EURANDOM PREPRINT SERIES
2011-010

Approximate Evaluation of Multi-Location Inventory Models with Lateral Transshipments and Hold Back Levels

A.C.C. van Wijk, I.J.B.F. Adan, G.J. van Houtum
ISSN 1389-2355

Approximate Evaluation of Multi–Location Inventory Models with Lateral Transshipments and Hold Back Levels

A.C.C. van Wijk*[1,2,3], I.J.B.F. Adan[1,3], and G.J. van Houtum[2,3]

Eindhoven University of Technology, Eindhoven, The Netherlands

[1]Department of Mathematics and Computer Science
[2]School of Industrial Engineering
[3]EURANDOM

February 23, 2011

Abstract

We consider a continuous-time, single-echelon, multi-location inventory model with Poisson demand processes. In case of a stock-out at a local warehouse, a demand can be fulfilled via a lateral transshipment (LT). Each warehouse is assigned a predetermined sequence of other warehouses where it will request for an LT. However, a warehouse can hold its last part(s) back from such a request. This is called a hold back pooling policy, where each warehouse has hold back levels determining whether a request for and LT by another warehouse is satisfied. We are interested in the fractions of the demand satisfied from stock (fill rate), satisfied via a lateral transshipment, and via an emergency shipment from an external source. From this the average costs of a policy can be determined. We present two approximation algorithms for the evaluation of a given policy, approximating the above mentioned fractions. The first one, the Poisson overflow algorithm, is an extension of algorithms known in the literature. The second one, the On/Off overflow algorithm is new and more sophisticated. Instead of approximating the stream of LT-requests from a warehouse as a Poisson process, we use an interrupted Poisson process. This is a process that is turned alternatingly On and Off for exponentially distributed durations. In a numerical study we show that both algorithms perform very well. The On/Off algorithm is significantly more accurate than the Poisson algorithm, but requires longer computation times.

Keywords: inventory, lateral transshipment, approximation algorithm, interrupted Poisson process.

* *Corresponding author:* P.O. Box 513, 5600MB Eindhoven, The Netherlands, a.c.c.v.wijk@tue.nl

1 Introduction

Pooling of inventory has proven to be an interesting option for costs reductions and service level improvements. By sharing inventory between local warehouses such pooling benefits can be achieved. In case of a stock-out at one warehouse, a demand can be satisfied via a stock transfer from another warehouse. These stock transfers, which happen within the same echelon, are called *lateral transshipments*. One possible strategy for this is the so-called *complete pooling* policy, in which the local warehouses basically act as being one. Complete pooling however, might not always be optimal. In this paper we study the so-called *hold back* pooling policy. This policy has recently been proven to be optimal under certain conditions in a two location setting (Van Wijk et al. 2009). The policy was introduced by Xu et al. (2003) in a periodic review setting, and also arose to be optimal in a related two location problem (Archibald et al. 1997). Under a hold back pooling policy a warehouse can hold back its last part(s) in stock from a lateral transshipment request from another warehouse. The *hold back levels* of the warehouses determine how many parts are held back. For determining the optimal hold back levels, evaluation of the costs of a given setting is necessary. These costs can be calculated when one knows the fractions of the demand that are satisfied from stock (fill rate), satisfied via a lateral transshipment, and satisfied via an emergency shipment. For this, we present two approximation algorithms which can be used for evaluation. Hence, these algorithms facilitate the search for optimal hold back and base stock levels. The distinct feature of both algorithms compared to current algorithms given in the literature, is that they can handle hold back levels. Moreover, our second algorithm approximates the lateral transshipments requests between the warehouses more precisely than current algorithms. These are commonly approximated by a Poisson process (Axsäter 1990, Alfredsson and Verrijdt 1999, Kukreja et al. 2001, Kutanoglu 2008, Kranenburg and Van Houtum 2009, Reijnen et al. 2009), where we use a more accurate On/Off Poisson process. This improves the accuracy of the results.

In the present paper, we consider an inventory model consisting of N local warehouses, executing a base stock policy with one-for-one replenishments. The demand at each of the warehouses follows a Poisson process and the replenishment times are exponentially distributed. In case of a positive stock level, an incoming demand is directly satisfied from stock. In

case of a stock-out, there are two possibilities: either the demand is satisfied via a lateral transshipment from one of the other local warehouses, or the demand is lost for the local warehouse. In the latter case it has to be fulfilled at higher costs from either a central warehouse or another external source (having ample capacity). We refer to this as an *emergency shipment*. However, lost sales also fit in the presented model. A local warehouse is only willing to hand over a part if it has sufficient inventory, that is, if its inventory level is above a certain threshold, called the *hold back level*. This hold back level can depend on the location the LT request originates from. Furthermore, each warehouse follows a prescribed sequence of other warehouses it will contact for a lateral transshipment. Note that complete pooling, a strategy often assumed in literature, is a special case of this.

The given model is motivated by a spare parts inventory system, which services an installed base of technically advanced machines. As downtimes of these machines are very expensive, ready-for-use spare parts are kept in stock to be able to quickly respond to a failure of a machine. Typically for these settings, demand rates are low and spare parts may be costly. Defective parts are returned to the warehouse, repaired and added back to the inventory. Because of the expensive downtimes, back-orders are not allowed. For demands not satisfied, a costly emergency repair procedure has to be carried out. In such a setting, Kranenburg and Van Houtum (2009) show that based on data of the company ASML, an original equipment manufacturer in the semiconductor industry, the use of lateral transshipments may lead to a 50% cost reduction in comparison to no use of LTs (while keeping the service levels the same). Also Robinson (1990) shows that substantial cost savings can be realized by the use of lateral transshipments, even when the transportation costs are high. Moreover, the model described here applies not only to spare parts, but in general to multi-location, single-echelon inventory systems.

As inventory pooling can reduce costs and improve service levels, a lot of research has been devoted to the use of lateral transshipments, see Wong et al. (2006) and Paterson et al. (2011) for overviews. There are many options for the decisions on when to apply lateral transshipments. Generally, we distinguish between *complete pooling* and *partial pooling*. In the first case, all local warehouses act as being one: a demand is only lost in case all are stocked out. For partial pooling all kinds of restrictions are possible, e.g. lateral transshipments can

only take place between geographically nearby warehouses (Caggiano et al. 2009, Kutanoglu 2008, Kutanoglu and Mahajan 2009, Reijnen et al. 2009), the lateral transshipments might be executed in only one way (Axsäter 2003, Liu and Lee 2007, Olsson 2010), not all inventory has to be shared (Van Wijk et al. 2009, Xu et al. 2003). We take these restrictions into account in the following way. Firstly, each warehouse is assigned a sequence of warehouses it consults for a lateral transshipment. Reijnen et al. (2009) motivate this by a time constraint on the fulfillment of a demand: only warehouses close enough are consulted. Another motivation may be the transport facilities nearby some warehouses. In this way, also transshipments in one-direction only can be taken into account. Next to this, we use a *hold back policy* for the pooling of inventory. This policy was introduced by Xu et al. (2003) in a periodic review setting. In this case the outgoing lateral transshipments are limited by *hold back levels*. Only when the inventory of a warehouse is above its hold back level, it is willing to satisfy a lateral transshipment request. That is, a warehouse can hold back its last part(s) on stock. We focus here on a continuous review model, for which we consider an equivalent policy. We refer to this as a hold back policy as well, as this will not cause any confusion. Note that by the combination of these partial pooling options, we have a very general form of partial pooling: restrictions on the warehouses between which lateral transshipments take place (including one directional transshipments), and not all inventory has to be shared. This policy includes complete pooling as a special case, when each warehouse can request for a lateral transshipment from all other warehouses and all hold back levels are set to zero.

Lateral transshipments limited by holding back inventory is mainly considered in decentralized inventory models (Zhao et al. 2006, 2008). In such a setting, the local warehouses are independently owned and operated, which gives a game theoretical setting. Another reason for holding back parts is a periodic review setting. Based on the remaining time until a scheduled replenishment, the decision is taken if a lateral transshipment takes places (Archibald et al. 1997). This also occurs in a periodic review setting when the replenishment lead times are non-zero (Tagaras and Cohen 1992). We, however, concentrate on a continuous review model under central control.

Our incentive for the introduction of hold back levels is the recent work in Van Wijk et al. (2009), in which it is proven for two local warehouses that the optimal lateral transship-

ment policy structure is a hold back policy, under two (sufficient) conditions on the cost parameters. These conditions are typically satisfied when the lateral transshipment costs are non-negligible, and the emergency shipment costs at both locations are not too asymmetric. The setting assumed is identical to the setting as presented here. The benefit of holding back inventory occurs when a warehouse has only one or a few parts left in stock. When handing e.g. the last part out to a lateral transshipment request, costs have to be made for this. This warehouse is stocked out until the next replenishment. When it faces a demand during this time, this demand has to be satisfied either via a lateral transshipment or emergency shipment. In both cases, more costs have to be made than when the first lateral transshipment request was refused.

From the results of Van Wijk et al. (2009) for two locations, we might expect such a policy to be optimal too, or at least to perform well, for a multi-location setting. Hence, in this work we assume a hold back policy. Although exact evaluation and optimization is theoretically possible via Markov chain analysis, this is infeasible for large instances by the curse of dimensionality, and calculation times explode when trying to optimize the hold back and base stock levels. Consequently, there is a need for a good and fast approximation of the performance characteristics: the fractions of the demand that are fulfilled directly from stock, via a lateral transshipment or via an emergency shipment. We present two algorithms for this purpose, which evaluate the performance characteristics when the hold back and base stock levels are given. This can then be used in heuristic optimization procedures for the hold back and base stock levels themselves.

