} (S_j - c_j - \omega_{ij})$ initial states, with $\omega_{ij} = 1$ if $j < i$, and $\omega_{ij} = 0$ otherwise. In a traditional Markov-chain approach, the total number of states is $\prod_{i=1}^{N} (S_i - s_i)$. For instance, the traditional Markov-chain approach to evaluate a can-order policy for four items with $\mathbf{s} = \{0, 0, 0, 0\}$, $\mathbf{c} = \{13, 7, 11, 7\}$, and $\mathbf{S} = \{18, 11, 16, 11\}$ uses 34,848 states, whereas our method only requires 256 initial states.

The significant reduction of the state space to only initial states allows the exact evaluation of joint replenishment policies for instances that until now could only be evaluated using approximation procedures. Ignall (1969) restricts his analysis of the optimal joint replenishment policy to small two-item problems with a combined inventory of maximum 8 units. To limit the dimensions of the state space, Kayış et al. (2008) and Padilla Tinoco et al. (2017) restrict their analysis of the best can-order policy to only two items. Feng et al. (2015) are also limited to analyzing small instances with only a few items. Our method allows the analysis of sizeable instances with multiple items. To date, the literature relied on approximation procedures for such problem instances.

Example *We continue with the same example introduced earlier in this section, and illustrate how the transition matrix of the DTMC is set up. For instance, take initial state* $\mathbf{Y}_6 = \{7, 3\}$*. From* $\mathbf{Y}_6$*, we can end up in any of the following trigger states:* $\mathbf{X}_6 = \{0, 3\}$*,* $\mathbf{X}_7 = \{0, 2\}$*,* $\mathbf{X}_8 = \{0, 1\}$*,* $\mathbf{X}_9 = \{7, 0\}$*,* $\mathbf{X}_{10} = \{6, 0\}$*,* $\mathbf{X}_{11} = \{5, 0\}$*,* $\mathbf{X}_{12} = \{4, 0\}$*,* $\mathbf{X}_{13} = \{3, 0\}$*,* $\mathbf{X}_{14} = \{2, 0\}$*, and* $\mathbf{X}_{15} = \{1, 0\}$*. Using Eq.* (1)*, we can determine the transition probability from* $\mathbf{Y}_6$ *to each of these trigger states. For instance, the probability to transition to* $\mathbf{X}_8 = \{0, 1\}$ *is:*

$$P_{68} = \frac{8!}{6!2!}\left(\frac{12}{28}\right)^7\left(\frac{16}{28}\right)^2 = \frac{139,968}{5,764,801} = 0.0243.$$

The logic behind the above calculation is illustrated in Figure 1, that represents the probability tree when departing from initial state $\mathbf{Y}_6$*. At any given state in the tree that is not a trigger state, we can either move up (if demand for item 1 happens first) or move down (if demand for item 2 happens first). The probability of moving up or down is dictated by the law of competing exponentials; e.g., demand for item 1 happens first with probability* $\lambda_1\lambda_{\mathbf{N}}^{-1} = 12/28$*. From* $\mathbf{Y}_6$ *there are many paths to* $\mathbf{X}_8 = \{0, 1\}$*, and all need to be considered when determining* P_{68}*, the probability to reach trigger state 8 from initial state 6. As illustrated above, and as given by Eq.* (1)*, the multinomial distribution can be used to calculate* P_{68}*.*

When the system ends up in a trigger state, an order is placed, and a transition is made to an initial state. The expected time until order placement T_6*, its corresponding order cost* O_6*, and the expected cumulative holding cost* H_6 *can be derived by summing (over all trigger states) the product of* P_{6x} *with respectively, the expected transition time, the cost of the order placement, and the expected cumulative holding cost.*

In Table 1 we describe for a transition from $\mathbf{Y}_6$ *to each trigger state* x*: the transition probability* P_{6x}*; the initial state after placing the new order; the cost of the order placement; the number of transitions prior to order placement* Δ_{6x}*; the expected transition time; and*

Figure 1: Probability tree that illustrates the transition probabilities when departing from initial state $\mathbf{Y}_6 = \{7, 3\}$ (printed in purple) to end up in each of the accessible trigger states (printed in blue).

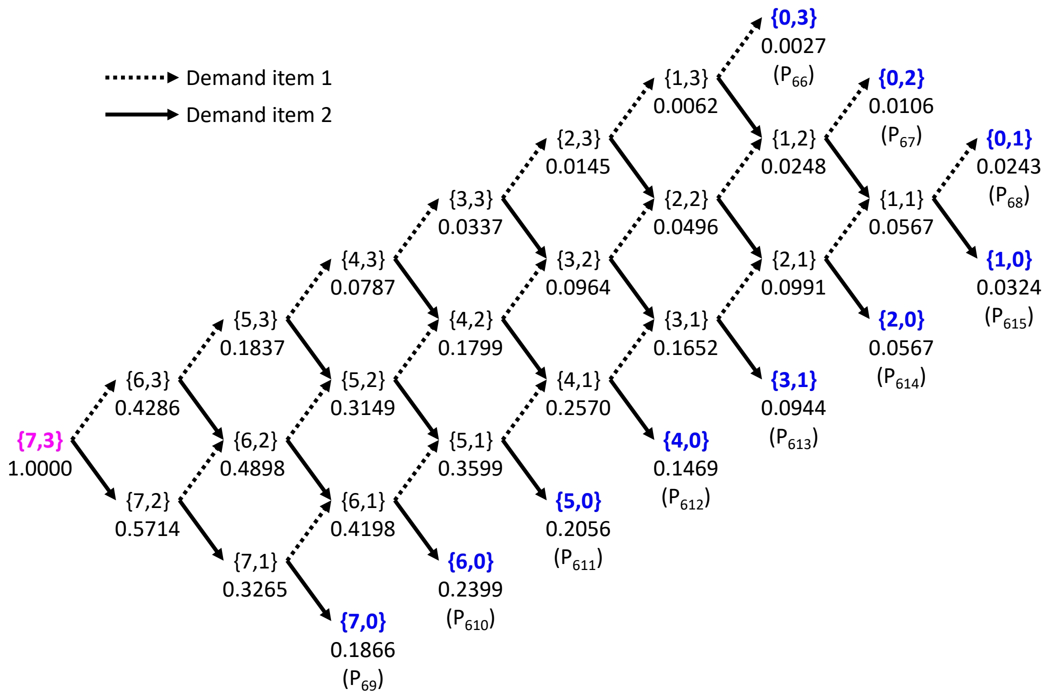


Table 1: Starting from $\mathbf{Y}_6 = \{7, 3\}$, the transition probabilities P_{6x} to each of the trigger states x, the initial state after the order is placed, the corresponding cost of the order placement, the number of transitions to reach trigger state x, its corresponding expected transition time, and the expected cumulative holding cost.

x	Transition probability P_{6x}	State after order	Order cost	Number of transitions	Expected transition time	Expected holding cost (cum.)
$\{0,3\}$	0.0027	$\{7,3\}$	32	7	$7/28$	29.2500
$\{0,2\}$	0.0106	$\{7,8\}$	53	8	$8/28$	30.1429
$\{0,1\}$	0.0243	$\{7,8\}$	53	9	$9/28$	30.2143
$\{7,0\}$	0.1866	$\{7,8\}$	46	3	$3/28$	13.9286
$\{6,0\}$	0.2399	$\{6,8\}$	46	4	$4/28$	17.7143
$\{5,0\}$	0.2056	$\{5,8\}$	46	5	$5/28$	21.0714
$\{4,0\}$	0.1469	$\{7,8\}$	53	6	$6/28$	24.0000
$\{3,0\}$	0.0944	$\{7,8\}$	53	7	$7/28$	26.5000
$\{2,0\}$	0.0567	$\{7,8\}$	53	8	$8/28$	28.5714
$\{1,0\}$	0.0324	$\{7,8\}$	53	9	$9/28$	30.2143

the expected cumulative holding cost until the subsequent order. For instance, starting from initial state $\mathbf{Y}_6 = \{7, 3\}$, *it takes 9 transitions to end up in* $\mathbf{X}_8 = \{0, 1\}$. *Each transition is expected to take* $\lambda_{\mathbf{N}}^{-1} = 1/28$ *time units, and hence, we expect that it will take* $9/28$ *time units before we end up in* $\mathbf{X}_8 = \{0, 1\}$. *During each of these transitions, inventory is kept for both items. For item* 2, *an average of 2 (i.e.,* $1 + (3 - 1) \times 2^{-1}$*) units is kept in inventory for 9 transitions. Therefore, for item* 2, *the expected cumulative holding cost amounts to* $2 \times 9 \times 23 \times 28^{-1} = 414/28$. *Analogously, for item* 1, *the expected cumulative holding cost is* $432/28$. *For both items, the total expected cumulative holding cost is* $846/28$. *Using Table 1, we can calculate* $T_6 = 0.1842$, $O_6 = 48.5194$, *and* $H_6 = 20.9369$, *using Eqs. (3–5). For instance,* $T_6 = (0.0027 \times 7/28) + (0.0106 \times 8/28) + (0.0243 \times 9/28) + (0.1866 \times 3/28) + (0.2399 \times 4/28) + (0.2056 \times 5/28) + (0.1469 \times 6/28) + (0.0944 \times 7/28) + (0.0567 \times 8/28) + (0.0324 \times 9/28) = 0.1842$.

Once probabilities P_{yx} *are derived, we can determine* $\phi_{yy'}$ *from Eq.* (2) *to set up the DTMC. Table 2 reports the DTMC and its steady-state distribution* $\boldsymbol{\pi}$. *For instance, the probability to transition from initial state* $\mathbf{Y}_6 = \{7, 3\}$ *to initial state* $\mathbf{Y}_7 = \{6, 8\}$ *is given by* $\phi_{67} = 0.2399$, *and can also be derived from Figure 1; in fact,* ϕ_{67} *equals* P_{610} *as trigger state* $\mathbf{X}_{10} = \{6, 0\}$ *is the only trigger state that links to initial state* $\mathbf{Y}_7$ *when departing from initial state* $\mathbf{Y}_6$. *In order to calculate the cost of policy* $\mathcal{P}$, *it suffices to use steady-state probabilities* $\boldsymbol{\pi}$ *as weights for* O_y, T_y, *and* H_y. *Table 3 summarizes the results for each initial state* y. *We can now use Eq.* (6) *to find* $C(\mathcal{P}) = 109.0152/0.3838 = 284.0749$.

