Designing logistics support systems

Level of repair analysis and spare parts inventories

Rob Basten

Dissertation committee

Chairman / Secretary	Prof. dr. P.J.J.M. van Loon
Promotor	Prof. dr. W.H.M. Zijm
Assistant Promotors	Dr. M.C. van der Heijden
	Dr. ir. J.M.J. Schutten
Members	Prof. dr. J.L. Hurink
	Prof. dr. R.J. Boucherie
	Prof. dr. E. Kutanoglu
	Prof. dr. ir. G.J.J.A.N. van Houtum
	Prof. dr. ir. R. Dekker

This thesis is number D128 of the thesis series of the Beta Research School for Operations Management and Logistics. The Beta Research School is a joint effort of the departments of Technology Management, and Mathematics and Computing Science at the Technische Universiteit Eindhoven and the Centre for Telematics and Information Technology at the University of Twente. Beta is the largest research centre in the Netherlands in the field of operations management in technology-intensive environments. The mission of Beta is to carry out fundamental and applied research on the analysis, design, and control of operational processes.

This research has been funded by the Innovation-Oriented Research Programme 'Integrated Product Creation and Realization (IOP IPCR)' of the Netherlands Ministry of Economic Affairs.

Ph.D. thesis, University of Twente, Enschede, the Netherlands

Printed by Wöhrmann Print Service

The image on the front cover is based on a photo by Shonna Cunningham of the U.S. Navy. It shows the Hr. Ms. De Zeven Provinciën, a frigate of the Royal Netherlands Navy. Its equipment includes an APAR and SMART-L by Thales Nederland.

ISBN 978-90-365-2967-9

DESIGNING LOGISTICS SUPPORT SYSTEMS

LEVEL OF REPAIR ANALYSIS AND SPARE PARTS INVENTORIES

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. H. Brinksma,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 8 januari 2010 om 15:00 uur

door

Robertus Johannes Ida Basten
geboren op 16 november 1981
te Boxmeer

Dit proefschrift is goedgekeurd door de promotor:

prof. dr. W.H.M. Zijm

en de assistent-promotoren:

dr. M.C. van der Heijden
dr. ir. J.M.J. Schutten

Acknowledgements

Writing a PhD thesis has not only required a huge effort from me, but also from people around me. Therefore, I would like to thank the following people.

First, I thank my supervisors. My department, OMPL, has had four chairmen in the last four years, but Matthieu van der Heijden and Marco Schutten provided steady supervision. They complement each other, having a different focus and different skills, and when I needed help, Marco and particularly Matthieu freed enormous amounts of time. In the last year, Henk Zijm became the chairman of OMPL and my promotor. He made a significant contribution by giving extensive feedback on draft versions of the various chapters in this thesis.

Second, I am grateful to those who facilitated my research. Erhan Kutanoglu provided me with the opportunity to work in Austin, Texas, which I really enjoyed. The logistic engineers at Thales Nederland provided a lot of information and were always there to answer questions. In particular, I thank Cees Doets and Jürgen Donders. Next, I am grateful to the students who made a contribution, most importantly Martijn Smit. He did a lot of work at Thales Nederland and without him, Chapter 6 might not have been there. Lastly in this group, I gratefully acknowledge the support of the Innovation-Oriented Research Programme 'Integrated Product Creation and Realization' (IOP IPCR) of the Netherlands Ministry of Economic Affairs, and I thank the people who attended the meetings of the user committee of the IOP IPCR project. They kept me focused on doing practically relevant research.

Third, my thanks go out to the colleagues at OMPL. In particular, I thank my roommate Leendert, both for his help on (mainly mathematical) problems that I faced and for the pleasant working atmosphere. I also greatly appreciated the coffee breaks with the other PhD candidates and some faculty members.

Finally, I thank my parents, brothers, my girlfriend Miriam, and my friends. I appreciate that they allowed me to talk about my thesis and that they gave me the possibility to take my mind off of it. Furthermore, I am grateful for Berteun's unrivaled LaTeX skills.

Rob Basten
Enschede, December 2009

Contents

Chapter 1

Introduction

Manufactured products and installations are often prone to failure. Inexpensive products, such as many consumer goods, will be discarded upon failure. Capital goods, which are more expensive products and installations, will be repaired. Capital goods are physical systems that are used to produce products or services. The focus in this thesis is on the logistics support system that is required to maximize the operational availability during the lifetime of capital goods. We concentrate on capital goods that are expensive and have high downtime costs. Examples are manufacturing systems, power plants, defence systems, medical devices, and airplanes. In many cases, safety regulations require regular inspections, during which (upcoming) failures are detected; in other cases, the capital good simply stops functioning due to a failure. High downtime costs result in these cases from lost production, missions that need to be aborted, patients that cannot be treated, and flights that are delayed or cancelled. Typical characteristics of capital goods that are relevant in the context of this thesis are, besides their high price and high downtime costs, their technical complexity, low failure rate, geographically dispersed installed base, and long life cycle.

The focus is on corrective maintenance rather than preventive maintenance, since the negative consequences of system downtime arise from unexpected failures, whereas preventive maintenance is usually scheduled. Quick recovery of the system is of utmost importance, which means that capital goods are typically restored by *repair by replacement* of a component. An identical spare part is put in the system, so that it functions again. Since those components can be expensive as well, up to more than several hundreds of thousands of euros, these are repaired by replacement (of a subcomponent) too. In all cases, the question is whether a component (or subcomponent) should be repaired or discarded upon failure, if repair is technically feasible. This economic trade-off is complex, due to the complex product structure, the geographically dispersed installed base, the spare parts that are required, and the various resources (e.g.,

test and repair equipment) that are required to perform repairs. In this thesis, we will develop mathematical models to support the economic trade-off.

The remainder of this chapter is structured as follows. In Section 1.1, we further motivate this research, and in Section 1.2, we define in detail the problem that we focus on in this thesis. An example in Section 1.3 serves to illustrate the problem. We dicuss the relevant literature in Section 1.4, and our contribution to the literature in Section 1.5. This section includes the research objective and research questions. Finally, in Section 1.6, we give the outline of this thesis.

1.1 Motivation

The research in this thesis is part of the IOP-IPCR[1] project *life-cycle oriented design of capital goods*. The goal of the project is to develop a set of quantitative techniques that can be used for an integrated balancing of system availability and life cycle costs (LCC). These techniques are to be used in the development process of capital goods, to gain insights into the impact of design decisions on the LCC and the availability of the product. Below, we motivate the need for such methods.

First, we discuss in Section 1.1.1 why a focus on the total LCC is relevant already at the design phase. Then, in Section 1.1.2, we argue that downtime costs form a large part of the LCC. In Section 1.1.3, we explain that the maintenance strategy is the key factor determining downtime, and, in Section 1.1.4, we show that the maintenance costs, including the costs of setting up maintenance facilities, form a large part of the LCC. The high costs of both maintenance and downtime, and the clear relation between these two cost factors, is the reason that we focus in this thesis on the relation between maintenance (costs) and downtime . Finally, we explain in Section 1.1.5 why we do not optimize the product design itself.

1.1.1 Life cycle costs are a key factor in purchasing decisions

Instead of focusing on the initial purchasing price, customers increasingly take the total life cycle costs into account in their purchasing decisions (Ferrin and Plank, 2002). Therefore, original equipment manufacturers (OEMs) need methods to estimate those LCC. We also observe a trend in which customers outsource activities for product upkeep to the OEM, using service contracts that guarantee a certain service level against fixed annual costs. For the OEM, this may be attractive, since selling services is generally more profitable than selling products (AberdeenGroup, 2005; Cohen et al., 2006; Deloitte, 2006; Murthy et al., 2004; Oliva and Kallenberg, 2003).

[1] In Dutch 'Innovatiegerichte onderzoeksprogramma's – Integrale productcreatie en -realisatie' or 'Innovation oriented research programs – Integral product creation and realisation', which is funded by the Netherlands Ministry of Economic Affairs.

For the acquisition of defence systems, it is required by both the United States Department of Defense (MIL-STD-1388-1A, United States Department of Defense, 1993)[2] and the United Kingdom Ministry of Defence (DEF STAN 00-60 (PART 0), United Kingdom Ministry of Defence, 2004a) that instead of the purchasing costs, the complete life cycle costs are considered, especially the costs for system upkeep.

For OEMs, this means a change in the way of working. The traditional way of working is that the OEM sells a product and is responsible for a functioning system during a limited warranty period. After this period, the OEM can earn from system upkeep by selling spare parts or performing maintenance. In this setting, the sound choice for an OEM at the design phase is to focus on the manufacturing costs of the product. Preventing failures during the warranty period is useful as well; preventing failures during the remainder of the life cycle is less useful from a cost perspective. Nowadays, customers increasingly ask for an LCC estimate or a service contract for system upkeep with a given target availability. Therefore, the OEM needs methods to estimate the maintenance costs. Besides, those costs should be lowered, since that leads to a higher probability of actually selling the product or service contract. Higher costs during the production phase (e.g., more reliable components or more redundancy in a system) can be balanced against lower costs during the use phase and at product disposal (e.g., a more efficient service organization, less downtime costs, lower energy or manpower usage, or lower disposal costs).

1.1.2 Downtime costs are high for capital goods

The focus in this thesis is on capital goods. Since capital goods are very expensive in general, capacity on this type of equipment is usually tight, and uptime or availability is highly important. Consequences of downtime may be very serious. For example:

- If a defence radar system on a naval vessel breaks down, the vessel is vulnerable since it cannot detect incoming missiles anymore.
- If a baggage handling system at an airport stops to function, direct costs of sending luggage at a later point in time, and indirect costs of unsatisfied customers, are very high.
- If an MRI scanner needs to be shut down, there is probably not enough excess capacity to reschedule patients to other MRI scanners, and patients have to be sent home.
- If a lithography system in a semiconductor fabrication facility fails, an entire product line may be down, since this system is often the bottleneck in semiconductor manufacturing.

[2] MIL-STD-1388-1A contains requirements. It is superseded by MIL-HDBK-502 (United States Department of Defense, 1997), which is for guidance only.

Some sources report that downtime costs may be up to $100,000 per hour, e.g., for the computer systems of large e-commerce companies or brokerage firms (cnet news, 2001; Patterson, 2002; Downtime Central, 2009).

1.1.3 The maintenance strategy determines downtime

For a given product design, the key factor that determines the downtime costs is the responsiveness of the logistics support system. Therefore, we have to balance the costs of logistics support, including the costs of setting up maintenance facilities (e.g., locating resources and spares), and the downtime costs. The focus is on corrective maintenance rather than preventive maintenance, since the negative consequences of system downtime arise from unexpected failures, whereas planned maintenance is usually scheduled. However, in Section 1.2, we will see that part of our research can also be applied in a preventive maintenance setting. Furthermore, since maintenance that results from regular inspections can often not be scheduled either, we consider this to be corrective maintenance, and the methods that we develop can be used in this case (see also Section 1.3).

1.1.4 Maintenance costs are high for capital goods

By focusing on the costs of corrective maintenance, we exclude other life cycle costs. In order to show which parts of the LCC are considered, and which parts are not, we discuss the product life cycle. Asiedu and Gu (1998) and Ullman (1997) distinguish four phases in the product life cycle: design & development, production, use, and disposal. Costs during the use phase can be split into operational costs and maintenance (or support) costs (Blanchard, 1998; Blanchard and Fabrycky, 1998). By focusing on corrective maintenance costs, we therefore exclude: design & development costs, production costs, operational costs, preventive maintenance costs, and disposal costs. However, for capital goods, the use phase is typically the longest phase; it can last from a couple of years to up to more than 30 years (e.g., for planes). As a result, the percentage of the LCC that is due to upkeep activities or activities during the use phase in general is quite large:

- Gupta (1983) states that more than 75% of total life cycle costs are made during the use phase. Saranga and Dinesh Kumar (2006) state that it is 80-85%.
- From research that is performed as part of the IOP-IPCR project at Thales Nederland (Basten, 2006), VanderLande Industries (Franssen, 2006), and PANalytical (Meutstege, 2007), we know that 30-50% of LCC is made up of (corrective and preventive) maintenance costs.

An example of relatively high costs during the use phase is also of current interest in the Netherlands. The Netherlands are selecting a new fighter type plane to replace the current F-16 fighter planes. Journalists believe that Saab

has offered 85 planes (Saab Gripen NG) for an initial price of € 4.8 billion, with a maintenance contract for 30 years for another € 4.8 billion (Vrij Nederland, 2009). This means that maintenance costs and purchasing price are the same over the life cycle of the plane. At the moment of writing, it is most probable that another plane will be acquired (F-35 Lightning II, also known as Joint Strike Fighter). The budget to buy 85 planes was raised to € 6.1 billion, and the Netherlands Ministry of Defence estimates that usage and maintenance for 30 years will cost almost € 10 billion (nu.nl, 2009). Assuming that disposal costs are relatively low, this means that costs during the use phase, including operational costs, form more than 60% of the total LCC (the purchasing price covers both design & development costs and production costs).

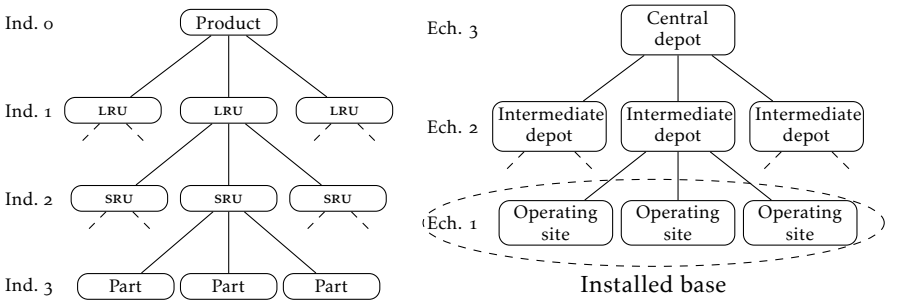
1.1.5 Only a few product designs need to be considered

In practice, only a limited number of product designs is considered during the later stages of the product design process; in the earlier stages, there is not enough detailed information to estimate the costs of maintenance. Therefore, we can focus on a method that determines the costs of maintenance for a given product design. By estimating the maintenance costs for each product design, the product designs can be compared, and the best one can be selected. In such a comparison, other costs, such as the production costs, may be compared as well, and other considerations may play a role too. In this way, the methods that we develop can be part of a design for maintainability or design for serviceability approach (see, e.g., Gershenson and Ishii, 1993).

1.2 Level of repair analysis and spare parts stocking

As mentioned in the introduction of this chapter, capital goods are generally maintained by a repair by replacement policy: a failed component is removed from the product and replaced by a functioning spare part, if available. Otherwise, the replacement has to wait until a functioning component arrives. In the military world, the components that are taken out of the product are called *LRUs* or *line replaceable units*. Defective LRUs can either be discarded or repaired. If it is discarded, a new LRU needs to be purchased. If it is repaired, then possibly a subcomponent needs to be replaced by a functioning one. Since this replacement is typically performed in a repair shop, these subcomponents are called *SRUs* or *shop replaceable units*. The SRU should in turn be repaired, possibly by replacement of a *part*, or discarded itself. The product is thus characterized by a *multi-indenture product structure* as shown in Figure 1.1a, where an indenture level is the level in the product structure. If the indenture level is not important, we use the term *components*, which can have *subcomponents*. In principle, any number of indenture levels is possible.

In practice, the installed base, consisting of all systems that are sold and still in



(a) Product structure (b) Repair network

Figure 1.1: Examples, including the naming convention that we use

use, is usually dispersed over a large geographical area. A support network is needed with facilities that are not too far from the installed base locations, the *operating sites*. However, locating repair and test equipment and spares close to each of the operating sites is usually expensive. Therefore, often more central locations are used to stock some spare parts and to locate more expensive equipment. As a consequence, a repair network usually consists of multiple *echelon levels*. Figure 1.1b shows an example including the naming convention that we use. In principle, any number of echelon levels is possible. In practice however, it is usually limited to three. If there are various echelon levels, it should be decided where to perform repairs, where to locate resources (e.g., test and repair equipment), and where to stock spare parts. The OEM is usually responsible for the support network, except in the military world. There, the customer often owns its own support network.

The numbering of the indenture levels and echelon levels might be a bit confusing at first sight. However, it is used both in practice and in the literature (see, e.g., Sherbrooke, 2004). The logic is that the repair of a system starts by finding the LRU (indenture level 1) that failed, repairing this LRU by replacing an SRU (indenture level 2), and so on. At the moment the system fails, it is at the operating site (obviously), which is at echelon level 1. Components that failed may then be moved upstream in the repair network to higher echelon levels.

In Sections 1.2.1 to 1.2.3, we discuss the problems that we consider in this thesis. However, we first clearly define the term *availability* as we use it (see, e.g., Sherbrooke, 2004, for multiple definitions of availability). A system is available, or operational, if it is not down for either maintenance or because it is waiting for spare parts. This is reflected by the definition of operational availability, which is $\frac{\text{MTBM}}{\text{MTBM}+\text{MCMT}+\text{MPMT}+\text{MSD}}$, with MTBM being the mean time between maintenance, MCMT being the mean corrective maintenance time, MPMT being the mean preventive maintenance time, and MSD being the mean supply delay, so the time waiting for spares. The operational availability may also be

approximated as the product of two availabilities: maintenance availability and supply availability, with maintenance availability being $\frac{\text{MTBM}}{\text{MTBM+MCMT+MPMT}}$, and the supply availability being $\frac{\text{MTBM}}{\text{MTBM+MSD}}$. In this thesis, we are interested in the delay time due to a lack of spares. As a consequence, we wish to minimize the MSD, and therefore, the term availability in this thesis refers to the *supply availability*.

Furthermore, notice that all times (e.g., MTBM) are mean times: since we consider stochastic variables, the availability as we use it, is the expected average availability over the life cycle of all capital goods in the installed base.

1.2.1 Level of repair analysis problem

The level of repair analysis (LORA) problem is to determine whether a component should be repaired or discarded upon its failure, and at which location in the repair network to do that. To enable certain types of repairs, resources have to be located in the repair network as well. The goal is to achieve the lowest costs over the life cycle of the product. Those costs consist of both fixed costs and costs that are variable in the number of failures. Variable costs include costs of hiring service engineers and transportation of components; fixed costs include costs for resources such as test equipment and tools. The number of spare parts that need to be stocked in the network and the availability of the installed base are not considered in the classical LORA, but in a spare parts stocking problem that is solved after the LORA has been solved. We come back to this in Section 1.2.2.

Commercial LCC estimation tools generally contain a LORA part, see for example PRICE HL (2007) and EDCAS (2009),[3] but it is not clear how they function.[4] In the military world, a LORA is usually requested by the customer.

In practice, not all components in the product structure are considered in the (economic) LORA that we focus on in this thesis. For other components, not all repair/discard options may be available. This is a result of the non-economic LORA that is typically performed before a(n economic) LORA is performed (see also Section 6.5.1).[5] In the non-economic LORA, it is determined that some components cannot be repaired at all, or can be repaired by the OEM only. Other repairs cannot be performed at the operating site, since, for example, there is no space for certain resources, or a vibration-free environment is required. Other components (e.g., bulk items such as screws or cables) are so inexpensive that they are discarded by default.

[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., Barros, 1998).

[4] It is noticed that it is unclear how commercial LORA tools function both by experts who have been using these tools and in the literature (e.g., Brick and Uchoa, 2009).

[5] In the remainder of this thesis, the term LORA refers to the economic LORA, unless stated otherwise.

1.2.2 Spare parts stocking problem

The spare parts stocking problem is generally solved using the decisions that result from the LORA as an input. The goal is to allocate spare parts inventory in a repair network such that a certain availability of the installed base is achieved against the lowest possible spare parts costs. In the military world, a recommended spares list is usually requested at the acquisition phase. Various commercial tools exist that can perform the spare parts stocking problem. For example, a tool, which comes from the same company as the aforementioned EDCAS (2009), is VMetric (2009).

1.2.3 Joint problem of LORA and spare parts stocking

The focus in this thesis is on the joint problem of LORA and spare parts stocking. Since both the LORA and spare parts stocking problem are well known in the military world, we illustrate both problems and possible solutions by means of case material of Thales Nederland, a manufacturer of naval sensors and naval command and control systems.

In practice, the joint problem of LORA and spare parts stocking is usually solved sequentially, as mentioned in Section 1.2.2. However, the LORA problem is often not solved explicitly using a formal model, but implicitly using expert knowledge and the decisions made for earlier products. Spreadsheets are used to calculate the costs for a few scenarios only. After solving the LORA problem and spare parts stocking problem, it may turn out that spare parts costs for some components are very high. In that case, a second iteration is sometimes made in which another LORA decision is taken for those components. Obviously, it is not guaranteed that the optimal solution is found in this sequential or iterative manner. Furthermore, this way of working is time consuming and there is a lot of room for errors (see also Section 6.5.1).

We explained that we focus on corrective maintenance. Although a LORA can be used for preventive maintenance as well, the joint problem of LORA and spare parts stocking is different for preventive maintenance. The key reason for this is that preventive maintenance is usually scheduled, which means that demand for spare parts occurs at set intervals only (some components are always replaced, other components are inspected and replaced if necessary), whereas for corrective maintenance the occurrence of demands is a (more or less) continuous stochastic process.

1.3 Example

An example may serve to illustrate the joint problem of LORA and spare parts stocking. Let us consider an airline (e.g, KLM) with a fleet of planes (e.g., Boeing 777-200). A plane is generally used for a large number of years, and it will need maintenance during its life span. Because of safety regulations, vital

components of the airplane are inspected at each airport where it lands. At the main hub of the airline (e.g., Schiphol, Amsterdam), the airplane is periodically inspected more thoroughly. At both types of inspections, components that do not function according to specification are maintained, and replaced if required. (At the latter type of inspection, some components are also preventively maintained or replaced; we do not consider these components.) Although this setting reflects condition based preventive maintenance, the inspection intervals are so short, that the occurrence of demands can be seen as a continuous stochastic process, which means that the methods that we develop in this thesis may be applied in this situation.

At the design phase of the plane, or at the moment the airline acquires the planes, it is determined what will be done if a part needs maintenance. Let us assume that if the engine fails, it is replaced in total (e.g., if a bird flies into the engine). The engine can then be sent to the OEM for repair. However, shipping a complete engine is expensive. Besides that, the engine is away for a long time. If the probability is relatively high that any of the other planes needs a spare engine in that period as well, then to keep the planes available for 99% of the time, at least two, but maybe even more spare engines may be required. Therefore, the airline may decide to acquire repair equipment so that it can stock the subcomponent of the engine that failed. If the equipment is available at the hub, the subcomponent that failed can be located and replaced by a spare part. This spare part is usually far less expensive than a complete spare engine. If locating and replacing the defective subcomponent can be done quickly, stocking one spare engine only may be sufficient. Subcomponents need to be stocked instead of complete engines, and subcomponents are sent to the OEM for repair instead of complete engines, which leads to lower shipping costs. Instead of repairing the subcomponents, some subcomponents may also be discarded, or the airline may decide to repair some of these subcomponents itself.

In our example, decisions are taken on whether to discard or to repair components, where to perform the repairs, whether or not to buy test equipment, and on the number and locations of spare parts to stock in order to achieve a target availability of the planes. These are the decision problems that are studied in this thesis. At other companies, the same kind of problems exist, although the number of options may differ, for example, if the number of echelon levels in the repair network is smaller.

1.4 Literature

Most of the related literature focuses on one of the two problems: LORA and spare parts stocking. To the best of our knowledge, only one paper exists that provides a model and solution method that can be used to solve the two problems simultaneously. We discuss the literature below; in Chapter 2, we discuss the literature in more detail.

1.4.1 LORA

The literature on LORA is limited; to the best of our knowledge, only the papers by Barros (1998), Barros and Riley (2001), Saranga and Dinesh Kumar (2006), and Brick and Uchoa (2009) discuss the key issues of LORA. Some of the presented models can be used for multi-indenture product structures and multi-echelon repair networks, others are more restricted in this perspective. However, the multi-indenture, multi-echelon models have very restrictive assumptions on the resource-component relations: all components at one indenture level require the same resource in order to be repaired (there are as many resources as there are indenture levels), or each component requires its own resource (no sharing of resources).

Problem instances in practice (e.g., at Thales Nederland, see Section 2.1.1) generally require that multi-echelon repair networks, multi-indenture product structures, and fairly loose restrictions on the resource-component relations can be modelled. None of the models in the literature fits these requirements.

1.4.2 Spare parts stocking

A vast amount of literature exists on the (multi-item) spare parts stocking problem. In the context of this research, we are interested in expensive, slow moving, repairable components. The paper of Sherbrooke (1968) is generally seen as the seminal paper in this field. He developed the METRIC model (multi-Echelon Technique for Recoverable Item Control), which is the basis for a huge stream of METRIC type models. We refer to Sherbrooke (2004) and Muckstadt (2005) for an extensive overview of the literature on METRIC type models.

Given the long tradition in spare parts stocking, the current state-of-the-art is sufficient to solve spare parts stocking problems in practice. In the defence industry, it is quite common to apply such models.

1.4.3 Simultaneous LORA and spare parts stocking

To our knowledge, the paper by Alfredsson (1997) is the only paper in which the LORA and spare parts stocking problem are solved simultaneously (instead of solving the LORA first and then the spare parts stocking problem). However, the model is too restrictive to be used in practice, since the author assumes one indenture level and two echelon levels, and uses very strong assumptions on the resource-component relations (see Section 2.4).

1.5 Contribution

This section gives the research objective and the research questions. We elaborate on these questions and thereby show the contribution of this thesis.

1.5.1 Research objective

Our research objective is:

To develop a method that companies can use to analyze the joint problem of level of repair analysis and spare parts stocking for multi-indenture, multi-echelon problem instances.

The solution to the joint problem prescribes for a given product design and repair network:

- Which components to repair upon failure, and which to discard,
- for each component that will be repaired, where in the repair network to do this,
- for each required resource (e.g., test or repair equipment), where in the network to install it, and
- the locations and amounts of spare parts to stock,

such that a target availability is achieved against the lowest possible life cycle costs.

1.5.2 Research questions

We noticed in Section 1.4 that there is little literature on LORA, let alone on the joint problem of LORA and spare parts stocking, whereas there is a lot of literature available on the spare parts stocking problem. We have to know exactly what literature is available and what is needed in practice. This means that our first research question is:

(1) Which methods are available to analyze the LORA and spare parts stocking problem, what is required in practice, and what are therefore the gaps in the literature?

For the spare parts stocking problem, we find that what is required in practice (e.g., Thales Nederland) is available in the literature. However, the models that are available for the LORA problem are not sufficient to be used in practice. Therefore, we focus on the LORA problem, and our second research question is:

(2) What is a suitable LORA model that can be solved in a reasonable amount of time for problem instances with a size that is realistic in practice?

To answer this question, we formulate two subquestions. The main problem with the models in the literature is that the assumptions on the sharing of

resources between components are too restrictive. Therefore, our first subquestion to answer is:

(2a) In what way can we generalize the models that are available in the literature?

Answering this question leads to a model that gives insights into the existing models and the LORA problem in general. For example, we use the model to show that the LORA problem is NP-hard in general. Since this model generalizes the existing models, it provides a good basis for further work. To be able to model realistic problem instances, the model needs to be extended, for example by allowing for a probability of unsuccesssful repair (instead of assuming that repair is always successful). Therefore, the second subquestion is:

(2b) How can we model the extensions that may be needed in practice?

To model the extensions, we reformulate the LORA model as a minimum cost flow model with side constraints. We extend that model with practically relevant extensions. This means that we have developed a LORA model that can be used in practice.

Since there are already good models and methods to solve the spare parts stocking problem, we reached, in a way, our research objective. However, we do not know the quality of the total solution (LORA and spare parts): when solving both problems sequentially, the LORA may result in the need to perform many repairs at a central location, since this means that repair equipment needs to be located at one location only. This implies that the installed base faces long repair lead times, which increases the amount of spare parts inventory. If repairs would be performed at the operating sites, we need more repair equipment, but less spares, to achieve the same target availability of the installed base. In an integrated model, the higher costs of resources can be balanced against the lower costs of spares. For this reason, Alfredsson (1997) and Brick and Uchoa (2009) stress the importance of an integrated model, which is what we pursue next:

(3) What is a suitable method to solve the joint problem of LORA and spare parts stocking?

There are various ways to approach this joint problem. One of the more obvious ways is to use existing methods to solve the LORA and spare parts stocking problems and make a feedback loop to get the results of the spare parts stocking problem to the LORA. In an iterative way, this should lead to a good overall solution. Therefore, our first subquestion is:

(3a) How can we iteratively use a LORA model and a spare parts stocking

model to solve the joint problem of LORA and spare parts stocking?

This iterative method often leads to lower total costs than solving the LORA and spare parts stocking problems sequentially. This shows that usage of a method that solves the joint problem is useful. However, although the iterative method often achieves interesting cost reductions, it does not guarantee to find a good solution: robustness appears to be an issue. This leads to the second and final subquestion:

(3b) Which method can we use to solve the joint problem of LORA and spare parts stocking in a more robust way, leading to a solution that is close to optimal?

Answering this subquestion completes our research.

1.6 Outline of the thesis

The outline of the thesis closely follows the research questions. In Chapter 2 (question 1), we discuss the relevant literature in more detail and we discuss what is needed in practice. This leads to a list of gaps in the literature. In Chapter 3 (question 2a), we present a LORA model that generalizes the models that exist in the literature. This model closely resembles the existing models. In Chapter 4 (first step in answering question 2b), we then reformulate the LORA model as a minimum cost flow model with side constraints, which can be solved fast and proves to be easy to extend. We discuss possible extensions in Chapter 5 (second step in answering question 2b). This chapter concludes our discussion of the LORA models. In the next chapter, Chapter 6 (question 3a), we present an iterative method to solve the joint problem of LORA and spare parts stocking. We conclude our discussion of the joint problem in Chapter 7 (question 3b) with an integrated method. This thesis ends with Chapter 8, in which we give conclusions and recommendations for further research.

Chapter 2

Literature and research challenges

In this chapter, we answer research question 1: "Which methods are available to analyze the LORA and spare parts stocking problem, what is required in practice, and what are therefore the gaps in the literature?" To this end, we start in Section 2.1 with a discussion of the requirements that problem instances in practice pose on LORA and spare parts stocking models. In the next few sections, we discuss parts of the literature: the literature on the LORA problem in Section 2.2 and the literature on the spare parts stocking problem in Section 2.3. In Section 2.4, we discuss the one paper in which an algorithm is developed to solve the joint problem of LORA and spare parts stocking. In Section 2.4, we also discuss related literature that may be used when developing an algorithm to solve the joint problem. Section 2.5 concludes this chapter by listing the gaps in the literature.

2.1 Requirements in practice

Throughout this thesis we refer to a case study at Thales Nederland, a manufacturer of naval sensors and naval command and control systems. Section 2.1.1 discusses that case study and Section 2.1.2 lists the requirements that are posed by problem instances in practice, partly based on the case study.

2.1.1 Case study

This section discusses the case study that we performed, which is representative for the LORA and spare parts stocking problems that Thales Nederland faces. Two examples of Thales radar systems are the SMART-L, which is a rotating long range surveillance radar, and the APAR or Active Phased Array Radar, which

does not rotate. APAR can be used for surveillance, tracking of hostile missiles, and guidance of missiles and canons. A non-rotating radar has the advantage that there are less mechanical parts that are prone to wear-out. Moreover, it is easier to keep out (salt) water and other environmental influences. Rotating radar systems are generally less expensive. There also exist electro-optical surveillance systems, which use, for example, infrared cameras to detect hostile units. The advantage of such systems is that they are passive, whereas radar systems transmit a signal, which may be detected by enemies. The case study concerns a combined radar and electro-optical surveillance system that is mounted on a naval vessel (operating site). The exact system is confidential; we will refer to the system as *sensor system*.

Some spare parts are typically stocked on board the ship, and the crew can exchange many components (the LRUs or line replacable units). Some repairs can be performed on board the ship, but most specialized equipment is typically not available there. The case study concerns twelve ships; seven of them are located at one base (intermediate depot), and five ships are located at another base. The two bases are linked to a central depot, which, in turn, can send components to Thales Nederland for repairs. New components may also be ordered at Thales Nederland. Thus, the repair network consists of four echelon levels.

We consider three indenture levels in the product structure and over 200 components, of which 40% are LRUs. The sensor system consists of more components and more indenture levels, but not all components are relevant. This is explained in more detail in Section 6.5, but one may think, for example, of bulk items such as screws, bolts, and wiring.

In order to test and repair components, there are 54 different resources, 34 of which are 'adapters'. These adapters are used in concurrence with other test/repair equipment. This means that some expensive equipment can be used to test or repair a set of components, but distinct adapters need to be acquired for each component that is actually tested or repaired on the equipment. Notice that this means that there are sets of components requiring the same resource, and there are components that require two (or more) resources simultaneously.

Some of the resources that we consider are required for calibration: some LRUs need to be tuned when they are put back in the system. Therefore, there is a choice to have this equipment at each ship, or have it at the bases only. In the latter case, if a failure occurs in that LRU, the sensor system will be down until the ship returns to its base.

Costs of the components can be upto one million euros, and costs of resources can be upto a couple of million euros. As a result, the life cycle costs of twelve sensor systems are tens of millions of euros. Resources are not used intensively. Even at depot, usage rates are below 25% in general, which means that no two resources of the same type are required at any one location.

The annual failure rate of each LRU per ship is in the range of 0.001 to 1.5.

Lead times can be more than a year for newly acquired components, upto half a year for repairs at the OEM, and over one and a half months for repairs in the customer's repair network. The lead time to get a component at ship from base is related to the average mission time, which is an input from the customer. In the case study it is two weeks. This lead time could be reduced by using emergency shipments, for example, by using a helicopter to send a new component to a ship. However, we do not consider this in the model, since customers do not want to perform such shipments on a requlars basis.

Not all repairs are successful in practice. For many components, a probability of unsuccessful repair is specified, which is usually 5%. There are also components that are returned to be repaired, but no failure is found when diagnosing them. After extensive testing, such a component is returned to stock. However, Thales Nederland does not have the data available to incorporate 'no-fault-founds' in the considerations yet. Still, this may be desirable in the future. Furthermore, there are components in which multiple failure modes can be distinguished: for example, simple failures that can be repaired without any resource, and more difficult failures (of the same component) that require a resource in order to be repaired.

We conclude that in order to solve this case study, we need a model that can deal with multi-indenture product structures, multi-echelon network structures, and very loose restrictions on the resource-component relations. In addition, it would be useful if the model can cope with probabilities of successful repair and no-fault-found probabilities.

2.1.2 General requirements

Based on the case study, on experience at other companies (in particular those that participate in the IOP-IPCR project), and on the literature, this section gives the requirements that are generally posed on LORA and spare parts stocking models and methods.

The number of indenture levels in the case study at Thales Nederland is three; in the case study of Saranga and Dinesh Kumar (2006) there are two indenture levels. In general, multiple indenture levels exist in the product structure. Multiple echelon levels in the repair network are common as well. Cohen et al. (1997) notice in a benchmark study that three-echelon networks prevail, followed in popularity by two-echelon networks. In the case study, there are four echelon levels.

The repair network may be asymmetrical, which means that, for example, not the same number of operating sites is supplied by each intermediate depot. In the case study, five ships are attached to one base and seven ships are attached to another base. This means that taking the same decision at each location at one echelon level may not be optimal. A network may also be unbalanced due to higher failure rates of the products (due to more intensive usage), higher costs, or longer lead times in parts of the network.

Resource-component relations may be very general. Some components require multiple resources in order to be repaired, and some resources are required by multiple components, possibly at multiple indenture levels. At Thales Nederland, the usage rates of the resources are so low, that the waiting time for a resource will never be significant. Therefore, at any location, at most one resource (of the same type) is required, which means that we may assume uncapacitated resources. However, this is not the case at all componanies (see, e.g., the assumptions in Alfredsson, 1997).

We conclude that the key requirements for a model that is to be used in practice, are that it should cover multiple indenture levels and multiple echelon levels. Besides, it should cover fairly general resource-component relations. For some problem instances, it may be required to model the exact repair network, whereas for other problem instances, data may be aggregated such that the same decision is taken at each location at the same echelon level (which is often done in the literature, see Section 2.2). Other extensions may be required as well, such as the usage of capacitated resources, multiple failure modes per component, a probability of unsuccessful repair, and a no-fault-found probability.

The current way of working in practice is using a sequential approach of solving a LORA first, and then a spare parts stocking problem. In the case study, spare parts costs make up over 50% of the total costs (LORA and spare parts stocking). Since these costs are not included in the LORA, it is doubtful whether an overall optimal solution can be achieved using the sequential approach. Another approach in which the two problems are solved simultaneously may be required.

2.2 Level of repair analysis

This section dicusses the literature that exists for the LORA problem (Barros, 1998; Barros and Riley, 2001; Saranga and Dinesh Kumar, 2006; Brick and Uchoa, 2009). The analysis of a LORA model should prescribe for a given product design and repair network:

- for each component, whether to repair or discard it upon failure, and
- where in the repair network to do this, and
- for each required resource (e.g., test or repair equipment), where in the network to install it,

such that the lowest possible life cycle costs are achieved. Those costs consist of both fixed costs and costs that are variable in the number of failures. Variable costs include costs of hiring service engineers and transportation of components; fixed costs include costs for resources such as test equipment and tools.

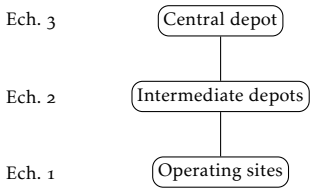


Figure 2.1: Three-echelon repair network (aggregated)

Barros (1998) and Barros and Riley (2001) use the same (mixed) integer programming model. Throughout these two papers, two-indenture product structures and two-echelon repair networks are assumed. However, the authors state that the model can be used to solve problem instances with any number of indenture levels and echelon levels. All data is aggregated per echelon level, which means that the same decision is taken for each location at one echelon level; each three-echelon repair network would be represented as in Figure 2.1.

There are $|E| + 1$ possible decisions for each component, with E being the set of echelon levels in the repair network: a component can be repaired at one of the echelon levels, or it can be discarded. The assumption with respect to the resource-component relations is that all components at one indenture level require the same resource in order to be repaired. As a result, the number of resources in the model is equal to the number of indenture levels in the product structure, and every component requires exactly one resource. The resources are uncapacitated.

Barros (1998) uses a commercial linear programming solver (LINDO) to solve the model[1], and Barros and Riley (2001) use a dedicated branch-and-bound method.

Saranga and Dinesh Kumar (2006) assume three-indenture product structures and three-echelon repair networks, but the extension to general multi-indenture, multi-echelon problem instances is straightforward. The authors model three possible decisions per echelon level: repair, discard, or move to the next higher echelon level. If the last decision is taken, one of the three options at the next higher echelon level needs to be chosen. At the highest echelon level, the move option is not available. The result is that for each component one repair or discard will be chosen at one of the echelon levels. Resources are not shared between multiple components, and each component requires exactly one resource: the number of resources is equal to the number of components. As in the model of Barros, data is aggregated per echelon level and resources are uncapacitated. The model is an integer programming model, which is solved using genetic algorithms.

[1] We mentioned that the model is a (mixed) integer programming model. However, Barros (1998) assumes that the relaxation of the integer programming formulation leads to a natural integer solution. We come back to this in Section 3.2.3, in which we show that this is not always true.

Barros (1998), Barros and Riley (2001), and Saranga and Dinesh Kumar (2006) focus purely on the LORA problem. Brick and Uchoa (2009) add to the LORA problem the problem of where to locate repair facilities, the well known facility location problem (see, e.g., Daskin, 1995). Since this increases the complexity of the problem, the authors have to make simplifying assumptions. They assume single-echelon repair networks and two-indenture product structures, but they pose fairly general restrictions on the resource-component relations. Resources are capacitated, which means that multiple resources (of the same type) may be required at one location, and data is not aggregated per echelon level. Instead, the exact network is modelled. The model is a mixed integer programming model, which is solved using a commercial solver (CPLEX 9.1).

2.3 Spare parts stocking

Given the repair/discard decisions per component, which result from the LORA, the spare parts stocking problem is to find the most cost effective allocation of spare parts in a network that achieves a target availability of the installed base.

A vast amount of literature exists on the multi-item spare parts stocking problem. Lines of research can be distinguished based on their focus on repairable or consumable components, and on fast moving, inexpensive components or slow moving, expensive components. In the context of this research, we are interested in expensive, slow moving, repairable components. The paper of Sherbrooke (1968) is generally seen as the seminal paper in this field. He developed the METRIC model (Multi-Echelon Technique for Recoverable Item Control), which is the basis for a huge stream of METRIC type models. The initial model can be used for single-indenture items only. Muckstadt (1973) developed the first multi-echelon, multi-indenture model, called MOD-METRIC. The development of the VARI-METRIC models (Slay, 1984; Graves, 1985; Sherbrooke, 1986) has been another important step forward; the relations between various echelon levels and indenture levels are more accurate in these models (see also Section 7.1). These relations can also be calculated exactly (Graves, 1985; Rustenburg et al., 2003), but this is computationally intensive. We refer to Sherbrooke (2004) and Muckstadt (2005) for an extensive overview of the literature on METRIC type models. We will use VARI-METRIC in our experiments if we require a spare parts stocking analysis method. Therefore, we will often use the term VARI-METRIC from now on, even though the statements are also valid for most other METRIC type models.

VARI-METRIC aims to find the most cost effective allocation of spare parts in a network that achieves a target availability of the installed base. Equivalently, the availability can be maximized for a given budget and maximizing the availability is approximately equivalent to minimizing expected backorders (EBO) for LRUS at operating sites. A backorder arises if a request for a spare part cannot be fulfilled immediately. Backorders at higher echelon levels or higher indenture levels influence the availability only in an indirect way, since they

influence the lead times of requests for LRUs at operating sites.

Two key assumptions that are generally used in the METRIC type models are:

- A location in the repair network at echelon level e is only supplied from its parent-location at echelon level $e+1$, not by a lateral supply from another location at echelon level e or by emergency shipments from locations at an echelon level $> e+1$.

- One for one $(s-1, s)$ replenishment is appropriate for every component at every echelon level.

The METRIC type models can be solved using a Lagrangian method or using a marginal analysis approach. In general, the former method leads to a solution that is not as good as the solution of the latter method. Moreover, the latter method leads to an EBO-curve, which we require for our method in Chapter 7, whereas the former method does not.

We focus on the marginal analysis approach in detail in Section 7.1. To explain the basic idea, we first define an LRU family as an LRU including all its subcomponents at any indenture level (Muckstadt, 2005). Since the EBO of all LRUs can be summed in order to get the EBO of the complete product, the overall spare parts stocking problem is separable per LRU family. For each LRU family a subproblem should be solved, which results in an EBO-curve. An EBO-curve is a set of EBO-costs-combinations, in which each combination corresponds to a number of spare parts, allocated to the locations in the repair network.

