Level of Repair Analysis: A Generic Model

R.J.I. Basten*, J.M.J. Schutten, M.C. van der Heijden

University of Twente, School of Management and Governance,

Department of Operational Methods for Production and Logistics,

P.O. Box 217, 7500AE, Enschede, Netherlands

February 6, 2008

Abstract

Given a product design and a repair network, a level of repair analysis (LORA) determines for each component in the product (1) whether it should be discarded or repaired upon failure and (2) at which echelon in the repair network to do this. The objective of the LORA is to minimize the total (variable and fixed) costs. We propose an IP model that generalizes the existing models, based on cases that we have seen in practice. Analysis of our model reveals that the integrality constraints on a large number of binary variables can be relaxed without yielding a fractional solution. As a result, we are able to solve problem instances of a realistic size in a couple of seconds on average. Furthermore, we suggest some improvements to the LORA analysis in the current literature.

Keywords: Maintenance, Level of repair analysis, Integer programming

*Corresponding author. Tel.: +31 53 489 4443; fax: +31 53 489 2159
E-mail adresses: r.basten@utwente.nl, j.m.j.schutten@utwente.nl, m.c.vanderheijden@utwente.nl

Figure 1: A multi-indenture system

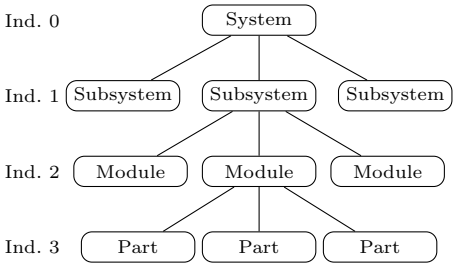
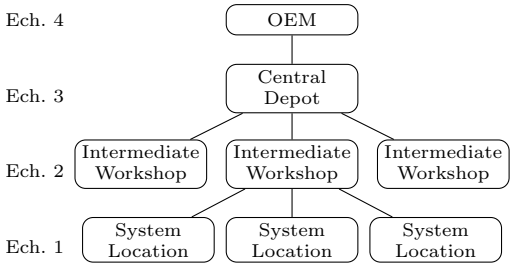


Figure 2: A multi-echelon repair network



1 Introduction

Every product that is manufactured, will one day fail. In the case of capital goods, such as military naval equipment, MRI-scanners, or trains, it will be cost effective to repair the product upon failure, instead of buying a new one. Customers know this, and they increasingly take total life cycle costs (LCC) into account in their purchasing decisions [1]. Sometimes, they even buy a service contract from the original equipment manufacturer (OEM). OEMs should be willing to provide service and sell service contracts, since selling services is generally more profitable than selling products [2, 3, 4]. Altogether, this means that more and more, the OEM should take the costs of maintenance into account when he makes decisions in the development process of a new product. In an early development stage, the product design can still be changed in order to lower the expected maintenance costs, and thus the expected life cycle costs of the system. In the later development stages, the actual maintenance plans need to be made, and a maintenance organization should be set up.

Generally, capital goods are repaired by replacement, which means that the component that failed, is removed from the system and replaced by a functioning one. A defective component can either be discarded (scrapped) or repaired. If it is discarded, a new component needs to be purchased. If the component is repaired, the subcomponent that failed will be replaced by a functioning one. The subcomponent should in turn be repaired or discarded itself. The system is thus seen as a multi-indenture system such as shown in Figure 1.

There is also the question of where to perform maintenance. If we consider military naval equipment, repairs can be performed on board the ship, at its marine base, a central depot, or even at the OEM. A network that connects all ships, bases, depots, and the OEM is called a repair network. Figure 2 shows an example of such a multi-echelon repair network.

Some questions related to maintenance received quite a lot of attention already: Much has been written on the subject of determining when to perform preventive maintenance, see for example Gertsbakh [5] or Dekker et al. [6]. The question of where to locate repair facilities also received quite some attention, see for example Daskin [7]. There is a vast amount of research on the spare part stock levels that are necessary to achieve a given availability of the user's products, given the product design and the repair network; see for example Sherbrooke [8] or Muckstadt [9]. However, a related problem did not receive much attention yet: The level of repair analysis (LORA). Given a product design and a repair network, a LORA determines for each component in the product (1) whether it should be discarded or repaired upon failure and (2) where to do that.

To be able to repair a system or component at a certain location, both variable and fixed costs have to be made. Costs that are variable in the number of failures that occur, include costs for working hours of service engineers, usage of spare parts, and transportation costs. Fixed costs include costs for (test) equipment and spare parts holding costs. The spare part inventory levels could simultaneously be included in the optimization, which would mean that spare parts holding costs are not seen as inputs anymore. However, this is generally not done in the LORA. The objective of the LORA is to minimize the total (variable and fixed) costs.

Barros [10] and Saranga and Dinesh Kumar [11] propose two models for the LORA, which are both integer programming (IP) models. However, we think that the requirements that these models pose to problem instances are so strict that, in general, these models cannot be used in practice. We base this assumption on cases that we have seen at Thales Nederland, a manufacturer of naval sensors and naval command and control systems. In this paper, we generalize the models. Section 2 discusses these two models, the requirements they pose to problem instances, and other related literature.

Section 3 presents our basic formulation of the LORA problem, which is also an IP model, and explains how it differs from the models in the literature. Section 3 also shows that, in general, removing the integrality constraints from the IP model yields a fractional solution in our basic model. Therefore, Section 4 provides an improved version of the model in which the integrality constraints on most of the variables can be removed, without yielding a fractional solution.[1] This positively influences the time it takes to solve our model.

This improved model is also used to show which integrality constraints can be removed if we use the model assumptions of the already existing LORA models. Section 4.3 shows that it is not possible to remove all integrality constraints in the model of Barros [10] (or in ours, using her model assumptions), although she claims this. Section 4.4 shows that if we use the model assumptions of Saranga and Dinesh Kumar [11] in our model, all integrality constraints can be removed. This means that a linear programming (LP) problem remains, which can be solved in polynomial time. Saranga and Dinesh Kumar [11] use genetic algorithms to solve problem instances.

Section 5 provides results for computational experiments. We based our tests on cases that we have seen at Thales Nederland and it turns out that problem instances of a realistic size can be solved by CPLEX in a reasonable amount of time. The paper ends with conclusions and directions for further research in Section 6.

2 Literature review

In the introduction, we already mentioned some research that is related to maintenance. We also mentioned that the LORA did not get a lot of attention. Although it is requested by, for example, both the United States Department of Defense [MIL-STD-1388-1A, 12][2] and the United Kingdom Ministry of Defence [DEF STAN 00-60 (PART 1), 14] that a contractor performs a LORA during the acquisition phase, only a few papers have been dedicated to

[1] In the remainder of the paper, if we say that 'integrality constraints can (cannot) be removed', this means that 'integrality constraints can (cannot) be removed without yielding a fractional solution'.

[2] MIL-STD-1388-1A contains requirements. It is superseded by MIL-HDBK-502 [13], which is for guidance only. This means that nowadays, LORA is not officially required anymore. However, it is usually still requested from contractors.

it [15, 10, 16, 17, 11]. It is even more surprising that LORA received so little attention in the past, since commercial life cycle cost (LCC) estimation tools generally contain a LORA part, see for example PRICE-HL [18] and EDCAS [19].[3]

Barros [10] presents an integer programming model for the LORA problem. She assumes that fixed costs (see Section 1) are borne by all the components at one indenture level. For example, if subsystem A should be repaired at echelon 2, fixed costs are taken into account. If subsystem B should be repaired at the same echelon, no additional fixed costs have to be taken into account. She also assumes that the discard option does not incur any fixed costs and that the variable costs for discard of a certain component are equal at every echelon.

Barros solves instances of the problem with two indenture levels and two echelons. Following Gutin et al. [17], we call this type of LORA instances LORA-BR. In her model, two echelons means that there are three repair options: 'discard', 'repair at echelon 1' and 'repair at echelon 2'. Barros states that her model can be used to solve problem instances with any number of indenture levels and echelons (disregarding the computation time, which could get very large). However, from Gutin et al. [17] we know that she only tested on the LORA-BR.[4] Her model needs some small modifications if it is to be used for more indenture levels and echelons.

