An approximate approach for the joint problem of level of repair analysis and spare parts stocking

**R.J.I. Basten · M.C. van der Heijden · J.M.J. Schutten ·
E. Kutanoglu**

Abstract For the spare parts stocking problem, generally METRIC type methods are used in the context of capital goods. A decision is assumed on which components to discard and which to repair upon failure, and where to perform repairs. In the military world, this decision is taken explicitly using the level of repair analysis (LORA). Since the LORA does not consider the availability of the capital goods, solving the LORA and spare parts stocking problems sequentially may lead to suboptimal solutions. Therefore, we propose an iterative algorithm. We compare its performance with that of the sequential approach and a recently proposed, so-called integrated algorithm that finds optimal solutions for two-echelon, single-indenture problems. On a set of such problems, the iterative algorithm turns out to be close to optimal. On a set of multi-echelon, multi-indenture problems, the iterative approach achieves a cost reduction of 3 % on average (35 % at maximum) as compared to the sequential approach. Its costs are only 0.6 % more than those of the integrated algorithm on average (5 % at maximum). Considering that the integrated algorithm may take a long time without guaranteeing optimality, we believe that the iterative algorithm is a good approach. This result is further strengthened in a case study, which has convinced Thales Nederland to start using the principles behind our algorithm.

**Keywords** Spare parts · Inventories · Level of repair analysis · Optimization · Heuristic

1 Introduction

Capital goods are physical systems that are used to produce products or services. They are expensive and technically complex, and they have high downtime costs. Examples of capital goods are manufacturing equipment, defense systems, and medical devices. Before capital

R.J.I. Basten (✉) · M.C. van der Heijden · J.M.J. Schutten
University of Twente, P.O. Box 217, 7500 AE, Enschede, The Netherlands
e-mail: r.basten@utwente.nl

E. Kutanoglu
University of Texas at Austin, 1 University Station, C2200, Austin, TX 78712, USA
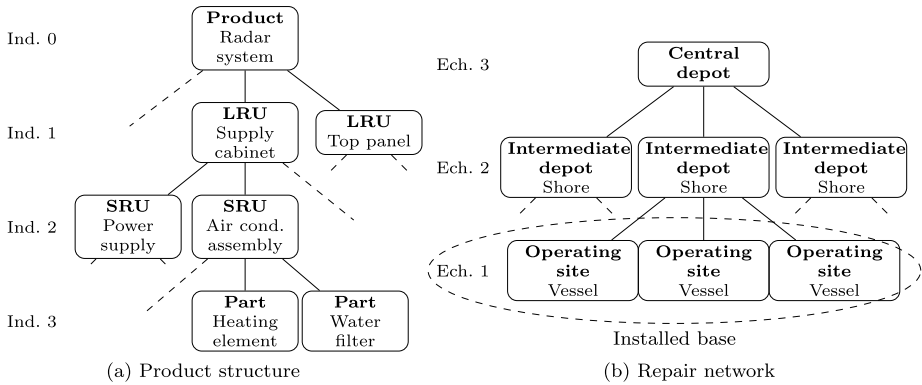
Fig. 1 Examples, with our naming convention in bold text (adapted from van der Heijden et al. 2012, p. 2)

goods are deployed, several tactical level questions concerning their corrective maintenance need to be answered: which components to repair upon failure and which to discard, where to perform the repairs, and which amount of spare parts to stock at which locations in the repair network. These questions should be answered such that a target availability of the capital goods (the installed base) is achieved against the lowest possible costs.

Due to the high downtime (unavailability) costs of capital goods, a defective capital good will usually be *repaired by replacement* of a component by a functioning spare part. In the defense industry, the components that are replaced are called *LRUs* or *line replaceable units*. It should be decided for each (type of) LRU whether it will be repaired or discarded upon failure, with discard implying that a new LRU needs to be acquired. Furthermore, it should be decided how many spare parts to stock for each LRU.

In general, the problem is more complicated due to two reasons. The first reason is that if LRUs are repaired, this is typically done by replacement of a subcomponent, called *SRU* or *shop replaceable unit*. Such an SRU may itself consist of subcomponents, called parts. This means that spare parts need to be stocked both to enable quick repairs of the capital goods (by stocking spare LRUs) and to enable quick repairs of LRUs and SRUs (by stocking spare SRUs and parts, respectively). Figure 1a gives an example of a *multi-indenture product structure*, including the naming convention that we use. We use the terms *component* and *subcomponent* if the indenture level is irrelevant.

The second complication is due to the fact that repairs and discards may be performed at various echelon levels in the *multi-echelon repair network*, an example of which is shown in Fig. 1b, including the naming convention that we use. Notice that in general, there may be any number of indenture levels and echelon levels.

To be able to perform repairs, discards, or movements of components from one echelon level to the next, *resources* may be required. Resources include test, repair, and transportation equipment, but one time training of service engineers may also be considered as a resource for which a one-time investment is required.

The *level of repair analysis* (*LORA*) can be used to make the above mentioned *LORA decisions*:

1. which components to repair upon failure and which to discard;
2. at which locations in the repair network to perform the repairs and discards; and
3. at which locations to deploy resources.

The first two sets of decisions are called the *repair/discard decisions*.

The LORA is typically modeled as a deterministic integer linear optimization problem (see, e.g., Basten et al. 2009, 2011a). Including the waiting times for spare parts (unavailability of the capital goods) would lead to non-linear constraints. Since the spare parts stocking problem in itself is already hard to solve to optimality (see, e.g., Sherbrooke 2004; Muckstadt 2005), it is improbable that such a non-linear model can be solved easily, or that the non-linear constraints can be linearized in a meaningful way (especially in the case of multi-echelon, multi-indenture problems).

As a result, only an estimate of the spare parts holding costs may be considered in the LORA problem, and often those costs are ignored completely. Instead, the other relevant costs are considered, consisting of both fixed costs and costs that are variable in the number of failures. Fixed costs are due to the resources. They result from a certain repair/discard decision, but are incurred no matter how often components are actually repaired or discarded, for example, costs for training of service engineers and depreciation costs of repair equipment. Variable costs may include transportation costs, working hours of service engineers, and usage of bulk items.

Using the LORA decisions as an input, the *spare parts stocking* problem is solved to decide which components to put on stock at which location(s) in the repair network in which quantity, such that a target availability of the capital goods is achieved against minimum holding costs. A well-known method to solve this problem is (VARI-)METRIC (see, e.g., Sherbrooke 2004; Muckstadt 2005), which is a greedy heuristic that is known to find solutions that are close to optimal (see also Sect. 2.2).

Performing the LORA first and then the spare parts stocking analysis, the *sequential approach*, may lead to a solution that is not optimal. For example, if repairs are performed at the operating sites, each operating site requires a resource, whereas only one resource may be required in total if repairs are performed at the central depot. As a result, the LORA often recommends to perform repairs at the central depot (repairing centrally leads to higher transportation costs of components, but these costs are generally low compared to costs for resources in a high-tech environment). The LORA neglects the fact that when repairs are performed centrally, the repair lead times (including transportation lead times) are higher than when repairs are performed at the operating sites, thus leading to higher spare parts requirements. This is especially problematic if the holding costs make up a large percentage of the total costs, as we have observed in a case study in the defense industry (see Sect. 6).

We propose an iterative algorithm to solve the joint problem of LORA and spare parts stocking. The basic idea is to first solve a LORA, next use VARI-METRIC to solve the spare parts stocking problem, and then use the results of VARI-METRIC to add an estimate of the holding costs to the LORA inputs and start a second iteration. We continue in this way until we do not find a different solution anymore. We compare our results with the sequential solution (this is the solution of the first iteration of our algorithm) and with the solutions resulting from an algorithm that was recently proposed by Basten et al. (2012) for two-echelon, single-indenture problems. Their so-called *integrated algorithm* finds optimal solutions, or in fact, efficient points on the curve of costs versus expected number of backorders (see Sect. 2.2). The integrated algorithm can be extended to multi-echelon, multi-indenture problems, but Basten et al. (2012, Appendix) explain that in that case, finding efficient points cannot be guaranteed. However, the integrated algorithm still finds solutions that are close to optimal (see Sect. 5.2.1). The key drawback of the integrated algorithm is that it is very slow because it implicitly enumerates all possible solutions. For example, we are able to solve a case study (see below) using our iterative algorithm in less than one minute, whereas the integrated algorithm requires almost two days.

We perform an extensive numerical experiment to test the performance of our algorithm. On a set of two-echelon, single-indenture problems, the iterative algorithm achieves a cost reduction of 3.80 % on average compared with the sequential approach, whereas the integrated approach achieves a cost reduction of 5.07 % on average. This means that the iterative algorithm closes most of the optimality gap of the sequential approach. Using a set of multi-echelon, multi-indenture problems, we find that the iterative algorithm is much faster than the integrated algorithm, while its solution value is on average only 0.58 % higher than that of the integrated algorithm (5.26 % higher at maximum). Compared with the sequential procedure, the iterative algorithm achieves a cost reduction of 2.85 % on average and 34.69 % at maximum. In a case study at Thales Nederland, a manufacturer of naval sensors and naval command and control systems, we show that solving the joint problem iteratively instead of sequentially leads to a cost reduction of almost 10 %, which is worth a couple of millions of euros over the life time (over 20 years) of twelve sensor systems. Because of these results, the principles behind our algorithm are now in use at Thales Nederland.