To solve the overall problem, it is used that each EBO-curve is convex. Starting without any spares, that EBO-curve is picked for which adding a spare leads to the highest EBO-reduction per dollar (biggest bang for the buck). This spare is added, and the next best spare to add is found, et cetera. Since the curves are convex, the first step on a curve leads to an EBO-reduction per dollar that is at least as high as that of the next step. This guarantees that it is optimal to look ahead one step per curve only when adding spares. Spares are added until an EBO-value is reached that corresponds with an expected availability that is at least as high as the target availability. Except for the approximation errors in VARI-METRIC, the solution that is thus found is an efficient point, which means that the same availability cannot be achieved against lower costs. However, there may be solutions that achieve a lower availability, which is still higher than the target availability, against lower costs. We come back to this in our detailed explanation of VARI-METRIC in Section 7.1.

For the special case of a single system per operating site, Rustenburg (2000) shows that it is better to use the sum of the backorder probabilities for all LRUs at the operating sites (PBO) instead of the sum of the expected number of backorders (EBO). We come back to this in Section 7.1 as well.

2.4 Joint problem of LORA and spare parts stocking

To the best of our knowledge, only one paper so far focuses on the joint problem of LORA and spare parts stocking (Alfredsson, 1997). However, more literature exists on a related problem: the joint problem of facility location and inventory stocking. Therefore, we first discuss the paper by Alfredsson, and then we discuss the literature on the related problem.

Alfredsson (1997) assumes a single-indenture product structure, and a two-echelon repair network, but the extension to more echelon levels is straightforward. The data is not aggregated per echelon level. Instead, the exact network is modelled. Each component requires one specific tester (resource), which is required by one component only. Furthermore, one multi-tester exists. This multi-tester can be used for the repair of one component, 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 component, the original specific resource for that component is not used anymore. Resources are capacitated, which means that multiple resources of the same type may be required at one location. Furthermore, system downtime includes the waiting times for the resources, the repair times, and the waiting times for spares. The model is a non-linear integer programming model.

Alfredsson uses a decomposition method that sequentially decomposes the overall problem in smaller subproblems. We focus on the author's method in more detail in Section 7.2, since we use his ideas in the method that we develop there. The basic idea is that he can decompose the problem into subproblems per resource, in a way similar to how the problem is decomposed per LRU family in the marginal analysis approach for the METRIC type models. The other way of decomposing the problem is to fixate a decision variable. For example, a resource can be either at echelon level 1 or at echelon level 2 in his model (not at both echelon levels or none of the echelon levels). Therefore, he solves a subproblem in which the resource is at level 1, and a subproblem in which the resource is at level 2. Using a METRIC type marginal analysis method he gets an EBO-curve for each of the subproblems and next he finds the convexification of the lower envelope of these two curves. This is the EBO-curve for the total subproblem of this resource.

Although only one paper exists that focuses on the joint problem of LORA and spare parts stocking, related work exists in which the facility location problem and the inventory stocking problem are solved simultaneously (not necessarily in the context of logistics support systems, but, for example, in the context of retail). This means that given possible facility locations and demand points with a given annual Poisson-distributed demand, it should be decided:

- which facilities to open,
- which demand points to connect to each of these facilities, and
- the amount of spare parts to stock at each facility.

The goal is to minimize the total costs (facility operating costs, transportation costs, and inventory holding costs) such that a target fill rate is achieved. The fill rate is the percentage of requests for components that can be satisfied immediately. A target can be specified for the fill rate per component or for some weighted average over the fill rates of multiple components.

This is a similar problem to ours since it also covers the decisions of locating resources (facilities), assigning demands to locations, and stocking inventory at locations, such that a certain service criterion is achieved. Besides some smaller differences, there are three key differences with our model:

- a constraint on the fill rate is used instead of a constraint on the availability, as mentioned above,
- the focus is on single-indenture product structures only, often even a single component, instead of multi-indenture product structures, and
- resources can usually be located at one echelon only. Although customers need to be assigned to the facilities, these models therefore basically consider single-echelon repair networks, instead of multi-echelon repair networks.

As a result of the first two differences, often an item approach is used instead of a system approach.

Most of the literature in this field uses an ordering policy similar to a (Q, r) policy. For example, determining the order quantity Q using an economic order quantity (EOQ) model, and then determining the reorder point r (see, e.g., Daskin et al., 2002; Gabor and Van Ommeren, 2006). This makes these models more applicable in the setting of fast-moving components with relatively high set-up costs, which are, for example, found in supply chains for consumer goods, whereas we develop a model that can be used in the context of a service supply chain with low demands. For an overview of facility location problems in the former context, we refer to Melo et al. (2009). The authors survey the literature published in the last decade, associated with both the facility location problem and supply chain management (but not service supply chains). They identify approximately 120 papers, out of which 33 integrate inventories with the facility location problem in some way.

However, we mention two papers that are applicable in the setting that we are interested in: slow moving components for which one for one $(s-1, s)$ replenishment is used: Candas and Kutanoglu (2007) and Jeet et al. (2009). In both papers, the standard fill rate defined above is adapted such that a certain percentage of requests for spare parts should be fulfilled in a pre-specified time window (e.g., two hours). One of the inputs is the time it takes to ship a spare part from each facility to each demand point. A request is not in time if it is assigned either to a facility that cannot ship within the requested time window, or to a facility that can ship in the requested time window, but has no spares on hand. The target fill rate is specified per component, which means that this

is an item approach. Although one may argue that it is a system approach since the fill rate is the average over all locations in the network, this is not a system approach in the sense that we use it.

In the model of Candas and Kutanoglu (2007), multiple components (at one indenture level) are considered, and if a request cannot be fulfilled, it is backordered. The authors formulate an integer programming model that is non-linear due to the fill rate functions: the item fill rate as a function of lead time demand and number of spare parts. The key idea of their solution technique is to approximate the fill rate function for all reasonable spare part stock levels with a function that is piecewise linear in the lead time demand. Since demands are low, the potential fill rates for all reasonable demand levels and reasonable stock levels can be tabulated a priori. This leads to additional variables and constraints, but the non-linearity is removed.

Because of the approximation, their solution does not necessarily fulfil the service requirements (the fill rate in their approximation can be both higher and lower than the actual fill rate). Therefore, to guarantee that the fill rate is achieved for each component, the authors post-process the solution: they keep the decision on which facilities to open and which demand points to connect to each of these facilities, and they solve a 'normal' spare parts stocking problem to determine the amount of spare parts to stock at each facility. The solution that is thus found, necessarily fulfils the service requirements. The authors compare this solution with the solution of solving a facility location problem first, and then a spare parts stocking problem (the so-called sequential approach).

There are two differences between the model that Jeet et al. (2009) use and the model that Candas and Kutanoglu (2007) use: Jeet et al. (2009) assume that any unfulfilled demand is lost (instead of backordered) and they consider a single-item model only. Therefore, this paper is less relevant in the context of our research, and we will not focus on it in more detail.

Both Candas and Kutanoglu (2007) and Jeet et al. (2009) have to approximate the fill rate (or, in the latter case, a variable that is substituted for a couple of variables including the fill rate). For single-indenture product structures and single-echelon repair networks, it may be possible to approximate the EBO in a similar way. However, as explained in Section 2.3, the backorders at higher echelon levels and at higher indenture levels in multi-indenture or multi-echelon problem instances, increase the lead times for requests of LRUs at operating sites. This means that in this setting, we would combine multiple approximations, which may lead to unsatisfactory results. Therefore, the methods that we develop in Chapters 6 and 7 for the joint problem of LORA and spare parts stocking will not be based on the methods of Candas and Kutanoglu (2007) or Jeet et al. (2009).

2.5 Conclusions

We conclude that none of the existing LORA models fits well on practical problem instances. Barros (1998), Barros and Riley (2001), and Saranga and Dinesh Kumar (2006) use very restrictive assumptions on the resource-component relations, whereas Brick and Uchoa (2009) consider two-indenture product structures and single-echelon repair networks only. Aggregating data per echelon level, as Barros (1998), Barros and Riley (2001), and Saranga and Dinesh Kumar (2006) do, may lead to suboptimal solutions if the network structure is unbalanced, but we do not now how big the gap with the optimal solution is. It may be so small that aggregating all data per echelon level is not a problem in practice.

Given the long tradition in spare parts stocking, the current state-of-the-art is sufficient to solve spare parts stocking problems in practice. Improvements are still possible, but in this thesis, we will not focus on these problems.

Only one paper (Alfredsson, 1997) exists in which the problems are integrated, but the author considers single-indenture product structures and two-echelon repair networks only. Besides, the assumptions on the resource-component relations are very restrictive. The methods that are developed for the related problem of simultaneously solving the facility location problem and the spare parts stocking problem (Candas and Kutanoglu, 2007; Jeet et al., 2009) consider single-indenture, single-echelon problems only, and the extension to multi-indenture, multi-echelon problems seems problematic.

It would be most useful to have one method to solve the joint problem of LORA and spare parts stocking for multi-echelon multi-indenture problems with general restrictions on the resource-component relations. However, since such a general model does not even exist for the LORA problem itself, developing such a LORA model is an important first step, which we start with in the next chapter.

Chapter 3

Basic LORA model[1]

In Chapter 2, we discussed the fact that no LORA model exists that fits the requirements that are posed on such models in practice. Therefore, we posed in Section 1.5 research question 2: "What is a suitable LORA model that can be solved in a reasonable amount of time for problem instances with a size that is realistic in practice?" The 'pure' LORA models that are available in the literature (Barros, 1998; Barros and Riley, 2001; Saranga and Dinesh Kumar, 2006) fit multi-indenture, multi-echelon problems, but the restrictions on the resource-component relations are too strict to be used in practice. Development of a model with more general restrictions on the resource-component relations is therefore the first step to take, which leads to research question 2a: "In what way can we generalize the models that are available in the literature?"

In Section 3.1, we present an integer (linear) programming (IP) formulation of the LORA problem. This formulation is intuitive and it closely resembles the models in the literature. We can therefore easily show how it generalizes the existing models. In Section 3.1, we also show that, in general, removing the integrality constraints in the IP model yields a fractional solution. Therefore, we provide in Section 3.2 an improved version of the model in which the integrality constraints on most of the variables can be removed without yielding a fractional solution.[2] This positively influences the time it takes to solve the model.

We use this improved model to show which integrality constraints can be removed if we use the model assumptions of the already existing LORA models. In Section 3.2.3, we show that it is not possible to remove all integrality constraints in the model of Barros (1998), although the author claims this. In Section 3.2.4, we show that if we use the model assumptions of Saranga and

[1] Based on Basten et al. (2009b)

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

Dinesh Kumar (2006) 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 (2006) use genetic algorithms. We further use this model formulation to show, in Appendix B, that the LORA problem is NP-hard.

Section 3.3 provides results for the computational experiments. We base our problem instances on cases that we have seen at Thales Nederland and it turns out that problem instances of a realistic size can be solved using CPLEX in a reasonable amount of time. The chapter ends with conclusions in Section 3.4.

3.1 Model

This section provides our basic IP model. We give the model assumptions in Section 3.1.1, and the notation that we use in Section 3.1.2. A summary of the notation that we use throughout this thesis can be found in Appendix A. In Section 3.1.3, we give the model formulation and in Section 3.1.4 we show why the integrality constraints cannot be removed in this formulation without yielding a fractional solution. Therefore, we propose in Section 3.2 an improved version of the model in which most integrality constraints can be removed.

3.1.1 Model assumptions

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 list the key assumptions below. Most of them were already mentioned in Section 1.2:

- Each time a repair is performed, variable costs are incurred. To be able to perform the repair of a certain component at a certain echelon level, annual fixed costs are incurred. For example, if fixed costs are related to acquiring test equipment, they represent the annual depreciation costs.
- The system itself (indenture level 0) is never moved from its location, but is always repaired by replacing an LRU.
- The repair of a component is in principle 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 modelled at that level. It may also be that the failure in a certain component c is caused by a failure of component b_1 in 30% of the cases, a failure of component b_2 in 40% of the cases, a simultaneous failure of components b_1 and b_2 in 20% 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 required.

- A failed component can be moved only from a certain echelon level e to echelon level $e+1$.
- Combining the previous two assumptions means that if, for example, an LRU is repaired at echelon level 2, the failed SRU that was contained in the LRU, can only be repaired at echelon level $e \geq 2$.
- If the choice is made to repair a certain component at a certain echelon level, the probability of a successful repair is 100%.
- All data at a certain echelon level is aggregated. This means that the exact structure of the repair network is not known by the model. Instead, we use, for example, average repair costs at each echelon level and if there are ten locations at a certain echelon level, either zero or ten resources may be installed at that echelon level.
- As a result of the previous assumption, the repair of a certain component should always be performed at the same echelon level, independent of the operating site from which the component originates. This may be suboptimal in practice; for example, if the repair network is asymmetrical (see Section 2.1.2).
- As a result of the previous assumption, the number of locations between any operating site and the most central location should be equal for all operating sites. For example, it is not possible that one operating site is connected to the central depot directly, whereas another operating site is connected to an intermediate depot, which in turn is connected to the central depot.

3.1.2 Notation

Let the set C consist of all components in the product structure and let I consist of all indenture levels in the product structure. C_i is het set of all components at indenture level i. For example, $C_1 \subseteq C$ is the set of LRUs. Obviously, $\bigcup_i C_i = C$ and $C_i \cap C_j = \emptyset$ if $i \neq j$. A component (parent) may contain subcomponents (children), which are at the next higher indenture level. Γ_c denotes the set of children of component c. This set may be empty (for components at the highest indenture level).

As explained in Section 2.2, Saranga and Dinesh Kumar (2006) assume that fixed costs are due to one component, which means that each component requires its own type of test/repair equipment, whereas Barros (1998) assumes that fixed costs are incurred by all the components at one indenture level. As mentioned in Section 2.1, in practice (e.g., at Thales Nederland), these kind of sets are too restrictive, since equipment is sometimes used by multiple components at various indenture levels, and some components require two or more resources simultaneously. Therefore, we model fairly general sets of components that share fixed costs $G \in \mathcal{G}$ ($G \subseteq C$, $G \neq \emptyset$). Not all components need to be in one of the sets, and components may be in more than one set,

indicating that multiple resources are required simultaneously. If every set contains exactly one component and every component belongs to exactly one set[3], 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[4], the assumptions of Barros are used.

Generally, the repair network consists of multiple echelon levels. These echelon levels form the set E. At each echelon level $e \in E$, except for the highest echelon level e^{CEN}, there are three possible decisions $d \in D$ to take for each component $c \in C$:

- Discard: component c is scrapped and a new one is acquired.
- Repair: component c is repaired by replacing its defective child(ren) $b \in \Gamma_c$ with an operating one (or by repairing c directly). One of the decisions needs to be taken for component b at the same echelon level e.
- Move: component c is moved to echelon level $e+1$. At echelon level $e+1$, a decision needs to be taken. Note that the 'move' option does not exist at the highest echelon level (e^{CEN}).

The set D_e consists of all decisions that are available at echelon level e. So, for all $e \in E$ with $e \neq e^{\text{CEN}}$: $D_e = \{\text{discard}, \text{repair}, \text{move}\}$. Furthermore, $D_{e^{\text{CEN}}} = \{\text{discard}, \text{repair}\}$.

$vc_{c,e,d}$ are the variable costs of taking action d (discard, repair, or move) for one component c at echelon level e. As mentioned before, variable costs include costs for working hours of service engineers, usage of spare parts, and transportation costs. $fc_{G,e,d}$ are the fixed costs that have to be incurred to enable at echelon level e the action d for all components that are in set G. We will also call this 'enabling decision d at echelon level e for set G'. Fixed costs include costs for test and repair equipment, but also training of service engineers. Notice that if a component c is part of both G_1 and G_2, fixed costs for decision d at echelon level e for both these sets need to be taken into account before that decision can be taken for component c.

For LRUs $c \in C_1$, λ_c is the total annual number of failures in LRU c. If an LRU is repaired, then a certain fraction of the repairs leads to a failed SRU for which a decision needs to be taken. If the LRU is discarded however, then no decisions need to be taken for any of the SRUs contained in this LRU. As a result, the observed annual number of failures in components $c \in C \setminus C_1$ depends on the decisions taken for its parent components. We define λ_c such that the observed demand of components $c \in C \setminus C_1$ is either 0 or λ_c. This means that λ_c is an input for all $c \in C$. If multiple children of component c fail at the same time, then it may be that $\sum_{b \in \Gamma_c} \lambda_b > \lambda_c$. Furthermore, $\sum_{b \in \Gamma_c} \lambda_b$ may also be smaller than λ_c if some repairs can be performed directly without replacement of a subcomponent.

[3] $|\mathcal{G}| = |C|$, for all $G \in \mathcal{G}$ it holds that $|G| = 1$, and if $G_1 \neq G_2$, then $G_1 \cap G_2 = \emptyset$.
[4] $|G| = |I|$, for all $G \in \mathcal{G}$ there exists an $i \in I$ such that $G = C_i$, and if $G_1 \neq G_2$ then $G_1 \cap G_2 = \emptyset$.

If for a component b it holds that $b \in \Gamma_{c_1}$ and $b \in \Gamma_{c_2}$ (commonality), we treat b as being two different components b_1 and b_2, with for all G: $b_1 \in G \iff b_2 \in G$. This means that two different decisions may be taken for b_1 and b_2. If, for example, c_1 is repaired at echelon level 1 and c_2 is repaired at echelon level 2, it may be optimal to discard b_1 at echelon level 1 and repair b_2 at echelon level 2.

The model uses two sets of decision variables:

$$X_{c,e,d} = \begin{cases} 1, \text{ if for component } c \in C \text{ at echelon } e \in E \text{ decision } d \in D \text{ is taken} \\ 0, \text{ otherwise} \end{cases}$$

$$Y_{G,e,d} = \begin{cases} 1, \text{ if for all components in set } G \in \mathcal{G} \text{ decision } d \in D \text{ is enabled at} \\ \quad \text{echelon } e \in E \\ 0, \text{ otherwise} \end{cases}$$

3.1.3 Model formulation

We propose the following model formulation:

$$\text{minimize} \sum_{c \in C} \sum_{e \in E} \sum_{d \in D} vc_{c,e,d} \cdot \lambda_c \cdot X_{c,e,d} + \sum_{G \in \mathcal{G}} \sum_{e \in E} \sum_{d \in D} fc_{G,e,d} \cdot Y_{G,e,d} \tag{3.1}$$

subject to:

$$\sum_{d \in D} X_{c,1,d} = 1 \ , \forall c \in C_1 \tag{3.2}$$

$$X_{c,e,\text{move}} \leq \sum_{d \in D_{e+1}} X_{c,e+1,d} \ , \forall c \in C \ , \forall e \in E \mid e \neq e^{\text{CEN}} \tag{3.3}$$

$$X_{c,e,\text{repair}} \leq \sum_{d \in D_e} X_{b,e,d} \ , \forall c \in C \mid \Gamma_c \neq \emptyset \ , \forall b \in \Gamma_c \ , \forall e \in E \tag{3.4}$$

$$X_{c,e,d} \leq Y_{G,e,d} \ , \forall G \in \mathcal{G} \ , \forall c \in G \ , \forall e \in E \ , \forall d \in D \tag{3.5}$$

$$X_{c,e,d}, Y_{G,e,d} \in \{0,1\} \ , \forall c \in C \ , \forall e \in E \ , \forall d \in D \ , \forall G \in \mathcal{G} \tag{3.6}$$

The objective function minimizes the sum of all annual variable and fixed costs. Constraint 3.2 guarantees that a decision is taken for every LRU at echelon level 1. If the move option is chosen for a component at echelon level e, Constraint 3.3 assures that a decision is taken for that component at the next higher echelon level $e+1$. Constraint 3.4 assures that if repair is chosen at an echelon level for a component, a decision is taken for all its child components at that echelon level.

The inequalities in both constraints 3.3 and 3.4 cannot be changed to equalities. To show why, assume that there is a component c with one child component b:

- If c is moved from echelon level 1 to 2, where it is repaired, a decision needs to be taken for b at echelon level 2. This means that $\sum_{d \in D} X_{b,2,d} = 1$. An equality in Constraint 3.3 would then imply that $X_{b,1,\text{move}} = 1$, which is incorrect.

- If c is repaired at echelon level 1 and b is moved to echelon level 2, a decision needs to be taken for b at echelon level 2. This means that $\sum_{d \in D} X_{b,2,d} = 1$. An equality in Constraint 3.4 would then imply that $X_{c,2,\text{repair}} = 1$, which is incorrect.

If discard is chosen for a component, no decision has to be taken 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 (1998) and Saranga and Dinesh Kumar (2006), in which choosing the discard option for a parent component means that discard should also be chosen for all its child components. We believe that it is intuitively more logical that nothing needs to be done with the children if the parent is discarded.

Constraint 3.5 assures that fixed costs are taken into account for set G if a decision is taken for any component $c \in G$.

3.1.4 LP relaxations

The model uses two sets of binary decision variables: $X_{c,e,d}$ and $Y_{G,e,d}$. In this section, we give a problem instance that shows that, in general, the integrality constraints on the $X_{c,e,d}$ variables cannot be removed without yielding a fractional solution.

It is, however, possible to remove the integrality constraint on $Y_{G,e,d}$, since Constraint 3.5 assures that $Y_{G,e,d} = 1$ if any $X_{c,e,d} = 1$ with $c \in G$. If $X_{c,e,d} = 0$ for all $c \in G$, the minimization in the objective function will cause $Y_{G,e,d}$ to be 0.[5] However, we prefer to remove the integrality constraint on $X_{c,e,d}$ (which is possible for the model we give in Section 3.2), since there are generally more $X_{c,e,d}$ than $Y_{G,e,d}$ variables.

To see why the integrality constraint on the $X_{c,e,d}$ variables cannot be removed in our basic model, consider a system consisting of components c_1 and c_2, with c_1 being the parent of c_2. The repair netwerk consists of two echelon levels, and there are no fixed costs for enabling a decision. Table 3.1a shows the variable costs and the demand rates.

Table 3.1b shows the resulting optimal LP solution, which is not an integer solution. The objective value is 1.5, but it 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 integer, see Figure 3.1. The figure shows in a graph which decisions can be taken for components c_1 and c_2. Each displayed arc represents a decision $X_{c,e,d}$. At the top node, only one arc or decision should be chosen, so that the associated $X_{c_1,1,d} = 1$. If two options are chosen simultaneously, both associated $X_{c_1,1,d} = 0.5$. What happens in the example, is that via two ways[6],

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

[6] The two ways are: (1) Component c_1 is moved to echelon level 2 (c_1, 1, move) and is repaired there (c_1, 2, repair). Component c_2 results at echelon level 2 in need for repair. (2) Component c_1 is repaired at echelon level 1 (c_1, 1, repair). Component c_2 results at echelon level 1 in need for repair

Component	c_1	c_2
λ_c	1	1
$vc_{c,1,\text{discard}}$	10	10
$vc_{c,1,\text{repair}}$	1	1
$vc_{c,1,\text{move}}$	0	0
$vc_{c,2,\text{discard}}$	10	10
$vc_{c,2,\text{repair}}$	1	1

(a) Inputs

Component	c_1	c_2
$X_{c,1,\text{discard}}$	0	0
$X_{c,1,\text{repair}}$	0.5	0
$X_{c,1,\text{move}}$	0.5	0.5
$X_{c,2,\text{discard}}$	0	0
$X_{c,2,\text{repair}}$	0.5	0.5

(b) Outputs

Table 3.1: Instance with fractional solution (Section 3.1.4)

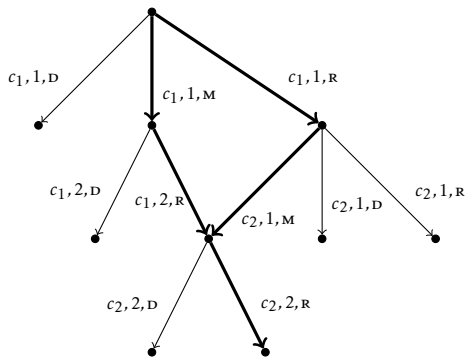


Figure 3.1: Resulting fractional solution (Section 3.1.4)

Each arc in the graph represents a decision $X_{c,e,d}$, with d being D for discard, R for repair, or M for move. The bold arcs represent the options that are selected in the example ($X_{c,e,d} = 0.5$).

component c_2 reaches echelon level 2 in need for repair (the bold arcs in the figure). Constraint 3.3 and Constraint 3.4 ensure that the value of $X_{c_2,2,\text{repair}}$ is greater than or equal to $X_{c_2,1,\text{move}}(= 0.5)$ and $X_{c_1,2,\text{repair}}(= 0.5)$, respectively, which means that $X_{c_2,2,\text{repair}}$ needs to be 0.5 only, although it is clear that $X_{c_2,2,\text{repair}}$ should be 1. Since we explained in Section 3.1.3 that we cannot replace the inequalities in these constraints with equalities, we cannot prevent the problem in this formulation without using the integrality constraints on the $X_{c,e,d}$ variables.

3.2 Improved model

Since the integrality constraints in the model provided in Section 3.1 could not be removed for the $X_{c,e,d}$ variables, we show an improved model in Section 3.2.1. We still use the assumptions outlined in Section 3.1.1. Then, in Section 3.2.2, we show that the integrality constraints on the $X_{c,e,d}$ variables can

and is moved to echelon level 2 ($c_2, 1, \text{move}$).

be removed in the improved model. In Section 3.2.3, we use this result to show which integrality constraints can be removed in the model of Barros (1998); in Section 3.2.4, we uses the result to show that all integrality constraints can be removed when the assumptions of Saranga and Dinesh Kumar (2006) are used in our model.

3.2.1 Model formulation

The improvement in the LORA formulation is inspired by the problem shown in Section 3.1.4. We show the model below, and explain the differences with the basic model afterwards.

$$\text{minimize} \sum_{c\in C}\sum_{e\in E}\sum_{d\in D} vc_{c,e,d}\cdot\lambda_c\cdot X_{c,e,d} + \sum_{G\in\mathcal{G}}\sum_{e\in E}\sum_{d\in D} fc_{G,e,d}\cdot Y_{G,e,d} \tag{3.7}$$

subject to:

$$\sum_{d\in D} X_{c,1,d} = 1\ ,\forall c\in C_1 \tag{3.8}$$

$$X_{c,e,\text{move}} = \sum_{d\in D_{e+1}} X_{c,e+1,d}\ ,\forall c\in C_1\ ,\forall e\in E \mid e\neq e^{\text{CEN}} \tag{3.9}$$

$$X_{c,1,\text{repair}} = \sum_{d\in D_1} X_{b,1,d}\ ,\forall c\in C \mid \Gamma_c\neq\emptyset\ ,\forall b\in\Gamma_c \tag{3.10}$$

$$X_{c,e+1,\text{repair}} + X_{b,e,\text{move}} = \sum_{d\in D_{e+1}} X_{b,e+1,d}\ ,\forall c\in C \mid \Gamma_c\neq\emptyset\ ,\forall b\in\Gamma_c\ ,\forall e\in E \mid e\neq e^{\text{CEN}} \tag{3.11}$$

$$X_{c,e,d}\leq Y_{G,e,d}\ ,\forall G\in\mathcal{G}\ ,\forall c\in G\ ,\forall e\in E\ ,\forall d\in D \tag{3.12}$$

$$X_{c,e,d}, Y_{G,e,d}\in\{0,1\}\ ,\forall c\in C\ ,\forall e\in E\ ,\forall d\in D\ ,\forall G\in\mathcal{G} \tag{3.13}$$

There are four differences with the original model:

- Constraint 3.9 is similar to Constraint 3.3, but is used for the LRUs ($c\in C_1$) only, instead of for all components.
- Constraint 3.10 is similar to Constraint 3.4, but is used for echelon level 1 only, instead of for all echelon levels.
- Constraint 3.11 is added to deal with the problem shown in Section 3.1.4. It combines Constraints 3.9 and 3.10 in that it assures that a decision is taken for a child component if it is either moved from a lower echelon level, or its parent component is repaired at the current echelon level.
- Constraints 3.3 and 3.4 are inequalities (and cannot be changed to equalities, see Section 3.1.3), but Constraints 3.9, 3.10 and 3.11 are equalities.

Component	c_1	c_2	c_3
λ_c	1	1	1
$vc_{c,1,\text{discard}}$	100	0	0
$vc_{c,1,\text{repair}}$	0	100	0
$vc_{c,1,\text{move}}$	0	0	100
$vc_{c,2,\text{discard}}$	0	0	0
$vc_{c,2,\text{repair}}$	0	0	0

(a) Variable costs and annual demand

Set	G_1
$fc_{G,1,\text{discard}}$	100
$fc_{G,1,\text{repair}}$	100
$fc_{G,1,\text{move}}$	100
$fc_{G,2,\text{discard}}$	0
$fc_{G,2,\text{repair}}$	0

(b) Fixed costs

Table 3.2: Inputs for instance with fractional solution (Section 3.2.2.1)

Component	c_1	c_2	c_3
$X_{c,1,\text{discard}}$	0	0.5	0.5
$X_{c,1,\text{repair}}$	0.5	0	0.5
$X_{c,1,\text{move}}$	0.5	0.5	0
$X_{c,2,\text{discard}}$	0.5	0.5	0
$X_{c,2,\text{repair}}$	0	0	0

(a) $X_{c,e,d}$

Set	G_1
$Y_{G,1,\text{discard}}$	0.5
$Y_{G,1,\text{repair}}$	0.5
$Y_{G,1,\text{move}}$	0.5
$Y_{G,2,\text{discard}}$	0.5
$Y_{G,2,\text{repair}}$	0

(b) $Y_{G,e,d}$

Table 3.3: Outputs for instance with fractional solution (Section 3.2.2.1)

3.2.2 LP Relaxations

The model uses two sets of binary decision variables: $X_{c,e,d}$ and $Y_{G,e,d}$. In Section 3.2.2.1, we show that we cannot remove the integrality constraint on both sets of variables without yielding a fractional solution. However, we show in Section 3.2.2.2 that we can remove the integrality constraint on the $X_{c,e,d}$ variables.

3.2.2.1 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, we consider a two-echelon repair network and three LRUs (c_1, c_2, c_3) without child components. The LRUs share fixed costs, so $G_1 = \{c_1, c_2, c_3\}$. Table 3.2a gives the annual demand rate per component and the variable costs per repair action. Table 3.2b gives the fixed costs.

The optimal LP solution value for this instance is 150, but the optimal IP solution value is 200. Tables 3.3a and 3.3b show the values of $X_{c,e,d}$ and $Y_{G,e,d}$ in the optimal LP solution. The optimal IP solution can be achieved in multiple

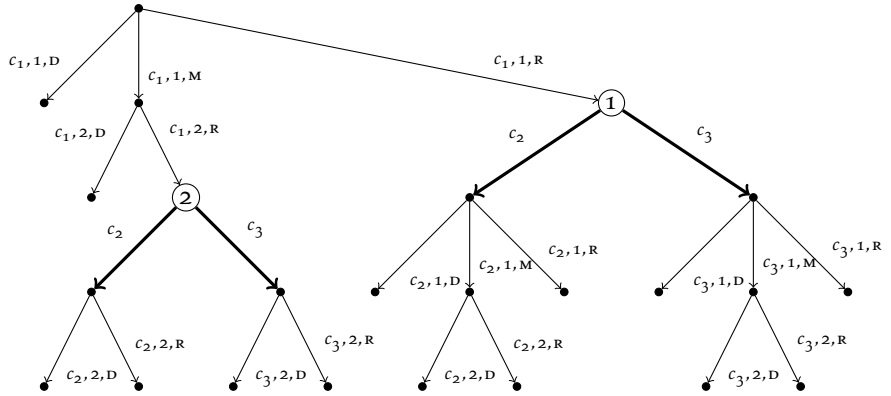


Figure 3.2: Decision tree: three components, two echelon and indenture levels
Each non-bald arc in the graph represents a decision $X_{c,e,d}$, with d being D for discard, R for repair, or M for move. The bold arcs show that below node 1 and 2, decisions need to be taken simultaneously for components c_2 and c_3.

ways. Since repair and discard on echelon level 2 do not incur any costs, we can focus on the three decisions at echelon level 1:

- Enabling one decision leads to fixed costs of 100. Depending on the option we would open, one component would incur variable costs of 100. Fixed costs and variable costs together would be 200.
- Opening two or more decisions 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.

3.2.2.2 Removing integrality constraints on the $X_{c,e,d}$ variables

In this section, we discuss removing the integrality constraints on the $X_{c,e,d}$ variables and we consider the resulting optimal solution. We show that all $X_{c,e,d}$ 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 three components, components c_1, c_2, and c_3, with $\Gamma_{c_1} = \{c_2, c_3\}$, and we have a two-echelon repair network. Figure 3.2 shows the decision tree for the repair options of the system. If the decision repair is chosen for c_1 at either echelon levels 1 or 2, a decision needs to be taken for both c_2 and c_3, which is indicated by the bold arcs originating at node 1 and 2 respectively. We show below that the optimal decisions 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 level 1 and 2, respectively).

We consider the optimal solution of the IP model in which the integrality constraints on the $X_{c,e,d}$ variables are removed. The $Y_{G,e,d}$ variables are still binary. Since we are looking at the optimal solution, it is fixed, for example, at which echelon level test equipment is available and at which echelon level it is not. This in turn means that not all decisions may be possible anymore: in Figure 3.2, not all arcs can be chosen.

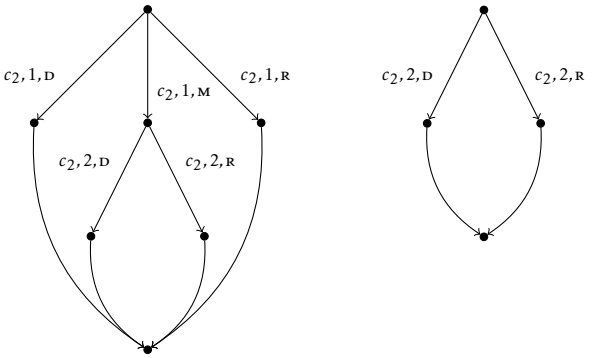
We need one further observation: components at the same indenture level can only be connected to each other by their parent component (or a parent of a parent et cetera) and by the sets of components sharing fixed costs ($\mathcal{G}$). It follows that given a repair decision for all the parents and given the values for $Y_{G,e,d}$, decisions for components at the same indenture level can be made independently. The LRUs do not have a parent component modelled, 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 flow problem (refer to Figure 3.2 if required). Figure 3.3a shows the network that is used in the minimum cost flow problem for component c_2, given that c_1 is repaired at echelon level 1. If c_1 is repaired at echelon level 2, the network for c_2 is shown in Figure 3.3b. The capacity of an arc is 1 (effectively uncapacitated) if the associated decision is enabled (if $Y_{G,e,d} = 1$, $\forall G \in \mathcal{G}$ with $c_2 \in G$). The capacity is 0 otherwise. The costs of using an arc are equal to the associated variable costs times the associated annual demand ($vc_{c_2,e,d} \cdot \lambda_{c_2}$).

It is well known in the literature that a minimum cost flow problem has an optimal integer solution, provided that all capacities, supplies and demands are integer (see, e.g., Ahuja et al., 1993). 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 $X_{c_2,e,d} \in \{0, 1\}$.

The reasoning for component c_3 goes analogous to the reasoning for c_2 in the previous two paragraphs. What may differ are the capacities of the arcs and the costs for using the arcs.

We add the sum of the costs of the best decisions of c_2 and c_3 at echelon level 1 (the optimal solution for both c_2 and c_3, see Figure 3.3a) to the costs of repairing c_1 at echelon level 1 (the arc ending in node 1 in Figure 3.2). In the same way, we add the sum of the costs of the best decisions of c_2 and c_3 at echelon level 2 (Figure 3.3b) to the costs of repairing c_1 at echelon level 2 (the arc ending in node 2 in Figure 3.2). The result is that for component c_1, we have a similar network as shown in Figure 3.3a (replace c_2 with c_1). Since c_1 is an LRU in our example, Constraint 3.8 assures an inflow of 1. With a reasoning analogous to the reasoning in the previous paragraph, this shows that all $X_{c_1,e,d} \in \{0, 1\}$, and



(a) Two echelon levels (b) One echelon level

Figure 3.3: Minimum cost flow models

Each arc in the graph represents a decision $X_{c,e,d}$, with d being D for discard, R for repair, or M for move. An arc has capacity 1 if the associated decision is feasible (given the values of $Y_{G,e,d}$). The capacity is 0 otherwise.

therefore $X_{c,e,d} \in \{0, 1\}$ for $c \in \{c_1, c_2, c_3\}$.

It may happen that some of the minimum cost flow problems for component c_2 or c_3 do not have a feasible solution (Figures 3.3a and 3.3b). This happens if no path through the network has a capacity of more than 0, due to the values of the $Y_{G,e,d}$ variables. These networks originate in node 1 or 2 in Figure 3.2, 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 c_2 and c_3 to the arc corresponding to $X_{c_1,e,\text{repair}}$ (for $e \in \{0, 1\}$), this arc gets a capacity of 0. However, in the minimum cost flow problem for each LRU (c_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 echelon levels, 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 $X_{c,e,d}$ variables and it is still guaranteed that there exists an optimal solution in which all $X_{c,e,d} \in \{0, 1\}$, provided that all $Y_{G,e,d} \in \{0, 1\}$. If any $X_{c,e,d} \notin \{0, 1\}$ 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).

3.2.3 Model of Barros

Barros (1998) 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 echelon levels or more than two indenture levels. Although Barros states that

Component	c_1	c_2	c_3
λ_x	1	1	1
$vc_{c,\text{discard}}$	200	200	200
$vc_{c,\text{repair at 1}}$	100	0	0
$vc_{c,\text{repair at 2}}$	0	100	0
$vc_{c,\text{repair at 3}}$	0	0	100

(a) Variable costs and annual demand

Set	G_1
$fc_{G,\text{discard}}$	0
$fc_{G,\text{repair at 1}}$	100
$fc_{G,\text{repair at 2}}$	100
$fc_{G,\text{repair at 3}}$	100

(b) Fixed costs

Table 3.4: Inputs for instance with fractional solution (Section 3.2.3)

her model can be used for any number of echelon levels and indenture levels, we know from Gutin et al. (2006) that Barros tests her model for the case of two echelon levels and two indenture levels only. We cannot find a counter example for that specific case. Furthermore, Gutin et al. (2006) show that this specific problem can be solved in polynomial time, by reducing it to the maximum weight independent set problem on a bipartite graph. They use the fact that the the problem can be represented as a minimum cost homomorphism problem on a monotone bipartite graph.

The example used in the current section resembles the example given in Section 3.2.2.1. Table 3.4a shows the annual demand rate and the variable costs, Table 3.4b 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 level e is incorporated in the decision d in Barros' model.
- Barros assumes that no fixed costs need to be incurred for opening the discard option.
- Barros assumes one discard option only.
- Barros does not distinguish the move option. Costs for moving a component are part of the costs of repairing that component at the higher echelon level.
- Fixed costs are incurred by all the components at one indenture level. In our case, this means that all components are in the same set G_1, because they are all at indenture level 1.

The resulting outputs are shown in Tables 3.5a and 3.5b. The explanation of the results is analogous to the explanation of the results in Section 3.2.2.1 and is therefore not repeated.

3.2.4 Model of Saranga and Dinesh Kumar

Saranga and Dinesh Kumar (2006) assume that fixed costs are due to one

Component	c_1	c_2	c_3
$X_{c,\text{discard}}$	0	0	0
$X_{c,\text{repair at 1}}$	0	0.5	0.5
$X_{c,\text{repair at 2}}$	0.5	0	0.5
$X_{c,\text{repair at 3}}$	0.5	0.5	0

(a) $X_{c,e,d}$

Set	G_1
$Y_{G,\text{discard}}$	0
$Y_{G,\text{repair at 1}}$	0.5
$Y_{G,\text{repair at 2}}$	0.5
$Y_{G,\text{repair at 3}}$	0.5

(b) $Y_{G,d}$

Table 3.5: Outputs for instance with fractional solution (Section 3.2.3)

component. Therefore, these fixed costs are not really different from variable costs. We can construct 'new' variable costs $vc'_{c,e,d}$ from the 'old' variable costs and fixed costs in the following way (remember that in our model, fixed costs are the mean annual fixed costs): $vc'_{c,e,d} = vc_{c,e,d} + \frac{fc_{G,e,d}}{\lambda_c}$ with $G = \{c\}$. Using these new variable costs, the new fixed costs are zero. If all fixed costs are zero, all $Y_{G,e,d}$ variables can be removed from the model (or set to 1). Section 3.2.2 shows that no integrality constraints are needed on the $X_{c,e,d}$ variables if all $Y_{G,e,d} \in \{0, 1\}$. 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 (2006). Using genetic algorithms for these problem instances, which Saranga and Dinesh Kumar do, is therefore not necessary using our model formulation.

3.3 Computational experiments

To test the model, we generate instances of the LORA problem and solve 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 use only one core of the dual core processor.

In Section 3.3.1, we explain how we generate the test instances. In Section 3.3.2, we then provide the inputs that we use and we discuss some issues concerning the actual testing. We show the results of the tests in Section 3.3.3.

3.3.1 Problem instance generator

In this section, we explain the basic idea of our problem instance generator. Based on our experience at Thales Nederland, we believe that the size and structure of the problem instances is realistic in practice. More extensive information can be found in Appendix C.

Our problem instance generator receives as inputs the number of components ($|C|$), the number of indenture levels ($|I|$), the number of echelon lev-

els ($|E|$), the number of fixed costs sets ($|\mathcal{G}|$)[7], and the maximum number of fixed costs sets that each component will be part of ($s^{\max}$). For each number of fixed costs sets s with $0 \leq s \leq s^{\max}$, a percentage P_s has to be specified, such that $\sum_{s=0}^{s^{\max}} P_s = 100\%$. 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 ($s^{\max} = 1$), 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 one fixed costs set. These percentages should add up to 100%.

Depending on the number of components and indenture levels, we calculate how many children every parent component should have approximately, in order to get a 'balanced' system structure. This means that the average number of child components per parent component is the same at all indenture levels.

The last inputs are the minimum and maximum values for $vc_{c,e,d}$, $fc_{G,e,d}$, and λ_c. The actual values are drawn from a uniform distribution ranging from the provided minimum to the provided maximum. We adapt the $vc_{c,e,\text{discard}}$ and λ_c by adding the values of the child components to the values of their parents (so, $\lambda_c > \sum_{b \in \Gamma_c} \lambda_b$).

