A SIMULATION-OPTIMIZATION APPROACH FOR A SERVICE-CONSTRAINED MULTI-ECHELON DISTRIBUTION NETWORK

Abstract

Academic research on (*s,S*) inventory policies for multi-echelon distribution networks with deterministic lead times, backordering, and fill rate constraints is limited. Inspired by a real-life Dutch food retail case we develop a simulation-optimization approach to optimize (*s,S*) inventory policies in such a setting. We compare the performance of a Nested Bisection Search (NBS) and a novel Scatter Search (SS) metaheuristic using 1280 instances from literature and we derive managerial implications from a real-life case. Results show that the SS outperforms the NBS on solution quality. Additionally, supply chain costs can be saved by allowing lower fill rates at upstream echelons.

Keywords: supply chain performance, multi-echelon inventory, metaheuristics, service-constrained, simulation-optimization, scatter search

1. Introduction

In supply chain operations it is important to manage inventory levels correctly in order to serve customers on time and to minimize inventory investments and ordering costs (Axsäter, 2003c). The optimal amount of stock in a supply chain not only depends on demand, supply and lead times of an individual supply chain entity, but also on the inventory levels of other stock-keeping entities in the supply chain. Multi-echelon inventory models determine optimal inventory levels for all stock-keeping entities in a supply chain by trading off order and holding costs against backorder costs, or by minimizing order and holding costs subject to a given service level constraint (Özer and Xiong, 2008). Since many firms typically do not know their backorder costs, they often set service level constraints to optimize their inventory levels. While service-constrained models are practically more relevant they are more difficult to solve from a computational and analytical standpoint.

This paper studies multi-echelon retail distribution networks with deterministic lead times, backordering, periodic (*s,S*) inventory policies, and fill rate constraints. Such network configurations and inventory policies are common in retail supply chains yet academic research on the topic remains limited (Silver et al., 2009). Our research was inspired by the challenges faced by a Dutch food retailer with a supply chain network consisting of several brands. Similar to other supermarket retailers this retailer is using a periodic review with re-order and order-up-to levels. The ordering systems that are commonplace in supermarket retail are often based on some form of (*s,S*) policy with periodic review (van Donselaar et al., 2006). Determining the right inventory parameters was deemed important to accommodate target fill rates in a supply chain consisting of a number of semi-autonomous decision-making units while minimizing supply chain wide costs. To the best of our knowledge, only Schneider et al. (1995) and Li et al. (2010) provide solution approaches for a similar setting and do so by determining the (*s,S*) policies using respectively power approximations, or via simulation-optimization.

Simulation-optimization models have been recently proposed as an alternative to traditional mathematical programming or simulation approaches. In the literature, the use of mathematical programming methods often requires oversimplifying real-life cases (Peidro et al., 2009). Simulation is capable of modeling more realistic problem settings; however, the process of generating a sufficient number of scenarios that all need to be evaluated before finding (near-)optimal solutions is usually quite time-consuming (Saetta et al., 2012). Alternatively, to obtain fast and accurate solutions, simulation has been integrated with optimization methods in an iterative process (Chu et al., 2015, Fleischhacker et al., 2015, Almeder et al., 2008).

In this paper, we propose a Scatter Search based simulation-optimization method that allows for modelling and solving realistic settings. To this end, Section 2 summarizes the relevant literature on multi-echelon inventory models and specifically on service-constrained models. The problem description and formulation are presented in Section 3. In Section 4, two simulation-optimization approaches are compared. The first method is a Nested Bisection Search heuristic based on the work of Li et al. (2010) and the second method is a novel Scatter Search metaheuristic proposed in this paper. In Section 5, the performance of the solution approaches will be examined using 1280 synthetic problem instances from the literature and a real-life food retail case. Finally, in Section 6 conclusions are drawn and avenues for further research are proposed.

2. Literature review

Inventory management for a single stage in a supply chain is relatively straightforward and well covered in most supply chain management handbooks (e.g. Silver et al., 2009, Simchi-Levi et al., 2009, Slack et al., 2010). However, if inventory is held in more than one stage of the supply chain (i.e. in a network) then determining optimal inventory parameters becomes more difficult (Ravi Ravindran and Warsing Jr., 2012). In case of local control, every entity (or installation) in the network controls its own inventory, which is also known as an installation stock policy. At the same time, the entities in this network are dependent on each other since demand at a downstream stage triggers an order at an upstream stage. Inventory levels at different stages thus influence each other (Schneider et al., 1995). In the case of central control, where one entity manages all inventories in a network, the echelon inventory position is used to manage the inventory level of all echelons. The echelon inventory of a certain location consists of the installation inventory of that location plus all downstream installation inventories (Axsäter, 2003c). When applying echelon stock policies information on inventory levels should be shared between the locations of the different echelons, which can be technically and organizationally challenging to implement in industry (Tüshaus and Wahl, 1998).

Clark and Scarf (1960) introduced the first multi-echelon inventory model using a serial network in which each stage has a single supplier and a single customer. Building on the seminal Clark and Scarf (1960) paper, a variety of multi-echelon inventory models were developed for other network structures, including diverging distribution networks (e.g. Rong et al., 2012) and converging assembly networks (e.g. Cheng et al., 2002). In the literature, multi-echelon inventory models are regularly divided into backorder-cost (also full cost) and service-constrained (also partial cost with service level constraints) models (Chen and Krass, 2001, Özer and Xiong, 2008). Since backorder cost are difficult to determine, service-constrained models are often more applicable to real-life settings. Furthermore, service levels are known as the most used performance measures (Silver et al., 1998). Generally, the objectives of papers on multi-echelon inventory networks are to minimize cost and find optimal inventory parameters. For service-constrained models an additional requirement is added to safeguard a given service level while minimizing costs and inventory levels. Below, we first discuss articles that focus on determining service levels in a network (without minimizing costs) and then articles that mainly focus on minimizing costs while enhancing a certain service level.

In the first stream of papers, the first paper to introduce service-constrained distribution models was by Rosenbaum (1981), who uses simulation to test a heuristic for determining the best combination of distribution center (DC) service levels to achieve a given external customer service level. Desmet et al. (2010) use a similar approach by approximating the effect of a reduction in the warehouse fill rate on the system safety stock (the total safety stock in a network). Schwarz et al. (1985) develop approximations and heuristics to maximize the system fill rate with a constraint on the system safety stock. Tüshaus and Wahl (1998) provide a robust and numerically inexpensive cycle-based approximate mathematical representation for a two-echelon distribution system with service level constraints, which can be used for determining performance measures or for use with an optimization method. Caggiano et al. (2009) approximate and simulate system-wide optimal (echelon) inventory levels by computing channel fill rates for time-based service levels. These papers focus on the influence of service levels on the system, or on computing target service levels.

A second stream of papers focuses on minimizing costs while maintaining a certain service level. van der Heijden (2000) proposes an approximate optimization procedure to find optimal base-stock policies for a multi-echelon distribution network sequentially using target fill rates. He assumes that perfect information is available and therefore uses an echelon stock policy with balanced stock rationing in case of a stock-out. Balanced stock rationing does not account for differences in holding costs of the downstream locations, which may be relevant in situations with heterogeneous downstream locations. Van der Heijden (2000) argues that there is a trade-off between guaranteed service levels with low costs versus minimal costs with reasonable service levels. Simchi-Levi and Zhao (2005) obtain optimal base-stock policies for a variety of network configurations while meeting certain service level requirements of external customers. They propose an algorithm that is based on dynamic programming and the two-moment approximation of Graves and Willems (2000). Simchi-Levi and Zhao (2005) assume that demand is Poisson distributed, the probability distributions of the transportation lead times are known; their model incorporates a continuous installation stock policy. Özer and Xiong (2008) provide an exact algorithm, heuristics, and approximations for a two-echelon distribution system to set optimal base-stock levels by minimizing the average inventory holding costs subject to fill rate constraints. They apply a continuous review policy without order cost, which implies that every time demand is faced, an order is immediately placed. Their exact algorithm assumes Poisson-distributed demand and its use requires substantial CPU time, data and advanced modelling knowledge. As such, to gain insights on system performance the authors also provide heuristics and closed-form approximations but these methods do not allow for determining optimal inventory parameters. Similarly, Fleischhacker et al. (2015) propose a non-linear and a deterministic linear integer optimization model to determine shipping quantities and inventory levels for a divergent network with a fill rate constraint. They assume Poisson distributed demand, a finite time horizon, a continuous review policy and constant lead times. Chu et al. (2015) recently described a simulation-based optimization framework for a divergent multi-echelon network under stochastic demand and stochastic lead times. They propose a three step procedure to optimize (*r,Q*) policies by minimizing total cost while maintaining acceptable fill rates. First, they apply an agent-based system that simulates the inventory system and returns performance measures, second they use a Monte Carlo method and third, they developed a cutting plane algorithm to determine optimal inventory parameters. Tsai and Liu (2015) study a multi-item, multi-echelon spare parts inventory system and minimize costs with the expected response time as service measure. They follow a continuous review base-stock policy, which implies that every time demand is faced an order is placed immediately. They develop two different algorithms within a simulation-based optimization framework, a ranking and selection method and a stochastic genetic algorithm and compare these to a sample-average-approximation. As in the models of Chu et al. (2015) and Tsai and Liu (2015), simulation is used here to model the inventory dynamics; using this approach demand and lead times can be modeled using any probability distribution.

In retail supply chains periodic (*s,S*) inventory policies are fairly common (Silver et al., 2009). Such a periodic (*s,S*) policy implies a different optimization procedure than a base-stock policy, which is widely used in the related literature (van der Heijden, 2000, Simchi-Levi and Zhao, 2005, Özer and Xiong, 2008, Fleischhacker et al., 2015, Tsai and Liu, 2015). For an (*s,S*) policy, decision variables *s* and *S* are dependent on each other as *S* always needs to be strictly larger than *s*, which makes the optimization procedure inherently more difficult than for a base-stock policy. For example, when using a base-stock policy every review period an order is placed by default, which may not be desirable if high fixed order costs are involved. Of course review periods can be adapted to avoid frequent ordering and thus high order costs, but instead (*s,S*) policies could be implemented. In the past, (*s,S*) policies have been applied mainly to single-echelon models (e.g. Schneider and Ringuest, 1990, Bashyam and Fu, 1998, Chen and Krass, 2001, Moors and Strijbosch, 2002, Silver et al., 2009) or to serial systems (e.g. Eltawil and Elnahar, 2007).

Service-constrained models and (*s,S*) inventory policies are common in retail supply chains, but research on these topics remains limited. To the best of our knowledge only two papers discuss multi-echelon divergent networks with an (*s,S*) inventory policy under service-level constraints. Schneider et al. (1995) determine optimal safety stock placements that minimize total costs using a service level measure as decision variable. They propose power approximations by assuming independent and identically distributed demand, stock-out cost at the stores and excluding partial delivery. Additionally, they model stochastic lead times and non-stock out probabilities. Their model does not suit our problem setting as we do not look into optimal safety stock placement, but into determining optimal (*s,S*) inventory policies for given locations. Li et al. (2010) propose a simulation-optimization framework with a bisection search to obtain optimal (*s,S*) inventory policies that minimize costs for a two-echelon distribution network. To obtain near-optimal inventory policies for thousands of products quickly, they propose approximations and regression methods that may be used to characterize the inventory policies for similar products. Customer demand and all lead times are assumed to be normally distributed. Additionally, if inventory levels are insufficient, customer demand is considered lost, but replenishment orders are backlogged.

We propose two different optimization methods to determine near-optimal values for (*s,S*) inventory policies in multi-echelon distribution networks. The first method, Nested Bisection Search, is based on an existing solution method, the bisection search of Li et al. (2010), for a similar multi-echelon inventory problem. The second is a newly proposed evolutionary approach based on a Scatter Search metaheuristic, which is able to intensify and diversify the solution search space systematically (Glover et al., 2003).

3. Problem description and formulation

Consider a single-item service-constrained inventory optimization problem in an *n*-echelon distribution network (for an example of such a network see Figure 1). Each node (i.e., location) is replenished from a designated node at the next-higher echelon (the parent location). In case the parent location has sufficient inventory, the replenishment order will arrive at location $j \in J$ after a deterministic lead time L_j. Demand from the locations in the (next) downstream echelon (the child locations) is then fulfilled. Locations at the lowest echelon (echelon $v=n$) face external aggregated (customer) demand from external customer demand points (in this paper also shortened to 'external demand') and the highest echelon (echelon $v=1$) has infinite external supply. All locations control their inventory levels locally using a periodic (*s,S*) inventory policy. Unmet demand is backordered at all locations in the network.

