Master’s Thesis

Speeding up Supply Chain Planning at a High-Tech Company

Author:
Joost de Kruijff
0815948

Supervisors:
Cor Hurkens (TUe)
Hidde Gort (ASML)

August 29, 2014
Preface

The research, I present in this thesis has been performed at the Supply Chain Planning department of ASML, as part of my Master’s degree program of Industrial and Applied Mathematics at Eindhoven University of Technology. The project was very interesting for me, since it gave me the opportunity to apply mathematics to a problem from industry. Furthermore, I learned a lot about managing supply chains, solving mixed integer programs and working for a big company.

I am very grateful to everyone involved in this project. I want to thank Cor for all his great ideas and high quality feedback, Hidde for helping me a lot by explaining everything about ASML and the problem and answering all my questions, Fred for his enthusiasm and support and Marcel for his hints and suggestions. Furthermore, I want to thank my parents and my girlfriend for their support and patience during the difficult times.
Abstract

ASML is the world’s leading provider of lithography systems for the semiconductor industry. Their Supply Chain Planning department creates high level release plans which make sure that there are enough resources available to produce the materials and that all components of the materials are available before the materials are released, while minimizing the costs of inventory and backlog. There are two models available to make these plans, of which the Extended model gives the best results. However, solvers need more time than desired to find optimal solutions. Hence the goal of the research described in this thesis is to search for methods to find (near) optimal solution of the Extended model with a substantial lower running time.

Three alternative models are described and tested, of which Alternative model 1 and Simplified model 2 look promising. Both have an exponential number of constraints. For Alternative model 1 some algorithms to solve the model without adding all the constraints are described. These algorithms use a maximum flow problem to find missing constraints. Two of these algorithms are tested and one turns out to be faster than solving the Extended model. For the second model is described how it can be used in a method to find near optimal solutions. Besides this method some simple algorithms to find near optimal solutions are described and tested.
# Contents

1 **Introduction**  
2 **Problem description** 5  
   2.1 ASML’s supply chain 5  
   2.2 Current models 6  
   2.3 Research goal 7  
3 **Current models** 9  
   3.1 The Basic model 9  
   3.2 The Extended model 12  
   3.3 Some numbers 15  
4 **Alternative models** 16  
   4.1 Alternative capacity constraints 16  
      4.1.1 The idea 16  
      4.1.2 The equivalence of the capacity constraints 17  
      4.1.3 Alternative model 1 23  
      4.1.4 Test results for Alternative model 1 24  
   4.2 Alternative inventory constraints 26  
      4.2.1 The idea 26  
      4.2.2 Simplified model 1 27  
      4.2.3 Test results of Simplified model 1 28  
   4.3 Alternative capacity and inventory constraints 30  
      4.3.1 Simplified model 2 30  
      4.3.2 Test results of Simplified model 2 30  
5 **Algorithms using Alternative model 1** 32  
   5.1 Ideas 32  
      5.1.1 First basic algorithm 32  
      5.1.2 LP-relaxation 34  
      5.1.3 Branch and cut 34  
      5.1.4 Callbacks 36  
   5.2 Algorithm 4 36  
   5.3 Algorithm 5 36  
   5.4 Test results 38  
6 **Approximation algorithms** 42  
   6.1 LP relaxation 42  
   6.2 Machine planning 45  
7 **Discussion** 47  
   7.1 Conclusion 47  
   7.2 Further research 47
1 Introduction

ASML is the world’s leading provider of lithography systems for the semiconductor industry. ASML builds complex machines that are critical for the production of integrated circuits or chips. It is a Dutch company and headquartered in Veldhoven. In 2013 ASML shipped 199 systems to customers all over the world. This resulted in a net sales of 5.245 billion euros and a net profit of 1.015 billion euros in 2013. In 2013 ASML had 13225 employees, of which the biggest part is working in Veldhoven. Two of the biggest customers of ASML are Intel and Samsung.

ASML has a large R&D budget and develops the technologies used in their machines. All machines are designed and built by ASML. Each machine is built from a number of modules, which are mostly produced by ASML. Every module consists of smaller parts, which are mostly produced by suppliers. An example of an important supplier is Zeiss, which is a producer of lenses.

Big companies use high level production plans to avoid shortages and inefficient use of resources. Decisions involve a lot of money and many aspects of production have to be taken into account. At ASML these plans are made by the Supply Chain Planning department.

Every month the Central Planning department makes, with help of the Sales and Marketing department, a demand forecast for 1.5 years ahead. After that Supply Chain Planning tries to make a Master Production Schedule such that this demand is satisfied as good as possible. In the Master Production Schedule, Supply Chain Planning plans the start and end date of the construction of every machine. Once a preliminary Master Production Schedule is created the Business Engineering department will check if there is enough capacity, like workforce and tooling, available to really follow this plan. At the same time it is checked if all materials are available at the start of the assembling of the machines. If it is needed, the plan will be adjusted, otherwise the Master Production Schedule will be approved and executed.

To make this process more efficient ASML is developing and implementing a tool at this moment, named Scenario Planning and Optimization Tool (SPOT). This tool is able to create a release plan for all non-negligible materials, given the demand, available capacities, current state of the supply chain and how much time, resources and parts each part, module and machine needs to be created, which minimizes the costs of inventory and backlog. The model used in SPOT is a variation on the SCOP model described in (Spitter, Hurkens, de Kok, Lenstra & Negenman, 2005).

The remainder of this thesis is organized as follows. In the next chapter, a detailed description of the problem is given. In chapter 3 the currently available models, the Basic model and the Extended model, are described and in chapter 4 three alternative models are presented: Alternative model 1, Simplified model 1 and Simplified model 2. In chapter 5 some algorithms to find optimal solutions using Alternative model 1 are given. In the sixth chapter some algorithms to find near optimal solutions are described of which one uses Simplified model 2. Finally, in chapter 7 the results are summarized and recommendations for further research are given.
2  Problem description

In this chapter a complete description of the problem is given, by sketching the supply chain, describing the current model and formulating the research goal.

2.1  ASML's supply chain

ASML produces machines, where every machine is built from a number of modules, which are mostly produced by ASML. Examples of modules are the Metro Frame and the Wafer Stage. Every module consists of smaller parts, which are produced by ASML or by a supplier. The laser and the lens are examples of important and expensive parts. Some materials require multiple components and some materials are components of multiple other materials. Hence, the supply chain of ASML is both convergent and divergent. An illustration of a convergent and divergent supply chain can be found in figure 1. In total a machine consists of approximately 1600 materials, of which approximately 200 have a cost and a production time that are non-negligible.

Before starting the production of a material, all the components of this material must be available. Producing a material takes some time and requires some resources, like workforce and tooling. For each material the production time and, if it is produced by ASML, the amount of resources needed is known. Lenses, lasers and the machines themselves are materials with long lead times. The production of a lens can take for example up to 40 weeks. On average the production of one material uses 1-3 different resources and there are limited amounts of these resources available.

The production of a machine is divided into two parts, named the FSSN and the CSSN. The components of the FSSN are the different modules and the production of a FSSN can take for example 7 weeks. The amount and type of resources required can differ per week, for example during the first two weeks the machine is assembled and 80 work hours of the assemblers is needed in both of these weeks, while during the other 5 weeks the machine is tested and for example 20 work hours of testers is needed in each week. Before CSSN can start its components, one FSSN and some client specific options (materials or software), have to be available. CSSN starts with adding the client specific options to the machine; this takes a negligible amount of
time and resources, so it is not taken into account. After this the machine will be further tested and when this is done the machine will be packed, for which work hours of packers are needed. For example, if the production of the CSSN takes 6 weeks, the packing of the machine could take 2 weeks.

In total ASML produces approximately 15 machine types and every type can be delivered with a different configuration. All types can be produced left orientated or right orientated and in some types the laser and/or lens can be chosen by the customer. This, plus the fact that many clients want some of the approximately 30 client specific options on their machines, is the reason why almost no two machines produced by ASML are exactly the same. A part of the materials needed for a left orientated machine are different from a right orientated machine, although the resources and time needed to make these materials are the same.

Within Sales and Operations Planning, the Supply Chain Planning department makes the Master Production Schedule, which specifies when to start and end the production of machines and thus implicit of all materials, such that there are enough resources available for the production of every material, the needed component materials are available before starting the production and the demand of customers is satisfied in the best way possible. Customer demand is given by quantities, configuration and requested delivery date. The goal is to minimize the costs of inventory, which occur by starting production too early, plus the costs of delivering a machine too late. Both of these costs are based on the value of the material. The costs of late delivery should be at least 100 times as high as the costs of having it as inventory. ASML makes these plans for a 1.5 year horizon ahead, in which ASML would like to produce about 150-300 machines, depending on the demand forecast, which highly fluctuates.

Note that the resource availability is not constant over time. The most important example of this is the lower availability of workforce during holidays. Because of this machines that are produced during holidays are planned with a longer lead time than the other machines.

2.2 Current models

The problem is a variation of the Supply Chain Operation Planning (SCOP) problem described by Spitter et al. (2005). Currently there are two models available, the Basic model and the Extended model, which are both described in the next chapter and by Gort (2013). Both models are variation on the model with balance equations described by Spitter et al. (2005). All the models use planned lead times, so the production time of a material is determined up front. The planning lead times will be measured in weeks. The component materials needed to produce a material should be available before the production starts and the material is ready for usage in another material after the planned lead time. During the production of a material there should be enough resources available. The goal of the models is to minimize the cost of inventory which occur by starting production too early plus the costs of delivering a machine too late.

There are two important differences between the model with balance equations described by Spitter et al. (2005) and the available models. First, the model with
balance equations is a pure linear program, which is sufficient for the planning of high volume products, but for ASML it is useless to consider the production of a fractional number of machines. Hence the current models are mixed integer linear programs. Second, the first model gives the freedom to claim the needed resources anywhere in the given lead time. The Basic model does not give this freedom, while the Extended model limits this freedom, because the tasks have to be done in a specific order.

The amount of time and materials needed for the production of a material and how much of each resource is needed each week is input for the current models. Other input is the current inventory level, the materials that are in production and ordered at a supplier, the amounts of resources available per week and a demand forecast. The output of the current models is the release moments of the materials. The current models guarantee that the required materials are available on time and that enough resources are available in every week of the production process, for every material which starts production at the given release date.

The most important part of the supply chain is the combination of the FSSN and the CSSN because the production of those has the longest lead time (except some materials produced by suppliers) and the value of these materials is the highest. The testing of a machine takes the most weeks of the planned lead time. For the testing specialized testers are needed and during holidays there are fewer testers available. In the Basic model this would mean that the whole testing of a machine should be done before or after the holiday. This causes unnecessary inventory and backlog costs. To avoid these unnecessary costs extended lead times are introduced in the Extended model. These extended lead times give the option to extend the testing period by a few weeks. An illustration of this can be found in figure 2. The Extended model has to choose in which week of testing how many test resources should be used. This amount should not be less than some minimum or more than some maximum per week and the sum over all weeks should be equal to the total required capacity for testing.

### 2.3 Research goal

Gort (2013) showed that using the Extended model instead of the Basic model reduces the total costs (inventory costs plus backlog costs). The downside of the Extended model is that a solver requires a lot more time to find an optimal solution of the Extended model than the optimal solution of the Basic model. The computation times in (Gort, 2013) were significant, even though the used test case was less complex than a real scenario and far less complex than the scenarios that should be solved in the future. Hence the goal of the research presented in this thesis is to search for methods to find (near) optimal solution of the Extended model with a substantial lower running time.
Figure 2: Examples of the production of 3 items, each item is represented by one of the green rectangles. The items normally take 5 weeks to produce and 10 units of some resource is required in each week. In the last two examples there is less of this required resource available during week 6 and 7. Because of this the Basic model releases one item 3 and one item 4 weeks earlier, while the Extended model extends the lead time of each item by one week.
3 Current models

There are currently two models available; both can be found in (Gort, 2013) and are based on the model with balance equations described by Spitter et al. (2005). In this chapter these models are described and some comments are added to them.

3.1 The Basic model

The Basic model does not include extended lead times, hence every material can be made in only one way and if the release date of the material is set the exact amount of each used resource in each time slot is fixed. This is an important difference with the model with balance equations described by Spitter et al. (2005), where it is possible to choose how much from which resource to use in each time slot from a set of resources. Another difference between the two models is that the Basic model is a mixed integer linear program, where the other is a pure linear program.

The base of the Basic model is to keep track of the amount of inventory $I_{lt}$ and backlog $B_{lt}$ for each item over time, which are influenced by the dependent demand $G_{lt}$, independent demand $D_{lt}$ and release orders $R_{lt}$. If an order of size $R_{lt}$ of item $i$ is released at time $t$, it will be in production in the time slots $t + 1$ to $t + \tau_i$, where $\tau_i$ is the lead time of item $i$ and a time slot $t$ is defined as the interval $(t−1, t]$. Before the production starts, the components of item $i$ have to be collected. Hence, if $j$ is an component of $i$, the dependent demand at time $t$ is $G_{jt} = h_{ji}R_{lt}$, where $h_{ji}$ is the number of items $j$ that are needed to produce one item $i$.

During the production, so in time slots $t + 1$ to $t + \tau_i$, some resources are claimed. To be precise in time slot $t + q$, $p_{i qu}$ units of resource group $u$ are claimed for the production of one item $i$. Obviously no more resources can be claimed than there are available per resource group in each time slot. At time $t + \tau_i$, $R_{lt}$ items $i$ are available to fulfill demand or to be put on stock.

![Figure 3: Inventory, $I_{lt}$, of item $i$ over time.](image-url)
The size of the stock of item \(i\) is \(I_{i,t-1}\) until a moment just before time \(t\). At that moment the production is finished and \(R_{i,t-\tau_i}\) is added to stock. Immediately new orders are released and thus the gross requirements \(G_{it}\) will be taken from the stock. The remaining stock can be used to fulfill demand \(D_{it}\) or decrease backlog \(B_{i,t-1}\). After these actions the size of the inventory \(I_{i,t}\) and backlog \(B_{i,t}\) will be calculated. The inventory and backlog are of size \(I_{i,t}\) and \(B_{i,t}\) from time \(t\) until a moment just before \(t + 1\). Having one item \(i\) on inventory or backlog during this period of time costs \(\alpha_{it}\) respectively \(\beta_{it}\).