3.3.2 Inputs and general issues

In each of our tests, we vary one parameter only. The other parameters get their default values, which are: $|C| = 1{,}000$, $|I| = 3$, $|E| = 3$, $|\mathcal{G}| = 100$, and $s^{\max} = 2$. For each parameter value, we generate 1,000 problem instances.

If the maximum number of fixed costs sets that any component may be part of is set to 2 ($s^{\max} = 2$), then 10% of the components will not be in any fixed costs set, 10% will be in one of those sets, and 80% will be in two of those sets. In general: for any number of fixed costs sets s with $0 \leq s < s^{\max}$, 10% of the components will be in that number of sets. As a result, $100\% - s^{\max} \cdot 10\%$ of the components will be in the maximum number of fixed costs sets ($s^{\max} \leq 5$ in our tests).

In all the tests, we set the minimum and maximum input values for $vc_{c,e,d}$ to 50 and 1,000 respectively, for $fc_{G,e,d}$ to 500 and 10,000, and for λ_c to 0.05 and 5.

As explained in Section 3.1, our model generalizes the models of Barros (1998) and Saranga and Dinesh Kumar (2006). The former assumes that fixed costs are incurred by all the components at one indenture level; the latter assumes that fixed costs are incurred by one component. In our model, fixed costs are incurred 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

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

# Components	500	1,000	2,000	5,000	10,000	20,000
Gen.	2.43[a]	4.06[b]	12.42[c]	117.60[d]	—	—
Barros	0.10	0.31	1.03	5.03	16.36	55.34
SDK	0.03	0.08	0.21	0.74	2.09	6.12

Table 3.6: Computation times (seconds), varying the number of components

[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

Saranga and Dinesh Kumar). For the 'SDK' tests, we solve the model as an LP problem, as explained in Section 3.2.4. In all other cases, we model $Y_{G,e,d}$ as binary variables.

In some cases, solving the problem instances takes so much time, that we restrict 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 exceed the time limit, which only happens for 'Gen.' tests. We exclude these problem instances from the calculations of the computation times. At the end of Section 3.3.3, we discuss the problem instances that exceed the time limit. For now, it suffices to mention that we find feasible solutions for all of them.

3.3.3 Results

Table 3.6 shows the mean computation times for various numbers of components in the system. In Tables 3.7 and 3.8, we vary the number of indenture levels and echelon levels respectively. 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 echelon levels increases. The run times increase strongly if the number of indenture levels increases from one to two. This is logical, since one indenture level means that components are not connected to each other in the product structure (they are all LRUs). 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 two to three for the 'Gen.' tests.

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 (Saranga and Dinesh Kumar, 2006) or branch-and-bound techniques (Barros and Riley, 2001) . 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

# Indenture levels	1	2	3	4	5
Gen.	0.26	4.77[a]	4.06[b]	5.69[c]	8.89[d]
Barros	0.18	0.27	0.31	0.43	0.53
SDK	0.03	0.06	0.08	0.09	0.11

Table 3.7: Computation times (seconds), varying the number of indenture levels

[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

# Echelon levels	1	2	3	4	5
Gen.	0.23	1.83	4.06[a]	6.48[b]	7.60[c]
Barros	0.04	0.16	0.31	0.46	0.53
SDK	0.01	0.05	0.08	0.11	0.14

Table 3.8: Computation times (seconds), varying the number of echelon levels

[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

is that components are more 'connected' in the 'Gen.' tests. For example, if $G_1 = \{c_1, c_2\}$ and $G_2 = \{c_2, c_3\}$, a change in the decision of c_1 can change the best option for c_3.

The findings in Table 3.9, in which we vary the maximum number of fixed costs sets per component ($s^{\max}$), support this assumption. Notice that the computation times increase a lot if the maximum number of fixed costs sets per component increases from one to two. This is not surprising, since one 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 described in the previous paragraph. Notice however, that they are still connected to other components in the product structure.

$s^{\max}$	1	2	3	4	5
Gen.	0.64	4.06[a]	5.94[b]	7.15[c]	8.89[d]

Table 3.9: Computation times (seconds), varying the maximum number of fixed costs sets of which a component can be part of

[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

# Sets	25	50	100	250	500
Gen.	1.63	3.01[a]	4.06[b]	11.60[c]	2.41[d]

Table 3.10: Computation times (seconds), varying the total number of fixed costs sets

[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

In order to be complete, Table 3.10 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 test what happens 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 two sets each and there are 1,000 sets, there will be on average less than two components per set. With 250 sets, there will be a little less than eight components per set. If $G_1 = \{c_1, c_2\}$ and $G_2 = \{c_2, c_3\}$, a change in the decision of c_1 can change the best option for c_3. This will probably happen more often if there are eight components per set than if there are two 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 calculate the gap between the best IP solution that was found at the moment that the solver is stopped, and the best lower bound that CPLEX has found at that moment. The gap is mostly below 2%, with exceptional cases of gaps up to 6.6%. It also happens 15 times (out of the 20,000 'Gen.' tests that we performed) that no IP solution is found before the test is stopped. 14 of these tests were problem instances with 5,000 components, the other test is 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 exceed the time limit (both those for which we already 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 five echelon levels, 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 ten 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. However, if the LORA is one building block in an iterative method to solve the joint problem of LORA and spare parts stocking (such as we develop in Chapter 6), such an optimization time is too long. We believe that a gap of below 2% is not problematic, since the input data generally consists of rather rough estimates.

3.4 Conclusions

We developed a LORA model that generalizes the two LORA models that exist in the literature (Barros, 1998; Saranga and Dinesh Kumar, 2006). 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 incurred 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 (2006) are used, so that there is no need to solve problem instances using genetic algorithms, as they do. We also showed that it is not possible to remove all integrality constraints in the model of Barros (1998), which the author claims.

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. The computation times also increase if any of the following increases: the number of indenture levels in the system, the number of echelon levels in the repair network, and 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 up to over 100 times larger than the computation time for models restricted to the assumptions of Barros or, especially, Saranga and Dinesh Kumar.

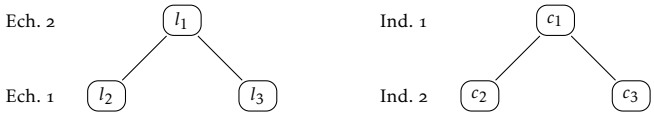
Chapter 4

Flow model for the LORA problem[1]

In the previous chapter, we presented a LORA model that generalizes the models that are available in the literature. We thus answered research question 2a. In this chapter we take the first step in answering research question 2b: "How can we model the extensions that may be needed in practice?" Although the formulation that we presented in the previous chapter is very intuititve, and allowed us to study the models of Barros (1998) and Saranga and Dinesh Kumar (2006), modelling extensions in this formulation is not straightforward. The key problem is that incorporating an extension, e.g., a probability of unsuccessful repair, leads to constraints that are very complicated. Incorporating multiple extensions simultaneously is even more problematic. Therefore, we reformulate in this chapter the LORA model as a minimum cost flow model with side constraints. This formulation generalizes the model that we presented in the previous chapter, by allowing for more general resource-component relations and for modelling the exact repair network, and it allows us to model extensions in an intuitive way, as will be shown in Chapter 5.

The structure of this chapter is as follows. In Section 4.1, we discuss the inputs that the model needs and the assumptions that we make. Then, in Section 4.2, we present the LORA problem as a minimum cost flow model with side constraints. The results of the computational experiments can be found in Section 4.3, and the chapter ends with conclusions in Section 4.4.

[1] Based on Basten et al. (2008)



(a) Repair network (b) Product structure

Figure 4.1: Example used throughout this chapter

4.1 Model assumptions and input data

The assumptions that we make and the input data that we require are very similar to those in the previous chapter, see Section 3.1. A summary of the notation can be found in Appendix A.

The key difference is that instead of aggregating all data per echelon level, as we did in Chapter 3, we model a multi-echelon repair network, with L being the set of all locations. The structure is divergent, that is, each location in the network has a single upstream location to which it can move its failed components. All locations that supply failed components to location l constitute the set Φ_l. For example, in Figure 4.1a, $\Phi_{l_1} = \{l_2, l_3\}$. The set of locations at echelon level e is L_e. We define the echelon levels such that the operating sites are at echelon level 1. The locations that are a parent-location of operating sites (locations in L_1) only, form the set L_2. The locations that are a parent-location of locations in L_1 and L_2 only, form the set L_3, et cetera. As a result, if we add to the network in Figure 4.1a two locations (l_0 and l_4) such that $\Phi_{l_0} = \{l_1, l_4\}$ (l_0 is the central depot and l_4 is an operating site), then $L_1 = \{l_2, l_3, l_4\}$, $L_2 = \{l_1\}$, and $L_3 = \{l_0\}$.

Modelling the explicit repair network means that instead of $vc_{c,e,d}$, we now use $vc_{c,l,d}$ as the variable costs of taking action d (discard, repair, or move) for one component c at location l ($\in L$). Next, instead of using λ_c, we now define $\lambda_{c,l}$ as the annual failure rate of component $c \in C$ at operating site $l \in L_1$ (in Chapter 6, we will define $\lambda_{c,l}$ for locations $l \in L \setminus L_1$). Furthermore, we define $q_{c,b}$ as the fraction of failures in component c that is due to a failure in component b ($b \in \Gamma_c$): $q_{c,b} = \frac{\lambda_{b,l}}{\lambda_{c,l}}$. This means that we asssume that although the number of failures may differ over the various operating sites (due to various usage intensities), the factors $\frac{\lambda_{b,l}}{\lambda_{c,l}}$ do not change. If a failure of a certain component c is caused by a failure of component b_1 in 30% of the cases, a failure of component b_2 in 40% of the cases, a simultaneous failure of components b_1 and b_2 in 20% of the cases, and a failure that can be repaired directly in 10% of the cases, then $q_{c,b_1} = 0.5$ and $q_{c,b_2} = 0.6$

Instead of modelling sets of components sharing fixed costs, which we did in the previous chapter in order to resemble the models of Barros (1998) and Saranga and Dinesh Kumar (2006), we make a generalization by modelling resources $r \in R$. Let Ω_r consist of all combinations of a component c and

a decision d for which resource r is required. So, if component c requires resource r in order to enable decision d, then the 2-tuple $(c,d) \in \Omega_r$[2]. One tuple may be an element of multiple sets Ω_r, indicating that more resources are required simultaneously. We assume that the resource capacity is infinite, although we can extend our model to finite capacities (see Chapter 5).

Without loss of generality, we have chosen to minimize the average total costs per year with our definition of $\lambda_{c,l}$. Therefore, we define $fc_{r,l}$ to be the annual fixed costs to locate resource r at location l. These costs represent, for example, the annual depreciation costs of the resource and costs of capital.

Finally, instead of decision variables $X_{c,e,d}$, we now use:

$$X_{c,l,d} = \begin{cases} 1, \text{ if for component } c \in C \text{ at location } l \in L \text{ decision } d \in D \text{ is taken} \\ 0, \text{ otherwise} \end{cases}$$

4.2 Minimum cost flow model

In this section, we explain how the LORA problem can be modelled as a minimum cost flow model with side constraints (see, for example, Ahuja et al., 1993, for an overview of minimum cost flow models). To define the graph $G = (V, A)$ underlying the flow model, with V being the set of all nodes in the graph and A being the set of all arcs in the graph, we need four different node types: *source nodes* ($v \in V^{\mathrm{s}} \subseteq V$) are used to represent the occurrence of failures of LRUs (indenture level 1) at operating sites (echelon level 1). The flows from these source nodes arrive at *decision nodes* ($v \in V^{\mathrm{d}} \subseteq V$) where a decision is made between the three available options: discard, repair, and move. The variable costs are attached to the outgoing arcs of the decision node, each representing a possible decision. If repair is chosen, then a *transformation node* ($v \in V^{\mathrm{t}} \subseteq V$) is used to represent that a failure in a parent is due to a failure in any of the children. If no decisions need to be made anymore, the flow goes to a *sink node* (we do not need a distinct subset for the sink nodes). We model the use of resources by side constraints on the minimum cost flow model: if the outgoing arc of a decision node represents a decision for a component that can only be chosen if a resource is available, then the capacity of this arc is 0 if the resource is not available.

In Section 4.2.1, we explain how to construct the graph that forms the basis of the flow model. Then, in Section 4.2.2, the resources are added to the model as side constraints. Finally, we provide the formal model formulation in Section 4.2.3.

[2] Modelling the sets of components sharing fixed costs that we used in Chapter 3 can be done as follows. For each of the three possible decisions $d \in D$ model one resource $r \in R$ for each set of components sharing fixed costs $G \in \mathcal{G}$ and let $(c,d) \in \Omega_r \iff c \in G$. This means that $|R| = 3 \cdot |\mathcal{G}|$.

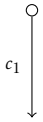


Figure 4.2: Source node
Failures of component c_1 originate at the source node.

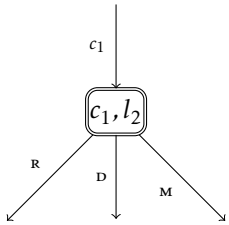


Figure 4.3: Decision node
Failures of component c_1 at location l_2 go into the decision node. Decisions R (repair), D (discard), and M (move) can be taken.

4.2.1 Construction of the graph

In this section, we explain how the node types are used in the model and we define the incoming and outgoing arcs (v, w) and the relevant cost parameters. To illustrate our model, we use the following example throughout this section: we have a two-indenture system with three components ($\Gamma_{c_1} = \{c_2, c_3\}$, see Figure 4.1b) and a two-echelon repair network with three locations ($\Phi_{l_1} = \{l_2, l_3\}$, see Figure 4.1a). We show an example of each node type, and after we have introduced all the node types, we show the complete flow model. In all figures related to this example, some arcs represent a component, whereas a letter next to an arc represents an option (R is repair, D is discard, and M is move).

4.2.1.1 Source nodes

Source nodes represent occurrences of failures of a certain LRU at a certain operating site. So, for every component $c \in C_1$ and every location $l \in L_1$, there is one source node $v \in V^s \subseteq V$ with outflow $o_v := \lambda_{c,l}$. In our example, failures in component c_1 occur at locations l_2 and l_3. This means that we have two source nodes in our flow model, one of which is shown in Figure 4.2. If we assume that this source node represents failures in component c_1 at location l_2, then the flow out of this source node is equal to λ_{c_1,l_2}.

4.2.1.2 Decision nodes

If an LRU at an operating site fails, there are three options to choose from:

- Move the component to the next higher echelon level.
- Repair the component, which means replacing a subcomponent or repairing the component directly.
- Discard the component.

In the flow model, it means that an arc originating at a source node terminates at a decision node $v \in V^{\mathrm{d}} \subseteq V$.[3] Every arc going out of the decision node represents one of the possible decisions. The variable costs, $ac_{v,w}$, for using an arc (v,w) with $v \in V^{\mathrm{d}}$ are equal to $vc_{c,l,d}$, where arc (v,w) represents decision d for component c at location l. In this way, the variable costs of the LORA model are attached to the arcs originating at the decision nodes; all other arcs have zero costs associated to them ($ac_{v,w} = 0, \forall v,w \in V \mid v \notin V^{\mathrm{d}}$). Figure 4.3 shows the decision node for component c_1 at location l_2 in our example. If arc (v,w) represents the decision repair (R), then the variable costs for using this arc, $ac_{v,w}$, are equal to the variable costs of repair for component c_1 at location l_2: $vc_{c_1,l_2,\mathrm{repair}}$.

If the decision is taken to move component c_1 from location l_2 to location l_1, a decision should to be taken at location l_1. This means that the arc representing the move option (M) in our example (Figure 4.3) terminates at the decision node representing component c_1 at location l_1. In general, the arc representing the move decision for component c at location k terminates at the decision node representing component c at location l, where $k \in \Phi_l$. Note that at a node representing a component at the highest echelon level, location l_1 in our example, the move option is not available. The arcs representing the repair and discard options are discussed below.

4.2.1.3 Transformation nodes

Transformation nodes $v \in V^{\mathrm{t}} \subseteq V$ represent the repair of a parent component c by replacement of any of the subcomponents $b \in \Gamma_c$. If arc (v,w) represents component $b \in \Gamma_c$ resulting from a repair of component c, then the fraction of inflow in node v that flows out on arc (v,w) is defined as $p_{v,w}$, and $p_{v,w} := q_{c,b}$. Notice that the total inflow and outflow of a transformation are not necessarily equal, which is not common in minimum cost flow models.

In our example, a failure in component c_1 can be caused by a failure in component c_2 or c_3 ($\Gamma_{c_1} = \{c_2, c_3\}$). The arc representing the decision repair (R) for component c_1 at location l_2 (in Figure 4.3) terminates at the transformation node that is shown in Figure 4.4. The two arcs originating at the transformation node represent failures of components c_3 and c_2 respectively. Suppose that 50% of those failures are caused by failures in component c_2 and 40% are caused by failures in component c_3 ($q_{c_1,c_2} = 0.5$ and $q_{c_1,c_3} = 0.4$), then, if the outgoing arcs (v,w) and (v,u) represent components c_3 and c_2 respectively, then $p_{v,w} = 0.4$ and $p_{v,u} = 0.5$. Notice that no further decisions need to be taken for the failures that are repaired directly, which is why this is not modelled in the transformation node.

[3] The source node could also be integrated in the decision node, but for clarity we prefer to have a distinction between source and decision nodes.

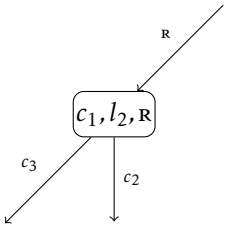


Figure 4.4: Transformation node
The flow going into the node results from a repair (R) of component c_1 at location l_2, components c_3 and c_2 flow out of the node.

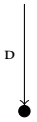


Figure 4.5: Sink node
The flow that is discarded (D) goes into the sink.

4.2.1.4 Sink nodes

If no other decisions need to be taken after a certain decision has been made, then the flow goes to a sink node. In the example, arcs that terminate at a sink node represent the decision 'discard' for any component at any location and the decision 'repair' for component c_2 or c_3 at any location. Figure 4.5 shows how we represent a sink node.

4.2.1.5 Example

We already showed parts of the flow model that results from the LORA problem that we used as an example throughout this section. Figure 4.6 shows the complete resulting flow model. The dotted arcs for component c_3 should be replaced by flows similar to the flows for component c_2. We omitted them to improve the readability of the figure.

4.2.2 Modelling resources as side constraints

In some cases, resources $r \in R$ are required before a certain decision can be taken for certain components. $\Theta_{r,l}$ is the set of arcs that are enabled due to the location of resource r at location l. So, for each location l and every resource r we define $\Theta_{r,l} = \{(v,w) \mid (v,w)$ denotes decision d for component c at location l if $(c,d) \in \Omega_r\}$. Notice that fixed costs in our model are only related to the arcs originating at the decision nodes (the same holds for the variable costs, as explained in Section 4.2.1.2).

For example, resource r_1 may be required if and only if component c_1 is to be repaired. In that case, $\Omega_{r_1} = \{(c_1, \text{repair})\}$. At all three locations in our example repair network, we may decide to locate resource r_1. If resource r_1 is available at location l_3, then the arc representing repair of component c_1 at location l_3 is enabled. If this is the arc (v,w), then $\Theta_{r_1,l_3} = \{(v,w)\}$.

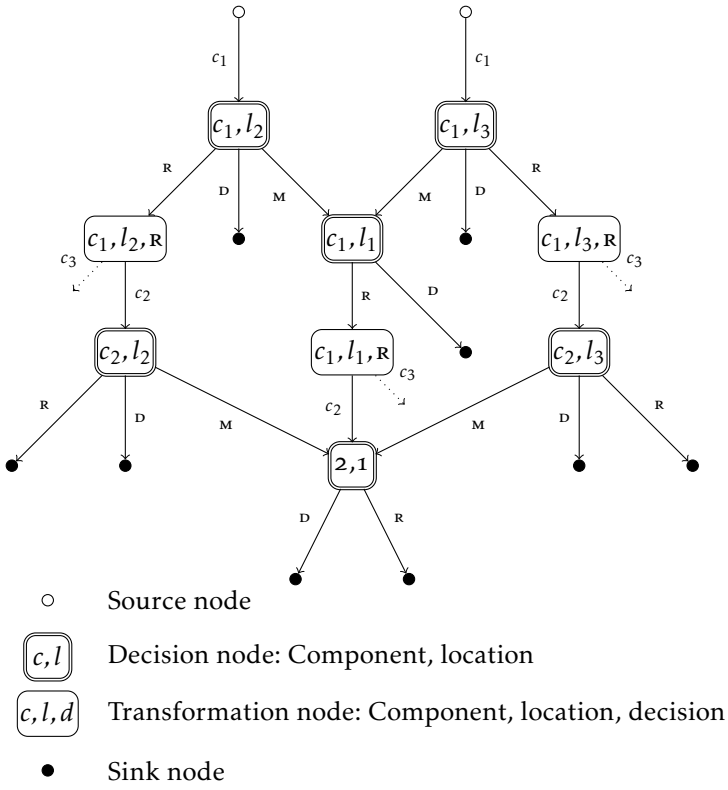


Figure 4.6: Example: Network flow problem

Some of the arcs represent components c, other arcs represent a decision that can be taken, with R is repair, D is discard, and M is move.

4.2.3 Flow model formulation

In our model, there are two sets of decision variables:

$$F_{v,w} = \text{the amount of flow through arc } (v,w)$$

$$Y_{r,l} = \begin{cases} 1, \text{ if resource } r \text{ is located at location } l \\ 0, \text{ otherwise} \end{cases}$$

Furthermore, we introduce a big M that is used to make the resources uncapacitated. The value of the big M is set for each arc (v,w) such that it is equal to the maximum possible value of $F_{v,w}$. The resulting minimum cost flow model with side constraints is:

$$\text{minimize} \sum_{(v,w)\in A} ac_{v,w} \cdot F_{v,w} + \sum_{r\in R}\sum_{l\in L} fc_{r,l} \cdot Y_{r,l} \tag{4.1}$$

subject to:

$$F_{v,w} = o_v \ , \forall (v,w) \in A \mid v \in V^{\mathrm{s}} \tag{4.2}$$

$$\sum_{u|(u,v)\in A} F_{u,v} = \sum_{w|(v,w)\in A} F_{v,w} \ , \forall v \in V^{\mathrm{d}} \tag{4.3}$$

$$F_{v,w} = p_{v,w} \cdot \sum_{u|(u,v)\in A} F_{u,v} \ , \forall (v,w) \in A \mid v \in V^{\mathrm{t}} \tag{4.4}$$

$$F_{v,w} \leq M \cdot Y_{r,l} \ , \forall r \in R \ , \forall l \in L \ , \forall (v,w) \in \Theta_{r,l} \tag{4.5}$$

$$F_{v,w} \geq 0 \ , \forall (v,w) \in A \tag{4.6}$$

$$Y_{r,l} \in \{0,1\} \ , \forall r \in R \ , \forall l \in L \tag{4.7}$$

Constraint 4.2 states that the outflow of each source node v is equal to o_v and Constraint 4.3 assures that the inflow into any decision node is equal to the outflow (balancing constraint). For any arc (v,w) going out of a transformation node, Constraint 4.4 assures that the inflow into that transformation node is transformed to outflow on arcs (v,w). Constraint 4.5 assures that only arcs are used that are enabled due to the availability of resources. We effectively make an arc uncapacitated if the required resources (if any) are available by setting the value of the big M equal to the maximum possible value of $F_{v,w}$. Arcs that are not in any $\Theta_{r,l}$ are uncapacitated as well.

4.3 Computational experiments

In Section 4.3.1 we compare the time it takes to solve problem instances using the flow model and the model that we presented in the previous chapter. A more efficient optimization method is useful to solve large problem instances, but it is also useful if a LORA is performed multiple times. This is for example the case if an iterative procedure is used to solve the joint problem of LORA and spare parts stocking, as we do in Chapter 6. In Section 4.3.2, we compare modelling the repair network exactly with aggregating all inputs per echelon level. We compare the time it takes to solve problem instances, the cost reductions that can be achieved by modelling the repair network exactly, and the repair strategies that result from modelling the repair network exactly and aggregating all data per echelon level.

For our tests, we generate instances of the LORA problem and solve these using the CPLEX callable library version 11 (with default settings), running under windows XP, service pack 2, on an Intel Centrino Duo, 2 GHz with 2 GB RAM. Although CPLEX 11 can use both cores of the dual core processor, it seldomly does for these problems.

Parameter (varied)	Values
# Components	500 & 1,000 & 2,000 & 5,000
# Echelon levels	2 & 3
# Indenture levels	2 & 3
Max. # resources per component	1 & 2

Parameter (not varied)	Value
# Resources	100

Parameter (not varied)	Range
Annual demand	[0.05; 5]
Variable costs	[50; 1,000]
Fixed costs	[500; 10,000]

Table 4.1: Input parameters

4.3.1 Comparison with the basic LORA model in terms of optimization time

We generate problem instances in which all data is aggregated per echelon level, corresponding to the assumption that we made in the previous chapter. In our model, this is equivalent to a network structure with one location at each echelon level. We use the problem instance generator as described in Section 3.3.1 and Appendix C.

We vary the four input parameters that most heavily influenced the optimization time in the previous chapter: the number of components, the number of echelon levels, the number of indenture levels, and the maximum number of resources per component. If the maximum number of resources per component is two, this means that in order to repair a component, at most two different resources are required. See Table 4.1 for the settings. For each combination of parameters, we generated 25 problem instances. In total, this makes $4 \cdot 2 \cdot 2 \cdot 2 \cdot 25 = 800$ test runs.

Table 4.2 shows the average time it takes to solve the LORA problem instances for each parameter that we varied. It is clear that our model increasingly outperforms the basic LORA model if the number of components, the number of indenture levels, the number of echelon levels, or the maximum number of resources per component increases.

4.3.2 Effect of modelling the exact repair network

In this section, we show the circumstances under which exactly modelling the repair network reduces the total costs and those under which inputs can be aggregated. We also compare the time it takes to solve these problems and the repair strategies that result.

# Components	Basic LORA model	Flow model
500	0.9	0.9
1,000	1.2	0.9
2,000	2.2	1.2
5,000	20.6	4.2

# Echelon levels	Basic LORA model	Flow model
2	2.5	1.2
3	10.0	2.4

# Indenture levels	Basic LORA model	Flow model
2	1.8	1.2
3	10.7	2.4

Max. # resources per component	Basic LORA model	Flow model
1	1.9	0.8
2	10.6	2.8

Table 4.2: Comparison of optimization times (seconds)

We generate instances that are realistic in practice, based on our experience at Thales Nederland. We make comparisons (1) for a base situation with a symmetrical repair network, and for an asymmetrical network (2) in terms of the number of operating sites that is attached to each intermediate depot and (3) in terms of the costs for moving and repairing components. Section 4.3.2.1 explains the problem instances we use and Section 4.3.2.2 discusses the results.

4.3.2.1 Problem instances

In all tests, the system structure consists of 25 components at the first indenture (LRUS), 125 at the second level and 625 at the third level. We use random generators to construct 10 instances of a product structure with corresponding failure rates and cost factors, see Appendix D for details.

For our tests, we use various repair networks, of which some are balanced in terms of the number of operating sites per intermediate depot and some are not. We call them balanced and unbalanced in the locations. The smallest balanced network consists of one central depot, two intermediate depots and four operating sites (two per intermediate depot). We vary the number of intermediate depots (2 or 10) and the number of operating sites per intermediate depot (2 or 10).

In the unbalanced networks, there are 2 operating sites per intermediate depot for half of the intermediate depots and 10 operating sites per intermediate depot for the other half of the intermediate depots. Below, we call the left half of the intermediate depots with the attached operating sites the 'left half'. The

other intermediate depot(s) with the attached operating sites are called the 'right half'. This holds for both balanced and unbalanced networks.

Besides being unbalanced in the locations, repair networks can be unbalanced in terms of the costs. In our tests, costs in the left half and at the central depot are always equal. Repair and move costs in the right half can differ from the costs in the left half and at the central depot. We test what happens if the repair costs in the right half are 0.5 or 2 times the repair costs in the left half (and at the central depot). We say that the relative repair costs are 0.5 or 2. In the same way, we test with relative move costs of 0.25 or 4. These values are chosen because for a European OEM, the costs of moving components to Asia can be a number of times as high as those costs in Europe. Repair costs can differ as well, but the relative difference is assumed to be smaller. Discard costs are assumed to be approximately the same at all locations, since a main part of these costs are due to the costs of purchasing a new component.

If we do not include spare parts costs in the LORA problem, which is what happens usually in practice, we see that many repairs are performed at a central location. The explanation is that in that case only 1 resource of each type needs to be acquired. If repairs are performed decentrally, many resources are needed. In the problem instances that we discuss below, 86% of the costs for resources would be made at the central depot if we do not include spare parts costs. Integrating spare parts optimization into the LORA is what we do in Chapters 6 and 7, but it does not fit in the scope of this chapter. However, we add spare parts costs in a basic way, by assuming lead times for all possible decisions (discard, repair and move) and relating spare parts costs to these lead times. This is elaborated on in more detail in Appendix D.

There are 10 or 25 resources (types of test equipment) and we distinguish two cases for the number of resources that each component requires. In the first case, 70% of the components needs no resources, 20% needs one resource, and 10% needs 2 resources: '0.7–0.2–0.1'. The other case is '0.25–0.5–0.25'.

Summarizing, we have 7 experimental factors, namely the number of intermediate depots, the number of operating sites in the left half, the number of operating sites in the right half, the relative move costs in the right half, the relative repair costs in the right half, the total number of resources, and the number of resources per component. This gives $2 \cdot 2 \cdot 2 \cdot 3 \cdot 3 \cdot 2 \cdot 2 = 288$ combinations. For each combination, we generate 10 problem instances as described in Appendix D. In this way we prevent that we draw conclusions based on one exceptional case.

4.3.2.2 Results

Table 4.3 shows for the 8 repair networks, the average time it takes to solve the exact and aggregated model. It also shows the average and maximal difference in optimal costs for both models. In each test we vary over all other parameters, as explained in the previous section. We focus on the other parameters below.

#Intermediate depots	# operating sites per interm. depot[a] Left half	Right half	Optimization time Exact	Aggr.[b]	Cost difference exact – aggr.[b] Mean	Max
2	2	2	0.3s.	0.1s.	0.1%	0.9%
		10	1.7s.	0.1s.	1.0%	5.1%
	10	2	1.6s.	0.1s.	1.0%	4.7%
		10	1.0s.	0.1s.	0.1%	0.9%
10	2	2	1.6s.	0.1s.	0.1%	1.2%
		10	8.1s.	0.1s.	2.7%	7.1%
	10	2	8.4s.	0.1s.	2.6%	7.4%
		10	10.2s.	0.1s.	0.2%	1.1%

Table 4.3: Comparison of various repair networks

It may seem strange that there is a difference between row 2 and 3 (2 intermediate depots and unbalanced network). However, these cases are not the same because, for example, if the relative repair or move costs are not 1, this affects more operating sites in row 2 than in row 3.

[a] Per intermediate depot.

[b] Data is aggregated per echelon level.

We see that the aggregated models can be solved much faster than the exact models. Still, it took only 104 seconds to solve the most time consuming problem instance using the exact model. This means that the exact model can be solved fast enough to be used in practice.

The maximum cost difference for networks that are balanced in the locations is 1.2% and only 8% of these problem instances lead to a cost difference of more than 0.5%. Since modelling the exact network requires more inputs, there is hardly any reason to do this if the repair network is balanced in the locations, even if the network is unbalanced in the costs. However, in our tests, the spare parts costs that are added to the variable costs, are always balanced in the network, even if the remainder of the variable costs for repair and move are unbalanced. In a global repair network, the spare parts costs may be unbalanced too, due to differences in lead times. If we vary the spare parts costs with the move and repair costs in a network that is balanced in the locations, we see cost differences between the exact and aggregated model of over 5%. However, our basic way of incorporating spare parts, does not allow us to analyse this in detail.

The maximum cost difference for networks that are unbalanced in the locations is 7.4%. For 29% of the problem instances, a cost difference of more than 2.5% is achieved and 6% of the problem instances leads to a cost difference of more than 5%. We focus in more detail on the networks that are unbalanced in the locations in Table 4.4, in which we vary the three parameters that most heavily influence the cost differences: The number of intermediate depots, the number of resources, and the number of resources per component. We see that high cost differences are mainly achieved in problem instances with ten intermediate

#Intermediate depots	# Resources		Cost difference exact – aggr.[a]	
	Total	Per comp.[b]	Mean	Max
2	10	0.7–0.2–0.1	1.1%	3.9%
		0.25–0.5–0.25	0.1%	1.2%
	25	0.7–0.2–0.1	0.5%	1.9%
		0.25–0.5–0.25	2.4%	5.1%
10	10	0.7–0.2–0.1	2.0%	6.1%
		0.25–0.5–0.25	2.1%	6.9%
	25	0.7–0.2–0.1	2.5%	6.8%
		0.25–0.5–0.25	4.0%	7.4%

Table 4.4: Comparison of three most important parameters

[a] Data is aggregated per echelon level.
[b] Per component.

Type of model	Costs (×1,000)		
	Variable	Resources	Total
Exact	43,925	12,404	56,329
Aggregated	42,241	15,308	57,549
Difference	-4.0%	19.0%	2.1%

Table 4.5: Example of different solutions

depots. If there are also many resources in total and many components need one or two resources, a large cost reduction by modelling the exact network is almost guaranteed. However, we cannot define a broad category of problem instances in which we can guarantee that there are no cost reductions possible.

If relatively small cost reductions can be achieved by modelling the exact network, the solutions of the exact model and the aggregated model can still differ substantially. If there are multiple, really different solutions that lead to approximately the same total costs, there can be other, more qualitative reasons to choose for another solution than the one with the lowest total costs. Industry might be interested in tools that can provide these different solutions. To give an example, we focus on the problem instances in which the repair network is unbalanced in the locations, there are ten intermediate depots, the number of resources per component is 0.25–0.5–0.25, and there are ten resources. Table 4.5 shows that for these problem instances, the costs of resources are 19% higher on average if the model is aggregated. However, this is compensated for by lower variable costs, so that the total costs differ only 2.1%.

Figure 4.7 shows that more in general, resource costs are higher if we aggregate the data in an unbalanced network than if we model the exact network, especially if there are ten intermediate depots. If, for those problem instances, we divide the amount of money that is spent on resources at the intermediate

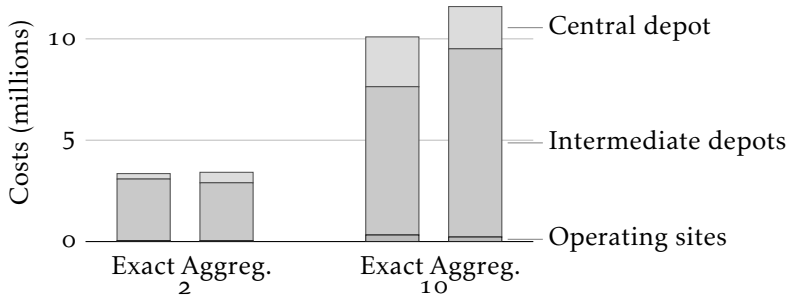


Figure 4.7: Resource costs in repair networks that are unbalanced in the locations

depots by the number of sytem locations in the network, there is a clear difference between the half of the network with two operating sites per intermediate depot (€153,000) and the half with ten operating sites (€116,000). Such a distinction cannot be made if the data is aggregated, because in that case, the model implicitly assumes there are six operating sites per intermediate depot on average.

Figure 4.8 gives some insights into how costs (divided by the number of operating sites) change if a network (that is balanced in the locations) grows due to more operating sites per intermediate depot (compare bars 1 and 2, and bars 3 and 4) or more intermediate depots (compare bars 1 and 3, and bars 2 and 4). Notice that both bars 2 and 3 give results for networks with 20 operating sites, although the results differ substantially. The reason is that repair at intermediate depot is much more expensive if there are ten intermediate depots, than if there are only two, due to the number of required resources. As a result, many more repairs are performed at the central depot instead of at the intermediate depot, which Figure 4.9 shows. Since many repairs are performed at the central depot, it is relatively inexpensive to repair the subcomponents of those components at the central depot too. Therefore, the discard costs in the network with ten intermediate depots are lower than in the network with two intermediate depots. In general, we see that the more operating sites there are, the more attractive it becomes to acquire resources and repair components instead of discarding them. As a result, total costs increase not as much as the number of operating sites increases.

We conclude that modelling the repair network exactly brings cost reductions of almost 2% on average for networks that are unbalanced in the locations. In some cases, the cost reductions are over 7%, which means that it is worthwhile to model the repair network exactly for unbalanced networks. For networks that are balanced in the locations, cost reductions are never higher than 1.2%, which means that it is doubtful whether the additional effort of acquiring all

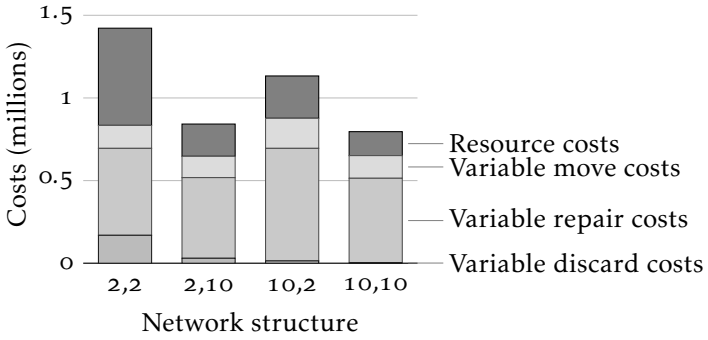


Figure 4.8: Total costs divided by number of operating sites in balanced repair networks
x, y (e.g., 2, 10) means: x intermediate depots and y operating sites per intermediate depot

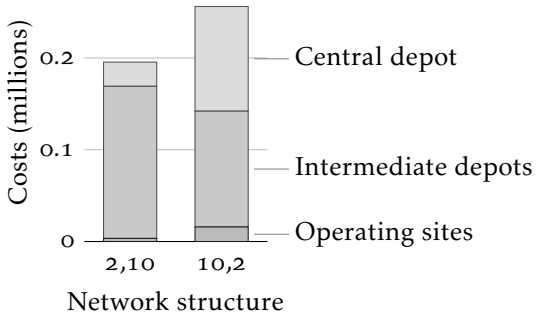


Figure 4.9: Resource costs divided by number of operating sites in balanced repair networks
x, y (e.g., 2, 10) means: x intermediate depots and y operating sites per intermediate depot

inputs is worth it. In this case, aggregating all data does not lead to much higher costs. However, we do note that if the costs for spare parts are not equal in the whole network, due to a difference in lead time between different parts of the global repair network, it might be necessary to explicitly model the repair network. However, more research is needed on the integration of spare parts optimization into the LORA before this question can be answered.

4.4 Conclusions

We have modelled the LORA problem as a minimum cost flow model with side constraints. This formulation allows us to model all kinds of extensions in an elegant manner. Such extensions include probabilities of unsuccessful repair, no-fault-found probabilities, and equipment with a finite capacity. Besides that, we have shown that the LORA problem with all data aggregated per echelon

level, can be solved much faster using our formulation than it could using the formulation that we developed in the previous chapter.

Our model allows us to explicitly model the repair network instead of aggregating all information per echelon level, as is often done in the literature. We have shown that with networks that are unbalanced in the locations, cost reductions of over 7% can be achieved by modelling the exact network. If networks are balanced in the locations, then the maximal costs reductions that can be achieved are only 1.2%, even if the network is unbalanced in the costs.

However, our research suggests that if lead times differ across the repair network, significant cost reductions may be achieved by modelling the exact network, even for networks that are balanced in the locations. To be able to analyse this, integration of spare parts optimization in the LORA is necessary.

We have shown that in some cases, only small cost differences exist between using the exact and the aggregated network, although the decisions that are taken differ a lot. If there are multiple ways to achieve almost the same total costs, there can be other, more qualitative reasons to choose for another solution than the one with the lowest total costs. Future research could lead to an approach that results in multiple alternative solutions that differ a lot in terms of decisions, but lead to almost the same costs.

Chapter 5

Extensions to the LORA flow model

In the previous chapter, we reformulated the LORA model that we developed in Chapter 3 as a minimum cost flow model with side constraints. In this chapter, we show the flexibility of this formulation by explicitly modelling several practically relevant model extensions. Thereby, we take the second and final step in answering research question 2b: "How can we model the extensions that may be needed in practice?" Two of the model extensions, the possibility that a repair is unsuccessful and the option to outsource repairs, are particularly relevant because we require them for our case study at Thales Nederland (see Section 2.1.1).

In Section 5.1, we give an overview of the model extensions and we discuss their practical relevance. Two of the extensions appear to be straightforward; for the remaining three extensions, we give the mathematical formulation in Section 5.2. Then, we perform an extensive numerical experiment in Section 5.3, in which we examine how both the repair strategies and the computation times change as a result of incorporating the latter three extensions. Finally, we draw conclusions in Section 5.4.

5.1 Motivation of model extensions

In this section, we give the motivation for the extensions that we model in this chapter; Sections 5.1.1 to 5.1.5 each discuss one extension. We pick these extension based on discussions with industry representatives (in particular those who participate in the IOP-IPCR project). However, more extensions can be modelled, see, for example, Section 8.3.1.2.

5.1.1 Unsuccessful repair

Until now, we assumed that all repairs are successful. In practice, however, this does not need to be true. First, a failure can occassionally be very serious. For example, a component may be seriously overheated, yielding unrecoverable damage. In that case, the component has to be discarded. Second, components can generally not be repaired over and over again. After they have been repaired a number of times, they can only be discarded. Third, it is possible that a failure is occassionally rather complex, so that specific expertise is required. In the latter case, it may happen that repair at the operating site is unsuccessful, but a second repair at the central depot is successful. At Thales Nederland, a rule of thumb is that about 5% of the repairs is unsuccessful and results in discarding the component.

In general, the costs of an unsuccessful and a successful repair may differ: on the one hand, if a component is visibly heavily damaged, costs of an unsuccessful repair may be low, since no testing is required (the subsequent discard action is just as expensive as any other discard action for this component). On the other hand, if the exact failure type is not easily determined, many tests may be required before it can be determined that a component cannot be repaired. In the latter case, unsuccessful repairs may be more expensive than successful repairs.

We have seen that if a repair is unsuccessful, a component can sometimes only be discarded, whereas in other cases a second repair attempt at a higher echelon level is useful. If a component needs to be discarded, there is the option to discard it at the location where the repair attempt was made, but if the discard costs are unequal at the various echelon levels, it may also be cost-effective to ship the component to a higher echelon level first. If a second repair attempt is still useful, the probability of unsuccessful repair at the second repair attempt may be the same as the probability at the first attempt, but it is probably higher. Since there are various options, we model four ways of incorporating the unsuccessful repair in Section 5.2.1.