Table 2: Transition probability matrix and steady-state distribution of the DTMC that models the transitions between initial states.

$\phi_{yy'}$	$\{7,8\}$	$\{7,7\}$	$\{7,6\}$	$\{7,5\}$	$\{7,4\}$	$\{7,3\}$	$\{6,8\}$	$\{5,8\}$
$\{7,8\}$	0.6751	0.0106	0.0243	0.0416	0.0595	0.0748	0.0390	0.0752
$\{7,7\}$	0.6994	0.0027	0.0106	0.0243	0.0416	0.0595	0.0597	0.1023
$\{7,6\}$	0.6970	0.0000	0.0027	0.0106	0.0243	0.0416	0.0895	0.1343
$\{7,5\}$	0.6640	0.0000	0.0000	0.0027	0.0106	0.0243	0.1306	0.1679
$\{7,4\}$	0.6081	0.0000	0.0000	0.0000	0.0027	0.0106	0.1828	0.1958
$\{7,3\}$	0.5518	0.0000	0.0000	0.0000	0.0000	0.0027	0.2399	0.2056
$\{6,8\}$	0.6428	0.0212	0.0425	0.0647	0.0832	0.0951	0.0114	0.0390
$\{5,8\}$	0.5632	0.0413	0.0708	0.0944	0.1079	0.1110	0.0000	0.0114
$\boldsymbol{\pi}$	0.6516	0.0119	0.0248	0.040	0.055	0.0677	0.0606	0.0883

Table 3: For all initial states $y : y \in \mathcal{Y}$, the steady-state probability π_y, the expected order cost O_y, the expected transition time T_y, the expected cumulative holding costs H_y, contribution to expected total cost $\pi_y (O_y + H_y)$, and contribution to expected total transition time $\pi_y T_y$.

y	π_y	O_y	T_y	H_y	$\pi_y (O_y + H_y)$	$\pi_y T_y$
$\{8,7\}$	0.6516	47.6402	0.4267	70.9474	77.2744	0.2780
$\{7,7\}$	0.0119	48.8153	0.3885	59.7237	1.2903	0.0046
$\{6,7\}$	0.0248	49.5268	0.3447	49.0343	2.4483	0.0086
$\{5,7\}$	0.0400	49.6959	0.2955	38.9584	3.5498	0.0118
$\{4,7\}$	0.0550	49.3245	0.2417	29.5708	4.3382	0.0133
$\{3,7\}$	0.0677	48.5194	0.1842	20.9369	4.7056	0.0125
$\{8,6\}$	0.0606	46.0733	0.3942	63.5789	6.6396	0.0239
$\{8,5\}$	0.0883	43.6824	0.3517	55.6045	8.7690	0.0311
				Total	109.0152	0.3838

3 Characteristics of the optimal policy

In Section 2 we showed how a DTMC that only considers transitions between initial states can be used to efficiently evaluate the cost of a joint-replenishment policy. We illustrated our approach by means of an evaluation of a can-order policy heuristic. In this section we characterize the *optimal* joint replenishment policy, and use our findings to develop a greedy-optimal algorithm in Section 4. We first define a number of preliminary notions:

- If a set of items $\mathbf{N}' \subseteq \mathbf{N}$ makes up the order that was triggered in trigger state x, an order cost $K + \sum_{i \in \mathbf{N}'} k_i$ is incurred, and a set of initial states $\mathcal{Y}_{x\mathbf{N}'} \subseteq \mathcal{Y}$ becomes accessible (i.e., if item i orders, it can transition to initial states where item i's inventory has been replenished).

- Each initial state may be associated with a future cost, defined as the long-run cost starting from that initial state and following policy $\mathcal{P}$. In each set $\mathcal{Y}_{x\mathbf{N}'}$ we can identify at least one initial state that has the smallest future cost.

- In trigger state x, optimal policy $\mathcal{P}^*$ defines the set of items $\mathbf{N}'$ and the initial state $y : y \in \mathcal{Y}_{x\mathbf{N}'}$ that minimize the sum of the order and future cost; policy $\mathcal{P}^*$ either decides to include item i in the order such that more (and potentially better) initial states become accessible, or not to include item i in order to avoid the minor order cost.

In what follows we also assume that, if an order is triggered, any excess inventory can be returned at no cost; i.e., we allow for negative orders of item j whenever an order is triggered by item $i \neq j$. Therefore, if a set of items $\mathbf{N}' \subseteq \mathbf{N}$ is included in the order that was triggered in trigger state x, a transition can be made to the set of initial states $\mathcal{Y}_{x\mathbf{N}'} = \{y | I_{yi} \leq I_{xi}, \forall i \in \mathbf{N} \setminus \mathbf{N}' \wedge I_{yi} \in \mathbb{Z}, \forall i \in \mathbf{N}'\}$. The reason why we include negative orders in the analysis is motivated by the observation that the optimal policy can include negative orders. Negative orders may be seen as inventory that is returned to the supplier.

We first show that for a given set of items $\mathbf{N}' \subseteq \mathbf{N}$ ordered, it is always optimal to transition to one and the same initial state. The latter is dependent on the inventory positions of the items that are not ordered.

THEOREM 1. *In trigger states x and x', it is optimal to transition to one and the same initial state y if: (1) the same set of items $\mathbf{N}'$ are included in the order that was triggered in trigger states x and x', and (2) $I_{xl} = I_{x'l}$ for any item $l \notin \mathbf{N}'$.*

PROOF. Let y denote the initial state that has the smallest future cost among all initial states in $\mathcal{Y}_{x\mathbf{N}'}$. Hence, if items $\mathbf{N}'$ are ordered in trigger state x, it is optimal to transition to initial state y. Next, consider trigger state x', and let $\mathcal{Y}_{x'\mathbf{N}'}$ denote the set of initial states that become accessible if the set of items $\mathbf{N}'$ is ordered in trigger state x'. Because $I_{xl} = I_{x'l}$ for any item l that is not included in the order, it follows that $\mathcal{Y}_{x'\mathbf{N}'} \subseteq \mathcal{Y}_{x\mathbf{N}'}$, and initial state y is also accessible in trigger state x'. Therefore, if initial state y has the smallest future cost among all initial states in $\mathcal{Y}_{x\mathbf{N}'}$, it also has the smallest future cost among all initial states in $\mathcal{Y}_{x'\mathbf{N}'} \subseteq \mathcal{Y}_{x\mathbf{N}'}$. Therefore, it is optimal to transition to initial state y if the items $\mathbf{N}'$ are ordered in trigger state x'. □

Corollary 1. *In any trigger state x, if all items $\mathbf{N}$ are included in the order, it is always optimal to transition to one and the same initial state, i.e., the state with minimal future costs among all possible initial states. This initial state is also referred to as the renewal state.*

Proof. Corollary 1 follows from Theorem 1. □

Corollary 2. *Given a set of items $\mathbf{N}' \subseteq \mathbf{N}$ included in the order, it is optimal to transition to the renewal state if a transition can be made to this renewal state from trigger state x.*

Proof. Among all possible initial states, the renewal state is the state that has the smallest future cost. Hence, a transition is made towards the renewal state if it becomes accessible. Note that, not all items need to be included in the order to make the renewal state accessible; the inventory position of some items may already be at/above their level in the renewal state. □

We next show that, when it is optimal to order a set of items $\mathbf{N}' \subseteq \mathbf{N}$ for a given set of inventory positions of items $l \notin \mathbf{N}'$, it is also optimal to order the same set $\mathbf{N}'$ when items $\mathbf{N}'$ have a lower inventory position.

Theorem 2. *If it is optimal to order the set of items $\mathbf{N}' \subset \mathbf{N}$ in trigger state x, it is also optimal to order the same set $\mathbf{N}'$ in trigger state x' if: (1) $I_{x'i} \leq I_{xi}$ for any item $i \in \mathbf{N}'$, and (2) $I_{x'l} = I_{xl}$ for any item $l \notin \mathbf{N}'$.*

Proof. Suppose that in trigger state x it is optimal to order items $\mathbf{N}'$ with order cost $K + k_i + \sum_{j\in\mathbf{N}'} k_j$ such that initial states $\mathcal{Y}_{x\mathbf{N}'}$ become accessible. In addition, let y denote the initial state with the smallest future cost among all initial states in $\mathcal{Y}_{x\mathbf{N}'}$. Then, for trigger state x' with $I_{x'l} = I_{xl}$ for any item $l \notin \mathbf{N}'$, and $I_{x'i} \leq I_{xi}$ for all items $i \in \mathbf{N}'$, it follows that $\mathcal{Y}_{x'\mathbf{N}'} \subseteq \mathcal{Y}_{x\mathbf{N}'}$. Furthermore, note that initial state y also becomes accessible in x' if we incur order cost $K + k_i + \sum_{j\in\mathbf{N}'} k_j$; the same order cost that was required to access y in trigger state x. Because y minimizes the sum of order and future costs in x, and because all initial states accessible in x' can also be made accessible in x at equal cost, it follows that y also minimizes the sum of order and future costs in x'. Therefore, if it is optimal to order items $\mathbf{N}'$ in trigger state x, it is also optimal to order the same set of items in trigger state x'. □

Corollary 3. *Item j has an optimal can-order level for each subset of trigger states $\mathcal{X}' \subset \mathcal{X}$, for which $I_{xl} = I_{x'l}$, for all items $l \neq j$, and for all $x, x' \in \mathcal{X}'$.*

Proof. Corollary 3 follows from Theorem 2. □

Corollary 4. *In a system with two items, each item has a single optimal can-order level.*

Proof. Corollary 4 follows from Theorem 2. In a system with two items, each item i only has one subset of trigger states $\mathcal{X}' \subset \mathcal{X}$ for which $I_{xl} = I_{x'l}$ for $l \neq i$, and for all $x, x' \in \mathcal{X}'$. □

Finally, we show that when an item is included in an order, its optimal order size will never exceed the optimal order size when it would be replenished individually.