The presented approximation algorithms are related to the approximate evaluation method as described in Axsäter (1990). He decomposes the network of local warehouses into individual local warehouses. The lateral transshipments between them are modeled as overflow demand streams, which are approximated by Poisson processes. In an iteration the rates of these Poisson streams and the performance characteristics of the individual local warehouses, are alternately updated. Similar algorithms are used in a.o. Alfredsson and Verrijdt (1999), Kukreja et al. (2001), Kutanoglu (2008), Kranenburg and Van Houtum (2009), Reijnen et al. (2009), each focusing on a different setting. For example, in Kranenburg and Van Houtum (2009) a pooling structure is considered with so-called main and regular warehouses. Between

the mains, a complete pooling policy is assumed. As a consequence, a demand is only lost if all main warehouses are out-of-stock. Hence the stock-out probability can be derived exactly by the Erlang loss function. Reijnen et al. (2009) use a pooling structure based on the geographical locations of the local warehouses, where lateral transshipments can only be executed between nearby warehouses.

Our model extends these earlier models, as we allow for all earlier studied options for partial pooling. Moreover, in our second algorithm we use a more accurate approximation. Instead of assuming the overflow demand streams to be Poisson processes, we approximate them by a Poisson process that can be turned On and Off, known as *interrupted Poisson processes* (cf. Kuczura 1973). When a warehouse has parts in stock, the demand overflow to another local warehouse where it requests for lateral transshipments, is turned Off. During a stock-out, however, the demand overflow is turned On and follows a Poisson process. When the consulted warehouse is that low on inventory that it does not fulfill lateral transshipment requests, the overflow demand stream is routed to a next warehouse, etc. We approximate the durations of these On and Off times to be exponentially distributed. We call this the *On/Off overflow algorithm*. We call our first algorithm, using an ordinary Poisson overflow process, the *Poisson overflow algorithm*.

The lateral transshipment problem as studied here is also related to models used in telecommunication systems, especially in call centers (see Gans et al. (2003) for an overview). In these systems calling customers can be handled by different so-called operators or groups of operators. The inventory in our model is the equivalent of the operators in these call center models. Certain types of calls can only be handled by a subset of operators having appropriate skills. This leads to skill-based-routing of the customers. Typically, the overflow of customers is only one-way. Here we would have an X-design (cf. Gans et al. 2003) as every incoming demand can basically be fulfilled from every warehouse. For these designs, however, hardly any results seem to known in the call center literature. Next to that, there is a difference in the way rerouted customers/demands are handled. In call centers rerouting a customer would lead to a slower service rate for that customer, where in a lateral transshipment model, rerouting of a demand brings along direct costs, but the service (i.e. replenishment) rate remains equal.

This paper contributes to the literature in the following way. We study a model with hold

back levels. We present two approximate evaluation algorithms incorporating this feature. The first one uses Poisson overflow streams and extends earlier algorithms in the literature. Our main contribution is the second algorithm. In that we use On/Off Poisson processes, which are more precise than the Poisson processes. By a numerical study we show that the approximation produced by this algorithm is very accurate, while still being efficient in terms of computation time. The algorithms include complete pooling as a special case.

The outline is as follows. We start by introducing the model and notation used in Section 2. We also briefly discuss the exact analysis. In Section 3 we present the two approximation algorithms: the *Poisson overflow algorithm* and the *On/Off overflow algorithm*. In Section 4 a numerical study is conducted to test the performance of the On/Off overflow algorithm, using the Poisson overflow algorithm as the benchmark. We also compare the outcomes to exact evaluation. Finally we draw conclusions in Section 5.

2 Model and notation

We consider an inventory system with N local warehouses, numbered $i = 1, 2, \ldots, N$. These warehouses keep a single, consumable stock keeping unit (SKU) on stock. Each warehouse executes a basestock policy, with basestock level $S_i \in \mathbb{N} \cup \{0\}$, $i = 1, \ldots, N$, and one-for-one replenishments. The actual stock level (on hand stock) is denoted by $x_i \in \{0, 1, \ldots, S_i\}$. Define $\underline{x} = \{x_1, \ldots, x_N\}$. In case of a stock-out, a demand can be fulfilled from another warehouse. In this case, a part is transshipped from a warehouse with positive on hand stock to the warehouse where the demand arose. This is called a *lateral transshipment* (LT). Each warehouse has a set of hold back levels, determining if lateral transshipment requests form other warehouses will be accepted. Let $h_{i,j} \in \{0, \ldots, S_i\}$ be the hold back level at warehouse i for accepting an LT request from warehouse $j \neq i$. Only if $x_i > h_{i,j}$ the request is fulfilled. Define $h_{i,i} = 0$ and let $\underline{h}_i = (h_{i,1}, \ldots, h_{i,N})$ be the vector of hold back levels at warehouse i. The replenishment lead time is exponentially distributed with mean $1/\mu_i > 0$ at warehouse i. This assumption does not seem to be very restrictive, as typically the performance characteristics of the system are relatively insensitive to the lead time distribution (see Alfredsson and Verrijdt 1999, Reijnen et al. 2009). We assume ample replenishment capacity, i.e. the

replenishment rate is equal to $(S_i - x_i)\mu_i$. The latter assumption is not restrictive as only minor changes have to be made to deal with other settings. For example, a repair rate of μ_i, independently of the number of outstanding orders, could be used as well.

The demand at warehouse i is given by a Poisson process with rate $\lambda_i > 0$, $i = 1, \ldots, N$. We refer to this as class i demand. If there is stock on hand at the warehouse where the demand arises, the demand is directly fulfilled. Otherwise, the warehouse requests for an LT at other warehouses. For this, each warehouse has a pre-specified order by which it will contact other warehouses. Denote this sequence for warehouse i by σ_i, whose entries are in $\{1, \ldots, N\}\backslash\{i\}$. If e.g. $\sigma_1 = \{2, 3\}$, then warehouse 1 will first ask 2 for an LT, and then 3, and otherwise the demand is satisfied by an external source and thus lost for the local warehouses (see Figure 1). By $j \in \sigma_i$ we denote that j is an element of the sequence σ_i, and $\sigma_i(k)$ denotes the kth element of the sequence, $1 \leq k \leq |\sigma_i|$, where $|\sigma_i|$ denotes the number of elements in the sequence. As $i \notin \sigma_i$ we have $0 \leq |\sigma_i| \leq N - 1$. Location $j \neq i$ accepts warehouse i's LT request only if $x_j > h_{j,i}$. In this case a part is taken from warehouse j's stock and used to fulfill warehouse i's demand. If $x_j \leq h_{j,i}$ for all $j \in \sigma_i$ (or if $\sigma_i = \emptyset$), then the demand is lost for the local warehouses. We do not allow for back orders, neither for rebalancing of stock not triggered by demands. We assume all demand and replenishment processes to be mutually independent.

So, for a demand at warehouse i, there are three possibilities: fulfill it directly from stock, via a lateral transshipment or it fulfilled via an emergency shipment. We are interested in the fractions of the demands that are fulfilled in either way. For this, we define the following fractions, for warehouse i:

- β_i: fraction of the class i demand that is fulfilled directly from stock;
- $\alpha_{i,j}$: fraction of the class i demand that is fulfilled via lateral transshipment from j ($j \neq i$), and by definition $\alpha_{i,j} = 0$ if $j \notin \sigma_i$;
- θ_i: fraction of the class i demand fulfilled via emergency shipment.

Furthermore, we define α_i to be the total fraction of the class i demand at warehouse i that is fulfilled via lateral transshipment:

$$\alpha_i = \sum_{j \in \sigma_i} \alpha_{i,j}, \tag{1}$$

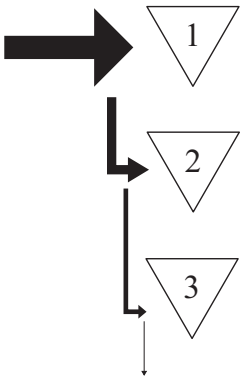


Figure 1: Demand overflow from warehouse 1 when $\sigma_1 = (2, 3)$.

where an empty sum equals zero. By definition, for all i it holds that:

$$\beta_i + \alpha_i + \theta_i = 1. \tag{2}$$

The model can be evaluated exactly as a Markov process for given hold back and base stock levels. The state of the system is given by the vector of stock levels $\underline{x} = \{x_1, \ldots, x_N\} \in \mathcal{S}$ where the state space $\mathcal{S}$ is given by $\mathcal{S} = \{0, 1, \ldots, S_1\} \times \ldots \times \{0, 1, \ldots, S_N\}$. For $N = 2$ the transition rates are shown in Figure 2. Denote by Q the transition rate matrix and by π the stationary probability distribution of $\underline{x}$. It is well known that π can be found by solving the following system:

$$\begin{cases} \pi.Q = \underline{0}, \\ \sum_{\underline{x} \in \mathcal{S}} \pi(\underline{x}) = 1. \end{cases} \tag{3}$$

From π, the values of β_i, $\alpha_{i,j}$ and θ_i follow.

By the dimension of $\mathcal{S}$, which is $|\mathcal{S}| = \prod_{i=1}^{N}(S_i + 1)$, and hence the dimension of Q, evaluation of the stationary probability distribution by solving (3) is not feasible for larger values of N and S_i because of the curse of dimensionality. Hence, there is a need for fast and accurate approximations for the β_i's, $\alpha_{i,j}$'s and θ_i's. In the next section two of such algorithms are presented.