5.1.2 No-fault-found

In practice, a certain fraction of the components that a repair shop receives, appears to have no problem, which is known as a no-fault-found. There are various reasons why a no-fault-found may occur: the diagnosis of the failure may be wrong, or a failure may occur due to the interaction of a component with the system; if the component is used in another system, it simply works. After extensive testing, the component goes back in stock as-good-as-new.

Since a no-fault-found is actually not a failure, it may seem reasonable to exclude it from the failure rate. However, the component may be tested extensively, possibly requiring a resource or shipment of the component to another location. As a result, the costs for the no-fault-founds cannot be neglected.

Furthermore, the costs of finding a no-fault-found and performing a normal repair may differ. Another difference is that no decisions need to be taken for subcomponents if a no-fault-found is detected, whereas they need to be taken if a component is repaired by replacing a subcomponent. We show how to model the extension to no-fault-found probabilities in Section 5.2.2.

5.1.3 Capacitated resources

In our basic LORA model, we assume that we need at most one resource of the same type at each location. In other words, resources have infinite capacity. At Thales Nederland, this is a realistic assumption since the resource utilization rate is generally very low (below 25%). However, at other companies it is possible that one resource is insufficient to accommodate all demand, and multiple resources need to be installed. This may happen especially if repairs are performed upstream in the repair network (e.g., at the central depot) and the installed base is relatively large, consisting of hundreds of systems. This is quite common for, e.g., medical equipment and airplanes. In that case, we should take the resource capacities explicitly into account, because it may impact the repair/discard decisions and the costs for resources. For example, if we need two resources at the central depot to handle all repair jobs, it could be more attractive to repair at the intermediate depots. If there are two intermediate depots, we may still require two resources in total, but we avoid shipping costs in this way. In Section 5.2.3, we show how we can use a step function for the fixed costs for resources to accommodate resource capacities.

We do not take into account waiting times that arise from resource utilization, as this does not fit within the framework of a deterministic optimization model. We assume that we can specify in advance which resource utilization rate is acceptable (e.g., 70%). A drawback of this approach is that if the utilization rate is just slightly above the acceptable rate (e.g., 71% instead of 70%), we require two resources, whereas in practice, one resource might still be sufficient.

5.1.4 Multiple failure modes per component

In the basic model, we implicitly assume that all failures of a certain component should be handled in the same way. That is, all those failures require the same resources and lead to the same variable repair costs. Therefore, we should take a single repair/discard decision for each component, irrespective of the exact failure mode. Obviously, this does not need to be true: multiple failure modes may occur in one type of component, and some failure modes may be more serious than others.

As an example, an optimal decision for a certain component (1) may have the following structure: component 1 has two failure modes: failure modes 1_a and 1_b. We diagnose the failure at an operating site, and our repair/discard decision depends on the failure mode that we find. In case of failure mode 1_a, we have a

simple, mechanical failure: repair is cheap and we do not need any resource. Therefore, we repair the component at the operating site. In case of failure mode 1_b, we have an electronic failure, and we need an expensive resource that we install at the central depot. As a result, we have multiple repair/discard decisions for the same component.

To include multiple failure modes in the LORA model, we do not need to change the model. Instead, we can replace 'component' by 'failure mode', and change costs, parent-child relations between failure modes, and relations between resources and failure modes. However, the size of the problem grows, and we need more detailed information on failures modes as input in the model. Particularly for the last reason, it makes sense to group failure modes in advance based on resource requirements and similar variable repair costs. Still, the increase in input data requirements prohibits the inclusion of failure modes in our case study at Thales Nederland; data at this level of detail is not available.

Finally, note that we have to be careful with modelling failure modes when combining LORA with spare parts stocking, as we do in Chapters 6 and 7: although we may take different repair/discard decisions for a certain component depending on the specific failure mode, we have a single set of spare parts that we allocate in the network, irrespective of the failure mode. We will return to this issue in Chapter 8.

5.1.5 Outsourcing of repair

For some repair jobs, it may be cost-effective to outsource repair instead of carrying out repairs in the own service network. This is especially true if expensive resources are required to perform the repairs. Acquiring such a resource may be only cost-effective if it can be used for other repairs as well. If the decision is to not buy the resource, it may still be better to outsource the repair than to discard the component.

Including the outsourcing option in the minimum cost flow model is straightforward. We can simply add an option 'outsource' at each location, or just at the central depot if all outsourced components should first be shipped to the central depot. The outsource option is specified similar to the discard option, so no decisions have to be taken for the component or its subcomponents if the outsource option is chosen. Furthermore, we have no resource costs, so only variable repair costs are to be taken into account (which may be higher than the variable repair costs for normal repairs). At Thales Nederland, we encountered the outsourcing decision for some components (outsource to the OEM).

Notice that the choice for outsourcing may also impact the spare parts requirements, because the repair lead times will be different. This can be an extra reason to include the outsourcing decision in the joint optimization models for LORA and spare part stocking that we discuss in Chapters 6 and 7.

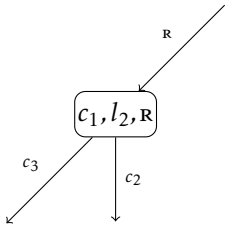


Figure 5.1: No unsuccessful repair
Node: A tuple (component, location, state), with state R is repair decision chosen. *Arcs*: A component for which no decision has been taken yet, or a letter representing a state.

5.2 Model formulation of extensions

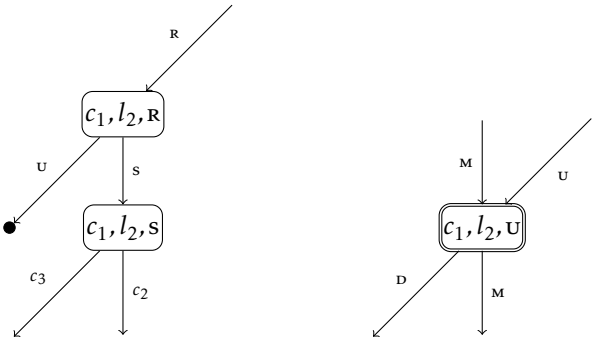
In this section we model three of the extensions that we discussed in Section 5.1: unsuccessful repair, no-fault-found, and capacitated resources. As explained, the other two extensions do not require a fundamental change in the model.

5.2.1 Unsuccessful repair

As discussed in Section 5.1.1, a new decision needs to be taken for the unsuccessfully repaired components. In some circumstances we can only discard the component, whereas in other circumstances a second repair attempt is possible (or even more attempts). Step-by-step, we include more options in our model:

- In Section 5.2.1.1, we discuss the most restricted case: after unsuccessful repair at a certain location, the component is discarded at that location.
- In Section 5.2.1.2, we allow the component to be shipped to a higher echelon level to be discarded there.
- In Section 5.2.1.3, we allow for a second repair attempt at a higher echelon level. We make the assumption that the probability of unsuccessful repair does not depend on whether or not a previous repair attempt has been made.
- In Section 5.2.1.4, we allow the probability of unsuccessful repair to be higher at a second repair attempt if we know that a first attempt failed.

As a result of adding more options to the model, the model requires more nodes and arcs. We will clarify this throughout this section using a small example: we consider a component (c_1), having two subcomponents (c_2 and c_3). We assume that component c_1 is repaired at location l_2. In Figure 5.1, we show the transformation node in our basic flow model from the previous chapter; in the sections below, we show what modifications we make to this part of the flow model to incorporate a probability of unsuccessful repair.



(a) Discard at location of repair attempt

(b) Discard at higher echelon

Figure 5.2: Unsuccessful repair

Nodes: A tuple (component, location, state), with state R is repair decision chosen, S is successfuly repaired, and U is unsuccessfully repaired. *Arcs*: A component for which no decision has been taken yet, or a letter representing a state. In addition to states R, S, and U, state M is move decision chosen and D is discard decision chosen.

5.2.1.1 Discard at location of repair attempt

The simplest case of incorporating a probability of unsuccessful repair, is by assuming that after an unsuccessful repair, the component will be discarded at the location where the repair attempt was made. To model this, we add an additional transformation node, see Figure 5.2a, such that the incoming flow of components that are to be repaired, is split into two flows:

1. A flow of successfully repaired components, which goes to the already existing transformation node (which is renamed to c_1, l_2, S, with S denoting successfully repaired).
2. A flow of unsuccessfully repaired components that will be discarded. This flow ends in a sink node.

Note that the two transformation nodes could be combined to one transformation node, but for clarity, we treat them as being separate.

Since the costs of unsuccessful repair and successful repair may be different, the variable repair costs can be attached to the arc representing unsuccessfully repaired (U) and successfully repaired (S), respectively.

5.2.1.2 Discard at a higher echelon level

If the costs of discard are different at the various echelon levels, it may be cost-effective to ship a component to a higher echelon level, and discard it there. In that case, we add a transformation node in the same way as we did in the previous section (Figure 5.2a). However, the flow representing the

unsuccessfully repaired components (U) is sent to a decision node (instead of a sink node), see Figure 5.2b. At this node, there are the options to discard and to move the component. The arc representing the move option (the outgoing arc M) terminates at a decision node representing the next upstream location; flow on the incoming arc M represents components that were unsuccessfully repaired at a lower echelon level and are moved to location l_2. Notice that at the central depot, only the discard option is available (a decision node is not necessary in that case).

5.2.1.3 Second repair attempt at a higher echelon level: independent probabilities

If the probability of unsuccessful repair is lower at a higher echelon level, or if engineers at different echelon levels have different skills or different equipment, it may be cost-effective to make a second repair attempt at a higher echelon level after a first attempt failed.

In the model in Section 5.2.1.2 (Figure 5.2b), the unsuccesfully repaired components are in a distinct part of the network; they do not mix with the components for which no repair attempts have been made yet. A simple way to incorporate the possibility of a second repair attempt is by merging the flow of unsuccessfully repaired components that are moved to a higher echelon level (outgoing arc M) with the flow of components that were directly moved to that higher echelon level, without a repair attempt first. This means that these two types of components cannot be distinguished anymore at the higher echelon level. Consequently, we assume that the probability of unsuccessful repair is equal for both flows.

As an example, let us assume that the probability of unsuccessful repair of component c_1 is 0.2 at location l_2, and 0.1 at location l_1. $\Phi_{l_1} = \{l_2\}$, which means that components that are moved upstream at location l_2, are shipped to location l_1. If a repair is attempted at both locations l_2 and l_1, then 2% ($0.2 \cdot 0.1 = 0.02$) of the components is unsuccessfully repaired after the two attempts.

5.2.1.4 Second repair attempt at a higher echelon level: generalized model

In Section 5.2.1.3, we assumed that the probability of unsuccessful repair is independent on the number of previous repair attempts. In some cases, this may be correct, but as explained in Section 5.1.1, the probability of unsuccessful repair is generally higher for a second repair attempt than for a first repair attempt.

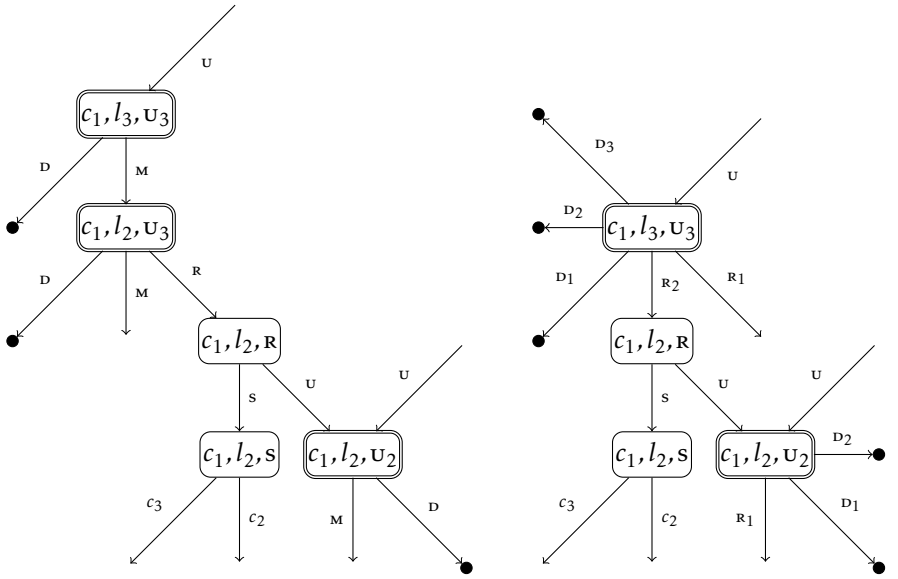
It is possible to assign different probabilities for the second (and any further) repair attempt at any location. To explain how we can model conditional probabilities, we use an example of three locations at three different echelon levels. Location l_3 (at echelon level 1) is the operating site. If components are

Location of			Fraction unsuccessfully repaired			
First attempt	Second attempt	Third attempt	First attempt	Second attempt	Third attempt	In total
Loc. l_3	—	—	0.30	—	—	0.30
Loc. l_3	Loc. l_2	—	0.30	0.67	—	0.20
Loc. l_3	Loc. l_2	Loc. l_1	0.30	0.67	0.50	0.10
Loc. l_3	Loc. l_1	—	0.30	0.33	—	0.10
Loc. l_2	—	—	0.20	—	—	0.20
Loc. l_2	Loc. l_1	—	0.20	0.50	—	0.10
Loc. l_1	—	—	0.10	—	—	0.10

Table 5.1: Fractions of components that are unsuccessfully repaired

moved upstream in the network, they are shipped to location l_2, which can in turn ship components to location l_1 ($\Phi_{l_1} = \{l_2\}$ and $\Phi_{l_2} = \{l_3\}$). The probability of unsuccessful repair is 0.3 at location l_3, 0.2 at location l_2 and 0.1 at location l_1. We now know the values in the fourth column (First attempt) in Table 5.1. The values in the fifth and sixth columns can be specified, which automatically leads to a value in the last column. In this way, we cover the general case of probabilities that may depend on the number and locations of all previous repair attempts. It also allows us to specify the probabilities such that we reproduce the model that we showed in the previous section.

However, although conditional probabilities fit perfectly within the framework of the minimum cost flow model with side constraints from a modelling point of view, we face two drawbacks. First, it is hard to specify all the required probabilities from a practical perspective. Second, the flows corresponding to each unsuccessful repair at each location should be separated, which yields a less transparent and larger model. Therefore, we propose a model in which only a few probabilities need to be specified. To this end, we assume that failures can be ranked according to their complexity. This means that if the probability of unsuccessful repair at one location is lower than at another location, all failures that will successfully be repaired at the latter location, will also successfully be repaired at the former location. In our example, this means that the fraction of the repairs that are successful at location l_3 (0.7) concern the least complicated failures, whereas the fraction that can be repaired at location l_2, but not at location l_3 (0.8-0.7=0.1) is more complicated, et cetera. Under these assumptions, if a first repair attempt is made at location l_3 and a second repair attempt is made at location l_2, then a fraction of $\frac{0.2}{0.3} = 0.67$ of the components that are unsuccessfully repaired at location l_3, are also unsuccessfully repaired at location l_2. Note that under these assumptions, a second repair attempt is useful only if the probability of unsuccessful repair is lower at the higher echelon level. Furthermore, it is only relevant to know the last location where a repair attempt has been performed; it does not matter whether any other repair attempts have been made. In Table 5.1, we show the fractions of components that are unsuccessfully repaired at the first attempt,



(a) Standard version (b) Merged version

Figure 5.3: Unsuccessful repair: discard or repair later

Nodes: A tuple (component, location, state), with state R is repair decision chosen, S is successfuly repaired, and U is unsuccessfully repaired. A subscript indicates the location of the last unsuccessful repair. *Arcs*: A component for which no decision has been taken yet, or a letter representing a state. In addition to states R, S, and U, state M is move decision chosen and D is discard decision chosen. A subscript indicates where the repair or discard is to be performed.

the second attempt (if applicable), the third attempt (if applicable), and overall, for various combinations of repair locations in our example. The values for the second and third attempt, and the overall value, result from the three probabilities that are specified. In the remainder of this chapter, we use this restricted version of conditional probabilities, instead of the general version of conditional probabilities in which columns five and sixe in Table 5.1 need to be specified.

We give the description of our new model for the example discussed above; the new model is shown in Figure 5.3a. We start with a flow of components that were repaired unsuccessfully at location l_3 (30% of the failures, represented as U_3 (unsuccessful at location l_3) in the decision node). Such components can either be discarded or moved to the next higher echelon level. At location l_2 (the decision node c_1, l_2, U_3), there are three possible decisions:

1. The component can be discarded.
2. The component can be moved to location l_1.
3. A (second) repair attempt can be made.

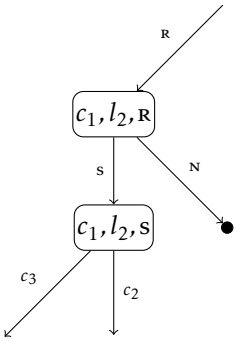


Figure 5.4: No-fault-found

Nodes: A tuple (component, location, state), with state R is repair decision chosen and S is successfuly repaired. *Arcs*: A component for which no decision has been taken yet, or a letter representing a state. In addition to states R and S, N is no-fault-found.

A repair attempt is either successful (33% of the components) or unsuccessful (67%). In the latter case, the component cannot be distinguished from a component which was unsuccessfully repaired at location l_2 only (20%). Therefore, we can merge these two streams of components in the decision node c_1, l_2, U_2.[1]

Finally, note that we can simplify the representation by merging decision nodes such that after an unsuccessful repair, we immediately decide what the next repair/discard decision will be (discard at the current or a higher echelon level or repair at a higher echelon level), see Figure 5.3b.

5.2.2 No-fault-found

Modelling the no-fault-found probability is similar to modelling the simplest case of a probability of unsuccessful repair (see Section 5.2.1.1). In Figure 5.4, we show an example in which we assume that there is a probability of a no-fault-found (at location l_2). We add a transformation node, such that the incoming flow of components that are to be repaired, is split into two flows:

1. A flow of successfully repaired components, which goes to the already existing transformation node (which is renamed to c_1, l_2, s, with s denoting successfully repaired).
2. A flow of components in which no failures is found, which terminates in a sink node, since these components are put back in stock, and no additional decision needs to be taken.

Note that the two transformation nodes could be combined to one transformation node, but for clarity, we treat them as being separate.

[1] Notice: in the general case of probabilities that may depend on the number and locations of all previous repair attempts, merging these two streams is not possible, resulting in a larger model.

Since the costs of a no-fault-found and a normal repair may be different, the variable repair costs can be attached to the arc representing no-fault-found (N) and successfully repaired (S), respectively.

5.2.3 Capacitated resources

To model capacitated resources, we change the side constraint that links arcs to resources in the minimum cost flow model. In Section 4.2.2, we defined $\Theta_{r,l}$ such that if a flow on an arc (v,w) is only allowed if resource r is available at location l, then $(v,w) \in \Theta_{r,l}$. We change Constraint 4.5, which allows flows on arcs (v,w) only if all required resources r are available at location l ($F_{vw} \leq M \cdot Y_{r,l}\ , \forall r \in R\ , \forall l \in L\ , \forall (v,w) \in \Theta_{r,l}$) such that the total flow over all arcs $(v,w) \in \Theta_{r,l}$ should be lower than the capacity per resource r times the number of resources r at location l. For the detailed technical description, see the technical note at the end of this chapter.

5.3 Computational experiments

In this section, we perform an extensive numerical experiment to answer the following two questions for the three extensions that we modelled in the previous section:

1. How do repair strategies and total costs change as a result of incorporating the extensions that we modelled?
2. How do optimization times change as a result of incorporating these extensions?

The main goal of our tests is to examine the importance of incorporating the extensions in order to get accurate, practically useful results. For example, we will see that incorporating a certain probability of unsuccessful repair leads to almost doubling the total costs on average. This means that neglecting this probability leads to a significant underestimation of the true costs in practice. The goal of the second question is to determine whether incorporating these extensions leads to a model that can still be solved in a reasonable amount of time. This is especially relevant if the LORA model is solved several times in an iterative approach to solve the joint problem of LORA and spart parts stocking, as we do in Chapter 6.

For our numerical experiment, we generate problem instances using a slightly modified version of the generator that was presented in the previous chapter (Section 4.3.2.1) and is described in detail in Appendix D. We make two changes:

- Due to incorporating the model extensions, the problem sizes increase and, more importantly, there are more possible decisions. As a result, CPLEX requires much more memory when solving the problem instances,

especially for the problem instances with capacitated resources. This means that CPLEX is not able to solve the largest problem instances when incorporating the capacitated resources. Therefore, we reduce the problem size by decreasing the size of the repair network: instead of performing tests with 2 and 10 intermediate depots, and 2 and 10 operating sites per intermediate depot (100 operating sites at maximum), we perform tests with 2 and 5 intermediate depots, and 2 and 5 operating sites per intermediate depot (25 operating sites at maximum). This means that the size of the largest repair networks is still larger than the repair network in our case study (see Section 2.1.1).

- In the previous chapter, we varied seven parameters, leading to 288 (combinations of) parameter settings. In the current chapter, we perform tests for symmetrical repair networks only, which means that we vary four parameters, leading to 16 parameter settings (not considering the extensions). Assuming symmetrical repair networks means that we can merge all results per echelon level, which makes it easier to analyze the effect of incorporating the extensions. Besides, we concluded in Chapter 4 that assuming symmetrical decisions in asymmetrical networks generally yields a small cost increase only.

We generate ten problem instances for each of the parameter settings, resulting in 160 basic problem instances.

We perform tests for the three extensions that we discussed in the previous section, one by one. Although we showed four variants to incorporate the probability of unsuccessful repair, we perform tests for only two of them: the simplest and the most complicated one, so 'discard at location of repair attempt', see Section 5.2.1.1, and 'discard or repair at any location', see Section 5.2.1.4. Furthermore, we assume that the costs of an unsuccessful repair, a successful repair, and finding a no-fault-found are equal. Finally, we assume that the failure rates of the LRUs are not changed if an extension is incorporated. This means that no-fault-founds are part of this failure rate, so the number of 'actual failures' decreases if we incorporate a no-fault-found probability. In Sections 5.3.1 to 5.3.3, we give for each of the three extensions, respectively, the additional parameters that we use and the test results.

5.3.1 Probability of unsuccessful repair

For the probability of unsuccessful repair, we distinguish nine settings. We give the values that we use for the probabilities in each setting in Table 5.2; the indenture level of the component and the echelon level of the location together determine the probability in the problem instances. In settings 1 to 3, the probabilites are equal for all echelon levels and for all indenture levels. Since, under the assumptions given in Section 5.2.1.4, equal probabilities of unsuccessful repair means that no additional repairs can be performed at the higher echelon levels, performing tests using these settings is less useful

	Echelon 1			Echelon 2			Echelon 3		
Setting	LRU	SRU	part	LRU	SRU	part	LRU	SRU	part
1	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06
2	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12
3	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18
4	0.09	0.09	0.09	0.06	0.06	0.06	0.03	0.03	0.03
5	0.18	0.18	0.18	0.12	0.12	0.12	0.06	0.06	0.06
6	0.27	0.27	0.27	0.18	0.18	0.18	0.09	0.09	0.09
7	0.06	0.075	0.09	0.045	0.06	0.075	0.03	0.045	0.06
8	0.12	0.15	0.18	0.09	0.12	0.15	0.06	0.09	0.12
9	0.18	0.225	0.27	0.135	0.18	0.225	0.09	0.135	0.18

Table 5.2: Parameters: probability of unsuccessful repair

for the tests with 'second repair attempt at higher echelon level', so we will not perform such tests. In settings 4 to 6, we vary the probabilities over the echelon levels, but keep them equal for the various indenture levels. The probabilities are lower at the higher echelon levels, since in general, more specialized engineers and equipment will be available at the higher echelon levels. In settings 7 to 9, we vary the probabilities over both the echelon levels and the indenture levels. The lower the indenture level of the component (e.g., LRU), the lower the probability of unsuccessful repair. The reason is that these components are more expensive, which makes it worthwhile to invest time and money if the failure is hard to repair. Besides, it is more probable that a part of the LRU can be replaced, thus repairing the LRU, whereas this is often not possible for the higher indenture level components (e.g., parts).

Since there are 160 problem instances for each parameter setting, there are $9 \cdot 160 = 1{,}440$ problem instances in total for the tests 'discard at location of repair attempt', and $6 \cdot 160 = 960$ problem instances for the tests 'second repair attempt at at higher echelon level'.

5.3.1.1 Discard at location of repair attempt

Incorporating a probability of unsuccessful repair leads to almost a doubling of the costs on average: 24.8 million versus 12.8 million (95% increase). The maximum cost increase is 256%. This means that ignoring a probability of unsuccessful repair may lead to a huge underestimation of the true costs. The reason is that many components cannot be repaired successfully and need to be discarded. However, discarding components is very expensive, since that means that replacements need to be purchased.

If the probability of unsuccessful repair at all echelon levels and all indenture levels is 0.18 (setting 3), costs increase with 60% at minimum, 146% on average and 240% at maximum. Comparing the various settings shows that these values almost double if the probability of unsuccessful repair doubles.

	Variable costs			Resource costs			Total
	Discard	Repair	Move	Ech. 1	Ech. 2	Ech. 3	costs
No extension	553	6,732	1,670	64	2,513	1,221	12,753
Settings 1-9	12,004	7,176	2,513	22	1,767	1,335	24,817
Settings 1-3	13,888	6,381	1,635	64	2,502	1,061	25,532
Settings 4-9	11,062	7,574	2,952	0	1,399	1,472	24,456

Table 5.3: Costs (×1,000), resulting from using a probability of unsuccesful repair (immediate discard)

The computation time increases from 0.5 seconds to 1.3 seconds on average if the probability of successful repair is incorporated (less than 30 seconds at maximum), which means that from the model point of view, this extension can easily be included. Of course, the question is whether in practice, all the required data is available.

In Table 5.3, we see that not only the costs change, the repair strategy changes as well (first two rows) the variable costs of discard increase significantly, which is a direct result of incorporating the probability of unsuccessful repair. To understand the other results, we have to look in more detail to the results (last two rows). There is a clear difference between settings 1 to 3 on the one hand, and settings 4 to 9 on the other hand. In the problem instances that are generated using settings 1 to 3, the probability of unsuccessful repair is equal at all echelon levels for all components, whereas it decreases in the other problem instances.

In the former case, less repairs are performed and less components are moved. Since a probability of unsuccessful repair for components means that in some cases, no subcomponents are replaced, there are simply less components in total that require a repair or move action. This decrease in demand also means that in some cases, it is not worthwhile to locate a resource, as can be seen in Table 5.3: if a component is not too expensive, its repair requires a resource, and there is a significant probability that repair is not successful, it can be sensible to discard the component immediately. This also leads to a decrease in the variable repair and move costs (there is no reason to move a component to a higher echelon level if it will be discarded). Since the decrease in resource costs is relatively small (and at echelon level 3 only) and the decrease in repair and move costs can partly be explained by a decrease in the number of failed SRUs and parts for which any decision needs to be taken, we conclude that the repair strategy is affected to a very small extend only if a probability of unsuccessful repair is incorporated that is equal at all echelon levels.

In the latter case, the problem instances generated using settings 4 to 9, results are quite different. The repair strategy changes significantly due to the inclusion of a probability of unsuccessful repair; we see a large increase in the variable move costs and in the resource costs at central depot, but a decrease of the resource costs at the lower echelon levels: it is worthwhile to ship components

	Variable costs			Resource costs			Total
	Discard	Repair	Move	Ech. 1	Ech. 2	Ech. 3	costs
No extension	553	6,732	1,670	64	2,513	1,221	12,753
Settings 4-9	7,549	8,564	2,322	14	1,641	1,689	21,780

Table 5.4: Costs (×1,000), resulting from using a probability of unsuccesful repair (repair/discard at higher echelon)

to the central depot, since that leads to a low probability of discarding the component. We also see that the repair costs increase, which means that a repair attempt is made on more components (instead of a direct discard). The reasoning is as follows. If components are sent to the central depot in order to be repaired there, then the subcomponents are replaced at the central depot. If no resource is required in order to repair this subcomponent, it can be repaired at the central depot. More importantly, if a resource is required, chances are that this resource is available at the central depot, whereas in the solution without a probability of unsuccessful repair, the resource may be available only at a lower echelon level, or the component may be replaced at a lower echelon level, and moving the component to repair it at a higher echelon level may be more expensive than discarding it.

5.3.1.2 Second repair attempt at at higher echelon level

The inclusion of a second repair attempt at a higher echelon level yields a cost increase of 71% on average compared with ignoring unsuccessful repair: 21.8 million versus 12.8 million. The maximum cost increase is 179%. However, we have seen in Section 5.3.1.1 that if we allow for discard at the location of the repair attempt only, costs increase with 95% on average, and 256% at maximum. This means that if the probabilities differ at the various echelon levels, it is worthwhile to allow for a second repair attempt. However, the additional possibilities come at a price: the optimization time increases from 0.5 seconds for the case of no probability of unsuccessful repair, to 1.3 seconds for the case that only discard at the location of the repair attempt is allowed, to 5.7 seconds if all options are open. Still, the maximum optimization time is under 56 seconds (compared with 29 seconds if only immediate discard is allowed). Obviously, the computation time is not a problem for usage of this model extension in practice.

Let us look at the results in more detail, see Table 5.4. Compared with the last row in Table 5.3 (discard only), the discard and move costs increase less, whereas the variable repair costs increase more. The resource costs decrease far less and even increase at the central depot. The reason is that instead of shipping components to the highest echelon level, and performing one repair attempt there, now a first repair attempt is made on many components at the lower echelon levels, and a second (and third) repair attempt is made at higher echelon levels.

Setting	LRU	SRU
1	0.06	0.06
2	0.12	0.12
3	0.18	0.18
4	0.09	0.03
5	0.18	0.06
6	0.27	0.09

Table 5.5: Parameters: no-fault-found probability

5.3.2 No-fault-found probability

To examine the impact of the no-fault-found probability on the results, we perform tests with six different settings, see Table 5.5. Since in our tests, finding a no-fault-found is equally expensive as performing a repair, and the component goes back in stock in as-good-as-new state in both cases, the only difference is that a no-fault-found means that no decisions need to be taken for subcomponents. Therefore, setting a no-fault-found probability is useful only for components that have subcomponents, i.e., not for parts. Since the no-fault-found probability is mainly related to the components themselves, we assume that the values are equal at all echelon levels. Furthermore, we assume that they are equal at both indenture levels in settings 1 to 3, and they differ for both indenture levels in settings 4 to 6. In the latter case, the probability for the LRUs is higher than that for the SRUs, since LRUs are technically more complex. Besides, LRUs are always replaced in the field, whereas SRUs are typically replaced in a repair shop. This means that the probability of an incorrect replacement (another component failed) is generally higher for LRUs than it is for SRUs.

In general, incorporating a no-fault-found probability leads to lower costs (-4% on average) and an increase in the computation time (0.5 seconds versus 1.6 seconds). The maximum cost reduction that can be achieved is over 12%, which makes it worthwhile to incorporate the no-fault-found probability in a LORA model. Obviously, the higher the no-fault-found probability, the higher the cost reduction compared with not incorporating the no-fault-found probability. Both the average and maximum cost reduction almost double if the no-fault-found probability doubles.

We see in more detail how costs change in Table 5.6. The reasoning to explain the results is as follows. Some of the LRUs that are repaired, turn out to be a no-fault-found, which means that its SRUs need not be repaired or discarded. The same holds for SRUs and parts, respectively, which means that the total demand for repair and discard actions decreases in the LORA problem. For some resources, the demand gets so low that it is cost-effective to discard all components that require that resource in order to be repaired, instead of locating the resource and repairing the components. As a result, less resources

	Variable costs			Resource costs			Total
	Discard	Repair	Move	Ech. 1	Ech. 2	Ech. 3	costs
No extension	553	6,732	1,670	64	2,513	1,221	12,753
Settings 1-6	566	6,402	1,644	67	2,455	1,163	12,297

Table 5.6: Costs (×1,000), resulting from using a no-fault-found probability

	Costs (×1,000)	Optimization time (seconds)
No extension	12,753	0.5
$RR = 2$	13,383	8.1
$RR = 4$	15,023	34.6

Table 5.7: Results for various settings for the capacitated resources

are located in the network, the discard costs increase, and the (variable) repair and move costs decrease. This leads to a reduction of the total costs. However, the change in the repair strategy is small.

5.3.3 Capacitated resources

To test the extension to capacitated resources, we need to set two sets of parameters:

- the demand in hours for a resource that results from performing one repair action for a component that requires that resource, and
- the annual number of hours that each resource can be used (capacity of the resource).

For simplicity, we set all demands to 1. Then, notice that for each resource, the demand for that resource at any location, cannot be higher than the demand for that resource at the central depot if all components are sent to the central depot in order to be repaired. We design our experiments such that it holds for all resources that if all components are repaired at the central depot, we require RR resources at the central depot. We perform tests with $RR = 2$ and $RR = 4$. Notice that $RR = 1$ effectively represents uncapacitated resources.

The cost difference due to assuming capacitated resource compared with assuming uncapacitated resources is quite large, as can be seen in Table 5.7. If $RR = 4$, it is 18% on average and 59% at maximum. This suggests that companies need to consider whether or not the assumption of uncapacitated resources is realistic in their business settings; if uncapacitated resources are assumed where this is not realistic, this leads to a significant underestimation of the costs.

	Variable costs			Resource costs			Total
	Discard	Repair	Move	Ech. 1	Ech. 2	Ech. 3	costs
No extension	553	6,732	1,670	64	2,513	1,221	12,753
$RR \in \{2,4\}$	1,477	5,935	1,422	490	4,129	748	14,203

Table 5.8: Costs (×1,000), resulting from using capacitated resources

The optimization time increases significantly as well; it is almost half an hour at maximum (compared with 2.3 seconds if no extensions are included). In itself, such an optimization time is not problematic, but if the problem size increases, or if the LORA is a building block in an iterative approach to solve the integrated problem of LORA and spare parts stocking analysis (see Chapter 6), such a time may be problematic.

In Table 5.8, we give more detailed results. If resources are capacitated, the repair option gets relatively more expensive compared with the discard option. Therefore, there is an increase in the discard costs and a decrease in the repair costs (notice that the resource costs are not part of the variable repair costs). If resources are capacitated, two or more of the same resources may be required if all repairs are performed at the central depot. In that case, it may be cost-effective to perform the repairs at a lower echelon level, thus reducing the move costs, while not increasing the total number of required resources. Therefore, the move costs decrease and the resources are located at lower echelon levels. We also see an increase in the total costs for resources, which is due to the fact that sometimes a couple of resources of the same type are required if they are capacitated. Upto RR resources of the same type are located at the central depot.

As shown in Table 5.9, the cost difference between assuming capacitated resources and assuming uncapacitated resources decreases if the number of intermediate depots increases. If the number of intermediate depots is 2 and $RR = 4$, then locating a resource at the intermediate depots may mean that two resources per intermediate depot are required. However, if the number of intermediate depots is 5, then locating one resource at each of the intermediate depots is always sufficient, due to the way we construct our problem instances (see the beginning of this section for our definition of RR). This means that with five intermediate depots, additional resources are required only if repairs are performed at the central depot.

If the number of operating sites per intermediate depot decreases or the number of resources increases, the cost difference increases as well. This makes sense, since in that case, the resource costs make up a larger percentage of the total costs, and in Table 5.8, we have seen that the cost difference results mainly from the cost increase in the resource costs.

Parameter	Setting	Cost difference
# intermediate depots	2	16.9%
	5	10.3%
# operating sites per intermediate depot	2	16.5%
	5	10.7%
# resources	10	11.6%
	25	15.6%

Table 5.9: Cost difference due to using capacitated resources for various parameter settings

The cost difference is the average over all problem instances of: the total costs incorporating capacitated resources minus the total costs neglecting the resource capacities, divided by the total costs neglecting the resource capacities.

5.4 Conclusions

In this chapter, we presented a number of extensions to the LORA model that we presented in Chapter 4:

- a probability of unsuccessful repair,
- a no-fault-found probability,
- capacitated resources,
- multiple failure modes per component, and
- outsourcing of repair.

We showed how to model the first three extensions: the first two extensions require a change in the nodes and arcs; the capacitated resources require a change in the constraint that links components to resources. The last two extensions do not require a fundamental change in the model.

We tested the first three extensions by generating problem instances in a similar way as we did in the previous chapter. We have seen that incorporating a probability of unsuccessful repair leads to a cost increase (compared with neglecting this probability) of more than 200% at maximum. The cost increase is larger if an unsuccessfully repaired component can only be discarded at the location where the repair was performed (95% on average), than if we allow a second (and third) repair attempt at a higher echelon level (71% on average). If the probabilities are equal at all echelon levels, the change in repair strategy (compared with not incorporating unsuccessful repair) is small: slightly less resources are located at the central depot. If the probability of unsuccessful repair decreases with an increasing echelon level, the repair strategy changes significantly. The key difference is that the resource costs at central depot increase, wherease they decrease at the lower echelon levels. This holds both if components need to be discarded after an unsuccessful repair attempt, and if a

second repair attempt is allowed.

Incorporating a no-fault-found probability leads to a reduction of the costs of 4% on average and over 12% at maximum. The change in the repair strategy is small.

Incorporating capacitated resources leads to a cost increase of 11% on average and 59% at maximum. The optimization time increases significantly as well: it is almost half an hour at maximum compared with 2.3 seconds if no extensions are included. Since repair is more expensive if resources are capacitated, more components are discarded. Furthermore, resources are located at lower echelon levels.

For all extensions it holds that the costs change significantly. Even for the no-fault-found probability it is over 12% at maximum. For the extension to capacitated resources and the inclusion of a probability of unsuccessful repair that decreases with an increasing echelon level, it also holds that the repair strategy changes. Therefore, these two extensions should be incorporated in practice. Incorporating a probability of unsuccessful repair that is equal at all echelon levels and incorporating a no-fault-found probability does not lead to a significant change in the repair strategy. Therefore, it may be possible to neglect these probabilities during optimization, and adapt the total costs with a certain percentage afterwards.

Technical note

In this technical note, we define formally the extension to capacitated resources that we presented in Section 5.2.3. To this end, we define two parameters:

- $h_{c,r,d}$ is the demand (in hours) for resource r per component c for which decision d is taken, and
- u_r is the annual number of hours that resource r can be used (capacity of resource r assuming a certain maximum utilization rate, e.g., 70%).

We change $Y_{r,l}$ from being binary to being integer, and we change Constraint 4.5 to the constraint:

$$\sum_{(v,w)\in\Theta_{r,l}} g_{v,w,r} \cdot F_{v,w} \leq u_r \cdot Y_{r,l} \;, \forall r \in R \;, \forall l \in L \tag{5.1}$$

With $g_{v,w,r}$ being the demand (in hours) for resource r if a flow of one goes through arc (v,w): $g_{v,w,r} := h_{c,r,d}$ if arc (v,w) represents decision d for component c at a certain location.

Chapter 6

Iterative method for the joint problem of LORA and spare parts stocking[1]

In the previous chapters, we focused on the LORA problem, since we concluded in Section 2.5 that that should be the first step in our research. Solving the joint problem of LORA and spare parts stocking can be done by executing a LORA first, and then solving a spare parts stocking problem, the so-called *sequential approach*. However, we expressed our concerns about the quality of a solution that results from this approach in Section 1.5, while Alfredsson (1997) and Brick and Uchoa (2009) express similar doubts. Now that we have a general LORA model, we are ready to answer research question 3: "What is a suitable method to solve the joint problem of LORA and spare parts stocking?"

Question 3 is split in two subquestions. In Chapter 7, we answer research question 3b; in this chapter we answer research question 3a, which is: "How can we iteratively use a LORA model and a spare parts stocking model to solve the joint problem of LORA and spare parts stocking?" This means that we restrict ourselves to an algorithm that is based on known building blocks. As building blocks, we use our model from Chapter 4 and the VARI-METRIC model (see, e.g., Muckstadt, 2005; Sherbrooke, 2004). However, other LORA models and METRIC type spare parts stocking models could be used as well. In order to keep the model transparent, we do not include the extensions that we modelled in Chapter 5. In the next chapter, we develop an algorithm that finds a solution that is optimal in the sense that the achieved availability cannot be achieved against lower costs (an efficient point) and except for the approximation errors in VARI-METRIC. That algorithm can be used as a benchmark for the *iterative algorithm* that we develop in the current chapter.

[1] Based on Basten et al. (2009c)

This chapter is structured as follows. Section 6.1 outlines the integrated model for LORA and spare parts stocking that we use. In Section 6.2, we explain the general idea behind our iterative method. Next, we specify a basic algorithm and several variants in Section 6.3. In Section 6.4, we examine the added value of our iterative approach compared with the sequential approach in an extensive numerical experiment. We show that our algorithm significantly improves the sequential approach with a maximum cost reduction of more than 35% (over 3% on average). Also, we identify the variant of our iterative algorithm that performs best in terms of cost reduction. We apply this variant in Section 6.5 to the case study at Thales Nederland (see Section 2.1.1) and show that we achieve a cost reduction of almost 10% compared with the sequential approach. Also, we show that the current way of working at Thales Nederland leads to a cost reduction that is about a quarter of the cost reduction that we achieve. This chapter ends with a possible extension to our algorithm in Section 6.6 and the conclusions in Section 6.7.

6.1 Model

In this section, we outline the model that we use. The general description is presented in Section 6.1.1, while Section 6.1.2 lists the assumptions. We give the mathematical model formulation in Section 6.1.3. In Section 6.1.4 we make two further assumptions. These are not critical for our approach, but they do ease both the presentation in the remainder of this thesis and the implementation. They also decrease the problem size.

6.1.1 Description

We aim to solve the joint problem of LORA and spare parts stocking. This means that given a product design, an installed base, and a repair network, we decide on:

- which components to repair upon failure, and which to discard,
- the repair location for each component that we decide to repair,
- the location(s) (if any) where we install resources (e.g., repair equipment), and
- the locations and amounts of spare parts to stock for each component,

such that a target availability is achieved against the lowest possible life cycle costs (LCC). The LCC include all relevant costs of the LORA problem (e.g., costs of hiring service engineers and transportation of components, and costs for resources such as test equipment and tools), as well as the spare parts holding costs.

We model the LORA part of the problem similar to the model in the previous chapters; at each location there are three possible decisions for each component:

repair, discard, and move to the next higher echelon. This means that upon failure of an LRUS at the operating sites, we choose one of the three options:

- If we decide to discard a component (and purchase a new one), no further repair/discard decisions need to be taken for this component or its subcomponents (spare parts still need to be located for the component).
- If a component is repaired that has subcomponents, a decision (repair, discard, or move) needs to be taken for the subcomponent that failed, at the same location.
- If a component is moved to the location at the next higher echelon, a decision needs to be taken for that component at that location. Notice that failed components and subcomponents can move only in the direction of the highest echelon level.