The remainder of this paper is organized as follows. In Sect. 2, we discuss the related literature. We outline our model for the joint problem of LORA and spare parts stocking in Sect. 3, and in Sect. 4, we present the iterative algorithm. In Sect. 5, we show the results of our numerical experiment, and we then present the results of the case study that we performed in Sect. 6. We give conclusions and recommendations for further research in Sect. 7.

2 Literature review

We discuss the literature on LORA, spare parts stocking, and the joint problem of LORA and spare parts stocking in Sects. 2.1, 2.2, and 2.3, respectively.

2.1 Level of repair analysis

Barros (1998) proposes a multi-echelon, multi-indenture LORA model in which decisions are taken per echelon level. So, if it is decided to repair a certain component at a certain operating site, it is also repaired at all other operating sites. Barros further assumes that all components at a certain indenture level require the same resource and that resources are uncapacitated. The latter means that there is no downtime waiting for resources, and either zero or one resource is located at each location. As in all papers on LORA, Barros formulates her model as an integer programming model. She solves it using CPLEX. Barros and Riley (2001) use the same model as Barros does and solve it using a branch-and-bound approach.

Saranga and Dinesh Kumar (2006) make the same assumptions as Barros (1998), except that the former assume that each component requires its own unique resource. They use a genetic algorithm to solve the model. Basten et al. (2009) generalize the two aforementioned models by allowing for components requiring multiple resources and multiple components requiring the same resource. As in the remaining three papers in this section, Basten et al. (2009) use CPLEX to solve the model.

Basten et al. (2011a) generalize the model of Basten et al. (2009) by allowing for different decisions at the various locations at one echelon level. They show that the LORA problem can be modeled efficiently as a generalized minimum cost flow model. Basten et al. (2011b) propose a number of extensions to the model of Basten et al. (2011a) so that, for example, a probability of unsuccessful repair can be modeled, or capacitated resources. The latter does not mean that waiting times are incorporated.

Brick and Uchoa (2009) use similar assumptions as Basten et al. (2011a), except that the former assume that resources have a maximum capacity (as Basten et al. 2011b, do). They further consider one echelon level only and effectively assume two indenture levels. Integrated in their LORA is the decision of which facilities to open (facility location problem).

2.2 Spare parts stocking

In the area of capital goods, the paper of Sherbrooke (1968) is generally seen as the seminal paper on the multi-item spare parts stocking problem. Sherbrooke develops the METRIC model (Multi-Echelon Technique for Recoverable Item Control), which is the basis for a huge stream of METRIC type models. These models can be used both for repairable and for consumable parts. The goal is to find the most cost effective allocation of spare parts in a network, such that a target availability of the capital goods is achieved. This is achieved by focusing on the expected number of backorders (EBO): if a spare part is requested, but not available yet, this is called a *backorder*. As an approximation, the number of backorders of LRUs at operating sites equals the number of systems that are unavailable waiting for spares. The METRIC type methods focus on minimizing the expected number of backorders, instead of maximizing the availability, because this allows for decomposing the overall problem into subproblems per LRU. A marginal analysis approach is used to construct an EBO-curve. Each point on this curve shows the spare parts holding costs versus expected number of backorders resulting from an allocation of a set of spare parts to one or various locations. Construction of the curve is stopped as soon as the number of backorders has decreased enough to achieve the target availability. Generally, the achieved availability is somewhat higher than the target availability since the EBO-curve consists of a discrete set of points. This is called *overshoot*.

One key assumption in these models is that demand at the operating sites follows a Poisson process. A second key assumption is that an $(S-1, S)$ continuous review inventory control policy is used. This means that if a spare part is requested from a stock point, this stock point immediately requests a spare part at the next higher echelon level (or immediately orders a new component or repairs the broken component, depending on the repair/discard strategy that is used). As a result, demand at higher echelon levels follows a Poisson process as well, and the number of components in repair or in the replenishment loop (after discard) at the highest location is thus Poisson distributed. However, the number of backorders at that location is *not* Poisson distributed if there is a positive number of spare parts located there. As a result, analysis of the so-called pipeline at the lower echelon levels gets complicated, the pipeline being the number of components that is sent upwards for repair or discard, and not replaced by a functioning component yet, plus the number of components in the repair loop at the current location. Sherbrooke (1968) chooses to approximate the number of items in the pipeline by assuming that it is also Poisson distributed. Muckstadt (1973) extends the work by Sherbrooke (1968); the latter considers single-indenture product structures only, whereas the former develops a multi-echelon, multi-indenture model, called MOD-METRIC. The development of the VARI-METRIC models (Graves 1985; Sherbrooke 1986) has been the next important step forward: a two-moment approximation is used for the pipelines. It is also possible to evaluate the model exactly (Graves 1985; Rustenburg et al. 2003), but this is computationally intensive, and VARI-METRIC is known to give small errors only. Furthermore, backorders at higher echelon levels are not the only cause of delays; backorders for subcomponents can delay repairs of components in a way that is similar to what we described above.

2.3 Joint problem of level of repair analysis and spare parts stocking

We are aware of two papers in which a method is presented to solve the joint problem of LORA and spare parts stocking: Alfredsson (1997) and Basten et al. (2012).

Alfredsson (1997) assumes a single-indenture product structure and a two-echelon repair network. He further assumes that each component requires exactly one tester (resource) and that all components that require the same tester are repaired at the same location. Furthermore, one multi-tester exists. It can be used for the repair of a number of components, and adapters can be added in a fixed order to enable the multi-tester to be used for the repair of additional components. If the multi-tester can be used to repair a certain component, then this component necessarily uses the multi-tester instead of the original resource that it used. Resources are capacitated, which means that multiple resources of the same type may be required at one location. System downtime includes the waiting times for the resources, the repair times, and the waiting times for spares. The problem is modeled as a non-linear integer programming model and Alfredsson uses a decomposition method that sequentially decomposes the overall problem into smaller subproblems to solve the model.

Basten et al. (2012) also consider single-indenture, two-echelon problems, but they allow for more general component-resource relations: components may share resources and a component may require multiple resources simultaneously. This substantially complicates the problem. The basic idea of their so-called *integrated algorithm* is to recursively decompose the problem in a smart way such that all possible solutions are enumerated without taking too much time. For the single-indenture, two-echelon problem, Basten et al. (2012) find convex EBO-curves consisting of efficient points. This means that it is not possible to achieve a lower expected number of backorders for the costs that they find. They also show that it cannot be guaranteed that efficient points are found for general multi-indenture problems.

3 Model

In this section, we outline the model that we use. We present our assumptions in Sect. 3.1, and in Sect. 3.2, we give the mathematical model formulation.

3.1 Assumptions

A key assumption is that we make the same (LORA and stocking) decisions at all locations at one echelon level for each component and resource. This implicitly means that we assume symmetrical repair networks, i.e., we have the same costs, demand rates, lead times, et cetera at all locations at one echelon level, and the same number of locations being replenished from a location at the next higher echelon level. In such a network, taking the same decision at all locations at one echelon level is an optimal strategy, except that the overshoot increases (see Sect. 2.2). We discuss relaxation of this assumption in Sect. 7.

The remaining assumptions that we make are commonly made for the LORA problem (see, e.g., Basten et al. 2011a) and in the METRIC type models (see, e.g., Sherbrooke 2004; Muckstadt 2005). We assume that:

– Components fail according to a Poisson process with a constant rate.
– Replacement of a defective LRU by a functioning one takes zero time. This effectively means that we focus on the supply availability, not on the operational availability that also includes the actual replacement time (see, e.g., Sherbrooke 2004, p. 38).

After a component is replaced by a function spare part, if one is available, the defective component may either be repaired at the operating site, discarded at the operating site, or be shipped to the next higher echelon level (move option). At that echelon level, the same three options are available (only at the central depot, there is no move option). We assume that:

– Discarding a component implies that its subcomponents are discarded as well.
– The replenishment lead times for (the newly purchased replacements of) discarded components are independent and identically, generally distributed random variables. The replenishment lead time is the time between failure of the component and reception of the newly purchased component from an external source at the discard location.
– Each subcomponent may cause the failure of a component (otherwise, this subcomponent need not be modeled), so repairing a component may result in replacement of any one subcomponent. As a result, if it is decided to repair a component at a certain echelon level, a further decision needs to be taken for each subcomponent at the same echelon level (repair, discard, or move the subcomponent).
– A failure in a component is caused by a failure in at most one subcomponent. In other words, a component cannot fail due to failure of two or more subcomponents simultaneously.
– Repairs are always successful.
– The repair lead times are independent and identically, generally distributed random variables that include the time used for sending the failed component to the repair location and for diagnosing the failure cause.
– A failed (sub)component may not be shipped to a lower echelon level. So, if a component is repaired at echelon level e by replacing a subcomponent, this subcomponent may only be repaired at an echelon level $f \geq e$.
– The move lead time (to move a functioning, repaired or newly purchased, component from a location to one of its child locations) is deterministic.
– Resources are uncapacitated, meaning that at most one resource of a certain type needs to be installed at each location.
– Minimizing the expected number of backorders is a good approximation of maximizing the availability (see Sect. 2.2).
– There are no lateral transshipments between locations at the same echelon level or emergency shipments from locations at a higher echelon level; functioning spare parts are only supplied from one specific location at the next higher echelon level.
– For each component at each location, an $(S - 1, S)$ continuous review inventory control policy (one for one replenishment) is used (see Sect. 2.2).

To ease the presentation in the remainder of this paper and to decrease the problem size, we make three additional assumptions. Our algorithm is easily modified such that these assumptions can be removed:

– There is no commonality, so a subcomponent may not be part of two different components.
– Since resources that are required to enable discard or movement do not occur frequently in practice, e.g., not in our case study, we assume that resources may be required to enable repair only.
– Since the discard costs mainly consist of the costs of acquiring a new component, and since those costs are generally much higher than move costs, we consider discard at the highest echelon level only. If newly purchased components can enter the repair network at the central warehouse only, then this assumption does not influence the replenishment times.