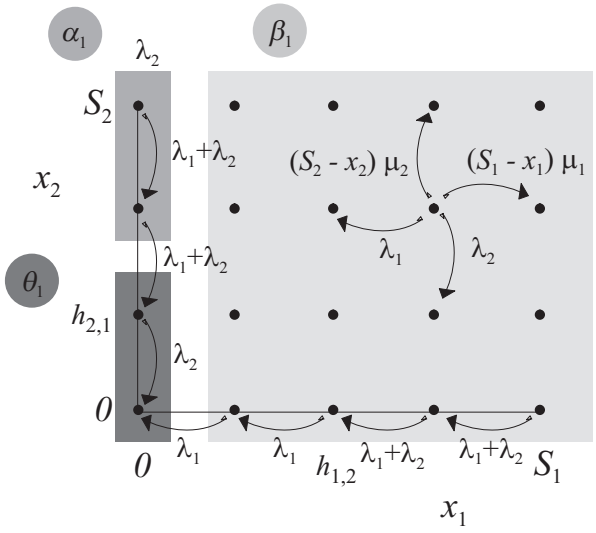


Figure 2: Diagram of the transition rates for $N = 2$, when $\sigma_1 = (2)$, $\sigma_2 = (1)$ and the hold back levels are given by $h_{1,2} = 2$ and $h_{2,1} = 1$. Note that when $x_1 = 0$ and $x_2 > h_{2,1} = 1$, demands at warehouse 1 are satisfied by lateral transshipment from warehouse 2, and vice versa when $x_2 = 0$ and $x_1 > h_{1,2} = 2$. Indicated are the states that contribute to β_1 (light gray), α_1 (=$\alpha_{1,2}$, gray) and θ_1 (dark gray).

3 Approximation Algorithms

In this section we present two algorithms for approximating β_i, $\alpha_{i,j}$ and θ_i for all i, j when the hold back and base stock levels are given. We first explain the general idea behind both algorithms. Then we present the *Poisson overflow algorithm*, followed by the *On/Off overflow algorithm*. The performance of both is tested and compared in a numerical study in the next section.

3.1 Idea behind approximations

Our approximations are based on a similar idea as that in Axsäter (1990). We make a *decomposition* of the network of warehouses into individual local warehouses. In this way, we only deal with solving a Markov process per local warehouse, with states x_i, instead of solving the Markov process with states $\underline{x}$. The lateral transshipments are modeled as overflow demand streams. These constitute an additional demand stream at the other warehouses. The stream from warehouse i at $j \in \sigma_i$ is referred to as the overflow stream (i, j). This is graphically depicted in Figure 1. Consequently, each warehouse can be evaluated individually

as an Erlang loss system ($M/M/S/S$ queue) with state dependent arrival rates. In earlier approximations (Axsäter 1990, Alfredsson and Verrijdt 1999, Kukreja et al. 2001, Kutanoglu 2008, Kranenburg and Van Houtum 2009, Reijnen et al. 2009), these overflow demand streams have been assumed to be Poisson processes. In our first algorithm, this approximation is used as well, but then with the addition of hold back levels. In our second algorithm, however, we approximate the overflow streams by the use of On/Off overflow processes, more precisely, interrupted Poisson processes (cf. Kuczura 1973).

We expect the approximation using the On/Off processes to be more precise, as it better follows the actual overflow demand streams. Overflow demands occur when the warehouse is stocked out. So, during a stock-out the overflow demand process is turned On. The demands it is then facing, flow over to other warehouses as lateral transshipment requests. On the other hand, when the warehouse has a positive stock level, there are no overflow demands. So, the overflow process is turned Off. Hence, it is one step more accurate than a Poisson process. We call this algorithm the On/Off overflow algorithm. The approximation here is that we *assume* the On and Off durations to be independently, exponentially distributed. We take the means of both to be equal to the actual means. It will turn out that this approximation performs very well.

The algorithms both consist of two main steps, which are alternately executed:

Step 1: Evaluation of the steady-state distribution (and hence performance characteristics) of the individual warehouses, given the overflow demand streams;

Step 2: Updating of the overflow demand streams, given the steady-state distribution of the individual warehouses.

The two steps are executed until the changes in consecutive iterations are smaller than some pre-specified, small value ε.

3.2 Poisson overflow algorithm

We approximate the overflow demand streams by Poisson processes. Let $\lambda_{i,j}$ be the rate of the overflow demand stream (i, j), $j \in \sigma_i$. Define $\lambda_{i,i} = \lambda_i$ and $\lambda_{i,j} = 0$, $j \notin \sigma_i,\ j \neq i$. The overflow demand rates $\lambda_{i,j}$ for all $j \in \sigma_i$ are calculated from the probabilities that a demand

is satisfied either directly from stock or via a lateral transshipment. Recall that a lateral transshipment is carried out from warehouse j to warehouse i only if $x_j > h_{j,i}$. We denote the probability that the latter is true by $p_{i,j}$ for all i and $j \in \sigma_i \cup \{i\}$, defining $p_{i,i} = \mathbb{P}[x_i > 0]$. So, $p_{i,i}$ is the fill rate of warehouse i: the probability that a demand can be directly fulfilled from stock. The $p_{i,j}$ are derived when evaluating an individual warehouse.

We initially assume for all overflow demand rates $\lambda_{i,j} = 0$ and update these in each iteration of the algorithm. Based on these, the probabilities $p_{i,j}$ are updated in each iteration as well. Below we first explain these two parts of the iteration in more detail, as well as the finalization, and then state the algorithm.

3.2.1 Evaluation of individual warehouses

Suppose that the overflow rates $\lambda_{i,j}$ for all i and $j \in \sigma_i$ are given. Then we determine the probabilities that a demand can be fulfilled, either directly from stock, or via lateral transshipment.

Each of the warehouses is evaluated individually. For warehouse i, we consider the Markov process the state of which is given by its stock level x_i on the state space $\{0, 1, \ldots, S_i\}$. We have the following transitions. The replenishment rate is given by $(S_i - x_i)\,\mu_i$. Class i demands arise with rate λ_i, and are satisfied when $x_i > 0$. Moreover, the warehouse faces the demand overflow streams from the other warehouses, namely with rate $\lambda_{j,i}$ from warehouse j. However, only when $x_i > h_{i,j}$ such a demand is satisfied. Therefore, the demand rate depends on the state of the system.

Denote by $\gamma_i(x_i)$ the arrival rate when the stock level equals x_i. Then $\gamma_i(x_i)$ is the sum of the (overflow) demand arrival rates that are satisfied when in state x_i:

$$\gamma_i(x_i) = \sum_{j=1}^{N} \lambda_{j,i} \cdot 1\{x_i > h_{i,j}\},$$

for $x_i \in \{0, 1, \ldots, S_i\}$, taking $h_{i,i} = 0$. Note that $\gamma_i(0) = 0$ and $\gamma_i(x_i)$ always includes $\lambda_i = \lambda_{i,i}$ for $x_i > 0$. Figure 3 shows an example of the transitions rates, taking as state the number of outstanding orders, denoted by $y = S_i - x_i$.

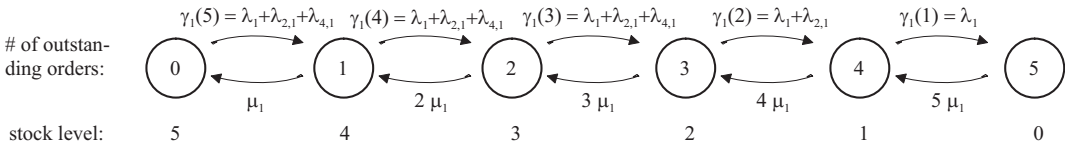


Figure 3: Example of the transitions rates at warehouse 1*, where* $N = 4$*,* $S_1 = 5$ *and* $h_1 = (0, 1, 5, 2)$*.*

Consequently, we can analyze warehouse i separately using the Erlang loss model. The steady-state behavior of the number of outstanding orders is identical to the steady-state behavior of the number of busy servers y in an Erlang loss system ($M/M/S_i/S_i$ queue) with S_i servers, a state-dependent arrival (i.e. demand) rate $\gamma_i(S_i - y)$, and the mean replenishment lead time $1/\mu_i$ as mean service time. Define $\tilde{L}_i$ to be the stationary probability distribution of $y \in \{0, 1, \ldots, S_i\}$ at warehouse i. When denoting by $\tilde{L}_i(y)$ its yth element, it is given by:

$$\tilde{L}_i(y) = \frac{\dfrac{\prod_{j=0}^{y-1} \gamma_i(S_i - j)}{\mu_i^y \, y!}}{\sum_{n=0}^{S_i} \dfrac{\prod_{m=0}^{n-1} \gamma_i(S_i - m)}{\mu_i^n \, n!}}, \quad y = 0, \ldots, S_i.$$

Recall that $\gamma_i(\cdot)$ is fully determined by the vectors $\underline{h}_i$ and $(\lambda_{1,i}, \ldots, \lambda_{N,i})$.

From the stationary probability distribution $\tilde{L}_i(\cdot)$ the probabilities $p_{i,j}$ can be computed:

$$\begin{aligned} p_{i,j} &= \mathbb{P}[x_j > h_{j,i}] \\ &= \sum_{y=0}^{S_j - h_{j,i} - 1} \tilde{L}_j(y), \quad \text{for all } i \text{ and } j \in \sigma_i, \end{aligned} \tag{4}$$

and 0 otherwise. Note that $\tilde{L}_i(\cdot)$ depends on the overflow demand rates $\lambda_{i,j}$.