According to Barros [10], her formulation of the LORA problem "...provides a natural integer solution in its relaxed linear programming version" (p. 409). Although we assume that this is true for the LORA-BR, we show in Section 4.3 that this is not true for the general case with more than two echelons or more than two indenture levels. Barros might have realized the same thing, since Barros and Riley [16] use a branch and bound method to solve the LORA problem.

Gutin et al. [17] reduce the LORA-BR problem to the maximum weight independent set problem on a bipartite graph, and show that the LORA-BR can be solved in polynomial

[3] Although it does not become clear from their websites that these tools contain a LORA part, we know this both from experts who have been using these tools and from the literature [e.g., 10].

[4] Gutin et al. know this from private communications with Barros.

time. They use the fact that the LORA-BR can be represented as a homomorphism problem on a monotone bipartite graph and they show that their result is valid for any monotone bipartite graph. However, the bipartite graph needed for a LORA problem with more than two indenture levels is not monotone. Since our model can be used for any number of indenture levels, their result is not applicable to our model.

Saranga and Dinesh Kumar [11] present a different integer programming model. The main difference between the models of Saranga and Dinesh Kumar [11] and Barros [10], is that the former assume that fixed costs are borne by a single component, whereas the latter assumes that they are borne by all the components at one indenture level. In our model we generalize this such that fixed costs can be borne by any arbitrary set of components, since we found this generalization was needed at Thales Nederland. The other difference lies in the costs for discard. Saranga and Dinesh Kumar (and we) assume that these can differ per echelon and that the fixed costs for discard need not be 0. Barros assumes that variable costs for discard are equal at every echelon and that fixed costs for discard are 0.

Saranga and Dinesh Kumar [11] solve their model using a genetic algorithm. As we already mentioned, we show in Section 4.4 that we can solve problem instances that fit their restrictions in polynomial time. This is due to the fact that in that case, we can remove all integrality constraints without yielding a fractional solution, which means that an LP problem remains.

Alfredsson [15] combines the LORA with the optimization of spare part stock levels under METRIC-like assumptions [20]. He models it as a fairly complicated IP model, which can solve problems with one indenture level. The model considers buying more than one tester of the same type if necessary. The waiting time for using a tester is calculated with an M/M/s-queue; only the mean demand rate is taken into account. Every component has its own tester and one multi-tester exists. This multi-tester can be used for one component and adapters can be added in a fixed order, in order to enable the multi-tester to be used for the repair of additional components. Furthermore, all items that can be repaired with the same multi-tester have to be repaired at the same location.

The model of Alfredsson [15] is already quite complicated, but does not take more than one indenture level into account and restricts the test equipment in a very strict way. Because our focus is on the LORA of a generic system (any number of indenture levels) and a generic repair network (any number of echelons), we decided not to base our model on the model of Alfredsson. Instead, our model is loosely based on the one of Saranga and Dinesh Kumar [11].

3 Basic IP model

This section provides our basic IP model. We give the model assumptions (Section 3.1), the notation we use (Section 3.2), and the model formulation (Section 3.3). Section 3.4 shows why the integrality constraints cannot be removed in this formulation without yielding a fractional solution. Therefore, Section 4 provides an improved version of the model in which most integrality constraints can be removed.

3.1 Model assumptions

Figure 1 (in Section 1) shows a typical multi-indenture system. In the remainder of this paper, we will use the names as given in the figure, so subsystems are at indenture level 1, modules are at indenture level 2, and parts are at indenture level 3. We use the term 'component' if the indenture level is irrelevant. In a general LORA, components and subcomponents are considered until a detail level that the user decides not to be relevant, which means that there can be any number of indenture levels. Figure 2 shows a multi-echelon repair network. In general, there can be any number of echelons in the repair network. The numbering of the echelons and indenture levels might be a bit confusing at first sight. However, it is used both in practice and in the literature [see, e.g., 8]. The logic is that the repair of a system starts by finding the subsystem (indenture level 1) that failed, then the module (indenture level 2), and so on. At the moment the system fails, it is at the system location (obviously), which is echelon 1. Components that failed can then

be moved upwards in the repair network to higher echelons.

A number of assumptions are generally made, both in the literature and, to the best of our knowledge, by companies developing and using commercial LORA-software. We also use these assumptions:

- Each time a repair is performed, variable costs are made. To be able to perform the repair of a certain component at a certain echelon, yearly fixed costs are made. For example, if fixed costs are related to acquiring test equipment, they represent the yearly depreciation costs.
- The system itself (indenture level 0) is never moved from its location, but is always repaired by replacing a subsystem. Therefore, indenture level 0 is not modeled.
- The repair of a component is in principal accomplished by replacing a subcomponent that failed, with a working one. A component is repaired directly if it is at the highest indenture level, since there are no subcomponents modeled at that level. It can also be that the failure in a certain component x is caused by a failure of component y_1 in 50% of the cases, a failure of component y_2 in 40% of the cases, and a failure that can be repaired directly in 10% of the cases. In this last 10% of the cases, no replacement of a subcomponent is needed.
- A failed component can be moved only from a certain echelon e to echelon $e + 1$.
- Combining the previous two assumptions means that if, for example, a subsystem is repaired at echelon 2, the failed module that was contained in the subsystem, can only be repaired at echelon $e \geq 2$.
- If the choice is made to repair a certain component at a certain echelon, the probability of a successful repair is 100%.
- All data at a certain echelon is aggregated. This means that the exact structure of the repair network is not known by the model.
- As a result of the previous assumption, the repair of a certain component x should always be performed at the same echelon e, independent of the system location from which the component x originates. This may be suboptimal in practice; for example, if the repair network is asymmetric.

3.2 Notation

As explained in the previous section, Saranga and Dinesh Kumar [11] assume that fixed costs are borne by one component, whereas Barros [10] assumes that fixed costs are borne by all the components at one indenture level. At Thales Nederland, these kind of sets are too restrictive; in our model, fixed costs are allocated to more general sets of components $\mathscr{G}_g \in \mathscr{G}$ ($\mathscr{G}_g \subseteq X$, $\mathscr{G}_g \neq \emptyset$, where X is the set of all components). Not all components need to be in one of the sets and components might be in more than one set. To show when this might happen, suppose there are three components: a blower, a power generator, and a transmitter. Suppose that a certain tester is needed to test the blower and the power generator. Another tester is needed to test the power generator and the transmitter. The power generator will then be in two sets.

If every set contains exactly one component and every component belongs to exactly one set ($|\mathscr{G}_g| = 1, |\mathscr{G}| = |X|, g_1 \neq g_2 \Rightarrow \mathscr{G}_{g_1} \neq \mathscr{G}_{g_2}$), fixed costs are incurred per component, as in the model of Saranga and Dinesh Kumar. If every set consists of all components at one indenture level, the assumptions used in Barros' model are used.

A component ('parent') can contain subcomponents ('children'). If a parent is at indenture level i, its children are at indenture level $i+1$. The set $\Gamma_x = \{y \mid y \text{ is a child of } x\}$ is used in our model to link parents to children. The set X_S ($\subseteq X$) is the set of subsystems, so the components at indenture level 1. These subsystems do not have any parent component in the model (remember that the complete system, indenture level 0, is not modeled).

Generally, the repair network consists of multiple echelons. These echelons form the set E. At each echelon $e \in E$, except for the highest, there are three repair options for each component $x \in X$:

- Discard: Component x is scrapped and a new one should be bought.
- Repair: Component x is repaired by replacing its defective child y with an operating one (or by repairing x directly). One of the repair options should be chosen for component y at the same echelon e.

- Move: The component is moved to echelon $e+1$. At echelon $e+1$, a repair option has to be chosen. Note that the 'move' option does not exist at the highest echelon.

Together, these repair options r make up the set R. The set of echelons without the 'highest' one, so the set of echelons with three repair options, is called E_L.

In practice, not all combinations of e, r, and x are possible. There might not be enough room on board a ship for certain test equipment, or proprietary knowledge can prohibit the user from repairing a component. A so-called non-economic LORA is performed to exclude the combinations that are not possible, after which the (economic) LORA as explained in this paper is performed. At Thales Nederland, more than half of the combinations may be excluded already after the non-economic LORA. Furthermore, the very small parts (screws, transistors, etc.) are excluded from consideration by the non-economic LORA: They are always discarded.