In the multi-echelon literature using echelon stock, it is common to use an allocation policy that takes customer service levels at downstream locations into account, for example through balanced stock rationing (e.g. Lagodimos, 1992, Diks et al., 1996, Van der Heijden et al., 1997). However, as we do not study an echelon stock policy, this is not suitable. Studies with a similar problem setting as we do usually have either a first-come first-served policy (e.g. Desmet et al., 2010), or do not ship anything when there is insufficient inventory to fulfill all orders (e.g. Li et al., 2010). In this paper, we allocate the remaining inventory of one location based on the proportion of outstanding orders of one child location as compared to all outstanding orders of all child locations, following the proportional allocation rule of Tüshaus and Wahl (1998). First (a part of) the backorders of the last period are fulfilled and in case there is inventory left, (a part of) the new demand is fulfilled. The fill rate is used as a service level as it is the most common service level in practice. It is defined as the fraction of demand that can be immediately satisfied from available inventory (Muckstadt, 2006, p.48). Shipments between locations of the same echelon (such as in Paterson et al., 2011) are not allowed. Lead times are assumed to be deterministic. The objective is to minimize the expected total costs, including inventory holding and ordering costs, subject to minimal fill rate requirements at the locations of echelon *v=n*. The expected inventory holding cost equals the expected number of products on inventory times the inventory holding cost. The ordering cost equals the expected number of transport units ordered times the cost for ordering one transport unit, which consists of multiple units of the same product.

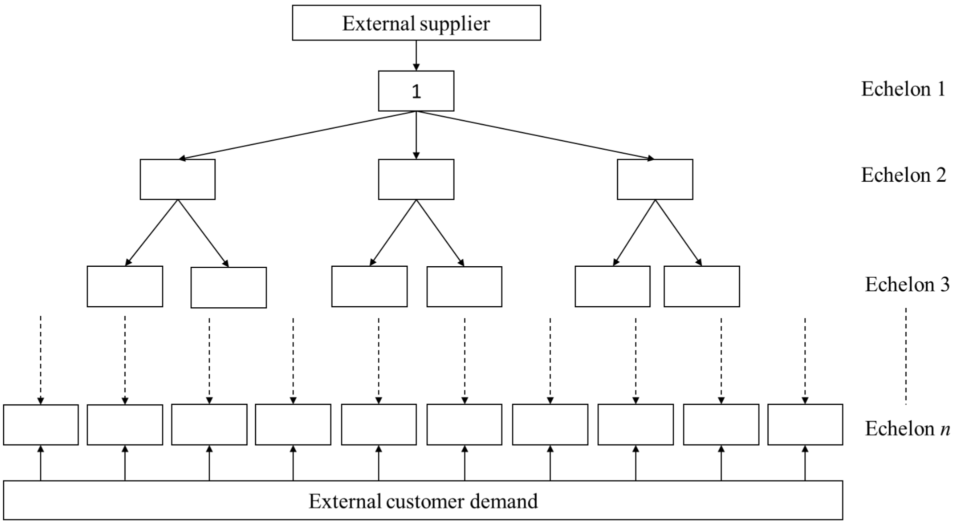

Figure 1: An *n*-echelon distribution network

We use the following notation:

Network parameters

J the set of locations, indexed by j

$\mathrm{J^v}$ the subset of locations at echelon $\mathrm{v} = 1, 2, \dots, \mathrm{n}$, with $\cup_{\mathrm{v}=1}^{\mathrm{n}} \mathrm{J^v} = \mathrm{J}$, and $\mathrm{J^{v_1}} \cap \mathrm{J^{v_2}} = \emptyset, \mathrm{v_1} \neq \mathrm{v_2}$

$\mathrm{p(j)}$ the parent location of location j in the distribution network, $j \in J \backslash J^1$

$\mathrm{R(j)}$ the child locations of location j in the distribution network, $\mathrm{j} \in \mathrm{J} \backslash \mathrm{J^n}$

Y the number of products in one transport unit (e.g. a pallet)

Inventory policies

$\mathrm{s_j}$ the re-order level of location $j \in \mathrm{J}$

$\mathrm{S_j}$ the order-up-to level of location $j \in \mathrm{J}$

Performance measure parameters

$\mathrm{F_j}$ the minimal fill rate requirement of location $\mathrm{j} \in \mathrm{J^n}$

$\mathrm{h_j}$ the holding cost per product and per period at location $j \in \mathrm{J}$

$\mathrm{K_j}$ the order cost per transport unit at location $\mathrm{j} \in \mathrm{J}$

Performance measurements

$\mathrm{E_j(s,S)}$ the expected shortage (as a percentage of the total demand) of location $\mathrm{j} \in \mathrm{J^n}$ for a given (*s*,*S*) policy

$\mathrm{I_j(s,S)}$ the expected inventory level of location $j \in \mathrm{J}$ for a given (*s*,*S*) policy

$\mathrm{P_j(s,S)}$ the expected order quantity (in number of transport units) of location $j \in \mathrm{J}$ for a given (*s*,*S*) policy

The multi-echelon inventory optimization problem with minimal fill-rate requirements (MEIO) is formulated as follows.

$$
\begin{array}{llll}
\text{(MEIO):} & \min & \sum_{j \in J} h_j \, I_j(\mathrm{s,S}) + \sum_{j \in J} K_j \, P_j(\mathrm{s,S}), & \text{(1A)} \\
 & \text{s.t.} & 1 - \mathrm{E_j(s,S)} \geq \mathrm{F_j}\,, \qquad \forall\, j \in \mathrm{J^n}, & \text{(1B)} \\
 & & S_j > s_j\,, \qquad \forall\, j \in \mathrm{J}, & \text{(1C)} \\
 & & s_{j,}\, S_j \geq 0, \ \text{integers}, \qquad \forall\, j \in \mathrm{J}. & \text{(1D)}
\end{array}
$$

The objective function (1A) minimizes the total expected inventory holding costs and ordering costs of all locations. Constraints (1B) ensure that all minimal fill-rate requirements at the lowest echelon locations are satisfied. As formulated by Constraints (1C) and (1D), we only consider (*s*,*S*) policies that are nonnegative integer numbers and order-up-to levels always need to be higher than re-order levels. The (*s*,*S*) policies for all locations are inter-related decisions and, ideally, simultaneously optimized. All inventory dynamics related to the objective function and side constraints are explained in the simulation model in Section 4.3.

4. A simulation – optimization approach

Many real-world problems are too complex to compute performance measurements and optimal decision variables analytically due to nonlinearities, combinatorial relationships, and uncertainties (Glover et al., 1999). Computer simulation can be used to evaluate complex systems; however, this approach is limited because (a) it is not able to give the best or optimal solution in most cases and (b) a large number of scenarios need to be evaluated (Tekin and Sabuncuoglu, 2004). To find a near-optimal solution to a complex problem it can be useful to integrate simulation and optimization. Such an approach uses an optimization method to find the best values of given decision variables to minimize/maximize an objective function; these solutions are then evaluated using a simulation model that incorporates uncertainty (Ólaffson and Kim, 2002, Jalali and Nieuwenhuyse, 2015).

Optimization for simulation, which is also known as *optimization via simulation* or *simulation optimization*, consists of different techniques to optimize stochastic simulations (Amaran et al., 2016). Simulation-optimization first evaluates an optimizer's candidate solutions, often using discrete-event simulation in which the system can change only when a certain event occurs (Law, 2015), and then returns the performances measurements to the optimizer. The optimization method uses the outputs of the simulation to decrease the search space (which consists of possible solutions) and to make decisions regarding the next trial solution (Fu, 2002, April et al., 2003, Amaran et al., 2016).

In the next sections, we explain two optimization methods that are included in our simulation-optimization approach. In Section 4.1, a Nested Bisection Search based on Li et al. (2010) is explained. Afterwards, a Scatter Search metaheuristic is developed in Section 4.2. Section 4.3 explains the simulation model that is used as an evaluative function for the optimization methods.

4.1 Determining (*s,S*) values: Nested Bisection Search (NBS)

Li et al. (2010) propose a simulation-based optimization framework as the best method in terms of solution quality to obtain optimal re-order levels for a single item in a two-echelon distribution network. By using a bisection search, first an initial re-order level is set for a location in the highest echelon and second, the re-order levels of all downstream locations are updated using again a bisection search. The re-order levels are updated recursively, from the highest to the lowest echelon. The simulation model evaluates every newly obtained solution and returns performance measurements. As Li et al. (2010) assume fixed order quantities, the order-up-to level can be obtained easily once the re-order level has been determined.

In this paper, we adjust the Li et al. (2010) heuristic to obtain (near-) optimal (*s,S*) policies for our problem setting. We model backorders instead of lost sales, we use different demand distributions, assume deterministic lead times instead of stochastic lead times and we follow a different inventory allocation policy (i.e. proportional rationing). However, similar to Li et al. (2010) we determine the fixed difference between the re-order level and the order-up-to level for each location using the standard EOQ formula with the expected demand for that location (EOQ_j).

As the work of Li et al. (2010) provided insufficient detail on the exact implementation of the Nested Bisection Search (NBS), we used the work of Benkherouf (1995) on bisection search for detailing the NBS algorithm (see Algorithm 1). We determine the re-order levels for all locations that minimize expected total cost while satisfying fill rate requirements. The NBS algorithm keeps track of which locations have been updated already and the locations in each echelon are always processed in the same sequence. Additionally, to speed up the search the search spaces of the (*s,S*) values in the NBS and Scatter Search (SS) are restricted within lower and upper limits. For each location all lower limits for the (*s,S*) values have been set to zero and all upper limits to twenty times the average demand (of a time period). Based on preliminary testing we concluded that these upper limits are sufficiently high. Re-order level *s* is initialized to the average demand of a location multiplied by the lead time to that location. Order-up-to level *S* is initialized to re-order level *s* plus the *EOQ* value of that location.

Algorithm 1: NBS

```
Main Procedure
        Step 1: Set the initial (s,S) values.
        Step 2: Calculate the Economic Order Quantity $EOQ_j$ for each location $j \in J$.
        Step 3: Set the current location to the root location (location 1 in echelon 1) and call the
        bisection subroutine (step 4 to 7) recursively.

Bisection Search Subroutine
•   Step 4: Let j denote the current location.
•   Step 5: Set a to the lower limit $ll_j$ and b to the upper limit $ul_j$.
•   Step 6: Let $\varepsilon = 1$ be the minimal difference between $a$ and $b$.
•   Step 7: While $b - a > \varepsilon$
        o   Set the mid-point $m$ as
                $m = \left\lceil \frac{b+a}{2} \right\rceil$.
        o   Set $s_j = m$ and $S_j = s_j + EOQ_j$.
        o   If location $j$ is in echelon 2,
                ▪ Evaluate the total expected cost and expected fill rates with MEIO given by (1) by
                  running the simulation model for the entire network, which will be explained in
                  Section 4.3.
            Else,
                ▪ For each child location in R(j), call the bisection subroutine (step 4 to 7)
                  recursively to evaluate the total expected cost and the expected fill rates.
```

- Update the best solution to the current (***s,S***) values if the current solution has a lower expected cost and the fill rate requirements are satisfied.
- If the fill-rate requirements are satisfied, set $b = m$; else set $a = m$.

End while

- *Step 8:* Return the best (***s,S***) values with the expected cost and fill rates.

4.2 Determining (*s,S*) values: Scatter Search metaheuristic (SS)

Multiple approaches exist for optimizing decision variables in a simulation optimization approach (e.g. stochastic approximation, response surface methodology, and sample path optimization (April et al., 2003)). Metaheuristic algorithms are highly relevant for complex optimization problems as they provide high quality solutions in short computing time (Juan et al., 2015, Figueira and Almada-Lobo, 2014) and they are effective in solving multi-echelon inventory optimization problems (Paul and Rajendran, 2011). Metaheuristic approaches can guide other procedures, such as heuristics, to overcome local optimality for complex problems (Fu et al., 2005). They are applicable to simulation models with discrete decision variables, a large or near-infinite space of feasible solutions, and a stochastic environment (Swisher et al., 2000, Jalali and Nieuwenhuyse, 2015). Due to their effectiveness and general applicability, metaheuristics can be seen as one of the most practical approaches to solve complex real-life problems (Ólafsson, 2006). Additionally, Juan et al. (2015) argues that for complex real-life problems, it is in general preferred to obtain an approximate solution to an accurate model of a real system using metaheuristics with simulation over obtaining an optimal solution to an oversimplified model. Commonly, four types of metaheuristic approaches are applied in simulation optimization: Simulated Annealing, Genetic Algorithms, Tabu Search, and Scatter Search (Keskin et al., 2010). Fu et al (2005) state that the latter two are by far the most effective methods. The use of Tabu Search and Scatter Search is explained in e.g. Vojvodic et al. (2016) and Goh et al. (2017).