Theorem 3. *For any item i, it is never optimal to order up to an inventory position that exceeds Q_i, where Q_i is the economic order quantity of item i when considering order cost $K + k_i$.*

Proof. When considering only the costs of item i, it is never beneficial to order up to an inventory position that exceeds Q_i. Therefore, ordering up to an inventory higher than Q_i only makes sense if this results in additional opportunities for other items to join an order triggered by item i. A larger order size of item i, however, generates less orders triggered by item i. Therefore, it is never optimal to order up to an inventory position that exceeds Q_i. □

Based on the above theorems and corollaries, we conclude that the optimal policy has a can-order structure; a can-order level determines whether item j joins the order. The can-order level itself is dependent on the inventory positions of the items that are not included in the order. From Theorem 3, we also know that item i will never replenish more than Q_i units. In fact, we have observed that if item i is included in the order, it will most of the times order up to its inventory position in the renewal state, i.e., the order-up-to level if all items are jointly ordered. Nevertheless, if not all items are replenished simultaneously, it may sometimes be better to synchronize the order of item i with the order cycle of an item l that was not included in the order. In those cases, it may be optimal to order more (or less) than up to the renewal inventory level.

The optimal policy is therefore different from the conventional can-order policy, where each item's can-order level remains fixed, independent of the items ordered and the inventory positions of the items that are not ordered. In addition, any order raises the inventory position up to its respective order-up-to level. In what follows, we introduce a greedy-optimal algorithm to modify the conventional can-order policy using the theorems above. We refer to this modified can-order policy as the "generalized can-order policy".

4 Generalizing the can-order policy

We use the characteristics of the optimal policy to generalize the conventional can-order policy. To find the best generalized policy, we start from the conventional can-order policy $\mathcal{P}$, with parameter values obtained using the decomposition heuristic, or obtained through a numerical search over the parameter range.

We initialize the conventional can-order policy $\mathcal{P}$ with re-order levels $\mathbf{s} = \{s_1, s_2, \ldots, s_N\}$, can-order levels $\mathbf{c} = \{c_1, c_2, \ldots, c_N\}$, and order-up-to levels $\mathbf{S} = \{S_1, S_2, \ldots, S_N\}$ as:

$$\begin{aligned} I_{\mathcal{P}(x)i} &= S_i \quad \text{if } I_{xi} \leq c_i, \quad \forall i \in \mathbf{N}, x \in \mathcal{X}, \\ I_{\mathcal{P}(x)i} &= I_{xi} \quad \text{otherwise}, \quad \forall i \in \mathbf{N}, x \in \mathcal{X}, \end{aligned}$$

where $\mathcal{X}$ contains all trigger states x for which at most one item i has inventory position $I_{xi} = s_i$.

Next, we generalize can-order policy $\mathcal{P}$, using the greedy algorithm outlined in Algorithm 1. Note that we adopt a greedy approach because it is not tractable to identify the optimal policy using a brute-force approach for systems with more than two items. For each item i triggering the order, and for each item j that may join the order triggered by i, we define the subset of trigger states $\mathcal{X}'$ for each combination of values of $I_{x'l}$, for all items $l \neq \{i, j\}$ (if there are only two items, $\mathcal{X}'$ holds all trigger states of item i). For each trigger state in $\mathcal{X}'$, the algorithm finds the optimal value of the can-order level for item j, as well as the optimal values of the order-up-to levels for items i and j in case both items are jointly ordered, and the optimal order-up-to level for item i if only item i is ordered. More specifically, for each subset of trigger states $\mathcal{X}'$, the algorithm performs the following three steps:

1. First, we initialize y_R, the best initial state in which we end up if item j returns excess inventory. Next, we define y_J, the initial state in which we end up if items i and j are jointly ordered. From Theorem 1, it indeed follows that a transition is made to y_J in any of the trigger states in $\mathcal{X}'$ if i and j are jointly ordered; y_J is then the initial state

that minimizes the future cost among all accessible initial states from $\mathcal{X}'$ if i and j are both ordered.

2. In a second step, we process all trigger states $x : x \in \mathcal{X}'$ in ascending order of I_{xj}. For each trigger state x, we determine y_M, the initial state in which we end up if item j maintains its inventory level (i.e., it does not join the order nor does it return excess inventory). If the cost of keeping item j at the same inventory level is lower than the cost of returning excess inventory, y_R is updated, and y_M will become the new best initial state in which excess inventory is returned. Next, we compare the cost of a joint order (i.e., a transition is made to y_J) with the cost of maintaining or reducing item j's inventory (i.e., a transition is made to y_R). If it is beneficial for item j to join item i's order, policy $\mathcal{P}$ is adapted accordingly, and trigger state x is removed from $\mathcal{X}'$ (i.e., it has been processed). If, on the other hand, joining is no longer beneficial, we proceed to step 3. Recall that, from Theorem 2, it follows that item j has an optimal can-order level in a set of trigger states $\mathcal{X}'$, and for trigger states where the inventory of item j exceeds this can-order level, it is no longer optimal for item j to join.

3. In a third step, we process any remaining trigger states $x : x \in \mathcal{X}'$ in ascending order of I_{xj}. Since item j no longer joins the order of item i, it may no longer be optimal for item i to order up to its inventory level in y_J, the initial state in which we end up if items i and j are jointly ordered. Therefore, we first evaluate all order-up-to levels of item i to determine y_M; the best initial state where item j maintains it inventory level. The remainder of step 3 is in fact a repetition of the first part of step 2: we evaluate whether it is better for item j to maintain its inventory (i.e., transition to y_M) or to return its excess inventory (i.e., transition to y_R), and update y_R if necessary. Next, policy $\mathcal{P}$ is adapted accordingly, and trigger state x is removed from $\mathcal{X}'$. Step 3 finishes if all trigger states in $\mathcal{X}'$ have been processed.

These steps are illustrated in the numerical example in the next section.

Although our greedy algorithm exploits the optimal policy characteristics, it is not perfect. First, policy $\mathcal{P}$ is sequentially updated per trigger state x. Due to its sequential nature, it is possible that the optimal policy for trigger state x becomes suboptimal for a trigger state x' that was processed earlier. Therefore, we iteratively run Algorithm 1 until policy $\mathcal{P}$ converges. Second, we process the items that can join item i's order one by one. As a result, when searching for item j's optimal can-order level, the algorithm assumes that the policy of the remaining $\mathbf{N} \setminus \{i, j\}$ items remains the same (i.e., changes that consider multiple items simultaneously are not considered).

Although the algorithm is computationally very fast, it can be further sped up by introducing the following adjustments. First, from Theorem 3, we know that an item's order-up-to level will never exceed its economic order quantity. Hence, it never makes sense to evaluate order-up-to levels that exceed the economic order quantity. Second, assuming that the costs are convex in the order-up-to levels (this assumption has been used for the conventional can-order policy, see e.g., Zheng (1994)), it suffices to evaluate only new order-up-to levels as long as the costs improve, rather than evaluating all possible order-up-to levels.

Note that Algorithm 1 requires the exact evaluation of a policy. As the algorithm compares policies that only differ in a single trigger state, the cost difference is too small to identify using simulation with a high level of confidence. For sizeable instances, such an exact evaluation is only possible with the embedded Markov-chain method introduced in Section 2.

Algorithm 1: Greedy algorithm to generalize policy $\mathcal{P}$

```
forall the $i \in \mathbf{N}$ do
    forall the $j \in \mathbf{N} \setminus \{i\}$ do
        forall the $\mathcal{X}' \subset \mathcal{X}$ do
            Step 1:
            Let $y_R = -1$, and evaluate all order-up-to levels of $i$ and $j$ to determine $y_J$;
            Step 2:
            do
                Let $x = x \in \mathcal{X}'$ for which $I_{xj} = \min\{I_{x'j} | x' \in \mathcal{X}'\}$;
                if $C(\mathcal{P}|\mathcal{P}(x) = y_M) < C(\mathcal{P}|\mathcal{P}(x) = y_R)$ then
                    $y_R = y_M$;
                if $C(\mathcal{P}|\mathcal{P}(x) = y_J) < C(\mathcal{P}|\mathcal{P}(x) = y_R)$ then
                    $\mathcal{P}(x) = y_J \wedge \mathcal{X}' = \mathcal{X}' \setminus \{x\}$;
                else
                    break;
            while $\mathcal{X}' \neq \emptyset$;
            Step 3:
            while $\mathcal{X}' \neq \emptyset$ do
                Let $x = x \in \mathcal{X}'$ for which $I_{xj} = \min\{I_{x'j} | x' \in \mathcal{X}'\}$;
                Evaluate all order-up-to levels of item $i$ to determine $y_M$;
                if $C(\mathcal{P}|\mathcal{P}(x) = y_M) < C(\mathcal{P}|\mathcal{P}(x) = y_R)$ then
                    $y_R = y_M$;
                $\mathcal{P}(x) = y_R \wedge \mathcal{X}' = \mathcal{X}' \setminus \{x\}$;
```

Table 4: The transitions from trigger state x to initial state y in the conventional can-order policy $\mathcal{P}$, defined by $S_1 = 7, S_2 = 8, c_1 = 4, c_2 = 2$, and the generalized policies $\mathcal{P}'$ and $\mathcal{P}''$, obtained after one and two iterations, respectively. Policy $\mathcal{P}''$ is also the optimal policy.