$vc_{e,r,x}$ are the variable costs per repair action of component x at echelon e for repair option r (discard, repair, or move). As mentioned before, variable costs include costs for working hours of service engineers , usage of spare parts, and transportation costs. $fc_{e,r,\mathscr{G}_g}$ are the fixed costs related to enabling at echelon e the repair option r for all components that are part of set $\mathscr{G}_g$. We will also call this 'opening option r at echelon e for set $\mathscr{G}_g$'. Fixed costs include spare parts holding costs and costs for (test) equipment. Notice that if a component x is part of both $\mathscr{G}_{g_1}$ and $\mathscr{G}_{g_2}$, fixed costs for option r at echelon e for both these sets need to be taken into account before that repair option can be chosen for component x.

λ_x is the yearly demand rate (number of failures) of component x. λ_x is an input for all $x \in X$. If $\sum_{y \in \Gamma_x} \lambda_y > \lambda_x$ for component x, this would mean that the children of x fail more often than x itself. This can only happen if multiple children fail at the same time. This is not a problem for the model. We already discussed that $\sum_{y \in \Gamma_x} \lambda_y$ can also be smaller than λ_x.

If a component $y \in \Gamma_{x_1}$ and $y \in \Gamma_{x_2}$ (commonality), we will treat y as two different components y_1 and y_2, with for all $\mathscr{G}_g$: $y_1 \in \mathscr{G}_g \iff y_2 \in \mathscr{G}_g$. This means that two

different repair options may be chosen for y_1 and y_2. If, for example, x_1 is repaired at echelon 1 and x_2 is repaired at echelon 2, it can be optimal to discard y_1 at echelon 1 and discard y_2 at echelon 2.

The model uses two types of decision variables:

$$N_{e,r,x} = \begin{cases} 1, \text{ if at echelon } e \in E \text{ repair option } r \in R \text{ is selected for component } x \in X \\ 0, \text{ otherwise} \end{cases}$$

$$M_{e,r,\mathscr{G}_g} = \begin{cases} 1, \text{ if at echelon } e \in E \text{ repair option } r \in R \text{ is selected for set } \mathscr{G}_g \in \mathscr{G} \\ 0, \text{ otherwise} \end{cases}$$

3.3 Model formulation

We propose the following model formulation:

$$\text{minimize} \sum_{e \in E} \sum_{r \in R} \sum_{x \in X} vc_{e,r,x} \cdot \lambda_x \cdot N_{e,r,x} + \sum_{e \in E} \sum_{r \in R} \sum_{\mathscr{G}_g \in \mathscr{G}} fc_{e,r,\mathscr{G}_g} \cdot M_{e,r,\mathscr{G}_g} \tag{1}$$

subject to:

$$\sum_{r \in R} N_{1,r,x} = 1 \ , \forall x \in X_S \tag{2}$$

$$N_{e,\text{move},x} \leq \sum_{r \in R} N_{e+1,r,x} \ , \forall e \in E_L \ , \forall x \in X \tag{3}$$

$$N_{e,\text{repair},x} \leq \sum_{r \in R} N_{e,r,y} \ , \forall e \in E \ , \forall x \in X \ , \forall y \in \Gamma_x \tag{4}$$

$$N_{e,r,x} \leq M_{e,r,\mathscr{G}_g} \ , \forall e \in E \ , \forall r \in R \ , \forall \mathscr{G}_g \in \mathscr{G} \ , \forall x \in \mathscr{G}_g \tag{5}$$

$$N_{e,r,x}, M_{e,r,\mathscr{G}_g} \in \{0,1\} \ , \forall e \in E \ , \forall r \in R \ , \forall x \in X \ , \forall \mathscr{G}_g \in \mathscr{G} \tag{6}$$

The objective function minimizes the sum of all yearly variable and fixed costs. Constraint 2 guarantees that a repair option is chosen for every subsystem on echelon 1. If 'move' is chosen for a component x on an echelon e, Constraint 3 assures that a repair option is chosen for component x on the next higher echelon $e+1$. Constraint 4 assures

that if 'repair' is chosen at an echelon for a component, a repair option is chosen for all its child components at that echelon.

The inequalities in both constraints cannot be changed to equalities. Assume that there is a component x with one child component y:

- If x is moved from echelon 1 to 2, where it is repaired, a repair option needs to be chosen for y at echelon 2. This means that $\sum_{r \in R} N_{2,r,y} = 1$. An equality in Constraint 3 would then imply that $N_{1,\text{move},y} = 1$, which is incorrect.
- If x is repaired at echelon 1 and y is moved to echelon 2, a repair option needs to be chosen for y at echelon 2. This means that $\sum_{r \in R} N_{2,r,y} = 1$. An equality in Constraint 4 would then imply that $N_{2,\text{repair},x} = 1$, which is incorrect.

If 'discard' is chosen for a component, no repair option has to be chosen for its children. The costs of discard include the costs of discard of the children. This is different from the model formulations of both Barros [10] and Saranga and Dinesh Kumar [11], in which choosing the discard option for a parent component means that discard should also be chosen for all its child components. We think that it is intuitively more logical that nothing needs to be done with the children if the parent is discarded.

Constraint 5 assures that fixed costs are taken into account for set $\mathscr{G}_g$, if a repair option is chosen for any component $x \in \mathscr{G}_g$.

3.4 LP relaxations

Our model contains two types of integer variables: $N_{e,r,x}$ and $M_{e,r,\mathscr{G}_g}$. In this section, we give a problem instance that shows that, in general, the integrality constraints on the $N_{e,r,x}$ variables cannot be removed without yielding a fractional solution.

It is possible to remove the integrality constraint on $M_{e,r,\mathscr{G}_g}$, since Constraint 5 assures that $M_{e,r,\mathscr{G}_g} = 1$ if any $N_{e,r,x} = 1$ with $x \in \mathscr{G}_g$. If $N_{e,r,x} = 0$ for all $x \in \mathscr{G}_g$, the minimization in the objective function will cause $M_{e,r,\mathscr{G}_g}$ to be 0.[5] However, we prefer to remove the

[5]Except when $fc_{e,r,\mathscr{G}_g} = 0$, but in that case, an optimal integer solution exists as well.

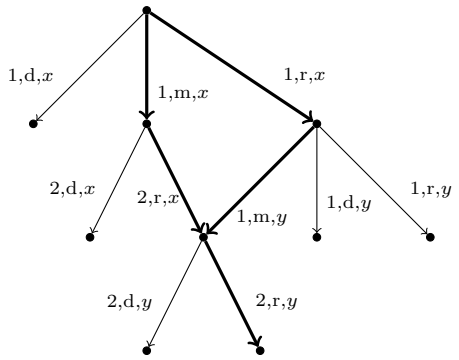
Table 1: Variable costs and yearly demand for erroneous instance

Component	x	y
$\lambda_{x/y}$	1	1
$vc_{1,\text{discard},x/y}$	10	10
$vc_{1,\text{repair},x/y}$	1	1
$vc_{1,\text{move},x/y}$	0	0
$vc_{2,\text{discard},x/y}$	10	10
$vc_{2,\text{repair},x/y}$	1	1

Table 2: Outputs for erroneous instance

Component	x	y
$N_{1,\text{discard},x/y}$	0	0
$N_{1,\text{repair},x/y}$	0.5	0
$N_{1,\text{move},x/y}$	0.5	0.5
$N_{2,\text{discard},x/y}$	0	0
$N_{2,\text{repair},x/y}$	0.5	0.5

Figure 3: Result of erroneous instance[a]



[a]Each arc in the graph represents a repair option $N_{e,r,x/y}$: d=discard, r=repair and m=move. The bold arcs represent the options that are selected in the example ($N_{e,r,x/y} = 0.5$).

integrality constraint on $N_{e,r,x}$ (which is possible for the model we give in Section 4), since there are generally more $N_{e,r,x}$ than $M_{e,r,\mathscr{G}_g}$ variables.