3.2 Mathematical model

In Sect. 3.2.1, we introduce the notation that we use and we give the mathematical model in Sect. 3.2.2.

3.2.1 Notation

Let C be the set of all components, with $C_1 \subseteq C$ being the set of LRUs. Γ_c is the (possibly empty) set of subcomponents of component $c \in C$ at the next higher indenture level.

The set E consists of all echelon levels, the highest echelon level being $e_{\max}$. The set D consists of the possible decisions that can be made: $D = \{\text{discard, repair, move}\}$. The set of options that is available at echelon level $e \in E$ is D_e. For $e \in E \setminus e_{\max}$, $D_e = D$, and $D_{e_{\max}} = \{\text{discard, repair}\}$.

Let R be the set of resources. $\Omega_r \subseteq C$ is the set of components that require resource r in order to be repaired (if component c requires two resources, r_1 and r_2, then $c \in \Omega_{r_1}$ and $c \in \Omega_{r_2}$).

We define the following decisions variables:

$$X_{c,e,d} = \begin{cases} 1, & \text{if for component } c \in C \text{ at echelon level } e \in E \text{ decision } d \in D_e \text{ is made,} \\ 0, & \text{otherwise;} \end{cases}$$

$$Y_{r,e} = \begin{cases} 1, & \text{if resource } r \text{ is located at echelon level } e, \\ 0, & \text{otherwise;} \end{cases}$$

$S_{c,e} =$ the number of spare parts of component c located at each location at echelon level e.

Furthermore, we denote $\mathcal{X}$ as the three-dimensional array with entries $X_{c,e,d}$ and $\mathcal{S}$ as the two-dimensional array with entries $S_{c,e}$.

For each component $c \in C$, we define λ_c (> 0) as the total annual failure rate over all operating sites. We define three cost types. For component $c \in C$ at echelon level $e \in E$, $v_{c,e,d}$ (≥ 0) are the variable costs of making decision $d \in D$. Since we have chosen, without loss of generality, to minimize the total annual costs with our definition of λ_c, we define $f_{r,e}$ (≥ 0) to be the annual fixed costs to locate resource $r \in R$ at echelon level $e \in E$ and we define $h'_{c,e}$ (> 0) to be the annual costs of holding one spare of component $c \in C$ at each location at echelon level e (we use the prime to ease notation later on).

3.2.2 Mathematical model formulation

We define our model as follows:

$$\text{minimize} \sum_{c \in C} \sum_{e \in E} \sum_{d \in D_e} v_{c,e,d} \cdot \lambda_c \cdot X_{c,e,d} + \sum_{r \in R} \sum_{e \in E} f_{r,e} \cdot Y_{r,e} + \sum_{c \in C} \sum_{e \in E} h'_{c,e} \cdot S_{c,e} \quad (1)$$

subject to:

$$\sum_{d \in D_1} X_{c,1,d} = 1, \quad \forall c \in C_1, \quad (2)$$

$$X_{c,e,\text{move}} \leq \sum_{d \in D_{e+1}} X_{c,e+1,d}, \quad \forall c \in C, \ \forall e \in E \setminus e_{\max}, \quad (3)$$

$$X_{c,e,\text{repair}} \leq \sum_{d \in D_e} X_{b,e,d}, \quad \forall c \in C, \ \forall b \in \Gamma_c, \ \forall e \in E, \quad (4)$$

$$X_{c,e,\mathrm{repair}} \le Y_{r,e}, \quad \forall r \in R, \ \forall c \in \Omega_r, \ \forall e \in E, \tag{5}$$

$$\mathrm{availability}(\mathcal{X}, \mathcal{S}) \ge \mathrm{target\ availability}, \tag{6}$$

$$X_{c,e,d}, Y_{r,e} \in \{0, 1\}, \tag{7}$$

$$S_{c,e} \in \mathbb{N}. \tag{8}$$

Constraints (2) to (5) are the 'LORA constraints' and define the same model as Basten et al. (2011a) use, except that they do not necessarily take the same decision at each location at one echelon level. Constraint (2) assures that for each LRU a decision is made at the operating sites. If a component is discarded, no further decisions need to be made for that component or its subcomponents. If a component is moved, Constraint (3) assures that a decision is made for that component at the next higher echelon level, and if a component is repaired, Constraint (4) assures that a decision is made for each of its subcomponents. Some options are only available if all resources are present, which is guaranteed by Constraint (5). Finally, Constraint (6) assures that the target availability is met; this is the only 'spare parts stocking constraint'. As explained in Sect. 2.2, given the annual demand for spare parts at the various locations, enough spare parts need to be stocked throughout the network to achieve such expected number of backorders that the availability of the capital goods is higher than the target availability. Since the availability is a non-linear function of all repair/discard decisions $\mathcal{X}$ and all spare parts decisions $\mathcal{S}$, our model cannot be solved using an ILP solver. Therefore, we propose an iterative algorithm in Sect. 4. Notice that in Constraint (6), the various lead times play a role (we have not introduced notation for those lead times).

4 Iterative algorithm

As mentioned in Sect. 1, the joint problem of LORA and spare parts stocking analysis is in practice usually solved sequentially. First, a LORA is performed, focusing on achieving the lowest possible costs, consisting of both fixed costs ($\sum_{r \in R} \sum_{e \in E} f_{r,e} \cdot Y_{r,e}$), and costs that vary with the number of failures ($\sum_{c \in C} \sum_{e \in E} \sum_{d \in D} v_{c,e,d} \cdot \lambda_c \cdot X_{c,e,d}$). Next, given the decisions that result from the LORA, a spare parts stocking problem is solved (e.g., using VARI-METRIC) that determines where to locate spare parts in the repair network, such that a target availability of the capital goods is achieved against the lowest possible spare parts holding costs ($\sum_{c \in C} \sum_{e \in E} h'_{c,e} \cdot S_{c,e}$).

We propose an iterative algorithm that uses in each iteration the same two building blocks as the sequential approach (see Fig. 2). After the first iteration, we therefore have the solution of the sequential method. The spare parts holding costs are then used to adapt the LORA inputs so that a second iteration of LORA and spare parts stocking may be performed. The key idea is that, as an approximation, we may decompose the holding costs into holding costs per component, so that for each component c the holding costs are $\sum_{e \in E} h'_{c,e} \cdot S_{c,e}$. The implicit assumption is that the holding costs that result from a repair/discard decision are independent of the decisions taken for the other components. Of course, this assumption is violated, since VARI-METRIC is a multi-item approach (an example is given below, with the data presented in Table 1). Although our iterative algorithm is somewhat similar to Bender's decomposition (Benders 1962) in that we use a master problem (the LORA problem) and a subproblem (the spare parts stocking problem), there are two key differences: (1) our algorithm is a heuristic, whereas Bender's decomposition is guaranteed to find the optimal solution, and (2) we are not adding constraints or cuts to the master problem, but we are changing the coefficients in the constraints.
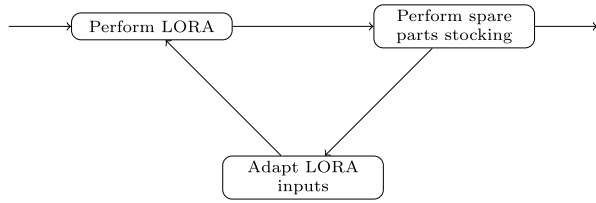
Notice that the move decisions can be seen as 'intermediate' decisions; the decision to repair or discard a component is the 'final' decision. Therefore, we need to adapt the costs for the repair and discard decisions only. We define $h_{c,e,d}^i$ as the spare parts holding costs that are added to the variable costs of component $c \in C$ for decision $d \in \{\text{discard, repair}\}$ at echelon level $e \in E$ in iteration $i \geq 1$. So, the variable costs that are used in the LORA for component $c \in C$ at echelon level $e \in E$ in iteration i are $v_{c,e,d} + h_{c,e,d}^i$ for decision $d \in \{\text{repair, discard}\}$, and $v_{c,e,d}$ for decision $d = \text{move}$. In the first iteration, $h_{c,e,d}^1 = 0$ for all tuples (c, e, d). For each tuple (c, e, d) for which $X_{c,e,d} = 1$ in iteration $i - 1$ ($i > 1$), we set $h_{c,e,d}^i = \frac{\sum_{f \in E} h_{c,f}' \cdot S_{c,f}}{\lambda_c}$ ($S_{c,f}$ resulting from iteration $i - 1$; division by λ_c because $v_{c,e,d} + h_{c,e,d}^i$ is multiplied by λ_c in the objective function). For all other possible repair and discard decisions ($X_{c,e,d} = 0$ in iteration $i - 1$), we set $h_{c,e,d}^i = h_{c,e,d}^{i-1}$. This means that the holding costs that we include in the LORA inputs are changed in iteration i only if the related repair/discard decision was chosen in iteration $i - 1$. In this way, we gradually find an estimate for the resulting holding costs for all relevant repair/discard decisions and the algorithm will eventually find a LORA solution that leads to low *total* costs: LORA costs (excluding the added holding costs) plus holding costs resulting from the spare parts stocking analysis. We stop the algorithm as soon as the LORA solution is identical in two consecutive iterations. If in the second of these iterations, two different LORA solutions exist that lead to the same costs, we choose the one we also had in the previous iteration so that the algorithm terminates. In Appendix B, we show that the algorithm cannot cycle between two solutions and that it therefore necessarily terminates after a finite number of iterations.