The Basic model uses the following parameters as input:

- \(n\): \(n \in \mathbb{N}\), the number of different types of items, labeled \(1, ..., n\).
- \(\tau_i\): \(\tau_i \in \mathbb{N}\), \(\tau_i \geq 1\), \(i = 1, ..., n\), the standard planned lead time of item \(i\).
- \(T\): \(T \in \mathbb{N}\), the planning horizon. The total planning interval is \([-\tau_i, T]\), which includes a starting time \(-\tau_i\) and \(T + \tau_i\) time slots \(t\), defined as \((t - 1, t]\), \(t = 1 - \tau_i, ..., T\).
- \(H\): \(H = (h_{ij})_{1 \leq i, j \leq n}\), \(h_{ij} \in \mathbb{N}\), is the matrix representing the bills of material, i.e. \(h_{ij}\) is the number of units of item \(i\) needed for the production of a single unit of item \(j\). It is a natural assumption that the matrix \(H\) is lower triangular and that the diagonal of \(H\) contains only zero elements. This implies that no item can be a component of itself.
- \(\alpha_{it}\): \(\alpha_{it} > 0\), \(i = 1, ..., n\), \(t = 0, ..., T - 1\), the inventory costs of keeping one single unit of item \(i\) on stock at time \(t\) and during time slot \(t + 1\).
- \(\beta_{it}\): \(\beta_{it} > 0\), \(i = 1, ..., n\), \(t = 0, ..., T - 1\), the backorder costs of one single unit of item \(i\) at time \(t\) and during time slot \(t + 1\).
- \(D_{it}\): \(D_{it} \in \mathbb{N}\), \(i = 1, ..., n\), \(t = 0, ..., T - 1\), the independent demand, for item \(i\) in time slot \(t\).
- \(k\): \(k \in \mathbb{N}\), the number of resource groups, labeled \(1, ..., k\).
- \(\mathcal{R}_i\): The set of resource groups that are used by item \(i\), \(i = 1, ..., n\).
- \(\mathcal{I}_u\): The set of items that use resource group \(u\) during their production, \(u = 1, ..., k\).
- \(\rho_{i,u}\): \(\rho_{i,u} \geq 0\), the amount of resources to be allocated to item \(i\) in time slot \(q\) of the planned lead time from resource group \(u\), \(i = 1, ..., n\), \(q = 1, ..., \tau_i\), \(u \in \mathcal{R}_i\).
- \(c_{u,t}\): \(c_{u,t} \in \mathbb{N}\), \(u = 1, ..., k\), \(t = 1, ..., T\), the maximum amount available of resource group \(u\) in time slot \(t\).
- \(I_{i,t-1}\): The inventory level of item \(i\) at time \(-1\), \(i = 1, ..., n\).
- \(B_{i,t-1}\): The backorder level of item \(i\) at time \(-1\), \(i = 1, ..., n\).
- \(\overline{R}_{it}\): The planned work order release of item \(i\), already determined in the past, \(i = 1, ..., n\), \(t = -\tau_i, ..., -1\).
- \(\overline{Z}_{its}\): The part of the planned work, related to work order release \(\overline{R}_{it}\), that will be executed in time slot \(s\) = \(1, ..., t + \tau_i\) on resource group \(u \in \mathcal{R}_i\), \(i = 1, ..., n\), \(t = -\tau_i, ..., -1\).

The decision variables of the Basic model are:
\( I_{it} \): The inventory of item \( i \) during time interval \((t, t+1), i = 1, ..., n, t = 0, ..., T - 1. \)

\( B_{it} \): The backlog of item \( i \) at time \( t \), \( i = 1, ..., n, t = 0, ..., T - 1. \)

\( G_{it} \): The gross requirements (dependent demand) of item \( i \) at time \( t \), \( i = 1, ..., n, t = 0, ..., T - 1. \)

\( R_{it} \): The work order release of item \( i \) at time \( t \), \( i = 1, ..., n, t = -\tau_i, ..., T - 1. \)

\( V_{itu} \): The amount of capacity of resource group \( u \) allocated to item \( i \) in time slot \( t \), \( i = 1, ..., n, t = 1, ..., T, u \in R_i. \)

\( Z_{itsu} \): The part of the planned work, related to work order release \( R_{it} \), executed in time slot \( s \) on resource group \( u \), \( i = 1, ..., n, u \in R_i, t = -\tau_i + 1, ..., T - 1, s = t + 1, ..., t + \tau_i, \) such that \( 1 \leq s \leq T. \)

Note that ASML produces machines, so it does not make sense to produce a non-integer number of machines, hence \( R_{it} \) has to be integer. This yields that \( I_{it}, B_{it} \) and \( G_{it} \) will be integer too.

The mixed integer linear programming formulation of the problem for items \( i = 1, ..., n \) with release moments \( t = 0, ..., T - 1 \) is defined as follows:

\[
\min \sum_{t=0}^{T-1} \sum_{i=1}^{n} \alpha_{it} I_{it} + \sum_{t=0}^{T-1} \sum_{i=1}^{n} \beta_{it} B_{it}
\]

subject to

\[
I_{it} = I_{i,t-1} + R_{i,t-\tau_i} - D_{it} - G_{it} + B_{it} - B_{i,t-1} \quad i = 1, ..., n, t = 0, ..., T - 1 \tag{1}
\]

\[
G_{it} = \sum_{j=1}^{n} h_{ij} R_{jt} \quad i = 1, ..., n, t = 0, ..., T - 1 \tag{2}
\]

\[
B_{it} - B_{i,t-1} \leq D_{it} \quad i = 1, ..., n, t = 0, ..., T - 1 \tag{3}
\]

\[
Z_{itsu} = p_{iju} R_{it} \quad i = 1, ..., n, t = -\tau_i + 1, ..., T - 1, u \in R_i, q = 1, ..., \tau_i, s = t + q, 1 \leq s \leq T \tag{4}
\]

\[
\sum_{s=t-\tau_i}^{t-1} Z_{itsu} = V_{itu} \quad i = 1, ..., n, t = 1, ..., T, u \in R_i \tag{5}
\]

\[
\sum_{u \in R_i} V_{itu} \leq c_{ut} \quad u = 1, ..., k, t = 1, ..., T \tag{6}
\]

\[
R_{it} = R_{i,t-1} \quad i = 1, ..., n, t = -\tau_i, ..., -1 \tag{7}
\]

\[
Z_{itsu} = Z_{itsu} \quad i = 1, ..., n, t = -\tau_i + 1, ..., -1, s = 1, ..., t + \tau_i, u \in R_i \tag{8}
\]

\[
R_{it}, I_{it}, B_{it} \geq 0 \quad i = 1, ..., n, t = 0, ..., T - 1 \tag{9}
\]

\[
V_{itu} \geq 0 \quad i = 1, ..., n, t = 1, ..., T, u \in R_i \tag{10}
\]

\[
Z_{itsu} \geq 0 \quad i = 1, ..., n, t = -\tau_i + 1, ..., T - 1, s = t + 1, ..., t + \tau_i, 1 \leq s \leq T, u \in R_i \tag{11}
\]

The objective minimizes the total inventory and backlog during the planning horizon. The first three equations are called the inventory constraints and make sure that
the inventory and backlog are correct in every time slot \( t \). The next three equations are called the capacity constraints and verify that the capacity used by the releases is not higher than the available capacity for all resource groups \( u \) in every time slot \( t \). The next two equations make sure that already planned work is taken into account.

Note that constraint (2) can be substituted into constraint (1). This would reduce the number of constraints and variables, since constraint (2) and variable \( G_{it} \) are not needed in that case. Another observation is that constraint (4) can be substituted into constraint (5) and constraint (5) into constraint (6). This would reduce the number of constraints and variables significantly, since constraint (4), constraint (5), variable \( Z_{itu} \) and variable \( V_{itu} \) are not needed any longer. In a test case described in section 3.3 these changes halved the number of constraints and non-integer variables. These adjustments did not result in a clear improvement in computation time: In some scenarios the solver was faster using the adjusted model, but in most cases it was not.

### 3.2 The Extended model

The Extended model is based on the Basic model. The difference is the possibility to use extended planned lead times and vary the allocation of resources within the planned lead time interval. In the Extended model some variables and parameters depend on the extra planned lead time \( w \).

At time \( t \) an order of size \( R_{itw} \) is released for item \( i \) with extra planned lead time \( w \). This order will be in production from time slot \( t + 1 \) until \( t + \tau_i + w \). Hence \( R_{itw} \) units of items \( i \) will be available for the next stage of production at time \( t + \tau_i + w \) but not before. The extra planned lead time \( w \), can be chosen between 0 and the maximal extra planned lead time of an item \( i \), \( \lambda_i \), hence item \( i \) can be released with \( \lambda_i + 1 \) different planned lead times. If an item \( i \) is released with extra planned lead time \( w \), the amount of resources of resource group \( u \) claimed per time slot \( q \) can be chosen between \( p_{iquw}^{\min} \) and \( p_{iquw}^{\max} \), which are input parameters of the Extended model. In total a certain amount of resources is needed to produce this item: \( p_{iquw}^{\text{total}} \), which is another new input parameter. Using extra planned lead time costs more than not using it and the goal is to minimize this kind of costs, besides the costs of inventory and backlog. For this purpose the parameter \( \gamma_{itw} \) is introduced.

The Extended model uses the following parameters as input, apart from the parameters used by the Basic model:

- \( \lambda_i \): \( \lambda_i \in \mathbb{N}, i = 1, \ldots, n \), the maximum extra planned lead time on top of the standard planned lead time of item \( i \).
- \( T \): \( T \in \mathbb{N}, \) the planning horizon. The total planning interval is \([−\tau_i − \lambda_i, T]\), which includes a starting time \( −\tau_i − \lambda_i \) and \( T + \tau_i + \lambda_i \) time slots \( t \), defined as \((t − 1, t]\), \( t = 1 − \tau_i − \lambda_i, \ldots, T \).
- \( \gamma_{itw} \): \( \gamma_{itw} > 0, i = 1, \ldots, n, t = 1, \ldots, T, w = 1, \ldots, \lambda_i \), the extra work in progress inventory costs occurring from releasing one single unit of item \( i \) with extra planned lead time \( w \) at time \( t \).
The mixed integer linear programming formulation of the problem for items \( i = 1, ..., n \) with release moments \( t = 0, ..., T - 1 \) is defined as follows:

\[
\min \sum_{t=0}^{T-1} \sum_{i=1}^{n} \alpha_{it} I_{it} + \sum_{t=0}^{T-1} \sum_{i=1}^{n} \beta_{it} B_{it} + \sum_{t=0}^{T-1} \sum_{i=1}^{n} \sum_{w=1}^{\lambda_i} \gamma_{itw} R_{itw}
\]

subject to

\[
I_{it} = I_{it-1} + \sum_{w=0}^{\lambda_i} R_{i,t-\tau_i-w,w} - D_{it} - G_{it} + B_{it} - B_{i,t-1} \quad i = 1, ..., n, t = 0, ..., T - 1 \tag{12}
\]

\[
G_{it} = \sum_{j=1}^{n} \sum_{w=0}^{\lambda_i} h_{ij} R_{itw} \quad i = 1, ..., n, t = 0, ..., T - 1 \tag{13}
\]

\[
B_{it} - B_{i,t-1} \leq D_{it} \quad i = 1, ..., n, t = 0, ..., T - 1 \tag{14}
\]
The objective minimizes the total inventory costs, backlog costs and extra costs because of extended planned lead time during the planning horizon. Similar with the Basic model, the first three equations are the inventory constraints, the next four equations the capacity constraints and the next three equations make sure that already planned work is taken into account. Note that the Basic model is the special case of the Extended model where $A_{G}^{\text{CH}} = 0$ and $q = 0$ for all combinations of $i, u$ and $w$.

Note that constraint (13) can be substituted into constraint (12) and constraint (17) into constraint (18). This would reduce the number of constraints and variables, since constraint (12), constraint (17), variable $V_{itu}$ and variable $G_{it}$ are not needed in that
scenario. In a test case described in section 3.3 these changes reduced the number of constraints and non-integer variables with approximal 1 percent. This adjustments did not affect the computation times. Apparently both variants are equivalent for the solver after preprocessing.

3.3 Some numbers

The test case used in the previous two sections covers the simplified and convergent supply chain of 4 machines, which each contain 6 modules, a FSSN and a CSSN. The case cover 2 years of demands on CSSN’s and thus contains 104 time slots. The case includes 3 resource groups, which are required by the FSSN’s and/or the CSSN’s. In the Extended model the FSSN’s and the CSSN’s have the possibility to extend their lead time with maximal 3 time slots. The number of constraints, (integer) variables and non-zeros the Basic model and the Extended model use to solve this test case can be found in table 1.

<table>
<thead>
<tr>
<th></th>
<th>Basic</th>
<th>Extended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraints</td>
<td>21625</td>
<td>206599</td>
</tr>
<tr>
<td>Variables</td>
<td>24591</td>
<td>110733</td>
</tr>
<tr>
<td>Integer variables</td>
<td>3332</td>
<td>5852</td>
</tr>
<tr>
<td>Non-zeros</td>
<td>67289</td>
<td>380327</td>
</tr>
</tbody>
</table>

Table 1: The number of variables and constraints of the Basic model and the Extended Model.