$\mathcal{P}$ (can-order)			$\mathcal{P}'$ (first iteration)			$\mathcal{P}''$ (second iteration)		
x		y	x		y	x		y
$\{0,8\}$	$\rightarrow$	$\{7,8\}$	$\{0,8\}$	$\rightarrow$	$\{7,8\}$	$\{0,8\}$	$\rightarrow$	$\{7,8\}$
$\{0,7\}$	$\rightarrow$	$\{7,7\}$	$\{0,7\}$	$\rightarrow$	$\{7,7\}$	$\{0,7\}$	$\rightarrow$	$\{7,7\}$
$\{0,6\}$	$\rightarrow$	$\{7,6\}$	$\{0,6\}$	$\rightarrow$	$\{7,6\}$	$\{0,6\}$	$\rightarrow$	$\{7,6\}$
$\{0,5\}$	$\rightarrow$	$\{7,5\}$	$\boldsymbol{\{0,5\}}$	$\boldsymbol{\rightarrow}$	$\boldsymbol{\{6,5\}}$	$\{0,5\}$	$\rightarrow$	$\{7,5\}$
$\{0,4\}$	$\rightarrow$	$\{7,4\}$	$\boldsymbol{\{0,4\}}$	$\boldsymbol{\rightarrow}$	$\boldsymbol{\{6,4\}}$	$\boldsymbol{\{0,4\}}$	$\boldsymbol{\rightarrow}$	$\boldsymbol{\{6,4\}}$
$\{0,3\}$	$\rightarrow$	$\{7,3\}$	$\{0,3\}$	$\rightarrow$	$\{7,3\}$	$\{0,3\}$	$\rightarrow$	$\{7,3\}$
$\{0,2\}$	$\rightarrow$	$\{7,8\}$	$\{0,2\}$	$\rightarrow$	$\{7,8\}$	$\{0,2\}$	$\rightarrow$	$\{7,8\}$
$\{0,1\}$	$\rightarrow$	$\{7,8\}$	$\{0,1\}$	$\rightarrow$	$\{7,8\}$	$\{0,1\}$	$\rightarrow$	$\{7,8\}$
$\{7,0\}$	$\rightarrow$	$\{7,8\}$	$\{7,0\}$	$\rightarrow$	$\{7,8\}$	$\{7,0\}$	$\rightarrow$	$\{7,8\}$
$\{6,0\}$	$\rightarrow$	$\{6,8\}$	$\boldsymbol{\{6,0\}}$	$\boldsymbol{\rightarrow}$	$\boldsymbol{\{6,7\}}$	$\boldsymbol{\{6,0\}}$	$\boldsymbol{\rightarrow}$	$\boldsymbol{\{6,7\}}$
$\{5,0\}$	$\rightarrow$	$\{5,8\}$	$\boldsymbol{\{5,0\}}$	$\boldsymbol{\rightarrow}$	$\boldsymbol{\{5,7\}}$	$\boldsymbol{\{5,0\}}$	$\boldsymbol{\rightarrow}$	$\boldsymbol{\{5,7\}}$
$\{4,0\}$	$\rightarrow$	$\{7,8\}$	$\boldsymbol{\{4,0\}}$	$\boldsymbol{\rightarrow}$	$\boldsymbol{\{4,7\}}$	$\boldsymbol{\{4,0\}}$	$\boldsymbol{\rightarrow}$	$\boldsymbol{\{4,7\}}$
$\{3,0\}$	$\rightarrow$	$\{7,8\}$	$\{3,0\}$	$\rightarrow$	$\{7,8\}$	$\{3,0\}$	$\rightarrow$	$\{7,8\}$
$\{2,0\}$	$\rightarrow$	$\{7,8\}$	$\{2,0\}$	$\rightarrow$	$\{7,8\}$	$\{2,0\}$	$\rightarrow$	$\{7,8\}$
$\{1,0\}$	$\rightarrow$	$\{7,8\}$	$\{1,0\}$	$\rightarrow$	$\{7,8\}$	$\{1,0\}$	$\rightarrow$	$\{7,8\}$
$C(\mathcal{P}) = 284.0749$			$C(\mathcal{P}') = 283.8574$			$C(\mathcal{P}'') = 283.8571$		

5 Numerical analysis

We illustrate how the algorithm can be used to identify the generalized can-order policy by three numerical examples.[1] The first example illustrates, for the numerical setting introduced in Section 2, how the algorithm updates the conventional can-order policy in two iterations towards the optimal policy $\mathcal{P}''$, and reduces costs slightly, see Table 4.

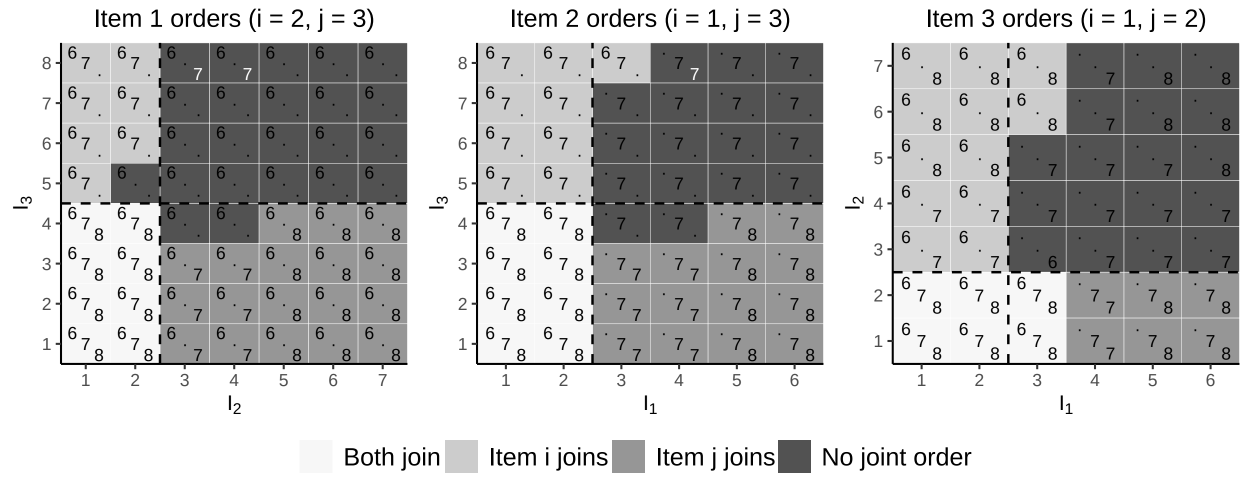
Table 4 can also be used to illustrate Algorithm 1. In order to obtain policy $\mathcal{P}'$, we start from can-order policy $\mathcal{P}$, and select the first item (item i) in $\mathbf{N}$. In the example, there are only two items, and therefore, only the second item (item j) remains in $\mathbf{N} \setminus \{i\}$. If the first item triggers the order, the subset of trigger states $\mathcal{X}'$ contains trigger states $\mathbf{X}_1 = \{0,8\}$, $\mathbf{X}_2 = \{0,7\}$, $\mathbf{X}_3 = \{0,6\}$, $\mathbf{X}_4 = \{0,5\}$, $\mathbf{X}_5 = \{0,4\}$, $\mathbf{X}_6 = \{0,3\}$, $\mathbf{X}_7 = \{0,2\}$, and $\mathbf{X}_8 = \{0,1\}$. We will now illustrate the three steps of Algorithm 1. In a first step, we initialize y_R (the best initial state in which we can end up after returning inventory of the second item) at -1. Next, we let $y_J = \mathbf{Y}_1 = \{7,8\}$, the initial state in which we end up if both items join the order in can-order policy $\mathcal{P}$. In a second step, we select (from $\mathcal{X}'$) the

[1]Our findings are also confirmed by an extensive numerical experiment of 154 problem instances with 2, 3 and 4 items that have different cost parameters.

trigger state that has the smallest inventory for the second item (i.e., $\mathbf{X}_8 = \{0, 1\}$). For this trigger state, we evaluate whether or not it is better for the second item to maintain its inventory, or to reduce its inventory. As the inventory of the second item cannot be reduced in state $\mathbf{X}_8 = \{0, 1\}$, the minimal cost is obtained when the second item maintains its inventory, and we let $y_R = y_M = \{7, 1\}$. Next, we evaluate whether or not the second item should join the order triggered by the first item; i.e., we compare the cost of the policy in which the second item joins the order in trigger state $\mathbf{X}_8 = \{0, 1\}$ against the policy in which the second item does not join the order in trigger state $\mathbf{X}_8 = \{0, 1\}$. In case of trigger state $\mathbf{X}_8 = \{0, 1\}$, the cost of joining is smaller than the cost of not joining (i.e., $C(\mathcal{P}|\mathcal{P}(x) = y_J) < C(\mathcal{P}|\mathcal{P}(x) = y_R)$), and we update the generalized policy $\mathcal{P}'$ accordingly (i.e., in trigger state $\mathbf{X}_8 = \{0, 1\}$, the second item should join the order triggered by the first item, and a transition is made to initial state $y_J = \mathbf{Y}_1 = \{7, 8\}$). Afterwards, we remove trigger state $\mathbf{X}_8 = \{0, 1\}$ from the subset of trigger states $\mathcal{X}'$. At this point, step 2 has been performed for trigger state $\mathbf{X}_8 = \{0, 1\}$, and we move on to the next trigger state $\mathbf{X}_7 = \{0, 2\}$. For trigger state $\mathbf{X}_7 = \{0, 2\}$, the procedure is similar to that of trigger state $\mathbf{X}_8 = \{0, 1\}$, and we will omit its discussion here. For trigger state $\mathbf{X}_6 = \{0, 3\}$, however, it is no longer beneficial for the second item to join the order of the first item, and hence, in step 2, $C(\mathcal{P}|\mathcal{P}(x) = y_J) > C(\mathcal{P}|\mathcal{P}(x) = y_R)$, and we stop processing step 2 (i.e., we "break" the second step). Note that at this point in time, $y_R = y_M = \mathbf{Y}_6 = \{7, 3\}$, and $\mathcal{X}'$ still contains trigger states $\mathbf{X}_1 = \{0, 8\}$, $\mathbf{X}_2 = \{0, 7\}$, $\mathbf{X}_3 = \{0, 6\}$, $\mathbf{X}_4 = \{0, 5\}$, $\mathbf{X}_5 = \{0, 4\}$, and $\mathbf{X}_6 = \{0, 3\}$. From Theorem 2 it follows that the second item will no longer join the order triggered by the first item in any of the remaining trigger states in $\mathcal{X}'$. In each of the trigger states, however, we still need to: (1) verify the best order-up-to level of the first item, and (2) determine whether it is beneficial to return excess inventory for the second item. In case of trigger state $\mathbf{X}_6 = \{0, 3\}$, the can-order policy $\mathcal{P}$ cannot be improved (i.e., we transition to initial state $y_M = \mathbf{Y}_6 = \{7, 3\}$), and after it is processed, it is removed from subset $\mathcal{X}'$. In case of trigger states $\mathbf{X}_4 = \{0, 5\}$ and $\mathbf{X}_5 = \{0, 4\}$, however, the order-up-to level of the first item is reduced to 6; i.e., y_M (the best initial state in which we can end up if we maintain the inventory of the second item) equals $\{6, 5\}$ and $\{6, 4\}$, respectively. There are no changes (when compared to the can-order policy $\mathcal{P}$) for the other, remaining trigger states, and we finish step 3. After step 3 has been completed, we have processed all trigger states where the first item triggers the order. What remains to obtain $\mathcal{P}'$ is to also process all trigger states where the second item triggers the order. Last but not least, $\mathcal{P}''$ is then obtained by running Algorithm 1 a second time, using policy $\mathcal{P}'$ as the initial policy.