To see why the integrality constraint on the $N_{e,r,x}$ variables cannot be removed in our basic model, consider a system consisting of components x and y, with x being the parent of y. The repair system consists of 2 echelons, 1 and 2, and there are no fixed costs for opening a repair option. Table 1 shows the variable costs and the demand rates.

Table 2 shows the resulting optimal LP solution, which is not an IP solution. The objective value is 1.5 and is 2 for the optimal IP solution. To understand why the LP solution differs from the IP solution, and why the LP solution is not feasible, see Figure 3. The figure shows in a graph which repair options can be chosen for components x and y. Each displayed

arc represents a repair option $N_{e,r,x}$ or $N_{e,r,y}$. At the top node, only one arc or repair option should be chosen, so that the associated $N_{1,r,x} = 1$. If two options are chosen simultaneously, both associated $N_{1,r,x} = 0.5$. What happens in the example, is that via two ways[6], component y reaches echelon 2 in need for repair (the bold arcs in the figure). Because Constraint 3 and Constraint 4 ensure that the value of $N_{2,\text{repair},y}$ is greater than or equal to $N_{1,\text{move},y}(= 0.5)$ and $N_{2,\text{repair},x}(= 0.5)$ respectively, $N_{2,\text{repair},y}$ needs to be only 0.5 instead of 1. This means that half of the amount of y is lost. In Section 3.3, we explained that we cannot replace the inequalities in these constraints with equalities, so we cannot prevent the problem in this formulation without using the integrality constraints on the $N_{e,r,x}$ variables.

4 Improved IP model

Because the integrality constraints in the model provided in Section 3 could not be removed for the $N_{e,r,x}$ variables, we show an improved model in Section 4.1. We still use the assumptions outlined in Section 3.1. Section 4.2 shows which integrality constraints can be removed in the improved model and Section 4.3 uses these results to show which integrality constraints can be removed in the model of Barros [10]. Section 4.4 uses the results of Section 4.2 to show that all integrality constraints can be removed when the assumptions of Saranga and Dinesh Kumar [11] are used in our model.

4.1 Model formulation

The improvement in the LORA formulation is inspired by the problem shown in Section 3.4. We show the model below, and explain the differences with the basic model afterwards.

[6]The two ways are: (1) Component x is moved to echelon 2 $(1, \text{move}, x)$ and is repaired there $(2, \text{repair}, x)$. Component y results at echelon 2 in need for repair. (2) Component x is repaired at echelon 1 $(1, \text{repair}, x)$. Component y results at echelon 1 in need for repair and is moved to echelon 2 $(1, \text{move}, y)$.

$$\text{minimize} \sum_{e\in E}\sum_{r\in R}\sum_{x\in X} vc_{e,r,x}\cdot\lambda_x\cdot N_{e,r,x} + \sum_{e\in E}\sum_{r\in R}\sum_{\mathscr{G}_g\in\mathscr{G}} fc_{e,r,\mathscr{G}_g}\cdot M_{e,r,\mathscr{G}_g} \tag{7}$$

subject to:

$$\sum_{r\in R} N_{1,r,x} = 1 \ ,\forall x\in X_S \tag{8}$$

$$N_{e,\text{move},x} = \sum_{r\in R} N_{e+1,r,x} \ ,\forall e\in E_L \ ,\forall x\in X_S \tag{9}$$

$$N_{1,\text{repair},x} = \sum_{r\in R} N_{1,r,y} \ ,\forall x\in X \ ,\forall y\in\Gamma_x \tag{10}$$

$$N_{e+1,\text{repair},x} + N_{e,\text{move},y} = \sum_{r\in R} N_{e+1,r,y} \ ,\forall e\in E_L \ ,\forall x\in X \ ,\forall y\in\Gamma_x \tag{11}$$

$$N_{e,r,x} \le M_{e,r,\mathscr{G}_g} \ ,\forall e\in E \ ,\forall r\in R \ ,\forall\mathscr{G}_g\in\mathscr{G} \ ,\forall x\in\mathscr{G}_g \tag{12}$$

$$N_{e,r,x}, M_{e,r,\mathscr{G}_g} \in \{0,1\} \ ,\forall e\in E \ ,\forall r\in R \ ,\forall x\in X \ ,\forall\mathscr{G}_g\in\mathscr{G} \tag{13}$$

There are four differences with the original model:

- Constraint 9 is similar to Constraint 3, but is used only for the subsystems (X_S), instead of for all components.
- Constraint 10 is similar to Constraint 4, but is used only for echelon 1, instead of for all echelons.
- Constraint 11 is added to handle the problem shown in Section 3.4. It combines the previous two constraints in that it assures that a repair option is chosen for a child component if it is moved from a lower echelon, or the parent component is repaired at the current echelon.
- Constraints 3 and 4 are inequalities (and cannot be changed to equalities, see Section 3.3), but Constraints 9, 10 and 11 are equalities.

4.2 LP Relaxations

The model uses two types of integer variables: $N_{e,r,x}$ and $M_{e,r,\mathscr{G}_g}$. In this section, we show that we cannot remove the integrality constraint on both $N_{e,r,x}$ and $M_{e,r,\mathscr{G}_g}$, without

Table 3: Variable costs and yearly demand for erroneous instance

Component	x_1	x_2	x_3
λ_x	1	1	1
$vc_{1,\text{discard},x}$	100	0	0
$vc_{1,\text{repair},x}$	0	100	0
$vc_{1,\text{move},x}$	0	0	100
$vc_{2,\text{discard},x}$	0	0	0
$vc_{2,\text{repair},x}$	0	0	0

Table 4: Fixed costs for erroneous instance

Set	$\mathscr{G}_1$
$fc_{1,\text{discard},\mathscr{G}_g}$	100
$fc_{1,\text{repair},\mathscr{G}_g}$	100
$fc_{1,\text{move},\mathscr{G}_g}$	100
$fc_{2,\text{discard},\mathscr{G}_g}$	0
$fc_{2,\text{repair},\mathscr{G}_g}$	0

Table 5: $N_{e,r,x}$ for erroneous instance

Component	x_1	x_2	x_3
$N_{1,\text{discard},x}$	0	0.5	0.5
$N_{1,\text{repair},x}$	0.5	0	0.5
$N_{1,\text{move},x}$	0.5	0.5	0
$N_{2,\text{discard},x}$	0.5	0.5	0
$N_{2,\text{repair},x}$	0	0	0

Table 6: $M_{e,r,\mathscr{G}_g}$ for erroneous instance

Set	$\mathscr{G}_1$
$M_{1,\text{discard},\mathscr{G}_g}$	0.5
$M_{1,\text{repair},\mathscr{G}_g}$	0.5
$M_{1,\text{move},\mathscr{G}_g}$	0.5
$M_{2,\text{discard},\mathscr{G}_g}$	0.5
$M_{2,\text{repair},\mathscr{G}_g}$	0

yielding a fractional solution (Section 4.2.1). However, we also show that we can remove the integrality constraint on $N_{e,r,x}$ (Section 4.2.2).

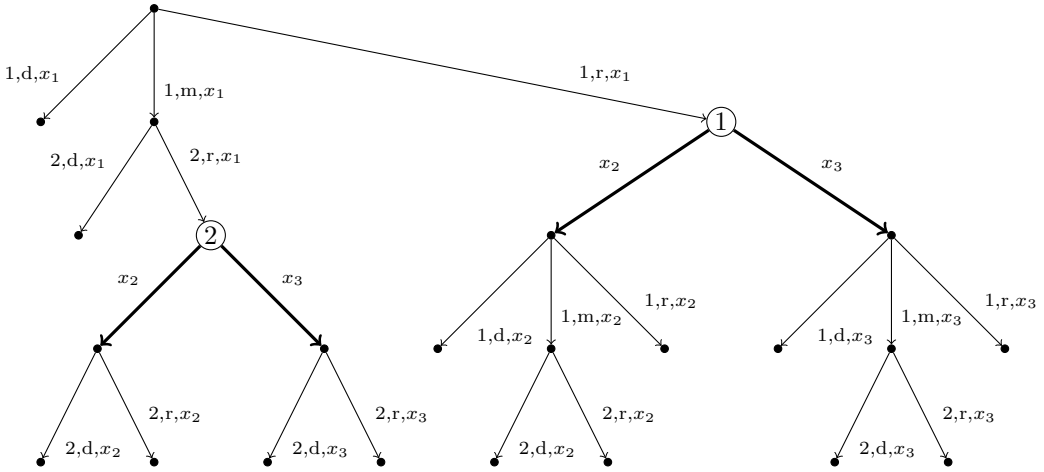
4.2.1 Removing all integrality constraints

In this section, we give an example of a LORA instance that leads to a non-integer solution (that cannot be adapted to an integer solution, while keeping the same objective function value). In the example, there are three subsystems (x_1, x_2, x_3) without child components. The subsystems are in one set that shares fixed costs, so $\mathscr{G}_1 = \{x_1, x_2, x_3\}$. Table 3 gives the yearly demand rate per component and the variable costs per repair action. Table 4 gives the fixed costs.