3.2.2 Updating overflow rates

Suppose that the probabilities $p_{i,j}$'s are given, for all i, j, then we derive the overflow demand rates $\lambda_{i,j}$. As a fraction $\beta_i = p_{i,i}$ of the demand is satisfied directly from stock, the overflow

demand stream $\lambda_{i,\sigma_i(1)}$ (assuming $\sigma_i \neq \emptyset$) is $\lambda_{i,\sigma_i(1)} = (1-\beta_i)\,\lambda_i$. Of this stream, a fraction $p_{i,\sigma_i(1)}$ is satisfied by warehouse $\sigma_i(1)$, hence the remaining overflow to $\sigma_i(2)$ is given by: $\lambda_{i,\sigma_i(2)} = (1-p_{i,\sigma_i(1)})\lambda_{i,\sigma_i(1)}$. In general, defining $\lambda_{i,i} = \lambda_i$ and $p_{i,\sigma_i(0)} = \beta_i$, the overflow demand stream $\lambda_{i,\sigma_i(k)}$ is recursively given by $\lambda_{i,\sigma_i(k)} = (1-p_{i,\sigma_i(k-1)})\,\lambda_{i,\sigma_i(k-1)}$. Hence, for $k = 1,\ldots,|\sigma_i|$:

$$\lambda_{i,\sigma_i(k)} = \lambda_i \prod_{l=0}^{k-1} (1-p_{i,\sigma_i(l)}). \tag{5}$$

Note that this make use of the assumed independence of the stock levels at the local warehouses.

3.2.3 Finalization

When the iteration step of the algorithm terminates, the values of β_i, $\alpha_{i,j}$ and θ_i can be computed in the following way. The fraction of demand that is directly satisfied from stock (the fill rate) is given by:

$$\beta_i = p_{i,i}. \tag{6}$$

For the fraction of the demands that is satisfied by lateral transshipment from warehouse j, we have

$$\alpha_{i,j} = p_{i,j}\,\frac{\lambda_{i,j}}{\lambda_i}, \text{ for all } i \text{ and } j \in \sigma_i. \tag{7}$$

Here $\lambda_{i,j}/\lambda_i$ is the fraction of the demand of i that is offered to warehouse j, and of this, a fraction $p_{i,j}$ is satisfied. Next, the total fraction that is satisfied via lateral transshipment, α_i, is given by (1). Finally, the fraction of the demand that is not satisfied by the local warehouses, θ_i, follows from (2).

3.2.4 Poisson overflow algorithm

Input: $\lambda_i, \mu_i, S_i, \underline{h}_i, \sigma_i$, for $i = 1,\ldots,N$, ε;
Output: β_i, $\alpha_{i,j}$ and so α_i, θ_i, for $i,j = 1,\ldots,N,\ i \neq j$.

Step 0: Initialize for all $i,\ j \in \sigma_i$: $\lambda_{i,j} = 0$ and $\lambda_{i,i} = \lambda_i$.
Step 1: Calculate for all $i,\ j \in \sigma_i \cup \{i\}$: $p_{i,j}$ using (4).

Step 2: Calculate for all i, $j \in \sigma_i$: $\lambda_{i,j}$ using (5).

Step 3: Repeat Steps 1 and 2 until the $p_{i,j}$'s do not change more than ε, for all i, j. Then return, for all $i, j \in \sigma_i$: β_i using (6), $\alpha_{i,j}$ using (7), α_i using (1), and θ_i using (2).

3.3 On/Off overflow algorithm

In the previous approximation we have assumed the overflow demand process to be Poisson processes. We now develop a more precise approximation for these processes. Namely, instead of a Poisson process, we use an *interrupted Poisson process* (cf. Kuczura 1973): a Poisson process which is alternately turned On, for an exponentially distributed time, and then turned Off, for another (independent) exponentially distributed time. Hence, the process is described by three parameters. For the overflow demand stream (i, j) let $1/\phi_{j,i}$ be the mean Off duration, $1/\eta_{j,i}$ the mean On duration, and λ_i the rate while On. So, we have to estimate two out of the three parameters of the interrupted Poisson process.

The idea is as follows. The overflow demand processes of warehouse i at warehouse $\sigma_i(1)$ can be in two states, based on warehouse i's stock level. If warehouse i has stock on hand, there is *no* overflow. So, the overflow process is turned Off. However, if warehouse i faces a stock-out, all demands flow over to warehouse $\sigma_i(1)$: the overflow process is turned On, and is given by a Poisson process with rate λ_i.

The same reasoning is applicable for the overflow demand of i at warehouse $\sigma_i(k)$ for $k = 2, \ldots, |\sigma_i|$. The overflow process is On at $\sigma_i(2)$ exactly when i is stocked out and warehouse $\sigma_i(1)$'s inventory level is below its hold back level for i, i.e., when $x_{\sigma_i(1)} \leq h_{\sigma_i(1),i}$. The overflow is turned Off otherwise.

When precisely overflows are turned On or Off, basically depends on the entire state space. The approximation we apply here, is to approximate the On and Off durations by exponential distributions. We choose the means of these to be equal to the estimated mean durations. These means are updated in each iteration of the algorithm.

The mean On duration at $\sigma_i(1)$ is the duration that the stock level x_i equals zero. It is exactly

exponentially distributed, with mean

$$1/\eta_{\sigma_i(1),i} = 1/(S_i\,\mu_i). \tag{8}$$

In general, for the other On and Off durations, the exponential distribution is an approximation.

Like in the Poisson overflow algorithm, we initialize by putting all overflow demand streams to zero. That is, all On/Off processes start being turned Off:

$$\begin{aligned} \phi_{j,i} &= 0 \text{ for } j \in \sigma_i, \\ \eta_{j,i} &= \infty \text{ for } j \in \sigma_i, j \neq 1. \end{aligned} \tag{9}$$

Below we first explain the two parts of the iteration and the finalization. Then we state the algorithm.

3.3.1 Evaluation of individual warehouses

Given the mean On and Off durations, we evaluate each of the individual warehouses. Consider warehouse i. Next to its own demand stream, it faces a number of overflow demands streams, namely from the warehouses j for which $i \in \sigma_j$. Denote this vector by $d_i = (j \mid i \in \sigma_j, j = 1, \ldots, N)$, and by $d_i(k)$ its kth element. As each of these overflow demand streams can either be turned On or Off, we have $2^{|d_i|}$ possible combinations, where $|d_i| \leq N-1$. Hence the joint process of the stock level and this state of the overflow demand streams, is a Markov process on a state space of dimension $2^{|d_i|}$ by $S_i + 1$. We encode the states by $(y, \underline{\delta})$ for $y = S_i - x_i$ the number of outstanding orders, $y \in \{0, 1, \ldots, S_i\}$, and $\underline{\delta} \in \{0,1\}^{|d_i|}$, where the$k$th component of $\underline{\delta}$ equals 0 if the overflow stream from $d_i(k)$ is Off, and 1 if On. This state space is denoted by $\mathcal{S}_i$. Figure 4 shows an example.

The transition rates of this Markov process are as follows. When the process is in state $(y, \underline{\delta})$ three types of transitions can occur: a replenishment, an (overflow) demand, or a change in whether one of the processes is On or Off. With rate $y\,\mu_i$ replenishments take place, moving the process to state $(y-1, \underline{\delta})$. A demand, occurring with rate λ_i, moves the process to $(y+1, \underline{\delta})$

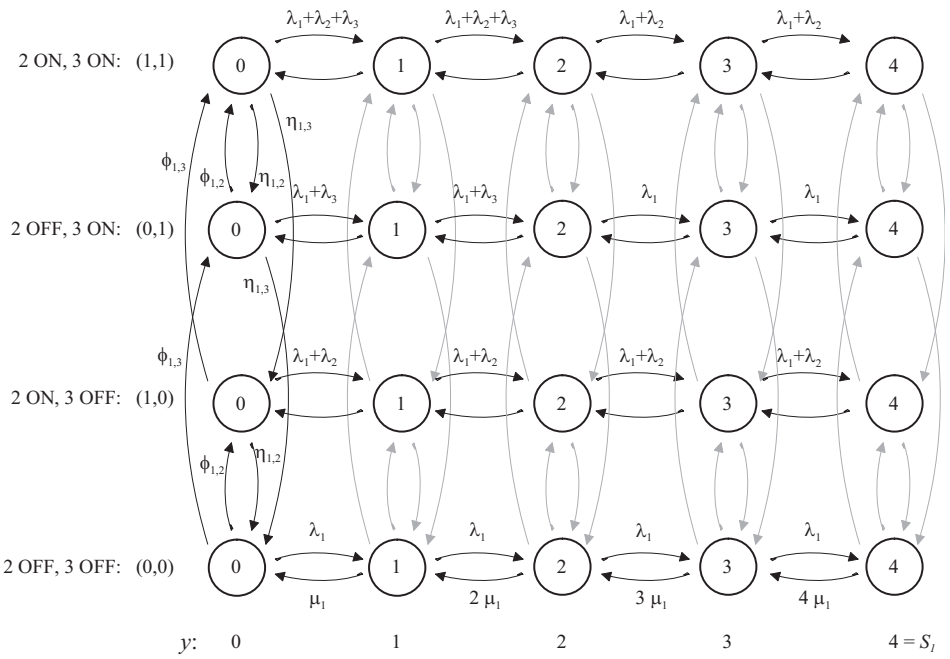


Figure 4: Example of the Markov processes of warehouse 1*, where* $N = 3$*,* $S_1 = 4$*,* $h_1 = (0, 0, 2)$ *and* $d_1 = \{2, 3\}$*.*

if $y < S_i$. An overflow demand, occurring at rate λ_k for all $k \in d_i$ when the overflow demand stream (k, i) is On, is only accepted if $S_i - y > h_{i,k}$. This also moves the process to state $(y + 1, \underline{\delta})$. Finally, each of the overflow demand processes can switch from On to Off or vice versa. Transitions are only possible between states for which $\underline{\delta}$ differs in exactly one entry. With rate $\phi_{i,d_i(j)}$ the overflow from warehouse $d_i(j)$ is switched On, $j = 1, \ldots, |d_i|$, hence with this rate the process moves to state $(y, \underline{\delta} + \underline{e}_{d_i(j)})$, where $\underline{e}_k$ denotes the unit vector of appropriate length with an 1 at position k. Analogously, with rate $\eta_{i,d_i(j)}$ the process moves to state $(y, \underline{\delta} - \underline{e}_{d_i(j)})$. See again Figure 4 for an example.