Both Tabu Search and Scatter Search make use of adaptive memory to store best solutions and differ from other metaheuristic approaches by not heavily relying on randomization. However, Tabu Search is applied from the perspective of adaptive memory and Scatter Search focuses on an evolutionary approach which generates new trial solutions based on existing solutions(Glover, 2006).

In this paper we adopt the Scatter Search (SS) method introduced by Glover (1977) as SS uses strategies to diversify and intensify the search which have proven to be effective in a variety of optimization problems (Martí et al., 2006, Russell and Chiang, 2006). Based on the basic SS framework (e.g. Laguna, 2014, Martí et al., 2006, Glover et al., 2003), which consists of five steps, a tailor-made solution approach with regard to how the steps are implemented is proposed for optimizing the (*s,S*) policies at all locations in *J.* The metaheuristic searches for promising inventory policies on the following solution space:

$$X \equiv \{(s,S) \in \mathbb{Z}^{|J|} \times \mathbb{Z}^{|J|} : ll_j \leq s_j < ul_j, ll_j < S_j \leq ul_j, s_j < S_j, \forall j \in J\} \quad (2)$$

where ll_j and ul_j are the lower and upper limits of the inventory parameters for location $j \in J$. Algorithm 2, based on Laguna (2014), provides an overview of the different steps in the SS metaheuristic. The next sections describe the major components of the SS in more detail: the diversification generation method for generating a diverse set of initial solutions, an improvement method including bisection search and a local search, updating the reference set from which solutions are used to generate new solutions, and a combination method using weighted linear combinations of solutions to obtain new solutions. Details about the specific parameter values used are reported in Section 5.1.

Algorithm 2: Scatter Search

```
• Diversification generation
• Improvement method
• While stopping criteria not satisfied do
    o Reference set update
    o While new reference solutions do
        ▪ Combination method
        ▪ Improvement method
        ▪ Reference set update
    o End while
    o Rebuild Reference set
• End while
```

4.2.1 Diversification generation

As SS aims to obtain better solutions by using combined solutions instead of original values, we start with generating a diverse set of solutions to initialize the search (Glover et al., 2003). A population pool is generated which can be done in multiple ways. We propose the following two heuristics (Construction Heuristics) for starting up the SS method:

- C1: For all $j \in J$, we set s_j as an integer number (denoted as $\tilde{s}_j$) that is randomly picked in $[ll_j$, $ul_j)$. Afterwards, we set S_j to an integer number ($\tilde{S}_j$) randomly picked in $(\tilde{s}_j$, $ul_j]$.
- C2: For all $j \in J$, s_j and S_j are set to integer numbers (denoted as $\tilde{s}_j$ and $\tilde{S}_j$ respectively) that are randomly picked in $[ll_j$, $ul_j]$. If $\tilde{s}_j > \tilde{S}_j$, we swap the values of $\tilde{s}_j$ and $\tilde{S}_j$; otherwise, if $\tilde{s}_j = \tilde{S}_j$ and $\tilde{S}_j < ul_j$, we increase $\tilde{S}_j$ by one. If $ll_j < \tilde{s}_j$ and $\tilde{s}_j = \tilde{S}_j$, we decrease $\tilde{s}_j$ by one.

4.2.2 Improvement method

In this step, the aim is to improve the initial solutions in terms of quality; in the case of non-feasible solutions the aim is to make them feasible. Several improvement methods can be used and the usual rule is to stop the search as soon as no improvement in the neighborhood of the current solution can be found (Laguna, 2014). In this paper, we propose two improvement method versions (Improvement Heuristics): a sequential and parallel bisection search for processing the locations.

- I1: Sequential version: At each iteration, one of the locations is selected and its re-order level and order-up-to level are optimized using a bisection approach. The locations are improved sequentially and locations in this sequence are selected randomly. If the solution cannot be further improved, the search is perturbed with an approach reminiscent from gradient search (Ólaffson and Kim, 2002) which we will define as the local search. By exploring the impact of a one-unit increase or decrease on the individual (*s,S*) values, the direction for cost improvement is determined. The new (*s,S*) values are obtained by applying a fixed step size to the previous (*s,S*) values in the direction just explored. If this solution gives again an improvement the step size increases with a predetermined amount, if the solution did not improve compared to the previous solution, the step size decreases with the same predetermined amount. This procedure stops when the maximum number of non-improving iterations is reached.
- I2: Parallel version: At each iteration, first the re-order levels of all locations and then the order-up-to levels of all locations are optimized. The same local search as in I1 is applied if no solution can be further improved using the bisection approach. However, different from the sequential version, at each iteration all locations are improved in parallel, instead of one location at the time.

The improvement heuristics always terminate after reaching a maximum number of evaluated solutions for all locations together.

4.2.3 Reference set update

The scatter search maintains a reference set of multiple solutions (the reference solutions) that are used to generate new solutions using the Combination method (see Section 4.2.4) (Martí et al., 2006). At each iteration of the search, the reference solutions are picked from the population pool as follows. The first 50% of the solutions we take into account are the best solutions based on the objective function values in the population pool. The second half of solutions are picked one-by-one based on the minimum distance to each of the reference solutions, i.e., the set of minimum distances. The solutions with the largest values of the minimum distances to the reference solutions are chosen. The set of solutions is thusly diversified in order to avoid staying in a local optimum. The distances are measured based on Euclidian distances, such that for any two solutions (s', S') and (s'', S'') in the solution space X (Equation 2) the distance of the two solutions is given by:

$$\sqrt{\sum_{j \in J} (s_j' - s_j'')^2} + \sqrt{\sum_{j \in J} (S_j' - S_j'')^2} \tag{3}$$

4.2.4 Combination method

To diversify the search, the reference set solutions are combined at each iteration of the scatter search by weighted linear combinations to generate new solutions, which are then added to the solution pool. For any two distinct solutions (s', S') and (s'', S'') in the reference set, with at least one of them being marked as "not yet processed", a new solution (s^*, S^*) is obtained as follows:

$$s_j^* = \min\{s_j', s_j''\} + |s_j' - s_j''|/2 \tag{4A}$$

$$S_j^* = \min\{S_j', S_j''\} + |S_j' - S_j''|/2 \tag{4B}$$

If (s^*, S^*) contains fractional values, they are rounded to the nearest integer values. As this new solution is the midpoint of two other solutions, it avoids already existing solutions and thereby diversifies the search.

Initially, all reference solutions are marked as "not yet processed." After the solution combination, all the current reference solutions are marked as "processed." New reference solutions obtained are initialized with "not yet processed." The SS terminates when all the reference solutions are marked as "processed" and no new reference solution can be obtained. When the SS is terminated, the best feasible solution is given.

4.3 Simulation model

The (*s,S*) solutions proposed by the optimization methods will be evaluated by a discrete-event simulation model to determine expected fill rates and costs. Each time period starts with supplies based on past orders (Section 4.3.1), then the locations determine how much to ship to their child locations based on demand, backorders and available on-hand inventory (Section 4.3.2). After products are shipped to their child locations, each location determines how much to order to replenish their inventory (Section 4.3.3). The performance measurements for evaluating the solutions are discussed in Section 4.3.4.

Let T be the set of time periods (indexed by t) and let T' *be* the time periods for which the performance is measured (excluding the warm-up period). For all locations $j \in J^n$ and time periods

$t \in T$, let d_j^t be the demand from the external customer demand points faced at location *j* in time period *t.* The demands follow a given probability distribution and are revealed at the beginning of each time period. The simulation procedure iteratively updates the following variables for each of the time periods.

A_j^t quantity arrived at location *j* at time period *t* from its parent location or external supplier, $j \in J, t \in T$

Z_j^t total quantity shipped at location *j* to all child locations $r \in R(j)$ or external customer demand points at time period *t*, $j \in J, t \in T$

ZB_{jr}^t quantity shipped at location *j* to child locations $r \in R(j)$ for satisfying the backorders, $j \in J \backslash J^n, t \in T$

$\widetilde{ZB}_j^t$ quantity shipped at location *j* to satisfy the backorders of external demand, $j \in J^n, t \in T$

ZD_{jr}^t quantity shipped at location *j* to child locations $r \in R(j)$ for meeting the orders of time period *t-1*, $j \in J \backslash J^n, t \in T$

$\widetilde{ZD}_j^t$ quantity shipped at location *j* for meeting the external demand of time period *t*, $j \in J^n, t \in T$

I_j^t inventory level at location *j* at time period *t,* $j \in J, t \in T$

B_{jr}^t backorders between location *j* and child locations $r \in R(j)$ at time period *t*, $j \in J \backslash J^n, t \in T$

$\widetilde{B}_j^t$ backorders between location *j* and the external customer demand points at time period *t*, $j \in J^n, t \in T$

O_j^t outstanding orders of location *j* at time period *t*, $j \in J, t \in T$

IP_j^t inventory position at location *j* at time period *t,* $j \in J, t \in T$

V_j^t products short at location *j* at time period *t*, $j \in J, t \in T$

Q_j^t quantity ordered by location *j* at time period *t*, $j \in J, t \in T$

For the locations in the lowest echelon beginning inventory levels are initialized to the average demand multiplied by the lead time. For the locations not in the lowest echelon, the beginning inventory levels are initialized to the sum of the average demands of the children locations multiplied by the lead time to that location. All other variables listed above are initialized to 0. The simulation procedure updates the variables for each iteration, as described in Sections 4.3.1-4.3.4.

In Section 4.3.1, at the beginning of time period *t*, the products ordered from the external supplier or parent location arrive at location *j* (denoted as $\mathrm{A}_{\mathrm{j}}^{\mathrm{t}}$). Afterwards, as described in Section 4.3.2, location *j* has to determine how much it can ship to its child locations or external customer demand points based on the demand faced by location *j*. The amount to ship ($\mathrm{Z}_{\mathrm{j}}^{\mathrm{t}}$) depends on the inventory level of the last time period of location *j* ($\mathrm{I}_{\mathrm{j}}^{\mathrm{t-1}}$), the backorders to the child locations or external customer demand points ($\mathrm{B}_{\mathrm{jr}}^{\mathrm{t}}$ or $\widetilde{\mathrm{B}}_{\mathrm{j}}^{\mathrm{t}}$) and the number of products arrived at the beginning of the time period. If there is on-hand inventory available, first (a part of) the backorders are fulfilled ($\mathrm{ZB}_{\mathrm{jr}}^{\mathrm{t}}$ or $\widetilde{\mathrm{ZB}}_{\mathrm{j}}^{\mathrm{t}}$) and then (a part of) the demand ($\mathrm{ZD}_{\mathrm{jr}}^{\mathrm{t}}$ or $\widetilde{\mathrm{ZD}}_{\mathrm{j}}^{\mathrm{t}}$). After all shipments are made, the inventory level, backorders to child locations or external customer demand points and the shortage per cycle ($\mathrm{V}_{\mathrm{j}}^{\mathrm{t}}$) are updated. In Section 4.3.3, at the end of time period *t*, a new order ($\mathrm{Q}_{\mathrm{j}}^{\mathrm{t}}$) can be placed, depending on the inventory position ($\mathrm{IP}_{\mathrm{j}}^{\mathrm{t}}$) and the (s_j, S_j) policy of location *j*. Last, the number of ordered products outstanding ($\mathrm{O}_{\mathrm{j}}^{\mathrm{t}}$) is updated. In Section 4.3.4, performance measurements for evaluating the solutions are discussed.