The optimal LP solution value for this instance is 150, but the optimal IP solution value is 200. Tables 5 and 6 show the values of $N_{e,r,x}$ and $M_{e,r,\mathscr{G}_g}$ in the optimal LP solution. The optimal IP solution can be achieved in multiple ways. Since repair and discard on echelon 2 do not incur any costs, we can focus on the three repair options at echelon 1:

- Opening one repair option leads to fixed costs of 100. Depending on the option

Figure 4: Decision tree: 3 components, 2 echelons, 2 indenture levels[a]



[a]Each non-bold arc in the graph represents a repair option $N_{e,r,x}$: d=discard, r=repair and m=move. The bold arcs show that below node 1 and 2, repair options need to be chosen simultaneously for x_2 and x_3.

we would open, one component would make variable costs of 100. Fixed costs and variable costs together would be 200.

- Opening two or more repair options leads to fixed costs of at least 200.

This example shows that, in general, not all integrality constraints can be removed. However, based on our experiments we conclude that only about 6% of the LORA problem instances leads to a non-integer solution if all integrality constraints are removed.

4.2.2 Removing integrality constraints on the $N_{e,r,x}$ variables

In this section, we discuss removing the integrality constraints on the $N_{e,r,x}$ variables and we consider the resulting optimal solution. We show that all $N_{e,r,x}$ variables will be integer.

The basic idea of our proof is that we take the costs of the optimal solution for all the children together, and add these to the parent component. Assume that we have a system consisting of a component x_1 with $\Gamma_{x_1} = \{x_2, x_3\}$, and we have a repair network with two echelons. Figure 4 shows the decision tree for the repair options of the system. If the repair option 'repair' is chosen for x_1 at either echelon 1 or 2, a repair option has to be

chosen for both x_2 and x_3, which is indicated by the bold arcs originating at node 1 and 2 respectively. We show below that the optimal repair options for the child components can be chosen independently of each other. In other words, the parts below nodes 1 and 2 can be solved independently. After that, these parts can be removed, and the optimal costs of these parts can be added to the cost of the arcs that end in nodes 1 and 2 (the options 'repair' at echelon 1 and 2 respectively).

We consider the optimal solution of the IP model in which the integrality constraints on the $N_{e,r,x}$ variables are removed. The $M_{e,r,\mathscr{G}_g}$ variables are still binary. Since we are looking at the optimal solution, it is fixed, for example, at which echelon test equipment is available and at which echelon it is not. This in turn means that not all repair options may be possible anymore: In Figure 4, not all arcs can be chosen.

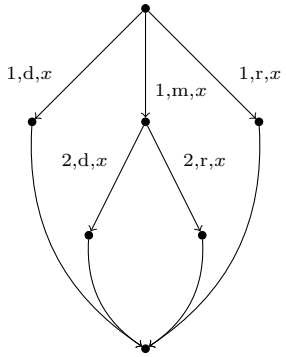
We need one further observation: Components at the same indenture level can only be connected to each other by their parent component (or a parent of a parent etc.) and by the sets of components sharing fixed costs ($\mathscr{G}_g$). It follows that given a repair decision for all the parents and given the values for $M_{e,r,\mathscr{G}_g} \in \{0,1\}$, decisions for components at the same indenture level can be made independently. The subsystems do not have a parent component modeled, so decisions for them can be made independently as well.

We are now ready to show that the repair decision for each component can be seen as a minimum cost network flow problem. (Refer to Figure 4 if necessary.) Figure 5 shows the network for component x_2, given that x_1 is repaired at echelon 1. If x_1 is repaired at echelon 2, the network for x_2 is shown in Figure 6. The capacity of an arc is 1 if the associated repair option is feasible. In other words, if $M_{e,r,\mathscr{G}_g} = 1 \ , \forall \mathscr{G}_g \in \mathscr{G} \mid x_2 \in \mathscr{G}_g$. The capacity is 0 otherwise. The costs of using an arc are equal to the associated variable costs times the associated yearly demand ($vc_{e,r,x_2} \cdot \lambda_{x_2}$).

The reasoning for component x_3 goes analogous to the reasoning for x_2 in the previous paragraph. What differs (probably) are the capacities of the arcs and the costs for using the arcs.

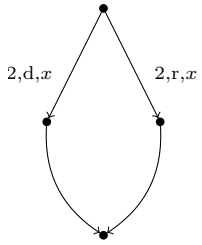
It is well known in the literature that minimum cost network flow problems always have

Figure 5: Graph: 2 echelons[a]



[a]Each arc in the graph represents a repair option $N_{e,r,x}$: d=discard, r=repair and m=move. An arc has capacity 1 if the associated repair option is feasible (given the values of $M_{e,r,\mathscr{G}_g}$). The capacity is 0 otherwise.

Figure 6: Graph: 1 echelon[a]



[a]Each arc in the graph represents a repair option $N_{e,r,x}$: d=discard, r=repair and m=move. An arc has capacity 1 if the associated repair option is feasible(given the values of $M_{e,r,\mathscr{G}_g}$). The capacity is 0 otherwise.

an optimal integer solution, provided that all capacities, supplies and demands are integer [see, e.g., 21]. Capacities are all 0 or 1 in our example. Supply at the top and demand at the sink (bottom vertex) is 1. It follows that all $N_{e,r,x_2}, N_{e,r,x_3} \in \{0,1\}$.

We add the sum of the costs of the best repair options of x_2 and x_3 at echelon 1 (the optimal solution for both x_2 and x_3, see Figure 5) to the costs of repairing x_1 at echelon 1 (the arc ending in node 1 in Figure 4). In the same way, we add the sum of the costs of the best repair options of x_2 and x_3 at echelon 2 (Figure 6) to the costs of repairing x_1 at echelon 2 (the arc ending in node 2 in Figure 4). The result is that for component x_1, we have the same network as shown in Figure 5. Since x_1 is a subsystem in our example, Constraint 8 assures an inflow of 1. With a reasoning analogous to the reasoning in the previous paragraph, this shows that all $N_{e,r,x_1} \in \{0,1\}$, and therefore all $N_{e,r,x} \in \{0,1\}$.

It may happen that some of the network flow problems for component x_2 or x_3 do not have a feasible solution (Figures 5 and 6). This happens if no path through the network has a capacity of more than 0, due to the values of the $M_{e,r,\mathscr{G}_g}$ variables. These networks originate in node 1 or 2 in Figure 4, which means that the arc ending in that node cannot be chosen in the optimal solution. This means that if we add the optimal values for x_2 and x_3 to the arc corresponding to N_{e,repair,x_1} (for e is 1 or 2), this arc gets a capacity

of 0. However, in the minimum cost network flow problem for each subsystem (x_1), it is still guaranteed that there is at least one path with capacity 1, since we are discussing the optimal solution.

The extension of our reasoning to more echelons, more indenture levels or more children per parent is straightforward. This means that in the general LORA problem, we can remove the integrality constraints on the $N_{e,r,x}$ variables and it is still guaranteed that there exists an optimal solution in which all $N_{e,r,x} \in \{0,1\}$, provided that $M_{e,r,\mathscr{G}_g} \in \{0,1\}$. If any $N_{e,r,x} \notin \{0,1\}$ in the resulting solution of our mixed integer programming model, we can construct an integer solution based on the reasoning above (however, we never encountered non-integer solutions in any of our tests).