Denote by Q_i the matrix of transitions rates and denote by $\pi_i(y, \underline{\delta})$ the stationary probability distribution of this process. Then π_i can be found by solving the system:

$$\begin{cases} \pi_i . Q_i = \underline{0}, \\ \sum_{(y,\underline{\delta}) \in \mathcal{S}_i} \pi_i(y, \underline{\delta}) = 1. \end{cases} \tag{10}$$

The dimension of Q_i is $(S_i + 1)2^{|d_i|}$ by $(S_i + 1)2^{|d_i|}$, where $|d_i| \leq N - 1$. We have to solve the system for all $i = 1, \ldots, N$ in each iteration of the algorithm. Note that this is of a much

smaller order than the original problem (3), where the dimension of Q is $\prod_{i=1}^{N}(S_i+1)$ by $\prod_{i=1}^{N}(S_i+1)$.

3.3.2 Updating On and Off durations

Given the stationary probability distribution of each of the individual warehouses, we update the mean On and Off durations. Consider warehouse i and first concentrate on $\sigma_i(1)$. Its mean On duration is fixed: $1/\eta_{\sigma_i(1),i} = 1/(S_i\,\mu_i)$. We show that, by using the stationary probability distribution π_i, we can directly find the mean Off duration $1/\phi_i$.

The fraction of class i's demands that is satisfied from stock is β_i. Hence, by PASTA, β_i is also the fraction of time that class i's overflow is turned Off. This gives that the fraction of time the overflow is turned On is $1-\beta_i$. The state space $\mathcal{S}_i$ can be split into two mutually exclusive subsets, say $\mathcal{S}_{i,\text{off}}$ (all states $(y,\underline{\delta}) \in \mathcal{S}_i$ for which $y < S_i$) and $\mathcal{S}_{i,\text{on}}$ (all states $(y,\underline{\delta}) \in \mathcal{S}_i$ for which $y = S_i$). Denote by $\mathbb{E}[\mathcal{S}_{i,\text{off}}]$ the expected duration the process is in subset $\mathcal{S}_{i,\text{off}}$, once first entered it, before leaving it again. Define $\mathbb{E}[\mathcal{S}_{i,\text{on}}]$ analogously. It is clear that the following should hold:

$$\frac{\beta_i}{1-\beta_i} = \frac{\mathbb{E}[\mathcal{S}_{i,\text{off}}]}{\mathbb{E}[\mathcal{S}_{i,\text{on}}]}.$$

Using that $\phi_{\sigma_i(1),i} = 1/\mathbb{E}[\mathcal{S}_{i,\text{off}}]$ and $\eta_{\sigma_i(1),i} = 1/\mathbb{E}[\mathcal{S}_{i,\text{on}}] = S_i\,\mu_i$, it follows that:

$$\phi_{\sigma_i(1),i} = S_i\,\mu_i\,\frac{1-\beta_i}{\beta_i}. \tag{11}$$

The β_i follows from the stationary probability distribution π_i:

$$\beta_i = 1 - \sum_{(S_i,\underline{\delta})\in\mathcal{S}_i} \pi_i(S_i,\underline{\delta}).$$

For the On and Off durations of i at $\sigma_i(k)$ for $k = 2,\ldots,|\sigma_i|$ some more work has to be done. Firstly, note that when the overflow stream $(i,\sigma_i(1))$ is turned On, there are periods that these demands are satisfied by warehouse $\sigma_i(1)$, and periods that this is not the case because $x_{\sigma_i(1)} \leq h_{\sigma_i(1),i}$. During the latter periods, the overflow $(i,\sigma_i(2))$ is turned On. The duration of this On period, and that of the Off period as well, of i at $\sigma_i(2)$ depends on the (Markov)

process at warehouse $\sigma_i(1)$. Hence, in general, the duration of the On and Off periods of i at warehouse $\sigma_i(k)$ follows from evaluation of the Markov process at warehouse $\sigma_i(k-1)$, $k = 1, \ldots, |\sigma_i|$.

To find the expected On and Off durations of an overflow stream, we have to determine the expected time the Markov process is in a certain subset of states from the moment on the process enters it, before leaving it again. For this, we split the state space $\mathcal{S}_{\sigma_i(k-1)}$ into two subsets, one consisting of the states in which the overflow of i to warehouse $\sigma_i(k)$ in turned On, and the other for which it is turned Off. To calculate the mean time spent in a certain subset, we view all states *except* the subset of interest as absorbing states, i.e. states that the Markov process cannot leave anymore once entered. Then we need to derive the mean time until absorption in these states. We first describe, following Grinstead and Snell (1997), how the mean time to absorption can be computed in general. Next we explain how this can be used to calculate the mean On and Off durations.

Consider a general, irreducible Markov process with state space $\mathcal{S}' = \mathcal{S}'_1 \cup \mathcal{S}'_2$, where $\mathcal{S}'_1 \cap \mathcal{S}'_2 = \emptyset$, with transition rates q_{ij} for $i, j \in \mathcal{S}'$. Let the matrix P be the transition probability matrix, given by $p_{ii} = (\nu_{\max} - |q_{ii}|)/\nu_{\max}$, and $p_{ij} = q_{ij}/\nu_{\max}$ for $i \neq j$, where $\nu_{\max} = \max_i |q_{ii}|$. Let $P_{\mathcal{S}'_1}$ be an $|\mathcal{S}'_1|$ by $|\mathcal{S}'_1|$ matrix with only the rows and columns of P that correspond to states in $\mathcal{S}'_1$. Its row sums are ≤ 1. Given that we start in a state $s \in \mathcal{S}'_1$, let t_s denote the expected number of steps to get absorbed in $\mathcal{S}'_2$. It is the unique solution of

$$(I - P_{\mathcal{S}'_1})\,\vec{t} = (1, 1 \ldots, 1)^T,$$

where $\vec{t} = (t_1, \ldots, t_{|\mathcal{S}'_1|})^T$ and I is the identity matrix of appropriate size. Then the vector with the mean times until absorption of the Markov process is $\vec{t}/\nu_{\max}$.

When we are given an initial distribution over the starting states in $\mathcal{S}'_1$, say $\vec{p} = (p_1, \ldots, p_{|\mathcal{S}'_1|})$, the mean time until absorption is

$$\mathbb{E}[T(\mathcal{S}'_1)] = \vec{p} \cdot \vec{t}/\nu_{\max}, \tag{12}$$

where the dot denotes the inner product of two vectors. When we want to find the mean duration the system is in $\mathcal{S}'_1$ before going to $\mathcal{S}'_2$, this initial distribution is given by the steady

state probability distribution that the first step out of $\mathcal{S}_2'$ is to $s \in \mathcal{S}_1'$, say p_s. Denote by $\pi = (\pi_{\mathcal{S}_1'}, \pi_{\mathcal{S}_2'})$ the stationary probability distribution. Let

$$\vec{\tilde{p}} = \pi_{\mathcal{S}_2'} \cdot P_{\mathcal{S}_2',\mathcal{S}_1'},$$

where $P_{\mathcal{S}_2',\mathcal{S}_1'}$ denotes the (non-square) matrix, which is the part of the transition probability matrix P of which the rows correspond to states in $\mathcal{S}_2'$, and the columns to states in $\mathcal{S}_1'$. We normalize $\vec{\tilde{p}}$ to find $\vec{p} = \{p_1, \ldots, p_{|\mathcal{S}_1'|}\}$:

$$\vec{p} = \vec{\tilde{p}} / \sum_{s \in \mathcal{S}_1'} \tilde{p}_s.$$

Hence, using (12) we have $\mathbb{E}[T(\mathcal{S}_1')]$ and the rate at which the process jumps from subset $\mathcal{S}_1'$ to $\mathcal{S}_2'$ is $1/\mathbb{E}[T(\mathcal{S}_1')]$.

In order to calculate the mean On and Off durations of the stream $(i, \sigma_i(k))$, we split the state space $\mathcal{S}_{\sigma_i(k-1)}$ into two mutually independent subsets. Let $j = \sigma_i(k-1)$. The overflow of i to $\sigma_i(k)$ is turned On when i's overflow to j is On but is not satisfied there, because the stock level $x_j \leq h_{j,i}$. Denote by $\mathcal{S}_j(i)$ the subset of states of $\mathcal{S}_j$ for which this holds. Recall that $y = S_i - x_i$, then this subset is given by:

$$\mathcal{S}_j(i) = \{(y, \underline{\delta}) \in \mathcal{S}_j \mid \underline{\delta}^{(i)} = 1 \text{ and } y > h_{j,i}\},$$

where $\underline{\delta}^{(i)}$ denotes the component of $\underline{\delta}$ corresponding to the overflow demand stream (i, j). Furthermore, define $\overline{\mathcal{S}}_j(i) = \mathcal{S}_j \backslash \mathcal{S}_j(i)$, that is, the complement of $\mathcal{S}_j(i)$ with respect to $\mathcal{S}_j$.