We use an example to illustrate the feedback mechanism. We consider a radar system that consists of two components ($C = \{A, B\}$). The radar system is installed at two ships (echelon level 1), which are supported by a depot (echelon level 2, so $E = \{1, 2\}$). LRUs A and B both require a unique resource in order to enable repair ($R = \{r_1, r_2\}$, $\Omega_{r_1} = \{A\}$, $\Omega_{r_2} = \{B\}$), the fixed annual costs of which are €10,000 and €25,000, respectively ($f_{r_1,1} = 20,000$, $f_{r_1,2} = 10,000$, $f_{r_2,1} = 50,000$, $f_{r_2,2} = 25,000$). For both LRUs ($c \in \{A, B\}$), the annual failure rate per ship is 1 ($\lambda_c = 2$), the discard costs are €15,000 ($v_{c,2,\text{discard}} = 15,000$), the variable repair costs are €6,000 ($v_{c,e,\text{repair}} = 6,000$), and the move costs are €0 ($v_{c,1,\text{move}} = 0$).

In the first iteration, holding costs of zero are included in the LORA problem. Therefore, the repair/discard options with the lowest LORA costs are chosen for both LRUs (see Table 1 for an overview of all resulting costs): A is repaired at depot, which leads to annual costs of €22,000 (variable repair costs are 2 times €6,000 and a resource at the depot costs €10,000), and B is discarded, which leads to annual discard costs of €30,000. Next, the spare parts stocking problem is solved, which results in stocking spare parts at both the ships and the depot, leading to annual holding costs of €16,000 for A and €30,000 for B. In the second iteration, the LORA is solved with modified inputs. The LORA chooses to discard A, since that leads to costs of €30,000, whereas repair at depot leads to total costs of €22,000 + €16,000 = €38,000. For B, repair at depot is the most cost effective option.

Table 1 Costs in the LORA problem (×€1,000)

Decision	LORA costs $(v_{c,e,d} \cdot \lambda_c + f_{r,e})$		Spare parts holding costs $(h_{c,e,d}^{i+1})$ after					
			iteration 1		iteration 2		iteration 3	
	LRU		LRU		LRU		LRU	
	A	B	A	B	A	B	A	B
Repair at ship	32	62	0	0	0	0	*4*	0
Repair at depot	22	37	*16*	0	16	*20*	16	*15*
Discard	30	30	0	*30*	*20*	30	20	30
Total costs (LORA and spares)			98		107		88	

We next find holding costs of €20,000 for both LRUs. Notice that the total costs in the second iteration (€107,000) are higher than those in the first iteration (€98,000). In the third iteration, it is decided to repair A at ship, and B at depot. This results in annual holding costs of €4,000 for A and €15,000 for B.

Notice that the holding costs for B change, although the repair/discard decision for B does not change. This is a result of the system approach that is used in VARI-METRIC: a change in the repair/discard decision for one LRU (A) may change the number of spare parts that should be stocked of another LRU (B). We simply replace the old costs by the newly calculated costs. Notice furthermore that for A, we found the holding costs estimate related to 'repair at depot' when 'discard' was chosen for LRU B. This value may be lower if B is repaired at ship or at depot and as a result, in the optimal solution, we may have to repair A at depot. However, the solution in the next iterations will be to repair A at ship and to repair B at depot. This risk of using holding costs that are too high is the key drawback of our approach and it may result in not selecting a cost-effective option anymore. As a result, we may end up with a non-optimal solution.

It is possible to slightly improve the feedback algorithm. For example, instead of replacing an old value $(h_{c,e,d}^{i-1})$ by a new value $(h_{c,e,d}^{i} = \frac{\sum_{f \in E} h'_{c,f} \cdot S_{c,f}}{\lambda_c})$, we may take a weighted average of the old and new value $(h_{c,e,d}^{i} = \alpha \cdot \frac{\sum_{f \in E} h'_{c,f} \cdot S_{c,f}}{\lambda_c} + (1-\alpha)h_{c,e,d}^{i-1}$, with $0 < \alpha < 1$). However, such improvements require setting additional values (what is a good value for α?), they make the algorithm more difficult to grasp and implement, and they lead to higher computation times because the values $h_{c,e,d}^{i}$ converge slowly to their correct value. Therefore, we do not consider them here. Basten (2010) shows the results of implementing three such improvements.

5 Numerical experiment

We design a numerical experiment that we present in Sect. 5.1. In Sect. 5.2, we discuss the results of our tests by answering the following questions:

1. What cost reduction can be achieved by solving the joint problem of LORA and spare parts stocking iteratively instead of sequentially?
2. How does the iterative algorithm perform compared with the integrated algorithm?
3. Which model parameters influence the cost reductions that may be achieved by solving the joint problem using the integrated or iterative algorithm instead of sequentially?

4. How do the repair strategies change when solving the joint problem using the integrated or iterative algorithm instead of sequentially?

For the two-echelon, single-indenture problem, Question 2 effectively means comparison of the iterative algorithm with the optimal solution. Therefore, our experiment consists of a set of two-echelon, single-indenture problem instances, and a set of multi-echelon, multi-indenture problem instances. For the latter set, we have extended the algorithm by Basten et al. (2012).

The algorithms are implemented in Delphi 2007 and problems instances are solved on an Intel Core 2 Duo P8600@2.40 GHz, with 3.5 GB RAM, under Windows XP SP 3. For the iterative and sequential algorithm, the LORA building block consists of the model of Basten et al. (2009) (but implemented using the minimum cost flow model of Basten et al. 2011a). *Optimization* in the spare parts stocking building block is done using the greedy heuristic that is typically used, and is described, for example, by Muckstadt (2005) and Basten et al. (2012). *Evaluation* is done using the two-moment approximation of Graves (1985) (VARI-METRIC, extended to the general multi-echelon, multi-indenture problem as described by Rustenburg et al. 2003) for two reasons. For multi-echelon, multi-indenture problems:

1. exact evaluation is known to be computationally intensive (see Sect. 2.2); and
2. using the greedy heuristic, it is not guaranteed to find optimal base stock levels, as shown, for example, in the appendix of Basten et al. (2012).

For the two-echelon, single-indenture test set, we also show the results for the integrated algorithm using the exact evaluation of Graves (1985) (the resulting solution of the greedy heuristic is optimal in this case). We can thus show that the difference between the exact and approximate evaluation is minor, which suggests that the (approximate) integrated algorithm will find solutions that are close to optimal for the multi-echelon, multi-indenture problem as well.

5.1 Design

A detailed description of how we generate the problem instances can be found in Appendix B; here we only give an overview.

We use the same generator as Basten et al. (2012) use to generate a set of 1,280 two-echelon, single-indenture problem instances. In each problem instance there are 100 LRUs, 10 resources, and 5 operating sites. Using a full factorial design, we vary the costs of each component and resource, the holding costs, and the discard, repair, and move lead times. We further vary the number of components that require the same resource. For each combination of parameter settings, we generate ten problem instances, in order to obtain a variety of problem instances. Each problem is solved using a target availability of 95 %.

The generator that we use to generate multi-echelon, multi-indenture problem instances is an extended version of the former generator. We have three sets of problem instances, each having its own focus:

1. Varying the problem size, the holding costs, and the lead times.
2. Varying the attractiveness of acquiring resources by changing the annual demand rate and the costs of resources and components (resulting in different variable repair, discard, and move costs).
3. Varying the component-resource relations.

In total, sets 1, 2, and 3 consists of 1,280, 160, and 80 problem instances, respectively.

Table 2 Overview of the results for the two-echelon, single-indenture problem instances

Algorithm	Approximate or exact	Average achieved availability	Cost reduction compared with sequential (approximate)	
			average	maximum
Sequential	approximate	95.14 %	–	–
Iterative[a]	approximate	95.07 %	3.80 %	35.46 %
Integrated	approximate	95.11 %	5.07 %	43.26 %
Integrated	exact	95.11 %	5.07 %	43.26 %

[a]The average number of iterations is 6.6, with a minimum of 5 and a maximum of 11

Table 3 Overview of the results for the multi-echelon, multi-indenture problem instances

	Computation time in seconds		Achieved availability		Cost reduction compared with			
					sequential		iterative	
	average	maximum	average	maximum	average	maximum	average	maximum
Sequential	0.18	1.83	95.25 %	96.32 %	–	–	–	–
Iterative[a]	4.50	41.09	95.11 %	96.12 %	2.85 %	34.69 %	–	–
Integrated	155.81	10,456.37	95.20 %	96.24 %	3.40 %	36.88 %	0.58 %	5.26 %

[a]The average number of iterations is 8.9, with a minimum of 5 and a maximum of 18

5.2 Results

We address the questions that we posed at the start of Sect. 5. In Sect. 5.2.1, we compare the results of the iterative algorithm with those of the sequential approach and the integrated algorithm of Basten et al. (2012) at a high level, and in Sect. 5.2.2, we analyze how repair strategies change and which parameters influence the results.

5.2.1 Comparison of sequential, iterative, and integrated algorithms

Table 2 gives an overview of the results on the two-echelon, single-indenture test set of Basten et al. (2012). Compared with solving the two problems sequentially, solving the joint problem using the integrated algorithm results in a cost reduction of 5.07 % on average and more than 43 % at maximum. Notice that both exact and approximate versions of the integrated algorithm produce nearly identical solutions (it differs in 2 of the 1,280 problem instances only). The iterative algorithm achieves most of the cost reductions that may be achieved in this test set.