4.3.1 Receiving orders

All orders $\mathrm{Q}_{\mathrm{j}}^{\mathrm{t}}$ placed by locations in the highest echelon, take one time period to arrive as demand at the external supplier (at *t+1*). This supplier has infinite supply and ships the demand to location $\mathrm{j} \in \mathrm{J}^1$, which takes lead time $\mathrm{L_j}$ to arrive. As such, the arrival quantity $\mathrm{A}_{\mathrm{j}}^{\mathrm{t}}$ that arrives at the beginning of time period *t* is equal to the order that was placed lead time $\mathrm{L_j} + 1$ time periods ago by location $\mathrm{j} \in \mathrm{J}^1$:

$$\mathrm{A}_{\mathrm{j}}^{\mathrm{t}} = \mathrm{Q}_{\mathrm{j}}^{\mathrm{t-1-L_j}}, \qquad \forall \mathrm{j} \in \mathrm{J}^1, \mathrm{t} \in \mathrm{T}. \tag{5A}$$

For all locations *j* not in the highest echelon ($\mathrm{J} \backslash \mathrm{J}^1$), the arrival quantity $\mathrm{A}_{\mathrm{j}}^{\mathrm{t}}$ of time period *t* is equal to the total shipment quantity (for backorders and new orders) of their parent location *p(j)* at time period $\mathrm{t} - \mathrm{L_j}$:

$$\mathrm{A}_{\mathrm{j}}^{\mathrm{t}} = \mathrm{ZB}_{\mathrm{p(j),j}}^{\mathrm{t-L_j}} + \mathrm{ZD}_{\mathrm{p(j),j}}^{\mathrm{t-L_j}}, \qquad \forall \mathrm{j} \in \mathrm{J} \backslash \mathrm{J}^1, \mathrm{t} \in \mathrm{T}. \tag{5B}$$

4.3.2 Meeting the demand

After the replenishment order arrives, the shipment quantity to the children nodes or external customer demand points can be determined based on the available on-hand inventory. The shipment quantity has two components, first the shipment quantity for the backorders $\mathrm{ZB}_{\mathrm{jr}}^{\mathrm{t}}$ is determined and second, for the demand faced ZD_{jr}^{t} that period. For all locations $j \in J \setminus J^n$, the orders of the child locations arrive the next time period and the total shipment quantity Z_j^t to child locations $r \in R(j)$ at time period *t* is given by:

$$Z_j^t = \sum_{r\in R(j)} \mathrm{ZB}_{jr}^t + \sum_{r\in R(j)} \mathrm{ZD}_{jr}^t, \qquad \forall j \in J \setminus J^n, t \in T \tag{6A}$$

where,

$$\mathrm{ZB}_{jr}^t = \begin{cases} \left\lfloor \frac{B_{jr}^{t-1}}{\sum_{r\in R(j)} B_{jr}^{t-1}} \left(I_j^{t-1} + A_j^t \right) \right\rfloor, \text{if} \sum_{r\in R(j)} B_{jr}^{t-1} > I_j^{t-1} + A_j^t; \\ B_{jr}^{t-1}, \text{ otherwise,} \end{cases} \qquad \forall j \in J \setminus J^n, r \in R(j), t \in T. \tag{6B}$$

$$\mathrm{ZD}_{jr}^t = \begin{cases} \left\lfloor \frac{Q_r^{t-1}}{\sum_{r\in R(j)} Q_r^{t-1}} \left(I_j^{t-1} + A_j^t - \sum_{r\in R(j)} ZB_{jr}^t \right) \right\rfloor, \text{if} \sum_{r\in R(j)} Q_r^{t-1} > I_j^{t-1} + A_j^t - \sum_{r\in R(j)} ZB_{jr}^t; \\ Q_r^{t-1}, \text{ otherwise,} \end{cases}$$
$$\forall j \in J \setminus J^n, r \in R(j), t \in T. \tag{6C}$$

Note that rounding down the shipment quantities in case of a shortage may result in a small number of requested products not being shipped. At most this amount will be equal to the number of child locations which requested products minus one.

If the on-hand inventory is insufficient to fulfill all backorders and demand, all unshipped products need to be backordered. The number of backorders B_{jr}^t between location *j* and its child locations depends on the backorders of last period, the new order (placed at time period *t-1*) and the shipped quantities of time period *t*:

$$B_{jr}^t = B_{jr}^{t-1} + Q_r^{t-1} - \mathrm{ZB}_{jr}^t - \mathrm{ZD}_{jr}^t, \qquad \forall j \in J \setminus J^n, r \in R(j), \ t \in T. \tag{7}$$

To determine the fill rate of location *j* we have to calculate the expected shortage per cycle V_j^t, which equals the amount of orders of time period *t-1* that cannot immediately be shipped in time period *t*:

$$V_j^t = \sum_{r\in R(j)} Q_r^{t-1} - \sum_{r\in R(j)} \mathrm{ZD}_{jr}^t, \quad \forall j \in J \setminus J^n, t \in T. \tag{8}$$

All locations $j \in J^n$ in the lowest echelon face external demand (instead of demand from child locations). These locations ship products to customers that represent the external customer demand points. Similar to the other locations, the locations in the lowest echelon first determine the shipment quantity based on the outstanding backorders $(\widetilde{\mathrm{ZB}}_j^t)$ and afterwards the shipment quantity of the demand faced in time period *t* $(\widetilde{\mathrm{ZD}}_j^t)$. First the shipment quantities are determined (Equation 9A-C), second the number of backorders is updated (Equation 10) and third the number of products short in time period *t* (Equation 11):

$$\mathrm{Z}_j^t = \widetilde{\mathrm{ZB}}_j^t + \widetilde{\mathrm{ZD}}_j^t \qquad \forall j \in J^n, t \in T. \tag{9A}$$

Where,

$$\widetilde{\mathrm{ZB}}_j^t = \begin{cases} \widetilde{\mathrm{B}}_j^{t-1}, & \text{if } \widetilde{\mathrm{B}}_j^{t-1} \leq \mathrm{I}_j^{t-1} + \mathrm{A}_j^t\,; \\ \mathrm{I}_j^{t-1} + \mathrm{A}_j^t, & \text{otherwise,} \end{cases} \qquad \forall j \in J^n, t \in T. \tag{9B}$$

$$\widetilde{\mathrm{ZD}}_j^t = \begin{cases} \mathrm{d}_j^t, & \text{if } \mathrm{d}_j^t \leq \mathrm{I}_j^{t-1} + \mathrm{A}_j^t - \widetilde{\mathrm{ZB}}_j^t\,; \\ \mathrm{I}_j^{t-1} + \mathrm{A}_j^t - \widetilde{\mathrm{ZB}}_j^t, & \text{otherwise,} \end{cases} \qquad \forall j \in J^n, t \in T. \tag{9C}$$

$$\widetilde{\mathrm{B}}_j^t = \widetilde{\mathrm{B}}_j^{t-1} + \mathrm{d}_j^t - \mathrm{Z}_j^t, \qquad \forall j \in J^n, t \in T. \tag{10}$$

$$\mathrm{V}_j^t = \mathrm{d}_j^t - \widetilde{\mathrm{ZD}}_j^t, \qquad \forall j \in J^n, t \in T. \tag{11}$$

4.3.3 Determining order quantities

At the end of time period *t*, the inventory levels for all locations j ∈ J will be updated, based on the inventory level of time period *t-1*, the arrival quantity at the beginning of time period *t* and the number of products shipped during time period *t*:

$$\mathrm{I}_j^t = \mathrm{I}_j^{t-1} + \mathrm{A}_j^t - \mathrm{Z}_j^t, \qquad \forall j \in J, t \in T. \tag{12}$$

To replenish its on-hand inventory, each location can place an order at the end of time period *t*. The order quantity for each of the locations is determined based on the inventory position, the reorder point *s*, and the order-up-to level *S* of the individual location.

For all locations j ∈ J, let IP_j^t be the current inventory position of location *j*, which consists of the on-hand inventory level I_j^t plus the quantity for all products that are ordered but not yet received $(\mathrm{O}_j^{t-1} - \mathrm{A}_j^t)$. In here, we calculate the number of outstanding orders of time period *t-1* minus what arrived at the beginning of time period *t*.

$$\mathrm{IP}_j^t = \mathrm{I}_j^t + \mathrm{O}_j^{t-1} - \mathrm{A}_j^t\,, \qquad \forall j \in J, t \in T. \tag{13}$$

If the inventory position of location *j* at time period *t* falls below reorder point *s,* an order will be placed equal to the difference between the order-up-to level *S* and the inventory position IP_j^t. In all other situations, the inventory position is above *s* and therefore nothing needs to be ordered:

$$Q_j^t = \begin{cases} S_j - IP_j^t, \text{ if } IP_j^t \leq s_j; \\ 0, \text{ otherwise.} \end{cases}, \qquad \forall j \in J, t \in T. \tag{14}$$

Last, after determining the order quantities the outstanding orders for time period *t* are updated by subtracting the arrival quantity and adding the order placed in this time period:

$$O_j^t = O_j^{t-1} - A_j^t + Q_j^t, \qquad \forall j \in J, t \in T. \tag{15}$$

4.3.4 Performance measurements

The simulation runs over a sufficiently large number of time periods, after which the performance can be measured to evaluate the (*s,S*) policies (solutions) obtained by the different optimization methods. In our model, fill rate requirements can be set for all locations at each echelon. For our specific problem we set these fill rate requirements only at the locations in the lowest echelon. In our computational experiments, however, we will report the actual fill rate for all locations $j \in J$ at all echelons and therefore calculate the expected shortage per cycle during the steady-state T' by dividing the shortage of that cycle (or time period) through the faced demand of that cycle. For the locations $j \in J^n$ in the lowest echelon, the faced demand is the demand from the external customer demand points and the expected shortage is given by:

$$E_j(s, S) = \frac{\sum_{t \in T'} V_j^t}{\sum_{t \in T'} d_j^t}, \qquad \forall j \in J^n. \tag{16A}$$

For all other locations $j \in J \backslash J^n$ that are not in the lowest echelon, the demand faced is equal to the orders of their child locations of time period *t-1.* The expected shortage is calculated as follows:

$$E_j(s, S) = \frac{\sum_{t \in T'} V_j^t}{\sum_{t-1 \in T'-1} \sum_{r \in R(j)} Q_r^{t-1}}, \qquad \forall j \in J \backslash J^n. \tag{16B}$$

The actual fill rate is given by 1-$E_j(s, S)$.

Additionally, the expected inventory levels and order quantities (in number of transport units) for all locations $j \in J$ have to be determined for calculating the expected total cost:

$$I_j(s, S) = \frac{\sum_{t \in T'} I_j^t}{|T'|}, \qquad \forall j \in J. \tag{17}$$

$$P_j(s, S) = \frac{\sum_{t \in T'} \left\lceil \frac{Q_j^t}{Y} \right\rceil}{|T'|}, \qquad \forall j \in J. \tag{18}$$

where Y is the number of products in one transport unit.

The expected total cost for all locations j is the sum of the holding cost per product multiplied by the expected inventory level and the order cost per transport unit multiplied by the expected number of transport units ordered:

$$\sum_{j \in J} h_j \, I_j(\mathrm{s},\mathrm{S}) \; + \sum_{j \in J} K_j \, P_j(\mathrm{s},\mathrm{S}). \tag{19}$$

5. Computational results

In this chapter we examine the performance of our two proposed solution approaches on a benchmark set derived from the synthetic problem instances of van der Heijden (2000) and on our real-life food retail case. van der Heijden (2000) provides a benchmark set for divergent 2-, 3- and 4-echelon networks, ranging from 3 up to 59 locations within the network. Two network structures consist of 2 echelons, one with 3 locations (abbreviated 2E3L) and one with 7 locations (abbreviated 2E7L). van der Heijden (2000) furthermore provides two network structures for 3-echelon networks, one with 7 locations (3E7L) and one with 15 locations (3E15L). For the 4-echelon networks, he provides four different network structures, with 15 locations (4E15L), 27 locations (4E27L), 31 locations (4E31L) and 59 locations (4E59L). We adjusted the problem instances by incorporating a fixed order cost and number of units per transport unit. Furthermore, instead of a base-stock policy, we optimize (*s,S*) policies; we follow proportional rationing instead of balanced stock rationing and use installation stock policies instead of echelon stock policies. Opposed to van der Heijden (2000), we do not use a single parameter for describing the stock level at a location as we use simulation-optimization models to determine stock levels.