In a second example we extend the same numerical setting with a third item with parameters $\lambda_3 = 30$, $h_3 = 30$, and $k_3 = 7$. The best conventional can-order policy for this example

Figure 2: Generalized can-order policy obtained using Algorithm 1. Each panel shows the policy when item 1 (left panel), item 2 (middle), or item 3 (right) triggers the order. The color of the cell determines whether other items join the order or not. The numbers inside the cells represent the respective order-up-to levels of the three items. The can-order levels of the best conventional can-order policy $\mathcal{P}$ are indicated by the dashed lines.



has $\mathbf{s} = \{0, 0, 0\}$, $\mathbf{c} = \{2, 2, 4\}$, and $\mathbf{S} = \{6, 7, 7\}$, and a total cost of 513.56. The generalized policy $\mathcal{P}'$ obtained by Algorithm 1 has a slightly lower cost of 512.70. We cannot guarantee that $\mathcal{P}'$ is optimal, as a brute-force approach would require the evaluation of 512^{192} policies; our greedy algorithm only has to evaluate 5,666 policies. The best conventional and the best generalized can-order policies are illustrated in Figure 2. Each panel shows the generalized policy when item 1 (left panel), item 2 (middle), or item 3 (right) triggers the order. The color of the cell determines whether other items join the order or not. The numbers inside the cells represent the respective order-up-to levels of the three items. For instance, if item 1 triggers the order while items 2 and 3 each have 2 units in inventory, they are jointly ordered up to respectively 6 (item 1), 7 (item 2), and 8 (item 3). The can-order levels of the best conventional can-order policy $\mathcal{P}$ are indicated by the dashed lines. The figure shows how the generalized can-order policy differs from the conventional can-order policy. The numbers printed in white indicate negative orders. A negative order may be optimal when, for instance, the demand for a particular item i was far less than expected, and another item $j \neq i$ triggers a replenishment order that results in high inventory levels for all items. In such a situation, it may be beneficial to get rid of some of the inventory of item i in order to not incur the holding cost during the long time until the next replenishment order is triggered.

We note that the model complexity is the same whether we include negative orders or not. The reason why we have included negative orders in the analysis is motivated by the observation that the optimal policy can include negative orders. We have also run our greedy

Table 5: Parameter values of the three instances of Federgruen et al. (1984)

Parameter	item 1	item 2	item 3	item 4
Instance 1				
λ_i	10	5	10	5
h_i	1	1	2	1
k_i	3	3	3	3
K	33			
Instance 2				
λ_i	10	5	10	5
h_i	1	1	2	1
k_i	5	5	5	5
K	30			
Instance 3				
λ_i	10	5	10	5
h_i	1	1	2	1
k_i	5	5	5	5
K	15			

algorithm without allowing negative orders. The resulting cost is 512.703319 compared to 512.702941 when negative orders are allowed. Although negative orders do exist in the optimal policy, the cost difference with the optimal policy that does not allow negative orders is only minimal.

In the third example, we use our embedded Markov-chain method and greedy algorithm to analyze the instances used in the numerical experiment of Federgruen et al. (1984) with four items. The parameters of these instances are summarized in Table 5. We optimize the can-order policy parameters using the decomposition approach proposed by Federgruen et al. (1984), and evaluate the obtained policy costs using their cost approximations, as well as exact. Indeed, our embedded Markov-chain model allows the exact evaluation of their can-order policy, as it requires only 256, 300, and 853 initial states for the three instances respectively. A traditional Markov-chain approach has 34,848, 34,848, and 18,000 states for these three instances, which makes it impractical to compute its steady-state performance. (it may take hours to calculate the the steady-state distribution; using our embedded Markov-chain method, we can evaluate the cost of the can-order policies in a few seconds or less). Federgruen et al. (1984) therefore relies on approximations to evaluate its cost performance. Table 6 reports the cost performance of the can order-policy with the parameters obtained by the decomposition heuristic, respectively using the cost approximations proposed in Federgruen et al. (1984) and its exact cost evaluation using our method. We also report the cost performance of the best conventional can-order policy. Finally we report the cost performance of the best generalized can-order policy, which we found by

Table 6: Cost performance of the can-order policy identified by the decomposition approach, evaluated with the cost approximation in Federgruen et al. (1984) or evaluated using our exact approach, the best conventional can-order policy, and the best generalized can-order policy. The percentages indicate the difference with the best generalized can-order policy.

	Decomposition (approx. eval.)	Decomposition (exact eval.)	Best can-order	Best generalized
Instance 1	88.71 (14.66%)	81.03 (4.74%)	77.51 (0.19%)	77.36
Instance 2	89.98 (11.45%)	83.62 (3.58%)	80.87 (0.17%)	80.73
Instance 3	71.53 (6.05%)	68.52 (1.58%)	67.80 (0.53%)	67.45

applying our greedy algorithm.

We find that the exact cost evaluation of the can-order policy identified by the decomposition approach performs extremely well, especially compared to the cost approximation initially proposed by Federgruen et al. (1984). Among others, this implies that it is not the decomposition approach that performs bad, but rather the cost approximation that is used to evaluate their can-order policy. Our results provide evidence that application of the decomposition approach provides decent results to instances where an exact evaluation remains infeasible. We also find that the best conventional can-order policy performs close to the generalized can-order policy. The generalization of the can-order policy differs only slightly from the conventional can-order policy and thus provides only marginal improvements. The conventional can-order policy, when applied with the optimal parameter set, performs very well.

6 Generalization of the decomposition approach

The decomposition approach proposed by Federgruen et al. (1984) decomposes the N-item problem into N independent single-item problems where, for each item i, discounted order opportunities arrive at a rate μ_i. Whereas Federgruen et al. (1984) rely on a cost approximation to optimize the parameters of the can-order policy, our method is capable to evaluate the costs in an exact way. Algorithm 2 shows how the can-order policy parameters of the N single-item problem are iteratively optimized using our exact cost evaluation method, where each single-item problem is considered as a two-item system consisting of item i and a virtual item j with parameters $c_j = 0$, $S_j = 1$, $\lambda_j = \mu_i$, $h_j = 0$, and $k_j = -K$.

Note that Algorithm 2 differs from the decomposition approach reported in Federgruen et al. (1984). Whereas the latter uses an approximate cost function to evaluate and optimize the N single-item problems, Algorithm 2 evaluates the cost function in an exact manner using

Algorithm 2: Exact decomposition approach with N single-item problems

```
Initialize $\mathcal{P}$, the can-order policy where $S_i = Q_i$, and $c_i = 0$, for all $i \in \mathbf{N}$;
Initialize $\mu_i$ using same logic as in decomposition approach for all $i \in \mathbf{N}$;
do
    Let done = true;
    forall the $i \in \mathbf{N}$ do
        Let $\mathcal{P}_i$ denote the two-item can-order policy with parameters $c_1 = c_i$, $c_2 = 0$,
        $S_1 = S_i$, $S_2 = 1$, $\lambda_1 = \lambda_i$, $\lambda_2 = \mu_i$, $h_1 = h_i$, $h_2 = 0$, $k_1 = k_i$, and $k_2 = -K$;
        forall the $S_i' \leq Q_i$ do
            forall the $c_i' < S_i'$ do
                Let $\mathcal{P}_i'$ denote the two-item can-order policy with parameters $c_1 = c_i'$,
                $c_2 = 0$, $S_1 = S_i'$, $S_2 = 1$, $\lambda_1 = \lambda_i$, $\lambda_2 = \mu_i$, $h_1 = h_i$, $h_2 = 0$, $k_1 = k_i$, and
                $k_2 = -K$;
                if $C(\mathcal{P}_i') < C(\mathcal{P}_i)$ then
                    Let $c_i = c_i'$, $S_i = S_i'$, and $\mathcal{P}_i = \mathcal{P}_i'$;
                    Let done = false;
    Update $\mathcal{P}$;
    Update $\mu_i$ using same logic as in decomposition approach for all $i \in \mathbf{N}$;
while done = false;
```

our embedded Markov-chain method described in Section 2. The resulting optimization of the can-order policy parameters leads to a different, and generally better, can-order policy.

In addition to that, our method is not restricted to the analysis of single-item problems, as it can evaluate the cost of two (or more) items in an exact way. A straightforward generalization of the decomposition approach uses $N(N-1)$ two-item problems, rather than N single-item problems. Algorithm 3 illustrates how for each item i, the can-order policy that minimizes the costs over all pairs $(i, j) : i \neq j \wedge i, j \in \mathbf{N}$ is identified; for each pair the three-item system —with a virtual third item representing discounted order opportunities— is analyzed using our embedded Markov-chain method. In a similar way we could also consider $N(N-1)(N-2)$ three-item problems, etc. As the multi-item problems are a better approximation of the N-item problem than a single-item problem, we expect the resulting can-order policies to be closer to the best can-order policy.