To enable the repair, discard, or movement of certain components, resources may be required. Therefore, the decision that a certain component is to be repaired, discarded or moved at a certain location, means that the required resources (if any) should be installed at that location as well.

As in the standard METRIC type models, spare components may be stocked at any location where demand for that component is positive. In the standard models, each component has a fixed probability that it can be repaired at a certain location (r in METRIC terms). In our model, this parameter is determined by the LORA decisions and $r \in \{0, 1\}$. As a result, if an LRU is removed from the system that failed (echelon level 1) and repaired at the intermediate depot (echelon 2) by replacing an SRU, spare LRUS may only be stocked at echelon levels 1 and 2, since there is no demand at the higher echelon level(s)[2]. If the SRU is repaired at the intermediate depot as well, it may be stocked at echelon level 2 only, since that is the only echelon level with a positive demand for the SRU.

The goal is to achieve the lowest possible LCC subject to a constraint on the availability. In the unavailability, we only account for downtime waiting for spares.

6.1.2 Assumptions

Similar to the standard VARI-METRIC model (see, e.g., Muckstadt, 2005; Sherbrooke, 2004), we assume that:

- The number of failures (demands) in a time period of any fixed length is Poisson distributed with constant rate. This means, for example, that we

[2] Due to pooling effects of the spare parts, it may be useful to stock spare parts at a higher echelon level. However, in that case, it would generally be a good idea to perform the repair at the higher echelon level as well, thus reducing the number of required resources. In that case, the assumption is met again, since spare parts are stocked at locations with positive demand only.

do not take into account that the installed base grows at the start of its life cycle, and declines at the end of its life cycle.

- A failure in a component with subcomponents is always due to a failure in at most one subcomponent. Notice that in the previous chapters, we allowed failures to be due to a failure in two or more subcomponents simultaneously.
- A location in the repair network at echelon e is only supplied from its parent-location at echelon $e+1$, not by a lateral supply from another location at echelon e or by emergency shipments from locations at an echelon $> e+1$.
- One for one replenishment (or an $(s-1, s)$ inventory control policy) is appropriate for every component at every echelon level.

In Section 6.1.1, we mentioned the fixed probability that a component can be repaired at a certain location (r in METRIC terms) and we mentioned that in our model $r \in \{0, 1\}$. If we model multiple failure modes in one component (see Section 5.1.4), a non-integer value between 0 and 1 may result. For example, 40% of the failures in the component (failure mode A) do not require any resource and are repaired at the operating site, whereas the other 60% of the failures (failure mode B) require an expensive resource and are repaired at the central depot. However, we do not include this model refinement.

In addition to the assumptions above, we assume that at all locations at one echelon level we take the same LORA decisions (the repair/discard decisions and the location of resources); we call this *symmetrical LORA decisions*. The locations and amounts of spare parts stocks need not be symmetrical, and the repair networks that we consider need not be symmetrical either. This means that, for example, the number of operating sites per intermediate depot may differ for the various intermediate depots, or that repair costs are not equal at all operating sites. However, having a three-echelon repair network with one or more operating sites that are connected directly to the central depot is not allowed.

In symmetrical repair networks, the optimal solution consists of symmetrical LORA decisions; in asymmetrical networks this is not necessarily true. For example, if a European OEM has some customers in Asia, it might be better to perform certain repairs at a central location in Asia, whereas all other repairs are performed at the central depot in Europe. However, we assume symmetrical LORA decisions for ease of implementation and testing of our iterative method. Furthermore, in many cases, companies prefer to treat all locations at one echelon level equally, be it just for ease of communication. In Section 6.6, we discuss the extension to non-symmetrical decisions, which, among other things, leads to a growth of the problem size.

In practice, there is more flexibility than we model. For example, emergency supplies from a higher echelon can be used if a spare LRU is not available at the

operating site, or cannibalization can be used; cannibalization means that if a system needs a spare component that is not available, a component is disassembled from another system that is already waiting for a spare part. However, exclusion of these flexibility options makes the model more transparent. A similar reasoning holds for the $(s-1,s)$ replenishment policy: costs may be lower in practice due to batching of replenishments. However, the model is more transparent if we assmue a simple replenishment policy. We refer to Section 8.3.2.2 for a discussion of including more flexibility.

6.1.3 Mathematical model

In Sections 3.1.2 and 4.1, we introduced notation for the LORA model, and it is summarized in Appendix A. In Section 6.1.3.1, we introduce one set that we require in addition to the sets that we introduced before. Then, in Section 6.1.3.2, we introduce the input parameters that we require in addition to the input parameters that we introduced before, and in Section 6.1.3.3 we introduce the decision variables. Next, we give two mathematical model formulations. We present the model formulation of the general model in Section 6.1.3.4. Then, we discuss in Section 6.1.3.5 the model that results if symmetrical LORA decisions only are used; we use the latter formulation to discuss our algorithm in Sections 6.2 and 6.3.

6.1.3.1 Sets

The set D_l consists of all decisions that are available at location l, which means that for all $l \in L \setminus L_{e^{\text{CEN}}}$: $D_l = \{\text{discard}, \text{repair}, \text{move}\}$. Furthermore $D_l = \{\text{discard}, \text{repair}\}$ if $l \in L_{e^{\text{CEN}}}$.

6.1.3.2 Input parameters

In Section 4.1, we defined $\lambda_{c,l}$ as the failure rate for component $c \in C$ at operating site $l \in L_1$. Since we require, in this and next chapter, that a failure in a component with subcomponents is due to a failure in at most one subcomponent, it should now hold that: $\sum_{b \in \Gamma_c} \lambda_{b,l} \leq \lambda_{c,l}$, $\forall l \in L_1$, $\forall c \in C$.

Notice that if component c is discarded at operating site l, the observed demand of its subcomponent $b \in \Gamma_c$ at operating site l is 0. In general, the observed demand of component b at operating site $l \in L_1$ is $q_{c,b} \cdot \lambda_{c,l} \cdot X_{c,l,\text{repair}}$. Instead of $q_{c,b} \cdot \lambda_{c,l}$, we use $\lambda_{b,l}$, see also Section 4.1. This means that the observed demand of component b at operating site l is equal to either 0 or $\lambda_{b,l}$, depending on the decisions taken for its parent c. $\lambda_{b,l}$ is an input parameter, the observed demand is not.

With a similar reasoning, the observed demand of LRU $c \in C_1$ at location $l \in L \setminus L_1$ is equal to $\sum_{k \in \Phi_l} \lambda_{c,k} \cdot X_{c,k,\text{move}}$, and we define $\lambda_{c,l} = \sum_{k \in \Phi_l} \lambda_{c,k}$. For component $b \in C \setminus C_1$ at location $l \in L \setminus L_1$, the observed demand is more complicated

since it can result both from moving that component upstream in the network and from repairing its parent component at location l (the two ways that we discussed in Section 3.1.4). The observed demand of component $b \in C \setminus C_1$ at location $l \in L \setminus L_1$ is equal to $\sum_{k \in \Phi_l} \lambda_{b,k} \cdot X_{b,k,\text{move}} + q_{c,b} \cdot \lambda_{c,l} \cdot X_{c,l,\text{repair}} \mid b \in \Gamma_c$. Since $\sum_{k \in \Phi_l} \lambda_{b,k} = q_{c,b} \cdot \lambda_{c,l}$, we can define $\lambda_{b,l}$ as either $\sum_{k \in \Phi_l} \lambda_{b,k}$ or as $q_{c,b} \cdot \lambda_{c,l}$. As a result, using the defintion of $\lambda_{c,l}$ for components $c \in C$ at operating sites $l \in L_1$ given in Section 4.1, we can now define $\lambda_{c,l}$ for components $c \in C$ at locations $l \in L \setminus L_1$ recursively as $\sum_{k \in \Phi_l} \lambda_{c,k}$.

hc_c is the annual costs of holding one spare of component c.

6.1.3.3 Decision variables

There are three sets of decisions variables:

$$X_{c,l,d} = \begin{cases} 1, \text{ if for component } c \in C \text{ at location } l \in L \text{ decision } d \in D \text{ is taken} \\ 0, \text{ otherwise} \end{cases}$$

$$Y_{r,l} = \begin{cases} 1, \text{ if resource } r \text{ is located at location } l \\ 0, \text{ otherwise} \end{cases}$$

$S_{c,l}$ = the number of spare parts of component c located at location l

In addition, $\mathcal{X}$ denotes the matrix of all repair/discard decisions $X_{c,l,d}$ and $\mathcal{S}$ denotes the matrix of all spare parts decisions $S_{c,l}$.

6.1.3.4 General model formulation

The resulting model:

$$\text{minimize} \sum_{c \in C} \sum_{l \in L} \sum_{d \in D} vc_{c,l,d} \cdot \lambda_{c,l} \cdot X_{c,l,d} + \sum_{r \in R} \sum_{l \in L} fc_{r,l} \cdot Y_{r,l} + \sum_{c \in C} \sum_{l \in L} hc_c \cdot S_{c,l} \quad (6.1)$$

subject to:

$$\sum_{d \in D} X_{c,l,d} = 1 \ , \forall c \in C_1 \ , \forall l \in L_1 \quad (6.2)$$

$$X_{c,k,\text{move}} \leq \sum_{d \in D_l} X_{c,l,d} \ , \forall c \in C \ , \forall l \in L \setminus L_1 \ , \forall k \in \Phi_l \quad (6.3)$$

$$X_{c,l,\text{repair}} \leq \sum_{d \in D_l} X_{b,l,d} \ , \forall c \in C \ , \forall b \in \Gamma_c \ , \forall l \in L \quad (6.4)$$

$$X_{c,l,d} \leq Y_{r,l} \ , \forall r \in R \ , \forall (c,d) \in \Omega_r \ , \forall l \in L \quad (6.5)$$

$$\text{availability}(\mathcal{X}, \mathcal{S}) \geq \text{target availability} \quad (6.6)$$

$$X_{c,l,d}, Y_{r,l} \in \{0,1\} \quad (6.7)$$

$$S_{c,l} \in \mathbb{N} \quad (6.8)$$

Constraint 6.2 guarantees that a decision is taken for each of the failures that occur in LRUs at the operating sites. Constraint 6.3 makes sure that if the decision is taken to move a component to the next higher echelon level, a decision is taken at the location at that echelon level. Constraint 6.4 assures that if the decision is taken to repair a component at a certain location, a decision is taken for all its subcomponents at that location as well. Constraint 6.5 guarantees that repair, discard, or move actions are only performed at locations at which all required resources are available. Constraints 6.2 to 6.5 are the 'LORA constraints', and together with the first two terms in the objective function, they make up exactly the LORA model that we used in Chapter 4.[3] Constraint 6.6 is the 'spare parts stocking constraint'; it makes sure that the target availability is met. Together with the third (and last) term in the objective function, it makes up the spare parts stocking problem. However, the availability is a non-linear function of the repair/discard decisions and the spare parts decisions. Therefore, this mathematical model cannot be solved in this form, but is used to define the problem clearly. To calculate the availability, we use the sum of the expected backorders (EBO) of the LRUs $c \in C_1$ at operating sites $l \in L_1$. For details, we refer to Section 7.1, Sherbrooke (2004), or Muckstadt (2005).

6.1.3.5 Model formulation using symmetrical LORA decisions

We can add a constraint to the model formulation presented in Section 6.1.3.4 to achieve symmetrical LORA decisions. However, we prefer to present a simplified model with far less decision variables.

First, we define the input parameters that we use in the new model, using the definitions of the input parameters that we used in the model in Section 6.1.3.4: for all echelon levels e and all components c: $\lambda_c = \sum_{l \in L_e} \lambda_{c,l}$. Notice that we defined $\lambda_{c,l}$ recursively in Section 6.1.3.2 such that this equality holds indeed. For all decisions d for all components c at all echelon levels e: $vc_{c,e,d} = \frac{\sum_{l \in L_e} vc_{c,l,d} \cdot \lambda_{c,l}}{\sum_{l \in L_e} \lambda_{c,l}}$. And for all resources r at all echelon levels e: $fc_{r,e} = \sum_{l \in L_e} fc_{r,l}$. Notice that we define λ_c and $vc_{c,e,d}$ in the same way as we did in Chapter 3.

Second, we define new decision variables, using the definitions of the decision variables presented in Section 6.1.3.3. It holds for all decisions d for all components c at all echelon levels e that $X_{c,e,d} = X_{c,l,d}$, $\forall l \in L_e$. Next, it holds for all resources r at all echelon levels e that $Y_{r,e} = Y_{r,l}$, $\forall l \in L_e$. Notice that we define $X_{c,e,d}$ in the same way as we did in Chapter 3.

Since the models are very similar, it should be clear how the constraints in the model in the previous section relate to the constraints in the model that we show below.

[3] The formulation of the LORA model in Chapter 4 is different from the formulation that we use here. The latter formulation is easier to grasp; the former formulation is faster, and is therefore used in our implementation. Furthermore, notice that the formulation here looks like the formulation in Chapter 3 (although data is aggregated per echelon level in that formulation), but the constraint on the resource-component relations is more general (Equation 6.5).

$$\text{minimize} \sum_{c\in C}\sum_{e\in E}\sum_{d\in D} vc_{c,e,d}\cdot\lambda_c\cdot X_{c,e,d} + \sum_{r\in R}\sum_{e\in E} fc_{r,e}\cdot Y_{r,e} + \sum_{c\in C}\sum_{l\in L} hc_c\cdot S_{c,l} \quad (6.9)$$

subject to:

$$\sum_{d\in D} X_{c,1,d} = 1\ ,\forall c\in C_1 \quad (6.10)$$

$$X_{c,e,\text{move}} \le \sum_{d\in D_{e+1}} X_{c,e+1,d}\ ,\forall c\in C\ ,\forall e\in E \mid e\neq e^{\text{cen}} \quad (6.11)$$

$$X_{c,e,\text{repair}} \le \sum_{d\in D_e} X_{b,e,d}\ ,\forall c\in C\ ,\forall b\in\Gamma_c\ ,\forall e\in E \quad (6.12)$$

$$X_{c,e,d} \le Y_{r,e}\ ,\forall r\in R\ ,\forall (c,d)\in\Omega_r\ ,\forall e\in E \quad (6.13)$$

$$\text{availability}(\mathcal{X},\mathcal{S}) \ge \text{target availability} \quad (6.14)$$

$$X_{c,e,d}, Y_{r,e} \in \{0,1\} \quad (6.15)$$

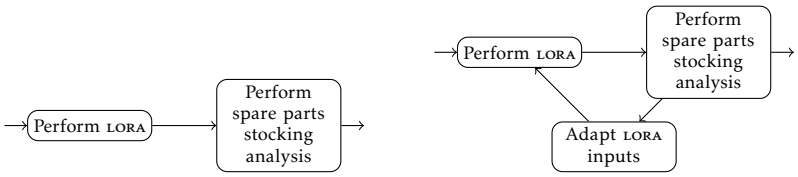
$$S_{c,l} \in \mathbb{N} \quad (6.16)$$

In the third term of the objective function and in the spare parts stocking constraint, Constraint 6.14, we use the data per location, instead of per echelon level. The reason is that we assume that the LORA decisions are symmetrical, but the spare parts decisions need not be symmetrical. If, for example, lead times differ for the various operating sites, the amount of spare parts to stock at each operating site may differ as well.

6.1.4 Demarcation

Now that we have displayed the model that we use, we make two additional assumptions that are used in the remainder of this thesis. These are not critical for our approach, but they do ease both the presentation in the remainder of this thesis and the implementation. They also decrease the problem size.

- We only consider resources that are required to enable the repair option; resources that are required for discard and move do not occur frequently in practice (e.g., not in the case study).
- Discard costs are equal at all echelon levels, which is a reasonable assumption since the main part of the discard costs consists of the costs of acquiring a replacement component. As a result, we consider one discard option only: discard at the central depot (most upstream location in the network). This means that spare components can be stocked at any location in the repair network upstream from and including the location where its parent component is repaired. LRUS, which do not have a parent component, can be stocked at all locations in the repair network.



(a) Sequential (b) Iterative

Figure 6.1: Schematic representation of the algorithms

6.2 General approach

The basic way of executing a LORA and spare parts stocking analysis, is by the sequential approach as depicted in Figure 6.1a. A LORA recommends for a given product design and repair network, which components to repair, and which to discard, where to perform the repairs, and where to locate resources. The goal is to achieve the lowest possible life cycle costs. These costs consist of both fixed costs ($\sum_{r\in R}\sum_{e\in E} fc_{r,e}\cdot Y_{r,e}$, see the objective function 6.9), and costs that vary with the number of failures ($\sum_{c\in C}\sum_{e\in E}\sum_{d\in D} vc_{c,e,d}\cdot\lambda_c\cdot X_{c,e,d}$).

The spare parts holding costs are sometimes included (see, e.g., Saranga and Dinesh Kumar, 2006), but it is not clear how these costs should be estimated. The reason is that the number of spare parts that should be stocked as a result of a repair/discard decision is not only related to the repair/discard decision for the component itself, but also to the decisions for other components. For example, if an availability of 95% should be achieved for a product consisting of two components (A and B), then an unavailability of at maximum 5% is allowed for the two components (the availability of component A times the availability of component B should be higher than 95%). It is not necessarily optimal to achieve an unavailability of 2.5% for both components; if component A is relatively inexpensive, it may be optimal to achieve an unavailability of 1.0% for component A, and an unavailability of 4.0% for component B. Furthermore, if the repair/discard decision for component B is changed, so that the lead time for that component is reduced, then less spares are needed to achieve the same unavailability of 4.0%. As a result, it may now be optimal to achieve an unavailability of 3.0% for component B and an unavailability of 2.0% for component A. VARI-METRIC, which uses a so-called system approach, implicitly finds the best way of dividing the allowed unavailability over the components. An item approach fails to do this due to its focus on individual components.

Therefore, we see in practice (e.g., at Thales Nederland) that spare parts holding costs are not included in the LORA. Instead, given the repair/discard decisions that result from the LORA, a spare parts stocking analysis is conducted (e.g., using VARI-METRIC) to determine where to locate spare parts in the repair network such that a target availability of the installed base is achieved against the lowest possible spare parts holding costs ($\sum_{c\in C}\sum_{l\in L} hc_c\cdot S_{c,l}$).

Our iterative approach, which is depicted in Figure 6.1b, uses the same two building blocks as the sequential approach does. However, instead of stopping after executing one LORA and one spare parts stocking analysis, we use the annual spare parts holding costs that result from the spare parts stocking analysis to adapt the LORA inputs, and perform a second iteration of LORA and spare parts stocking. The goal of the feedback loop is to incorporate an estimate of the spare parts holding costs that result from a LORA decision in the LORA problem. In this way, we aim to find a LORA solution that leads to low total costs. Since the LORA decisions are used to solve a spare parts stocking problem, we satisfy the constraint on the availability after each iteration.

In terms of our mathematical model formulation in Section 6.1.3.5, we first solve the model without the last term in the objective function, and without Constraint 6.14. Then, using the repair decisions ($\mathcal{X}$) as an input, we use VARI-METRIC to determine the amount of spare parts to stock for all components in the complete network ($\mathcal{S}$), such that Constraint 6.14 is satisfied, and the spare parts holding costs ($\sum_{c\in C}\sum_{l\in L} hc_c \cdot S_{c,l}$) are minimized. The aim of our feedback loop is then to add the calculated spare parts holding costs to either the variable costs ($\sum_{c\in C}\sum_{e\in E}\sum_{d\in D} vc_{c,e,d} \cdot \lambda_c \cdot X_{c,e,d}$) or the fixed costs ($\sum_{r\in R}\sum_{e\in E} fc_{r,e} \cdot Y_{r,e}$), so that these are taken into account in the LORA problem in the next iteration (the model without the last term in the objective function, and without Constraint 6.14). Below, we discuss the basic idea of our algorithm and the feedback loop. A formal description, including the stopping criterion, is given in Section 6.3.

The iterative procedure starts with executing a LORA, ignoring the spare parts holding costs, and then the execution of a spare parts stocking analysis using the LORA decisions. At this stage, after one iteration, we have a LORA solution, consisting of repair/discard decisions for each component ignoring spare parts holding costs, and the spare parts stocking solution, consisting of a stock allocation in the complete network and resulting spare parts holding costs for each component. The key idea is that, as an approximation, we may decompose the spare parts holding costs into spare parts holding costs per component, so that for each component c the spare parts holding costs are $\sum_{l\in L} hc_c \cdot S_{c,l}$. That is, if the LORA solution recommends to repair a component at a certain echelon in the repair network, or it recommends to discard the component, we assume that the spare parts holding costs that result from this decision are always the same, no matter which decisions are taken for the other components. As discussed before, this is just an approximation, since the decisions taken for one component may influence the amount of spare parts to stock for another component. However, using such a simple approach, we only need to solve a limited number of LORA and spare part stocking problems to get a (rough) estimate of the spare parts holding costs that result from all other (reasonable) LORA solutions.

We feed back the spare parts holding costs of each component in the entire network to the LORA problem; the costs are added to the costs of the repair/-

discard option that was chosen in the first iteration for that component (the technical specification can be found in Section 6.3). Most probably, this repair/-discard option will now be expensive, compared with the other possible repair/-discard options, so that another option is chosen in the LORA in the second iteration. A second spare parts stocking analysis is performed, and the spare parts holding costs that are thus found are decomposed into spare parts holding costs per component and added to the repair/discard decision taken for that component in the second iteration. By performing a couple of iterations, we will gradually find spare parts holding costs for more repair/discard decisions. We expect that after a couple of iterations, the LORA will recommend an option that leads to low total life cycle costs: LORA costs excluding the added spare parts holding costs, plus spare parts holding costs resulting from the spare parts stocking analysis. Each time we encounter a solution that leads to lower life cycle costs than the lowest we have found thus far, we store this solution.

Before we move to the technical description, we point out that we encounter two issues. First, our algorithm does not converge monotonically. Instead, the total costs may both increase and decrease during the iterations, and even cycling between two (or more) solutions is possible. This mean that we have to define a stopping criterion carefully, see Section 6.3.2.

Second, we may find different estimates for the spare parts holding costs related to a specific repair/discard decision for a certain component, since these costs are influenced by the decisions taken for the other components. In our basic algorithm, we always use the newest estimate in the next iteration. However, one may also use the oldest estimate or a weighted average of the two estimates, see Section 6.3.3.

6.3 Algorithm

After explaining our general approach, we now explain the algorithm in more detail. We distinguish four parts in our algorithm:

- The LORA building block: The minimum cost flow formulation that we presented in Chapter 4.
- The spare parts stocking building block: VARI-METRIC.
- The feedback loop: Discussed in Section 6.3.1.
- The stopping criterion: Discussed in Section 6.3.2.

We further propose three variants of the basic algorithm in Section 6.3.3.

6.3.1 Feedback mechanism

In a standard LORA, there are two cost components: the variable costs per repair/discard action $vc_{c,e,d}$, and the fixed costs to locate resources $fc_{r,e}$. In

	LORA costs (fixed and variable)		Spare parts holding costs ($vc^{s}_{c,e,d,j}$) at the end of iteration					
			1		2		3	
Component	A	B	A	B	A	B	A	B
Repair at ship	30	60	0	0	***4***	0	4	0
Repair at depot	20	35	***12***	0	12	***15***	12	***20***
Discard	30	30	0	***22***	0	22	***20***	22
Total costs (LORA and spares)			84		84		105	

Table 6.1: Costs in the LORA problem (× € 1,000)

principle, we can include the spare parts holding costs in both cost types; we choose to include them in the variable costs. For each repair and discard option (not for the move options) for each component, we add to the original variable costs $vc_{c,e,d}$ the stored spare parts holding costs $vc^{s}_{c,e,d,j}$ after iteration j (input for iteration $j+1$). In the input of the first iteration ($j=0$), all $vc^{s}_{c,e,d,0}=0$ in our basic algorithm. At the end of each iteration j, we set for each tuple $(c,e,d,)$ with $d \in \{\text{repair}, \text{discard}\}$ for which $X_{c,e,d} > 0$ in iteration j: $vc^{s}_{c,e,d,j} = \frac{\sum_{l \in L} S_{c,l} \cdot hc_c}{\lambda_c}$. For all other repair and discard decisions (all tuples $(c,e,d,)$ with $d \in \{\text{repair}, \text{discard}\}$ for which $X_{c,e,d}=0$ in iteration j), we set $vc^{s}_{c,e,d,j} = vc^{s}_{c,e,d,j-1}$.

We explain the feedback mechanism using an example. We consider a radar system that consists of two components (A and B). The radar system is installed at two ships, which are supported by a depot. So, we consider a two-echelon, single-indenture problem instance. We assume that for both components:

- the annual failure rate is 1 per ship,
- variable move costs are € 0,
- repair costs are € 5,000, and
- discard costs are € 15,000.

Both components require a unique resource in order to enable repair. The one for component A has fixed annual costs of € 10,000, the other one has fixed annual costs of € 25,000.

In the first iteration, there are no spare parts holding costs in the LORA problem. Therefore, the repair/discard options with the lowest LORA costs are chosen for both components. For component A this is 'repair at depot', since that leads to annual repair costs of € 10,000 (2 failures, 1 at each ship) and a resource at the central depot; for component B this is 'discard', since that leads to annual discard costs of € 30,000 and no resource costs. The annual LORA costs for each repair/discard option can be found in the second and third columns of Table 6.1. Next, we solve the spare parts stocking problem, and we find that

spares for component A should be stocked at both the ships and the depot, leading to annual spare parts holding costs of € 12,000 for component A only. For component B, we find annual spare parts holding costs of € 22,000. The fourth and fifth columns in Table 6.1 show the costs related to spare parts in our LORA input after the first iteration (input for the second iteration). The costs that are changed are bold and italic.

In the second iteration, we solve the LORA with the modified input. The LORA will choose for component A one of the two options 'repair at the ships', or 'discard', since both options lead to total costs of € 30,000 + € 0 (LORA costs + spare parts holding costs), whereas repair at depot leads to total costs of € 20,000 + € 12,000. We assume that repair at ships is chosen. For component B, repair at the depot is the most cost effective option. Of course, after solving the spare parts stocking problem, we find that the spare parts holding costs are not zero for these repair options. They are € 4,000 and € 15,000 for the two components respectively. We feed this information to the LORA problem again, see columns six and seven in Table 6.1.

In the third iteration, we decide to discard component A. For component B, it turns out that repair at ships is not an interesting option, even if this leads to zero spare parts holding costs. As a result, component B is repaired at depot. Executing the spare parts stocking analysis leads to annual spare parts holding costs of € 20,000 for each of the components A and B. Notice that we do not modify the LORA decision for component B, but the spare parts holding costs for component B are changed, due to a change in the repair/discard decision for component A. We simply replace the old stored costs of € 15,000 by the newly calculated costs of € 20,000 (in the basic algorithm), see the last two columns in Table 6.1. Notice that the costs of € 20,000 mean that the option to repair component B at depot has become so expensive, that it is not chosen anymore in later iterations, whereas we know that the costs may also be lower if combined with another decision for component A (for example, repairing both components at depot). Instead, in the next iterations, the solution will be to repair component A at depot and to discard component B.

The storage of spare parts holding costs that are too high is the key drawback of our approach. As a result, we may end up with a non-optimal solution, and we cannot give an indication of the quality of our algorithm compared with the optimal solution. However, we do show in the numerical experiments in Section 6.4 what we gain compared with the sequential approach. Furthermore, in Chapter 7, we give an algorithm that finds a solution that is optimal in the sense that the achieved availability cannot be achieved against lower costs (an efficient point) and except for the approximation errors in the VARI-METRIC methods. We will use that algorithm to further analyze the quality of our iterative algorithm. For now, we try to reduce the problem of storing spare parts holding costs that are too high, which is the main reason why we develop variants of our algorithm in Section 6.3.3. First however, we discuss the stopping criterion for our algorithm in Section 6.3.2.

6.3.2 Stopping criterion

As mentioned in Section 6.2, the stopping criterion for our iterative algorithm is not trivial because of the non-monotonous behaviour of the objective function. Stopping when the total costs in two consecutive solutions are almost equal does therefore not work, the costs being the total of the LORA costs without the added spare parts holding costs, plus the spare parts holding costs resulting from the spare parts stocking analysis. Instead, when defining an appropriate criterion, we have to take into account two events that may occur:

1. Two solutions are chosen alternately ('cycling'), so that the algorithm will not find better solutions anymore.

2. The costs in two consecutive iterations are almost equal, but spare parts holding costs in the LORA problem are still being changed (values $vc^{s}_{c,e,d,j}$, which are not part of the objective function). If the costs are still being changed, the LORA might still find a better solution during a later iteration. This may happen in the first few iterations, when the algorithm is still 'exploring' all the options; two different LORA solutions may accidently lead to almost the same total costs.

We can cope with the occurence of both events using a stopping criterion consisting of two conditions that should both be satisfied:

1. After each iteration, we store the solution if it is better than the best we have found thus far. We stop our algorithm if we have not found a better solution during j_1 iterations.

2. Solving the LORA problem leads to certain 'LORA costs', and certain spare parts holding costs as a result of spare parts holding costs estimates that we added to the variable costs in the LORA problem. We assume that the estimates of the spare parts holding costs in the LORA inputs (all values $vc^{s}_{c,e,d,j}$) are accurate enough, if the total spares parts holding costs that is part of the LORA solution ($\sum_{c\in C}\sum_{e\in E}\sum_{d\in D} vc^{s}_{c,e,d,j}\cdot\lambda_c\cdot X_{c,e,d}$) deviates less than p% from the total spare parts holding costs that is calculated in the spare parts stocking problem after the LORA has been solved ($\sum_{c\in C}\sum_{l\in L} hc_c\cdot S_{c,l}$). We can stop calculations j_2 iterations afterwards.

The choice of the number of iterations (j_1 and j_2) and the percentage difference p is more or less arbitrary. We choose $j_1 = j_2 = 10$ and $p = 1\%$ in our experiments. We performed tests on 80 of the most difficult problem instances (see Section 6.4) to see whether results would change if $j_1 = j_2 = 100$ and $p = 0.1\%$, but for none of the problem instances they do. Therefore, we conclude that our stopping criterion serves its purpose in the sense that we do not stop too early (which leaves open the possibility that we can stop even earlier, but this is less important given the speed of the algorithm as we will see in Section 6.4).

6.3.3 Variants

As mentioned in Section 6.3.1, the spare parts holding costs for a certain repair/-discard option may be very high in a single iteration, due to the decisions that have been taken for the other components. As a result, that option is not chosen in later iterations anymore, although it may be an interesting option if we take alternative decisions for the other components. We discuss two variants of our algorithm to cope with this issue.

In the first variant, we do not simply replace the stored spare parts holding costs in the LORA problem by newly calculated spare parts holding costs, but we take a weighted average. In this way, we aim to avoid storing spare parts holding costs that are too high. In the second variant, we run the basic algorithm until the stopping criterion is reached. We then restore the best solution we found thus far, decrease the stored spare parts holding costs that belong to this solution, and start our algorithm again. In this way, we test whether the stored spare parts holding costs for the repair/discard options that were not chosen in the final soltuion are too high.

Next to these two variants, we introduce a variant in which we try to improve the starting solution of our algorithm. Instead of starting with stored spare parts holding costs of zero for all repair/discard options, we make an initial estimate of those costs. Sections 6.3.3.1, 6.3.3.2 and 6.3.3.3 discuss each of these three variants respectively.

6.3.3.1 Using a weighted average to update stored spare parts holding costs

If our algorithm finds spare parts holding costs for a certain repair/discard option, we add these to the inputs of the LORA problem. If new spare parts holding costs are found for a repair/discard option for which we already stored spare parts holding costs in the LORA problem, we simply replace the old value with the new value in the basic algorithm. A result of this approach is that if the costs are very high in one iteration, the spare parts holding costs that we store are very high as well, and the corresponding repair/discard option may be excluded in the LORA in any later iteration. Therefore, we use a different procedure to update the spare parts holding costs in this variant: we take a weighted average of the old and new value. The new value gets weigth $0 \leq \alpha \leq 1$, and the old value gets weigth $1 - \alpha$. More specifically, if we get a new spare part holding costs estimate ($vc^{\text{est}}_{c,e,d,j}$) for option d at echelon e for component c at the end of iteration j, we do not store it directly, but instead, we store the value $vc^{\text{s}}_{c,e,d,j} = \alpha \cdot vc^{\text{est.}}_{c,e,d,j} + (1-\alpha) vc^{\text{s}}_{c,e,d,j-1}$, with:

- $vc^{\text{s}}_{c,e,d,j}$ being the spare parts holding costs stored in the database for option d at echelon level e for component c at the end of iteration j.
- α being the weight.

- $vc^{\text{est.}}_{c,e,d,j}$ being the estimate for the spare parts holding costs for option d at echelon level e for component c that results from the spare parts stocking analysis at the end of iteration j (notice that we only adapt spare parts holding costs for the repair/discard options that are chosen in the last iteration).
- $hc_{c,e,d,0}$ being 0 in the basic model, or the initial estimate as discussed in Section 6.3.3.3.

If we select the same option in a number of consecutive iterations, we gradually update the spare parts holding costs to the value that we find repeatedly, which makes the approach similar to the exponential smoothing technique in forecasting (see, e.g., Brown, 1959). A drawback of using a weighted average is that the number of required iterations will usually grow. Furthermore, since it is a different method than the basic algorithm, there is no guarantee that the solution that is found is at least as good as the solution that is found using the basic algorithm.

Besides, the second part of our stopping criterion gets more important when using a weighted average to update spare parts holding costs. The reason is that the total costs in a number of consecutive iterations may be almost equal, since the stored spare parts holding costs are not changed that much. After the spare part holding costs are adapted a few times, and are getting closer to the actual spare parts holding costs, the LORA may find another LORA solution.

6.3.3.2 Decreasing stored spare parts holding costs

Instead of preventing stored spare parts holding costs in the LORA problem of becoming too high, which is what we tried to do in the previous section, we may also decrease stored spare parts holding costs if they are too high. However, we do not know which costs are too high. Therefore, we have to decide when to decrease the stored spare parts holding costs, which costs to decrease exactly, and with how much to decrease them.

We choose to lower the costs at the moment that the basic algorithm reaches the stopping criterion. At that moment, we have reached a local optimum, and we try to get the algorithm out of that local optimum, hoping that it will find a better (local) optimum. There are various ways to decide which stored spare parts holding costs to decrease. For example:

- Decrease the spare parts holding costs for repair/discard options that have not been chosen in the last j_1 iterations (for example, $j_1 = 10$).
- Decrease the spare parts holding costs for randomly picked repair/-discard options (x% of the total number of repair/discard options, with, for example, $x = 25$).
- Decrease the x% highest stored spare parts holding costs (for example, $x = 25$).

If we know which spare parts holding costs to decrease, we can still decide to decrease all of them with the same percentage, or with the same amount of money. We can also differentiate in some way.

Since each of the options has pros and cons, we choose a relatively simple approach[4], inspired by the variable neighborhood search (see, e.g., Mladenović and Hansen, 1997): if the stopping condition is reached, we restore the stored spare parts holding costs that we had in the best iteration (the lowest total life cycle costs), which is not necessarily the last iteration. We decrease the restored costs by a certain percentage p_1 for all repair/discard options, except for those that were picked in the best LORA solution (the LORA solution in the best iteration), since those values are up-to-date. We also reset our stopping condition. If our stopping condition is reached again without finding a better solution, we restore the best solution again, and decrease all stored spare parts holding costs with a percentage $p_2 > p_1$ (again, not those that were set in the best iteration), et cetera. If we do find a better solution, we start all over again, in that we store that better solution, and when the stopping condition is reached, we lower the stored spare parts holding costs with p_1, et cetera. Obviously, the number of iterations that is required will grow. However, the solution that is found is at least as good as the solution that is found using the basic algorithm, since we extend that basic algorithm.

6.3.3.3 Initial estimate of spare parts holding costs

In the basic algorithm, all spare parts holding costs are initially zero in the LORA problem. In this variant, we try to start with more realistic estimates for the spare parts holding costs; we make these estimates using an item approach that is commonly used for fast moving consumables. Although this is not really applicable in our setting, we hope that these estimates improve the initial solution (solution after first iteration). In general, a good initial solution speeds up a local search and leads to better results on average (for examples of such behaviour, see Gademann and Schutten, 2005; Guldemond et al., 2008). We test whether this holds for our algorithm as well.

We estimate the initial spare parts holding costs ($vc^{\mathrm{s}}_{c,e,d,\mathrm{o}}$)[5] for repair/discard option d at echelon level e for component c using a safety-stock approach per item (see, e.g., Silver et al., 1998, pp.255–257). Remember that for the spare parts calculations, we use the data per location, instead of per echelon, so that our estimate is as follows: $vc^{\mathrm{s}}_{c,e,d,\mathrm{o}} = \sum_{l \in L_e} hc_c \left(\lambda_{c,l} \cdot lt_{c,l,d} + k\sqrt{\lambda_{c,l} \cdot lt_{c,l,d}} \right)$, with:

- hc_c being the annual holding costs of one spare component c.
- $\lambda_{c,l}$ being explained in detail in Section 6.1.3.2, but it can be read here as the annual demand for spares of component c at location l.

[4] We also tested a more sophisticated approach, but the change in results was minor.

[5] Remember that $vc^{\mathrm{s}}_{c,e,d,j}$ represents the holding costs for all spare parts of component c located throughout the repair network.

- $lt_{c,l,d}$ being the lead time for component c at location l if option d is selected. For the discard option, the lead time includes the order-and-ship time from the external supplier to the central depot, and the time it takes to ship it from the central depot to the operating site. For the option 'repair at central depot' in a three-echelon repair network, this lead time includes the time it takes to ship the component from the operating site to the central depot, repair it there, and ship it back to the operating site. There are two issues to notice here:

  1. If subcomponents are required to perform a repair, their unavailability may lead to an increase in the lead time. We ignore this delay, thereby underestimating the effective lead time for components with subcomponents.

  2. In the lead time, we assume that the component 'originates' at the operating site. This is true for LRUs, but for components that have a parent component, this need not be true, since the parent may have been shipped to a higher echelon to be repaired there. However, we do not know the repair location of the parent at the moment the initial spare parts holding costs estimate is made. Therefore, the lead time for components that are not LRUs, may be overestimated.

- k being the safety factor.

$\lambda_{c,l} \cdot lt_{c,l,d}$ is the mean demand over the lead time. Since we assume a Poisson distributed number of failures in any period, the variance is equal to the mean, and the standard deviation of the demand over the lead time is the square root of the mean (and of the variance): $\sqrt{\lambda_{c,l} \cdot lt_{c,l,d}}$. We estimate that the number of spare parts that need to be stored is equal to the mean demand over the lead time, plus k times the standard deviation of the demand over the lead time. Notice that the number of spares that we estimate need not be an integer, although we know that the actual number of spares will be integer. However, rounding the estimated value up to the next integer may lead to a huge overestimation of the spare parts holding costs, whereas rounding it down may lead to zero spare parts holding costs, which means that the estimate does not help with directing the LORA to choosing repair/discard options that lead to low overall costs (LORA costs and spare parts holding costs). The exact amount of spares will be calculated after the LORA has been solved, in the spare parts stocking analysis.

One of the major drawbacks of estimating the spare parts holding costs per item is that we treat all items in the same way, whereas we mentioned in Section 6.2 that in practice, relatively few items are stocked for expensive components, and relatively many items are stocked for inexpensive components. Differentiating between types of components (expensive slow movers versus inexpensive relatively fast movers) may help. However, we have analysed the results of our case study at Thales Nederland, and notice that some components are not put on stock at all, which effectively means that a negative k should be used,

whereas for some other components, one spare part needs to be put on stock at each ship. If, at the same time, the lead time demand is very small, since we have slow movers, we may effectively have a safety factor $k > 20$. A system approach is required to be able to find such results; this is inherently not possible using an item approach, particularly not if we have many (extreme) slow movers.

There is no guarantee that using an initial estimate of the spare parts holding costs leads to a solution that is at least as good as the solution of the basic algorithm. However, we hope that on average we will find a better solution, and we aim for a decrease in the number of iterations.

6.4 Computational experiments

In this section, we perform an extensive numerical experiment to answer the following questions:

1. What cost reduction can be achieved by performing the LORA and spare parts stocking analysis iteratively, compared with performing them sequentially?
2. Which variant of our iterative method gives the best results?
3. How do the repair strategies change if we use the iterative approach instead of the sequential approach?
4. Under which parameter settings does the iterative approach yield most cost reductions compared with the sequential approach?

To answer these questions, we design a numerical experiment that we present in Section 6.4.1. In Section 6.4.2, we discuss the results of our tests and answer the questions.

6.4.1 Design

In our experimental design, we restrict ourselves to problem instances that are completely symmetrical in the network structure, the cost factors, the demand rates, and the throughput times. Some parameter settings are the same in all problem instances, others are varied to see their impact. For some parameters, we give a range, instead of one value. In that case, we randomly draw values in the given range for that parameter. These randomly drawn values are the same for all settings of the other parameters.

We use three sets of problem instances; each with its own focus, which will be explained below. In each of these sets, we use a full factorial design, in that a couple of parameter settings are varied and we test each combination of parameter settings. For each of these combination of settings we generate ten problem instances, in order to obtain a variety of problem instances. Since

each parameter setting has a default value (or range) that is the same in the three test sets, there are ten problem instances that are part of each of the three sets. The parameter settings for these ten problem instances are used below, where we explain how we generate problem instances. After this explanation, we show which parameters are varied in each test set. Details on the generator can be found in Appendix E.

6.4.1.1 Default scenario

The repair network consists of three echelon levels, with a central depot, two intermediate depots, and ten operating sites. We use the following lead times:

- the discard time, so the time it takes to receive a new component at the central depot, is in the range [1/10, 1/2],
- the replenishment lead time from one echelon to the next is in the range [2/52, 4/52] (2 weeks to 4 weeks), and
- the repair time is in the range [2/52, 4/52].

The product structure consists of three indenture levels, with 50 LRUs, 100 SRUs, and 200 parts. The annual demand for a component is equal to the annual demand of its subcomponents, and the annual demand of a part (indenture level 3) is in the range $[0.01/2^2, 0.25/2^2]$. This means that the demand of each LRU is close to the range [0.01, 0.25], see for details Appendix E. The net component price is in the range [1,000, 10,000]. Using these prices, we calculate the variable costs as follows:

- repair costs as a fraction of the net component price are in the range [25%, 75%],
- for the discard, move and holding costs, we recursively add the costs of each subcomponent to its parent to get the gross component price of the parent,
- the discard costs as a fraction of the gross component price are in the range [75%, 125%],
- the move costs as a fraction of the gross component price are 1%,
- the annual costs of holding one spare part of a component are 20% of the gross component price.