We are mainly interested in the multi-echelon, multi-indenture problem instances. For a fair comparison, we use the approximate evaluation for all three algorithms here. Table 3 gives an overview of the results over the three test sets. A key observation is that the results are in line with the results on the two-echelon, single-indenture problem instances. This suggests that the integrated algorithm finds solutions that are still close to optimal and that the iterative algorithm is very robust. The iterative algorithm achieves a cost reduction of 2.85 % on average and almost 35 % at maximum compared with the sequential approach. In 9.5 % of the problem instances, a cost reduction of over 10 % is achieved (not shown in a table). The gap with the integrated algorithm is less than 2.5 % in over 95 % of the problem

Table 4 Cost reduction for important parameter settings (multi-echelon, multi-indenture problem instances)

Test set	Parameter	Setting	Average cost reduction		
			iterative versus sequential	integrated versus sequential	integrated versus iterative
1	# LRUs	50	4.35 %	4.98 %	0.69 %
		100	1.10 %	1.46 %	0.37 %
	Move lead time	[0.5/52; 4/52]	4.59 %	5.29 %	0.76 %
		[2/52; 4/52]	0.86 %	1.15 %	0.30 %
2	Demand per LRU	[0.01; 0.10]	10.96 %	12.06 %	1.20 %
		[0.01; 0.25]	2.47 %	2.73 %	0.27 %
		[0.01; 0.50]	1.32 %	2.77 %	1.47 %
		[0.01; 1.00]	3.09 %	4.61 %	1.56 %

instances, which means that chances of being a few percent off are very small using the iterative algorithm.

The integrated algorithm requires on average about 35 times as much computation time as the iterative algorithm, due to the integrated algorithm's enumerative approach (see Table 3). At maximum, the integrated algorithm requires almost three hours, which is more than 250 times as much as the iterative algorithm. This clearly shows that the iterative algorithm scales much better (this effect is even stronger for the case study, see Sect. 6.3).

There are some problem instances for which the integrated algorithm yields higher costs than the iterative approach, at most 2.76 % (not shown in a table). This is due to the overshoot problem (see Sects. 2.2 and 6.3, especially Fig. 4). The integrated algorithm yields a higher availability in these cases as well. There are no problem instances on which the iterative algorithm yields both lower costs and a higher availability than the integrated algorithm.

5.2.2 Detailed analysis of repair strategies and important parameters

Here, we focus on the multi-echelon, multi-indenture problem instances only. For the computation times, all results are as may be expected. Computation times increase using either of the three approaches when the number of indenture levels, number of LRUs, number of echelon levels, or the demand increases. For the integrated algorithm, the computation times also increase when components require more resources on average.

Table 4 gives the cost reductions for those parameter settings that substantially impact the cost reductions that we achieve. We discuss the results below, including a discussion of the changes in the repair strategies. The other parameters that we varied in test sets 1, 2, and 3 do not have a substantial impact on the achieved cost reductions.

We see that if the difference between the integrated and the sequential algorithm increases, then the difference between the integrated and the iterative algorithm increases as well, but not as fast. This is interesting, since it means that if it becomes more important to solve the two problems of LORA and spare parts stocking jointly, then the performance of the iterative algorithm relative to the integrated algorithm improves. This suggests that it is quite safe to use the iterative algorithm instead of the integrated algorithm.

Table 5 gives some more detailed results on the problem instances consisting of either 50 or 100 LRUs. Before we discuss the differences that result from the difference in number of

Table 5 Detailed results on (multi-echelon, multi-indenture) problem instances consisting of 50 or 100 LRUs

| | Number of LRUs | % of the demand for LRUs that is repaired | | | | % spare LRUs that is located | | | |
| | | at echelon level | | | in | at echelon level | | | in |
		1	2	3	total	1	2	3	total
Sequential	50	90.4 %	0.0 %	3.6 %	94.0 %	97.7 %	0.7 %	1.6 %	100 %
	100	93.4 %	0.0 %	3.3 %	96.7 %	99.0 %	0.2 %	0.7 %	100 %
Iterative	50	82.6 %	3.2 %	8.0 %	93.8 %	94.3 %	2.6 %	3.1 %	100 %
	100	90.2 %	2.7 %	3.7 %	96.6 %	97.8 %	1.2 %	1.0 %	100 %

Notice: Some problem instances consist of two echelon levels only (echelon levels 1 and 3)

LRUs, we first discuss how the repair strategies change when using the iterative algorithm instead of the sequential approach.

Notice that in the sequential solution, repairs are never performed at the intermediate depots (in test set 1). The reason is that resources are never located at the intermediate depot, probably because they are so expensive that they are interesting only at the central depot. As a result, repairs can be performed at the operating site if no resources are required (why pay more to ship them to the intermediate depot?) or at the central depot if resources are required and available there. Using the iterative algorithm, the numbers change, due to two reasons:

1. More resources are located at the central depot (26% and 7% more for 50 and 100 LRUs, respectively; not shown in a table), which means that some components that are discarded in the sequential solution, are now repaired. As a result, the lead time decreases for those components and less spare parts are required.
2. If repairs of a certain component are performed at the operating sites, then spare components may only be located at the operating sites as well. As a result, some components that do not require any resource in order to be repaired, are repaired at a more central location in the solution of the integrated solution so that risk pooling effects may be used (a spare part can now be located at a more central location and be used at various operating sites).

Reason 1 above partly explains the difference in achieved cost reduction for the problem instances with 50 and 100 LRUs (see Table 4): we do not vary the number of resources in our problem instances, which means that in problem instances with 50 LRUs a higher percentage of LRUs requires a resource in order to be repaired than in problem instances with 100 LRUs. As a result, there is more to be gained (relatively) when there are 50 LRUs only.

Next, we notice that if we increase the target availability for the problem instances with 50 LRUs to 97.5 % (not shown in a table), the achieved cost reduction reduces to about 2 %. A target availability of 97.5 % for 50 LRUs leads to a target availability per LRU of 99.95 % ($= 1 - (1 - 0.975)^{1/50}$). This is almost equal to the target availability per LRU in problem instances consisting of 100 LRUs and having a target availability of 95 %: 99.95 % ($= 1 - (1 - 0.95)^{1/100}$). This means that if the target availability per LRU increases, the potential cost reduction decreases. This also partly explains why the cost reduction in problem instances with 50 LRUs is higher than in the problem instances with 100 LRUs.

We then look at the achieved cost reductions for two values of the move lead time (see Table 4). A relatively low move lead time (compared with the repair and discard lead times)

means that on average the sequential approach leaves a lot of room for improvement for the iterative (and integrated) algorithm. The reason is that the total lead time (repair lead time plus move lead time) when repairing at a higher echelon level is only slightly higher than the repair lead time when repairing at the operating sites. As a result, the disadvantage of repairing at a higher echelon level is relatively small, and the advantage of being able to use risk pooling effects outweighs more often that disadvantage.

If we finally look at the cost reductions that may be achieved for the various values of the demand per LRU (see Table 4), we see that it is lowest for our second lowest setting ([0.01; 0.25]); it is higher when the demand is either lower or higher. It appears that multiple effects (e.g., target availability and demand per LRU) interact, as a result of which there is sometimes a lot to be gained from solving the two problems jointly instead of sequentially, and sometimes not. We are not able to state beforehand which of the two cases will happen.

6 Case study at Thales Nederland

We perform a case study on a sensor system (combined radar and electro-optical surveillance system) manufactured by Thales Nederland. The goal of this study is to find out which cost reduction we may obtain in practice and which advantages and drawbacks of our joint approach we can identify for application in practice. Thales Nederland is part of the Thales Group, which is a high-technology company active in aerospace, space, defense, security, and transportation. Thales Nederland is a manufacturer of naval sensors and naval command and control systems. Since Thales Nederland is active in the defense industry, it is a perfect company for a case study because both the LORA problem and the spare parts stocking problem have been well known in the military world for decades. Thales' customers include many navies, e.g., the Royal Netherlands Navy. If such a navy acquires a set of sensor systems, it also demands a plan on how to maintain the systems, which includes a LORA and a recommended spares list. Although Thales has to supply this plan, it should be optimized for the navy.

In Sect. 6.1, we discuss how a logistic engineer at Thales Nederland solves the LORA and spare parts stocking problems, and the associated difficulties. We give the technical details of the case study in Sect. 6.2, and in Sect. 6.3, we compare the results of the iterative algorithms with those of the sequential and integrated algorithms, and those of the logistic engineer.

6.1 Current practice

The logistic engineer at Thales Nederland first conducts a so-called *non-economic LORA*. The goal of this non-economic LORA is to exclude unrealistic repair or discard options and to simplify the problem. Questions that are posed are, for example:

– Is the component prone to failure? For example, casings do not usually fail under normal circumstances and are therefore not considered in the LORA.
– Does the customer prescribe the maintenance policy for the component? If so, this policy is followed.
– Does the value of the component exceed a certain threshold? If not, it can be discarded by default, since it is not worth repairing.

The result is that for some components, repair/discard options are excluded. If one option only remains, no decision needs to be taken in the LORA problem for that component, but

the component is still taken into account in the spare parts stocking problem, because of its influence on the availability. Furthermore, only part of the resources are included in the LORA, mainly the expensive ones.