Note that the Extended model has approximately 10 times as many constraints, 4 times as many variables and 6 times as many non-zeros as the Basic model. In both models the integer variables are the release orders $R_{it(w)}$. In both models there are about 10000 variables and 10000 constraints used to keep track of the inventory $I_{it}$, backlog $B_{it}$ and dependent demand $G_{it}$. Doubling the number of time slots or doubling the number of items will double these numbers. The remaining variables and constraints are used to make sure there is enough capacity. In the Basic model the number of periods $T$ and the total number of resources used per item per time slot $\sum_{i=1}^{n} x_i \tau_i$ (where $x_i$ is the number of resources used during the production of item $i$) are the most important factors in the remaining number of variables and constraints. Doubling one of these two numbers will approximately double the remaining number of variables and constraints. For the Extended model it is little bit more complicated, because $\lambda_i$ in that case also plays a role.
4 Alternative models

In this chapter three alternatives for the Extended model are described. In the first part of this chapter alternative capacity constraints are introduced and in the second part alternative inventory constraints.

4.1 Alternative capacity constraints

As shown in the previous chapter, the biggest part of variables and constraints is used in the capacity constraints. Hence it is logical to look at alternative formulations for the capacity constraints. In the first subsection the idea for the alternative capacity constraints is given and in the second subsection is proven that these alternative capacity constraints are equivalent with the capacity constraints of the Extended model. In the third subsection a model using these alternative capacity constraints is formulated, which is compared to the Extended model. ¹

4.1.1 The idea

In the Extended model variables $Z_{itusw}$ and $V_{itu}$ are used in the capacity constraints to make sure there is enough capacity available to produce all items according to the release plan. They represent the amount of capacity claimed of resource group $u$ in time slot $s$ by release order $R_{itus}$ and the total amount of capacity claimed of resource group $u$ in time slot $s$ by release orders of item $i$, respectively. However it is not necessary to determine these numbers exactly: The goal is to find release orders $R_{itus}$, such that it is possible to produce the materials according to the release orders without claiming more resources than available. Moreover, notice that the Extended model does not use knapsack constraints by construction. The idea for the first alternative model is to use knapsack constraints to formulate the capacity constraints instead of using the variables $Z_{itusw}$ and $V_{itu}$.

Knapsack constraints have the form $d_1 x_1 + \cdots + d_k x_k \leq d_{k+1}$, where the $d_i$’s are constants and the $x_i$’s integer variables. This type of constraints can be used to make the feasible set of the relaxation of the mixed integer linear problem smaller: It is possible to divide the constraint by some number $\delta$ and round all constants $d_i$ down. This results in a new constraint $\lfloor \frac{d_1}{\delta} \rfloor x_1 + \cdots + \lfloor \frac{d_k}{\delta} \rfloor x_k \leq \lfloor \frac{d_{k+1}}{\delta} \rfloor$, which is satisfied by all integer solutions in the feasible set of the relaxation, but the new constraint may cut off some non-integer solutions of the relaxation. Solvers use relaxations to find lower bounds and use this method to find better lower bounds faster. Hence in general solvers will solve problems with knapsack constraints faster.

To verify that there is enough capacity to produce the released materials, it should be possible to assign every release to a sufficient amount of resources in the right time slots. To be more specific: If an item of type $i$ is released at time $t$ with extra lead time $w$, from time slot $t+1$ to $t+\tau_i + w$, for every resource group $u \in R_i$, in total $p_{i,u,w}^{total}$ and in every time slot $s$ at least $p_{i,s-t,u,w}^{min}$ and maximal $p_{i,s-t,u,w}^{max}$ units of resource

¹Background information on the theory used in this section could for example be found in Combinatorial Optimization, Polyhedral and Efficiency, volume A: Paths, Flows, Matchings, by A. Schrijver.
group $u$ should be available for the production of this item. For each resource group $u$ this can be visualized in a bipartite graph, with release orders and time slots as nodes. There are edges between a release order and a time slot if the release order can claim capacity of resource group $u$ in that time slot.

Inspired by Hall’s marriage theorem, the idea of this model is to look at sets of time slots. In a set of time slots $\sigma_u$ the total available capacity of resource group $u$ is $\sum_{t \in \sigma_u} c_{ut}$, which should be bigger than the total amount of resource $u$ claimed by all release orders in $\sigma_u$. Let $\chi_{itw,\sigma_u}$ be the minimal number of units of resource group $u$, claimed in $\sigma_u$ by the release order for one item $i$ with extra lead time $w$ in time slot $t$. Now, the minimal number of units of resource group $u$, claimed in $\sigma_u$ by all releases should be smaller than the total number of units of resource group $u$ claimed by all releases in $\sigma_u$. Hence, to be able to produce the released materials, $\sum_{itw} \chi_{itw,\sigma_u} R_{itw} \leq \sum_{t \in \sigma_u} c_{ut}$ is a necessary condition for each resource group $u$ and each set of time slots $\sigma_u$. In the next subsection is proven that these inequalities form a sufficient condition and thus these inequalities will be the alternative capacity constraints.

$\chi_{itw,\sigma_u}$ can be calculated as $\max(\sum_{q:t \in \sigma_u} p_{iquw}^{\min} R_{itw} - \sum_{q:t \in \sigma_u} p_{iquw}^{\max})$. As an example $\chi_{itw,\sigma_u}$ is calculated for three different release orders, with $\sigma_u = \{1, 2, 3\}$. Figure 4 depicts this example. The first release order releases an item $i$ with lead time 2 at time slot 1. It uses 10 units of resource group $u$ in time slots 2 and 3. Hence $\chi_{i,1,0,\sigma_u} = 20$. The second release order releases the same item $i$ at time slot 1, but with extra lead time 1 and thus lead time 3. It requires in total 20, and in time slots 2, 3 and 4 at least 4 and maximal 10 units of resource group $u$. Thus maximal 10 units of resource group $u$ can be claimed outside $\sigma_u$. Hence $\chi_{i,1,1,\sigma_u} = 20 - 10 = 10$. The third release order releases the same item $i$ with extra lead time 1, but at time slot 2. It requires in total 20, and in time slots 3, 4 and 5 at least 4 and maximal 10 units of resource group $u$. Since it is possible to claim 16 units in time slot 4 and 5, only the required 4 units have to be claimed in time slot 3. This leads that $\chi_{i,2,1,\sigma_u} = 4$.

### 4.1.2 The equivalence of the capacity constraints

Before it can be proven that the alternative capacity constraints are equivalent with the capacity constraints of the Extended model, a method to check if there is enough capacity to produce all materials according to given release orders $R_{itw}$, is given. For each resource group $u$ the graph described in the next paragraph and shown in figure 5, should be constructed. The maximum flows in these graphs are of size $\sum_{itw} R_{itw} p_{itw}^{\text{total}}$ for all resource groups $u$ if and only if there is enough capacity to produce all materials according to given release orders $R_{itw}$. This is proven in the three paragraphs after the description of the graphs.

Besides a source and a sink, there are two horizontal lines of nodes. The first line contains a node for every tuple $(i, t, w)$ for which $R_{itw} > 0$ and $u \in \mathcal{H}_i$. The second horizontal line of nodes contains a node for every time slot $s$ that could be used for the production of one of the released materials. There is an edge from the source to every node $(i, t, w)$ with capacity $R_{itw} p_{itw}^{\text{total}}$. Furthermore there is an edge from a node $(i, t, w)$ to a node $s$ if the production order of item $i$ that is released at time $t$ with
Figure 4: Examples of the calculation of $\chi_{i,t,w,u}$, with $\sigma_u = \{1, 2, 3\}$ and both $p_{i,quw}^{\text{min}}$ and $p_{i,quw}^{\text{max}}$ are the same in each time slot $q$. 
extra lead time $w$ could use the resource capacity of resource group $u$ at time slot $s$. This edge has capacity $R_{itw}p_{i,s-t,u,w}^{\text{max}}$ and a minimum flow requirement of $R_{itw}p_{i,s-t,u,w}^{\text{min}}$. Finally there are edges from the nodes $s$ to the sink, which have capacity $c_{us}$.

Consider the flow for which the flow on the edges from $(i, t, w)$ to $s$ is of size $R_{itw}p_{i,s-t,u,w}^{\text{min}}$. This is a feasible flow, unless $\sum_{i,t, w} R_{itw}p_{i,s-t,u,w}^{\text{min}} > R_{itw}p_{i,u,w}^{\text{total}}$ for some release $(i, t, w)$ and resource $u$ or $\sum_{i,t, w} R_{itw}p_{i,s-t,u,w}^{\text{min}} > c_{us}$ for some time slot $s$ and resource $u$. Note that the first of these two scenarios can not happen since $\sum q p_{i,t,u,w}^{\text{min}} \leq p_{i,u,w}^{\text{total}}$ by the definition of these parameters. If the second scenario appears, it is clear that there is not enough capacity of resource group $u$ in time slot $s$ to produce the materials according to the release orders $R_{itw}$. Note that in this case the constraint $\sum_{i,t, w} \chi_{itwuo_e} R_{itw} \leq \sum_{t e_o c u t} c_{ut}$ is violated for $s = \{s\}$.

Assume that the maximum flow is of size $\sum_{i,t, w} R_{itw}p_{i,u,w}^{\text{total}}$. Note that the maximum flow in this graph can not be bigger than this, since this is the maximum amount of flow that could leave the source. Hence, the flow over an edge from the source to a release $(i, t, w)$ is of size $R_{itw}p_{i,u,w}^{\text{total}}$. Let $Z_{itzuw}$ be the amount of flow from $(i, t, w)$ to $s$ and let $V_{it} = \sum_{w=0}^{A_i} (1 - \sum_{s=t-w}^{t-1} Z_{itzuw})$. The given release orders $R_{itw}$ with these $Z_{itzuw}$’s and $V_{it}$’s satisfy the capacity constraints of the Extended model, because constraint (15) is satisfied by the capacity constraints on the edges from $(i, t, w)$ to $s$, constraint (16) by the flow conservation constraint on nodes $(i, t, w)$, constraint (17) by definition and (18) by the combination of the capacity constraint on the edges from $s$ to the sink and the flow conservation constraint on node $s$. Hence, there is enough capacity of resource group $u$ to produce the materials according to the release orders $R_{itw}$.

On the other hand, if it is possible to produce the materials according to the release orders $R_{itw}$ without claiming more units of the different resources than available, the $Z_{itzuw}$’s exist and a maximum flow of size $\sum_{i,t, w} R_{itw}p_{i,u,w}^{\text{total}}$ can be reached by putting $R_{itw}p_{i,u,w}^{\text{total}}$ flow on the edges from source to nodes $(i, t, w)$, $Z_{itzuw}$ flow on the edges

![Figure 5: If the maximum flow in this graph is of size $\sum_{i,t, w} R_{itw}p_{i,u,w}^{\text{total}}$, there is enough capacity of resource group $u$ to produce all materials according to given release orders $R_{itw}$.](image)
Figure 6: The maximum flow graph of an example, which considers the production of 5 items $i$ released in 5 adjacent time slots. The lead time of the item $i$ is 3 and in total it requires 9 units of some resource group $u$. In each time slot during the production of one item $i$ at least 2 and maximal 4 units of resource group $u$ should be available for the production of this item. The capacity of resource group $u$ is 7 in every time slot.

from the nodes $(i, t, w)$ to the nodes $s$ and $\sum_{itw} Z_{itsuw}$ flow on the edges from the nodes $s$ to the sink.

In the rest of this subsection, it is proven that the alternative capacity constraints form a sufficient condition to guarantee that there is enough capacity to produce all materials. Consider the case that release orders $R_{itw}$ are such that there does not exist an appropriate assignment of resources. By the reasoning described in the first part of this subsection, this means that, for some resource group $u$, there does not exist a (maximum) flow of value $\sum_{itw} R_{itw}^{total}$ on the corresponding maximum flow graph. As it is noted before, if there exists no feasible flow, there is a time slot $s$ for which the constraint $\sum_{itw} X_{itw} \chi_{itw} a_{u} R_{itw} \leq \sum_{i \in \sigma_u} c_{it}$ is violated for $\sigma_u = \{s\}$. Assume that there is a feasible flow and consider the residual graph of the maximum flow on this graph. The residual graph has the same nodes as the original graph and an edge from node $a$ to node $b$ if the flow $F_{ab}$ is smaller than the capacity $C_{ab}$ on this edge in the original graph, or if the flow $F_{ba}$ is bigger than the minimal required flow $M_{ba}$ on the edge from $b$ to $a$ in the original graph. The capacity on such an edge is $C_{ab} - F_{ab}$ or $F_{ba} - M_{ba}$, respectively. Figures 6, 7 and 8 form an example of a maximum flow graph, a maximum flow and its residual graph. In the rest of this subsection it is shown that the constraint $\sum_{itw} X_{itw} \chi_{itw} a_{u} R_{itw} \leq \sum_{i \in \sigma_u} c_{it}$, for $\sigma_u$ the set of time slots that can be reached from the source in the residual graph, is violated by the release orders $R_{itw}$.

Since it is not possible to produce all materials, the maximum flow is not of size
\[ \sum_{itw} R_{itw} p_{tw}^{\text{total}} \] in the earlier described graph for some resource group \( u \). Hence the flow on the edges from the source to the nodes \( (i, t, w) \) is not for all edges \( R_{itw} p_{tw}^{\text{total}} \) and thus some nodes \( (i, t, w) \) can be reached from the source in the residual graph. Since \( \sum_s R_{itw} p_{s-t,u,w}^{\text{max}} \geq R_{itw} p_{tw}^{\text{total}} \), from each of these nodes \( (i, t, w) \) at least one node \( s \) can be reached in the residual graph. From the set of these \( s \) it is not possible to reach the sink, since this would contradict the fact that the flow is a maximum flow. If the flow from a node \( (i, t, w) \) that was not found before, to one of these \( s \) is bigger than \( R_{itw} p_{s-t,u,w}^{\text{min}} \), the nodes \( (i, t, w) \) can be reached from this \( s \) in the residual graph. If the flow from these new nodes \( (i, t, w) \) to nodes \( s \) that are not found yet, is smaller than \( R_{itw} p_{s-t,u,w}^{\text{max}} \), these nodes \( s \) can be reached from these nodes \( (i, t, w) \). These steps can be repeated and this way all reachable nodes can be found.