In the example of Figure 4, $S_1(2)$ consists of only two states: $(4, (1, 0))$ and $(4, (1, 1))$. Only when warehouse 1 is out-of-stock ($y = 4$), the overflow demand stream $(2, 1)$ is not satisfied when On. Hence, in these states, the overflow stream $(2, \sigma_2(k))$ is On, when $\sigma_2(k-1) = 1$. Analogously, $S_1(3)$ consists of six states, given by $S_1(3) = \{(y_1, \underline{\delta}) \in \mathcal{S}_1 \mid \underline{\delta} \in \{(0, 1), (1, 1)\} \text{ and } y_1 > 2 = h_{3,1}\}$. In these states the overflow stream $(3, \sigma_3(k))$ is On, when $\sigma_3(k-1) = 1$.

The mean On and Off durations of $(i, \sigma_i(k))$ now follow from the mean times spent in subset $\mathcal{S}_{\sigma_i(k-1)}(i)$, respectively subset $\overline{\mathcal{S}}_{\sigma_i(k-1)}(i)$, from the moment on the process enters the subset,

before leaving it again. Hence, the rates $\eta_{j,i}$ and $\phi_{j,i}$ follow and are given by:

$$\begin{aligned} \eta_{\sigma_i(k),i} &= 1/\mathbb{E}[T(\mathcal{S}_{\sigma_i(k-1)}(i))], \\ \phi_{\sigma_i(k),i} &= 1/\mathbb{E}[T(\overline{\mathcal{S}}_{\sigma_i(k-1)}(i))], \end{aligned} \tag{13}$$

for all i and $k = 2, \ldots, |\sigma_i|$. Here we define $1/0 := \infty$ and $1/\infty := 0$. If, for example, $\mathbb{E}[T(\mathcal{S}_{\sigma_i(k-1)}(i))] = 0$, then the overflow demand stream basically skips warehouse $\sigma_i(k-1)$ and is entirely routed to warehouse $\sigma_i(k)$. This overflow process then has the same mean On and Off durations. Furthermore, recall that when On, the demand rate of the overflow stream $(i, \sigma_i(k))$ equals λ_i, that is, we do not have to estimate the overflow demand rate in this algorithm.

3.3.3 Finalization

When the algorithm terminates, it remains to calculate the β_i, $\alpha_{i,j}$, α_i and θ_i from the π_i's. This comes down to taking the summation of π_i over certain subsets of states:

$$\begin{aligned} \beta_i &= 1 - \sum_{(S_i,\underline{\delta})\in\mathcal{S}_i} \pi_i(S_i, \underline{\delta}), \\ \alpha_{i,j} &= \sum_{\substack{(y,\underline{\delta})\in\mathcal{S}_j:\ \underline{\delta}^{(i)}=1, \\ y\le S_j - h_{j,i} - 1}} \pi_j(y, \underline{\delta}), \text{ for } j \in \sigma_i, \text{ and 0 otherwise}, \\ \alpha_i &= \sum_{j\in\sigma_i} \alpha_{i,j}, \\ \theta_i &= 1 - \beta_i - \alpha_i. \end{aligned} \tag{14}$$

3.3.4 On/off overflow algorithm

Input: $\lambda_i, \mu_i, S_i, \underline{h}_i, \sigma_i$, for $i = 1, \ldots, N$, ε;

Output: β_i, $\alpha_{i,j}$ and so α_i, and θ_i, for $i, j = 1, \ldots, N,\ i \neq j$.

Step 0: Initialize for all i, j: $\phi_{j,i}$ and $\eta_{j,i}$ using (8) and (9).

Step 1: Solve for all warehouses i the stationary probability distribution π_i using (10).

Step 2: Calculate for all $i, j \in \sigma_i$: $\eta_{j,i}$ and $\phi_{j,i}$ using (11) and (13).

TESTBED	
$\mu_1 = \mu_2 = \ldots = \mu_5$	1,
$f_1, f_2, f_3, f_4, f_5 \in$	{50%,70%,90%,95%} $\rightarrow$ determines $S_1, \ldots, S_5$ using (15),
$\bar{h}_i \in$	$\{0, 1, \ldots, S_i - 1\}$ for all i
$\lambda_1 \in$	$\{0.2, 0.5, 1, 2\}$,
$\lambda_2, \lambda_3 \in$	$\{0.2, 0.5, 1\}$,
$\lambda_4, \lambda_5 \in$	$\{1\}$,
$N = 2:$ $(\sigma_1, \sigma_2) \in$	$\{\{\{2\},\{1\}\},$
	$\{\{\},\{1\}\}\},$
$N = 3:$ $(\sigma_1, \sigma_2, \sigma_3) \in$	$\{\{\{2,3\},\{3,1\},\{1,2\}\},$
	$\{\{2\},\{3\},\{1\}\},$
	$\{\{\},\{1\},\{1\}\}\},$
$N = 5:$ $(\sigma_1, \ldots, \sigma_5) \in$	$\{\{\{2,3,4,5\},\{3,4,5,2\},\{4,5,1,2\},\{5,1,2,3\},\{1,2,3,4\}\},$
	$\{\{2\},\{3\},\{4\},\{5\},\{1\}\},$
	$\{\{\},\{1\},\{1\},\{1\},\{1\}\}\}.$

Table 1: Test bed for numerical study: factorial design of the given possibilities.

Step 3: Repeat Steps 1 and 2 until the elements of π_i do not change more than ε, for all i. Then return, for $i, j \in \sigma_i$: β_i, $\alpha_{i,j}$, α_i, and θ_i using (14).

4 Numerical study

In order to determine the performance of the two presented approximation algorithms, we execute a numerical study. We first focus on the case where all hold back levels are set to 0. For that the Poisson overflow algorithm boils down to the algorithm given in Reijnen et al. (2009). Hence we can test the performance gained by the use of the On/Off approximation. Then we allow for hold back levels. In both cases, we compare both the performance of the algorithms with respect to the exact outcomes (via Markov analysis as described in Section 2), as well as the mutual performance of the algorithms.

Testbed

We consider $N = 2$, 3, and 5 local warehouses. We perform a factorial design of the test bed given in Table 1. In the table, the values of λ_i, μ_i, and σ_i are given, as well as f_i. This

is the minimum fill rate of warehouse i in isolation, i.e. in a situation without any lateral transshipments. From f_i the base stock level S_i follows:

$$S_i = \min \{S \in \mathbb{N} \cup \{0\} \mid L(S, \lambda_i/\mu_i) \leq 1 - f_i\}, \tag{15}$$

where L is the Erlang loss function: $L(S, \rho) = \frac{\rho^S/S!}{\sum_{n=1}^{S} \rho^n/n!}$.

For $N = 2$, 3, and 5, we test 384, 1500, respectively 2500 instances *without* hold back levels, and 1000, 3000, respectively 5000 instances *with* hold back levels. That is, for $N = 2$ without hold back levels, we perform a full factorial design of the settings given in Table 1, and for the other cases we randomly select the indicated number of instances from the full factorial designs. We restrict our attention to a single hold back level per warehouse, i.e. $\bar{h}_i := h_{i,1} = \ldots = h_{i,N}$. For the instances with hold back levels, we take $\bar{h}_i \in \{0, 1, \ldots, S_i - 1\}$, excluding instances where $\{\bar{h}_1, \ldots, \bar{h}_N\} = \{0, \ldots, 0\}$. So, we can both have instances where all hold back levels are positive, as well as instances with combinations of zero and positive hold back levels.

For all instances we run:

- Poisson overflow algorithm, using $\varepsilon = 10^{-10}$;
- On/off overflow algorithm, using $\varepsilon = 10^{-10}$;
- Exact Markov analysis (see Section 2).

We concentrate on the average and maximum absolute errors in the β_i, α_i, and θ_i ($i = 1, \ldots, N$). Let

$$\Delta\beta_i = |\beta_{i,approx} - \beta_{i,exact}| * 100, \quad \Delta\alpha_i = |\alpha_{i,approx} - \alpha_{i,exact}| * 100, \quad \Delta\theta_i = |\theta_{i,approx} - \theta_{i,exact}| * 100.$$

That is, we consider the error $\Delta\beta_i$ as the differences in the *percentages* $\beta_{i,approx}$ and $\beta_{i,exact}$. By 'av$\Delta\beta$' and 'max$\Delta\beta$' we denote the average respectively maximum over all $\Delta\beta_i$ (α, θ analogously).

The summary of all results is given in Table 2, which gives the average absolute errors and the maximal absolute errors for β_i, α_i, and θ_i for both the Poisson overflow algorithm as well as the On/Off overflow algorithm. From these results, it turns out that the On/Off algorithm

clearly outperforms the Poisson algorithm. The average error for On/Off in β is about four times smaller. Also, the maximal errors are much smaller. Already from $N = 3$ on the errors in the On/Off algorithm are almost nil. In the following, we further investigate the results.

Graphs

In Figures 5 and 6 the errors are graphically represented. Again it becomes clear that the On/Off overflow algorithm outperforms the Poisson overflow algorithm.