After the input data has been structured and filtered, the logistic engineer finds a first solution, using decisions made for previous products, his experience, and spreadsheet calculations. Then, he uses a spare parts stocking tool (INVENTRI, based on VARI-METRIC and the work of Rustenburg 2000) to stock spare parts. Analyzing the results, he finds components for which holding costs are very high. If he thinks that it might help to change the LORA decision for these components, he does so and calculates the new LORA costs and solves a spare parts stocking problem. So in fact, the logistic engineer tries to perform a few manual iterations (less than ten), which has as its drawbacks that it is:

– time consuming, since such an analysis takes up to a few days after all data has been acquired;
– hard, if not impossible, to replicate, because of the judgmental feedback loop;
– error sensitive, since the engineer may easily overlook an opportunity for cost reduction.

Usage of the iterative (or integrated) algorithm would take away these drawbacks.

6.2 Case: a sensor system

Although the actual product structure of the sensor system consists of six indenture levels, we consider only three indenture levels, as a result of the non-economic LORA (see Sect. 6.1). For the same reason, although the product structure consists of over 1,500 components, only slightly more than 200 turn out to be relevant, of which 40 % are LRUs. For about one third of the components, only one repair/discard option remains, and for an additional one third, the repair/discard options that can be chosen are restricted. The repair network consists of twelve ships, attached to two intermediate depots, a central depot and Thales Nederland, the OEM (spare parts may not be stocked at the OEM and if repairs are performed at the OEM, then the variable repair costs per repair action are higher, but an investment in resources is not required for the navy). There are 54 resources.

The costs of the various components can be up to one million euros, and the costs of the various resources can be up to a couple of million euros. These costs are not used directly. Instead, there are three types of costs in the joint problem of LORA and spare parts stocking: variable costs per repair or discard action, fixed annual costs for locating resources, and annual spare parts holding costs. For each type of costs, we include the most important cost factors:

– Variable repair costs (customer's network): working hours (e.g., locating failure, exchanging subcomponents, and performing direct repair), variable costs for using resources (e.g., energy consumption and wear), and usage of additional parts (e.g., bulk items such as screws and wires).
– Variable repair costs (OEM and outsourced in general): listed repair price.
– Variable discard costs: procurement price for the component that replaces the discarded component and disposal costs or a residual value of the discarded component.
– Variable move costs: transportation, handling, and administrative costs.
– Fixed resource costs: depreciation costs, costs of capital, a risk factor (e.g., insurance against damage and theft), fixed operating costs (e.g., a location to operate the equipment), and maintenance costs of the resource. Resources may have a residual value after their economic lifetime.
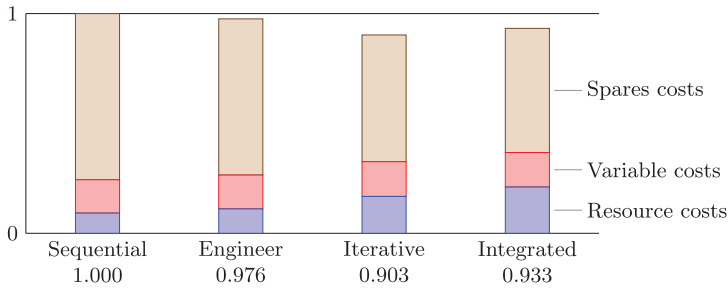
Fig. 3 Costs for Thales case (normalized)
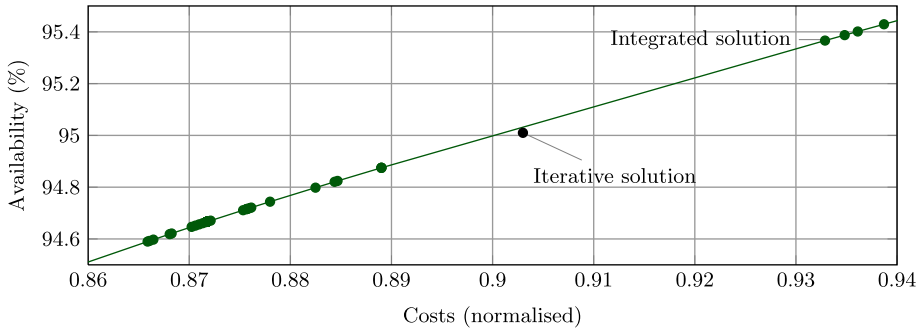
Fig. 4 Availability as a function of total costs for the case study

– Spare parts holding costs: costs of capital, a risk factor, and storage costs. Spares may have a residual value after the lifetime of the product.

The case study is solved for a target expected availability of 95% per ship.

6.3 Results

Figure 3 shows the results for the case study. We see that the iterative algorithm finds the best solution with a cost reduction of 9.7 %. The reason that the solution of the integrated algorithm leads to higher costs is the overshoot problem, as can be seen in Fig. 4. The figure shows the curve of availability versus costs that results from applying marginal analysis in the integrated algorithm, plus the solution that results from applying the iterative algorithm. We see that the latter solution lies below the curve, so that it is not an efficient point. Still, it is much closer to the target availability than the solution of the integrated algorithm, so that the resulting costs are much lower than those resulting from applying the integrated algorithm. We also see that some parts of the curve are quite dense, whereas others are not. This means that if we would have aimed for an availability of, for example, 94.8 %, the overshoot of the integrated algorithm would be very low.

The iterative algorithm requires less than one minute (11 iterations), whereas the integrated algorithm requires almost two days due to its enumerative approach. This means that for usage at Thales Nederland, the iterative algorithm fits best.

The cost reductions are achieved as follows. More resources are installed and more repairs are performed in the customer's network and in total. This leads to higher resource

costs and higher variable costs, but also to much lower holding costs. The cost reduction is achieved by:

– installing two resources at the depot that are not installed in the sequential solution;
– installing one resource at both intermediate depots instead of one at the central depot;
– installing one resource at all ships instead of one at each of the two intermediate depots.

The other resources are installed at the same echelon level in both solutions. The logistic engineer at Thales Nederland achieves about a quarter of the cost reduction that we achieve using the iterative algorithm: his solution is a combination of our iterative and sequential solution.

7 Conclusions and further research

In this paper, we presented an iterative algorithm for the joint problem of LORA and spare parts stocking for multi-indenture, multi-echelon problem instances with very mild restrictions on the resource-component relations.

We conclude that the iterative algorithm performs very well on average, and compared with the integrated algorithm, we observe cost differences of a few percent only in rare cases. This holds both for the approximate and exact version of the integrated algorithm since the difference between them is very small. We further conclude that the iterative algorithm scales very well; computation time is not a problem, whereas it is a huge problem for the integrated algorithm. This means that the iterative algorithm can be used in practice and it leads to a substantial cost reduction compared to solving the two problems sequentially. As a result, the principles behind our algorithm have been adopted by Thales Nederland.

The iterative algorithm can easily be extended if extensions do not affect the feedback mechanism. Examples of this are certain flexibility options in the spare parts stocking analysis, e.g., lateral transshipments or emergency shipments, or introduction of a probability of unsuccessful repair. Such extensions may be interesting from a business point of view, but probably not from an academical point of view. Our recommendations for further research are as follows.

First, the model may be extended so that the exact repair network can be modeled. The current model, in which completely symmetrical networks are assumed, can easily be extended such that we only require the same number of echelon levels in every part of the network. In other words, locations may be connected only to locations that are at the next higher or next lower echelon level. For instance, an operating site (echelon level 1) may not be connected directly to a central depot (echelon level 3). In this extended model, the LORA decisions should still be the same at all locations at one echelon level for each component and resource, but the spare parts stocking decisions may differ. This is often sufficient in practice. For example, in a naval environment, it is convenient that at each ship the same repair/discard decisions are taken, even if the demand rates differ due to their different mission profiles.

Allowing for different LORA decisions in one network or allowing for completely asymmetrical networks is more difficult. The key problem is how to decompose the holding costs that result from the spare parts stocking problem so that they can be fed back to the LORA. The holding costs for spares that are stocked at the central depot should be allocated to decisions made for failures originating at multiple operating sites. It may be possible to do this based on the failure rate at each operating site. The performance of the iterative heuristic will probably decline, since an additional approximation has to be introduced in the feedback mechanism.

Second, finite repair capacities may be introduced in the model. This is already difficult for the spare parts stocking problem alone, but there is some literature available (see, e.g., Sleptchenko et al. 2002). The feedback mechanism changes since the holding costs will not be fed back to one possible repair/discard decision, but to a possible repair/discard decision including a number of resources (in case of repair). However, we do not expect too much problems with this change in the feedback mechanism.

With all possible extensions, it will be hard to compare the iterative algorithm with the integrated algorithm, since the computation time of the latter algorithm will explode.

Appendix A: Termination of the iterative algorithm

We prove that the iterative algorithm necessarily converges in a finite number of iterations. We do so by showing that the number of possible solutions is finite and that it is not possible to have a cycle (a number of iterations after which we find exactly the same solution). In our proof we assume that the discard option is available at each echelon level to decrease the notational complexity. The proof is straightforwardly adapted if discard is available at the central depot only.

Lemma 1 *The number of possible solutions is finite.*

Proof If there are $|E|$ echelon levels, then each resource can be located at the various echelon levels in $2^{|E|}$ ways. Since there are $|R|$ resources, the total number of combinations for the resource locations is $2^{|E||R|}$.

Each component may be repaired or discarded at any of the $|E|$ echelon levels, so there are $2 \cdot |E|$ repair/discard options per component. Since there are $|C|$ components, there are $(2 \cdot |E|)^{|C|}$ options in total. In fact, there are generally less combinations, since some repair/discard options are unavailable for a subcomponent given the repair/discard decision for its parent component.

Combining the options for the resources and the components shows that there are at most $2^{|E||R|} \cdot (2 \cdot |E|)^{|C|}$ possible solutions for the LORA problem. Due to the exact allocation of resources to echelon levels, some repair options may not be available for certain components.

