Integer programming models for mid-term production planning for high-tech low-volume supply chains

Citation for published version (APA):

Document license:
CC BY-NC-ND

DOI:
10.1016/j.ejor.2018.02.049

Document status and date:
Published: 16/09/2018

Document Version:
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Integer Programming Models for Mid-term Production Planning for High-tech Low-volume Supply Chains

Joost T. de Kruijff\textsuperscript{a,}*, Cor A. J. Hurkens\textsuperscript{b}, Ton G. de Kok\textsuperscript{a}

\textsuperscript{a}Eindhoven University of Technology, Department of Industrial Engineering and Innovation Sciences, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands

\textsuperscript{b}Eindhoven University of Technology, Department of Mathematics and Computer Science, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands

Abstract

This paper studies the mid-term production planning of high-tech low-volume industries. Mid-term production planning (6 to 24 months) allocates the capacity of production resources to different products over time and coordinates the associated inventories and material inputs so that known or predicted demand is met in the best possible manner. High-tech low-volume industries can be characterized by the limited production quantities and the complexity of the supply chain. To model this, we introduce a mixed integer linear programming model that can handle general supply chains and production processes that require multiple resources. Furthermore, it supports semi-flexible capacity constraints and multiple production modes.

Because of the integer production variables, size of realistic instances and complexity of the model, this model is not easily solved by a commercial solver. Applying Benders’ decomposition results in alternative capacity constraints and a second formulation of the problem. Where the first formulation assigns resources explicitly to release orders, the second formulation assures that the available capacity in any subset of the planning horizon is sufficient. Since the number of alternative capacity constraints is exponential, we first solve the second formulation without capacity constraints. Each time an incumbent is found during the branch and bound process a maximum flow problem is used to find missing constraints. If a missing constraint is found it is added and the branch and bound process is restarted. Results from a realistic test case show that utilizing this algorithm to solve the second formulation is significantly faster than solving the first formulation.

Keywords: Production, Integer Programming, Branch and Bound

*Corresponding author

Email addresses: j.t.d.kruijff@tue.nl (Joost T. de Kruijff), c.a.j.hurkens@tue.nl (Cor A. J. Hurkens), a.g.d.kok@tue.nl (Ton G. de Kok)
1. Introduction

In this paper we consider the mid-term production planning of high-tech low-volume industries. Mid-term production planning allocates the capacity of production resources, e.g. machines, specialized work force, tools and space, to different products over time and coordinates the associated inventories and material inputs so that known or predicted demand is met in the best possible manner (Missbauer & Uzsoy (2011)). The planning horizon ranges from 6 to 24 months, which enables the consideration of seasonal patterns (Fleischmann et al. (2015)). Mid-term production planning is of great importance for any manufacturing company, since it synchronizes the flow of materials along the entire supply chain, which results in reduced inventory levels, thereby contributing to higher returns of investment for the company and its suppliers (Albrecht et al. (2015)). More specifically, we study the mathematical programming models that arise when a rolling scheduling approach is applied, i.e. when each period a deterministic model is solved and the immediate decisions are implemented (c.f. Silver et al. (1998) or de Kok & Fransoo (2003)). This is by far the most common approach in practice and this has motivated the extensive literature on planning and scheduling. The determination of the exogenous parameters that enable to cope with uncertainty, such as safety stocks and nominal lead times, are outside the scope of this paper.

There is a vast body of literature on mid-term production planning (c.f. Bertrand et al. (1990) or Albrecht et al. (2015)), in which this is called the goods flow control process or the master planning process. However, the majority studies high-volume industries while high-tech low-volume industries (e.g. machine building and aerospace) are not adequately represented (Stadtler (2005b)). Therefore, the focus of this paper is on high-tech low-volume industries, which can be characterized by the limited production quantities and the complexity of the supply chains. Note that this paper is inspired by a real-life application from a company that produces semiconductor equipment, i.e. very complex machines.

The models for high-volume production planning use continuous production quantity variables (c.f. Buschkühl et al. (2010) or Missbauer & Uzsoy (2011)). The key difference between high-volume and low-volume production planning is that for the former the rounding of these variables is insignificant, while for the latter this is clearly not the case. This is crucial since it causes the mid-term production planning problem for high-tech low-volume industries to be discrete and, considering the similarities between choosing which items to produce using the available capacity and the knapsack problem, even NP-hard (see Appendix). However, in practice mid-term planning is done every week or month and before a decision is made the problem is solved for many different demand scenarios. Hence, the time required to solve an instance is of great importance.

Two papers that explore low-volume production planning are Kolisch (2000) and Stadtler (2005a), which propose to apply project scheduling. However, taking into account the features
of our real-life application, we propose to extend a model from the high-volume production planning literature. Because of the NP-hardness of the problem and the importance of fast solutions, we apply Benders’ decomposition (Benders (1962)). This results in alternative capacity constraints and a second formulation of the problem. We compare the tractability of both formulations on a realistic test case. Subsequently, we compare our formulations with a model in Spitter et al. (2005), which describes a similar problem.