4.3 Model of Barros

As we noted before, Barros [10] mentions that her formulation of the LORA problem "...provides a natural integer solution in its relaxed linear programming version" (p. 409). This section shows that this is not true for the general case with more than two echelons or more than two indenture levels. Although Barros states that her model can be used for any number of echelons and indenture levels, we know from Gutin et al. [17] that Barros tested her model for the case of two echelons and two indenture levels only (LORA-BR). We could not find a counter example for that specific case.

The example used in the current section, resembles the example given in Section 4.2.1. Table 7 shows the yearly demand rate and the variable costs, Table 8 shows the fixed costs. There are a couple of differences with the previous example, due to differences between our model and that of Barros:

- The echelon e is incorporated in the repair option r in Barros' model.
- Barros assumes that no fixed costs need to be made for opening the discard option.
- Barros assumes only one discard option, and not different variable costs for discard at every level in the repair network.
- Barros does not distinguish the move option. Costs for moving a component are part

Table 7: Variable costs and yearly demand for erroneous instance

Component	x_1	x_2	x_3
λ_x	1	1	1
$vc_{\text{discard},x}$	200	200	200
$vc_{\text{repair at 1},x}$	100	0	0
$vc_{\text{repair at 2},x}$	0	100	0
$vc_{\text{repair at 3},x}$	0	0	100

Table 8: Fixed costs for erroneous instance

Set	$\mathscr{G}_1$
$fc_{\text{discard},\mathscr{G}_i}$	0
$fc_{\text{repair at 1},\mathscr{G}_i}$	100
$fc_{\text{repair at 2},\mathscr{G}_i}$	100
$fc_{\text{repair at 3},\mathscr{G}_i}$	100

Table 9: $N_{r,x}$ for erroneous instance

Component	x_1	x_2	x_3
$N_{\text{discard},x}$	0	0	0
$N_{\text{repair at 1},x}$	0	0.5	0.5
$N_{\text{repair at 2},x}$	0.5	0	0.5
$N_{\text{repair at 3},x}$	0.5	0.5	0

Table 10: $M_{r,\mathscr{G}_i}$ for erroneous instance

Set	$\mathscr{G}_1$
$M_{\text{discard},\mathscr{G}_i}$	0
$M_{\text{repair at 1},\mathscr{G}_i}$	0.5
$M_{\text{repair at 2},\mathscr{G}_i}$	0.5
$M_{\text{repair at 3},\mathscr{G}_i}$	0.5

of the costs of repairing that component at the higher echelons.

- Fixed costs are borne by all the components at one indenture level. In our case, this means that all components are in the same set $\mathscr{G}_1$, because they are all at indenture level 1.

The resulting outputs are shown in Tables 9 and 10. The explanation of the results is analogous to the explanation of the results in Section 4.2.1 and is therefore not repeated.

4.4 Model of Saranga and Dinesh Kumar

Saranga and Dinesh Kumar [11] assume that fixed costs are borne by one component. Therefore, these fixed costs are not really different from variable costs. We can construct 'new' variable costs from the 'old' variable costs and fixed costs in the following way (remember that in our model, fixed costs are the mean fixed costs per year): $vc'_{e,r,x} = vc_{e,r,x} + \frac{fc_{e,r,x}}{\lambda_x}$. Using these new variable costs, the new fixed costs are zero. If all fixed costs are zero, all $M_{e,r,\mathscr{G}_g}$ variables can be removed from the model (or set to 1). Section 4.2 shows that no integrality constraints are needed on the $N_{e,r,x}$ variables if all $M_{e,r,\mathscr{G}_g} \in \{0,1\}$. This means that with a little pre-processing, all integrality constraints can be removed for problem instances that comply with the assumptions of Saranga and Dinesh Kumar [11].

Using genetic algorithms for these problem instances, which Saranga and Dinesh Kumar do, is therefore not necessary with our model formulation.

5 Computational experiments

To test the model, we generated instances of the LORA problem and solved these using the CPLEX callable library version 11 (with default settings), running under windows XP, service pack 2, on a Pentium 4, 3.4 GHz with 1 GB RAM. We used only one core of the dual core processor.

In Section 5.1 we explain how we generated the test instances. Section 5.2 provides the inputs we used and discusses some issues concerning the actual testing. Section 5.3 shows and discusses the results of the tests.

5.1 Problem instance generator

In this section, we explain the basic idea of our problem instance generator. More extensive information can be found in Appendix A.

Our problem instance generator receives as inputs the number of components ($|X|$), the number of indenture levels (I), the number of echelons ($|E|$), the number of fixed costs sets ($|\mathscr{G}|$)[7], and the maximum number of fixed costs sets in which each component will be (S). For each number of fixed costs sets $s \mid 0 \leq s \leq S$, a percentage P_s has to be specified, such that $\sum_{s=0}^{S} P_s = 100\%$. P_s is the percentage of components that will be in s sets of components sharing fixed costs. For example, if the components may be at maximum in 1 fixed costs set ($S = 1$), P_0 is the percentage of components that will be in no set at all and P_1 is the percentage of components that will be in 1 fixed costs set. These percentages should add up to 100%.

Depending on the number of components and indenture levels, we calculate how many

[7] In our model, we have sets of components that share fixed costs ($\mathscr{G}_g \in \mathscr{G}$). We will call these sets from now on 'fixed costs sets'.

children every parent component should have approximately in order to get a reasonable system structure. We use this value to draw the number of components at every indenture level and construct the system structure using these values. A reasonable system structure means that we prevent for example that indenture level 1 contains 800 components, and indenture levels 2 and 3 together contain 200 components.

The last inputs are the minimum and maximum values for $vc_{e,r,x}$, $fc_{e,r,\mathscr{G}_g}$, and λ_x. The actual values are drawn from a uniform distribution ranging from the provided minimum to the provided maximum. We adapt the $vc_{e,\text{discard},x}$ and λ_x by adding the values of the child components to the values of their parents.

5.2 Inputs and general issues

In each of our tests, we vary only one parameter. The other parameters get their default values, which are: $|X| = 1,000$, $I = 3$, $|E| = 3$, $|\mathscr{G}| = 100$, and $S = 2$. Every computation time shown in the next section, is the mean value of 1,000 test instances.

If the maximum number of fixed costs sets is set to 2 ($S = 2$), then 10% of the components will not be in any fixed costs set, 10% will be in 1 of those sets, and 80% will be in 2 of those sets. In general: For any number of fixed costs sets $s \mid 0 \leq s < S$, 10% of the components will be in that number of sets. As a result, $100\% - S \cdot 10\%$ of the components will be in the maximum number of fixed costs sets (S will not be larger than 5 in our tests).

In all the tests, we set the minimum and maximum input values for $vc_{e,r,x}$ to 50 and 1,000 respectively, for $fc_{e,r,g}$ to 500 and 10,000, and for λ_x to 0.05 and 5.

At Thales Nederland (a manufacturer of naval sensors and naval command and control systems), we did not see components that are in more than two sets of components sharing fixed costs (e.g., test equipment). The total number of fixed costs sets at Thales Nederland is in general less than 25, and the number of components is less than 1,000. From the literature and from cases at Thales Nederland, we know that both the number of echelons and indenture levels is typically five or less. That is to say, Thales Nederland uses more than

five indenture levels and more than 1,000 components. However, as explained in Section 3.2, some components are removed from consideration during the non-economic LORA that is performed before the LORA is performed as described in this paper. These components include small parts such as screws and relays, and parts that cannot be removed, such as casings. Some repair options for the remaining parts are removed from consideration as well in the non-economic LORA. There might not be enough room on board a ship for certain test equipment, or proprietary knowledge can prohibit the user from repairing a component. In our tests, we did not remove any repair option, so we consider more repair options than there would be in practice.

As explained in Section 3, our model generalizes the models of Barros [10] and Saranga and Dinesh Kumar [11]. The former assumes that fixed costs are borne by all the components at one indenture level; the latter assumes that fixed costs are borne by one component. In our model, fixed costs are borne by sets of components that can be defined freely. For each of these different assumptions about fixed costs, we performed tests with our model. We call tests with general fixed costs sets 'Gen.', tests with fixed costs per indenture level 'Barros' and tests with fixed costs per component 'SDK' (for Saranga and Dinesh Kumar). For the 'SDK' tests, we solved the model as an LP problem, as explained in Section 4.4. In all other cases, we modeled $M_{e,r,\mathcal{G}_g}$ as binary variables.