The algorithms have been coded in C++ and all experiments have been performed on a QEMU Virtual CPU processor (2.50 GHz) with 4 GB RAM and a 64-bit operating system. Preliminary testing showed that the simulation model should run for at least 5000 days with a warm-up period of 200 days to achieve steady-state results. The fixed initial step size for the local search procedure of the improvement method (see Section 4.2.2) is set to twenty and every iteration the step size is increased or decreased by two units. The step size is initialized every time the local search procedure starts for refining the (*s,S*) values of a certain location.

The structure of the rest of the chapter is as follows. First, we will determine appropriate parameter values for the Scatter Search metaheuristic in Section 5.1. The Nested Bisection Search does not contain parameters and therefore does not require parameter tuning. Second, in Section 5.2 we will compare the performance of the SS and NBS on the benchmark set described above. Third, in Section 5.3 we will report computational results on our real-life case to highlight managerial conclusions.

5.1 Parameter tuning

5.1.1 Design

Before applying the proposed solution approaches, the parameters for the Scatter Search (SS) metaheuristic have to be determined (also called *off-line parameter initialization* (Talbi, 2009, p.54)) to balance solution quality and computational time. The parameters are tuned by doing computational experiments on a subset of all benchmark instances. From each of the eight different network configurations (2 of 2-echelons, 2 of 3-echelons and 4 of 4-echelons) we choose the first instance to test different combination of parameter values.

A full factorial analysis (Coy et al., 2001) is conducted on the subset of benchmark instances for the population size (50, 100), the maximum number of non-improving local search iterations for refining the solution (5, 10), the maximum number of evaluated solutions within the local search function (50, 200, 1000), the construction heuristic (Section 4.2.1) (C1, C2) and the improvement heuristic (Section 4.2.2) (I1, I2). In total this gives 48 different combinations of parameters for each of the eight network configurations. There is a computation time limit of three hours for each instance for the reference set update within the SS and we report the current best solution if the metaheuristic is not finished within the time limit.

We analyzed the sensitivity and performance of the SS with regard to the specified parameter values and per network configuration following the approach proposed by Bräysy et al. (2009). We set the highest CPU time of the network configuration to 100% for comparing the CPU time and we report the average CPU time for a given parameter value as a percentage of the highest CPU time. For the solution quality, we compare the average expected total cost of a given parameter value to the lowest expected total cost found for the network configuration under consideration and report the difference. The averages and percentages for each parameter value per network configuration are reported in Appendix A.

5.1.2 Results

The best options (lowest values) in terms of solution quality and CPU time are indicated in Tables A.1-A.4 in Appendix A. Based on the results, we chose a population size of 50 initial solutions for all network configurations. In terms of CPU time, a population size of 50 is faster than a population size of 100 for 7 of the 8 network structures. For half of the networks this population size also gives the best solution quality and for the remaining networks using a population size of 50 or 100 only results in at most 1% cost difference. Therefore, we chose a population size of 50 initial solutions for all network configurations. The maximum number of non-improving iterations in the local search is set to 5 as this gives a shorter CPU time than 10 non-improving iterations for 7 of the 8 network structures. A maximum of 5 also gives a better solution quality than 10 for 3 network structures and for the others, the solution quality for 5 and 10 non-improving iterations is almost equal with a maximum difference of 2%. The maximum number of solutions to evaluate per local search has been set to 200 for all network structures, except for the largest network with four echelons and 59 locations. For the largest network, the SS approach performs best in terms of solution quality and CPU time with a maximum of 50 evaluated solutions. This is due to the fact that the maximum computational run time of the reference set update has been set to three hours. Therefore, it is more favorable to do more reference set updates with a smaller number of improvement iterations (50) instead of one reference set update with many improvement iterations (200 or 1000). For the other network structures, we considered 200 improvement iterations to offer the best compromise between solution quality and CPU time. Last, we compared the combinations of the solution construction (C1/C2) and improvement heuristics (I1/I2). As the CPU time of I2 is higher than I1 for 7 of the 8 network structures higher, with differences up to 20% for the larger networks, we decided to use I1 as improvement heuristic. 5 of the 8 network structures favor C2 in terms of solution quality. Therefore, we chose C2-I1 for all problem instances under consideration.

5.2 Benchmark testing

5.2.1 Introduction benchmark set

As discussed in the introductory text of Section 5, we use the synthetic problem instances of van der Heijden (2000) to construct a benchmark set. Details on the parameter settings for the 2-, 3- and 4-echelon networks can be found in Table B.1-B.3 of Appendix B. We have added the order cost per transport unit to the original benchmark set and in the simulation model the number of units ordered is rounded up to the next integer number of transportation units (one transport unit equals 100 products). In addition, the benchmark set contains different values for the fill rate requirement, lead times, holding cost and demand. We define echelon 1 as the most upstream echelon, which solely exist of root location 1 supplied by an external supplier. The echelon with the highest number is the most downstream echelon consisting of the customer locations facing external customer demand, which follows a normal distribution.

The entire benchmark set of instances (1280) consists of 128 instances for 2-echelon networks (2 structures), 384 for 3-echelon networks (2 structures) and 64 for 4-echelon networks (4 structures). As the Nested Bisection Search (NBS) is not capable of handling network structures with more than two echelons, we will compare the performance of the NBS and the proposed Scatter Search (SS) only on the 2-echelon networks (see Section 5.2.2). Afterwards, we will examine the performance of the SS on the 3- and 4-echelon networks in Section 5.2.3.

5.2.2 Comparison NBS and SS

The computational results for the NBS and SS are reported in Table 1, specifying the expected average total cost per day and CPU times for each network structure and parameter value. In this section, we focus on the 2-echelon network structures consisting of 3 or 7 locations (2E3L and 2E7L) as the NBS by Li et al. (2010) was designed to handle 2-echelon networks and cannot be straightforwardly extended to handle more than two echelons efficiently.

In terms of solution quality, we can conclude that for 95% of the problem instances the SS is performing better than the NBS. For 2E3L networks SS is performing better on all instances and for 2E7L SS is performing better for 115 of the 128 instances. However, the required CPU time is much lower for the NBS for both network structures. For 2E3L networks the average CPU time for NBS is 0.25 seconds and for the SS 2.5 minutes. For 2E7L networks the NBS has an average CPU time of 1.4 seconds and the SS 10.4 minutes. Over all the 2-echelon experiments, the NBS leads on average to 7.7% more expensive solutions compared our implementation of the SS, with the largest difference being 27.4%.

	Network	**2E3L**				**2E7L**			
Parameter		**NBS[a]**	**SS[a]**	**% Diff[b]**	**# Diff[c]**	**NBS[a]**	**SS[a]**	**% Diff[b]**	**# Diff[c]**
Fill rate requirement	**90%**	€ 145.28	€ 131.69	10%	64/64	€ 391.30	€ 375.91	4%	58/64
	99%	€ 199.19	€ 179.78	11%	64/64	€ 542.62	€ 519.62	4%	57/64
Lead time echelon 1 [days]	**1**	€ 167.85	€ 151.63	11%	64/64	€ 457.09	€ 443.42	3%	56/64
	3	€ 176.62	€ 159.83	11%	64/64	€ 476.82	€ 452.11	5%	59/64
Holding cost echelon 1 [Euro]	**€ 0.25**	€ 165.18	€ 149.44	11%	32/32	€ 461.23	€ 434.76	6%	31/32
	€ 0.50	€ 170.26	€ 154.64	10%	32/32	€ 467.45	€ 446.24	5%	30/32
	€ 0.75	€ 175.17	€ 158.14	11%	32/32	€ 469.91	€ 454.02	4%	26/32
	€ 1.00	€ 178.33	€ 160.72	11%	32/32	€ 469.23	€ 456.05	3%	28/32
Order cost [Euro]	**€ 25**	€ 119.70	€ 111.38	7%	64/64	€ 321.02	€ 308.10	4%	54/64
	€ 100	€ 224.78	€ 200.08	12%	64/64	€ 612.90	€ 587.43	4%	61/64
Demand [mean (standard deviation)]	**10(4)**	€ 94.85	€ 83.39	14%	32/32	€ 257.47	€ 248.03	4%	30/32
	10(8)	€ 116.53	€ 103.66	12%	32/32	€ 323.77	€ 307.11	5%	29/32
	30(12)	€ 202.70	€ 181.17	12%	32/32	€ 547.88	€ 519.57	5%	29/32
	30(24)	€ 274.86	€ 254.71	8%	32/32	€ 738.72	€ 716.35	3%	27/32

Table 1: Comparison NBS and SS on 2E3L and 2E7L networks

Note: a. In these columns we report the expected total cost per day for the solutions obtained by the NBS or SS. The cost is the average over all instances for a parameter

b. The columns '% Diff' report the percentage the NBS is, on average, more expensive than the SS.

c. The columns '# Diff' report the number of times the NBS is more expensive than the SS.

5.2.3 SS results for 3- and 4-echelon networks

In this section, we will examine the performance of the proposed SS metaheuristic on the 3- and 4-echelon instances from van der Heijden (2000). The computational results for the 3-echelon instances (3E7L and 3E15L) are reported in Table 2 and for the 4-echelon instances (4E15L, 4E27L, 4E31L and

4E59L) in Table 3. The results show that the SS metaheuristic is capable of solving small and large network problems and provides a feasible solution to all instances within acceptable CPU times.

The computational results show that the network structures of the benchmark instances can significantly impact the CPU time. First of all, the average CPU time increases with the number of locations in the network, as more (*s,S*) values need to be improved and evaluated. In addition, more locations per echelon lead to higher upper limits for the parent location supplying these locations. In particular, the upper limit for a location is set to twenty times the average demand (of a time period) faced by that location (see Section 4.1). As a result, the search space is larger in a situation with more child locations per parent location, which can increase the CPU time. Comparing the results of the SS metaheuristic in networks of 15 locations with different network structures (3E15L vs. 4E15L), shows that the average CPU time of the 3-echelon case is longer than the 4-echelon case (1954.44 seconds compared to 1684.24 seconds).Additionally, the 4-echelon case has on average lower total cost per day, but this is due to differences in the parameter values in the benchmark set proposed by van der Heijden (2000) for the 3- and 4-echelon cases, such as the lead time and holding cost. Furthermore, it can be noticed that the differences in the average total cost per day is small for varying lead times and holding cost for a specific echelon. For example, if the lead time for echelon 1 for the 3E7L network is shortened from 3 to 1 days, the difference in cost is only 1.4% (€386.64 for 1 day and €392.05 for 3 days lead time). This may be due to the fact that inventory is placed in another echelon when for example the holding cost increase for a specific echelon.

Moreover, due to rounding the order quantities up to an integer number of transport units for determining the order cost, an increase in holding cost may lead to slightly lower average total cost per

	Network	**3E7L**	**3E15L**
Parameter		**SS**[a]	**SS**[a]
Fill rate requirement	**90%**	€ 337.76	€ 978.51
	99%	€ 440.94	€ 1 304.55
Lead time echelon 1 [days]	**1**	€ 386.64	€ 1 149.57
	3	€ 392.05	€ 1 133.49
Lead time echelon 2 [days]	**1**	€ 386.23	€ 1 139.60
	2	€ 392.47	€ 1 143.46
Holding cost echelon 1 [Euro]	**€ 0.25**	€ 388.05	€ 1 142.23
	€ 0.50	€ 390.65	€ 1 140.84
Holding cost echelon 2 [Euro]	**€ 0.25**	€ 375.68	€ 1 108.75
	€ 0.50	€ 390.44	€ 1 144.98
	€ 1.00	€ 401.92	€ 1 170.86
Order cost [Euro]	**€ 25**	€ 263.40	€ 766.47
	€ 100	€ 515.30	€ 1 516.60
Demand [mean (standard deviation)]	**10(4)**	€ 201.36	€ 620.41
	10(8)	€ 248.96	€ 760.53
	30(12)	€ 474.70	€ 1 385.78
	30(24)	€ 632.37	€ 1 799.41
Average expected total cost per day [Euro][b]		€ 389.35	€ 1 141.53
Average CPU time [seconds][c]		648.14 s	1954.44 s

Table 2: Results 3-echelon instances

Note: a. In these columns we report the expected total cost per day for the solutions obtained by the SS approach. The cost is the average over all instances for a parameter.

b. The average expected total cost per day is reported over all instances for the specified network.

day. For example, for 3E15L the average total cost changes from €1142.23 to €1140.84 when changing the holding cost of echelon 1 (see Table 2).