Since a given solution for the LORA problem leads to one specific allocation of spare parts in the spare parts stocking problem, there are also at most $2^{|E||R|} \cdot (2 \cdot |E|)^{|C|}$ solutions for the joint problem.   □

We define $\mathcal{H}^i$ as the three-dimensional array with entries $h^i_{c,e,d}$ in iteration i.

Corollary 1 $\mathcal{H}^i$ *can be filled in at most* $(|C| \cdot |E| \cdot 2)^{1+2^{|E||R|} \cdot (2 \cdot |E|)^{|C|}}$ *ways.*

Proof As a result of Lemma [1], each entry $h^i_{c,e,d}$ in $\mathcal{H}^i$ can be filled in at most $2^{|E||R|}$. $(2 \cdot |E|)^{|C|}$ ways in each iteration i ($i > 1$). In addition, each $h^1_{c,e,d} = 0$. Since there are $|C| \cdot |E| \cdot 2$ values to be filled, $\mathcal{H}^i$ can be filled in at most $(|C| \cdot |E| \cdot 2)^{1 + 2^{|E||R|} \cdot (2 \cdot |E|)^{|C|}}$ ways. $\qquad\square$

We now show that cycling cannot occur, which is formalized in Lemma [2]. In iteration i, an estimate of the spare parts holding costs is included in the inputs for the LORA problem ($\mathcal{H}^i$). The iteration starts with solving the LORA problem, leading to a certain LORA solution LS^i (the repair/discard decisions and resource locations) with resulting LORA costs LC^i (variable and fixed costs) and resulting estimated holding costs EC^i. Next, the spare parts stocking problem is solved. The total costs of the solution in iteration i are now the LORA costs (excluding the estimated holding costs) LC^i plus the holding costs resulting from the spare parts stocking problem SC^i. In iteration $i + 1$, the estimates of the holding costs in the LORA inputs are adapted again (to $\mathcal{H}^{i+1}$) and a new LORA solution is found LS^{i+1}, with associated costs $LC^{i+1} + EC^{i+1}$. The proof of Lemma [2] states that cycling can occur only if $LC^{i+1} + EC^{i+1}$ is equal to $LC^i + SC^i$. In other words, the solution that is found in iteration $i + 1$ may be different from the solution that is found in iteration i, but they lead to the same costs using the LORA inputs in iteration $i + 1$ (including $\mathcal{H}^{i+1}$). Therefore, we could just as well pick solution LS^i again and terminate the algorithm, which is what we describe in Sect. [4].

Lemma 2 *The iterative algorithm cannot cycle between a number of realizations of $\mathcal{H}^i$.*

Proof We define $X^i_{c,e,d}$, $Y^i_{r,e}$, and $S^i_{c,e}$ as the realization in iteration i of $X_{c,e,d}$, $Y_{r,e}$, and $S_{c,e}$, respectively. Then, we define $LC^i = \sum_{c \in C} \sum_{e \in E} \sum_{d \in D_e} v_{c,e,d} \cdot \lambda_c \cdot X^i_{c,e,d} + \sum_{r \in R} \sum_{e \in E} f_{r,e} \cdot Y^i_{r,e}$, $EC^i = \sum_{c \in C} \sum_{e \in E} \sum_{d \in \{discard, repair\}} h^i_{c,e,d} \cdot \lambda_c \cdot X^i_{c,e,d}$, and $SC^i = \sum_{c \in C} \sum_{f \in E} h'_{c,f} \cdot S^i_{c,f}$. EC^i represents the estimated holding costs in the LORA solution in iteration i, whereas SC^i represents the actual holding costs resulting from solving the spare parts stocking problem in iteration i. Remember from Sect. [4] that for each tuple (c, e, d) (with $d \in \{repair, discard\}$) for which $X^{i-1}_{c,e,d} = 1$, we set $h^i_{c,e,d} = \frac{\sum_{f \in E} h'_{c,f} \cdot S^{i-1}_{c,f}}{\lambda_c}$. Therefore $SC^i = \sum_{c \in C} \sum_{f \in E} h'_{c,f} \cdot S^i_{c,f} = \sum_{c \in C} \sum_{e \in E} \sum_{d \in \{discard, repair\}} \lambda_c \cdot h^{i+1}_{c,e,d} \cdot X^i_{c,e,d}$. Finally, we define Δ^i as the difference between SC^i and EC^i, so $\Delta^i = SC^i - EC^i = \sum_{c \in C} \sum_{e \in E} \sum_{d \in \{discard, repair\}} (h^{i+1}_{c,e,d} - h^i_{c,e,d}) \cdot \lambda_c \cdot X^i_{c,e,d}$.

If $LC^{i+1} + EC^{i+1} = LC^i + SC^i$, we know that $LS^{i+1} = LS^i$ and the algorithm terminates. So, in a cycle of length n ($n > 1$) it should hold that $LC^{i+1} + EC^{i+1} < LC^i + SC^i = LC^i + EC^i + \Delta^i$. For iteration $i + 2$ it then holds that $LC^{i+2} + EC^{i+2} < LC^{i+1} + EC^{i+1} + \Delta^{i+1}$, so $LC^{i+2} + EC^{i+2} < LC^{i+1} + EC^{i+1} + \Delta^{i+1} < LC^i + EC^i + \Delta^i + \Delta^{i+1}$. For the nth iteration, we get $LC^{i+n} + EC^{i+n} < LC^i + EC^i + \sum_{k=0}^{n-1} \Delta^{i+k}$. In a cycle of length n ($n > 1$), we also know that $LC^{i+n} = LC^i$ and $EC^{i+n} = EC^i$, so it should hold that $\sum_{k=0}^{n-1} \Delta^{i+k} > 0$. However, $\sum_{k=0}^{n-1} \Delta^{i+k} = \sum_{k=0}^{n-1} \sum_{c \in C} \sum_{e \in E} \sum_{d \in \{discard, repair\}} (h^{i+k+1}_{c,e,d} - h^{i+k}_{c,e,d}) \cdot \lambda_c \cdot X^{i+k}_{c,e,d} = \sum_{c \in C} \sum_{e \in E} \sum_{d \in \{discard, repair\}} \sum_{k=0}^{n-1} (h^{i+k+1}_{c,e,d} - h^{i+k}_{c,e,d}) \cdot \lambda_c \cdot X^{i+k}_{c,e,d}$. We know that if $X^{i+k}_{c,e,d} = 0$, then $h^{i+k+1}_{c,e,d} = h^{i+k}_{c,e,d}$. We also know that $h^{i+n}_{c,e,d} = h^i_{c,e,d}$. It is now easily seen that $\sum_{k=0}^{n-1} (h^{i+k+1}_{c,e,d} - h^{i+k}_{c,e,d}) \cdot \lambda_c \cdot X^{i+k}_{c,e,d} = 0$ for all $c \in C$, $e \in E$, $d \in \{discard, repair\}$ and therefore $\sum_{k=0}^{n-1} \Delta^{i+k} = 0$. We have shown that a cycle cannot occur if we only pick another LORA solution in iteration $i + 1$ if that leads to a strictly better result than using the LORA solution of iteration i again. Therefore, the iterative algorithm cannot cycle. $\qquad\square$

Table 6 Deterministic values in the multi-echelon, multi-indenture test sets[a]

Parameter	Default value	Additional value(s)	Varied in test set
Network structure	$1 \times 2 \times 10$	1×5	1
Product structure	$50 \times 100 \times 200$	50×100 $100 \times 200 \times 400$ 100×200	1
# Resources	10	–	–
# Component types	4	3	3
% Resources used by 1 component	50 %	0 %	3

[a]For example, the product structure is fixed to the default one (50 LRUs, 100 SRUs, and 200 parts) in all problem instances in test sets 2 and 3, whereas in test set 1 it is varied; four different product structures are used, including the default one and including two two-indenture product structures

Theorem 1 *The iterative algorithm always terminates after a finite number of iterations.*

Proof This holds since the number of possible matrices $\mathcal{H}^i$ is finite, as stated in Corollary 1, and the algorithm cannot cycle between a number of realizations of $\mathcal{H}^i$, as stated in Lemma 2.   □

Appendix B: Problem instances generator

In this appendix, we supply the details of how we generate the problem instances that we use in our numerical experiment. Since the generator that we use to generate the multi-echelon, multi-indenture problem instances is an extended version of the generator that we use to generate the two-echelon, single-indenture problem instances, we find it more convenient to discuss the former generator first. For the latter generator it then suffices to show the parameter values only.

B.1 Multi-echelon, multi-indenture problem instances

We use three sets of problem instances, each having its own focus:

1. Varying the problem size, the holding costs, and the lead times.
2. Varying the attractiveness of acquiring resources by changing the annual demand rate and the costs of resources and components (resulting in different variable repair, discard, and move costs).
3. Varying the component-resource relations.