There are ten resources and their annual costs are in the range [10,000, 100,000]. 50% of the resources is required by one component only, the other 50% is required by 2 to 6 components. We distinguish 4 'component types', for example electronic versus 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. The result is that resources of one component type do not interact with resources of another component type, which is realistic in practice.

6.4.1.2 Three test sets

The problem instances are divided into three sets, as mentioned above. For each of these sets, we explain the focus and we give the values for each of the parameters that are varied:

1. This is a general test set, in which we vary the problem size, the spare parts holding costs and the lead times: # Echelons: 2 & 3, # indentures: 2 & 3, # LRUs: 50 & 100, annual holding costs: [20%, 20%] & [20%, 40%], discard time: [1/10, 1/2] & [1/4, 1/2], repair time: [0.5/52, 4/52] & [2/52, 4/52], and move time: [0.5/52, 4/52] & [2/52, 4/52] (1,280 instances).
2. In this set, we make acquiring resources more and less attractive, by changing the annual number of failures, and the costs of resources and components (and thereby the variable repair, discard and move costs): Annual demand of LRU: [0.01, 0.1], [0.01, 0.25], [0.01, 0.5] & [0.01, 1], net component price: [1,000, 10,000] & [1,000, 100,000], annual resource cost: [10,00, 100,000] & [10,00, 500,000] (160 instances).
3. In this set, we change the resource-component relations in various ways: # Component types: 3 & 4, % resources used by 1 component: 0% & 50%, # components per resources: 2 to 3 & 2 to 6 (80 instances).

6.4.2 Results

In this section, we compare the results of using our algorithm with those of using a sequential approach. Since the problem instances that we use are completely symmetrical, we decided to make the spare parts decisions symmetrical as well. This means that if we decide to stock a spare part at a certain location, we stock spare parts at all locations at that echelon level. Stocking spare parts symmetrically is optimal for symmetrical networks, except that the overshoot increases (the achieved availability may be higher than the target availability).

A key result is that using our basic algorithm, we achieve a cost reduction on average over the 1,280 problem instances in test set 1 of 2.73% compared with using the sequential method. At maximum, the cost reduction is almost 35%. This means that using a sequential approach may lead to costs that are far higher than necessary. Remember that the life cycle costs of a number of sensor systems are tens of millions of euros, which means that a huge amount of money can be saved.

In Section 6.4.2.1, we use test set 1 only to test which of the proposed variants (Section 6.3.3) leads to an improvement of the results in terms of more cost reduction compared with the sequential approach, and which setting performs best for each of the variants (e.g., which α for the weighted average). After we have thus found the settings to use for each variant, we perform, still in Section 6.4.2.1, a test combining the variants for test sets 1 to 3 to see whether combining the variants leads to a further improvement.

Parameter	Values
Weighted average α	0.9, 0.8, 0.6, 0.4
Decrease factors[a]	5% – 10% – 20%, 10% – 25% – 50%
Initial spares k-value	0, 1, 2, 4, 6

Table 6.2: Parameters of the iterative algorithm variants

[a] $p_1 - p_2 - p_3$ means that if the stopping condition is met the first time, values are decreased with p_1, if no better solution is found, values are decreased with p_2, and if still no better solution is found, values are decreased with p_3.

		Cost reduction compared with					
	# Iter-	Sequential			Basic iterative		
α	ations	Ave.	Min.	Max.	Ave.	Min.	Max.
—	17	2.73%	0.00%	34.69%	—	—	—
0.9	22	2.88%	0.00%	35.49%	0.15%	-2.26%	2.79%
0.8	25	2.90%	0.00%	35.57%	0.18%	-3.55%	2.54%
0.7	29	2.92%	0.00%	34.94%	0.20%	-2.70%	2.51%
0.6	33	2.91%	0.00%	34.95%	0.19%	-2.34%	3.07%
0.5	39	2.94%	0.00%	34.94%	0.22%	-3.61%	2.77%

Table 6.3: Varying the weighted average α

In Section 6.4.2.2, we then compare the actual repair strategies that result from using the sequential approach and using our algorithm. We also analyse which parameters impact the cost reductions that can be achieved. So, in Section 6.4.2.1, we answer the first two questions that we posed at the start of Section 6.4, and we answer the last two questions in Section 6.4.2.2.

6.4.2.1 Comparison of various settings for the proposed variants

For each of the three variants, Table 6.2 lists the parameter values that we use. Note that using $\alpha = 1$ would mean that the old stored spare parts holding costs are replaced by the new value, which is what the basic algorithm does. Note, furthermore, that $k = 0$ means that we use an initial spares estimate considering the mean demand only, which is not the same as using the basic algorithm.

We test the variants one at a time, and compare them with the basic algorithm. Next, we combine the three variants, each with its best setting, and examine whether this yields a further improvement. Tables 6.3, 6.4 and 6.5 show the cost reductions that we achieve by using each of the three variants respectively. The first row in each of the tables shows the results of our basic algorithm.

Table 6.3 shows that using a weighted average to update the stored spare parts holding costs does improve the results compared with using our basic algorithm. About 0.2% additional cost reduction on average may not seem

Decrease factors	# Iterations	Cost reduction compared with Sequential Ave.	Sequential Min.	Sequential Max.	Basic iterative Ave.	Basic iterative Min.	Basic iterative Max.
—	17	2.73%	0.00%	34.69%	—	—	—
5% – 10% – 20%	66	2.93%	0.00%	34.69%	0.21%	0.00%	2.77%
10% – 25% – 50%	70	2.94%	0.00%	35.13%	0.22%	0.00%	2.77%

Table 6.4: Varying the decrease factors

much, but we are talking about tens of millions of euros over the life time of a number of sensor system, for example. Besides, the cost reduction can be more than 3%. Using a weighted average increases the time it takes to solve problem instances, but it does certainly not explode; over all tests, we found a maximum of 66 iterations or 2.5 minutes (both with $\alpha = 0.5$). We see that, with a minor bump for $\alpha = 0.6$, the average cost reduction (compared with the basic algorithm) keeps increasing with a decreasing α. However, the number of iterations keeps increasing as well. As a result, we recommend to take α as low as possible for problem instances in practice, but for our further tests, we take $\alpha = 0.7$, since that gives a good combination of cost reduction and number of iterations.

The last two columns in Table 6.3 show that using a weighted average leads to a cost reduction compared with using the basic algorithm for some problem instances, and a cost increase for other problem instances. In practice, it may be worthwhile to solve a problem instance with a couple of different settings, and to choose the best of the solutions. However, we do not perform such tests here.

Table 6.4 presents the results of decreasing the stored spare parts holding costs with a certain percentage when the stopping criterion is met. We see that it does reduce the costs with just over 0.2% on average compared with our basic algorithm. We also see that the performance of both sets of decrease factors does not differ much, and that for both sets, the number of iterations increases, as expected; this leads to an average optimization time of about one and a half minute. In contrast to using a weighted average, we see that using decrease factors never yields a solution that is worse than the solution of the basic algorithm, as explained in Section 6.3.3.2.

Since both settings yield approximately the same results, we propose to use the decrease factors of 5% – 10% – 20%, since that leads to the least number of iterations. This may especially be important if the decrease factors are combined with using a weighted average, since that may lead to a further increase of the number of iterations.

It can be seen that using an initial spare parts holding costs estimate (Table 6.5) leads to worse results on average compared with the basic iterative algorithm. For some problem instances, a significant cost reduction can be achieved (more

		Cost reduction compared with					
	# Iter-	Sequential			Basic iterative		
k	ations	Ave.	Min.	Max.	Ave.	Min.	Max.
—	17	2.73%	0.00%	34.69%	—	—	—
0	17	2.71%	0.00%	34.82%	-0.01%	-2.12%	1.66%
1	18	1.29%	-4.22%	36.12%	-1.46%	-5.19%	4.19%
2	18	0.36%	-6.38%	36.12%	-2.39%	-6.87%	4.19%
4	17	-1.86%	-10.56%	36.12%	-4.63%	-10.64%	4.18%
6	16	-2.40%	-11.61%	36.02%	-5.17%	-11.69%	4.18%

Table 6.5: Varying the initial spare parts holding costs estimate

than 4%), but we also see cost increases of more than 11%. In Section 6.3.3.3, we explained that using an item approach that is used for fast moving consumables in the setting of slow moving repairables, may not be appropriate. It turns out that this is the case. Therefore, we recommend not using an initial spare parts holding costs estimate. In practice however, it may be worthwhile to solve a problem instance with a couple of different settings, and to choose the best of the solutions.

Based on the results for each of the variants individually, we perform tests using no initial spare parts holding costs estimate, a weighted average $\alpha = 0.7$ and the decrease factors 5% – 10% – 20%. The average cost reduction compared with using the sequential method is 3.04% in test set 1, which means that the additional cost reduction over using the basic iterative algorithm is 0.32% on average (compared with 0.20% for using only $\alpha = 0.7$ and 0.21% for using only the decrease factors 5% – 10% – 20%). The cost reduction compared with the basic iterative algorithm is -2.70% at minimum (so it is actually a cost increase), and 2.73% at maximum. Again, this suggests that in practice, it may be interesting to use a number of settings when solving a problem instance and to choose the best solution. Over all three test sets, the average cost reduction compared with the sequential approach is 3.17%, and it is over 35% at maximum. Since the first iteration in the iterative algorithm is exactly the same as using the sequential approach, there can be no cost increase. The maximum number of iterations that is required is 222, and the maximum running time is just over ten minutes.

We summarize the results in this section by answering the first two questions that we posed at the beginning of Section 6.4:

1 What cost reduction can be achieved by performing the LORA and spare parts stocking analysis iteratively, compared with performing them sequentially?

The cost reduction is 3.17% on average, and over 35% at maximum.

2 Which variant of our iterative method gives the best results?

Echelon level	Sequential	Iterative	Difference
Central depot	0.00	1.87	+1.87
Intermediate depots	0.00	0.89	+0.89
Operating sites	64.63	61.66	-2.97
Total	64.63	64.42	-0.21

Table 6.6: # annually repaired LRUs

The best variant is a combination of using a weighted average when updating spare parts holding costs, with $\alpha = 0.7$, and decreasing costs when the stopping condition is reached, using decrease factors 5% – 10% – 20% (but no initial spare parts holding costs estimate).

6.4.2.2 Detailed comparison of iterative and sequential method

Now we move to the third question that we posed at the beginning of Section 6.4, so we focus on the differences in the repair strategies that result from using the sequential and iterative approach. After that, we study the impact of the various input parameter settings on the cost reductions that we can achieve (question four). We perform the tests in this section using the best combination of variants (answer to question two).

The third question that we posed is:

3 How do the repair strategies change if we use the iterative approach instead of the sequential approach?

The iterative procedure places more resources in the network than the sequential approach does. Still, the number of resources is small: 1.35 (1.18 at central depot) and 1.11 (1.10 at central depot) on average respectively (9 of the 10 different resources at maximum for both the sequential and the iterative approach). As a result, slightly more repairs and less discards are performed in the iterative solution. In addition, we see that the iterative algorithm performs some more repairs at the higher echelon levels, instead of at the operating sites for components that do not require resources. To clearly show this effect, we focus on the 470 problem instances in test set 1 for which no resources are located in the results of either the sequential or the iterative algorithm. The average cost reduction that is achieved for these problem instances is 2.90% compared with 3.12% on average over all other problem instances in test set 1. If no resources are bought, as in all the 470 problem instances, the solution is simple for the sequential algorithm: repair all components that require no resource at the operating site, thus avoiding move costs, and discard all other components. In the result of the iterative algorithm, some repairs are performed at a more central location (and some more components are discarded), see Table 6.6. If repairs are performed at the operating sites, spare parts can be stocked at the operating sites only. If repairs are performed at a higher echelon level, pooling effects for the spare parts can sometimes be

Parameter	Setting	Cost reduction compared with sequential: Average	Maximum
# indenture levels	2	2.82%	29.43%
	3	3.26%	35.02%
# LRUS	50	4.80%	35.02%
	100	1.28%	13.60%
# echelon levels	2	3.27%	35.02%
	3	2.81%	20.62%
Annual holding costs	[20%, 20%]	2.87%	32.35%
	[20%, 40%]	3.21%	35.02%
Discard lead time	[1/10, 1/2]	3.05%	34.94%
	[1/4, 1/2]	3.03%	35.02%
Repair lead time	[0.5/52, 4/52]	2.81%	35.02%
	[2/52, 4/52]	3.27%	34.52%
Move lead time	[0.5/52, 4/52]	4.91%	35.02%
	[2/52, 4/52]	1.17%	6.08%

Table 6.7: Cost reduction for each parameter setting (test set 1)

used: instead of stocking one spare part at each operating site, leading to ten spare parts in total, maybe only two spare parts are required at each of the intermediate depots, leading to four spare parts in total.

Now that we have answered question three, we can go to the fourth question that we posed:

4 Under which parameter settings does the iterative approach yield most cost reductions compared with the sequential approach?

Tables 6.7, 6.8 and 6.9 give the cost reductions that the iterative algorithm achieves compared with the sequential algorithm for the various parameter settings in test sets 1, 2 and 3 respectively. We discuss the results for test sets 1 and 2 below; in test set 3, there is no parameter that has a big impact on the achieved cost reductions.

The two parameters that really influence the cost reduction that is achieved in test set 1, are the number of LRUS and the move lead time.

The influence of the number of LRUS on the cost reduction that can be achieved relates to the total unavailability of 5% (=100%-95%) that is allowed for the complete product. We know that if we want to achieve the same availability for a product with 50 LRUS as for a product with 100 LRUS, then the unavailability that is allowed per LRU is about twice as low in the case of 100 LRUS. If the unavailability that is allowed per LRU is so low (or equivalently, the EBO per LRU are so low), then using pooling effects with the spare parts is often not possible.

Parameter	Setting	Cost reduction compared with sequential: Average	Maximum
Demand per LRU	[0.01, 0.1]	11.61%	17.00%
	[0.01, 0.25]	2.81%	5.19%
	[0.01, 0.5]	1.43%	4.07%
	[0.01, 1]	3.31%	7.91%
Component price	[1,000, 10,000]	4.84%	15.87%
	[1,000, 100,000]	4.74%	17.00%
Annual resource costs	[10,000, 100,000]	4.80%	17.00%
	[10,000, 500,000]	4.78%	16.73%

Table 6.8: Cost reduction for each parameter setting (test set 2)

If there are no spares at the operating sites, the achieved availability is already too low. To verify that this is the reason that this happens, we performed a test for all problem instances with 50 LRUs, aiming at an availability of 97.5%. In that case the cost reduction that can be achieved by using the iterative approach compared with the sequential approach decreases to 2.03%, which is still higher than 1.28% (100 LRUs, 95% target availability), but much lower than 4.80% (50 LRUs, 95% target availability).

The effect of the move lead time on the cost reduction is counterintuitive. Intuitively, long lead times lead to high spare parts holding costs, and therefore, much can be gained using an iterative procedure. However, this is not what the results show. Above, we discussed that pooling effects are used by the iterative procedure, which are not used in the sequential approach. Now, if the move lead time is relatively long, using these pooling effects is not possible: if spares are not located at the operating site, there is always a long downtime if a failure occurs, since it takes a while before the spare part arrives. This means that with increasing lead times, the cost reduction that can be achieved decreases.

In test set 2, the demand per LRU is the parameter that mainly determines the cost reduction that can be achieved. The main reason is that if the demand per LRU is very low ([0.01, 0.1]), pooling effects can be used more effectively than if demands are higher. The costs for spare parts at the central depot increase with 69% compared with the sequential solution in the case of very low demands, whereas it decreases with 11% on average for the higher demands. The reason is that if demands are very low, the availability will be high without storing many spares. For higher demands, storing at least one spare part at a lower echelon, possibly even at the operating sites, may be required to achieve the target availability. As a result, we see a cost reduction of 13% of the spare parts holding costs if we compare the iterative solution with the sequential solution for the problem instances with an LRU demand in the range [0.01, 0.1], whereas this reduction is only 4% on average for the higher demands.

Parameter	Setting	Cost reduction compared with sequential: Average	Maximum
Component types	4	2.22%	4.76%
	5	1.93%	4.10%
Components per resource	[2, 3]	2.06%	4.52%
	[2, 6]	2.09%	4.76%
% resources used by 1 component	0%	2.17%	4.52%
	50%	1.97%	4.76%

Table 6.9: Cost reduction for each parameter setting (test set 3)

6.5 Case study at Thales Nederland

We performed a case study at Thales Nederland, which is described in Section 2.1.1 and summarized in Section 6.5.2. The goal of this study is to find out which cost reduction we may obtain in practice and which advantages and drawbacks of our new approach we can identify for application in practice. In Section 6.5.1, we discuss the current way of working at Thales Nederland to solve the LORA and spare parts stocking problem, and the difficulties this leads to. Section 6.5.3 compares the results of using our iterative algorithm with those of using the sequential algorithm and those of the logistic engineers at Thales Nederland.

6.5.1 Current way of working

Data for the case study is stored in the LSAR database, where LSAR stands for logistics support analysis record. In the military world, logistics support analysis (LSA) is the activity of generating source data and a maintenance plan for a newly acquired product (from the viewpoint of the customer). The LSA is part of the integrated logistics support (ILS) program, in which much more is done. For example, it is specified what training should be given to maintenance personel and which spare parts should be acquired. DEF STAN 00-60 (PART 1) (United Kingdom Ministry of Defence, 2004b) gives more information on LSA(R) and ILS. The logistic engineers at Thales Nederland are responsible for the LSAR database.

Before analyzing the LORA decisions based on cost considerations, logistic engineers at Thales Nederland first conduct a so-called non-economic LORA. The goal of this non-economic LORA is to exclude non-realistic repair or discard options and to simplify the problem. The most important questions posed are:

- Is the component prone to failure? In the product structure, there are also so-called 'structure parts', which are, for example, cabinets in which a couple of subcomponents are installed. These cabinets themselves are

not prone to failure, and they cannot be removed easily. Instead, the subcomponents in the cabinet will be removed. In this way, it is also determined which components are the LRUs, so the lowest indenture level items that can be replaced upon failure. These LRUs may be part of a subsystem that is not replaced upon failure.

- 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.
- Are intellectual property rights involved that prohibit the customer from performing repairs? If so, the OEM can repair it, or it can be discarded.
- Is the component procurable, and will it be procurable in the future? If not, it should not be discarded.
- Does the component have any handling constraints? For example, a component may be hazardous or repair can only be performed in a dust-free and vibration-free environment. Such an environment cannot be created on board a ship (it is far too expensive), so repair at ship is not a viable option.

The result is that for some components, repair/discard options may be excluded. Multiple options may remain, but it may also happen that only one option remains. In that case, 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 the corresponding spare part inventories influence the availability.

Furthermore, only part of the resources are included in the LORA, mainly the expensive ones. For example, every engineer has a screwdriver, so they need not be considered. Besides, some expensive resources may already be available at each location, so that they need not be acquired (and taken into account in the LORA).

After the input data has been structured and filtered, the logistic engineer finds a reasonable solution. He uses 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 spare parts costs are very high. If he thinks that it might help to repair (some of) these components at a lower echelon level, he changes the LORA decision for these components, calculates the new LORA costs and solves a spare parts stocking problem. So in fact, the logistic engineer tries to perform some manual iterations, which has as its drawbacks that it is:

- Time consuming, since an analysis takes up to a few days after all data has been acquired.

- Non-reproducable, because of the judgmental feedback loop. This means that two different logistic engineers may arrive at different solutions, and even a single logistic engineer may come to a different solution if he solves the same case at a later moment.
- Error sensitive. The change of the LORA decision of a parent has impact on the LORA decision of its children. If not thoroughly checked, costs may be doubled or left out. For example, a camera unit is repaired at the OEM in the solution of the logistic engineer. However, repair costs for the subcomponents of the camera unit are taken into account at the central depot, although repair at the OEM means that these subcomponents are necessarily repaired by the OEM as well.

In Section 6.5.3 we will see that the logistic engineer finds a better solution than the sequential approach does in the case study, but our algorithm leads to considerably better results. This is not surprising, since it is very hard, if not impossible, to cover all possible combinations of repair/discard options for all components using only a spreadsheet, even though the logistic engineer is very experienced.

6.5.2 Case description

We consider a sensor system manufactured by Thales Nederland. Although the product structure in the LSAR database consists of six indenture levels, we consider only three indenture levels, as a result of the non-economic LORA as discussed in the previous section. For the same reason, the product structure in the database consists of over 1,500 components, but only slightly more than 200 turn out to be relevant after the non-economic LORA, of which 40% are LRUS. For about one third of the components, only one repair/discard option remains after the non-economic LORA, and for an additional one third of the components, the repair/discard options that can be chosen are restricted.

In practice, there is also a certain probability of successful repair, as discussed in Chapters 2 and 5. Although a probability of successful repair can easily be included in the iterative algorithm, it is much more difficult in the integrated algorithm that we develop in the next chapter. To be able to compare the results of both algorithms, we assume a 100% probability of successful repair here.

The repair network consists of twelve ships, attached to two intermediate depots, a central depot and the OEM. There are 54 resources, 34 of which are 'adapters'. These adapters are used in concurrence with other test equipment.

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 spares holding costs. For each type of costs we give the most important cost factors that we included

(another overview of the cost factors to include can be found in Saranga and Dinesh Kumar, 2006):

- 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 of the discarded component. The disposal costs can also be 'negative', which means that the component has residual value.
- Variable move costs: transportation costs and handling and administrative costs.
- Fixed resource costs: depreciation costs, costs of capital, a risk factor (e.g., insurance against all kinds of damage and theft), fixed operating costs (e.g., a location to operate the tool or test equipment), and maintenance costs of the resource. Resources may have a residual value after their economic lifetime.
- Spares holding costs: costs of capital, a risk factor, and storage costs. Spares may have a residual value after the lifetime of the product.

6.5.3 Results

We solved the problem instance using the iterative algorithm, with weighted average $\alpha = 0.7$ and decrease factors 5% – 10% – 20%, without an initial spares estimate (as discussed in Section 6.4.2.1). Solving the instance takes less than one and a half minute (20 iterations) and Figure 6.2 shows the results. A cost reduction of 9.7% is achieved compared with the sequential method, which is worth millions over the life time of twelve sensor systems. 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, and
- installing one resource at all ships instead of one at each of the two intermediate depots.

The other resources are located at the same location in both solutions. This means that 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 repair/discard costs, but also to much lower spares costs. This is profitable, since spare parts tend to be very expensive in the defence industry.

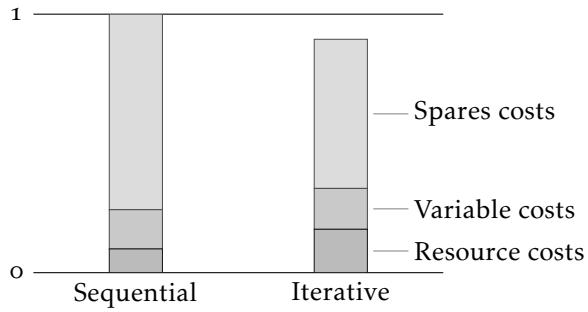


Figure 6.2: Costs for Thales case (normalised)

According to the logistic engineers at Thales, the solution that we find can be implemented in practice.

The logistic engineer at Thales Nederland achieves about a quarter of the cost reduction that we achieve using the iterative algorithm. The differences between his solution and the solution of the iterative algorithm are:

- one resource is put at depot, instead of at base,
- another is put at depot, instead of at ship, and
- two resources are not acquired at all, whereas they are located at central depot in the iterative solution.

The other resources are located at the same echelon levels in both solutions. This means that the logistic engineer puts resources at a more central location than our iterative algorithm does and he recommends to acquire less resource. However, he recommends to acquire more resources than the sequential algorithm does and puts them at more decentralized locations.

6.6 Extension to non-symmetrical LORA decisions

Although we found in Chapter 4 that aggregating all data per echelon level does not lead to a large increase in the costs in general, it was mentioned in Section 4.4 that if lead times differ across the repair network, significant cost reductions may be achieved by modelling the exact network, even for networks that are balanced in the locations.

In principle, our approach can be extended to non-symmetrical LORA decisions. For the LORA building block, there are no difficulties, see Chapter 4. Supplying the LORA decisions to the spare parts stocking problem, and solving the spare parts stocking problem causes no difficulties either. The difficulties come with the feedback loop, and more specifically, the decomposition of the spare parts holding costs over the various locations. In our current approach, we

assume that we can decompose, as an approximation, the spare parts holding costs $\sum_{c \in C} \sum_{l \in L} hc_c \cdot S_{c,l}$ into spare parts holding costs per component c, see Section 6.2. If we model non-symmetrical LORA decisions, we need to further decompose the spare parts holding costs into spare parts holding costs per location.

In some cases, this is relatively easy. Assume that in a repair network with three echelon levels, there are two intermediate depots I and II. If a component is repaired at the operating sites that are attached to intermediate depot I (first half of the network), but the component is repaired at intermediate depot II for the other operating sites (second half), it is easy to decompose the spare parts holding costs. In the first half, the spares at one operating site lead to spare parts holding costs related to the option repair at that operating site. In the other half, all spare parts lead to spare parts holding costs related to the option repair at intermediate depot II.

However, it becomes more difficult if 'flows mix'. For example, components failing in the first half are discarded (and resupplied through the central depot), whereas components failing at the other half are repaired at the central depot. In this case, both types of failures cause demand at the central depot, and therefore, it is not clear what part of the spare parts holding costs should be assigned to each repair/discard option and to each location. This leads to another approximation. An option is to use the effect of the demand at each location on the pipeline at central depot (demand times lead time), but this is further research.

A second drawback, besides the second approximation that is required, is that the problem will grow in size: there are more decision variables in each of the two problems (LORA and spare parts stocking analysis), and as a result, there are more repair options for which spare parts holding costs need to be estimated. This will probably lead to an increase in the number of iterations.

6.7 Conclusions

In this chapter, we presented an algorithm that can be used to perform the LORA and spare parts stocking analysis in an iterative way for multi-indenture, multi-echelon problem instances with very mild restrictions on the resource-component relations.

We have shown that we can solve real life problems by solving a case study at Thales Nederland, a manufacturer of naval sensors and naval command and control systems. We can solve the case study in about one and a half minute and achieve a cost reduction of almost 10% compared with performing the LORA and spare part stocking analysis sequentially.

We generated problem instances that are realistic in practice and showed that we can solve problems consisting of up to 700 components in just over ten

minutes at maximum. On average, a cost reduction of over 3% is achieved, compared with performing the LORA and inventory stocking analysis sequentially, with a maximum of over 35%.

The major drawback of our approach is that we do not know whether the solutions that we find are close to optimal. Therefore, we develop in the next chapter an algorithm that finds a solution that is optimal in the sense that the achieved availability cannot be achieved against lower costs (an efficient point) and except for the approximation errors in the VARI-METRIC methods.

Chapter 7

Integrated method for the joint problem of LORA and spare parts stocking[1]

The main drawback of the method that we presented in Chapter 6 for the joint problem of LORA and spare parts stocking, is that we do not know how close our solutions are to a theoretical optimum. Therefore, we develop a new method that finds efficient points, except for the approximation errors in the VARI-METRIC methods, which are known to be small for practically relevant problem instances. If the target availability is 95%, our method may find a solution that leads to an availability of 95.2%, which means that there is no solution with lower costs that still achieves an availability of 95.2%. However, there may be a solution with an availability between 95.0% and 95.2% that leads to lower costs. This chapter answers research question 3b: "Which method can we use to solve the joint problem of LORA and spare parts stocking in a more robust way, leading to a solution that is close to optimal?"

To find these solutions, we will develop a new, integrated method for simultaneous optimization of repair/discard decisions, location of resources, and spare parts stocking decisions. This method is based on the logic of the METRIC type methods for spare parts optimization, especially VARI-METRIC. Therefore, we discuss VARI-METRIC in Section 7.1. In Section 7.2, we outline our algorithm, and in Section 7.3, we show the results of the computational experiments. We achieve a cost reduction of 3.40% on average compared with the sequential method, with a maximum of almost 37%. Compared with the iterative method, we achieve a cost reduction of 0.26% on average, with a maximum of almost 5%. We also show the results of the case study at Thales Nederland. In Section 7.4, we present the conclusions.

[1] Based on Basten et al. (2009a)

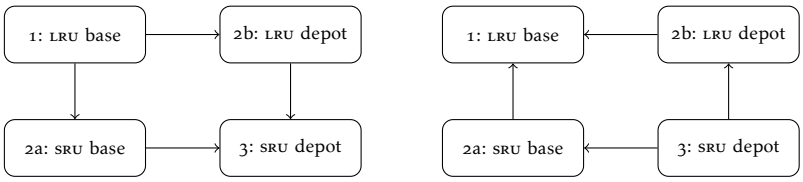
7.1 VARI-METRIC: the marginal approach

Given the repair/discard decisions per component, VARI-METRIC aims to find the most cost effective allocation of spare parts in a network that achieves a target expected operational availability of the installed base. A more elaborate discussion of VARI-METRIC and other METRIC type models, including proofs of some inferences we make, can be found in Muckstadt (2005) and Sherbrooke (2004); we base our explanation mainly on Muckstadt (2005). There is, however, one main difference. In the literature, the total investment in spare parts is minimized, whereas we minimize the total annual spare parts holding costs. Notice that if the spare parts holding costs are a certain fraction of the investment costs, the difference between these two types of costs is a constant factor so that the solution does not change (the solution value does).

Recall that an LRU family was defined as an LRU including all its subcomponents at any indenture level (see Section 2.3), and that LRUS are the first indenture components (see Figure 1.1a). Furthermore, the operating sites are the locations where the systems are located, see Figure 1.1b. We use these definitions in the explanation below.

Minimizing the costs for a given availability is the dual formulation of maximizing the availability for a given budget. Maximizing the expected availability of the installed base is approximately equivalent to minimizing LRU expected backorders (EBO) at operating sites (see, e.g., Sherbrooke, 2004, pp.39–40). A backorder arises if a request for a spare part cannot be fulfilled immediately. As an approximation, the number of LRU backorders at operating sites equals the number of systems that are down waiting for a spare part. Backorders at higher indenture levels in the LRU family or at higher echelon levels, influence the availability of the installed base only in an indirect way, since they influence the throughput times of requests for LRUS at operating sites.

For an example problem instance with two echelon levels and two indenture levels, Figure 7.1a shows that demand for LRUS at the lowest echelon level (operating sites, or bases in the terminology of Sherbrooke, 2004) leads to demand for SRUS at that level (if repair of the LRU at base is chosen) or demand for LRUS at depot (if LRUS are moved from base to depot). With an analogous reasoning, demand for SRUS at base and for LRUS at depot leads to demand for SRUS at depot. If a request for a spare SRU at depot cannot be fulfilled immediately, a backorder results. Such backorders increase the pipeline of SRUS at depot. The pipeline is defined as the number of components in repair at a location, or being resupplied to that location from a location at a higher echelon level (see, e.g., Sherbrooke, 2004, p.15). Both repairs of LRUS at the depot and the replenishments of SRUS at a base may be delayed, because a higher pipeline corresponds to longer throughput times (Little's law, see Little, 1961). Figure 7.1b shows how pipelines influence each other and shows that, for example, a higher number of backorders for SRUS at the depot leads to a higher number of backorders for LRUS at a base.



(a) Computation of the demand (b) Computation of the pipeline

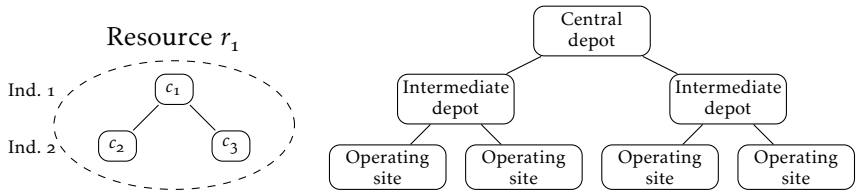
Figure 7.1: Computation of the demand and the pipeline
Based on Sherbrooke (2004, Figure 5-1)

The basic METRIC model considers the mean number of items in the pipeline only; the VARI-METRIC model, which we use, also considers the variance of the number of items in the pipeline. A negative binomial distribution is an approximation for the true distribution of the pipeline, and so VARI-METRIC is not exact, although the approximation error is known to be small for practically relevant problem instances[2]. An assumption in the VARI-METRIC model is that the failure rate is constant. For a fleet of planes at a base, this might be realistic. If one plane is not available, the other planes will probably make more flight hours. However, for a radar system on board a ship, this is not realistic, since no additional failures will occur if it is not available. This is a second reason why VARI-METRIC is not exact.

For the special case of a single system per operating site, Rustenburg (2000) shows that it is better to use the sum of the backorder probabilities for all LRUS at the operating sites (PBO) instead of the sum of the expected number of backorders (EBO). A drawback is that even for a simple single site, single indenture system the sum of the PBO is not a convex function of the spare part stock levels (for low stock levels), whereas this is true for the sum of the EBO. Due to the limited applicability and the non-convexity problem, we choose to use the EBO in this thesis. However, using PBO does not fundamentally change our approach, but it does make implementation more complicated. Furthermore, if the availability that should be achieved is high, the probability of having more than 1 backorder is very low, which means that the EBO is very close to the PBO.

Because our objective is to minimize the sum of the EBOS for all LRUS at the operating sites, the overall spare parts stocking problem is separable in small subproblems per LRU family, assuming that subcomponents are not shared between LRU families, i.e., there is no commonality (see, e.g., Sherbrooke, 2004, pp.39–40). First, we explain how to solve such a subproblem using an example. Next, we explain how to solve the overall problem given the solutions for each LRU family.

[2] As mentioned in Section 2.3, the pipeline distribution can also be calculated exactly, but this is computationally intensive.



(a) Two-indenture system structure

(b) Three-echelon repair network

Figure 7.2: Example system structure and repair network

LRU/SRU	Demand	Costs Discard	Repair	Move	Holding
LRU (c_1)	2	6	2	1	2.5
SRU (c_2,c_3)	1	2.5	2	1	1

Table 7.1: Demand and cost data of the components

Figure 7.2a shows the example product structure that we use. It represents an LRU with two subcomponents that all require the same resource for repair. We give the characteristics of the components in Table 7.1. It can be seen that failures in LRU c_1 are due to a failure in subcomponent c_2 in 50% of the cases, and due to a failure in subcomponent c_3 in the other 50% of the cases. Resource r_1 has annual costs of 7.5.

Figure 7.2b shows the example three-echelon repair network that we use. Moving a component to the next higher or lower echelon takes 1/10 of a year, repairing a component takes 1/100 of a year, and it takes 1/2 of a year before a newly purchased component arrives if a component is discarded.

Suppose that the LORA recommends to locate a resource at the central depot only and to repair all three components there. Then, notice that the repair time that we use in the calculations includes the time to move the component to that repair facility. This means that the repair time of the LRU is 1/10+1/10+1/100 = 21/100, whereas the repair time of the SRUs is 1/100. This is input for the spare parts stocking analysis.

To find the solution to a subproblem for a single LRU, as in our example, we proceed as follows (see, e.g., Muckstadt, 2005, pp.103–106):

- Construct an EBO-curve for each of the SRUs. Since the SRUs are both exchanged (replaced in the LRU) as well as repaired at the central depot, stocking them at the central depot is the only sensible option. Adding one SRU at a time leads to a curve that gives the EBO for that SRU at the central depot and the corresponding spares costs. Backorders of the SRU at the central depot may lead to a waiting time for repairs of LRUs at the

central depot.[3] The curves for each of the SRUS are convex, because these are simply independent single item, single site problems.

Result: An EBO-curve for each of the two SRUS, with the backorders of SRUS at depot as a function of SRU spare parts holding costs.

- Merge the EBO-curves of the two SRUS using marginal analysis. The idea of marginal analysis is that adding one spare SRU leads to a certain EBO-reduction per dollar (EBO of the SRUS, experienced by the LRUS at the central depot). Comparing the EBO-reduction per dollar of adding one spare SRU c_2 to the EBO-reduction per dollar of adding one spare SRU c_3, we can add that spare that leads to the highest EBO-reduction per dollar ('biggest bang for the buck'). This leads to one EBO-curve for the two SRUS, with each point on the curve corresponding to a number of spare SRUS c_2 and c_3, costs for these spares, and resulting EBO of SRUS at the central depot. Marginal analysis can be applied since each of the two SRU-curves is convex. As a result, it is guaranteed that the EBO-reduction per dollar keeps decreasing while adding more spares, so that finding the best EBO-reduction per dollar is guaranteed by looking ahead one step only. Furthermore, this also means that the curve that is constructed in this way is also convex.

  Result: EBO-curve for the two SRUS, with the backorders of SRUS at depot as a function of the SRUS spare parts holding costs.

- For each SRU investment level from the previous curve (point on the SRUS-curve), start adding spare LRUS. LRUS can be put on stock at each of the three echelon levels in our example. At operating sites, they decrease the total LRU EBO directly, at higher echelon levels, they do this indirectly; adding a spare LRU at a higher echelon level, leads to a decrease of the lead time for LRUS at the operating sites. Now we enumerate all reasonable LRU stock levels at the central depot. For each of the central depot LRU stock levels, we enumerate all reasonable stock levels at the intermediate depots. Then for each combination of a central depot LRU stock level and an intermediate depot LRU stock level, add LRUS at the operating sites, which leads to one EBO-curve per intermediate depot LRU stock level per central depot LRU stock level per SRU investment level (point on the SRUS-curve). These EBO-curves are convex.

  Result: EBO-curve for each intermediate depot LRU stock level for each central depot LRU stock level for each SRU investment level, with the backorders of LRUS at the operating site as a function of the LRU and SRUS spare parts holding costs.

- Find the lower envelope of these curves (the efficient points) and remove

[3] Muckstadt (2005, p.104) mentions that "the goal is to select [...] SRU stock levels that minimize the expected LRU waiting time for repairs [...]". In our model, this is exactly the same as selecting the SRU stock levels that minimize the SRU EBO, since we include neither repair probabilities (r in standard METRIC-notation) nor commonality.

all non-convex points so that one convex EBO-curve for the LRU family results. This curve gives the relation between total LRU backorders at the operating sites and the minimal spare parts holding costs required to obtain these backorders levels. Since non-convex points are removed, we can actually determine the convex hull, using the Graham scan (Graham, 1972)[4]. Each of the points on the curve corresponds to a number of spare LRUS (and their locations), a number of spare SRUS c_2 and c_3 (necessarily at central depot), costs for these spares, and resulting EBO of LRUS at operating sites.

Result: EBO-curve for the LRU-family, with the backorders of LRUS at the operating site as a function of the LRU and SRUS spare parts holding costs.

To find the solution to the overall problem consisting of multiple LRU families, the EBO-curves of all LRUS are merged, using marginal analysis, in the same way as the curves of the SRUS were merged. The overall EBO-curve that results is used to find the first point that achieves the target availability. The EBO-level that corresponds with this point is optimal for the spare parts holding costs that correspond to this point. Since maximizing the availability is approximated only by minimizing the backorders, the corresponding availability is not necessarily optimal for the given spare parts holding costs (and thus the lowest total costs, given the LORA decisions). Furthermore, since we find the efficient points only, the achieved availability may be slightly higher than the target availability (overshoot).

For the total costs in our example (LORA costs and spare parts holding costs), we have to add the costs of the resource and the variable repair and move costs. In our example, the total EBO of the LRUS at the four operating sites will be 3.36, and total costs will be 55.50, if no spares are stocked (first point on the curve). The resulting EBO-curve is shown in Figure 7.3.[5]

7.2 Algorithm

Section 7.2.1 describes the general idea of our algorithm, which is inspired by that of Alfredsson (1997). In Section 7.2.2, we show how this algorithm is used if all components in one LRU family together require at most one resource. Section 7.2.3 discusses the general case, in which any number of resources may be required. In the remainder of this chapter, we take symmetrical spare parts decisions. This means that if we decide to stock a spare part at one location, we stock spare parts at each location at that echelon level. This is optimal for problem instances that are completely symmetrical in the network structure, costs, et cetera, except for the overshoot problem that was mentioned in the

[4] The Graham scan is used to determine the convex hull of a set of points. We are only interested in the convexification of the lower envelope, which means that we can stop the scan before we have the complete hull.

[5] Although we use VARI-METRIC in our implementation, we use METRIC in this example since those results are easier to replicate by the interested reader.

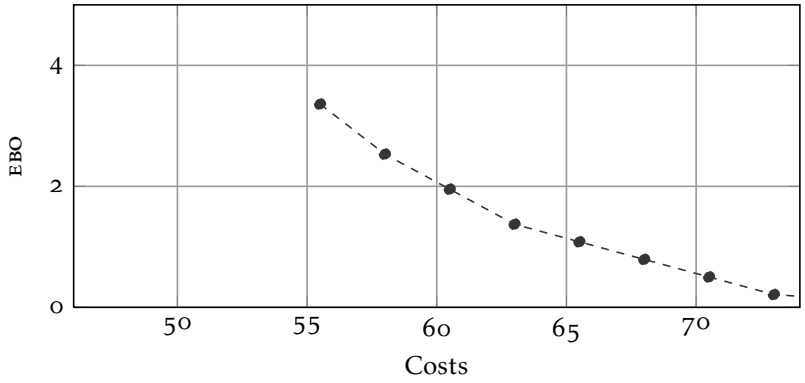


Figure 7.3: EBO-curve for one LRU family, repair at the central depot

previous section. We assume symmetrical spare parts decisions since it reduces the running time to solve problem instances. However, our approach is not restricted to this assumption.

7.2.1 General idea

In Section 7.1, we have shown how to determine spare parts stocks, given repair/discard decisions. A straightforward approach to solve the integrated problem of LORA and spare parts stocking, is to consider all combinations of repair/discard decisions, stock spare parts for each combination, and determine the lower envelope. Determining the lower envelope is also done in the METRIC type models to find the efficient points over the various LRU EBO-curves.

Due to the number of possible combinations to consider, this approach is only realistic for very small problem instances. However, we can decompose the problem into subproblems, using an algorithm that is inspired by that of Alfredsson (1997). He iteratively decomposes the problem in two ways (we will refer to them as decomposition type 1 and 2 respectively). In our algorithm, we use the same two ways of decomposing the problem, which we explain in detail Sections 7.2.2 and 7.2.3:

1. Alfredsson decomposes the problem into subproblems per resource. Splitting per resource is allowed since the location of one resource does not influence the location of other resources in the model. This results from assuming one indenture level only and the assumption that each component (LRU) requires exactly one resource to be repaired. The author further decomposes the problem into subproblems per LRU, as in the standard METRIC type models. In Section 7.1, we have shown how marginal analysis can be used to merge the results after solving each subproblem.