	Network	**4E15L**	**4E27L**	**4E31L**	**4E59L**
Parameter		**SS**[a]	**SS**[a]	**SS**[a]	**SS**[a]
Fill rate requirement	**90%**	€ 861.56	€ 1 787.66	€ 2 524.30	€ 5 230.26
	99%	€ 1 092.30	€ 2 293.75	€ 3 200.73	€ 7 065.36
Holding cost echelon 2 [Euro]	**€ 0.25**	€ 977.03	€ 2 054.58	€ 2 860.51	€ 6 157.25
	€ 0.50	€ 976.84	€ 2 026.84	€ 2 864.52	€ 6 138.37
Holding cost echelon 3 [Euro]	**€ 0.50**	€ 953.66	€ 1 516.22	€ 2 778.08	€ 6 100.07
	€ 1.00	€ 1 000.21	€ 2 046.62	€ 2 946.95	€ 6 195.55
Order cost [Euro]	**€ 25**	€ 632.27	€ 1 389.25	€ 1 939.87	€ 4 330.31
	€ 100	€ 1 321.59	€ 2 085.08	€ 3 785.16	€ 7 965.30
Demand [mean (standard deviation)]	**10(4)**	€ 500.09	€ 1 031.06	€ 1 527.25	€ 3 161.37
	10(8)	€ 604.16	€ 1 329.15	€ 1 812.20	€ 3 896.84
	30(12)	€ 1 231.99	€ 2 526.98	€ 3 585.26	€ 7 676.44
	30(24)	€ 1 571.49	€ 3 275.64	€ 4 525.35	€ 9 856.60
Average expected total cost per day [Euro][b]		€ 976.93	€ 2 040.71	€ 2 862.51	€ 6 147.81
Average CPU time [seconds][c]		1684.24 s	3936.58 s	6794.35 s	7428.08 s

Table 3: Results 4-echelon instances

Note: a. In these columns we report the expected total cost per day for the solutions obtained by the SS approach. The cost is the average over all instances for a parameter.

b. The average expected total cost per day is reported over all instances for the specified network.

c. The average CPU time is reported over all instances for the specified network.

5.3 Real-life case

5.3.1 Introduction

To examine the managerial implications of optimizing (*s,S*) values in a real-life setting, we use the proposed Scatter Search (SS) metaheuristic to find (near-) optimal inventory policies as this approach outperforms the Nested Bisection Search (NBS) on the benchmark set. We examine the case of a Dutch food retailer that supplies products from one central warehouse (WH) to four retailer distribution centers (DC1-4) and that aims to determine (*s,S*) values that yield a 98% fill rate to the external customer demand points while minimizing supply chain cost. The lead time between the external supplier and the warehouse is twelve days; and between the warehouse and the retailer DCs it is one day. The holding cost for the warehouse is €0.04 per product per day and for the retailer it is €0.05 per product per day. The warehouse pays order costs of €76 per pallet ordered while the retailers pay €14. The number of products ordered is rounded up to the next integer number of pallets, which consists of 256 products (boxes). Historical demand is available and fitted to probability distributions to simulate external customer demand (i.e., demand from the supermarket stores to the retail distribution centers). For DC1 the demand follows a Gamma distribution with α=4.234 and β=11.877, DC2 faces Weibull distributed demand with α=3.5332 and β=22.972, the demand of DC3 follows a Lognormal distribution with μ=3.4837 and σ=0.54546 and for DC4 the demand follows a Gamma distribution with α=4.5459 and β=2.8157. In Section 5.3.2, we report the results based on the data set described. In Section 5.3.3, we report the results of a sensitivity analysis based on cost estimates of practitioners to draw managerial conclusions.

5.3.2 Results

The results of the real-life case of the food retailer are presented in Table 4. The (*s,S*) inventory policies for all locations are reported, including the accompanying fill rates per location, the expected total cost per day for the complete network (€77.98) and the total CPU time of approximately 8.5 minutes (518.11 seconds) to solve this case. A fill rate requirement of 98% is only imposed on the retailer distribution centers. Table 4 illustrates that it is not needed to have a similar fill rate at the central warehouse in order to reach the target at the downstream retail distribution centers. Contrary to current company practices, a central warehouse (WH) fill rate of 74.87% is sufficient to achieve fill rates of 98% or higher at the retail DCs. As such, inventory and therefore supply chain costs can be saved by lowering fill rate requirements for the distribution from a central warehouse to the locations that supply the external customer demand points while guaranteeing the targeted product availability for the final customers.

DC	*s*	*S*	*Fill rate*
WH	1425	1820	74.87%
DC1	152	324	98.01%
DC2	48	151	98.04%
DC3	130	268	98.03%
DC4	29	124	98.01%
Expected total cost per day [Euro]			€ 77.98
CPU time [seconds]			518.11 s

Table 4: Results real-life case

5.3.3 Sensitivity analysis

In this section, we examine the managerial implications of optimizing (*s,S*) values in real-life settings for different parameters, resulting in 192 instances. The chosen values for the parameters in the sensitivity analysis are based on alternative data values provided by the Dutch food retailer. We conduct this analysis using the SS approach for different fill rate requirements (90%, 95%, and 98%), different lead times between the central warehouse and retail distribution centers (1, 2, 3, and 4 days), inventory holding costs (1, 2, 3, and 4 times the given holding cost), and order costs (0.5, 1, 1.5, and 2 times the given order cost).

Parameter	***Value***	***Average total expected cost per day***[a]	***Average Fill rate WH***[b]
Fill rate requirement	**90%**	€ 105.44	48.4%
	95%	€ 115.81	58.6%
	98%	€ 128.66	73.0%
Lead time [days]	**1**	€ 113.26	62.1%
	2	€ 115.39	56.9%
	3	€ 117.79	59.7%
	4	€ 120.11	61.3%
Holding cost [Euro]	**X 1**	€ 86.80	66.1%
	X 2	€ 108.05	61.3%
	X 3	€ 126.80	58.7%
	X 4	€ 144.91	54.0%
Order cost [Euro]	**X 0.5**	€ 72.63	53.9%
	X 1	€ 103.59	58.7%
	X 1.5	€ 131.88	62.9%
	X 2	€ 158.45	64.5%
Average total expected cost per day [Euro][c]		€ 116.64	
Average CPU time [seconds][d]		401.25 s	
Average Fill rate WH[e]		60.0%	

Table 5: Results sensitivity analysis real-life case

Note: a. a. In this column we report the expected total cost per day for the solutions obtained by the SS approach. The cost is the average over all instances for a parameter.

b. In this column we report the average fill rate of the central warehouse (WH) over all instances for one specified parameter.

c. The average expected total cost per day is reported over all instances.

d. The average CPU time is reported over all instances.

e. The average fill rate of the central warehouse (WH) is reported over all instances.

The results of the sensitivity analysis are presented in Table 5. It can be noticed that although the fill rate of the WH can be lower than the fill rate requirement, the WH fill rate increases on average when the fill rate requirement of the DCs increases as well. This means that the inventory level of the WH must be higher to guarantee higher fill rates at the DCs.

The results of the real-life case confirm the findings from the benchmark set that changes in holding cost at a given echelon lead to repositioning of inventory to other echelons. In particular, when increasing the holding cost at the central warehouse, we observed that optimal fill rates at the central warehouse are decreased (see Table 5).

6. Conclusions & further research

While inventory management is a widely studied topic in academia, it is still challenging due to constraints encountered in practice. Inspired by a real-life case this paper aimed to develop a simulation-optimization approach to optimize the (*s,S*) inventory policies of a multi-echelon distribution network with deterministic lead times, backordering, and fill rate constraints. From a practitioner's point of view, this is a common and important problem to address and to the best of our knowledge only two papers discuss models which fit such a setting well: Schneider et al. (1995) and Li et al. (2010). Both derive near-optimal inventory policies based on stochastic lead times via either power approximations or simulation-optimization. We propose a simulation-optimization approach with two different optimization methods: a Nested Bisection Search (NBS) based on Li et al. (2010) and a tailor-made Scatter Search (SS) metaheuristic.

The optimization of the two methods works iteratively with a simulation model to evaluate the obtained solutions in a simulation-optimization framework. To examine the performance of the two solution approaches, we test them on 1280 synthetic instances of van der Heijden (2000). Before running the experiments on the test instances, the parameters of the SS metaheuristic are determined by trading-off solution quality and CPU time for different parameter values. We use the 2-echelon instances to compare the NBS and SS solution approaches, as the existing NBS method is only capable of solving 2-echelon problems. We also report the performance of the SS metaheuristic on the 3- and 4-echelon instances to examine its performance on larger supply chain networks.

Computational results show that the proposed SS metaheuristic outperforms the existing NBS method in 95% of the instances. The NBS is a fast solution method, which optimizes the (*s,S*) policies for each location sequentially and considers a fixed difference between the *s* and *S* values. The SS metaheuristic provides a diverse set of solutions and refines the solutions with a local search without additional restrictions on the (*s,S*) values leading to better solution quality. The SS has on average 7.7% lower cost compared to the NBS, with savings up to 27.4%. The results of the experiments on the 3- and 4-echelon instances show that the SS method is also capable of solving larger-sized problems efficiently. It should be noted that even better results could be obtained at the expense of a higher CPU time.

To examine the managerial implications of finding (near-) optimal (*s,S*) inventory policies in real-life settings, we apply the SS metaheuristic to the case of a Dutch food retailer. First, we examine the actual parameter values and second, we conduct a sensitivity analysis based on alternative data values provided by the Dutch food retailer. From the computational results we can conclude that it is not necessary to impose the same fill rate requirement on all locations in the supply chain. Inventory and supply chain costs can be saved by allowing lower fill rates at upstream echelons than the downstream locations that experience external customer demand, contrary to current company practices.

The proposed Scatter Search simulation-optimization approach can help in making tactical decisions with regard to determining (near-) optimal inventory policies to save supply chain costs in a variety of supply chains. We showed that our model that is capable of accurately modeling real-life constraints, such as backordering, a variety of demand distributions and lead times, for small (3 locations) and large-sized (59 locations) networks. The approach has shown potential for tackling other practical constraints that are regularly discussed in the literature but are difficult to optimize (e.g. capacity constraints and multiple commodity problems) in future research.

Appendix A – Results parameter tuning

		Population size			
		Solution quality (€)		CPU time (s)	
Network		**50**	**100**	**50**	**100**
2E3L	**Average[a]**	43.92	43.87	119.92	156.36
	%[b]	1.15%	1.02%	34.93%	45.54%
2E7L	**Average**	133.31	132.09	646.00	781.75
	%	8.09%	7.10%	31.71%	38.38%
3E7L	**Average**	108.35	108.82	841.59	947.56
	%	4.90%	5.35%	31.20%	35.13%
3E15L	**Average**	326.97	325.74	2659.70	2909.69
	%	8.31%	7.91%	39.98%	43.74%
4E15L	**Average**	267.37	264.72	2844.62	3817.42
	%	7.17%	6.11%	29.31%	39.33%
4E27L	**Average**	559.20	563.56	6522.28	7036.13
	%	17.72%	18.64%	49.45%	53.34%
4E31L	**Average**	830.46	859.62	8177.01	8373.76
	%	19.53%	23.73%	59.99%	61.43%
4E59L	**Average**	2260.66	2793.31	12323.18	11683.36
	%	45.64%	79.95%	60.75%	57.60%

Table A.1: Population size

Note: The highlighted cells give the best option in terms of either solution quality or CPU time.

a. The average value reports the average of the total expected cost per day in Euros (for the solution quality) or the average CPU time in seconds (for the CPU time), specified per network and for population sizes 50 or 100.

b. For the solution quality the percentage shows the difference of the average compared to the lowest cost solution. For the CPU time the percentage for the average compared to the highest CPU time is reported.