In some cases, solving the problem instances took so much time, that we restricted CPLEX; we set a time limit of 120 seconds per 1,000 components for each problem instance. The tables provide the number of tests that exceeded the time limit, which only happened for 'Gen.' tests. We excluded these problem instances from the calculations of the computation times. At the end of Section 5.3, we discuss the problem instances that exceeded the time limit. For now, it suffices to mention that we found feasible solutions for all of them.

5.3 Results

Table 11 shows the mean computation times for different numbers of components in the system. In Tables 12 and 13, we vary the number of indenture levels and echelons respec-

Table 11: Computation times (seconds), varying the number of components

# Components	500	1,000	2,000	5,000	10,000	20,000
Gen.	2.433[a]	4.061[b]	12.42[c]	117.6[d]	—	—
Barros	0.101	0.314	1.025	5.033	16.36	55.34
SDK	0.031	0.078	0.205	0.741	2.088	6.122

[a] 2 runs exceeded the time limit of 1 minute
[b] 2 runs exceeded the time limit of 2 minutes
[c] 13 runs exceeded the time limit of 4 minutes
[d] 68 runs exceeded the time limit of 10 minutes

Table 12: Computation times (seconds), varying the number of indenture levels

# Indenture levels	1	2	3	4	5
Gen.	0.256	4.773[a]	4.061[b]	5.685[c]	8.887[d]
Barros	0.178	0.265	0.314	0.431	0.527
SDK	0.033	0.062	0.078	0.094	0.111

[a] 1 run exceeded the time limit of 2 minutes
[b] 2 runs exceeded the time limit of 2 minutes
[c] 7 runs exceeded the time limit of 2 minutes
[d] 15 runs exceeded the time limit of 2 minutes

tively. The run times increase more than linear with the number of components. The run times also increase, as expected, if the number of indenture levels or echelons increases. The run times increase strongly if the number of indenture levels increases from 1 to 2. This is logical, since 1 indenture level means that components are not connected to each other in the product structure (they are all subsystems, and the system is not modeled). They are however connected by sharing fixed costs sets. It is remarkable to see that the average computation time slightly decreases if the number of indenture levels increases from 2 to 3 for the 'Gen.' tests.

Table 13: Computation times (seconds), varying the number of echelons

# Echelons	1	2	3	4	5
Gen.	0.233	1.829	4.061[a]	6.476[b]	7.600[c]
Barros	0.041	0.158	0.314	0.455	0.533
SDK	0.008	0.048	0.078	0.110	0.135

[a] 2 runs exceeded the time limit of 2 minutes
[b] 5 runs exceeded the time limit of 2 minutes
[c] 6 runs exceeded the time limit of 2 minutes

Table 14: Computation times (seconds), varying the maximum number of fixed costs sets of which a component can be part of

# Sets	1	2	3	4	5
Gen.	0.636	4.061[a]	5.940[b]	7.151[c]	8.890[d]

[a] 2 runs exceeded the time limit of 2 minutes
[b] 5 runs exceeded the time limit of 2 minutes
[c] 4 runs exceeded the time limit of 2 minutes
[d] 2 runs exceeded the time limit of 2 minutes

Table 15: Computation times (seconds), varying the total number of fixed costs sets

# Sets	25	50	100	250	500
Gen.	1.628	3.011[a]	4.061[b]	11.60[c]	2.414[d]

[a] 3 runs exceeded the time limit of 2 minutes
[b] 2 runs exceeded the time limit of 2 minutes
[c] 106 runs exceeded the time limit of 2 minutes
[d] 10 runs exceeded the time limit of 2 minutes

The 'Gen.' tests take far more time than those of 'Barros' and 'SDK'. These last two types of problem instances can easily be solved using CPLEX, instead of using genetic algorithms [11] or branch-and-bound techniques [16] . We solve 'SDK' tests as LP problems, so it is not surprising that these are much faster than 'Gen.' tests. In the 'Barros' tests, the number of fixed costs sets is equal to the number of indenture levels. This means that the number of binary variables is much smaller in the 'Barros' tests than in the 'Gen.' tests. An additional explanation of the difference in computation times between the 'Barros' and 'Gen.' tests is that components are more 'connected' in the 'Gen.' tests; if $x_1, x_2 \in \mathscr{G}_1$ and $x_2, x_3 \in \mathscr{G}_2$, a change in the repair option of x_1 can change the best option for x_3.

The findings in Table 14, in which we vary the maximum number of fixed costs sets per component (S), support this assumption. Notice that the computation times increase a lot if the maximum number of fixed costs sets per component increases from 1 to 2. This is not surprising, since 1 fixed costs set per component means that components are connected only to the other components in that one fixed costs set, but they are not connected through these components to other fixed costs sets, as desribed in the previous paragraph. Notice however, that they are still connected to other components in the product structure.

In order to be complete, Table 15 shows how run times change if the total number of fixed costs sets changes. Run times increase with an increasing number of fixed costs sets. However, this changes when the number of sets increases from 250 to 500. We tested what happened with 1,000 sets: The mean optimization time decreases further to 0.564 seconds. A plausible explanation is that this is due to the components becoming less 'connected' to each other. If there are 1,000 components that are at maximum in 2 sets each and there are 1,000 sets, there will be on average less than 2 components per set. With 250 sets, there will be a little less than 8 components per set. If $x_1, x_2 \in \mathscr{G}_1$ and $x_2, x_3 \in \mathscr{G}_2$, a change in the repair option of x_1 can change the best option for x_3. This will probably happen more often if there are 8 components per set than if there are 2 components per set.

In most 'Gen.' tests, a small percentage of the problem instances is not solved to optimality, due to the time limit of 120 seconds per 1,000 components we set on solving the instances. We calculated the gap between the best IP solution that was found at the moment the solver was stopped, and the best lower bound that CPLEX had found at that moment. The gap is mostly below 2%, with exceptional cases of gaps upto 6.6%. It also happened 15 times (out of the 20,000 'Gen.' tests that we performed) that no IP solution was found before the test was stopped. 14 of these tests were problem instances with 5,000 components, the other test was a problem instance with 250 fixed costs sets. These kind of instances are not realistic at Thales Nederland.

If we solve the problem instances that exceeded the time limit (both those for which we found an IP solution and those for which we did not), and set a new time limit of one hour, all but three of the problem instances are solved to optimality. The remaining problem instances are one with 5 echelons, one with 250 fixed costs sets, and one with 5,000 components. If we solve these three problem instances with a time limit of three hours, they are solved to optimality. If we focus on the 'problematic' problem instance with 5,000 components, we see that the LP relaxation is solved after 10 minutes. The first IP solution is found a few seconds later, with a gap of 1.09%. After 20 minutes, the optimal solution is found, but optimality is not verified yet. After one hour, the gap is below 0.1%

and optimality of the solution is verified in three hours.

At the development stage of a product, we do not think that waiting for one hour is problematic. We also think that a gap of below 2% is not problematic, since the input data generally consists of rather rough estimates.

6 Conclusions and directions for further research

We developed a LORA model that generalizes the two LORA models that existed in the literature [10, 11]. We did this by using sets of components that share fixed costs that can be defined freely, instead of assuming that fixed costs are shared between all components at a certain indenture level (Barros) or assuming that fixed costs are borne by one component (Saranga and Dinesh Kumar). This generalization was needed to be able to model cases we found at Thales Nederland. We presented an IP formulation and showed when some of the integrality constraints can be removed (without yielding a fractional solution). Using these results, we were able to show that all integrality constraints can be removed if the model assumptions of Saranga and Dinesh Kumar [11] are used, so that there is no need to solve problem instances using genetic algorithms. We also showed that it is not possible to remove all integrality constraints in the model of Barros [10].