Note that partitioning the node set of this graph into the nodes which can be reached from the source in the residual graph and the nodes which can not be reached, is a source-sink cut. It is a fact that the size of the flow is equal to the amount of flow over the edges that cross the cut from the source side to the sink side, minus the amount of flow over the edges that cross the cut from the sink side to the source side. In figure 9 this calculation is illustrated. Denote the set of nodes \( s \) that can be reached from the source in the residual graph by \( \sigma_u \) and the set of nodes \( (i, t, w) \) that can be reached from the source in the residual graph by \( \rho_u \).

The edges from the source to the nodes \( (i, t, w) \notin \rho_u \) are the only edges that include the source and cross the cut. The flow on these edges is of size \( R_{itw} p_{tw}^{\text{total}} \). The edges from \( (i, t, w) \in \rho_u \) to \( s \notin \sigma_u \) are the only edges that include \( (i, t, w) \in \rho_u \) and cross the cut. The flow on these edges is of size \( R_{itw} p_{s-t,u,w}^{\text{max}} \). The edges from \( s \in \sigma_u \) to the sink are the only edges starting in \( s \in \sigma_u \) and cross the cut. The flow on these edges is of size \( c_{cut} \). The edges from \( (i, t, w) \notin \rho_u \) to \( s \in \sigma_u \) are the only edges that include
Figure 8: The residual graph of the maximum flow shown in figure 7. The blue nodes are the nodes which can be reached from the source. Note that the set of nodes is partitioned in blue and black nodes.

\((i, t, w) \notin \rho_u\) and cross the cut. The flow on these edges is of size \(R_{itw}p_{i,s-t,u,w}^{\min}\). Note that these last edges cross the cut from the sink side to the source side. Hence, in total the flow is of size

\[
\sum_{(i,t,w) \notin \rho_u} R_{itw}p_{iuw}^{\text{total}} + \sum_{(i,t,w) \in \rho_u} \sum_{s \in \sigma_u} R_{itw}p_{i,s-t,u,w}^{\max} + \sum_{s \in \sigma_u} c_{ut} - \sum_{(i,t,w) \notin \rho_u} \sum_{s \in \sigma_u} R_{itw}p_{i,s-t,u,w}^{\min}.
\]

Since the maximum flow is smaller than \(\sum_{itw} R_{itw}p_{iuw}^{\text{total}}\) the next equation holds

\[
\sum_{s \in \sigma_u} c_{ut} < \sum_{(i,t,w) \notin \rho_u} R_{itw}(p_{iuw}^{\text{total}} - \sum_{s \in \sigma_u} p_{i,s-t,u,w}^{\max}) + \sum_{(i,t,w) \notin \rho_u} \sum_{s \in \sigma_u} R_{itw}p_{i,s-t,u,w}^{\min}.
\]

Note that the right side of this equation is not bigger than \(\sum_{itw} R_{itw}x_{itwua_u}\) and thus these release orders indeed violate the constraint \(\sum_{itw} R_{itw}x_{itwua_u} \leq \sum_{s \in \sigma_u} c_{us}\). This proves that the alternative capacity constraints form a sufficient condition to guarantee that there is enough capacity to produce all materials.
Figure 9: The maximum flow shown in figure 7. The graph is partitioned into the nodes that can be reached in the residual graph (blue) and the nodes which cannot be reached (black). This partition is a source-sink cut. The size of the flow is equal to the amount of flow over the edges that cross the cut from the source side to the sink side (green), minus the amount of flow over the edges that cross the cut from the sink side to the source side (red).

4.1.3 Alternative model 1

The capacity constraints described in the previous subsection are called sets-based and will be used together with the inventory constraints of the Extended model to create Alternative model 1. Alternative model 1 uses the following parameters as input, apart from the parameters used by the Extended model:

- $\sigma_u$: A set of time slots in $[1, T]$, used to make sure that there is enough capacity of resource group $u, u = 1, \ldots, k$.
- $\Sigma_u$: A set of sets $\sigma_u$ of time slots in $[1, T]$, used to make sure that there is enough capacity of resource group $u, u = 1, \ldots, k$.
- $\chi_{itwu_\sigma}$: The minimal amount of resources of resource group $u$, claimed in $\sigma_u$ by the release of one item $i$ with extra lead time $w$ at time $t$, $i = 1, \ldots, n$, $t = 0, \ldots, T - 1$, $w = 0, \ldots, \lambda_i$, $u = 1, \ldots, k$, $\sigma_u \in \Sigma_u$.

Note that $\chi_{itwu_\sigma}$ can be calculated as $\max(\sum_{q:t=t} \min p_{i_{quw}}^\text{min}, p_{i_{uw}}^\text{total} - \sum_{q:t=t} \max p_{i_{quw}}^\text{max} )$.

Now the mixed integer linear programming formulation of Alternative model 1 for items $i = 1, \ldots, n$ with release moments $t = 0, \ldots, T - 1$ is defined as follows:

$$\min \sum_{t=0}^{T-1} \sum_{i=1}^{n} \alpha_{it} I_{it} + \sum_{t=0}^{T-1} \sum_{i=1}^{n} \beta_{it} B_{it} + \sum_{t=0}^{T-1} \sum_{i=1}^{n} \sum_{w=1}^{\lambda_i} \gamma_{itw} R_{itw}$$
subject to

\[ I_{it} = I_{it-1} + \sum_{w=0}^{\lambda_i} R_{i,t-\tau_i-w,w} - D_{it} - G_{it} + B_{it} - B_{i,t-1} \quad i = 1, \ldots, n, \quad t = 0, \ldots, T - 1 \quad (25) \]

\[ G_{it} = \sum_{j=1}^{n} \sum_{w=0}^{\lambda_j} h_j R_{jt} \quad i = 1, \ldots, n, \quad t = 0, \ldots, T - 1 \quad (26) \]

\[ B_{it} - B_{i,t-1} \leq D_{it} \quad i = 1, \ldots, n, \quad t = 0, \ldots, T - 1 \quad (27) \]

\[ \sum_{i \in I_u} \sum_{w=0}^{\lambda_i} \sum_{t : [t+1, t+\tau_i+w] \cap \alpha_u \neq \emptyset} R_{itw} \chi_{itw} \omega_u \leq \sum_{t \in \alpha_u} c_{ut} \quad \forall \sigma_u \in \Sigma_u, \quad u = 1, \ldots, k \quad (28) \]

\[ R_{itw} = \bar{R}_{itw} \quad i = 1, \ldots, n, \quad t = -\tau_i - w, \ldots, -1, \quad w = 0, \ldots, \lambda_i \quad (29) \]

\[ I_{it}, B_{it} \geq 0 \quad i = 1, \ldots, n, \quad t = 0, \ldots, T - 1 \quad (30) \]

\[ R_{itw} \geq 0 \quad i = 1, \ldots, n, \quad t = 0, \ldots, T - 1, \quad w = 0, \ldots, \lambda_i \quad (31) \]

Using all possible subsets of \([1, T]\) would make the number of constraints in Alternative model 1 exponential in \(T\). This is a problem, since ASML makes plans with at least 78 time slots the amount of constraints will be impossible to handle for the solver. Hence not all possible sets are used. However, if only a part of the possible sets is used it is not guaranteed that there are enough resources available. Hence the challenge is to decide which sets are needed to make sure that there are enough resources available and an feasible optimal solution is found.

### 4.1.4 Test results for Alternative model 1

To be able to compare Alternative model 1 with the Extended model, both were implemented, using AIMMS 3.13. For all tests in this thesis a HP EliteBook 8560w laptop with an Intel Core i7-2670QM 2.2GHz processor and 8 GB RAM was used. As a solver CPLEX 12.5 was used, but in most cases a similar test case was solved using GUROBI 5.5, which gave comparable results.

The test case is a simplified version of a realistic scenario. It has a convergent supply chain and contains 72 time slots, 32 items, of which 4 end items, and 1 resource group. This resource group is used by 8 different items, which are also the items which can be released with extended lead times. If these items are released without extra lead time, \( p_{i \in q \in w}^{\min} = p_{i \in q \in w}^{\max} \) for all \( i \) and \( q \), while \( w = 0 \) and \( u \) is the only resource. And if these items are released with extra lead time, then \( p_{i \in q \in w}^{\min} = 0 \). The cost parameters are constant over time, based on the values of the items and \( \alpha_{it} = \gamma_{it} \) and \( \beta_{it} = 10 \alpha_{it} \) for all items \( i \) and time slots \( t \). Different resource capacity scenarios are used to get multiple results.

The capacity is constant over all time slots except during the summer holiday, where the capacity is lower. Both capacities are varied. The \( a, b \) and \( c \) always correspond to a summer capacity of \( \frac{2}{3}, \frac{1}{2} \) and \( \frac{1}{3} \), respectively, times the normal capacity.
Table 2: The number of variables and constraints of the Extended model and Alternative model 1 in the given test case.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Extended</th>
<th>Alternative 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraints</td>
<td>72203</td>
<td>7014</td>
</tr>
<tr>
<td>Variables</td>
<td>42209</td>
<td>10913</td>
</tr>
<tr>
<td>Integer variables</td>
<td>4000</td>
<td>4000</td>
</tr>
<tr>
<td>Non zeros</td>
<td>146049</td>
<td>48007</td>
</tr>
</tbody>
</table>

Table 3: Comparison of the running time of the Extended model and Alternative model 1.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Extended</th>
<th>Alternative 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solving time (seconds)</td>
<td>Gap</td>
<td>Nodes evaluated</td>
</tr>
<tr>
<td>1.1a</td>
<td>1</td>
<td>0.00%</td>
</tr>
<tr>
<td>1.1b</td>
<td>24</td>
<td>0.00%</td>
</tr>
<tr>
<td>1.1c</td>
<td>900</td>
<td>0.42%</td>
</tr>
<tr>
<td>1.2a</td>
<td>1</td>
<td>0.00%</td>
</tr>
<tr>
<td>1.2b</td>
<td>900</td>
<td>1.01%</td>
</tr>
<tr>
<td>1.2c</td>
<td>900</td>
<td>0.23%</td>
</tr>
<tr>
<td>1.3a</td>
<td>1</td>
<td>0.00%</td>
</tr>
<tr>
<td>1.3b</td>
<td>765</td>
<td>0.00%</td>
</tr>
<tr>
<td>1.3c</td>
<td>679</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

The normal capacity is for the scenarios 1.1a, 1.1b and 1.1c the same, however the summer capacity is, respectively, $\frac{5}{3}$, $\frac{1}{2}$ and $\frac{1}{3}$ of the normal capacity for these scenarios. The normal capacity for the 1.3 scenarios is lower than the normal capacity for the 1.2 scenarios, which is lower than this capacity for the 1.1 scenarios.

Since it is exponentially big, Alternative model 1 was not used fully. Besides the sets that contain only one time slot, 30 more sets were added. These sets contained mostly time slots during and just before the summer holiday, because these are the only time slots where the resource capacity is limiting in these scenarios. In table 2 the number of variables and constraints of both models can be found. Note that the number of variables and constraints is significantly smaller in Alternative model 1.

The results can be found in table 3. Note that when Alternative model 1 was used the problem was solved faster in all scenarios, except in scenario 1.1c. In that scenario the solver did not solve the problem within 900 seconds using either of the two models, but with Alternative model 1 the gap between the lower bound and the best solution that was found by the solver, was smaller. Note that for scenarios with less capacity available Alternative model 1 found infeasible solutions. For these scenarios more sets should be added.

These results suggest that Alternative model 1 with the right sets finds optimal solutions significantly faster. However, the challenge is to decide which sets to take into consideration. With too many sets the model gets too big and the solver takes more time to find the optimal solution. But if not enough sets are used the model can return infeasible solutions. In chapter 5 two algorithms to find the needed sets
and the optimal solution are described. Note that the total costs Alternative model 1 returns always is a lower bound on the total costs of the optimal solution.

4.2 Alternative inventory constraints

The Extended model keeps track of the inventory and backlog of every item. However, this information is not explicitly needed. Hence it is natural to look at alternative formulations of the inventory constraints. In this section the idea and the formulation of a model with alternative inventory constraints are described and compared with the Extended model.

4.2.1 The idea

In the Extended model the inventory and backlog of every item is used to calculate the total costs and to guarantee that all components of an item are available when the production of this item starts. For ASML, the most challenging part of the planning is the planning of the CSSN’s and FSSN’s, because these are the items that can be produced with extended lead times. Recall that the CSSN’s are the end items and a FSSN is the most important component of the CSSN. This implies that the CSSN’s and FSSN’s are the most valuable items. Hence, the planning of these items is also the most important part of the whole planning.

Since one FSSN is used to create one CSSN and they are both built in the same cabin and directly after each other, it is reasonable to model the FSSN and CSSN together as one end item. Note that the other components of the CSSN, the client specific options, are ignored for the moment. The idea is to first plan these end items and after that the rest of the supply chain. It could happen that some parts in the supply chain can not be in time to produce the end items as planned. In this case it should be possible to add a constraint to the model to make sure these end items are planned later and solve the model again to get a new planning for the end items. By repeating this, an iterative process is defined and this process should give a good feasible solution in the end. Note that this process does not guarantee to find the optimal solution.

The main focus of this section is the planning of the end items and thus the iterative process is not considered here. The goal of this planning is to plan the release dates of the end items such that the resource capacity constraints are met and the costs of inventory, backlog and extra work in process are minimized. Inspired by the assignment problem one could say that every end item that is demanded by some customer, should be assigned to a time slot, in which it is released.