		Maximum number of Non-improving iterations			
		Solution quality (€)		**CPU time (s)**	
Network		**5**	**10**	**5**	**10**
2E3L	**Average**[a]	43.89	43.90	124.28	152.00
	%[b]	1.07%	1.10%	36.19%	44.27%
2E7L	**Average**	133.39	132.00	658.12	769.63
	%	8.16%	7.03%	32.31%	37.78%
3E7L	**Average**	108.82	108.35	836.63	952.52
	%	5.35%	4.90%	31.02%	35.31%
3E15L	**Average**	328.41	324.30	2740.18	2829.22
	%	8.79%	7.43%	41.19%	42.53%
4E15L	**Average**	265.31	266.78	3073.78	3588.26
	%	6.34%	6.93%	31.67%	36.97%
4E27L	**Average**	562.47	560.29	6775.60	6782.81
	%	18.41%	17.95%	51.37%	51.42%
4E31L	**Average**	845.26	844.82	8219.00	8331.77
	%	21.66%	21.60%	60.30%	61.13%
4E59L	**Average**	2449.03	2604.95	12023.05	11983.49
	%	57.77%	67.82%	59.27%	59.08%

Table A.2: The maximum number of non-improving iterations in the local search

Note: The highlighted cells give the best option in terms of either solution quality or CPU time.

a. The average value reports the average of the total expected cost per day in Euros (for the solution quality) or the average CPU time in seconds (for the CPU time), specified per network and for the maximum number of non-improving iterations (5 or 10).

b. For the solution quality the percentage shows the difference of the average compared to the lowest cost solution. For the CPU time the percentage for the average compared to the highest CPU time is reported.

		Maximum number of solutions to evaluate					
		Solution quality (€)			**CPU time (s)**		
Network		**50**	**200**	**1000**	**50**	**200**	**1000**
2E3N	**Average**[a]	44.20	43.84	43.65	65.88	143.65	204.89
	%[b]	1.78%	0.96%	0.51%	19.19%	41.84%	59.67%
2E7N	**Average**	134.74	134.43	128.93	328.07	638.57	1174.98
	%	9.25%	9.00%	4.54%	16.11%	31.35%	57.68%
3E7N	**Average**	111.19	107.91	106.65	308.27	721.34	1654.12
	%	7.65%	4.47%	3.25%	11.43%	26.74%	61.32%
3E15N	**Average**	340.17	324.97	313.93	1058.86	2627.72	4667.51
	%	12.69%	7.65%	3.99%	15.92%	39.50%	70.16%
4E15N	**Average**	276.31	263.83	258.01	905.49	2386.76	6700.81
	%	10.75%	5.75%	3.41%	9.33%	24.59%	69.04%
4E27N	**Average**	566.00	552.64	565.50	2266.31	6722.29	11349.02
	%	19.15%	16.34%	19.05%	17.18%	50.96%	86.04%
4E31N	**Average**	853.05	840.56	841.51	3423.82	9101.23	12301.12
	%	22.79%	20.99%	21.12%	25.12%	66.77%	90.25%
4E59N	**Average**	1892.27	2165.86	3522.84	8842.71	11533.11	15633.99
	%	21.90%	39.53%	126.95%	43.59%	56.86%	77.07%

Table A.3: Maximum number of solution to evaluate

Note: The highlighted cells give the best option in terms of either solution quality or CPU time.

a. The average value reports the average of the total expected cost per day in Euros (for the solution quality) or the average CPU time in seconds (for the CPU time), specified per network and for the maximum number of solutions to evaluate (50, 200 or 1000).

b. For the solution quality the percentage shows the difference of the average compared to the lowest cost solution. For the CPU time the percentage for the average compared to the highest CPU time is reported.

		Construction and Improvement Heuristics							
		Solution quality (€)				CPU time (s)			
Network		C1I1[c]	C2I1	C2I2	C1I2	C1I1	C2I1	C2I2	C1I2
2E3L	**Average[a]**	43.78	43.66	44.08	44.05	185.78	134.25	110.02	122.50
	%[b]	0.82%	0.55%	1.51%	1.45%	54.11%	39.10%	32.04%	35.68%
2E7L	**Average**	129.45	129.24	135.88	136.22	673.14	729.80	744.26	708.31
	%	4.96%	4.79%	10.18%	10.45%	33.05%	35.83%	36.54%	34.77%
3E7L	**Average**	110.10	108.32	107.61	108.32	888.75	752.78	1095.66	841.11
	%	6.59%	4.87%	4.18%	4.87%	32.95%	27.91%	40.62%	31.18%
3E15L	**Average**	322.50	319.72	330.40	332.81	2105.24	2141.81	3484.69	3407.05
	%	6.83%	5.91%	9.45%	10.25%	31.64%	32.19%	52.38%	51.21%
4E15L	**Average**	265.06	269.65	265.87	263.61	2874.05	2904.23	3548.21	3997.59
	%	6.24%	8.08%	6.56%	5.66%	29.61%	29.92%	36.56%	41.19%
4E27L	**Average**	530.83	554.18	572.95	587.57	5304.17	5321.88	7905.88	8584.89
	%	11.75%	16.66%	20.61%	23.69%	40.21%	40.35%	59.94%	65.08%
4E31L	**Average**	778.08	780.15	869.03	952.90	7065.50	6961.91	9656.61	9417.53
	%	11.99%	12.29%	25.09%	37.16%	51.84%	51.08%	70.85%	69.09%
4E59L	**Average**	2570.81	2535.55	2413.92	2587.67	10097.62	10104.64	13607.92	14202.90
	%	65.62%	63.35%	55.51%	66.70%	49.78%	49.81%	67.08%	70.02%

Table A.4: Construction and Improvement Heuristics
Note: The highlighted cells give the best option in terms of either solution quality or CPU time.
a. The average value reports the average of the total expected cost per day in Euros (for the solution quality) or the average CPU time in seconds (for the CPU time), specified per network and for the combination of construction and improvement heuristics (C1I1, C2I1, C2I2 or C1I2).
b. For the solution quality the percentage shows the difference of the average compared to the lowest cost solution. For the CPU time the percentage for the average compared to the highest CPU time is reported.
c. C1/2 is Construction Heuristic 1 or 2; I1/2 is Improvement Heuristic 1 or 2.

Appendix B –Parameter values of the benchmark set

Parameter	Description	Values in experiments
J^2	Number of customer locations in echelon 2 for each location in echelon 1	2, 6
$F_j, \quad \forall\, j \in J^2$	Fill rate requirement at customer locations *j* at echelon 2	90%, 99%
$h_j, \quad \forall\, j \in J^1$	Holding cost per unit per time period for location *j* at echelon 1	€0.25, €0.50, €0.75, €1.00
$h_j, \quad \forall\, j \in J^2$	Holding cost per unit per time period for locations *j* at echelon 2	€1.00
$K_j, \quad \forall\, j \in J^v$	Order cost per transportation unit for locations *j*	€25, €100
$L_j, \quad \forall\, j \in J^1$	Lead time for location *j* in echelon 1	1, 3
$L_j, \quad \forall\, j \in J^2$	Lead time for locations *j* in echelon 2	1
$d_j^t, \quad \forall\, j \in J^2$	Demand faced by customer locations *j* at echelon 2 with mean (standard deviation)	10 (4, 8), 30 (12, 24)

Table B.1: Parameter values for 2-echelon networks

Parameter	Description	Values in experiments
J^2	Number of child locations in echelon 2 for each location in echelon 1	2
J^3	Number of customer locations in echelon 3 for each location in echelon 2	2, 6
$F_j, \quad \forall\, j \in J^3$	Fill rate requirement at customer locations *j* at echelon 3	90%, 99%
$h_j, \quad \forall\, j \in J^1$	Holding cost per unit per time period for location *j* at echelon 1	€0.25, €0.50
$h_j, \quad \forall\, j \in J^2$	Holding cost per unit per time period for locations *j* at echelon 2	€0.25, €0.50, €1.00
$h_j, \quad \forall\, j \in J^3$	Holding cost per unit per time period for locations *j* at echelon 3	€1.00
$K_j, \quad \forall\, j \in J^v$	Order cost per transportation unit for locations *j*	€25, €100
$L_j, \quad \forall\, j \in J^1$	Lead time for location *j* in echelon 1	1, 3
$L_j, \quad \forall\, j \in J^2$	Lead time for locations *j* in echelon 2	1, 2
$L_j, \quad \forall\, j \in J^3$	Lead time for locations *j* in echelon 3	1
$d_j^t, \quad \forall\, j \in J^3$	Demand faced by customer locations *j* at echelon 3 with mean (standard deviation)	10 (4, 8), 30 (12, 24)

Table B.2: Parameter values for 3-echelon networks

Parameter	Description	Values in experiment
J^2	Number of child locations in echelon 2 for each location in echelon 1	2
J^3	Number of child locations in echelon 3 for each location in echelon 2	2, 4
J^4	Number of customer locations in echelon 4 for each location in echelon 3	2, 6
$F_j, \quad \forall\, j \in J^4$	Fill rate requirement at customer locations *j* at echelon 4	90%, 99%
$h_j, \quad \forall\, j \in J^1$	Holding cost per unit per time period for location *j* at echelon 1	€0.25
$h_j, \quad \forall\, j \in J^2$	Holding cost per unit per time period for locations *j* at echelon 2	€0.25, €0.50
$h_j, \quad \forall\, j \in J^3$	Holding cost per unit per time period for locations *j* at echelon 3	€0.50, €1.00
$h_j, \quad \forall\, j \in J^4$	Holding cost per unit per time period for locations *j* at echelon 4	€1.00
$K_j, \quad \forall\, j \in J^v$	Order cost per transportation unit for locations *j*	€25, €100
$L_j, \quad \forall\, j \in J^v$	Lead time for locations *j*	1
$d_j^t, \quad \forall\, j \in J^4$	Demand faced by customer locations *j* at echelon 4 with mean (standard deviation)	10 (4 or 8), 30 (12 or 24)

Table B.3: Parameter values for 4-echelon networks

References

ALMEDER, C., PREUSSER, M. & HARTL, R. F. 2008. Simulation and optimization of supply chains: alternative or complementary approaches? *OR Spectrum,* 31**,** 95-119.

AMARAN, S., SAHINIDIS, N. V., SHARDA, B. & BURY, S. J. 2016. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research,* 240**,** 351-380.

APRIL, J., GLOVER, F., KELLY, J. & LAGUNA, M. 2003. Practical introduction to simulation optimization *Winter Simulation Conference***,** 71-78.

AVIV, Y. & FEDERGRUEN, A. 2001. Design for postponement: A comprehensive characterization of its benefits under unknown demand distributions. *Operations Research,* 49**,** 578-598.

AXSÄTER, S. 1990. Simple solution procedures for a class of two-echelon inventory problems. *Operations Research,* 38**,** 64-69.

AXSÄTER, S. 2003a. Approximate optimization of a two-level distribution inventory system. *International Journal of Production Economics,* 81**,** 545-553.

AXSÄTER, S. 2003b. Note: Optimal Policies for Serial Inventory Systems Under Fill Rate Constraints. *Management Science,* 49**,** 247-253.

AXSÄTER, S. 2003c. Supply Chain Operations: Serial and Distribution Inventory Systems. 11**,** 525-559.

AXSÄTER, S. & ROSLING, K. 1993. Notes: Installation vs. echelon stock policies for multilevel inventory control. *Management Science,* 39**,** 1274-1280.

BASHYAM, S. & FU, M. C. 1998. Optimization of (s, S) Inventory Systems with Random Lead Times and a Service Level Constraint. *Management Science,* 44**,** S243-S256.

BENKHEROUF, L. 1995. On an inventory model with deteriorating items and decreasing time-varying demand and shortages. *European Journal of Operational Research,* 86**,** 293-299.

BOYACI, T. & GALLEGO, G. 2001. Serial Production/Distribution Systems Under Service Constraints. *Manufacturing & Service Operations Management,* 3**,** 43-50.

BRÄYSY, O., PORKKA, P. P., DULLAERT, W., REPOUSSIS, P. P. & TARANTILIS, C. D. 2009. A well-scalable metaheuristic for the fleet size and mix vehicle routing problem with time windows. *Expert Systems with Applications,* 36**,** 8460-8475.

CAGGIANO, K. E., JACKSON, P. L., MUCKSTADT, J. A. & RAPPOLD, J. A. 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**,** 744-749.

CHEN, F. & KRASS, D. 2001. Inventory models with minimal service level constraints. *European Journal of Operational Research,* 134**,** 120-140.

CHEN, F. & ZHENG, Y.-S. 1994. Lower bounds for multi-echelon stochastic inventory systems. *Management Science,* 40**,** 1426-1443.

CHENG, F., ETTL, M., LIN, G. & YAO, D. D. 2002. Inventory-service optimization in configure-to-order systems. *Manufacturing & Service Operations Management,* 4**,** 114-132.