We solved LORA problem instances with sizes that are realistic in practice (Thales Nederland), using CPLEX. Most problem instances could be solved in a couple of seconds. The most important factor that influences the computation time is the number of components in the system. The number of components in cases at Thales Nederland does not cause a problem, but at other companies it might do so. Performing a non-economic LORA could help in such a case to reduce the problem size. The computation times also increase if any of the following increases: The number of indenture levels in the system, the number of echelons in the repair network, or the number of fixed costs sets of which each component is part of. If the total number of fixed costs sets increases, the computation times increase as well, but only until a certain number of fixed costs sets is reached (around 250). After

that, computation times decrease. The computation time of the general model is to over 100 times larger than the computation time for models restricted to the assumptions of Barros or, especially, Saranga and Dinesh Kumar.

It would be interesting from both a theoretical and practical point of view, to link the LORA problem to the problem of determining the required number of spare parts to store at each location in the repair network, given a goal availability of the products. The latter problem is in general solved with (an extension to) METRIC [20, 8].

It may be useful to develop fast heuristics for our LORA model first, before the LORA is coupled to the spare parts optimization problem. Such a fast heuristic may also be useful for companies that develop much larger systems than Thales Nederland does.

Other interesting extensions to our model would be to:

- Explicitly model the repair network. This means that one component can be repaired at different echelons, depending on the system location from which the component originates.
- Introduce different types of failures per component. For example, in 60% of the cases a tester is needed, in 40% of the cases the tester is not needed. This might mean that the latter kind of repairs can be performed on board the ship, whereas the former kind of repairs should be performed at a higher echelon.
- Introduce a step function in the fixed costs, such that the $M_{e,r,\mathscr{G}_g}$ variables are not binary, but integer. For example, if fixed costs are related to buying a tester, this tester cannot be used to test an infinite amount of components. If a certain amount of components is reached, a second, and maybe even a third or fourth tester would be needed.
- Relax the assumption of a 100% probability of succesful repair. This assumption is not realistic in practice and is, for example, also not used in METRIC and its successors [8]. It would be better to have a certain probability of successful repair p. The other percentage of the cases $(1 - p)$ needs another treatment (repair at an higher echelon or discard).

Appendix A

In this appendix, we explain in more detail how we generate problem instances.

As explained in Section 5.1, our problem instance generator receives as inputs the number of components ($|X|$), the number of indenture levels (I), the number of echelons ($|E|$), the number of fixed costs sets ($|\mathscr{G}|$)[8], and the maximum number of fixed costs sets in which each component will be (S). For each number of fixed costs sets $s \mid 0 \leq s \leq S$, a percentage P_s has to be specified, such that $\sum_{s=0}^{S} P_s = 100\%$. P_s is the percentage of components that will be in s sets of components sharing fixed costs. For example, if the components may be at maximum in 1 fixed costs set ($S = 1$), P_0 is the percentage of components that will be in no set at all and P_1 is the percentage of components that will be in 1 fixed costs set. These percentages should add up to 100%.

For every component x, we draw a random number to decide in how many fixed costs sets the component will be. We draw that number of sets $\mathscr{G}_g$, with every set having equal probability. Component x will be in all of these sets. Notice that the number of components per set will in general not be the same for all sets.

Depending on the number of components and indenture levels, we calculate how many children every parent component should have approximately; we call this value c. This c should be such that $\sum_{i=1}^{I} c^i = |X|$ (or $(c - c^{I+1})/(1 - c) = |X|$). For $I \geq 4$ this cannot be solved exactly. Therefore, we use an approximation (for simplicity, we also use the approximation for $I < 4$): First, we determine an auxiliary variable c' such that $(c')^I = |X|$. Then we calculate:

$$c = c' \cdot \frac{|X|}{|X| + \frac{1}{I} \cdot \left(\sum_{i=1}^{I} [(c')^i] - |X| \right)}$$

For $|X| = 1,000$ and $I = 3$, this means that $c' = 10$ and $c \approx 9.65$. This in turn means that $\sum_{i=1}^{I} (c')^i = 1,110$ and $\sum_{i=1}^{I} c^i = 1,000.3$. This last value is very close to $|X|$, which was

[8] In our model, we have sets of components that share fixed costs ($\mathscr{G}_g \in \mathscr{G}$). We call these sets 'fixed costs sets'.

our goal.

To determine the number of components at indenture level i ($|X_i|$), we draw a random number from a uniform distribution ranging from $\frac{1}{2}c$ to $1\frac{1}{2}c$ and we multiply this value by the number of components at the next lower indenture level ($|X_{i-1}|$, notice that $|X_0| = 1$). We initialize $X_{\text{available}} = X$ and for every $i > 0$, we subtract $|X_i|$ from $X_{\text{available}}$. If $X_{\text{available}} < |X_i|$, we set $|X_i| = X_{\text{available}}$. The number of components at indenture level I is not drawn, but is equal to $X_{\text{available}}$ after we have drawn the values for all the lower indenture levels. Notice that it can happen that $|X_I| = 0$, which would mean that the system consists of $I - 1$ indenture levels.

For each of the components $y \in X_i$, we draw with an equal probability any one of the components $x \in X_{i-1}$. This x is the father component of y. Notice that in general, the number of children per parent will not be the same for all parents at a certain indenture level. Notice also that at indenture level 1, so the subsystem level, no father component needs to be drawn.

The last inputs are the minimum and maximum values for $vc_{e,r,x}$, $fc_{e,r,g}$, and λ_x. The actual values for the $vc_{e,r,x}$, $fc_{e,r,g}$, and λ_x are drawn from a uniform distribution ranging from the provided minimum to the provided maximum. Starting at the components with the one but highest indenture level and ending with the components with indenture level 1, the value of all λ_x will be changed to $\lambda_x + \sum_{y \in \Gamma_x} \lambda_y$. We do this, since in practice the demand for a parent component will generally be about the same as the demand for all its child components. In the same way, the variable costs of discard for all its child components are added to the costs of discard for the parent component.

References

[1] B. G. Ferrin, R. E. Plank, Total cost of ownership models: An exploratory study, Journal of Supply Chain Management 38 (3) (2002) 18–29.

[2] Deloitte, The service revolution in global manufacturing industries, 2006.

[3] D. N. P. Murthy, O. Solem, T. Roren, Product warranty logistics: Issues and challenges, European Journal of Operational Research 156 (2004) 110–126.

[4] R. Oliva, R. Kallenberg, Managing the transition from products to services, International Journal of Service Industry Management 14 (2) (2003) 160–172.

[5] I. Gertsbakh, Reliability theory. With applications to preventive maintenance, Springer, Berlin (Germany), 2000.

[6] R. Dekker, R. E. Wildeman, F. A. Van der Duyn Schouten, A review of multi-component maintenance models, Mathematical Methods of Operations Research 45 (1997) 411–435.

[7] M. S. Daskin, Network and discrete location: Models, algorithms, and applications, John Wiley & Sons, New York (NY), 1995.

[8] C. C. Sherbrooke, Optimal inventory modelling of systems. Multi-echelon techniques, 2nd Edition, Kluwer, Dordrecht (The Netherlands), 2004.

[9] J. A. Muckstadt, Analysis and algorithms for service parts supply chains, Springer, New York (NY), 2005.

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

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

[12] United States Department of Defense, MIL-STD-1388-1A Logistics Support Analysis (Notice 4), 1993.

[13] United States Department of Defense, MIL-HDBK-502 Acquisition Logistics, 1997.

[14] United Kingdom Ministry of Defence, Integrated Logistic Support. Part 1: Logistic Support Analysis (LSA) and Logistic Support Analysis Record (LSAR) (Issue 2), 1998.

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

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

[17] G. Gutin, A. Rafiey, A. Yeo, M. Tso, Level of repair analysis and minimum cost homomorphisms of graphs, Discrete Applied Mathematics 154 (6) (2006) 881–889.

[18] PRICE-HL, `http://www.pricesystems.com/products/price_hl.asp`, last checked on 25 July 2007 (2007).

[19] EDCAS, `http://www.itemuk.com/edcas.html`, last checked on 25 July 2007. (2007).

[20] C. C. Sherbrooke, Metric: A multi-echelon technique for recoverable item control, Operations Research 16 (1) (1968) 122–141.

[21] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, Network flows. Theory, algorithms, and applications, Prentice Hall, Englewood Cliffs (NJ), 1993.