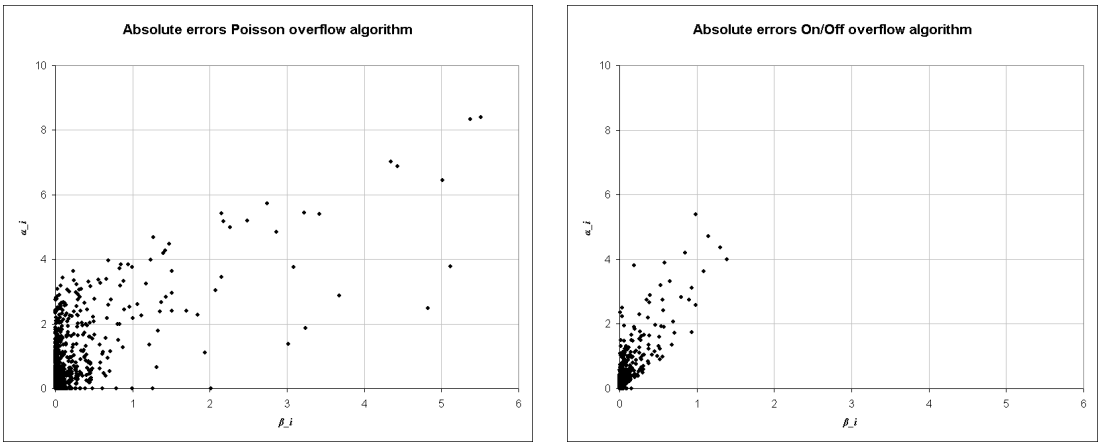
Hold back levels

When we split out the results according to whether we include hold back levels or not, the results are given in Table 3. Both algorithms perform better for the case *with* hold back levels, about a factor two to three. We expect that this is because of the smaller overflow streams in the case of hold back levels. When hold back levels are set, a higher fraction of the demand is already satisfied at the warehouse itself, because it holds stock back from LT. Hence the overflow to the next warehouse is smaller (Poisson algorithm), respectively longer turned Off (On/Off algorithm).

Complete pooling

For the special case of complete pooling, we compare the results to those of the approximation algorithm as given in Kranenburg and Van Houtum (2009). Under a complete pooling strategy, all warehouses basically act as being one large warehouse. Hence, the fraction θ_i is the same for all warehouses $i = 1, \ldots, N$. Moreover, it can be computed exactly using the Erlang loss formula and this is exploited in their approximation. For the rest, Kranenburg and Van Houtum (2009) use an iterative approximation algorithm comparable to our Poisson overflow algorithm. As θ_i is determined exactly, the absolute errors in β_i and α_i are always equal.

The results for complete pooling are given in Table 4, from which it becomes clear that both our On/Off overflow algorithm and Kranenburg and Van Houtum's algorithm perform



(a) Poisson overflow algorithm *(b)* On/Off overflow algorithm

Figure 5: Scatterplot of the absolute errors in α_i versus β_i, expressed as percentages, for $N = 2$ ($i = 1, 2$, in each plot 2,768 data points).

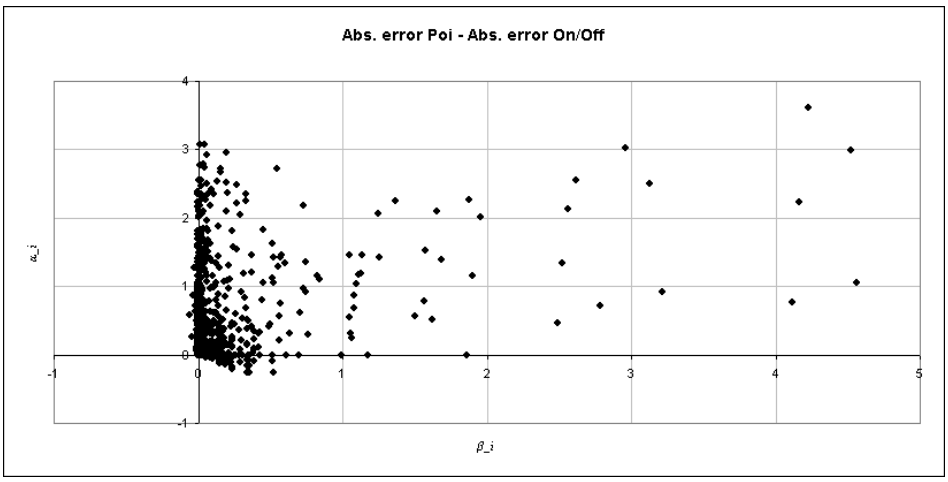


Figure 6: Scatterplot of the absolute errors, expressed as percentages, in the Poisson overflow algorithm minus those in the On/Off overflow algorithm, plotted for α_i versus β_i, for $N = 2$ ($i = 1, 2$, 2,768 data points). Explanation: each data point on the right of (above) 0 indicates a case where the On/Off overflow algorithm performs better for β_i (α_i); on the left (below) a case where the Poisson overflow algorithm performs better for β_i (α_i).

	Poisson overflow alg.						*On Off overflow alg.*					
N *(# inst.)*	*av* $\Delta\beta$	*max* $\Delta\beta$	*av* $\Delta\alpha$	*max* $\Delta\alpha$	*av* $\Delta\theta$	*max* $\Delta\theta$	*av* $\Delta\beta$	*max* $\Delta\beta$	*av* $\Delta\alpha$	*max* $\Delta\alpha$	*av* $\Delta\theta$	*max* $\Delta\theta$
all N (13384)	0.016	5.510	0.070	8.390	0.061	3.406	0.004	1.384	0.025	5.396	0.021	4.406
$N = 2$ (1384)	0.144	5.510	0.635	8.390	0.559	3.406	0.038	1.384	0.223	5.396	0.190	4.406
$N = 3$ (4500)	0.001	0.062	0.005	0.098	0.005	0.054	0.000	0.040	0.002	0.088	0.002	0.054
$N = 5$ (7500)	0.002	0.062	0.004	0.102	0.004	0.039	0.001	0.036	0.002	0.073	0.002	0.039

Table 2: Summary of numerical results: average and maximal absolute errors in β_i, α_i, and θ_i, expressed as percentages, for both the Poisson overflow algorithm and the On/Off overflow algorithm.

	Poisson overflow alg.						*On Off overflow alg.*					
N *(# inst.)*	*av* $\Delta\beta$	*max* $\Delta\beta$	*av* $\Delta\alpha$	*max* $\Delta\alpha$	*av* $\Delta\theta$	*max* $\Delta\theta$	*av* $\Delta\beta$	*max* $\Delta\beta$	*av* $\Delta\alpha$	*max* $\Delta\alpha$	*av* $\Delta\theta$	*max* $\Delta\theta$
all N (4384)	0.025	5.510	0.089	8.390	0.074	3.406	0.008	1.384	0.040	5.396	0.033	4.406
$N = 2$ (384)	0.267	5.510	0.961	8.390	0.805	3.406	0.087	1.384	0.440	5.396	0.362	4.406
$N = 3$ (1500)	0.001	0.040	0.003	0.088	0.003	0.054	0.001	0.040	0.003	0.088	0.003	0.054
$N = 5$ (2500)	0.003	0.062	0.006	0.102	0.005	0.039	0.001	0.036	0.001	0.073	0.001	0.039

(a) Zero hold back levels

	Poisson overflow alg.						*On Off overflow alg.*					
N *(# inst.)*	*av* $\Delta\beta$	*max* $\Delta\beta$	*av* $\Delta\alpha$	*max* $\Delta\alpha$	*av* $\Delta\theta$	*max* $\Delta\theta$	*av* $\Delta\beta$	*max* $\Delta\beta$	*av* $\Delta\alpha$	*max* $\Delta\alpha$	*av* $\Delta\theta$	*max* $\Delta\theta$
all N (9000)	0.012	5.113	0.061	6.441	0.055	3.339	0.003	0.851	0.018	4.201	0.016	3.350
$N = 2$ (1000)	0.097	5.113	0.509	6.441	0.465	3.339	0.019	0.851	0.139	4.201	0.125	3.350
$N = 3$ (3000)	0.001	0.062	0.006	0.098	0.006	0.045	0.000	0.025	0.002	0.067	0.002	0.046
$N = 5$ (5000)	0.001	0.022	0.003	0.053	0.003	0.033	0.001	0.016	0.003	0.031	0.002	0.030

(b) With hold back levels

Table 3: Summary of numerical results: average absolute errors and maximal absolute errors, for β_i, α_i, and θ_i, $i = 1, \ldots, N$, expressed as percentages.

significantly better than the Poisson overflow algorithm. Due to the exact calculation of the θ_i's Kranenburg and Van Houtum's algorithm performs slightly better than our On/Off algorithm. However, the errors are of the same order, and hence the On/Off algorithm achieves (almost) the same accuracy as Kranenburg and Van Houtum's algorithm, without the use of the exact θ_i's.

Pooling

In the testbed we distinguish three types of pooling strategies:

- complete overflow: $\sigma_i = \{i+1, \ldots, i+N-1\} \mod N$ (is complete pooling when no hold back levels are set);
- one step: $\sigma_i = \{i+1 \mod N\}$;
- all to 1: $\sigma_i = \{1\}$ for $i \neq 1$ and $\sigma_1 = \emptyset$.

Note that for $N = 2$ the first two strategies coincide. We split out the results according to these pooling strategies, see Table 5. Clearly, the results are best for the all-to-1 strategy, and remarkably of the same order for both the complete overflow and the one step strategies.

Fill rates

We specified four fill rates in the testbed: 50%, 70%, 90%, and 95%. In Table 6 we split out the results according to the fill rate at location 1. The performance of both algorithms increases as the fill rate increases. Note that the number of instance increases in f_1, as the higher f_1 the higher S_1, allowing for more possibilities for h_1.