To model this idea, every end item, of some type \(i\), that should be released, gets an unique number \(y\). For each end item \((y, i)\) that should be produced, a binary variable \(X_{ytwi}\) is introduced, where \(X_{ytwi} = 1\) if this end item is released at time \(t\) with extra lead time \(w\) and \(X_{ytwi} = 0\) otherwise. Obviously every machine should be released exactly once. Let \(d_{yi}\) be the time slot in which machine number \(y\) of type \(i\) is demanded. Using this demand time \(d_{yi}\), the lead time \(\tau_i\) and the extra lead time \(w\), it is possible
to calculate the number of time slots machine \((y, i)\) would be held in the inventory or delivered too late and thus the costs of choosing \(X_{ytwi} = 1\).

This construction replaces the inventory constraints, but it requires a slightly different formulation of the capacity constraints. In the next subsection the capacity constraints of the Extended model are altered such that they can be used to form a model with this alternative for the inventory constraints. In section 4.3 the same is done for the capacity constraints of Alternative model 1. This way two models are created and since they only consider end items, they will be called Simplified model 1 and Simplified model 2 respectively.

**4.2.2 Simplified model 1**

Simplified model 1 uses the variable \(Z_{yiyu},\) which represents how many units of resource group \(u\) are claimed by end item \((y, i)\) during time slot \(t\). The possible values of \(Z_{yiyu}\) depend on the values of the \(X_{ytwi}\)’s, \(p_{i}^{\min}\) and \(p_{i}^{\max}\). Like in the Extended model, the sum of the \(Z_{yiyu}\)’s over \(t\) is equal to \(p_{i}^{\text{total}}\) and the sum of the \(Z_{yiyu}\)’s over all end items \((y, i)\) should be smaller than the available capacity \(c_{iu}\). Simplified model 1 uses the following parameters as input, apart from the parameters that already were introduced in sections 3.1, 3.2 and 4.1.2:

\[
Y_i: \text{ The number of demanded machines of type } i, i = 1, ..., n.
\]

\[
d_{yi}: \text{ The due date of machine number } y \text{ of type } i, i = 1, ..., n, y = 1, ..., Y_i.
\]

\[
C_{ytwi}: \text{ The costs to release machine } (y, i) \text{ at time } t \text{ with extra lead time } w, i = 1, ..., n, y = 1, ..., Y_i, w = 0, ..., \lambda_i, t = 0, ..., T - 1.
\]

If the costs are constant over time, which is the case for ASML, since the costs depend on the value of an item, \(C_{ytwi}\) can be calculated as

\[
C_{ytwi} = \alpha_{it}(d_{yi} - (t + \tau_i + w)) + \gamma_{it}w \text{ if } d_{yi} \geq t + \tau_i + w
\]

or as \(C_{ytwi} = \beta_{it}(t + \tau_i + w - d_{yi}) + \gamma_{it}w \text{ if } d_{yi} < t + \tau_i + w\).

The decision variables of Simplified model 1 are:

\[
X_{ytwi}: \text{ Binary decision variable: } X_{ytwi} = 1 \text{ represents the decision to release machine } (y, i) \text{ at time } t \text{ with extra lead time } w \text{ and } X_{ytwi} = 0 \text{ represents the decision to not release machine } (y, i) \text{ at time } t \text{ with extra lead time } w, i = 1, ..., n, y = 1, ..., Y_i, w = 0, ..., \lambda_i, t = 0, ..., T - 1.
\]

\[
Z_{yiyu}: \text{ The amount of resources of group } u \text{ that is claimed by machine } (y, i) \text{ during time slot } t, i = 1, ..., n, y = 1, ..., Y_i, t = 0, ..., T - 1, u \in \mathcal{R}_i.
\]

The mixed integer linear programming formulation of Simplified model 1 is defined as follows:

\[
\min \sum_{t=0}^{T-1} \sum_{i=1}^{n} \sum_{y=1}^{Y_i} \sum_{w=0}^{\lambda_i} C_{ytwi} X_{ytwi}
\]
subject to

\[ \sum_{t=0}^{T-1} \lambda_i \sum_{w=0}^{\tau_i+w} X_{ytwi} = 1 \quad i = 1, \ldots, n, y = 1, \ldots, Y_i \quad (32) \]

\[ \sum_{w=0}^{\tau_i+w} X_{ytwi} p_{lquw}^{\text{min}} \leq Z_{yitu} \leq \sum_{w=0}^{\tau_i+w} X_{ytwi} p_{lquw}^{\text{max}} \quad i = 1, \ldots, n, t = 0, \ldots, T - 1 \quad u \in \mathcal{R}_i, y = 1, \ldots, Y_i \quad (33) \]

\[ \sum_{t=0}^{T-1} Z_{yitu} = \sum_{t=0}^{T-1} \sum_{w=0}^{\tau_i+w} X_{ytwi} p_{lquw}^{\text{total}} \quad i = 1, \ldots, n, y = 1, \ldots, Y_i \quad u \in \mathcal{R}_i \quad (34) \]

\[ \sum_{i=1}^{n} \sum_{y=1}^{Y_i} Z_{yitu} \leq c_{ut} \quad t = 0, \ldots, T, u = 1, \ldots, k \quad (35) \]

### 4.2.3 Test results of Simplified model 1

To be able to compare Simplified model 1 with the Extended model, both were implemented, using AIMMS 3.13. The used test is a variation on the test case used in the section 4.1. It contains 4 items, which are all end items, and 2 resource groups. Both resource groups are used by all 4 items. Moreover all four items can be released with extended lead times. For the first resource group, if these items are released without extra lead time, \( p_{lquw}^{\text{min}} = p_{lquw}^{\text{max}} \) for all \( i \) and \( q \). Additionally, if these items are released with extra lead time, \( p_{lquw}^{\text{min}} = 0 \) for all \( i, q \) and \( w > 0 \). For the second resource group \( p_{lquw}^{\text{min}} = p_{lquw}^{\text{max}} \) for all \( i, w \) and \( q \). The cost parameters are constant over time, based on the values of the items and \( \alpha_{it} = \gamma_{it} \) and \( \beta_{it} = 10\alpha_{it} \) for all items \( i \) and time slots \( t \). Different resource capacity scenarios are used to get multiple results.

The capacity of the first resource group is equal over all time slots except during the summer holiday, where the capacity is lower. Both capacities are varied. The normal capacity is for scenario 2.1a, 2.1b and 2.1c the same, however the summer capacity is, respectively, \( \frac{2}{3}, \frac{1}{2} \) and \( \frac{1}{3} \) of the normal capacity for these scenarios. The normal capacity for the 2.3 scenarios is lower than for the 2.2 scenarios, which is lower than for the 2.1 scenarios. The capacity of the second resource group is constant over all time slots and is not varied.

In table 4 the number of variables and constraints of both models for this test case can be found. Note that the number of constraints is significant smaller using Simplified model 1, while the number of (integer) variables and non zeros is bigger for the Simplified model 1. Furthermore, the number of variables and constraints of Simplified model 1 depends on the number of time slots and the total number of demands. Hence a comparable test case, with more demand, would result in more variables and constraints in Simplified model 1, while these numbers would not change for the Extended model.

The large number of integer variables could be an issue. A way to reduce the
The number of integer variables is to introduce release windows for each end item. A release window is a set of consecutive time slots, in which this particular end item should be released. If a customer wants to receive some end item in time slot 15, it does not make sense to release it in time slot 60, which is possible using Simplified model 1. A possible release window for this item could be [5, 15]. In that case there would be 11 integer variables instead of $T$, the number of time slots for this end item, integer variables. The risk of this idea is that the optimal release date could lay outside the release window, which means that this method will not find the optimal solution. Good release windows would be based on demand date, lead time, amount of resources required, costs of having inventory or backlog and how much resources other releases would like to claim around this time. This implies that it is not easy to find good release windows. The perfect release windows would contain only one time slot and thus be equal to a release plan.

Table 4: The number of variables and constraints of the Extended model and Simplified model 1 in the given test case.

<table>
<thead>
<tr>
<th></th>
<th>Extended</th>
<th>Simplified 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraints</td>
<td>65556</td>
<td>42631</td>
</tr>
<tr>
<td>Variables</td>
<td>33025</td>
<td>63073</td>
</tr>
<tr>
<td>Integer variables</td>
<td>1152</td>
<td>42048</td>
</tr>
<tr>
<td>Non zeros</td>
<td>105825</td>
<td>521745</td>
</tr>
</tbody>
</table>

Table 5: Comparison of the running time of the Extended model and Simplified model 1.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Solving time (seconds)</th>
<th>Gap</th>
<th>Nodes evaluated</th>
<th>Solving time (seconds)</th>
<th>Gap</th>
<th>Nodes evaluated</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1a</td>
<td>1</td>
<td>0.00%</td>
<td>85</td>
<td>5</td>
<td>0.00%</td>
<td>263</td>
</tr>
<tr>
<td>2.1b</td>
<td>4</td>
<td>0.00%</td>
<td>2320</td>
<td>6</td>
<td>0.00%</td>
<td>1025</td>
</tr>
<tr>
<td>2.1c</td>
<td>15</td>
<td>0.00%</td>
<td>7131</td>
<td>35</td>
<td>0.00%</td>
<td>1518</td>
</tr>
<tr>
<td>2.2a</td>
<td>14</td>
<td>0.00%</td>
<td>7641</td>
<td>13</td>
<td>0.00%</td>
<td>11493</td>
</tr>
<tr>
<td>2.2b</td>
<td>261</td>
<td>0.00%</td>
<td>164064</td>
<td>123</td>
<td>0.00%</td>
<td>136114</td>
</tr>
<tr>
<td>2.2c</td>
<td>900</td>
<td>1.01%</td>
<td>679506</td>
<td>900</td>
<td>1.87%</td>
<td>825856</td>
</tr>
<tr>
<td>2.3a</td>
<td>900</td>
<td>1.00%</td>
<td>594890</td>
<td>900</td>
<td>1.63%</td>
<td>685856</td>
</tr>
<tr>
<td>2.3b</td>
<td>900</td>
<td>1.04%</td>
<td>501470</td>
<td>900</td>
<td>1.41%</td>
<td>554866</td>
</tr>
<tr>
<td>2.3c</td>
<td>900</td>
<td>1.71%</td>
<td>346994</td>
<td>900</td>
<td>2.27%</td>
<td>318417</td>
</tr>
</tbody>
</table>

In table 5 the results of the test cases can be found. Note that Simplified model 1 is on average slightly faster in the scenarios where the optimal solution is reached. However in the scenarios where the optimal solution is not found the difference between the best found solution and the lower bound after 900 seconds is bigger for Simplified model 1 than for the Extended model. Some tests were done with release windows based on demand date and lead time, like having release windows $[d_{yi} - 3\tau_i, d_{yi} - 3\tau_i]$ and $[d_{yi} - 5\tau_i, d_{yi} - 3\tau_i]$. But these tests did not result in lower running times. So, the results suggest that Simplified model 1 is not faster than the Extended model.
<table>
<thead>
<tr>
<th></th>
<th>Extended</th>
<th>Simplified 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraints</td>
<td>65556</td>
<td>375</td>
</tr>
<tr>
<td>Variables</td>
<td>33025</td>
<td>42049</td>
</tr>
<tr>
<td>Integer variables</td>
<td>1152</td>
<td>42048</td>
</tr>
<tr>
<td>Non zeros</td>
<td>105825</td>
<td>484775</td>
</tr>
</tbody>
</table>

Table 6: The number of variables and constraints of the Extended model and Simplified model 2 in the given test case.

4.3 Alternative capacity and inventory constraints

In this section a model, which combines the alternative capacity constraints described in section 4.1 with the alternative for the inventory constraints described in section 4.2, is described and compared with the Extended model.

4.3.1 Simplified model 2

Simplified model 2 uses the same capacity constraints as Alternative model 1 and the same inventory constraints as Simplified model 1. Hence the introduction of new parameters and variables is not needed for this model and constraint (32) and a variation of constraint (28) is used. The mixed integer linear programming formulation of Simplified model 2 is defined as follows:

$$
\min \sum_{t=0}^{T-1} \sum_{i=1}^{n} \sum_{y=1}^{Y_i} \sum_{w=0}^{\lambda_i} C_{ytwi} X_{ytwi}
$$

subject to

$$
\sum_{t=0}^{T-1} \sum_{w=0}^{\lambda_i} X_{ytwi} = 1 \quad i = 1, \ldots, n, y = 1, \ldots, Y_i \quad (36)
$$

$$
\sum_{i \in I} \sum_{w=0}^{t} Y_i \sum_{w} X_{ytwi} \leq \sum_{t \in \sigma_u} c_{ut} \quad \forall \sigma_u \in \Sigma_u, u = 1, \ldots, k \quad (37)
$$

4.3.2 Test results of Simplified model 2

Like the other models, Simplified model 2 was implemented, using AIMMS 3.13. The same test case as in subsection 4.2.3 is used. Besides the 72 sets of length 1 for both resource groups, 90 sets for the first resource group were used. These sets were found by trial and error: Simplified model 2 was solved, after which a maximum flow problem was solved, which was used to see if there was enough capacity of the first resource group and where the shortages could be. These periods of shortages were used to add some constraints and this process was repeated. Note that for the second resource group holds that $p_{i|u|w}^{\min} = p_{i|u|w}^{\max}$ for all $i$, $w$ and $q$ and thus the sets of length 1 are sufficient.
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Solving time (seconds)</th>
<th>Nodes evaluated</th>
<th>Gap</th>
<th>Solving time (seconds)</th>
<th>Nodes evaluated</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1a</td>
<td>1</td>
<td>85</td>
<td>0.00%</td>
<td>1</td>
<td>0</td>
<td>0.00%</td>
</tr>
<tr>
<td>2.1b</td>
<td>4</td>
<td>2320</td>
<td>0.00%</td>
<td>3</td>
<td>2076</td>
<td>0.00%</td>
</tr>
<tr>
<td>2.1c</td>
<td>15</td>
<td>7131</td>
<td>0.00%</td>
<td>2</td>
<td>982</td>
<td>0.00%</td>
</tr>
<tr>
<td>2.2a</td>
<td>14</td>
<td>7641</td>
<td>0.00%</td>
<td>3</td>
<td>1621</td>
<td>0.00%</td>
</tr>
<tr>
<td>2.2b</td>
<td>261</td>
<td>164064</td>
<td>0.00%</td>
<td>11</td>
<td>15586</td>
<td>0.00%</td>
</tr>
<tr>
<td>2.2c</td>
<td>900</td>
<td>679506</td>
<td>1.01%</td>
<td>10</td>
<td>17059</td>
<td>0.00%</td>
</tr>
<tr>
<td>2.3a</td>
<td>900</td>
<td>594890</td>
<td>1.00%</td>
<td>900</td>
<td>1812183</td>
<td>0.58%</td>
</tr>
<tr>
<td>2.3b</td>
<td>900</td>
<td>501470</td>
<td>1.04%</td>
<td>900</td>
<td>1164304</td>
<td>0.85%</td>
</tr>
<tr>
<td>2.3c</td>
<td>900</td>
<td>346994</td>
<td>1.71%</td>
<td>900</td>
<td>1202704</td>
<td>1.15%</td>
</tr>
</tbody>
</table>

Table 7: Comparison of the running time of the Extended model and Simplified model 2.