CHU, Y., YOU, F., WASSICK, J. M. & AGARWAL, A. 2015. Simulation-based optimization framework for multi-echelon inventory systems under uncertainty. *Computers & Chemical Engineering,* 73**,** 1-16.

CLARK, A. & SCARF, H. 1960. Optimal policies for a multi-echelon inventory problem. *Management Science,* 6**,** 475-490.

COHEN, M. A., KLEINDORFER, P. R. & LEE, H. L. 1989. Near-optimal service constrained stocking policies for spare parts. *Operations Research,* 37**,** 104-117.

COY, S. P., GOLDEN, B. L., RUNGER, G. C. & WASIL, E. A. 2001. Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics,* 7**,** 77-97.

DESMET, B., AGHEZZAF, E. H. & VANMAELE, H. 2010. A normal approximation model for safety stock optimization in a two-echelon distribution system. *Journal of the Operational Research Society,* 61**,** 156-163.

DIKS, E., DE KOK, A. & LAGODIMOS, A. 1996. Multi-echelon systems: A service measure perspective. *European Journal of Operational Research,* 95**,** 241-263.

ELTAWIL, A. & ELNAHAR, G. 2007. Simulation optimization of an (s,S) inventory control system with random demand sizes, demand arrivals, and lead times. *37th International Conference on Computers and Industrial Engineering*, 2420-2432.

FEDERGRUEN, A. & ZIPKIN, P. 1984. Approximations of dynamic, multilocation production and inventory problems. *Management Science,* 30**,** 69-84.

FIGUEIRA, G. & ALMADA-LOBO, B. 2014. Hybrid simulation–optimization methods: A taxonomy and discussion. *Simulation Modelling Practice and Theory,* 46**,** 118-134.

FLEISCHHACKER, A., NINH, A. & ZHAO, Y. 2015. Positioning Inventory in Clinical Trial Supply Chains. *Production and Operations Management,* 24**,** 991-1011.

FU, M., GLOVER, F. & APRIL, J. 2005. Simulation optimization: a review, new developments, and applications. *Winters Simulation Conference*.

FU, M. C. 2002. Feature Article: Optimization for simulation: Theory vs. Practice. *INFORMS Journal on Computing,* 14**,** 192-215.

GALLEGO, G. & ÖZER, Ö. 2005. A new algorithm and a new heuristic for serial supply systems. *Operations Research Letters,* 33**,** 349-362.

GALLEGO, G., ÖZER, Ö. & ZIPKIN, P. 2007. Bounds, heuristics, and approximations for distribution systems. *Operations Research,* 55**,** 503-517.

GALLEGO, G. & ZIPKIN, P. 1999. Stock positioning and performance estimation in serial production-transportation systems. *Manufacturing & Service Operations Management,* 1**,** 77-88.

GLASSERMAN, P. & TAYUR, S. 1996. A simple approximation for a multistage capacitated production-inventory system. *Naval Research Logistics (NRL),* 43**,** 41-58.

GLOVER, F. 1977. Heuristics for integer programming using surrogate constraints. *Decision Sciences,* 8**,** 156-166.

GLOVER, F. 2006. What's all the commotion about scatter search and tabu search anyway? *In:* REGO, C. & ALIDAEE, B. (eds.) *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search.* Springer US.

GLOVER, F., KELLY, J. & LAGUNA, M. 1999. New advances for wedding optimization and simulation. *Winter Simulation Conference***,** 255-260.

GLOVER, F., LAGUNA, M. & MARTÍ, R. 2003. Scatter search. *Advances in evolutionary computing.* Springer.

GOH, S. L., KENDALL, G. & SABAR, N. R. 2017. Improved local search approaches to solve the post enrolment course timetabling problem. *European Journal of Operational Research*.

GRAVES, S. C. 1985. A multi-echelon inventory model for a repairable item with one-for-one replenishment. *Management science,* 31**,** 1247-1256.

GRAVES, S. C. 1996. A multiechelon inventory model with fixed replenishment intervals. *Management Science,* 42**,** 1-18.

GRAVES, S. C. & WILLEMS, S. P. 2000. Optimizing Strategic Safety Stock Placement in Supply Chains. *Manufacturing & Service Operations Management,* 2**,** 68-83.

JALALI, H. & NIEUWENHUYSE, I. V. 2015. Simulation optimization in inventory replenishment: a classification. *IIE Transactions,* 47**,** 1217-1235.

JUAN, A. A., FAULIN, J., GRASMAN, S. E., RABE, M. & FIGUEIRA, G. 2015. A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives,* 2**,** 62-72.

KESKIN, B. B., MELOUK, S. H. & MEYER, I. L. 2010. A simulation-optimization approach for integrated sourcing and inventory decisions. *Computers & Operations Research,* 37**,** 1648-1661.

LAGODIMOS, A. 1992. Multi-echelon service models for inventory systems under different rationing policies. *International Journal of Production Research,* 30**,** 939-958.

LAGUNA, M. 2014. Scatter Search. 119-141.

LAW, A. M. 2015. *Simulation modeling and analysis,* New York, NY :, McGraw-Hill Education.

LEE, H. L., BILLINGTON, C. & CARTER, B. 1993. Hewlett-Packard gains control of inventory and service through design for localization. *Interfaces,* 23**,** 1-11.

LI, L., SOURIRAJAN, K. & KATIRCIOGLU, K. 2010. Empirical methods for two-echelon inventory management with service level constraints based on simulation-regression. *Proceedings of the 2010 Winter Simulation Conference***,** 1846-1859.

MARTÍ, R., LAGUNA, M. & GLOVER, F. 2006. Principles of scatter search. *European Journal of Operational Research,* 169**,** 359-372.

MOORS, J. J. A. & STRIJBOSCH, L. W. G. 2002. Exact fill rates for (R, s, S) inventory control with gamma distributed demand. *Journal of the Operational Research Society,* 53**,** 1268-1274.

MUCKSTADT, J. A. 1973. A model for a multi-item, multi-echelon, multi-indenture inventory system. *Management science,* 20**,** 472-481.

MUCKSTADT, J. A. 2006. *Analysis and Algorithms for Service Parts Supply Chains*, Springer New York.

ÓLAFFSON, S. & KIM, J. 2002. Simulation optimization. *Winter Simulation Conference***,** 79-84.

ÓLAFSSON, S. 2006. Chapter 21 Metaheuristics. *Handbook in OR & MS.*

ÖZER, Ö. 2003. Replenishment strategies for distribution systems under advance demand information. *Management Science,* 49**,** 255-272.

ÖZER, Ö. & XIONG, H. 2008. Stock positioning and performance estimation for distribution systems with service constraints. *IIE Transactions,* 40**,** 1141-1157.

PATERSON, C., KIESMÜLLER, G., TEUNTER, R. & GLAZEBROOK, K. 2011. Inventory models with lateral transshipments: A review. *European Journal of Operational Research,* 210**,** 125-136.

PAUL, B. & RAJENDRAN, C. 2011. Rationing mechanisms and inventory control-policy parameters for a divergent supply chain operating with lost sales and costs of review. *Computers & Operations Research,* 38**,** 1117-1130.

PEIDRO, D., MULA, J., POLER, R. & LARIO, F.-C. 2009. Quantitative models for supply chain planning under uncertainty: a review. *The International Journal of Advanced Manufacturing Technology,* 43**,** 400-420.

RAVI RAVINDRAN, A. & WARSING JR., D. 2012. *Supply chain engineering: models and applications,* Boca Raton, CRC Press.

RONG, Y., BULUT, Z. & SNYDER, L. V. 2012. Heuristics for Base-Stock Levels in Multi-Echelon Distribution Networks. *SSRN Electronic Journal***,** 1-20.

ROSENBAUM, B. 1981. Service level relationships in a multi-echelon inventory system. *Management Science,* 27**,** 926-945.

RUSSELL, R. A. & CHIANG, W.-C. 2006. Scatter search for the vehicle routing problem with time windows. *European Journal of Operational Research,* 169**,** 606-622.

SAETTA, S., PAOLINI, L., TIACCI, L. & ALTIOK, T. 2012. A decomposition approach for the performance analysis of a serial multi-echelon supply chain. *International Journal of Production Research,* 50**,** 2380-2395.

SCHNEIDER, H. & RINGUEST, J. 1990. Power approximation for computing (s,S) policies using service level. *Management Science,* 36**,** 822-834.

SCHNEIDER, H., RINKS, D. & KELLE, P. 1995. Power approximations for a two-echelon inventory system using service levels. *Production and Operations Management,* 4**,** 381-400.

SCHWARZ, L., DEUERMEYER, B. & BADINELLI, R. 1985. Fill-rate optimization in a one-warehouse N-identical retailer distribution system. *Management Science,* 31**,** 488-498.

SHANG, K. H. & SONG, J.-S. 2003. Newsvendor bounds and heuristic for optimal policies in serial supply chains. *Management Science,* 49**,** 618-638.

SHANG, K. H. & SONG, J.-S. 2006. A Closed-Form Approximation for Serial Inventory Systems and Its Application to System Design. *Manufacturing & Service Operations Management,* 8**,** 394-406.

SHERBROOKE, C. C. 1968. METRIC: A multi-echelon technique for recoverable item control. *Operations Research,* 16**,** 122-141.

SILVER, E. A., NASERALDIN, H. & BISCHAK, D. P. 2009. Determining the reorder point and order-up-to-level in a periodic review system so as to achieve a desired fill rate and a desired average time between replenishments. *Journal of the Operational Research Society,* 60, 1244-1253.

SILVER, E. A., PYKE, D. F. & PETERSON, R. 1998. *Inventory management and production planning and scheduling*, Wiley New York.

SIMCHI-LEVI, D., SIMCHI-LEVI, E. & KAMINSKY, P. 2009. *Designing and managing the supply chain: Concepts, strategies, and cases*, McGraw-Hill New York.

SIMCHI-LEVI, D. & ZHAO, Y. 2005. Safety Stock Positioning in Supply Chains with Stochastic Lead Times. *Manufacturing & Service Operations Management,* 7, 295-318.

SLACK, N., CHAMBERS, S. & JOHNSTON, R. 2010. *Operations management*, Pearson education.

SNYDER, L. & SHEN, Z. 2011. *Fundamentals of supply chain theory,* Hoboken, John Wiley & Sons.

SOBEL, M. J. 2004. Fill rates of single-stage and multistage supply systems. *Manufacturing & Service Operations Management,* 6, 41-52.

SWISHER, J. R., HYDEN, P. D., JACOBSON, S. H. & SCHRUBEN, L. W. 2000. A survey of simulation optimization techniques and procedures. *Proceedings of the 2000 Winter Simulation Conference.,* 1, 119-128.

TALBI, E.-G. 2009. *Metaheuristics: from design to implementation*, John Wiley & Sons.

TEKIN, E. & SABUNCUOGLU, I. 2004. Simulation optimization: A comprehensive review on theory and applications. *IIE Transactions,* 36, 1067-1081.

TSAI, S. C. & LIU, C. H. 2015. A simulation-based decision support system for a multi-echelon inventory problem with service level constraints. *Computers & Operations Research,* 53, 118-127.

TÜSHAUS, U. & WAHL, C. 1998. Inventory positioning in a two-stage distribution system with service level constraints. *Advances in Distribution Logistics.* Springer.

VAN DER HEIJDEN, M. 2000. Near cost-optimal inventory control policies for divergent networks under fill rate constraints. *International Journal of Production Economics,* 63, 161-179.

VAN DER HEIJDEN, M., DIKS, E. & DE KOK, A. 1997. Stock allocation in general multi-echelon distribution systems with (*R,S*) order-up-to-policies. *International Journal of Production Economics,* 49, 157-174.

VAN DONSELAAR, K., VAN WOENSEL, T., BROEKMEULEN, R. & FRANSOO, J. 2006. Inventory control of perishables in supermarkets. *International Journal of Production Economics,* 104, 462-472.

VAN HOUTUM, G.-J. & ZIJM, W. 2000. On the relationship between cost and service models for general inventory systems. *Statistica Neerlandica,* 54, 127-147.

VOJVODIC, G., JARRAH, A. I. & MORTON, D. P. 2016. Forward thresholds for operation of pumped-storage stations in the real-time energy market. *European Journal of Operational Research,* 254, 253-268.

WATSON, N. & ZHENG, Y.-S. 2005. Decentralized serial supply chains subject to order delays and information distortion: Exploiting real-time sales data. *Manufacturing & Service Operations Management,* 7, 152-168.