Table 7 summarizes the results of this approach to find the best can-order policy for the instances considered in Federgruen et al. (1984). The table reports the cost performance, obtained in an exact way using our embedded Markov-chain method, of the conventional decomposition approach of Federgruen et al. (1984) that relies on an approximate cost function to optimize the can-order policy parameters; the exact decomposition approach described by Algorithm 2 that relies on the exact cost evaluation to optimize the single-item

Algorithm 3: Exact decomposition approach with $N(N-1)$ two-item problems

```
Initialize $\mathcal{P}$, the can-order policy where $S_i = Q_i$, and $c_i = 0$, for all $i \in \mathbf{N}$;
Initialize $\mu_{ij}$ using same logic as in decomposition approach for all $i, j \in \mathbf{N}$;
do
    Let $\mathsf{done} = \mathsf{true}$;
    forall the $i \in \mathbf{N}$ do
        Let $C_{ij} = 0$;
        forall the $j \in \mathbf{N} \setminus \{i\}$ do
            Let $\mathcal{P}_{ij}$ denote the three-item can-order policy with parameters $c_1 = c_i$,
            $c_2 = c_j$, $c_3 = 0$, $S_1 = S_i$, $S_2 = S_j$, $S_3 = 1$, $\lambda_1 = \lambda_i$, $\lambda_2 = \lambda_j$, $\lambda_3 = \mu_i$,
            $h_1 = h_i$, $h_2 = h_j$, $h_3 = 0$, $k_1 = k_i$, $k_2 = k_j$, and $k_3 = -K$;
            $C_{ij} = C_{ij} + C(\mathcal{P}_{ij})$;
        forall the $S'_i \leq Q_i$ do
            forall the $c'_i < S'_i$ do
                Let $C'_{ij} = 0$;
                forall the $j \in \mathbf{N} \setminus \{i\}$ do
                    Let $\mathcal{P}'_{ij}$ denote the three-item can-order policy with parameters
                    $c_1 = c'_i$, $c_2 = c_j$, $c_3 = 0$, $S_1 = S'_i$, $S_2 = S_j$, $S_3 = 1$, $\lambda_1 = \lambda_i$, $\lambda_2 = \lambda_j$,
                    $\lambda_3 = \mu_i$, $h_1 = h_i$, $h_2 = h_j$, $h_3 = 0$, $k_1 = k_i$, $k_2 = k_j$, and $k_3 = -K$;
                    $C'_{ij} = C'_{ij} + C(\mathcal{P}'_{ij})$;
                if $C'_{ij} < C_{ij}$ then
                    Let $c_i = c'_i$, $S_i = S'_i$, and $\mathcal{P}_{ij} = \mathcal{P}'_{ij}$;
                    Let $\mathsf{done} = \mathsf{false}$;
    Update $\mathcal{P}$;
    Update $\mu_{ij}$ using same logic as in decomposition approach for all $i, j \in \mathbf{N}$;
while $\mathit{done} = \mathit{false}$;
```

Table 7: Cost performance (all evaluated using our exact approach) of the can-order policies obtained using different decomposition approaches. The conventional decomposition approach relies on a cost approximation to optimize the policy parameters. The exact decomposition approaches (single-item, two-item, and three-item) rely on an exact evaluation of the costs to optimize the policy. The number between brackets represents the percentage deviation from the best can-order policy.

	Decomposition (exact eval.)	Exact Decomp. (single-item)	Exact Decomp. (two-item)	Exact Decomp. (three-item)	Best can-order
Instance 1	81.03 (4.54%)	80.07 (3.30%)	78.10 (0.76%)	77.97 (0.59%)	77.51
Instance 2	83.62 (3.51%)	82.66 (2.22%)	82.16 (1.59%)	81.27 (0.49%)	80.87
Instance 3	68.52 (1.05%)	68.70 (1.33%)	68.04 (0.34%)	67.96 (0.24%)	67.80

problems; the exact decomposition approach described in Algorithm 3 considering two-item problems; the exact decomposition approach considering three-item problems; and the best can-order policy. Table 8 reports the corresponding can-order policy parameters obtained using the different decomposition approaches. It shows how our extended decomposition approach approximates the best can-order policy parameters better than the conventional decomposition approach, thereby reducing the gap with the best can-order policy to below 1%.

7 Extensions of the model

The DTMC analysis described in Section 2 assumes that orders are instantaneously received in inventory and inventories of all items are non-negative. In what follows, we illustrate a number of extensions of the base case. We limit ourselves to the discussion of: (1) backlog, (2) lost sales, (3) compound Poisson demand, and (4) nonzero lead times. Interestingly, none of these extensions increase the number of initial states. Among others, this means that the complexity of the problem remains the same no matter the extension. This contrasts with a traditional Markov-chain approach that requires more states for any of the extensions (and hence requires inversion of a bigger matrix in order to calculate the steady-state distribution).

7.1 Backlog

In case of backlog, the inventory of item i is allowed to drop below zero, and backlog costs b_i are incurred per unit of time. The cumulative inventory costs H_y in Eq. (6) are then defined by $H_y = H_y^+ + H_y^-$, with H_y^+ the expected cumulative holding costs and H_y^- the cumulative

Table 8: Optimized can-order policy parameters obtained by the different decomposition approaches.

	Instance 1		Instance 2		Instance 3	
	S	**c**	**S**	**c**	**S**	**c**
Decomposition (exact eval.)	22,15,17,15	14, 9,11, 9	23,15,17,15	12, 8,10, 8	18,12,13,12	7,5,6,5
Exact Decomp. (single-item)	17,11,13,11	8, 5, 6, 5	18,12,14,12	8, 5, 6, 5	16,11,12,11	5,3,4,3
Exact Decomp. (two-item)	17,11,14,11	10, 6, 8, 6	18,12,14,12	9, 5, 7, 5	16,11,12,11	6,4,5,4
Exact Decomp. (three-item)	17,11,14,11	11, 6, 9, 6	18,12,15,12	10, 6, 8, 6	15,11,12,11	6,4,5,4
Best can-order	18,11,16,11	13, 7,11, 7	18,11,16,11	12, 7,11, 7	15,10,12,10	7,4,6,4

backlog costs. The expected cumulative holding costs are given by:

$$H_y^+ = \lambda_{\mathbf{N}}^{-1} \sum_{x=1}^{X} P_{yx} \Delta_{yx} \left(\sum_{i=1}^{N} I_{yxi}^+ h_i \right), \tag{7}$$

with I_{yxi}^+ the expected inventory of item i when moving from initial state y to trigger state x:

$$I_{yxi}^+ = \begin{cases} I'_{xi} + \dfrac{I_{yi} - I'_{xi}}{2} & \text{if } I_{yi} > 0 \wedge I_{yi} \geq I_{xi} \wedge I'_{xi} \geq 0, \\ \dfrac{I_{yi}\left(I_{yi}+1\right)}{2\left(I_{yi} - I'_{xi} + 1\right)} & \text{if } I_{yi} > 0 \wedge I_{yi} \geq I_{xi} \wedge I'_{xi} < 0, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

The expected cumulative backlog costs are given by:

$$H_y^- = \lambda_{\mathbf{N}}^{-1} \sum_{x=1}^{X} P_{yx} \Delta_{yx} \left(\sum_{i=1}^{N} I_{yxi}^- b_i \right), \tag{9}$$

where I_{yxi}^- is the expected backlog of item i when moving from initial state y to trigger state x:

$$I_{yxi}^- = \begin{cases} \dfrac{I'_{xi}\left(I'_{xi}-1\right)}{2\left(I_{yi} - I'_{xi} + 1\right)} & \text{if } I'_{xi} < 0 \wedge I_{yi} \geq I_{xi} \wedge I_{yi} > 0, \\ \dfrac{I'_{xi} - I_{yi}}{-2} - I_{yi} & \text{if } I'_{xi} < 0 \wedge I_{yi} \geq I_{xi} \wedge I_{yi} \leq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Table 9: Steady-state probability, expected transition time, expected cumulative holding cost, expected cumulative backlog cost, and expected lost sales cost for all $y : y \in \mathcal{Y}$

y	π_y	T_y	H_y^+	H_y^-	W_y
$\{7,8\}$	0.8460	$9/28$	75.4559	6.9813	54.4933
$\{7,7\}$	0.0027	$8/28$	63.8323	6.8208	54.1629
$\{7,6\}$	0.0074	$7/28$	53.1111	6.6721	53.6426
$\{7,5\}$	0.0151	$6/28$	43.3335	6.5420	52.9748
$\{7,4\}$	0.0254	$5/28$	34.5218	6.4358	52.2311
$\{7,3\}$	0.0372	$4/28$	26.6800	6.3561	51.5017
$\{6,8\}$	0.0219	$8/28$	68.6207	7.1952	54.9339
$\{5,8\}$	0.0442	$7/28$	61.8342	7.4302	55.0366

To illustrate how to model backlog, we revisit the best can-order policy of the example introduced in Section 2, and assume that items re-order if the inventory level hits -2. In addition, we have $h_1^b = h_2^b = 100$. The expected cumulative holding and backlog cost are $H^+ = 133.5324$ and $H^- = 13.0584$, respectively. For each initial state $y : y \in \mathcal{Y}$, Table 9 lists the: (1) steady-state probability, (2) expected transition time, (3) the expected cumulative holding cost, and (4) the expected cumulative backlog cost.

7.2 Lost sales

In case of lost sales, the inventory of item i is not allowed to drop below zero, and a cost w_i is incurred for each unit of unmet demand. The cost of lost sales is given by:

$$W_y = \sum_{x=1}^{X} P_{yx} \sum_{i=1}^{N} \max\{0, -I_{xi}\} w_i, \tag{11}$$

and may be integrated in the total cost function Eq. (6) as follows:

$$C(\mathcal{P}) = \frac{\sum_{y=1}^{Y} \pi_y (O_y + H_y + W_y)}{\sum_{y=1}^{Y} \pi_y T_y}, \tag{12}$$

with $H_y = H_y^+$ defined above in (7).

If we revisit the example, and assume $w_1 = w_2 = 25$, the costs of lost sales amount to 101.9418. Table 9 also reports the expected lost sales cost for each initial state $y : y \in \mathcal{Y}$.