In table 6 the number of variables and constraints of both models for this test case can be found. Note the small number of constraints of Simplified model 2, which should be convenient for the solver. In Simplified model 2, there is one constraint per demand and one per set $\sigma_i$. In table 7 the results of the test cases can be found. Note that Simplified model 2 is faster in the scenarios where the optimal solution is found. And in the scenarios where the optimal solution is not found, the difference between the best found solution and the lower bound after 900 seconds, is smaller for Simplified model 2. These results suggest that when the right set are used Simplified model 2 is significantly faster than the Extended model, when solving instances with only end items.
5 Algorithms using Alternative model 1

In this chapter the focus is on Alternative model 1, since it seems to be significantly faster than the Extended model if the right sets are used. Two algorithms to find sufficient sets and the optimal release plan will be presented and tested. In section 5.1 the ideas used in these algorithms are described and some simpler algorithms are given.

5.1 Ideas

The main idea is to solve Alternative model 1 without all capacity constraints, then find some violated constraints that have not been added yet, add those, resolve Alternative model 1 and repeat this process until a feasible optimal solution is found. To do this efficiently a fast method to find violated constraints is needed. In the first subsection a first basic algorithm is given and after this several ways to improve this algorithm are described.

5.1.1 First basic algorithm

The starting point is Alternative model 1 with all inventory constraints and only the capacity constraints with \( \sigma_u = \{s\} \) for all time slots \( s \) and resource groups \( u \). This way the number of capacity constraints is reasonable small and some of the most obvious constraints are added, since these constraints make sure that at least the sum of the minimal required capacities \( \sum_{itw} R_{itw} p_{\min}^{itw} \) in every time slot \( s \) of resource group \( u \) is available. Note that for each resource group \( u \) and all sets of release orders \( R_{itw} \) satisfying this constraints, there is a feasible flow in the maximum flow graph.

In chapter 4.1.2 a method to find a set of time slots \( \sigma_u \) for which the constraint \( \sum_{itw} R_{itw} \chi_{itw} \sigma_u \leq \sum_{s} \chi_{s} c_{us} \) of Alternative model 1 is violated, is described. This method finds a set \( \sigma_u \) belonging to a violated constraint if there is a feasible flow in the maximum flow graph and there is not enough capacity of resource group \( u \) to produce all materials according to the release orders. This method is called Algorithm 1.

| **Input:** | Release orders \( R_{itw} \) and resource group \( u \); Construct the maximum flow graph described in chapter 4.1.2 for resource group \( u \), given release orders \( R_{itw} \); Find the maximum flow in this graph; Construct the residual graph of the maximum flow; Find the set \( \sigma_u \) of all nodes \( s \) which can be reached from the source in the residual graph; |
| **Output:** | \( \sigma_u \); |

Algorithm 1: Method to find the set of time slots \( \sigma_u \) belonging to a violated constraint.
During the implementation an alternative for Algorithm 1 was used. The implemented method finds the minimum source-sink cut of the residual graph of the minimal required flow on the maximum flow graph, such that $\sigma_u$, the set of time slots $s$ which are connected to the source is the smallest. This method is based on the max-flow min-cut theorem. Another example of an alternative is a variation of Algorithm 1 which uses the LP-formulation of the maximum flow problem. Then $\sigma_u$ is equal to the set of time slots $s$ for which the flow conservation constraint has a positive shadow price after solving the linear program. Note that all steps in all of these methods take polynomial time and thus finding a violated constraint using one of these variants should not take much time, regardless of which one is used.

The most simple algorithm to solve Alternative model 1 without adding all capacity constraints is Algorithm 2, which uses Algorithm 1 to find missing constraints.

| ModelAlt := Alternative model 1 with all inventory constraints and only the capacity constraints with $\sigma_u = \{s\}$ for all time slots $s$ and resource groups $u$; |
| repeat |
| Solve ModelAlt ; |
| for $u = 1$ to $k$ do |
| Use Algorithm 1 to find the set of time slots $\sigma_u$ given $u$ and the solution of ModelAlt; |
| if $\sigma_u \neq \emptyset$ then |
| Add constraint $\sum_{itw} R_{itw} x_{itwua} \leq \sum_{tse} c_{ut}$ to ModelAlt; |
| end |
| end |
| until $\sigma_u = \emptyset, \forall u$; |

Algorithm 2: The most simple algorithm to solve Alternative model 1 without adding all capacity constraints.

Sometimes it is better to partition the set $\sigma_u$ found by Algorithm 1 into two or more subsets and add a capacity constraint for each of these subsets than to add the capacity constraint belonging to $\sigma_u$. This happens when it is possible to partition $\sigma_u$ into $l$ parts: $\sigma_u^1, \sigma_u^2, \ldots, \sigma_u^l$, such that for all release orders $R_{itw}$ holds that $x_{itwua} = x_{itwua}^1 + x_{itwua}^2 + \cdots + x_{itwua}^l$. If this is the case and $l$ is for example 2, the constraints $\sum_{itw} R_{itw} x_{itwua}^1 \leq \sum_{s} c_{us}^1$ and $\sum_{itw} R_{itw} x_{itwua}^2 \leq \sum_{s} c_{us}^2$ together imply the constraint $\sum_{itw} R_{itw} x_{itwua} \leq \sum_{s} c_{us}$. However the other way around is not always true. Note that based on this idea a heuristic method to split the sets found by Algorithm 1 was implemented for the rest of the algorithms. This method splits a set if the distance between two subsequent time slots in this set is as big or bigger than the longest lead time.

This, for example, is useful in a scenario, where the lead times are smaller than 20 weeks and the summer holiday $\sigma_u^1$ and the winter holiday $\sigma_u^2$ are the only times where there is not enough capacity of resource group $u$. Algorithm 1 will find the set $\sigma_u$ including both holidays, but there are no releases that could claim capacity in both holidays. Hence $x_{itwua}^1 = 0$ or $x_{itwua}^2 = 0$ and $x_{itwua} = x_{itwua}^1$ or $x_{itwua} = x_{itwua}^2$ for
all releases $R_{itw}$. Adding the constraints for $\sigma^1_u$ and $\sigma^2_u$ makes sure there is enough capacity during the summer and winter holiday, while adding the constraint for $\sigma_u$ does not guarantee this. That constraint makes sure that the total capacity in $\sigma_u$ is bigger than the number of units of resource group $u$ claimed in the summer holiday plus this amount in the winter holiday. However, for the release orders this leaves the possibility to claim less capacity than available in the winter holiday and more capacity than available in the summer holiday, as long as the total amount of claimed resources is smaller than the total available amount of resources in $\sigma_u$.

5.1.2 LP-relaxation

Algorithm 2 is in general not very fast, since it solves Alternative model 1 fully every time it turns out that another constraint is needed. To find some of the needed constraints fast the LP-relaxation of Alternative model 1 can be used, by replacing Alternative model 1 by its LP-relaxation in Algorithm 2. This does not take much time, because pure linear programs can be solved in polynomial time. Constraints will be found, which hold for the solution of the LP-relaxation of Alternative model 1 and thus also hold for the solution of Alternative model 1. After doing this with the LP-relaxation the constraints that were found can be added to the starting point of Algorithm 2. This speeds up the process, because Alternative model 1 is solved fewer times in this way. An algorithm to solve Alternative model 1 without adding all capacity constraints, which uses this idea is Algorithm 3. Note that the splitting of $\sigma_u$ is possible if $\chi_{itwuo_u} = \chi_{itwuo_1} + \chi_{itwuo_2} + \cdots + \chi_{itwuo_c}$ as described in the previous subsection.

5.1.3 Branch and cut

Algorithms like Algorithm 3 are sometimes called cut and branch algorithms. Using the idea of finding necessary constraints using the LP-relaxation, it is also possible to design a branch and cut algorithm. The branch and cut algorithm would start the same as the cut and branch algorithm: In the first node of the branch and bound tree the LP-relaxation of Alternative model 1 without all capacity constraints is solved and Algorithm 1 is used to find violated constraints, which are added and the LP-relaxation is solved again. This is repeated until no more violated constraints are found. Then there is branched: The problem will be split into two (or more) subproblems.

Each subproblem is a linear programming problem and is solved, after that Algorithm 1 is used to find and add violated constraints. These are called global cuts, since every feasible solution satisfies these constraints and not only the feasible solutions in this branch. After adding the violated constraints this subproblem is solved again and this is repeated until no more cuts are found. If the solution is integer the upper bound is updated and another branch is considered. Otherwise, there is branched again. Branches in which the objective value of the solution of the subproblem is higher than the upper bound is discarded. This will be repeated until all branches are taken into consideration.

There are several variations on this algorithm. For example by only looking for violated constraints if the subproblem is less than 5 branches deep in the branch tree
ModelAlt := Alternative model 1 with all inventory constraints and only the capacity constraints with \( \sigma_u = \{s\} \) for all time slots \( s \) and resource groups \( u \);
Change ModelAlt into its LP relaxation;
repeat
  Solve ModelAlt;
  for \( u = 1 \) to \( k \) do
    Use Algorithm 1 to find the set of time slots \( \sigma_u \) given \( u \) and the solution of ModelAlt;
    if \( \sigma_u \neq \emptyset \) then
      Split \( \sigma_u \) if possible;
      Add a capacity constraint for all parts of \( \sigma_u \) to ModelAlt;
    end
  end
until \( \sigma_u = \emptyset, \forall u \);
Change ModelAlt into MILP;
repeat
  Solve ModelAlt;
  for \( u = 1 \) to \( k \) do
    Use Algorithm 1 to find the set of time slots \( \sigma_u \) given \( u \) and the solution of ModelAlt;
    if \( \sigma_u \neq \emptyset \) then
      Split \( \sigma_u \) if possible;
      Add a capacity constraint for all parts of \( \sigma_u \) to ModelAlt;
    end
  end
until \( \sigma_u = \emptyset, \forall u \);

Algorithm 3: A simple algorithm to solve Alternative model 1 without adding all capacity constraints, using the LP-relaxation to speed up the process.
or if the solution of the subproblem is integer.

### 5.1.4 Callbacks

To solve Alternative model 1, and mixed integer linear programs in general, solvers use branch and bound. During this process several feasible solutions are found. The best one is remembered and used as upper bound (in case of minimisation) and when it is shown that the lower bound is equal to the upper bound and thus that there does not exist a better solution the solver is finished.

A callback function is a function that gets triggered by the solver if certain criteria are met during the branch and bound process. A New incumbent callback is triggered if the solver finds a feasible solution which is strictly better than the previous best feasible solution. This better feasible solution is called a new incumbent. In this function the new incumbent can be evaluated and if needed rejected. Furthermore a callback could save information about the solving process or stop the solver.

Algorithm 2 and Algorithm 3 start with Alternative model 1 without all capacity constraints. The constraints that are not added yet are sometimes called lazy constraints. The goal is to find the optimal solution that satisfies all lazy constraints, without adding all of them, if possible. If a solution is given, finding a lazy constraint that is not satisfied can be done using Algorithm 1. A Lazy constraint callback is triggered if the solver finds a new feasible solution and can be used to add a lazy constraint.

To further speed up the process of finding the needed sets and the optimal solution of Alternative model 1, the idea is to use these callback functions to check if the intermediate feasible solutions are indeed release plans that do not require more resources than available. Contrary to waiting until the solver is finished and checking if the optimal solution is a release plan that does not require more resources than available. This speeds up the process because if one of the intermediate feasible solutions requires more resources than available, there is a reasonable chance that the optimal solution also requires more resources than available. Besides this the solvers need quite some time to prove that the last found feasible solution is indeed the optimal one.

### 5.2 Algorithm 4

Algorithm 4 is an extension of Algorithm 3. It uses a New incumbent callback to find missing constraints: If a better feasible solution is found the algorithm checks if it satisfies the lazy constraints. If this is the case the solver continues. Otherwise, the solver stops, the missing constraints are added and the solver starts solving the model, with the additional constraints, from the beginning.