Calculation time

The calculation time of the Poisson algorithm is extremely fast, it needs merely a fraction of a second to run. In every iteration, for all N warehouses a Markov process on $S_i + 1$ states has to be solved. In the numerical study, on average the algorithm converged in 8.4 iterations, where the On/Off algorithm used on average 9.4 iteration before convergence. Also, the On/Off algorithm requires some more calculation time in every iteration than the Poisson

Compl. pooling	*Poisson overflow alg.*						*On Off overflow alg.*						*Kranenburg & Van Houtum*			
N (# inst.)	*av* $\Delta\beta$	*max* $\Delta\beta$	*av* $\Delta\alpha$	*max* $\Delta\alpha$	*av* $\Delta\theta$	*max* $\Delta\theta$	*av* $\Delta\beta$	*max* $\Delta\beta$	*av* $\Delta\alpha$	*max* $\Delta\alpha$	*av* $\Delta\theta$	*max* $\Delta\theta$	*av* $\Delta\beta$	*max* $\Delta\beta$	*av* $\Delta\alpha$	*max* $\Delta\alpha$
all N (1539)	0.060	5.510	0.215	8.390	0.165	3.406	0.023	1.384	0.112	5.396	0.091	4.406	0.026	1.622	0.026	1.622
$N = 2$ (192)	0.461	5.510	1.681	8.390	1.296	3.406	0.170	1.384	0.868	5.396	0.706	4.406	0.179	1.622	0.179	1.622
$N = 3$ (501)	0.002	0.040	0.008	0.088	0.007	0.054	0.002	0.040	0.008	0.088	0.007	0.054	0.003	0.038	0.003	0.038
$N = 5$ (846)	0.004	0.062	0.005	0.102	0.002	0.039	0.002	0.036	0.003	0.073	0.002	0.039	0.004	0.051	0.004	0.051

Table 4: Complete pooling: summary of numerical results, expressed as percentages. Results of algorithm Kranenburg & Van Houtum added for comparison (for which by construction $\Delta\theta_i \equiv 0$*).*

Pooling strategy	*Poisson overflow alg.*						*On Off overflow alg.*					
N (# inst.)	*av* $\Delta\beta$	*max* $\Delta\beta$	*av* $\Delta\alpha$	*max* $\Delta\alpha$	*av* $\Delta\theta$	*max* $\Delta\theta$	*av* $\Delta\beta$	*max* $\Delta\beta$	*av* $\Delta\alpha$	*max* $\Delta\alpha$	*av* $\Delta\theta$	*max* $\Delta\theta$
compl. overflow (4694)	0.039	5.510	0.162	8.390	0.133	3.406	0.012	1.384	0.066	5.396	0.056	4.406
one step (4749)	0.037	5.510	0.161	8.390	0.133	3.406	0.011	1.384	0.065	5.396	0.055	4.406
all to 1 (4633)	0.006	2.017	0.032	2.420	0.038	2.420	0.001	0.155	0.003	0.191	0.004	0.191

Table 5: Results of Table 2 split out to pooling strategy. Note: for $N = 2$ *'complete overflow' and 'one step' coincide.*

Fill rate at 1	*Poisson overflow alg.*						*On Off overflow alg.*					
N (# inst.)	*av* $\Delta\beta$	*max* $\Delta\beta$	*av* $\Delta\alpha$	*max* $\Delta\alpha$	*av* $\Delta\theta$	*max* $\Delta\theta$	*av* $\Delta\beta$	*max* $\Delta\beta$	*av* $\Delta\alpha$	*max* $\Delta\alpha$	*av* $\Delta\theta$	*max* $\Delta\theta$
$f_1 = 50\%$ (2072)	0.038	5.510	0.134	8.390	0.111	3.406	0.011	1.305	0.051	5.396	0.041	4.406
$f_1 = 70\%$ (2737)	0.025	5.374	0.102	8.328	0.087	2.993	0.007	1.384	0.038	4.720	0.032	3.570
$f_1 = 90\%$ (3802)	0.011	3.418	0.060	5.438	0.055	2.418	0.003	0.989	0.021	3.313	0.018	2.659
$f_1 = 95\%$ (4773)	0.005	1.505	0.031	2.838	0.031	2.418	0.001	0.523	0.010	1.664	0.009	1.349

Table 6: Results of Table 2 split out to fill rate at location 1.

algorithm, but is still reasonably fast. In every iteration a Markov process on $(S_i + 1) \cdot 2^{|d_i|}$ has to be solved, for all N warehouses. Hence, its speed depends on the pooling strategy chosen: complete overflow is slower than 'one step' and 'all to 1', as in the first case the Markov processes contains more states than in the latter two cases. However, there is a large gain compared to exact evaluation, which requires solving the steady state distribution of a Markov process on $\prod_{i=1}^{N}(S_i + 1)$ states. Furthermore, we mention that without hold back levels, both the Poisson and the On/Off algorithm needs on average two extra steps to converge, compared to the case hold back levels are set. We expect this to be due to the fact that with hold back levels, more demands are directly satisfied at the warehouse itself, and hence the overflow stream will be smaller, respectively, be longer turned Off. In that way, the system will more quickly converge.

5 Conclusion and further research

In this paper we extended the Poisson overflow algorithm of Reijnen et al. Reijnen et al. (2009) for hold back levels and introduced the On/Off overflow algorithm, which approximates overflow demand streams more accurately by On/Off processes. In an extensive numerical study, the On/Off algorithm turned out to be much more accurate than the Poisson overflow algorithm.

The presented algorithms can be used to optimize hold back levels. This would provide insights in how much can be gained by these. This, however, needs further research. Also, the presented algorithms can be easily adapted to deal with other replenishment rates. Next to that, one can easily include the random selection for the location where the lateral transshipment originates from.

References

Alfredsson, P. and J. Verrijdt (1999). Modeling emergency supply flexibility in a two-echelon inventory system. *Management Science 45*(10), 1416–1431.

Archibald, T., S. Sassen, and L. Thomas (1997). An optimal policy for a two depot inventory problem with stock transfer. *Management Science 43*(2), 173–183.

Axsäter, S. (1990). Modelling emergency lateral transshipments in inventory systems. *Management Science 36*(11), 1329–1338.

Axsäter, S. (2003). Evaluation of unidirectional lateral transshipments and substitutions in inventory systems. *European Journal of Operational Research 149*(2), 438–447.

Caggiano, K., P. Jackson, J. Muckstadt, and J. Rappold (2009). Efficient computation of time-based customer service levels in a multi-item, multi-echelon supply chain: A practical approach for inventory optimization. *European Journal of Operational Research 199*(3), 744–749.

Gans, N., G. Koole, and A. Mandelbaum (2003). Telephone call centers: Tutorial, review, and research prospects. *Manufacturing and Service Operations Management 5*(2), 79–141.

Grinstead, C. and J. Snell (1997). *Introduction to probability*. American Mathematical Society.

Kranenburg, A. and G. Van Houtum (2009). A new partial pooling structure for spare parts networks. *European Journal of Operational Research 199*(3), 908–921.

Kuczura, A. (1973). The interrupted Poisson process as an overflow process. *The Bell System Technical Journal 52*(3), 437–448.

Kukreja, A., C. Schmidt, and D. Miller (2001). Stocking decisions for low-usage items in a multilocation inventory system. *Management Science 47*(10), 1371–1383.

Kutanoglu, E. (2008). Insights into inventory sharing in service parts logistics systems with time-based service levels. *Computers & Industrial Engineering 54*(3), 341–358.

Kutanoglu, E. and M. Mahajan (2009). An inventory sharing and allocation method for a multi-location service parts logistics network with time-based service levels. *European Journal of Operational Research 194*(3), 728–742.

Liu, J. and C. Lee (2007). Evaluation of inventory policies with unidirectional substitutions. *European Journal of Operational Research 182*(1), 145–163.

Olsson, F. (2010). An inventory model with unidirectional lateral transshipments. *European Journal of Operational Research 200*(3), 725–732.

Paterson, C., G. Kiesmüller, R. Teunter, and K. Glazebrook (2011). Inventory models with lateral transshipments: A review. *European Journal of Operational Research 210*(2), 125–136.

Reijnen, I., T. Tan, and G. Van Houtum (2009). Inventory planning for spare parts networks with delivery time requirements. Beta working paper, Eindhoven University of Technology.

Robinson, L. (1990). Optimal and approximate policies in multiperiod, multilocation inventory models with transshipment. *Operations Research 38*(2), 278–295.

Tagaras, G. and M. Cohen (1992). Pooling in two-location inventory systems with non-negligible replenishment lead times. *Management Science 38*(8), 1067–1083.

Van Wijk, A., I. Adan, and G. Van Houtum (2009). Optimal lateral transshipment policy for a two location inventory problem. Eurandom report, Eindhoven University of Technology.

Wong, H., G. Van Houtum, D. Cattrysse, and D. Oudheusden (2006). Multi-item spare parts systems with lateral transshipments and waiting time constraints. *European Journal of Operational Research 171*(3), 1071–1093.

Xu, K., P. Evers, and M. Fu (2003). Estimating customer service in a two-location continuous review inventory model with emergency transshipments. *European Journal of Operational Research 145*(3), 569–584.

Zhao, H., V. Deshpande, and J. Ryan (2006). Emergency transshipment in decentralized dealer networks: When to send and accept transshipment requests. *Naval Research Logistics 53*(6), 547–567.

Zhao, H., J. Ryan, and V. Deshpande (2008). Optimal dynamic production and inventory transshipment policies for a two-location make-to-stock system. *Operations Research 56*(2), 400–410.