7.3 Compound Poisson demand

In case of compound Poisson demand, the demand for item i can be any integer number of units u with distribution function $f_i(u)$. Unfortunately, we can no longer use the multinomial distribution to determine P_{yx}. Instead, for each initial state y, we define a CTMC that has initial state $\mathbf{Y}_y$ and absorbing states $\mathcal{X}$. Using this CTMC, we can once more determine: (1) P_{yx}, (2) T_y, and (3) H_y.

For instance, consider the example system, and imagine that both items face a compound Poisson demand with $f_1(1) = f_1(2) = f_2(1) = f_2(2) = 1/2$ (i.e., both items face a demand of either 1 or 2 units, both with a 50% probability). In addition, we assume that an order is triggered as soon as the inventory of an item depletes. Next, we focus on initial state $\mathbf{Y}_6 = \{7, 3\}$. The CTMC associated with $\mathbf{Y}_6 = \{7, 3\}$ has absorbing states $\mathbf{X}_1 = \{0, 3\}$, $\mathbf{X}_2 = \{-1, 3\}$, $\mathbf{X}_3 = \{0, 2\}$, $\mathbf{X}_4 = \{-1, 2\}$, $\mathbf{X}_5 = \{0, 1\}$, $\mathbf{X}_6 = \{-1, 1\}$, $\mathbf{X}_7 = \{7, 0\}$, $\mathbf{X}_8 = \{7, -1\}$, $\mathbf{X}_9 = \{6, 0\}$, $\mathbf{X}_{10} = \{6, -1\}$, $\mathbf{X}_{11} = \{5, 0\}$, $\mathbf{X}_{12} = \{5, -1\}$, $\mathbf{X}_{13} = \{4, 0\}$, $\mathbf{X}_{14} = \{4, -1\}$, $\mathbf{X}_{15} = \{3, 0\}$, $\mathbf{X}_{16} = \{3, -1\}$, $\mathbf{X}_{17} = \{2, 0\}$, $\mathbf{X}_{18} = \{2, -1\}$, $\mathbf{X}_{19} = \{1, 0\}$, and $\mathbf{X}_{20} = \{1, -1\}$. The CTMC is given in Table 10, and allows us to determine $P_{6,x}$ and $P_6(z)$, where z is a transient state (i.e., a state that is visited only once; $\{7, 3\}$, $\{7, 2\}$, and $\{7, 1\}$ are example transient states). In turn, probabilities $P_6(z)$ can be used to determine both T_6 as well as H_6. For instance, the probability to visit state $\{7, 3\}$ is $P_6(\{7, 3\}) = 1$. We expect to stay $\lambda_{\mathbf{N}}^{-1}$ time units in state $\{7, 3\}$, and hence, its contribution to T_6 equals $P_6(\{7, 3\})\,\lambda_{\mathbf{N}}^{-1} = 1/28$. Analogously, we can determine its contribution to H_6 as $P_6(\{7, 3\})\,\lambda_{\mathbf{N}}^{-1}(7h_1 + 3h_2) = 153/28$. In general, for initial state y, we have:

$$T_y = \lambda_{\mathbf{N}}^{-1} \sum_{z=1}^{Z} P_y(z), \tag{13}$$

$$H_y = \lambda_{\mathbf{N}}^{-1} \sum_{z=1}^{Z} P_y(z) \sum_{i=1}^{N} I_{zi} h_i, \tag{14}$$

where Z denotes the number of transient states, and I_{zi} the inventory of item i in transient state z.

7.4 Nonzero lead times

Let L denote the lead time. In what follows, we assume that L is deterministic. Note, however, that it is possible to model L as a random variable. We assume that only one order can be outstanding at any point in time. In order to determine the steady-state distribution $\boldsymbol{\pi}$, we introduce transient state $z \in \{1, 2, \dots, Z\}$, in which we end up after an order has

Table 10: Infinitesimal generator of the CTMC that is associated with initial state $\mathbf{Y}_6 = \{7, 3\}$, and that can be used to determine P_{6x}, T_6, and H_6 (the initial state is printed in purple and the trigger states are printed in blue). The table also lists probabilities P_{6x}.

I_1	I_2	**7**	7	7	**7**	**7**	6	6	6	**6**	**6**	$\cdots$	2	2	2	**2**	**2**	1	1	1	**1**	**1**	**0**	**0**	**0**	**-1**	**-1**	**-1**	P_{6x}
		3	2	1	**0**	**-1**	3	2	1	**0**	**-1**	$\cdots$	3	2	1	**0**	**-1**	3	2	1	**0**	**-1**	**3**	**2**	**1**	**3**	**2**	**1**	
7	**3**	$-\lambda_\mathbf{N}$	$\frac{\lambda_2}{2}$	$\frac{\lambda_2}{2}$	0	0	$\frac{\lambda_1}{2}$	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	2	0	$-\lambda_\mathbf{N}$	$\frac{\lambda_2}{2}$	$\frac{\lambda_2}{2}$	0	0	$\frac{\lambda_1}{2}$	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	1	0	0	$-\lambda_\mathbf{N}$	$\frac{\lambda_2}{2}$	$\frac{\lambda_2}{2}$	0	0	$\frac{\lambda_1}{2}$	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	**0**	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1866
7	**-1**	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1050
6	3	0	0	0	0	0	$-\lambda_\mathbf{N}$	$\frac{\lambda_2}{2}$	$\frac{\lambda_2}{2}$	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	2	0	0	0	0	0	0	$-\lambda_\mathbf{N}$	$\frac{\lambda_2}{2}$	$\frac{\lambda_2}{2}$	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	1	0	0	0	0	0	0	0	$-\lambda_\mathbf{N}$	$\frac{\lambda_2}{2}$	$\frac{\lambda_2}{2}$	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	**0**	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0850
6	**-1**	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0500
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
2	3	0	0	0	0	0	0	0	0	0	0	$\cdots$	$-\lambda_\mathbf{N}$	$\frac{\lambda_2}{2}$	$\frac{\lambda_2}{2}$	0	0	$\frac{\lambda_1}{2}$	0	0	0	0	$\frac{\lambda_1}{2}$	0	0	0	0	0	
2	2	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	$-\lambda_\mathbf{N}$	$\frac{\lambda_2}{2}$	$\frac{\lambda_2}{2}$	0	0	$\frac{\lambda_1}{2}$	0	0	0	0	$\frac{\lambda_1}{2}$	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	$-\lambda_\mathbf{N}$	$\frac{\lambda_2}{2}$	$\frac{\lambda_2}{2}$	0	0	$\frac{\lambda_1}{2}$	0	0	0	0	$\frac{\lambda_1}{2}$	0	0	0	
2	**0**	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0367
2	**-1**	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0234
1	3	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	$-\lambda_\mathbf{N}$	$\frac{\lambda_2}{2}$	$\frac{\lambda_2}{2}$	0	0	$\frac{\lambda_1}{2}$	0	0	$\frac{\lambda_1}{2}$	0	0	
1	2	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	$-\lambda_\mathbf{N}$	$\frac{\lambda_2}{2}$	$\frac{\lambda_2}{2}$	0	0	$\frac{\lambda_1}{2}$	0	0	$\frac{\lambda_1}{2}$	0	
1	1	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	$-\lambda_\mathbf{N}$	$\frac{\lambda_2}{2}$	$\frac{\lambda_2}{2}$	0	0	$\frac{\lambda_1}{2}$	0	0	$\frac{\lambda_1}{2}$	
1	**0**	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0270
1	**-1**	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0174
0	**3**	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0136
0	**2**	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0171
0	**1**	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0306
-1	**3**	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0053
-1	**2**	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0072
-1	**1**	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0131

arrived. Next, define P_{yz}, the probability to transition from an initial state y to a transient state z (where $(\lambda L)^m \, \mathrm{e}^{-\lambda L} \, (m!)^{-1}$ is the probability of having m successes in an interval of length L given a Poisson process with rate parameter λ):

$$P_{yz} = \begin{cases} \displaystyle\prod_{i=1}^{N} \frac{(\lambda_i L)^{(I_{yi} - I_{zi})} \, \mathrm{e}^{-\lambda_i L}}{(I_{yi} - I_{zi})!} & \text{if } I_{yi} \geq I_{zi} \quad \forall i \in \mathbf{N}, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Note that, in any initial state $y : y \in \mathcal{Y}$, an order is always on its way (hence, after arrival of the order in transient state z, item i has incurred a demand of $I_{yi} - I_{zi}$ units, where I_{zi} is the inventory level of item i in transient state z). Next, define P_{zx}, the probability to transition from transient state z to trigger state x. If, for any $x : x \in \mathcal{X}$, $I_{zi} = I_{xi}$ for all $i : i \in \mathbf{N}$, $P_{zx} = 1$ and $P_{zx'} = 0$ for all $x' : x' \in \mathcal{X} \setminus \{x\}$. If no such trigger state x can be found, P_{zx} may be defined as follows (once more using the multinomial distribution):

$$P_{zx} = \begin{cases} \dfrac{(\Delta_{zx} - 1)!}{\prod_{i=1}^{N} (I_{xi} - I'_{xi})!} \displaystyle\prod_{i=1}^{N} \left(\frac{\lambda_i}{\lambda_{\mathbf{N}}} \right)^{(I_{zi} - I_{xi})} & \text{if } I_{zi} \geq I_{xi} \quad \forall i \in \mathbf{N}, \\ 0 & \text{otherwise,} \end{cases} \tag{16}$$

where $\Delta_{zx} = \sum_{i=1}^{N} (I_{zi} - I_{xi})$. P_{yx} may now be obtained as $P_{yz} \sum_{x=1}^{X} P_{zx}$.