   We also decompose the problem into subproblems per resource in the special case in Section 7.2.2. In the general case, it is slightly more com-

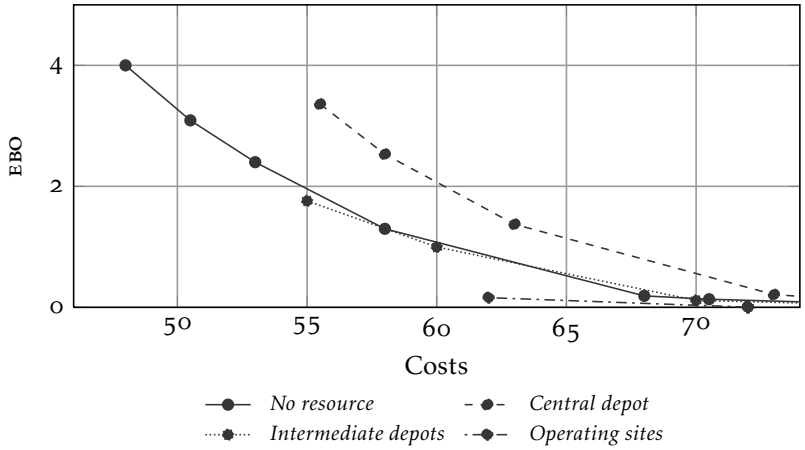


Figure 7.4: EBO-curves of one LRU, excluding convex overall EBO-curve

plicated, as will be explained in Section 7.2.3. We further decompose the problem into subproblems per LRU as well.

2. Alfredsson decomposes the problem into subproblems by fixating decision variables. For example, each resource in his model needs to be at exactly one of the two echelon levels that he assumes. For each resource, he solves a subproblem in which the resource is at the lower echelon, and a subproblem in which the resource is at the higher echelon. By combining the curves that result from each subproblem and finding the convexification of the lower envelope, the overall results are found.

   We also fixate decision variables, thus decomposing the problem into subproblems. Figures 7.4 and 7.5 show this idea graphically for the example discussed in Section 7.1. In that section, the spare parts stocking problem was solved under the assumption (LORA output) that the resource was available at the central depot and that all components were repaired there. We get four such curves by assuming that the resource is either available at operating sites, intermediate depots, central depot, or nowhere. In the first three cases, a further assumption is that all components are repaired at the same location, in the latter case, all components have to be discarded. Figure 7.5 also shows the convexification of the lower envelope.

   Notice that in our model, many more valid repair/discard decisions are possible. For example, the resource may be available both at the operating sites and at the central depot, so that the LRU and SRU 2 are repaired at operating site, but SRU 3 is repaired at the central depot. It may also happen that (one of) the SRUs are discarded, although the resource is available at at least one echelon level.

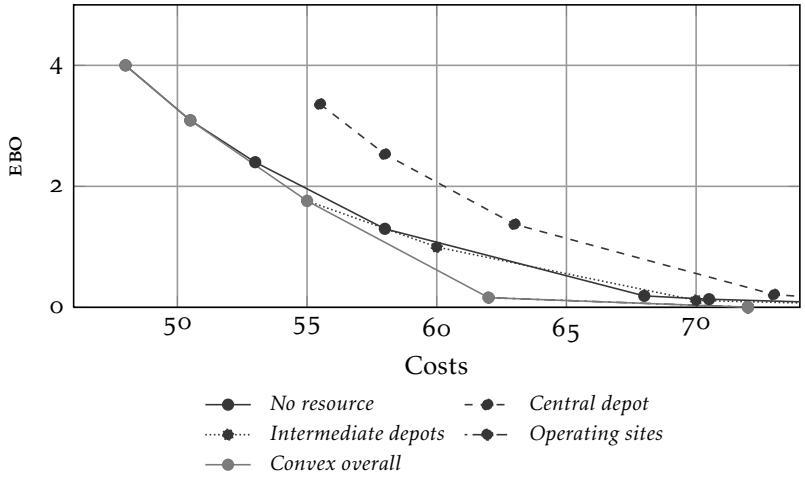


Figure 7.5: EBO-curves of one LRU, including convex overall EBO-curve

7.2.2 Special case: at most one resource per LRU-tree

If all components in an LRU family together require only one resource, we can decompose the problem as follows.

- Using a type 1 decomposition, we decompose the problem into subproblems per resource. For each resource there are $2^{|E|}$ scenarios, where E is the set of echelon levels in the repair network. We get these scenarios by either installing or not installing the resource at each of the echelon levels.
- We decompose (type 1 decomposition) each of the subproblems into a subproblem per LRU family, just as the standard METRIC type models do. Each LRU family that requires no resource at all, makes up such a subproblem too.
- We decompose (type 2 decomposition) each of these subproblems per LRU family by fixating either a repair-location for the LRU, or choosing the discard option. Notice that due to the location of the resource, some repair options might not be available. For example, if the repair of a component can only be performed if a certain resource is available, and that resource is not available at the operating sites, the component cannot be repaired at the operating sites.
- For the example product structure in Figure 7.2a, let us assume that we are solving the subproblem in which the resource is available at the operating sites, and the LRU is repaired at the operating sites. Section 7.1 explained that we have to construct an EBO-curve for each of the SRUs, merge these, and start adding spare LRUs for each of the resulting points

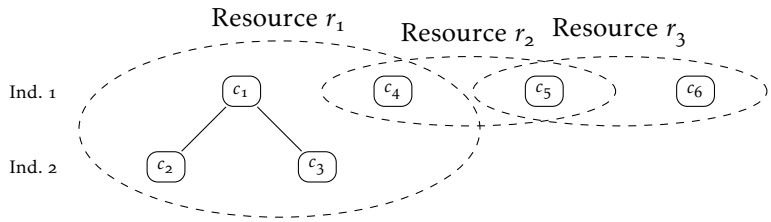


Figure 7.6: Two-indenture product structure

on the SRUS-curve. However, for each of the SRUS, there are still two options: repair it at the operating sites or discard it. Therefore, constructing the EBO-curve for each of the SRUS is somewhat more difficult. We decompose the problem by fixating the LORA decision for the SRU (type 2 decomposition). That is, we construct one curve assuming that the SRU is repaired at the operating sites, and one curve assuming that it is discarded. We find the convexification of the lower envelope of these two curves, which leads to one convex EBO-curve for each of the SRUS. Merging the curves for the two SRUS and solving the problem for the LRU can now be done in the same way as it is done in the METRIC type methods. This approach can be used recursively if there are multiple indenture levels in the product structure.

Merging the results of all subproblems means performing marginal analysis for type 1 decompositions and finding the convexification of the lower envelope for type 2 decompositions.

7.2.3 General case: multiple resources per LRU-tree

In general, one component may require multiple resources and components in one LRU family may need various resources. In that case, resources cannot be considered independent of each other, but form one *resource group*. Consider the example product structure in Figure 7.6 and a three-echelon repair network (see Figure 7.2b). To repair LRU c_4, resources r_1 and r_2 are required, and to repair LRU c_5, resources r_2 and r_3 are required. Therefore, the decision to install resource r_2 depends on the decision to install resource r_1 and vice versa. It also depends on the decision to install resource r_3 and vice versa. We say that resources r_1, r_2, and r_3 make up one resource group. Below, we give an exact definition of a resource group.

In Section 7.2.2, we showed that for each resource there are eight possible scenarios in a repair network with three echelon levels ($2^{|E|}$). If we consider all combinations of resource locations in our example, we get $8^3 = 512$ scenarios. In general, we get $2^{|E|^{|R^{\text{group}}|}}$ scenarios with $|R^{\text{group}}|$ being the number of resources in the resource group.

Since the number of scenarios explodes if the number of interacting resources

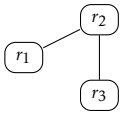


Figure 7.7: Graph of three resources

increases, we decompose the problem into subproblems (type 2 decomposition). In our example, we observe that once we know where resource r_2 is located (which may be at multiple levels in the repair network), resources r_1 and r_3 can be treated independent of each other (but not indepedent of resource r_2). Therefore, we make eight main scenarios in which one or more locations for resource r_2 are fixated. For each of these main scenarios, there are two subproblems: the first consists of resource r_1 and LRU families 1 and 4, the second consists of resource r_3 and LRU families 5 and 6. For each of these subproblems, there are eight scenarios. This means that for each of the eight main scenarios, we have to construct eight EBO-curves for each of the subproblems, resulting in $8 \cdot (8 + 8) = 128$ EBO-curves. Notice that in each of the 128 EBO-curves, we consider two LRUs as opposed to four LRUs for each of the 512 original curves.

In Section 7.2.2, we explained how to obtain the results for each of the subproblems (one resource and two LRU families). Using marginal analysis, we merge the results of the two subproblems to get a result for each main scenario. Finding the convexification of the lower envelope over the eight main scenarios leads to the overall EBO-curve for each resource-group.

Now the problem is how to decompose a general problem with shared resources in independent subproblems so that the total problem can be solved efficiently. To this end, we represent the interaction between the resources in a graph: a vertex represents a resource, and an edge between two vertices exists if there exists an LRU family that uses both resources. Figure 7.7 represents the graph for our example. Below, we first give a number of definitions, then we explain how we use the graph representation of the interaction between the resources. We assume a familiarity with the basics of graph theory; for further definitions, we refer to any book on general graph theory, for example, Diestel (2005) or Godsil and Royle (2001).

A graph $G = (V, E)$ consists of vertices $v \in V$ and edges between vertices $(v, w) \in E$. If graph $G' = (V', E')$, with $V' \subseteq V$ and $E' \subseteq E$, then G' is a subgraph of G. A graph is connected if any two of its vertices are linked by a path (the graph consisting of one vertex is connected as well). A maximal connected subgraph $G' = (V', E')$ is a subgraph of $G = (V, E)$, such that adding any more vertices $v \in V \setminus V'$ (and edges $(v, w) \in E \setminus E'$) leads to a disconnected subgraph. In other words, a maximal connected subgraph $G' = (V', E')$ is a connected graph and a subgraph of $G = (V, E)$, and there exists no graph $G'' = (V'', E'')$ so that G'' is a subgraph of G, G' is a subgraph of G'', and G'' is connected. A maximal connected subgraph is also called a connected component or just component; we will use the term 'graph component' to avoid confusion. A

depth first search can be used to identify the graph components in a graph (see, e.g., Hopcroft and Tarjan, 1973). A graph is complete if any two vertices are connected with each other by an edge: if for all vertices $v_1, v_2 \in V$ with $v_1 \neq v_2$ it holds that $(v_1, v_2) \in E$, then the graph $G = (V, E)$ is complete.

We now define a *resource group* as the set of all resources represented by the vertices in one graph component. We have already shown that the number of options for the resources in a resource group is $2^{|E|^{|R^{\text{group}}|}}$. We also know that fixating the location(s) for one resource leads to $2^{|E|}$ scenarios. Our goal is to decompose any resource group in such a way that we get the smallest number of scenarios.[6] Notice that decomposition of the resource group is useful only if it is represented by a graph component that is not complete. We decompose the problem using a recursive approach on the graph representation of the resource group as follows.

1. Check whether the graph component, representing a resource group, is complete. If so, we are done with this graph component and the number of scenarios is $2^{|E|^{|R^{\text{group}}|}}$. Otherwise, go to step 2.
2. For each vertex, representing a resource, in the graph component, remove the vertex from the graph component.[7] Using a depth-first search, find the graph components in the new graph. For each of the new graph components, go to step 1. The total number of scenarios is $2^{|E|}$ times the summation of the number of scenarios in each new graph components. Go to step 3.
3. Over all the vertices that can be removed, choose the vertex that leads to the smallest number of scenarios. This vertex represents the resource that should be fixated.

In this way, we enumerate all possible ways to decompose the problem and find the best way to do this. Since the number of resources is limited, this approach takes less than 0.1% of the total running time of the algorithm in our experiments (less than 1 second for our case-study at Thales). Hence, it does not make sense to find a more efficient method for problem decomposition.

7.3 Test results

In this section, we answer three questions:

1. What cost reduction can be achieved by using the integrated algorithm, compared with the iterative algorithm or the sequential approach?

[6] Notice that some scenarios may consist of many LRUs, whereas others may consist of a few LRUs only. One might want to incorporate this in the search for the best way of decomposing the resource group, but for sake of simplicity, we do not do that.

[7] Do not remove a vertex that is connected to one other vertex only, since that does not help in decomposing the graph component.

2. Which model parameters influence the cost reductions that we achieve?
3. Which model parameters influence the running time that we achieve?

The problem instances that we use to answer these questions are generated using the generator that was presented in the previous chapter (Section 6.4.1) and is described in detail in Appendix E. In Section 7.3.1, we answer the first two questions, using both the theoretical problem instances and the case study at Thales Nederland; in Section 7.3.2, we answer the third question. For the iterative algorithm, we use the best variant as found in Section 6.4.2.1: a combination of using a weighted average when updating spare parts holding costs, with $\alpha = 0.7$, and decreasing costs when the stopping condition is reached, using decrease factors 5% – 10% – 20%.

7.3.1 Comparison of algorithms

In this section, we answer the first two questions that we posed, so we focus on the cost reduction that we achieve using the integrated approach compared with the iterative and sequential approaches and the influence of the various parameter settings on this cost reduction. To this end, we first study the differences between the various approaches for the problem instances that we generated. Next, we use our integrated approach to find out to which extent we can improve the solution that we found for the Thales case in Section 6.5 using our iterative approach.

7.3.1.1 Numerical experiment

We start with answering question 1:

1 What cost reduction can be achieved by using the integrated algorithm, compared with the iterative algorithm or the sequential approach?

On the generated problem instances in the three test sets, our algorithm gives an average cost reduction of almost 3.40% compared with the sequential approach, with a maximum of almost 37%. Compared with the iterative method, we achieve a cost reduction of 0.26% on average, with a maximum of almost 5%. This means that the iterative approach works pretty well in most cases. Still, the maximum cost deviation shows that there are problem instances on which the integrated method performs significantly better. Furthermore, we found some problem instances for which the integrated approach yields higher costs than the iterative approach. Although this seems impossible at first sight, the explanation is that for those problem instances, the integrated approach achieves a higher availability than the iterative approach: our integrated method gives us an efficient frontier, so a set of optimal combinations of total costs and operational availability. We do, however, not find an optimal solution for an arbitrary value of the operational availability, which is also true for the standard VARI-METRIC method, and METRIC type methods in general.

Achieving a higher availability is called 'overshoot', as explained in Section 7.1. Comparing the operational availability, we see that the iterative approach finds solutions that are on average closer to the target value of 95% than the integrated method, 95.07% and 95.20% respectively (the maximally achieved availability is 96.12% and 96.24%, respectively). In principle, we can reduce this issue by accounting for the increase in availability upto the target availability only in our marginal analysis approach (any additional increase in availability is 'useless'), see also Kranenburg (2006, p.26). Another option is to post-process our solution, and check if we can remove a spare spart, thus reducing the costs, while still achieving the target availability. Since any additional availability is expensive if the availability is already quite high, overshoot is expensive, and it may be worthwhile to lower the overshoot.

Now that we know what cost reductions can be achieved on average, we move on to answering the second question that we posed:

2 Which model parameters influence the cost reductions that we achieve?

We show detailed results in Tables 7.2, 7.3 and 7.4 for test sets 1, 2 and 3, respectively. We explained above why it can happen that we see small cost increases, for example in the last row of Table 7.2. The cost reductions that our integrated model achieves compared with the sequential approach (see fourth and fifth column in the tables), are in line with the cost reductions that the iterative model achieves compared with the sequential approach (see Tables 6.7, 6.8 and 6.9). Therefore, we refer to Section 6.4.2.2 for a discussion.

Here, we only focus on the cost reduction that the integrated model achieves compared with the iterative model. In test set 1, the largest cost reduction on average is achieved in problem instances with a large number of echelon levels or small move lead times. An increase in the number of echelon levels leads to an increase in the possible repair/discard options. Therefore, it is not surprising that the iterative approach has more problems finding the optimal solution. On the other hand, we see that the maximum cost reduction that may be achieved decreases if the number of echelon levels increases. More importantly, in general if the problem size increases (number of indenture levels, number of LRUs, and number of echelon levels) the maximal cost reduction decreases, which means that the worst case performance of the iterative method improves with an increasing problem size.

The influence of the move lead time on the cost reduction that can be achieved using an iterative algorithm instead of using the sequential approach was explained in Section 6.4.2.2: the iterative algorithm performs repairs at a higher echelon level, so that it can use pooling effects of the spare parts. The reason that the integrated approach performs even better, is that the iterative approach simply misses some opportunities; the integrated approach performs even more repairs at higher echelon levels, thus making more use of the pooling effects of the spare parts, especially if the move lead times are low.

Parameter	Setting	Time (s.)	Cost reduction compared with Sequential Ave.	Sequential Max.	Iterative Ave.	Iterative Max.
# indenture levels	2	24	3.00%	30.58%	0.21%	4.88%
	3	257	3.44%	36.88%	0.21%	3.17%
# LRUs	50	225	4.98%	36.88%	0.22%	4.88%
	100	56	1.46%	14.58%	0.19%	3.17%
# echelon levels	2	8	3.38%	36.88%	0.13%	4.88%
	3	273	3.07%	21.25%	0.28%	3.24%
Holding costs	[20%, 20%]	144	3.04%	34.37%	0.19%	3.60%
	[20%, 40%]	137	3.40%	36.88%	0.23%	4.88%
Discard lead time	[1/10, 1/2]	140	3.22%	34.74%	0.20%	4.53%
	[1/4, 1/2]	141	3.22%	36.88%	0.21%	4.88%
Repair lead time	[0.5/52, 4/52]	132	2.97%	36.88%	0.19%	4.53%
	[2/52, 4/52]	149	3.47%	35.68%	0.23%	4.88%
Move lead time	[0.5/52, 4/52]	130	5.29%	36.88%	0.44%	4.88%
	[2/52, 4/52]	150	1.15%	7.14%	-0.02%	3.03%

Table 7.2: Cost reduction and optimization time for each parameter setting (test set 1)

Parameter	Setting	Time (m.)	Cost reduction compared with Sequential Ave.	Sequential Max.	Iterative Ave.	Iterative Max.
Demand per LRU	[0.01, 0.1]	8	12.06%	17.33%	0.50%	2.44%
	[0.01, 0.25]	14	2.73%	5.90%	-0.08%	1.78%
	[0.01, 0.5]	25	2.77%	5.58%	1.35%	2.53%
	[0.01, 1]	53	4.61%	8.88%	1.34%	2.17%
Component price	[1,000, 10,000]	28	5.59%	15.92%	0.77%	2.53%
	[1,000, 100,000]	21	5.49%	17.33%	0.78%	2.44%
Resource costs	[10,000, 100,000]	25	5.61%	17.33%	0.84%	2.53%
	[10,000, 500,000]	25	5.48%	17.18%	0.72%	2.31%

Table 7.3: Cost reduction and optimization time for each parameter setting (test set 2)

In test set 2, the cost reduction that is achieved using the integrated algorithm compared with the iterative algorithm is mainly determined by the demand per LRU. Although we see a clear difference between the setting [0.01, 0.25] and the other settings, we are not able to explain these results. Sta In test set 3, none of the parameter settings really influences the cost reductions that can be achieved.

Parameter	Setting	Time (m.)	Cost reduction compared with Sequential Ave.	Sequential Max.	Iterative Ave.	Iterative Max.
Component types	4	19	2.25%	4.84%	0.03%	1.69%
	5	23	1.88%	4.38%	-0.05%	1.40%
Components per resource	[2, 3]	11	2.00%	4.54%	-0.06%	1.04%
	[2, 6]	30	2.12%	4.84%	0.03%	1.69%
% res. used by 1 comp.	0%	32	2.23%	4.54%	0.06%	1.69%
	50%	10	1.89%	4.84%	-0.09%	1.05%

Table 7.4: Cost reduction and optimization time for each parameter setting (test set 3)

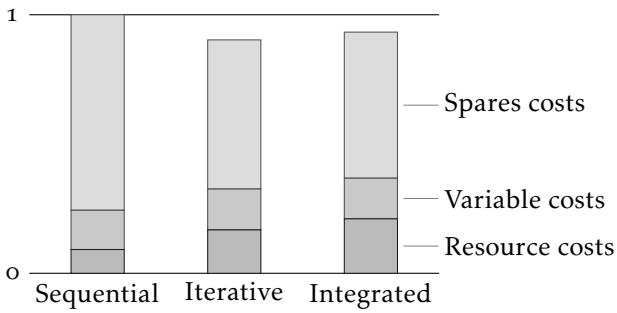


Figure 7.8: Costs for Thales case (normalised)

7.3.1.2 Thales case

We solved the case study that we presented in Sections 2.1.1 and 6.5 using our integrated algorithm. It turns out that 50 resources form one resource group; four resources need to be fixated before the remaining graph components are cliques. However, due to the adapters, many small cliques exist of only one resource. Still, it takes about two days to solve the case study, which makes it doubtful whether this method can be used in practice. However, in an optimized commercial implementation, optimization may be fast enough to be used in practice. Another option is to use the iterative method a number of times during the development process, and use the integrated method once at the end to find the solution that is to be implemented.

Figure 7.8 shows the costs that result from using each of the three methods to solve the case study. A cost reduction of 6.7% is achieved by our integrated method compared with the sequential method, which is worth millions over the life time of a sensor system. However we found a cost reduction of 9.7% using the iterative algorithm in less than one and a half minute (see Section 6.5). The reason is the overshoot problem that we discussed before (Sections 7.1 and 7.3.1.1): the iterative method achieves an availability of 95.01%, whereas

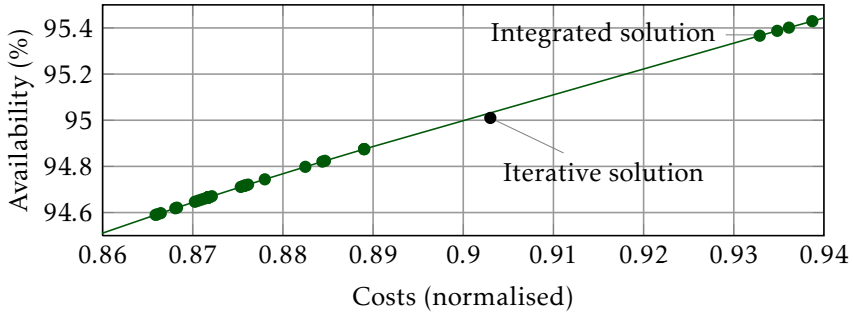


Figure 7.9: Availability as a function of total costs for the case study

the integrated method achieves an availability of 95.37%. Figure 7.9 shows the availability that the integrated model achieves as a function of the total costs (all efficient points). We see that the availability is above 95.01% for the costs of 90.3 (the solution of the iterative method), which means that the iterative solution is indeed below the efficient frontier that is found by the integrated method. 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 would be very low (and the solution would be very close to optimal). As explained in Section 7.3.1.1, the overshoot problem of the integrated method may be reduced in general.

7.3.2 Running time of the algorithm

In this section, we answer the third question that we posed at the beginning of Section 7.3:

3 Which model parameters influence the running time that we achieve?

Tables 7.2, 7.3 and 7.4, used in the previous section, also show the running times for test sets 1, 2 and 3 respectively. Notice that the running time is in seconds for test set 1, and in minutes for the other two sets. The maximum running time over all problem instances is near six hours, in test set 2. Notice however, that a computation time of six hours need not be problematic for usage of our algorithm in practice. However, using the iterative algorithm, the maximum running time is just over ten minutes, see Section 6.4.2.1.

In test set 1, the running time is mainly determined by the number of indenture levels, LRUs and echelon levels. Increasing the number of indenture levels leads to more components in the product structure, and increasing the number of echelon levels increases the number of possible repair/discard options. Therefore, these effects are not surprising. However, an increase in the number of LRUs leads to a decrease in the running time. The reason is that if the number of LRUs increases, the probability of generating problem instances with one large resource group decreases, since the number of resources, and the number

Size of largest resource group	# occurrences 50 LRUs	100 LRUs
1	16	48
2	80	272
3	256	176
4	224	112
5	32	32
6	32	0

Table 7.5: Largest resource groups varying the number of LRUs

of resource-component relations does not change. Finding the EBO-curve for a large resource group takes a lot of computation time. Table 7.5 shows the size of the largest resource group for all problem instances with 50 and 100 LRUs.

In test set 2, the running time increases sharply with an increasing demand: if demand increases, the EBO increases, and as a result, more spares need to be stocked. Due to the fact that we need to calculate more points on the EBO-curves, the running time increases.

In test set 3, the running times are mainly determined by the number of components per resource, and the percentage of components that is used by one component only. An increase in the number of components per resource, or a decrease in the percentage of components that is used by one component only, both lead to more resource-component relations, thus increasing the complexity of the problem. Therefore, it is not surprising that the average running time increases as well.

7.4 Conclusions

In this chapter, we presented an algorithm that can be used to solve the joint problems of LORA and spare parts stocking in an integrated way for multi-indenture, multi-echelon problem instances with very mild restrictions on the resource-component relations. Our method finds a solution that is optimal in the sense that the achieved availability cannot be achieved against lower costs (an efficient point), and except for the approximation errors in the VARI-METRIC methods, which are known to be small for practically relevant problem instances. The reason is that we implicitly enumerate all possible LORA decisions.

We have shown that we can solve real life problems by solving the case study at Thales Nederland. We can solve the case study and achieve a cost reduction of almost 7% compared with solving the LORA and spare part stocking problem sequentially.

We generated problem instances that are realistic in practice and showed that

we can solve problems consisting of upto 700 components. We achieve a cost reduction of 3.40% on average compared with the sequential method, with a maximum of almost 37%. Compared with the iterative method, we achieve a cost reduction of 0.26% on average, with a maximum of almost 5%. This means that we are able to state the quality of our iterative approach, which we could not do in Chapter 6: the iterative method works pretty well on average, but is less robust than the integrated method in the sense that the cost difference compared with the optimum can be significant for specific problem instances. However, if the problem size increases (number of indenture levels, number of LRUs, and number of echelon levels), the maximal cost reduction decreases, which means that the worst case performance of the iterative method improves.

Another advantage of the integrated approach is that we find a set of efficient points at once, so that we gain insight in the relation between the life cycle costs and the operational availability. To generate a similar curve using the iterative method, we have to optimize the model many times for different values of the target operational availability, and this curve need not be smooth and concave. On the other hand, the iterative method is much faster than the integrated method, and model generalizations are easier made, since the iterative method uses two separate building blocks for the LORA and spare parts stocking analysis. Furthermore, the overshoot in the iterative algorithm is generally smaller.

Chapter 8

Conclusions and further research

In this final chapter, we draw conclusions, discuss how the research may be applied in practice, and point out opportunities for further research. In Section 8.1, we answer the research questions that we formulated in Section 1.5 and we conclude to which extent we reached our research objective. Next we discuss how this research may be applied in practice in Section 8.2. In Section 8.3, we point out opportunities for further research; we discuss improvements to our LORA model and to the models for the joint problem of LORA and spare parts stocking.

8.1 Conclusions

The first research question that we posed in Chapter 1 is:

(1) Which methods are available to analyze the LORA and spare parts stocking problem, what is required in practice, and what are therefore the gaps in the literature?

We answered this question in Chapter 2. First, we concluded that none of the existing LORA models fits well on problem instances in practice. Barros (1998), Barros and Riley (2001), and Saranga and Dinesh Kumar (2006) use very restrictive assumptions on the resource-component relations, whereas Brick and Uchoa (2009) assume two indenture levels and one echelon level only. Aggregating data per echelon level, as Barros (1998), Barros and Riley (2001), and Saranga and Dinesh Kumar (2006) do, leads to suboptimal solutions if the network structure is unbalanced, but we did not know how large the gap with

the optimal solution is. We have focused on this question in Chapter 4 and we discuss the results below.

Second, we concluded that given the long tradition in spare parts stocking methods, the current state-of-the-art is sufficient to solve spare parts stocking problems in practice.

Third, we found only one paper in which an approach is presented for the joint problem of LORA and spare parts stocking (Alfredsson, 1997), but it considers one indenture level and two echelon levels only. Besides, the assumptions on the resource-component relations are very restrictive.

Our main conclusion in Chapter 2 was that it would be best to have one method to solve the integrated problem for multi-echelon multi-indenture problems with general restrictions on the resource-component relations. However, since such a general model does not even exist for the LORA problem itself, we stated the second research question:

(2) What is a suitable LORA model that can be solved in a reasonable amount of time for problem instances with a size that is realistic in practice?

To answer this question, we formulated two subquestions. The key problem with the models in the literature is that the assumptions on the sharing of resources between components are too restrictive. Therefore, our first subquestion is:

(2a) In what way can we generalize the models that are available in the literature?

We answered this question in Chapter 3. We developed a LORA model that generalizes the two pure LORA models that existed in the literature (Barros, 1998; Saranga and Dinesh Kumar, 2006). We did this by allowing sets of components that share fixed costs to be arbitrarily defined, instead of assuming that fixed costs are shared between all components at a certain indenture level (Barros) or assuming that fixed costs are due to 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 that the integrality constraints on most decision variables may 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 (2006) are used, so that there is no need to solve problem instances using genetic algorithms, as they do. We also showed that removing all integrality constraints in the model of Barros (1998) may yield a fractional solution, contrary to what the author claimed. Furthermore, the model gives insights into the LORA problem in general. For example, we use the model to show that the LORA problem is NP-hard.

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

However, to be able to model realistic problem instances, the model needs to be extended, for example by allowing for a probability of unsuccessful repair. Therefore, the second subquestion is:

(2b) How can we model the extensions that may be needed in practice?

To model the extensions, we first reformulated the LORA model as a minimum cost flow model with side constraints in Chapter 4. This formulation can be solved much faster than using the basic LORA formulation that we developed in Chapter 3. Furthermore, the model allows us to explicitly model the repair network instead of aggregating all information per echelon level, as is often done in the literature (and in Chapter 3). We have shown that with networks that are unbalanced in the locations, cost reductions of over 7% can be achieved by modelling the exact network. If networks are balanced in the locations, then the maximal costs reductions that can be achieved are only 1.2%, even if the network is unbalanced in the costs.

Next, in Chapter 5, we presented the extensions to the LORA model:

- A probability of unsuccessful repair.
- A no-fault-found probability.
- Capacitated resources.
- Multiple failure modes per component.
- Outsourcing of repair.

We showed how to model the first three extensions; the other two extensions do not require a fundamental change in the model. We tested the former three extensions by generating problem instances in a way similar to how we did that in Chapter 4. We have seen that incorporating a probability of successful repair leads to a cost increase (compared with neglecting this probability) of 95% on average if an unsuccessfully repaired component can only be discarded at the location where the repair was performed; if we allow a second (and third) repair attempt at a higher echelon level, the cost increase is 71% on average. If the probabilities are equal at all echelon levels, the change in repair strategy (compared with not incorporating unsuccessful repair) is small. However, if the probability of unsuccessful repair decreases with an increasing echelon level,

the repair strategy changes significantly: the resource costs at central depot increase, wherease they decrease at the lower echelon levels. Incorporating a no-fault-found probability leads to a reduction of the costs of 4% on average. The change in the repair strategy is small. Incorporating capacitated resources leads to a cost increase of 11% on average. The maximum optimization time increases from 2.3 seconds if no extensions are included to almost half an hour. Since repair is more expensive if resources are capacitated, more components are discarded. Furthermore, resources are located at lower echelon levels. We conclude that although the repair strategy does not always change significantly when incorporating the extensions that we modelled, the total costs do.

After having developed a model that can be used to solve the LORA problem, including all kinds of extensions, we are able to solve the total problem of LORA and spare parts stocking, by performing a LORA first, and then performing a spare parts stocking analysis: the sequential approach. However, when solving both problems sequentially, the LORA solution may lead to an unbalanced overall solution. For example, it may lead to performing many repairs at a central location, which implies that the installed base faces long repair lead times, which increases the amount of spare parts inventory. Therefore, we were not convinced that the sequential approach leads to a good overall solution and we posed research question 3:

(3) What is a suitable method to solve the joint problem of LORA and spare parts stocking?

To answer this question, we formulated two subquestions. The first of which aims for a method that uses existing building blocks:

(3a) How can we iteratively use a LORA model and a spare parts stocking model to solve the joint problem of LORA and spare parts stocking?

We presented, in Chapter 6, an algorithm that uses such an iterative approach to solve the integrated problem of LORA and spare parts stocking analysis for multi-indenture, multi-echelon problem instances with very mild restrictions on the resource-component relations. In fact, we used the LORA model that we developed in Chapter 4 as one building block, and the VARI-METRIC method as the other building block. By adapting the LORA inputs based on the outcome of VARI-METRIC in a number of iterations, we tend to find a better solution in terms of total costs (LORA and spares parts holding costs) than the sequential approach does.

We have shown that we can solve real life problems by solving a case study at Thales Nederland, a manufacturer of naval sensors and naval command and control systems. We can solve the case study in about one and a half minute and achieve a cost reduction of almost 10% compared with performing the LORA and spare part stocking analysis sequentially.

We generated problem instances that are realistic in practice and showed that we can solve problems consisting of up to 700 components in just over ten minutes at maximum. On average, a cost reduction of over 3% is achieved, compared with performing the LORA and inventory stocking analysis sequentially, with a maximum of over 35%.

The major drawback of our approach is that we do not know whether the solutions that we find are close to optimal. This leads to the second and final subquestion:

(3b) Which method can we use to solve the joint problem of LORA and spare parts stocking in a more robust way, leading to a solution that is close to optimal?

We answered this question in Chapter 7, in which we presented an algorithm that can be used to solve the joint problem of LORA and spare parts stocking in an integrated way under the same assumptions as in Chapter 6. Our method finds a solution that is optimal in the sense that the achieved availability cannot be achieved against lower costs (an efficient point), and except for the approximation errors in the VARI-METRIC methods, which are known to be small for practically relevant problem instances. The reason is that we implicitly enumerate all possible LORA decisions.

We have shown that we can solve real life problems by solving the case study at Thales Nederland. However, we achieve a cost reduction of almost 7% compared with solving the LORA and spare part stocking problem sequentially, which is less than the cost reduction that we achieved by using the iterative approach. The reason is the relatively large overshoot that our integrated algorithm often has: the achieved availability is higher than the target availability.

We generated problem instances in the same way as we did in Chapter 6, and we achieve a cost reduction of 3.40% on average compared with the sequential method, with a maximum of almost 37%. Compared with the iterative method, we achieve a cost reduction of 0.26% on average, with a maximum of almost 5%. This means that the iterative method works pretty well on average, but is less robust than the integrated method in the sense that the cost difference compared with the optimum can be significant for specific problem instances.

Our research objective, as stated in Section 1.5, is:

To develop a method that companies can use to analyze the joint problem of level of repair analysis and spare parts stocking for multi-indenture, multi-echelon problem instances.

We developed two such methods, a comparison of which can be found in Table 8.1. The key differences are that the integrated method finds a solution that is optimal in the sense that the achieved availability cannot be achieved

Characteristic	Iterative method	Integrated method
Solution value	Close to solution value of integrated solution, maximum difference is less than 5%	Efficient point, except for the approximation errors in the VARI-METRIC methods, which are known to be small for practically relevant problem instances
Overshoot	Small, 0.07% on average (target = 95%)	Larger, 0.20% on average (target = 95%)
Optimization speed	Fast (up to ten minutes)	Slow (up to six hours in our experiments, about two days for the Thales case)
Generating a complete EBO-curve	Possible by solving the problem a number of time with increasing availability target; the curve is not necessarily concave	The generation of this curve is the main characteristic of this method, so we find this curve at once
Flexibility	High; a more advanced LORA or spare parts stocking building block can easily be included (e.g., to include repair probabilities, or to extend to non-symmetrical LORA decisions).	Medium; extensions influence the complete model. Besides, the problem size and optimization time may easily explode

Table 8.1: Comparison of iterative and integrated method

against lower costs (an efficient point), and except for the approximation errors in the VARI-METRIC methods. However, in general, the overshoot is relatively large and the optimization time is much higher than for the iterative method. Since any additional availability is expensive if the availability is already quite high, overshoot is expensive. However, the iterative method does not guarantee an optimal solution and sometimes finds a solution that is 5% higher than the solution of the integrated method. In practice, we recommend using the iterative method, but if time is no issue, the integrated method should be used.

8.2 Usage in practice

As mentioned in Section 1.1, the research in this thesis is part of the IOP-IPCR project 'life-cycle oriented design of capital goods'. The goal of the project is

to develop a set of quantitative techniques that can be used for an integrated balancing of system availability and life cycle costs (LCC). These techniques are to be used in the development process of capital goods, to gain insights into the impact of design decisions on the LCC and the availability of the product.

The methods that we developed in this thesis may be used to support the development process indeed. They can be used to estimate the upkeep costs for each product design option. Since there are generally only a few product design options in the later stages of the product development process, as discussed in Section 1.1, the upkeep costs of each product design can be compared, possibly together with other characteristics of each product design. Based on this comparison, the best product design can be selected. In that way, a developer can decide, for example, if he should use a more expensive, but more reliable component, or a less expensive, less reliable component.

Another way to use the research in the product design process is as follows. If it turns out that there is a component that is relatively unreliable, the maintenance costs of the complete product that result when using that component may be compared with the maintenance costs that result when using the same component, but with a mean time between failures (MTBF) that is twice as high. The decrease in costs is the maximum amount of money that may be invested to improve the reliability of the component (double the MTBF). This can be done both before the first product is in the field, using reliability data resulting from tests in the factory, and when the product is already installed in several locations, using reliability data that results from actual use.

The research may also be applied in the repair network design process. For example, a company may wonder what cost reduction will be achieved by reducing the lead time between the operating sites and the intermediate depots. If this lead time reduction can be achieved against costs that are lower than the maintenance costs reduction, this is worthwhile. Another example is the question how much costs would reduce if outsourced repairs are performed in half the time it takes at this moment. The third party that performs these repairs, may be interested to speed up the repair process if it gets half the cost reduction (by getting a higher price per repair action).

8.3 Further research

In this thesis, we developed methods to solve the LORA problem and to solve the integrated problem of LORA and spare parts stocking analysis. In Section 8.3.1, we point out opportunities for further research for the former problem; for the latter problem, we do this in Section 8.3.2.

8.3.1 LORA problem

For the LORA problem, we discuss both improvements in the algorithm, in Section 8.3.1.1, and model extensions in Section 8.3.1.2.

8.3.1.1 Algorithm

Due to modelling extensions to the LORA problem, see Chapter 5, we saw an increase in running time, and especially in memory requirements by CPLEX (mainly for the capacitated resources, and, to a lesser extent, the probability of unsuccessful repair). This is due to an increase in the number of decisions to take when incorporating extensions. If the problem size increases (e.g., more components) or multiple extensions are incorporated simultaneously, or the LORA problem is part of an iterative algorithm (as discussed in Chapter 6), then we may require heuristic algorithms to solve the LORA problem.

We propose to develop specific heuristics that use the structure of the problem. For example, a local search heuristic that focuses on the locations of the resources may yield good results. If a resource is located at the central depot in one solution, and locating it at the intermediate depot does not reduce costs, it is unlikely that locating it at the operating sites will reduce the costs. If there are resources that are used mainly in conjunction (one component requires both resources), then locating them at two different locations will probably not be useful. A local search heuristic can easily incorporate such characteristics of the LORA problem.

8.3.1.2 Model extensions

The LORA model in itself is quite complete; extensions are possible, and may be required for specific business situations, but based on discussions with people in industry (e.g., those that participate in the IOP-IPCR project), we believe that we have modelled the most important aspects that we may encounter in practice. However, one relevant extension that we have not discussed in this thesis is the choice between resource types. In our model, we assume that we know which resource we need to repair a component and what the costs of this resource are. In practice, this does not need to be prespecified. For example, we can face the decision whether to buy a rather cheap resource that can be used for the repair of a single component or a more expensive resource that can be used to repair multiple components. Alfredsson (1997) gives a specific example for the choice between resources in his model, where a tester can be upgraded to a more generic tester in several steps. This extension may be included in the model by adapting the constraint that relates the repair option to the resources (Constraint 4.5). Still, this specific extension to the model needs to be tested; the problem size and running time will increase.

8.3.2 Integrated problem of LORA and spare parts stocking

We developed two methods to solve the integrated problem of LORA and spare parts stocking analysis. We believe that for usage in practice, the iterative method is most promising. It is significantly faster and more flexible than the integrated method. Of course, the integrated method guarantees optimality (except for the approximation errors in the VARI-METRIC methods), but for problem instances that are larger than the ones that we used, or if the method is to be extended with, for example, a probability of unsuccessful repair and non-symmetrical LORA decisions, the optimization time and memory requirements during optimization may pose limits for practical use. Using heuristics to solve the problem is not attractive, since that means that the main advantage of the integrated method (optimality) is gone. Still, extending the integrated method is useful, since it provides a benchmark for the iterative method

Therefore, we discuss in Section 8.3.2.1, improvements for both algorithms. For the iterative method, we discuss how the solution quality (relative gap to the optimal solution) can be improved, whereas for the integrated method, we discuss technical improvements that maintain the optimality of the method. Then, in Section 8.3.2.2, we discuss extensions to both methods. A key problem for extensions in the iterative algorithm is the feedback loop. We discuss that in a separate section, Section 8.3.2.3. We end with Section 8.3.2.4, in which we discuss one extension that requires a more extensive change in the algorithms: the inclusion of waiting times for resources (e.g., test equipment and engineers).

8.3.2.1 Algorithm

As announced, we first discuss how to improve the quality of the solution of the iterative method, and then technical improvements to the integrated method.

Iterative method

The variants of the basic method, which we discussed in Section 6.3.3, improved the solution significantly. It may be useful to analyze the characteristics of problem instances for which the variants do not improve the solution compared with the basic iterative method. This may give insights into how better variants can be developed. Since our variants are relatively straightforward, but already improve the solution quite a bit, we believe that this is a promising research opportunity. We are also still convinced that a good starting solution may help, but this requires new lines of thought not explored so far.

Another way to improve the iterative algorithm, is as follows. After finding a solution using the iterative approach, keep only the resource locations, and use the integrated approach to find the optimal solution, except for the approximation errors in the VARI-METRIC methods, given these resource locations. We have seen in Section 6.4.2.2 that in some cases, the resource locations are the same in the sequential and the iterative approach, but still, the iterative

method leads to lower costs by exploiting pooling effects of the spare parts. In Section 7.3.1.1 we found that the integrated approach sometimes improves on the iterative method by simply using these pooling effects to a larger extent. Since solving the integrated problem for a given set of resource locations is not very time-consuming (the approach is time-consuming since there are so many combinations of resource-locations), this post-processing of the iterative solution may be worthwhile.