Tables 6 and 7 give the exact parameter settings that we use in each of the test sets; the meaning of the parameters will be explained below. If we give a range for a parameter (Table 7), we randomly draw values from the given range. These values are the same for all settings of the other parameters. We use a full factorial design for each set: we test each possible combination of parameter settings. For each combination of parameter settings, we generate ten problem instances, in order to obtain a variety of problem instances. In total, sets 1, 2, and 3 consists of 1,280, 160, and 80 problem instances, respectively. Sets 2 and 3 are kept smaller than set 1, since the problem instances in the former sets require on average

Table 7 Stochastic values in the multi-echelon, multi-indenture test sets[a]

Parameter	Default range	Additional range(s)	Varied in test set
Annual demand per LRU	[0.01; 0.25]	[0.01; 0.10] [0.01; 0.50] [0.01; 1.00]	2
Net cost of component	[1,000; 10,000]	[1,000; 100,000]	2
Discard costs	[0.75; 1.25]	–	–
Repair costs	[0.25; 0.75]	–	–
Move costs	[0.01; 0.01]	–	–
Annual holding costs	[0.20; 0.20]	[0.20; 0.40]	1
Annual cost of resource	[10,000; 100,000]	[10,000; 500,000]	2
Discard lead time (in years)	[1/10; 1/2]	[1/4; 1/2]	1
Repair lead time (in years)	[0.5/52; 4/52]	[2/52; 4/52]	1
Move lead time (in years)	[0.5/52; 4/52]	[2/52; 4/52]	1
# Components per resource	[2; 6]	[2; 3]	3

[a]For example, the annual demand per LRU is in the default range ([0.01; 0.25]) for all problem instances in test sets 1 and 3, whereas in test set 2 it is varied; four different ranges are used, including the default one

about ten times as much computation time as those in set 1. A full factorial design over all three test sets would have led to a set that is too large (163,840 problem instances). Each problem is solved using a target availability of 95 %.

Since each parameter setting has a default value (or range) that is the same in each test set, there are ten problem instances that are part of each set. The parameter settings for these ten problem instances are used below to explain how we generate problem instances. In the explanation below, we do not mention which values are varied; this is shown in the tables.

The (symmetrical) repair network consists of a central depot, two intermediate depots, and ten operating sites. The discard lead time (in years) is drawn from a (continuous) uniform distribution in the range [1/10; 1/2] and differs per component. The repair lead time is in the range [0.5/52; 4/52] and differs per component, but is equal for all echelon levels. The move lead time is in the range [0.5/52; 4/52] and differs per echelon level, but is equal for all components. For a definition of the lead times, see Sect. 3.1.

The three-indenture product structure consists of 50 LRUs, 100 SRUs, and 200 parts. Each subcomponent is randomly assigned to one of the components at the next lower indenture level. As a result, in general, the number of subcomponents per component differs for the various components. The annual demand for a component is equal to the annual demand of its subcomponents (if any exist). We achieve this by drawing the annual demand of each part from a uniform distribution on the interval $[0.01/(\text{\#subcomp. per comp.})^2; 0.25/(\text{\# subcomp. per comp.})^2]$, and recursively calculating the annual demand of the SRUs and LRUs. The demand for SRUs or LRUs without subcomponents is drawn from the same interval as the demand for parts.

For each component, we draw a net price from a shifted exponential distribution with shift factor 1,000 and rate parameter $7/(10,000 - 1,000)$. As a result, we do not have components with a price below 1,000, since they are typically discarded by default. Furthermore, there are considerably more cheap components than expensive ones. On average 1‰ of the components get a value larger than 10,000, but we draw a new price for these components

Table 8 Deterministic values in the two-echelon, single-indenture test set

Parameter	Value(s)
Network structure	1×5
# LRUs	100
# Resources	10

Table 9 Stochastic values in the two-echelon, single-indenture test set[a]

Parameter	Range(s)
Annual demand per LRU	[0.01; 0.10]
Net cost of component	[1,000; 10,000] & [1,000; 100,000]
Discard costs	[0.75; 1.25]
Repair costs	[0.25; 0.75]
Move costs	[0.01; 0.01]
Annual holding costs	[0.20; 0.20] & [0.20; 0.40]
Annual cost of resource	[10,000; 100,000] & [10,000; 500,000]
Discard lead time (in years)	[1/10; 1/2] & [1/4; 1/2]
Repair lead time (in years)	[0.5/52; 4/52] & [2/52; 4/52]
Move lead time (in years)	[2/365; 4/52] & [1/52; 4/52]
# Components per resource	[2; 3] & [2; 6]

[a]For example, the net cost of each component is in the range [1,000; 10,000] in half of the problem instances, and in the range [1,000; 100,000] in the other half of the problem instances

to avoid odd problem instances. The annual costs of the resources are drawn from a shifted exponential distribution with shift factor 10,000 and rate parameter $7/(100,000 - 10,000)$. To avoid odd problem instances, we draw a new value if we have drawn a value higher than 100,000.

Using these prices, we calculate the variable costs as follows:

– repair costs are a fraction of the net component price. This fraction is drawn from a uniform distribution in the range [0.25; 0.75] and differs per component;
– the price of each subcomponent is added to the price of its parent to get the gross component price of the parent;
– discard costs are a fraction of the gross component price. This fraction is in the range [0.75; 1.25] and includes the costs for acquiring a new component;
– move costs are 1 % ([0.01; 0.01]) of the gross component price;
– annual costs of holding one spare part of a component are 20 % ([0.20; 0.20]) of the gross component price. These holding costs cover, for example, interest, storage costs, insurance costs, and obsolescence costs.

There are ten resources and their annual costs are in the range [10,000; 100,000]. Those costs are generated analogous to how the net price of each component is generated. 5 resources (50 % of the total) are required by one component only, the other are required by a number of components that is drawn from a discrete uniform distribution in the range [2; 6]. We distinguish 4 'component types', for example electronic and mechanical components. Each resource and each LRU family (an LRU including all its subcomponents at any indenture level) is randomly assigned to one of the component types, so that resources of one component type do not interact with resources of another component type, which is realistic in practice.

B.2 Two-echelon, single-indenture problem instances

To generate a set of two-echelon, single-indenture problem instances, we use the generator that Basten et al. (2012) use. Since the generator that we have described in Appendix B.1 is an extended version of that generator, it suffices to only give the parameter settings here, in Tables 8 and 9. In total this set consists of $10 \cdot 2^7 = 1,280$ problem instances. Notice that this set is disjoint with the set described in Sect. B.1.

References

Alfredsson, P. (1997). Optimization of multi-echelon repairable item inventory systems with simultaneous location of repair facilities. *European Journal of Operational Research*, *99*, 584–595.

Barros, L. L. (1998). The optimization of repair decisions using life-cycle cost parameters. *IMA Journal of Mathematics Applied in Business and Industry*, *9*, 403–413.

Barros, L. L., & Riley, M. (2001). A combinatorial approach to level of repair analysis. *European Journal of Operational Research*, *129*(2), 242–251.

Basten, R. J. I. (2010). *Designing logistics support systems. Level of repair analysis and spare parts inventories*. PhD thesis, University of Twente, Enschede, The Netherlands.

Basten, R. J. I., Schutten, J. M. J., & van der Heijden, M. C. (2009). An efficient model formulation for level of repair analysis. *Annals of Operations Research*, *172*(1), 119–142.

Basten, R. J. I., van der Heijden, M. C., & Schutten, J. M. J. (2011a). A minimum cost flow model for level of repair analysis. *International Journal of Production Economics*, *133*(1), 233–242.

Basten, R. J. I., van der Heijden, M. C., & Schutten, J. M. J. (2011b). Practical extensions to a minimum cost flow model for level of repair analysis. *European Journal of Operational Research*, *211*(2), 333–342.

Basten, R. J. I., van der Heijden, M. C., & Schutten, J. M. J. (2012). Joint optimization of level of repair analysis and spare parts stocks. *European Journal of Operational Research*, *223*(3), 474–483. doi:10.1016/j.ejor.2012.05.045.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*(1), 238–252.

Brick, E. S., & Uchoa, E. (2009). A facility location and installation or resources model for level of repair analysis. *European Journal of Operational Research*, *192*(2), 479–486.

Graves, S. C. (1985). A multi-echelon inventory model for a repairable item with one-for-one replenishment. *Management Science*, *31*(10), 1247–1256.

Muckstadt, J. A. (1973). A model for a multi-item, multi-echelon, multi-indenture inventory system. *Management Science*, *20*(4), 472–481.

Muckstadt, J. A. (2005). *Analysis and algorithms for service parts supply chains*. New York: Springer.

Rustenburg, W. D. (2000). *A system approach to Budget-Constrained spare parts*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands.

Rustenburg, W. D., van Houtum, G. J., & Zijm, W. H. M. (2003). Exact and approximate analysis of multi-echelon, multi-indenture spare parts systems with commonality. In J. G. Shanthikumar, D. D. Yao, & W. H. M. Zijm (Eds.), *Stochastic modelling and optimization of manufacturing systems and supply chains* (pp. 143–176). Boston: Kluwer.

Saranga, H., & Dinesh Kumar, U. (2006). Optimization of aircraft maintenance/support infrastructure using genetic algorithms—level of repair analysis. *Annals of Operations Research*, *143*(1), 91–106.

Sherbrooke, C. C. (1968). METRIC: a multi-echelon technique for recoverable item control. *Operations Research*, *16*(1), 122–141.

Sherbrooke, C. C. (1986). VARI-METRIC: improved approximations for multi-indenture, multi-echelon availability models. *Operations Research*, *34*, 311–319.

Sherbrooke, C. C. (2004). *Optimal inventory modelling of systems. Multi-echelon techniques* (2nd ed.). Dordrecht: Kluwer.

Sleptchenko, A., van der Heijden, M. C., & van Harten, A. (2002). Effects of finite repair capacity in multi-echelon, multi-indenture service part supply systems. *International Journal of Production Economics*, *79*, 209–230.

van der Heijden, M. C., Alvarez, E. M., & Schutten, J. M. J. (2012). Inventory reduction in spare part networks by selective throughput time reduction. *International Journal of Production Economics*. doi:10.1016/j.ijpe.2012.03.020.