### 5.3 Algorithm 5

Algorithm 5 is a variation on Algorithm 4 and could be called a branch and cut algorithm. It uses a Lazy constraint callback to find missing constraints: If a new feasible solution is found the algorithm checks if the solution satisfies all lazy constraints. If
ModelAlt := Alternative model 1 with all inventory constraints and only the capacity constraints with $\sigma_u = \{s\}$ for all time slots $s$ and resource groups $u$; Change ModelAlt into its LP relaxation;

repeat
  Solve ModelAlt;
  for $u = 1$ to $k$ do
    Use Algorithm 1 to find the set of time slots $\sigma_u$ given $u$ and the solution of ModelAlt;
    if $\sigma_u \neq \emptyset$ then
      Split $\sigma_u$ if possible;
      Add a capacity constraint for all parts of $\sigma_u$ to ModelAlt;
    end
  end
until $\sigma_u = \emptyset, \forall u$;

Change ModelAlt into MILP;

repeat
  Solve ModelAlt with callback:
  if new incumbent is found then
    for $u = 1$ to $k$ do
      Use Algorithm 1 to find the set of time slots $\sigma_u$ given $u$ and the solution of ModelAlt;
    end
    if $\sigma_u = \emptyset, \forall u$ then
      Continue solving ModelAlt;
    else
      Stop solving ModelAlt;
    end
  end
  for $u = 1$ to $k$ do
    if $\sigma_u \neq \emptyset$ then
      Split $\sigma_u$ if possible;
      Add a capacity constraint for all parts of $\sigma_u$ to ModelAlt;
    end
  end
until $\sigma_u = \emptyset, \forall u$;

Algorithm 4: An algorithm to solve Alternative model 1 without adding all capacity constraints, using the LP-relaxation and a New incumbent callback to speed up the process.
this is not the case the missing constraints are added and the solving is continued at
the point where the feasible solution was found. If all lazy constraints are satisfied the
solving is continued at the point where the feasible solution was found.

\[
\text{ModelAlt} := \text{Alternative model 1 with all inventory constraints and only the}
\text{capacity constraints with } \sigma_u = \{s\} \text{ for all time slots } s \text{ and resource groups } u;
\]
\[
\text{Change ModelAlt into its LP relaxation;}
\]
\[
\text{repeat}
\]
\[
\text{Solve ModelAlt ;}
\]
\[
\text{for } u = 1 \text{ to } k \text{ do}
\]
\[
\text{Use Algorithm 1 to find the set of time slots } \sigma_u \text{ given } u \text{ and the solution of}
\text{ModelAlt;}
\]
\[
\text{if } \sigma_u \neq \emptyset \text{ then}
\]
\[
\text{Split } \sigma_u \text{ if possible;}
\]
\[
\text{Add a capacity constraint for all parts of } \sigma_u \text{ to ModelAlt;}
\]
\[
\text{end}
\]
\[
\text{end}
\]
\[
\text{until } \sigma_u = \emptyset, \forall u;
\]
\[
\text{Change ModelAlt into MILP ;}
\]
\[
\text{Solve ModelAlt with callback:}
\]
\[
\text{if feasible solution is found then}
\]
\[
\text{for } u = 1 \text{ to } k \text{ do}
\]
\[
\text{Use Algorithm 1 to find the set of time slots } \sigma_u \text{ given } u \text{ and the solution of}
\text{ModelAlt;}
\]
\[
\text{if } \sigma_u \neq \emptyset \text{ then}
\]
\[
\text{Split } \sigma_u \text{ if possible;}
\]
\[
\text{Add a capacity constraint for all parts of } \sigma_u \text{ to ModelAlt;}
\]
\[
\text{end}
\]
\[
\text{end}
\]
\[
\text{Continue solving ModelAlt ;}
\]
\[
\text{end}
\]
\[
\text{end}
\]
\[
\text{Algorithm 5: An algorithm to solve Alternative model 1 without adding all capacity}
\text{constraints, using the LP-relaxation and a Lazy constraint callback to speed up}
\text{the process.}
\]

5.4 Test results

To be able to compare the running times of Algorithm 4 and Algorithm 5 with the
Extended model, both were implemented, using AIMMS 3.13. Two test cases were
used to compare the speed of Algorithm 4, Algorithm 5 and the Extended model. As
in the previous tests CPLEX 12.5 was used. However, comparable results were found
on a similar test case using GUROBI 5.5, although AIMMS 3.13 sometimes bugged in
the callback functions while using GUROBI 5.5. This is why the results using GUROBI
5.5 are not included in this thesis.
The first test case is the same as in section 4.1.3. It has a convergent supply chain and contains 72 time slots, 32 items, of which 4 end items, and 1 resource group. This resource group is used by 8 different items, which are also the items which can be released with extended lead times. If these items are released without extra lead time, \( p_{i,q,w}^{\text{min}} = p_{i,q,w}^{\text{max}} \) for all \( i \) and \( q \), while \( w = 0 \) and \( u \) is the only resource. If these items are released with extra lead time, then \( p_{i,q,w}^{\text{min}} = 0 \). The cost parameters are constant over time, based on the values of the items. Furthermore, it holds that \( \alpha_{it} = \gamma_{it} \) and \( \beta_{it} = 10\alpha_{it} \) for all items \( i \) and time slots \( t \). Different resource capacity scenarios are used to get multiple results.

The capacity is constant over all time slots except during the summer holiday, where the capacity is lower. Both capacities are varied. The \( a, b \) and \( c \) always correspond to a summer capacity of \( \frac{2}{3}, \frac{1}{2} \) and \( \frac{1}{3} \), respectively, times the normal capacity. The normal capacity is for scenario 1.1a, 1.1b and 1.1c the same. The normal capacity for the 1.5 scenarios is lower than for the 1.4 scenarios, which is lower than for the 1.3 scenarios and so on.

The second test case is an extension of the previous test case. It contains 2 extra resource groups. One is required by the 4 end items and the other is required by 3 of the components of the FSSN’s. Note that \( p_{i,q,w}^{\text{min}} = p_{i,q,w}^{\text{max}} \) for all \( i, q \) and \( w \), for both these resource groups. The available capacities of the two additional resource groups are constant over time and are not varied in the different resource scenarios. For the different scenarios the available capacity of the first resource group is varied in the same way as the first test case.

The results of the first test case can be found in table 8. Note that the running time includes the time the solver needs to generate the models here, because Algorithm 4 and Algorithm 5 generates multiple (integer) linear programs, since they are implemented with a linear program to find the minimum cut to check if any constraint is violated. Note that Algorithm 4 is faster than the Extended model in all scenarios, except in scenarios 1.1a, 1.2a, 1.3a, 1.1c and 1.5c. In 1.1c and 1.5c it is not clear which method is faster, but after approximal 900 seconds both methods found the same upper bounds and the gaps between the lower bounds and the best feasible solutions that were found, are smaller for Algorithm 4. In 1.1a, 1.2a, 1.3a the optimal solution has objective value 0 and most of the running time is spend on the generation of the (integer) linear program(s). Since Algorithm 4 and Algorithm 5 generate multiple (integer) linear programs it makes sense that they take slightly more time to run these scenarios.

The results of the second test case can be found in table 9. Note that a few scenarios are solved significantly faster by Algorithm 4 than by the Extended model, while the difference in solving time in the rest of the scenarios is small. Thus, in this test case Algorithm 4 is on average faster than the Extended model, but this is less clear than in the first test case. An explanation for this could be that \( p_{i,q,w}^{\text{min}} = p_{i,q,w}^{\text{max}} \) for all \( i, q \) and \( w \), for both of the added resources. The capacity constraints of Alternative model 1 for these resources and sets of time slots of length one are basically the same as the capacity constraint of the Basic model, which is also true for capacity constraints of the Extended model for these resources.

For each scenario in both test cases the running time or the gap of Algorithm 4 is
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Running time (seconds)</th>
<th>Gap</th>
<th>Running time (seconds)</th>
<th>Gap</th>
<th>Running time (seconds)</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended</td>
<td>Algorithm 4</td>
<td>Algorithm 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1a</td>
<td>2 0.00%</td>
<td>3 0.00%</td>
<td>3 0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1b</td>
<td>25 0.00%</td>
<td>7 0.00%</td>
<td>27 0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1c</td>
<td>900 0.42%</td>
<td>903 0.16%</td>
<td>905 0.41%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2a</td>
<td>2 0.00%</td>
<td>3 0.00%</td>
<td>3 0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2b</td>
<td>900 1.01%</td>
<td>154 0.00%</td>
<td>905 0.82%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2c</td>
<td>901 0.23%</td>
<td>23 0.00%</td>
<td>904 0.06%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3a</td>
<td>2 0.00%</td>
<td>2 0.00%</td>
<td>3 0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3b</td>
<td>765 0.00%</td>
<td>51 0.00%</td>
<td>906 0.66%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3c</td>
<td>679 0.00%</td>
<td>172 0.00%</td>
<td>43 0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4a</td>
<td>10 0.00%</td>
<td>10 0.00%</td>
<td>46 0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4b</td>
<td>900 0.50%</td>
<td>20 0.00%</td>
<td>908 0.50%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4c</td>
<td>901 0.63%</td>
<td>473 0.00%</td>
<td>913 0.02%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5a</td>
<td>54 0.00%</td>
<td>28 0.00%</td>
<td>703 0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5b</td>
<td>900 0.61%</td>
<td>52 0.00%</td>
<td>915 0.36%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5c</td>
<td>901 0.58%</td>
<td>1002 0.31%</td>
<td>922 0.56%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8: Comparison of the running time of the Extended model, Algorithm 4 and Algorithm 5.

smaller than those of Algorithm 5, except for scenarios 1.3c and 3.3c. In some scenarios the Extended model solves the instance faster and in other scenarios Algorithm

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Running time (seconds)</th>
<th>Gap</th>
<th>Running time (seconds)</th>
<th>Gap</th>
<th>Running time (seconds)</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1a</td>
<td>4 0.00%</td>
<td>11 0.00%</td>
<td>12 0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1b</td>
<td>395 0.00%</td>
<td>28 0.00%</td>
<td>300 0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1c</td>
<td>900 0.21%</td>
<td>912 0.05%</td>
<td>914 0.16%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.2a</td>
<td>4 0.00%</td>
<td>11 0.00%</td>
<td>11 0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.2b</td>
<td>187 0.00%</td>
<td>37 0.00%</td>
<td>913 0.10%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.2c</td>
<td>53 0.00%</td>
<td>41 0.00%</td>
<td>54 0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3a</td>
<td>4 0.00%</td>
<td>11 0.00%</td>
<td>11 0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3b</td>
<td>900 0.31%</td>
<td>151 0.00%</td>
<td>909 0.31%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3c</td>
<td>374 0.00%</td>
<td>306 0.00%</td>
<td>104 0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4a</td>
<td>9 0.00%</td>
<td>17 0.00%</td>
<td>24 0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4b</td>
<td>900 0.25%</td>
<td>1730 0.19%</td>
<td>914 0.25%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4c</td>
<td>902 0.29%</td>
<td>922 0.35%</td>
<td>919 0.38%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.5a</td>
<td>902 1.16%</td>
<td>312 0.00%</td>
<td>915 1.47%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.5b</td>
<td>902 0.19%</td>
<td>942 0.19%</td>
<td>916 0.19%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.5c</td>
<td>902 0.64%</td>
<td>1183 0.38%</td>
<td>923 0.57%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9: Comparison of the running time of the Extended model, Algorithm 4 and Algorithm 5, using a test case that contains 3 resource groups.
5. Hence, it is not clear if the Extended model or Algorithm 5 is faster. Apparently starting over the solving process after adding a violated constraint gives better results than continuing the solving process after adding a violated constraint. Thus the optimal solution of Alternative model 1 is often not close to the solution of the model without this violated constraint added in the branch and bound tree.

Note that the running times fluctuate a lot for each method. Some optimal solutions are found within a minute, where others take more than 15 minutes. In most cases a (near) optimal solution is found quite fast and a lot of time is spend on raising the lower bound. In some scenarios there are many (near) optimal solutions quite far away from each other in the branch and bound tree. There are some simple examples of instances that have 2 different solutions, both with the same objective value. Like having an item one time slot in inventory after its production or releasing the same item with one week extra lead time, without claiming any resources in this last week, costs the same and claims the same amount of resources in each time slot. Another example is wanting to produce two items at the same time, but this is not possible because of a small resource shortage in the middle of the production. Then releasing both items one week earlier and with one week extra lead time could solve this problem, while another solution could be to release one of the two items two weeks earlier and with two weeks extra lead time and release the other item at its original release time and without extra lead time. Both solutions cost the same. In a bigger scenario the solver has to consider many of this kind of options.


6 Approximation algorithms

In the previous chapter Algorithm 4, which can find optimal solutions faster than the Extended model is presented. However, in some cases the algorithm still took a considerable amount of time to solve the problem. Since SPOT should be able to solve even bigger instances, in this chapter some algorithms, which can be used to find near optimal solutions, are presented. In section 6.1 the solution of the LP-relaxation is used and in section 6.2 the machines are planned first and the rest of the supply chain later.

Note that the most simple approximation algorithm is to give the Extended model to a solver and let the solver run for \( X \) seconds. If \( X \) is big enough, the solver will find a feasible integer solution and a lower bound. The advantages of this method are that the lower bound helps to evaluate the quality of the approximation, it never uses more than \( X \) seconds and if \( X \) is big enough the optimal solution can be found. Approximation algorithm 1 is this method with \( X = 60 \). Results of this algorithm can be found in table 10.

6.1 LP relaxation

There are several ways to use the solution of the LP-relaxation to get a feasible solution of the original problem. One method is to solve the LP-relaxation, use the solution of the LP-relaxation to add constraints to the problem and then solve the integer linear program. The idea is that these constraints cut of a big part of the feasible set and thus make the integer linear problem easier to solve. However, the added constraints should not cut off all good integer solutions and for sure not all integer solutions. Three examples of this type of approximation algorithms are given. They all use Algorithm 4 as a basis, since it is a fast method and it can easily be adjusted. First, in Algorithm 6, Algorithm 4 is adjusted, such that it can be used as an approximation algorithm. After this, three approximation algorithms are described by explaining which constraints they add based on the LP-relaxation.

Approximation algorithm 2 adds the constraints \( R_{itw} \leq \lceil R_{itw}^{LP} \rceil \) for all \((i, t, w)\) for which \( R_{itw}^{LP} > 0 \), where \( R_{itw}^{LP} \) is the number of items \( i \) released at time \( t \) with extra lead time \( w \) in the solution of the LP-relaxation. Results of this method can be found in table 10. In almost all of the scenarios the optimal solution is found using this method and the running time is less than a minute, except for the last scenario. This suggest that this is a very strong approximation algorithm. However, there is an easy example in which it does not find a good solution.