The expected time until the subsequent order associated with initial state y is given by:

$$T_y = L + \lambda_{\mathbf{N}}^{-1} \sum_{z=1}^{Z} P_{yz} \left(\sum_{x=1}^{X} P_{zx} \Delta_{zx} \right). \tag{17}$$

In addition, define H_{zx}, the expected cumulative holding cost when making a transition from transient state z to trigger state x:

$$H_{zx} = \begin{cases} 0 & \text{if } I_{zi} = I_{xi} \forall i \in \mathbf{N}, \\ \lambda_{\mathbf{N}}^{-1} \Delta_{zx} \displaystyle\sum_{i=1}^{N} \left(I'_{xi} + \frac{I_{zi} - I'_{xi}}{2} \right) h_i & \text{otherwise.} \end{cases} \tag{18}$$

In addition, let H_x denote the expected cumulative inventory and backlog cost from the moment an order is triggered in trigger state x until the next order:

$$H_x = L \sum_{z=1}^{Z} P_{\mathcal{P}(x)z} \left(\sum_{i=1}^{N} \left(I_{xzi}^{+} h_i + I_{xzi}^{-} h_b \right) \right), \tag{19}$$

Table 11: Steady-state probability, expected transition time, expected order cost, and expected cumulative holding and backlog costs for all $y : y \in \mathcal{Y}$ in case of nonzero lead time

y	π_y	T_y	O_y	H_y
$\{7,8\}$	0.6506	0.4267	47.6402	59.9705
$\{7,7\}$	0.0119	0.3885	48.8153	50.2738
$\{7,6\}$	0.0249	0.3447	49.5270	41.2983
$\{7,5\}$	0.0401	0.2956	49.6968	33.1138
$\{7,4\}$	0.0550	0.2422	49.3294	25.8144
$\{7,3\}$	0.0678	0.1865	48.5414	19.6236
$\{6,8\}$	0.0609	0.3942	46.0734	53.6146
$\{5,8\}$	0.0888	0.3517	43.6824	46.9865

where I^+_{xzi} and I^-_{xzi} are the expected cumulative inventory and backlog of item i when making a transition from trigger state x to transient state z, respectively (obtained in analogy with Eq. (8) and Eq. (10)). Next, the expected cumulative inventory and backlog costs associated with initial state y may be obtained as follows (note that lost sales, rather than backlog, can also be modelled):

$$H_y = \left(\sum_{x=1}^{X} P_{yx} H_x \right) + \left(\sum_{z=1}^{Z} P_{yz} \left(\sum_{x=1}^{X} P_{zx} H_{zx} \right) \right). \tag{20}$$

If, in the example, we assume a lead time $L = 2\lambda_{\mathbf{N}}^{-1} = 1/14$, the holding and backlog costs amount to 135.8784. Table 11 presents the steady-state probability, the expected transition time, the expected order cost, and the expected cumulative holding and backlog costs for all $y : y \in \mathcal{Y}$.

8 Conclusions

We present an exact and efficient solution procedure that is novel to the inventory management (and by extension the joint replenishment) literature. Alternative exact methods rely on conventional Markov-chain analysis to evaluate inventory policies, combined by a search procedure to optimize the policy parameters, or relies on Markov decision processes to identify the optimal policy. Thanks to the substantial reduction in state space, our method allows an exact analysis for instances that until now could only be evaluated using approximation procedures. We also characterize the optimal joint replenishment policy as a generalization of the can-order policy. We find that the conventional can-order policy, when used with the optimal parameter set, performs very well. Our embedded Markov-chain model now makes it possible to identify this optimal parameter set due to its exact cost evaluation. Although our method is not capable to solve instances with a large number of items, we provide an excellent heuristic that improves and extends the conventional decomposition approach and we demonstrate its good performance. Future research can be devoted to other inventory management applications that can benefit from our Markov-chain analysis, or applications of joint replenishment in other environments, such as dual transport mode problems.

Appendix

References

Atkins, D. R., P. O. Iyogun. 1997. Periodic versus 'can-order' policies for coordinated multi-item inventory systems. *Management Sci.*, **34**(6), 791–796.

Balintfy, J. L. 1964. On a basic class of multi-item inventory problems. *Management Sci.*, **10**(2), 287–297.

Creemers, S., G. Woumans, R. Boute, J. Beliën. 2017. Tri-Vizor uses an efficient algorithm to identify collaborative shipping opportunities. *INFORMS J. Appl. Anal.*, **47**(3), 195–277.

Federgruen, A., H. Groenevelt, H. C. Tijms. 1984. Coordinated replenishments in a multi-item inventory system with compound Poisson demands. *Management Sci.*, **30**(3), 344–357.

Feng, H., Wu, Q., Muthuraman, K., Deshpande, V. 2014. Replenishment policies for multi-product stochastic inventory systems with correlated demand and joint-replenishment costs. *Prod. Oper. Management*, **24**(4), 647–664.

Goyal, S. K., A. T. Satir. 2008. Joint replenishment inventory control: Deterministic and stochastic models. *Eur. J. Oper. Res.*, **38**(1), 2–13.

Ignall, E. 1969. Optimal continuous review policies for two product inventory systems with joint setup costs. *Management Sci.*, **15**(5), 278–283.

Kayış, E., T. Bilgiç, D. Karabulut. 2008. A note on the can-order policy for the two-item stochastic joint-replenishment problem. *IIE Trans.*, **40**(1), 84–92.

Khouja, M., S. Goyal. 2008. A review of the joint replenishment problem literature: 19892005. *Eur. J. Oper. Res.*, **186**(1), 1–16.

Kiesmüller, G. P. 2010. Multi-item inventory control with full truckloads: A comparison of aggregate and individual order triggering. *Eur. J. Oper. Res.*, **200**(1), 54–62.

Melchiors, P. 2002. Calculating can-order policies for the joint replenishment problem by the compensation approach. *Eur. J. Oper. Res.*, **141**(3), 587–595.

Padilla Tinoco, S. V., S. Creemers, R. N. Boute. 2017. Collaborative shipping under different cost-sharing agreements. *Eur. J. Oper. Res.*, **263**(3), 827–837.

Pantumsinchai, P. 1992. A comparison of three joint ordering inventory policies. *Decis. Sci.*, **23**(1), 111–127.

Schultz, H., S. G. Johansen. 1999. Can-order policies for coordinated inventory replenishment with Erlang distributed times between ordering. *Eur. J. Oper. Res.*, **113**(1), 30–41.

Table 12: Infinitesimal generator of the CTMC used by a traditional Markov-chain approach to characterize a can-order policy with $S_1 = 7, S_2 = 8, c_1 = 4, c_2 = 2, s_1 = s_2 = 0$.

I_1	I_2	7	7	⋯	7	6	6	⋯	6	5	5	⋯	5	4	4	⋯	4	3	3	⋯	3	2	2	⋯	2	1	1	⋯	1
		8	7	⋯	1	8	7	⋯	1	8	7	⋯	1	8	7	⋯	1	8	7	⋯	1	8	7	⋯	1	8	7	⋯	1
7	8	$-\lambda_{\mathbf{N}}$	λ_2	⋯	0	λ_1	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0
7	7	0	$-\lambda_{\mathbf{N}}$	⋯	0	0	λ_1	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
7	1	λ_2	0	⋯	$-\lambda_{\mathbf{N}}$	0	0	⋯	λ_1	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0
6	8	0	0	⋯	0	$-\lambda_{\mathbf{N}}$	λ_2	⋯	0	λ_1	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0
6	7	0	0	⋯	0	0	$-\lambda_{\mathbf{N}}$	⋯	0	0	λ_1	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
6	1	0	0	⋯	0	λ_2	0	⋯	$-\lambda_{\mathbf{N}}$	0	0	⋯	λ_1	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0
5	8	0	0	⋯	0	0	0	⋯	0	$-\lambda_{\mathbf{N}}$	λ_2	⋯	0	λ_1	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0
5	7	0	0	⋯	0	0	0	⋯	0	0	$-\lambda_{\mathbf{N}}$	⋯	0	0	λ_1	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
5	1	0	0	⋯	0	0	0	⋯	0	λ_2	0	⋯	$-\lambda_{\mathbf{N}}$	0	0	⋯	λ_1	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0
4	8	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	$-\lambda_{\mathbf{N}}$	λ_2	⋯	0	λ_1	0	⋯	0	0	0	⋯	0	0	0	⋯	0
4	7	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	$-\lambda_{\mathbf{N}}$	⋯	0	0	λ_1	⋯	0	0	0	⋯	0	0	0	⋯	0
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
4	1	λ_2	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	$-\lambda_{\mathbf{N}}$	0	0	⋯	λ_1	0	0	⋯	0	0	0	⋯	0
3	8	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	$-\lambda_{\mathbf{N}}$	λ_2	⋯	0	λ_1	0	⋯	0	0	0	⋯	0
3	7	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	$-\lambda_{\mathbf{N}}$	⋯	0	0	λ_1	⋯	0	0	0	⋯	0
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
3	1	λ_2	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	$-\lambda_{\mathbf{N}}$	0	0	⋯	λ_1	0	0	⋯	0
2	8	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	$-\lambda_{\mathbf{N}}$	λ_2	⋯	0	λ_1	0	⋯	0
2	7	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	$-\lambda_{\mathbf{N}}$	⋯	0	0	λ_1	⋯	0
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
2	1	λ_2	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	$-\lambda_{\mathbf{N}}$	0	0	⋯	λ_1
1	8	λ_1	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	$-\lambda_{\mathbf{N}}$	λ_2	⋯	0
1	7	0	λ_1	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	$-\lambda_{\mathbf{N}}$	⋯	0
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
1	1	$\lambda_{\mathbf{N}}$	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	0	0	0	⋯	$-\lambda_{\mathbf{N}}$

Silver, E. A. 1974. A control system for coordinated inventory replenishment. *Internat. J. Production Res.*, **12**(6), 647–671.

van Eijs, M. J. G. 1994. On the determination of the control parameters of the optimal can-order policy. *Zeitschrift für Operations Research*, **39**(3), 289–304.

Viswanathan, S. 1997. Periodic review (s, S) policies for joint replenishment inventory systems. *Management Sci.*, **43**(10), 1447–1454.

Viswanathan, S. 2007. An algorithm for determining the best lower bound for the stochastic joint replenishment problem. *Oper. Res.*, **55**(5), 992–996.

Zheng, Y-S. 1994. Optimal control policy for stochastic inventory systems with Markovian discount opportunities. *Oper. Res.*, **42**(4), 721–738.