A third way to improve the iterative solution is by using a number of variants on the algorithm to solve one problem instance, and then choosing the best solution that is found by any of the variants.

Finally, a fourth way to improve the iterative solution is by using a kind of tabu search (see, e.g., Glover, 1989, 1990). If we find a solution that we cannot improve upon anymore, then this solution gets into a tabu list, and the last solution before that solution is restored. The algorithm will now find another solution, thereby possibly avoiding a local minimum. Instead of going back to the last solution that was found before the best solution, we can also go back to a solution that was found earlier. In that case, all later solutions should be in the tabu list. A point of attention is what should exactly be put in the tabu list. The total solution, so all LORA decisions, will not work. Changing one repair/discard decision would then lead to a different solution that is allowed. However, each LORA decision individually may not work either. It may be a good idea to put the location of a couple of important (expensive) resources in the tabu list, thus requiring that a new solution does not put those resources at that location anymore.

Integrated method

For the integrated method, we want to improve the optimization time, and the memory requirements: since many EBO-curves need to be constructed, and each point on these curves corresponds to certain repair/discard decisions, a certain amount and allocation of spare parts, and possibly the location of one or more resources, the integrated approach is both time consuming and memory intensive.

Reducing the optimization time may be possible by storing EBO-curves that are calculated for subcomponents. For example, the EBO-curve that is related to repair of a part at the central depot is constructed a number of times (for the situation that the SRU of which this part is a subcomponent is repaired at the operating sites, at the intermediate depots, and at the central depot)[1] Storing these curves in memory will probably not lead to much additional memory requirements, since all curves for the SRU are constructed one after each other, and then merged. At the moment that these curves are merged, we can remove the curve for the part (subcomponent of the SRU) again.

[1] Except for the move time from the repair location of the parent (e.g., SRU) to the repair location of the child (e.g., part), but this can be added to the EBO-curve after the curve has been constructed.

Reducing the memory requirements can be done by storing only the locations of the resources after constructing the EBO-curves. So, while the EBO-curve for one resource-group is being constructed, we require all repair/discard decisions, and we need to know how many spare parts we located already. However, if we have constructed the curve, it is sufficient to know where resources are located (for each point on the curve). Using that information, the algorithm can be run again to reconstruct the repair/discard decisions and the locations and numbers of the spare parts. The running time of the integrated method is caused by the large amount of possible combinations of resource locations in each resource-group (up to several thousands in our problem instances). Solving the problem for one additional combination of resource locations, leads to a relatively small increase in the optimization time.

8.3.2.2 Model extensions

For most model extensions, it holds that the problem in the iterative algorithm is the feedback mechanism. Therefore, we discuss that in Section 8.3.2.3. In the integrated approach, the problem is mainly that the problem size and running time increase significantly.

In Section 6.6, we discussed extending the iterative method to non-symmetrical LORA decisions. This requires a change in the LORA building block only, which is not a problem, as we have shown in Chapter 4.

In Chapter 5, we presented extensions to the LORA model:

- A probability of unsuccessful repair.
- A no-fault-found probability.
- Capacitated resources.
- Multiple failure modes per component.
- Outsourcing of repair.

Each of these extensions can also be included in the integrated problem. For the probability of unsuccessful repair, it also requires a change in the spare parts stocking analysis. However, this requires that each component has a fixed probability r that it can be repaired at a certain location, which is a standard assumption in the METRIC type models, see Section 6.1.2. Multiple failure modes per component leads to the usage of r as well, in a similar way. The other extensions do not influence the spare parts stocking analysis. The extension to failure modes means that we may take various repair/discard decisions for a certain component depending on the specific failure mode. This, in turn, means that we require a spare parts holding costs estimate for each failure mode, which does not change the feedback loop, but it does increase the problem size. The same holds for the outsourcing of repair option. The no-fault-found probability neither changes the feedback loop nor increases the problem size (except for the problem size in the LORA building block).

We can also make changes in the spare parts stocking analysis, by allowing for additional flexibility. If more than one system is located at each operating site, we may use cannibalization, which means that if a system needs a spare component that is not available, a component is disassembled from another system that is already waiting for a spare part. Two other flexibility options are emergency shipments directly from a location at a higher echelon level (not necessarily the next higher echelon level), which leads to a shorter lead time, but is more expensive, and lateral shipments, which means that if one operating site requires a spare part, it may be supplied from another operating site. Alfredsson and Verrijdt (1999) show that the use of emergency shipments and lateral shipments may lead to significant cost savings. In the iterative algorithm, such flexibility options can be incorporated without any problem, since it influences just one building block (not the feedback mechanism). However, in the integrated algorithm, we can only incorporate these extensions as long as it results in a convex EBO-curve, which effectively means that we rely on the use of a greedy approach (marginal analysis). Cannibalization can be included in a model that can be solved using marginal analysis, see Sherbrooke (2004). However, to the best of our knowledge, this does not hold for the other two flexibility options, which means that they cannot be incorporated in the integrated method.

8.3.2.3 Feedback mechanism

For the extensions that we mentioned in the previous section, the key problem is the feedback mechanism in the iterative algorithm. We mentioned in Section 6.2 that in our current method, there is one approximation only: we decompose the spare parts holding costs into spare parts holding costs per component in the feedback loop in order to get an estimate for the spare parts holding costs that result from taking a repair or discard decion at a certain echelon level for a certain component.

If we extend the model to non-symmetrical LORA decisions, we have to estimate the spare parts holding costs at each location, instead of at each echelon. This increases the number of estimates that we have to make, which means that we make more errors, and which possibly increases the number of iterations that we have to make. It may also affect the convergence of the algorithm. Furthermore, this leads to an additional approximation, since we have to decompose the total spare parts holding costs into spare parts holding costs per location (per component) now. See also Section 6.6.

Inclusion of a probability of unsuccessful repair or capacitated resources means that we have to have a spare parts holding costs estimate for each component that is related not only to the repair/discard decision at a certain location. but also to the location of the second repair attempt (if any), or the number of resources, respectively. Inclusion of these extensions may be quite complicated.

8.3.2.4 Waiting times for resources

As discussed in Section 5.1.3, it would be very interesting to explicitly model the waiting time for resources (both equipment and engineers), using a queueing model. However, in Chapters 6 and 7 we have seen that the combination of LORA and spare parts is already challenging, and the combination of capacitated resources and spare parts is also very challenging (without incorporating the LORA decisions), see, e.g., Rustenburg et al. (2001), Sleptchenko et al. (2002), and Zijm and Avşar (2003). For general multi-indenture, multi-echelon problem instances, such models are not available yet.

However, for smaller problems, for which such models are available (e.g., single-indenture or single-echelon), inclusion of waiting times may be possible in the iterative algorithm. It only affects the spare parts stocking analysis and the feedback loop, as discussed in Section 8.3.2.3. The running time will increase, but it may still be practically useful. In the integrated model, an increase in the running time is more problematic. In principle, inclusion of waiting times is possible, but due to the optimization time, probably very small problem instances only can be solved.

Appendix A

Notation

In this appendix, we summarize the notation that we use in this thesis. First, we define the sets, then the parameters, and finally the decision variables.

Set	Meaning
I	Indenture levels
C	Components
$C_i \subseteq C$	Components at indenture level i (e.g., C_1 consists of the LRUs)
Γ_c	Subcomponents of component c
$E = \{1, \ldots, e^{\text{CEN}}\}$	Echelon levels
L	Locations
$L_e \subseteq L$	Locations at echelon level e (e.g., L_1 consists of the operating sites)
Φ_l	Child locations of location l
D	Decisions (discard, repair, move)
D_e (D_l)	Decisions possible at echelon level e (location l)
G	Set of components sharing fixed costs ($G \subseteq C$, $G \neq \emptyset$)
$\mathcal{G}$	Set of all G
R	Resources
Ω_r	Set of tuples (c, d): resource r is required if for component c decision d is taken
V	Nodes
$V^{\text{s}} \subseteq V$	Source nodes
$V^{\text{d}} \subseteq V$	Decision nodes
$V^{\text{t}} \subseteq V$	Transformation nodes
A	Arcs
$\Theta_{r,l}$	Set of arcs (v, w) that are enabled by locating resource r at location l

Parameter	Meaning
λ_c ($\lambda_{c,l}$)	Annual number of failures in component c over all operating sites (at location l if l represents an operating site; see Section 6.1.3.2 for the explanation if l does not represent an operating site)
e^{CEN}	Highest echelon level (central depot)
$vc_{c,e,d}$ ($vc_{c,l,d}$)	Variable costs if for component c at echelon e (location l) decision d is chosen
$q_{c,b}$	The fraction of failures in component c that is due to a failure in subcomponent $b \in \Gamma_c$
$fc_{G,e,d}$	Annual fixed costs that have to be incurred in order to enable decision d at echelon level e for all components in G
$fc_{r,e}$ ($fc_{r,l}$)	Annual fixed costs if resource r is located at echelon level e (location l)
o_v	Outflow out of source node v
$ac_{v,w}$	Variable costs on arc (v,w)
$p_{v,w}$	Fraction of inflow on node v that flows out on arc (v,w)
$h_{c,r,d}$	Demand (in hours) for resource r per component c for which decision d is taken
u_r	Annual number of hours that resource r can be used (capacity of resource r)
$g_{v,w,r}$	Demand (in hours) for resource r if a flow of one goes through arc (v,w)
hc_c	Annual costs of holding one spare part of component c
$vc^{\text{s}}_{c,e,d,j}$	Annual spares holding costs (estimate) in the LORA inputs after iteration j for component $c$if decision d is taken at echelon level e

Decision variable	Type	Meaning
$X_{c,e,d}$ ($X_{c,l,d}$)	Binary	1, if for component c at echelon level e (location l) decision d is chosen, 0, otherwise
$\mathcal{X}$		Set of all decision variables $X_{c,l,d}$
$Y_{G,e,d}$	Binary	1, if for all components in set G at echelon level e decision d is enabled, 0, otherwise
$Y_{r,e}$ ($Y_{r,l}$)	Binary	1, if resource r is located at echelon level e (location l), 0, otherwise
$F_{v,w}$	Real	Flow on arc (v,w)
$S_{c,l}$	Integer	The number of spares of component c that are located at location l
$\mathcal{S}$		Set of all decision variables $S_{c,l}$

Appendix B

Proof that the LORA problem is NP-hard

We show that the LORA problem is NP-hard in general by reducing the uncapacitated facility location problem (UFL problem) in polynomial time to the LORA problem as presented in Chapter 3 (the LORA model in Chapter 4 generalizes the model in Chapter 3). The UFL problem is NP-hard, see for example Cornuejols et al. (1990), and reduction of an NP-hard problem in polynomial time to another problem shows that the latter problem is NP-hard as well.

The UFL problem may be stated as follows (Cornuejols et al., 1990; Daskin, 1995): there is a set of m clients $I = \{i_1, \ldots, i_m\}$ with a given demand for a single commodity, and a set of n sites $J = \{j_1, \ldots, j_n\}$ where facilities can be located. The fixed costs of opening a facility at site j are f_j, and $d_{i,j}$ are the costs of serving all demand of client i from the facility at site j. The goal is to minimize the costs that have to be made to serve all customer demands. Let $x_j = 1$ if facility j is open and $x_j = 0$ otherwise; $y_{i,j} = 1$ if the demand of client i is satisfied from facility j and $y_{i,j} = 0$ otherwise. The resulting IP formulation is:

$$\text{minimize} \sum_{i \in I} \sum_{j \in J} d_{i,j} \cdot y_{i,j} + \sum_{j \in J} f_j \cdot x_j \tag{B.1}$$

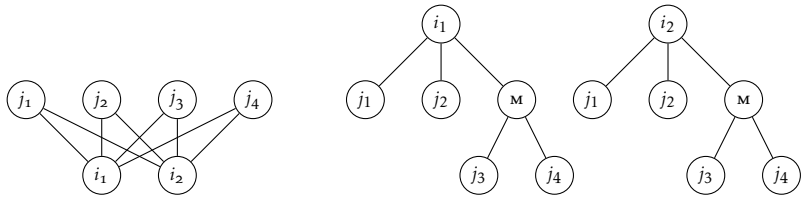
subject to:

$$\sum_{j \in J} y_{i,j} = 1 \ , \forall i \in I \tag{B.2}$$

$$y_{i,j} \leq x_j \ , \forall i \in I \ , \forall j \in J \tag{B.3}$$

$$x_j, y_{i,j} \in \{0, 1\} \ , \forall i \in I \ , \forall j \in J \tag{B.4}$$

Constraint B.2 guarantees that the demand of every client is satisfied; Constraint B.3 guarantees that clients are supplied only from open locations.



(a) Original problem (b) As LORA problem

Figure B.1: The UFL problem as LORA problem

The UFL problem can be modelled as a single-indenture LORA problem consisting of m components (number of clients in the UFL problem). Every facility in the UFL problem is modelled as a possible repair or discard option at one of the echelon levels in the LORA problem. This means that there is an even number of options. If there is an odd number of facilities in the UFL problem, one of the possible repair-discard options should get very high costs associated to it, so that that option will never be chosen. As a result, the repair network consists of $\lceil n/2 \rceil$ echelon levels (number of facilities divided by 2 and rounded up). The costs of supplying client i from facility j, $d_{i,j}$, are equal to the variable costs of repair option j for component i (move has zero costs). The annual demand per component is 1.

Components may only choose repair options that are enabled. This is equivalent to the constraint in the UFL problem that only facilities may be used that are opened. Therefore, we model one set of components sharing fixed costs G_1 consisting of all components. If any of the components uses a certain repair-discard option, fixed costs related to this option are incurred, so the corresponding $Y_{G_1,e,d} = 1$, which costs $fc_{G_1,e,d}$. This is equivalent to opening a facility j in the UFL problem, which costs f_j. All $fc_{G_1,e,\text{move}} = 0$.

We showed in Section 3.2.2 that exactly one repair-discard option j will be chosen in a LORA for failures in any component i, so exactly one facility j is chosen to supply client i in the UFL problem. The reduction as we have shown can be performed in polynomial time.

Figure B.1a shows an example of a UFL problem with two clients and four possible sites where facilities can be located. Figure B.1b shows the corresponding LORA problem. There are two components in which failures occur: i_1 and i_2, representing the two clients that need to be supplied from exactly one location. There are two echelon levels in the repair network, which means that there are four possible repair-discard options for each component (besides the 'intermediate' option M representing move from echelon level 1 to 2): repair at echelon level 1 which represents facility j_1, discard at echelon level 1 representing j_2, repair at echelon level 2 representing j_3, and discard at echelon level 2 representing j_4. The set of components sharing fixed costs $G_1 = \{i_1, i_2\}$.

Appendix C

Experimental design for the basic LORA model

In this appendix, we explain in more detail how we generate the problem instances that are used in Chapter 3 (and in Chapter 4 to compare the flow model with the basic LORA model).

As explained in Section 3.3.1, our problem instance generator receives as inputs the number of components ($|C|$), the number of indenture levels ($|I|$), the number of echelon levels ($|E|$), the number of fixed costs sets ($|\mathcal{G}|$)[1], and the maximum number of fixed costs sets in which each component will be ($s^{\max}$). For each number of fixed costs sets $s \mid 0 \leq s \leq s^{\max}$, a percentage P_s has to be specified, such that $\sum_{s=0}^{s^{\max}} P_s = 100\%$. 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 ($s^{\max} = 1$), 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, we draw a random number to decide in how many fixed costs sets the component will be. We draw that number of sets, with every set having equal probability. The component 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 a. This a should be such that $\sum_{i=1}^{|I|} a^i = |C|$ (or $(a - a^{|I|+1})/(1-a) = |C|$). 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

[1] In our model, we use sets of components that share fixed costs ($G \in \mathcal{G}$). We call these sets *fixed costs sets*.

auxiliary variable a' such that $(a')^{|I|} = |C|$. Then we calculate:

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

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

To determine the number of components at indenture level i ($|C_i|$), we draw a random number from a uniform distribution ranging from $\frac{1}{2}a$ to $1\frac{1}{2}a$ and we multiply this value by the number of components at the next lower indenture level ($|C_{i-1}|$, notice that $|C_0| = 1$, the complete system). We initialize $c^{\text{available}} = |C|$ and for every $i > 0$, we subtract $|C_i|$ from $C^{\text{available}}$. If $c^{\text{available}} < |C_i|$, we set $|C_i| = c^{\text{available}}$. The number of components at the highest indenture level ($i = |I|$) is not drawn, but is equal to $c^{\text{available}}$ after we have drawn the values for all the lower indenture levels. Notice that it can happen that $|C_{|I|}| = 0$, which would mean that the system effectively consists of $|I| - 1$ indenture levels.

For each of the components $b \in C_i$, we draw with an equal probability any one of the components $c \in C_{i-1}$. This c is the parent component of b. 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 $vc_{c,e,d}$, $fc_{G,e,d}$, and λ_c. The actual values for the $vc_{c,e,d}$, $fc_{G,e,d}$, and λ_c 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_c is then changed to $\lambda_c + \sum_{b \in \Gamma_c} \lambda_b$. 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.

Appendix D

Experimental design for the LORA flow model

In this appendix, we describe the random generator that we use to generate the problem instances that we use in in our numerical experiments in Section 4.3.2.1. We explain how we generate product structures, failure rates and cost factors. Generating the repair network structure is explained in Section 4.3.2.1. In that section, we also explain how costs can be unbalanced in the network.

In all tests, the system structure consists of 25 components at the first indenture (subsystems), 125 at the second level and 625 at the third level. Every second and third indenture component is randomly attached to a lower indenture component, so a first indenture component can have zero subcomponents. The demand per subsystem (first indenture) is drawn from a uniform distribution on the interval [0.01, 1]. The percentage of demand for a parent that is due to a failure in a specific child is drawn from a uniform distribution on the interval [0.5/(number of children), 1.25/(number of children)], with a maximum of 1.

For each component, we draw a net price, excluding the costs of the children, from a shifted exponential distribution with shift factor 1,000 and rate parameter 7/(100,000 – 1,000). In this way, on average 0.1% of the components get a value larger than 100,000. We draw a new price for these components to avoid odd problem instances. The reason for our choice is that most systems have a large diversity of items in price, but there are considerably more cheap items than expensive items. The cheapest items (in our case with a price below 1,000) are usually omitted from a regular LORA, because they are discarded by default.

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

- Repair costs as a fraction of the net price are drawn from a uniform distribution on the interval [0.1, 0.4].

- We recursively add the price of each child to its parent to get the gross item price. We do this after calculating the repair price, since repair of a parent means replacement of the child that was defective and taking a decision for the child, thus incurring costs for the child. Discarding or moving a parent does not lead to a decision, and thus costs, for the children.
- The discard costs as a fraction of the gross item price, including children, are drawn from a uniform distribution on the interval [0.75, 1.25]. 100% would be just the costs of replacing a defective component by a new one. However, on the one hand there may be disposal costs, on the other hand, some parts of a defective component can be recycled or re-used.
- The move costs are always 1% of the gross item price.

We also need to take spare part costs into account. If we neglect the spare parts costs, we see that resources are typically bought at the central depot. The explanation is that if a resource is used at the central depot, only one resource needs to be bought. However, multiple resources are needed if it is used at a lower echelon level, since one resource needs to be bought at every location at that echelon. In practice, repairing everything at the central depot means that the availability of the systems goes down, which should be compensated for by buying spare parts.

Integrating spare parts optimization into the LORA is interesting future research, but it does not fit in the scope of this paper. However, there are multiple approaches for adding spare parts costs in a basic way. The approach that we have chosen is based on the difference in lead times for the different options that can be chosen for each component. We assume that repair at the operating site or at the intermediate depot takes a month, whereas repair at the central depot takes three months. Discard (and buying a new item) has a lead time of half a year. Moving a component to a higher echelon leads to an additional lead time of half a month. Using these lead times, we estimate spare parts costs by multiplying the demand for the component, the lead time, a safety factor of 2, the price of the component and holding costs of 30%. A correct item approach would use the standard deviation of the demand instead, but this would make our model non-linear. A correct system approach, such as the METRIC-like approaches (see, for example, Sherbrooke, 2004), would be even more problematic. We tested other approaches and other safety factors, but this approach leads to reasonable results.

The price of each resource is drawn from a shifted exponential distribution with shift factor 10,000 and rate parameter $7/(1{,}000{,}000 - 10{,}000)$ in the same way as described above for the prices of the components. As a result, prices vary between 10,000 and 1,000,000. We randomly assign components to resources, such that the percentage of components that needs 0, 1 and 2 resources are 70%, 20% and 10% respectively in the first case, and 25%, 50% and 25% in the second case.

Appendix E

Experimental design for the joint model

In this appendix, we explain how we generate the problem instances that we use in our experiments in Chapters 6 and 7. The problem instances are divided into three test sets, each with its own focus. We vary in each test set:

1. Sizes of the product structures and network structures, lead times, and holding costs.
2. Annual demands and costs of components and resources.
3. Component-resource relations.

For each parameter that we use to generate these instances, we use the default setting in the text. Some values are set to a certain value, others are drawn from a given range. These drawn values are the same for all settings of the other parameters. Table E.1 (E.2) summarizes for all parameters that do not vary (do vary) their default setting, the alternative setting that we use, and the test set in which that alternative setting is used. We use a full factorial design and we generate 10 problem instances for each combination of parameters to decrease the risk of basing conclusions on one odd problem instance. As a result, test set 1, 2, and 3 consist of 1,280, 160, and 80 problem instances, respectively. Since the default value for each parameter is used in each test set, each test set contains the same ten 'default' problem instances.

Our problem instances are completely symmetrical in the network structure, the cost factors, the demand rates, and the throughput times. The repair network consists of three echelon levels, with a central depot, two intermediate depots, and ten operating sites.

The product structure consists of three indenture levels, with 50 LRUs, and two subcomponents per component on average. All subcomponents are randomly

assigned to one of the components, which means that in general the number of subcomponents per component differs for the various componentss. In our tests, the annual demand for a component is equal to the annual demand of its subcomponents. 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, excluding the costs of its subcomponents, from a shifted exponential distribution with shift factor 1,000 and rate parameter $7/(100{,}000 - 1{,}000)$. In this way, on average 0.1% of the components get a value larger than 100,000. We draw a new price for these components to avoid odd problem instances. The reason for our choice is that most products have a large diversity of components in price, but there are considerably more cheap components than expensive ones. The least expensive components (in our case those with a price below 1,000) are usually omitted from a regular LORA, because they are discarded by default.

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

- Repair costs as a fraction of the net component price are drawn from a uniform distribution on the interval [0.25, 0.75].
- For the discard, move and holding costs, we recursively add the costs of each subcomponent to its parent to get the gross component price for the parent. We do this since, for example, discarding a component should be more expensive than discarding all its subcomponents. However, repairing a component means replacement of the subcomponent that was defective and taking a decision for the subcomponent, thus incurring costs for that subcomponent. Therefore, we should not add the costs of repairing a subcomponent to the costs of repairing its parent.
- The discard costs as a fraction of the gross component price, are drawn from a uniform distribution on the interval [0.75, 1.25]. 100% would be just the costs of replacing a defective component by a new one. However, on the one hand, there may be disposal costs, on the other hand, some parts of a defective component may be recycled or re-used.
- The move costs as a fraction of the gross component price are 1%.
- The annual costs of holding one spare part of a component are 20% of the gross component price.

The discard time, so the time it takes to buy and receive a new component, is drawn from a uniform distribution on the interval [1/10, 1/2]. Both the discard and the repair times vary over the components, but are the same at all echelon levels for each component. The replenishment lead time from one echelon to

Parameter	Default value	Test set	Additional value
# Echelon levels	3	1	2
# Central depots	1	—	—
# Intermediate depots	2	—	—
# Operating sites[1]	10	—	—
# Indenture levels	3	1	2
# LRUs	50	1	100
# Subcomponents per parent component	2	—	—
# Resources	10	—	—
# Component types	4	3	3
% Resources used by 1 component	50%	3	0%

Table E.1: Fixed values

Parameter	Default range	Test set	Additional value(s)
Annual demand of LRU	[0.01; 0.25]	2	[0.01; 0.1] [0.01; 0.5] [0.01; 1]
Net cost of component	[1,000; 10,000]	2	[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]	1	[0.20; 0.40]
Annual cost of resource	[10,000; 100,000]	2	[10,000; 500,000]
Discard time (in years)	[1/10; 1/2]	1	[1/4; 1/2]
Repair time (in years)	[0.5/52; 4/52]	1	[2/52; 4/52]
Move time (in years)	[0.5/52; 4/52]	1	[2/52; 4/52]
# Components per resource	[2; 6]	3	[2; 3]

Table E.2: Values that vary over a range

the next is drawn from a uniform distribution on the interval [2/52, 4/52]. This value varies over the echelon levels, but is the same for all components. For each component, we draw a repair time from a uniform distribution on the interval [2/52, 4/52]. This time does not vary over the echelon levels.

There are ten resources and their annual costs are drawn from a shifted exponential distribution with shift factor 10,000 and rate parameter 7/(100,000 – 10,000). 50% of the resources will be used by one component only, the other 50% will be used by 2 to 6 components. We distinguish 4 'component types', for example electronic versus mechanical components. Each resource and each LRU family is randomly assigned to one of the component types. The result is that resources of one component type do not interact with resources of another component type, which is realistic in practice.

Bibliography

AberdeenGroup (2005). *The Service Parts Management Solution Selection Report. SPM Strategy and Technology Selection Handbook*. Service Chain Management. Featured Research Series.

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

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.

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

Asiedu, Y. and Gu, P. (1998). Product life cycle cost analysis: State of the art review. *International Journal of Production Research*, 36(4):883–908.

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

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

Basten, R. J. I. (2006). *Baselinestudy Thales Nederland*. University of Twente, Enschede (the Netherlands). Confidential report.

Basten, R. J. I., Kutanoglu, E., Van der Heijden, M. C., and Schutten, J. M. J. (2009a). An optimal approach for the joint problem of level of repair analysis and spare parts stocking. BETA working paper 298. Submitted for publication.

Basten, R. J. I., Schutten, J. M. J., and Van der Heijden, M. C. (2009b). An efficient model formulation for level of repair analysis. *Annals of Operations Research*. In press.

Basten, R. J. I., Van der Heijden, M. C., and Schutten, J. M. J. (2008). A minimum cost flow model for level of repair analysis. BETA working paper 254. Submitted for publication.

Basten, R. J. I., Van der Heijden, M. C., and Schutten, J. M. J. (2009c). An iterative method for the simultaneous optimization of repair decisions and spare parts stocks. BETA working paper 295. Submitted for publication.

Blanchard, B. S. (1998). *System Engineering Management*. Prentice Hall, Englewood Cliffs (NJ), second edition.

Blanchard, B. S. and Fabrycky, W. J. (1998). *Systems Engineering and Analysis*. Prentice Hall, Upper Saddle River (NJ), third edition.

Brick, E. S. and 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.

Brown, R. G. (1959). *Statistical Forecasting for Inventory Control*. McGraw-Hill, New York (NY).

Candas, M. F. and Kutanoglu, E. (2007). Benefits of considering inventory in service parts logistics network design problems with time-based service constraints. *IIE Transactions*, 39(2):159–176.

cnet news (2001). California power outages suspended–for now. http://news.cnet.com/2100-1017-251167.html, last checked on 22 September 2009.

Cohen, M. A., Agrawal, N., and Agrawal, V. (2006). Winning in the aftermarket. *Harvard Business Review*, 84(5):129–138.

Cohen, M. A., Zheng, Y.-S., and Agrawal, V. (1997). Service parts logistics: a benchmark analysis. *IIE Transactions*, 29:627–639.

Cornuejols, G., Nemhauser, G. L., and Wolsey, L. A. (1990). The uncapacitated facility location problem. In Mirchandani, P. B. and Francis, R. L., editors, *Discrete location theory*, chapter 3, pages 119–171. John Wiley & Sons, New York (NY).

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

Daskin, M. S., Coullard, C. R., and Shen, Z.-J. M. (2002). An inventory-location model: Formulation, solution algorithm and computational results. *Annals of Operations Research*, 110:83–106.

Deloitte (2006). *The service revolution in global manufacturing industries*.

Diestel, R. (2005). *Graph Theory*. Springer, Heidelberg (Germany), third edition.

Downtime Central (2009). TDC – real examples of downtime cost. http://www.downtimecentral.com/Examples.shtml, last checked on 23 September 2009.

EDCAS (2009). http://www.tfdg.com/edcas.php, last checked on 9 May 2009.

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

Franssen, R. (2006). *Life Cycle Cost Analysis. For Baggage Handling Systems of VanderLande Industries*. Technische Universiteit Eindhoven, Eindhoven (the Netherlands). Confidential report.

Gabor, A. F. and Van Ommeren, J. C. W. (2006). An approximation algorithm for a facility location problem with stochastic demands and inventories. *Operations Research Letters*, 34:257–263.

Gademann, N. and Schutten, M. (2005). Linear-programming-based heuristics for project capacity planning. *IIE Transactions*, 37:153–165.

Gershenson, J. and Ishii, K. (1993). Life-cycle serviceability design. In Kusiak, A., editor, *Concurrent Engineering. Automation, Tools, and Techniques*, chapter 14, pages 363–384. John Wiley, New York.

Glover, F. W. (1989). Tabu search–Part I. ORSA *journal on computing*, 1(3):190–206.

Glover, F. W. (1990). Tabu search–Part II. ORSA *journal on computing*, 2(1):4–32.

Godsil, C. and Royle, G. (2001). *Algebraic Graph Theory*. Springer, New York (NY).

Graham, R. (1972). An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132–133.

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

Guldemond, T. A., Hurink, J. L., Paulus, J. J., and Schutten, J. M. J. (2008). Time-constrained project scheduling. *Journal of Scheduling*, 11(2):137–148.

Gupta, Y. P. (1983). *Life Cycle Cost Models and Associated Uncertainties*, volume F3 of NATO ASI *Series*, pages 535–549. Springer, Berlin (Germany).

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

Hopcroft, J. and Tarjan, R. E. (1973). Efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378.

Jeet, V., Kutanoglu, E., and Partani, A. (2009). Logistics network design with inventory stocking for low-demand parts: Modeling and optimization. *IIE Transactions*, 41(5):389–407.

Kranenburg, B. (2006). *Spare Parts Inventory Control under System Availability Constraints*. PhD thesis, BETA research school, Eindhoven (The Netherlands).

Little, J. D. C. (1961). A proof for the queuing formula: $L = \lambda W$. *Operations Research*, 9(3):383–387.

Melo, M. T., Nickel, S., and Saldanha-da-Gama, F. (2009). Facility location and supply chain management - a review. *European Journal of Operational Research*, 196:401–412.

Meutstege, G. J. H. (2007). *Life cycle cost analysis at PANalytical. Analyzing and modelling the cost components throughout a products life cycle*. Master's thesis, University of Twente, Enschede (the Netherlands).

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

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*. Springer, New York (NY).

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

nu.nl (2009). Saab doet ultieme poging om de gripen te slijten. `http://www.nu.nl/algemeen/1950278/`, last checked on 2 May 2009, in Dutch.

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

Patterson, D. (2002). A simple way to estimate the cost of downtime. `http://roc.cs.berkeley.edu/talks/pdf/LISA.pdf`, last checked on 22 September 2009.

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

Rustenburg, W. D. (2000). *A System Approach to Budget-Constrained Spare Parts*. PhD thesis, BETA research school, Eindhoven (The Netherlands).

Rustenburg, W. D., van Houtum, G.-J. J. A. N., and Zijm, W. H. M. (2001). Spare parts management at complex technology-based organizations: An agenda for research. *International Journal of Production Economics*, 71:177–193.

Rustenburg, W. D., van Houtum, G.-J. J. A. N., and Zijm, W. H. M. (2003). Exact and approximate analysis of multi-echelon, multi-indenture spare parts systems with commonality. In Shanthikumar, J. G., Yao, D. D., and Zijm, W. H. M., editors, *Stochastic Modelling and Optimization of Manufacturing Systems and Supply Chains*, pages 143–176. Kluwer, Boston (MA).

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

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*. Kluwer, Dordrecht (The Netherlands), second edition.

Silver, E. A., Pyke, D. F., and Peterson, R. (1998). *Inventory Management and Production Planning and Scheduling*. John Wiley & Sons, Hoboken (NJ), third edition.

Slay, M. (1984). VARI-METRIC: An approach to modelling multi-echelon resupply when the demand process is poisson with a gamma prior. Technical report, Logistics Management Institute, Washington D.C. Report AF301-3.

Sleptchenko, A., Van der Heijden, M. C., and 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.

Ullman, D. G. (1997). *The Mechanical Design Process*. McGraw-Hill, New York, second edition.

United Kingdom Ministry of Defence (2004a). *Integrated Logistic Support. Part 0: Application of Integrated Logistic Support (ILS) (Issue 6)*.

United Kingdom Ministry of Defence (2004b). *Integrated Logistic Support. Part 1: Logistic Support Analysis (LSA) and Logistic Support Analysis Record (LSAR) (Issue 3)*.

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

United States Department of Defense (1997). *MIL-HDBK-502 Acquisition Logistics*.

VMetric (2009). `http://www.tfdg.com/vmetric.php`, last checked on 7 May 2009.

Vrij Nederland (2009). JSF: weg met dat vliegtuig. `http://www.vn.nl/Binnenland/ArtikelBinnenland/JSFWegMetDatVliegtuig.htm`, last checked on 2 May 2009, in Dutch.

Zijm, W. H. M. and Avşar, Z. M. (2003). Capacitated two-indenture models for repairable item systems. *International Journal of Production Economics*, 81-82:573–588.

Samenvatting

Het onderwerp van dit proefschrift is het inrichten van het onderhoudssysteem voor kapitaalgoederen. Voorbeelden van kapitaalgoederen zijn radarsystemen, vliegtuigen, industriële installaties en MRI-scanners. Kapitaalgoederen zijn kostbaar en technisch complex. Verder zijn de kosten van het niet-beschikbaar zijn van kapitaalgoederen hoog: het niet-functioneren van een radarsysteem op een schip in oorlogsgebied betekent dat het schip kwetsbaar is voor vijandige aanvallen. Het uitvallen van een industriële installatie leidt vaak tot een verminderde productie en inkomstenderving. Een goed ingericht onderhoudssysteem is daarom van groot belang.

Aangezien beschikbaarheid zo belangrijk is, worden kapitaalgoederen in de regel gerepareerd door niet-functionerende componenten te verwisselen, de zogenaamde *repair by replacement*. Deze component kan vervolgens gerepareerd worden, of hij kan worden afgestoten waarna een nieuwe component gekocht wordt. Wanneer een component gerepareerd wordt, gebeurt dit door uitwisseling van een subcomponent of de component kan direct gerepareerd worden. Een eventuele subcomponent kan ook weer afgestoten of gerepareerd worden. We hebben zo een *multi-indenture productstructuur*. Omdat in veel gevallen apparatuur nodig is voor de reparatie van componenten, worden reparaties niet per se op de locatie van het kapitaalgoed uitgevoerd. In veel gevallen bestaat het onderhoudsnetwerk uit meerdere *echelons* (niveaus in het netwerk), waarbij te denken valt aan regionale onderhoudscentra en een centraal onderhoudscentrum. Dit betekent dat ook bepaald moet worden op welk echelon-niveau reparaties uitgevoerd dienen te worden. Een voorwaarde is dat de benodigde apparatuur op die locaties aanwezig is. De *level of repair analysis* (LORA) is een wiskundig optimalisatiemodel dat antwoorden geeft op al deze vragen. Het doel is om de kosten te minimaliseren.

In de wetenschappelijke literatuur bestaan nog nauwelijks goede modellen voor de LORA. Wij hebben twee nieuwe modellen ontwikkeld. Het basismodel, in hoofdstuk 3, lijkt in aanpak vrij sterk op de bestaande modellen, maar is algemener, zodat een bredere klasse van problemen kan worden opgelost. Het verbeterde model, in hoofdstuk 4, is op een compleet nieuwe manier aangepakt (geformuleerd als een *minimum cost flow model with side constraints*). Dit model is weer iets algemener en we laten zien dat deze nieuwe aanpak efficiënt is, dus

weinig rekentijd vraagt. Verder is het model zeer flexibel; allerlei uitbreidingen die in verschillende praktijksituaties relevant zijn, kunnen eenvoudig in dit model worden opgenomen. In hoofdstuk 5 maken we hier gebruik van door zulke uitbreidingen te modelleren. We staan daar bijvoorbeeld toe dat reparatie slechts in een bepaald percentage van de gevallen succesvol is en dat reparatie-apparatuur maar een beperkte capaciteit heeft.

De LORA beantwoordt echter niet alle vragen. Er moeten ook nog reserve-onderdelen in het onderhoudsnetwerk neergelegd worden zodanig dat een bepaalde beschikbaarheid van de kapitaalgoederen gegarandeerd kan worden. De wetenschappelijke literatuur over *spare parts inventories* (voorraden reserve-onderdelen) is zeer uitgebreid. Er bestaan ook verschillende commerciële softwareproducten die gebruikt kunnen worden om te bepalen waar, en in welke hoeveelheden, voorraden neergelegd moeten worden zodanig dat tegen de laagst mogelijke voorraadkosten een bepaalde beschikbaarheid van de kapitaalgoederen bereikt wordt (bijvoorbeeld 95%). De de facto standaardmethode op het gebied van kapitaalgoederen is VARI-METRIC.

In de praktijk worden de LORA en VARI-METRIC *sequentieel* uitgevoerd zoals hierboven beschreven. Dat wil zeggen, eerst wordt bepaald welke componenten bij falen gerepareerd en welke vervangen moeten worden, waar reparaties uitgevoerd moeten worden en waar reparatie-apparatuur geïnstalleerd moet worden. Hierbij worden verschillende kosten meegenomen, zoals transportkosten, werkuren van onderhoudsmonteurs, kosten van nieuw gekochte componenten en afschrijvingskosten van apparatuur. De beschikbaarheid van het kapitaalgoed wordt echter niet meegenomen. Dit gebeurt pas als vervolgens VARI-METRIC wordt toegepast om te bepalen welke investering in reserve-onderdelen nodig is om een bepaalde beschikbaarheid van het kapitaalgoed te realiseren.

In de case-studie die we hebben uitgevoerd bij Thales Nederland, een producent van militaire sensoren (o.a. radarsystemen) en command- en controlsystemen, bedragen de kosten voor reserve-onderdelen meer dan 50% van de totale relevante kosten gedurende de levenscyclus. Omdat de kosten van de reserve-onderdelen niet worden meegenomen in de LORA, worden soms beslissingen genomen die niet kosteneffectief zijn. In de LORA betekent centraal repareren dat dure reparatie-apparatuur slechts op één locatie wordt neergezet, maar de transportkosten relatief hoog zijn. Direct bij de kapitaalgoederen repareren betekent dat de transportkosten nul zijn, maar dat mogelijk dure reparatie-apparatuur op verschillende locaties aangeschaft dient te worden. Aangezien de transportkosten relatief laag zijn ten opzichte van de kosten van reparatie-apparatuur, wordt vaak besloten centraal te repareren. Dit betekent echter wel dat de reparatie-doorlooptijden langer worden, omdat tijd nodig is voor het vervoer van componenten naar de centrale locatie. Dit heeft zijn weerslag op het aantal benodigde reserve-onderdelen: een langere reparatieduur zorgt in beginsel voor meer benodigde reserve-onderdelen omdat meer componenten in het reparatieproces zitten.

In hoofdstuk 6 en 7 ontwikkelen we daarom methoden waarmee het totale

probleem van LORA en voorraden reserve-onderdelen aangepakt kan worden. De *iteratieve methode* in hoofdstuk 6 maakt gebruik van twee bouwblokken: het LORA-model dat we hebben ontwikkeld in hoofdstuk 3 en 4, en VARI-METRIC. Na één iteratie van LORA en VARI-METRIC koppelen we de resultaten van VARI-METRIC terug naar de LORA. Het resultaat is dat de LORA informatie krijgt over de kosten van reserve-onderdelen die samenhangen met de verschillende beslissingen (repareren of afstoten, op verschillende echelon-niveaus). Na een aantal iteraties gaat de LORA daarom een oplossing kiezen die kosteneffectief is (inclusief kosten voor reserve-onderdelen). Deze methode blijkt tot een kostenreductie van gemiddeld meer dan 3% en in een enkel geval meer dan 35% te leiden ten opzichte van de sequentiële methode. Realistische problemen, zoals in de case-studie bij Thales Nederland, blijken opgelost te kunnen worden in minder dan tien minuten. In deze case-studie bereiken we een kostenreductie van meer dan 9%, wat over de levensduur van de twaalf sensorsystemen in de case-studie equivalent is met een besparing van enkele miljoenen euro's.

In hoofdstuk 7 ontwikkelen we een tweede methode die een curve van optimale oplossingen vindt. Deze *geïntegreerde methode* is aanmerkelijk langzamer dan de iteratieve methode, tot zes uur op onze test-instanties en bijna twee dagen voor de Thales-case, maar we weten wel zeker dat we de optimale oplossing vinden. Daarom is deze methode ook zeer geschikt om andere methoden mee te vergelijken, zoals onze iteratieve methode. Uit deze vergelijking blijkt dat de iteratieve methode het erg goed doet. Gemiddeld zijn de kosten van haar oplossingen slechts 0.2% hoger dan die van de oplossingen van de geïntegreerde methode, met een maximum van bijna 5%.

About the author

Rob Basten was Born in Boxmeer, the Netherlands (NL), on 16 November 1981. He completed his pre-university education at the Gymnasium Bernrode in Heeswijk-Dinther (NL) in 1999. In the same year, he started studying Industrial Engineering and Management at the University of Twente in Enschede (NL), with finance as specialism. In 2004, he carried out his final project at Keypoint Consultancy in Enschede and he obtained his ingenieur's degree (Ir., equivalent to Master of Science, MSc) in Industrial Engineering and Management. In addition, he studied Computer Science at the same university from 2000 to 2005, with human media interaction as specialism. Rob carried out his final project at the Deutsches Forschungszentrum für Künstliche Intelligenz in Saarbrücken, Germany, which led to obtaining his ingenieur's degree in Computer Science. In 2005, he started in a PhD project at the University of Twente, School of Management and Governance, Department of Operational Methods for Production and Logistics, under supervision of Henk Zijm, Matthieu van der Heijden, and Marco Schutten. Part of the research has been carried out during a five-month stay at the University of Texas at Austin (United States), in cooperation with Erhan Kutanoglu.