Suppose that the demand on an item \( i \) is 4 at the end of the third time slot. This item \( i \) has lead time 1 and uses 2 units of some resource group \( u \). Suppose the capacity of resource group \( u \) is 4 in the first time slot and 3 in the second and third time slot. The solution of the LP-relaxation releases 1 item \( i \) in time slot 0 and 1.5 in time slot 1 and 2. The optimal solution releases 2 items \( i \) in time slot 0 and 1 in time slot 1 and 2. However, the approximation algorithm adds the constraint \( R_{itw} \leq \lceil R_{itw}^{LP} \rceil \) for \( t = 0 \), which makes it impossible to release two items in time slot 0. Thus the solution of the approximation algorithm is to release 1 item \( i \) in time slot 0,1,2 and 3. This

42
ModelAlt := Alternative model 1 with all inventory constraints and only the capacity constraints with $\sigma_u = \{s\}$ for all time slots $s$ and resource groups $u$; Change ModelAlt into its LP relaxation;

repeat
  Solve ModelAlt;
  for $u = 1$ to $k$ do
    Use Algorithm 1 to find the set of time slots $\sigma_u$ given $u$ and the solution of ModelAlt;
    if $\sigma_u \neq \emptyset$ then
      Split $\sigma_u$ if possible;
      Add a capacity constraint for all parts of $\sigma_u$ to ModelAlt;
    end
  end
  until $\sigma_u = \emptyset, \forall u$;

Add constraints using the solution of the LP-relaxation. Change ModelAlt into MILP;

repeat
  Solve ModelAlt with callback:
  if new incumbent is found then
    for $u = 1$ to $k$ do
      Use Algorithm 1 to find the set of time slots $\sigma_u$ given $u$ and the solution of ModelAlt;
    end
    if $\sigma_u \neq \emptyset, \forall u$ then
      Continue solving ModelAlt;
    else
      Stop solving ModelAlt;
    end
  end
  for $u = 1$ to $k$ do
    if $\sigma_u \neq \emptyset$ then
      Split $\sigma_u$ if possible;
      Add a capacity constraint for all parts of $\sigma_u$ to ModelAlt;
    end
  end
  until $\sigma_u = \emptyset, \forall u$;

Algorithm 6: An adjusted version of Algorithm 4, used as an approximation algorithm. Which constraints are added based on the solution of the LP-relaxation can be varied.
means there is a backlog of 1 during time slot 4. This costs way more than having it in inventory for two time slots, which is the case in the optimal solution.

Approximation algorithm 3 adds the constraints \( R_{itw} = R_{itw}^{LP} \) for all \((i, t, w)\) for which \( R_{itw}^{LP} > 0 \) and \( R_{i,t+1,w}^{LP} \) and \( R_{i,t+2,w}^{LP} \) are integers. This way a lot of releases are fixed, but there still is space to move around the releases that are split in the solution of the LP-relaxation. In general these split releases are released in an earlier time slot, since a later time slot probably causes backlog, which costs more than inventory. Results of this method can be found in table 10. In all scenarios this approximation algorithm finished within 40 seconds, however the difference between the objective value of the solution found by the approximation algorithms and the best known solution, is quite big in comparison with the other approximation algorithms. Note that the scenarios for which the relative difference between the found solution and the best known solution are big, have the best known solutions with the smallest objective values.

Approximation algorithm 4 adds the constraints \( R_{itw} = 0 \) for all \((i, t, w)\) for which \( R_{itw}^{LP} \) and \( R_{i,t+1,w}^{LP} \) are equal to zero. This essentially eliminates a lot of (hopefully) useless variables and thus makes the problem easier to solve. In all scenarios this approximation algorithm finished within 45 seconds and the relative difference between the found solution and the best known solution is in almost all cases less than 1.00%.

<table>
<thead>
<tr>
<th>Scen.</th>
<th>Approx. 1</th>
<th>Approx. 2</th>
<th>Approx. 3</th>
<th>Approx. 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Run. time (s)</td>
<td>Approx. gap</td>
<td>Run. time (s)</td>
<td>Approx. gap</td>
</tr>
<tr>
<td>3.1a</td>
<td>4</td>
<td>0.00%</td>
<td>19</td>
<td>0.00%</td>
</tr>
<tr>
<td>3.1b</td>
<td>60</td>
<td>0.00%</td>
<td>28</td>
<td>0.00%</td>
</tr>
<tr>
<td>3.1c</td>
<td>62</td>
<td>0.00%</td>
<td>28</td>
<td>0.00%</td>
</tr>
<tr>
<td>3.2a</td>
<td>4</td>
<td>0.00%</td>
<td>18</td>
<td>0.00%</td>
</tr>
<tr>
<td>3.2b</td>
<td>60</td>
<td>0.00%</td>
<td>30</td>
<td>0.00%</td>
</tr>
<tr>
<td>3.2c</td>
<td>54</td>
<td>0.00%</td>
<td>26</td>
<td>0.08%</td>
</tr>
<tr>
<td>3.3a</td>
<td>4</td>
<td>0.00%</td>
<td>19</td>
<td>0.00%</td>
</tr>
<tr>
<td>3.3b</td>
<td>60</td>
<td>0.00%</td>
<td>29</td>
<td>0.00%</td>
</tr>
<tr>
<td>3.3c</td>
<td>62</td>
<td>0.00%</td>
<td>29</td>
<td>0.00%</td>
</tr>
<tr>
<td>3.4a</td>
<td>11</td>
<td>0.00%</td>
<td>22</td>
<td>0.00%</td>
</tr>
<tr>
<td>3.4b</td>
<td>60</td>
<td>0.56%</td>
<td>47</td>
<td>0.00%</td>
</tr>
<tr>
<td>3.4c</td>
<td>62</td>
<td>0.00%</td>
<td>50</td>
<td>0.00%</td>
</tr>
<tr>
<td>3.5a</td>
<td>62</td>
<td>0.00%</td>
<td>47</td>
<td>0.00%</td>
</tr>
<tr>
<td>3.5b</td>
<td>62</td>
<td>1.32%</td>
<td>54</td>
<td>0.00%</td>
</tr>
<tr>
<td>3.5c</td>
<td>62</td>
<td>0.02%</td>
<td>176</td>
<td>0.02%</td>
</tr>
<tr>
<td>avg.</td>
<td>46</td>
<td>0.12%</td>
<td>41</td>
<td>0.01%</td>
</tr>
</tbody>
</table>

Table 10: The running times and the relative difference between the objective value of the solution found by the approximation algorithms and the best known solution of 4 different approximation algorithms. It uses the same test cases as in chapter 5, which include three resources.

It is not recommended to blindly choose an approximation algorithm based on these results. In addition, note that for Approximation algorithm 3 and Approxima-
tion algorithm 4 there are examples like the one that shows that Approximation algo-
rithm 2 sometimes finds bad solutions. Furthermore, in all methods an integer linear
programm is solved and thus the running time can be very high for some instances.
These examples and other results suggest that the quality of the results of an approx-
imation algorithm is highly dependent on the solved instances. Furthermore these
three algorithms do not cover all possible approximation algorithms of this type.

6.2 Machine planning

Simplified model 2 plans end items and recall that for ASML the end items are ma-
chines. As argued in chapter 4.2 it makes sense to see the FSSN and the CSSN of a
machine together as one end item. The planning of the machines is the most difficult
and the most important part of the supply chain planning. The idea of the simplified
models was to first plan the end items and after this the rest of the supply chain. It
could happen that some parts in the supply chain can not be in time in order to be
able to produce the end items as planned. In this case it should be possible to add a
constraint to the model to make sure these end items are planned later and solve the
model again to get a new planning for the end items. Repeating this steps generates
an iterative process and this process should give a good feasible solution in the end.

Note that this process does not guarantee to find the optimal solution. Let A and
B be two end items, C a component of B and let all items have lead time 1. Suppose
that the end items both require 1 unit of the same resource group, of which 2 units
are available in each time slot, while C requires 1 unit of another resource group, of
which 1 unit is available in each time slot. Let the costs of having item A on stock for
one time slot be 4, while these costs are 3 for item B and 2 for item C. Suppose there
is a demand of size 2 for both end items at the end of time slot 4. The optimal release
plan is the release one C at time slot 1, one A, one B and one C at time slot 2 and one
A and one B at time slot 3. The stock in this case is one A and one B during time slot
4 and thus the total costs are 7. The release plan, by planning the end items first and
the rest later, releases one C at time slot 0 and 1, two B’s at time slot 2 and two A’s at
time slot 3. The stock in this case one C during time slot 2 and two B’s during time
slot 4 and thus the total costs are 8.

To plan the end items first Simplified model 2 is used since this model is, unlike
Simplified model 1, faster than the Extended model. Note that it should be decided
with which sets Simplified model 2 should be used. It is possible to translate a solution
of Simplified model 2 into a release plan like Alternative model 1 produces, by using
the equality \( R_{itw} = \sum_{y=1}^{Y_i} X_{ytwi} \). This way Algorithm 4 can be adjusted to solve Simp-
lified model 2. After solving Simplified model 2, the rest of the supply chain should be
planned.

In order to plan the rest of the supply chain, the Extended model could be used
as follows. Create an instance equal to the instance that should be solved, without
the end items, but with demand on the components of the end items that matches
the solution of Simplified model 2: \( D_{it} = \sum_{j=1}^{N_i} \sum_{w=0}^{W_i} h_{ij} \sum_{y=1}^{Y_j} X_{ytwj} \). Also, the costs of
backlog on the components of the end item should be equal to the costs of backlog on
the end item. Solve this instance using the Extended model. This should not take too
much time, since the most difficult of the planning, the machines, is excluded. The
optimal solution minimizes the backlog, so there is only backlog if it is really impossible
to produce all components in time or in some other rare cases. If there is no backlog
in the optimal solution the original problem is solved, since there are release orders
for all items and all components are in time. The total costs will be the objective value
of Simplified model 2 plus the objective value of the Extended model.

If, however, there is a backlog of size $B_{it}$ on some component $i$ in time slot $t$ it is not
possible to release $B_{it}$ machines $j$ that require component $i$ in time slots strictly before
$t$. To prevent releasing these machines in those time slots the following constraint
can be added to Simplified model 2: $\sum_{s}^{t-1} \sum_{j=1}^{n} \sum_{w=0}^{h_{ij}} \sum_{y=1}^{\lambda_{yj}} X_{yswj} \leq -B_{it} + \sum_{s}^{t-1} D_{is}$. If these are added for all $i$ and $t$ such that $B_{it} > 0$, Simplified model 2 can be solved
again. Then, it should be checked if the rest of the supply chain can be planned.
This way the process is repeated until there is no backlog in the Extended model. It
depends on the instance how many times this process should be repeated, how long
each iteration takes and how far the final solution is away from the optimal solution.
7 Discussion

7.1 Conclusion

Several methods to find (near) optimal solutions of the Extended model were presented. A few of these seem to be faster than solving the Extended model.

In chapter 4, three alternatives for the Extended model are presented. Alternative model 1 and Simplified model 2 use sets-based capacity constraints. An advantage of these capacity constraints is that they are knapsack constraints, however there are exponentially many of them. Because of this, the idea is to use these models without all constraints. It turns out that with the right set of constraints Alternative model 1 and Simplified model 2 are faster than the Extended model. Simplified model 1 and Simplified model 2 focus on the planning of end items and use an alternative for the inventory constraints. Simplified model 1 does not plan end items faster than the Extended model.

A maximum flow graph can be used to check if there is enough capacity to produce all materials according to a release plan. This graph can also be used to find a subset of time slots, which belongs to a capacity constraint in Alternative model 1 that is violated by a release plan. This idea is used in chapter 5 to develop Algorithms 4 and Algorithm 5, which start with Alternative model 1 without all capacity constraints and slowly add missing capacity constraints while solving the model until the optimal solution is found. They first check for missing constraints for solutions of the LP-relaxation and later for feasible integer solution encountered during the branch and bound process. Algorithm 4 turns out to be faster than solving the Extended model, while the difference in running time between Algorithm 5 and the Extended model is small.

In chapter 6 some methods to find near optimal solutions are presented. It is hard to say which one is the best and if this one is better than giving the Extended model to a solver and let the solver run for a specific amount of time.

7.2 Further research

This thesis leaves plenty of room for further research. Some of the described methods can be improved or other methods can be developed. It is important to think about what kind of instances should be solved and what kind of solution are desired, before doing any further research. Some ideas that can be used to improve the given methods or to design new methods are listed below.

If optimal solutions are desired, trying to improve Algorithm 4 or Algorithm 5 is an option. For example by using the branch and cut technique, which could be used to search if there are constraints violated by the solutions of the linear subproblems, if these are less than 5 branches deep in the branch and bound tree. Another option is also adding constraints for some of the subsets of the \( \sigma_u \)'s found by these algorithms. These are often violated by this or upcoming solutions. Another option is adding heuristic methods to find upper bounds to the branch and bound process. During this process a lot of LP-relaxations are solved and there are several ways to find an integer solution based on the solution of an LP-relaxation. If a very fast heuristic method does
this every time a LP-relaxation is solved, there is a chance that it finds a good integer solution.

Another option to find optimal solutions could be a variation on the machine planning described in chapter 6.2, that takes the costs of the rest of the supply chain into account while making the machine planning. For this, it might be possible to use Bender’s decomposition.

If near optimal solutions are desired the machine planning described in chapter 6.2 could be used. It is not tested yet and there are several ways to improve it. Like reusing the $\sigma_u$’s each time Simplified method 2 is solved. Or checking for intermediate solutions of the Extended model used in this method, if they include backlog. Other options are using other models than Simplified model 2 for the machine planning step and the Extended model for step in which the rest of the supply chain is planned.

Besides looking for faster algorithms, further research could focus on extending the model or dealing with uncertainty in the model. There is for example a lot of uncertainty in the demand, since it is a forecast. It would be interesting to be able to create release plans that have low costs for different demand scenarios.
References