Mid-term production planning issues work orders and thereby allocates materials and capacity, such that demand is met in the best possible manner (c.f. de Kok & Fransoo (2003)). In other words, given known or forecasted demand the objective is to minimize the costs of inventory and backlog over a finite horizon subject to material availability and capacity constraints. The costs of backlog and inventory are linear in time and size. Backlog of an item can only exist if there is external demand for that item. The time between the release and the due date specified by a work order is called the planned lead time and provides the lower planning level with the freedom to control the detailed scheduling (c.f. Spitter (2005) or Jansen (2012)). Material availability constraints ensure that work orders can only be released if the required components are available at the right moment.

The capacity constraints ensure that work orders can only be released if the required resources are available during the production processes. Deviating from the current literature, in which capacity is claimed at fixed offsets (e.g. Jans & Degraeve (2008)) or capacity can be claimed anywhere within the planned lead time (e.g. Spitter et al. (2005)), we introduce semi-flexible capacity constraints, which limit the size of the capacity claims per time slot. Compared to capacity claims at fixed offsets this provides extra planning flexibility. Since the production volumes are small and the lead times are long, this is very beneficial when the capacity is limiting. However, these constraints can also ensure that certain resources are claimed in specific time slots and thus preserve the order in the production process. This is crucial for high-tech low-volume industries since there the production of one item often entails multiple time consuming tasks and the order in which these are executed is critical. These tasks for example consist of testing functionality. Note that Naber & Kolisch (2014) introduce similar capacity constraints to a project scheduling problem. One important difference is that their limits on the size of the capacity claims are constant during lead time, while these limits may vary per time slot in our case.

Besides the semi-flexible capacity constraints we adopt another source of planning flexibility: alternative modes of production (e.g. Voß & Woodruff (2006) or Weglarz et al. (2011)). These production modes might differ in lead time, resource requirements and assembly sequences. In high-tech low-volume industries lead times are often long and production processes require specialized manpower. Planning around periods of lower resources availability (e.g. vacations) thus implies
sizable inventory and backlog costs. Using production modes with longer lead times and thus more flexible resource requirements, reduces these costs significantly. Similarly, in case of a material shortage an assembly sequence in which the missing component is required later saves a lot of time.

The rest of this paper is structured as follows. In the next section we further specify the considered production structure and introduce the notation used. In Section 3 we introduce an integer linear programming model of the problem. In Section 4 the application of Benders’ decomposition results in alternative capacity constraints and a second formulation of the problem. Section 5 proves the equivalence of the formulations described in Section 3 and 4. Section 6 describes an algorithm that solves the second formulation. In Section 7 both our formulations are compared against a set of test cases. In Section 8 we compare our model with the model of Spitter et al. (2005). Section 9 summarizes our findings and suggests further research.

2. Notation

We extend the model with balance equations of Spitter et al. (2005) by introducing production modes, limiting the flexibility in the capacity constraints and enabling material claims during lead time. We consider a supply chain consisting of \( n \) items. For each item \( i \) we define \( M_i \) as the set of production modes that can be used to produce \( i \). The planned lead time for the production of item \( i \) using mode \( m \) is \( \tau_{im} \). To produce one item \( j \) in mode \( m \), \( h_{ij} \) items \( i \) are required \( \delta_{ijm} \) time slots after the release.

We consider a planning horizon of \( T \) time slots \( s \) defined as \((s - 1, s]\). \( D_{it} \) and \( G_{it} \) represent the independent (exogenous) and dependent (endogenous) demand for item \( i \) at time \( t \) respectively. \( I_{it} \) and \( B_{it} \) represent the inventory during time interval \((t, t+1)\) and the backlog at time \( t \) for item \( i \) respectively. \( \alpha_{it} \) and \( \beta_{it} \) are the costs of these, i.e. \( \alpha_{it} \) is the cost of having a single unit of item \( i \) on inventory during \((t, t+1)\) and \( \beta_{it} \) is the cost of having a backlog of one unit of item \( i \) at time \( t \). Besides \( G_{it} \), \( I_{it} \) and \( B_{it} \), \( R_{itm} \) is a crucial variable. It represents the size of the work order of item \( i \) released at time \( t \) with production mode \( m \). Since items could have a lead time of multiple time slots, items could be halfway production at the start of the planning horizon. Therefore, we introduce \( \overline{R}_{itm} \) for work order releases in the past. The additional cost of releasing the work order for an item \( i \) in production mode \( m \) at time \( t \) is \( \gamma_{itm} \).

We consider \( k \) different resources. The available capacity of resource \( u \) during time slot \( s \) is \( c_{us} \). For the production of one item \( i \) in mode \( m \) in total \( p_{iu}^{\text{tot}} \) of resource \( u \) is required. In time slot \( q \) of the production of an item \( i \) in mode \( m \) at least \( p_{iquum}^{\text{min}} \) of resource \( u \) is required and at most \( p_{iquum}^{\text{max}} \) of resource \( u \) can be claimed for the production of this item. Like the model with balance equations in Spitter et al. (2005), the first formulation of our model explicitly assigns resources to releases. In other words, the variable \( Z_{itsum} \) determines how much of resource \( u \) in time slot \( s \) should be
used for the production of the work order for item $i$ that is released at time $t$ with release mode $m$.

To summarize we define the following parameters:

- $n$: $n \in \mathbb{N}$, the number of different items, which are labeled $i = 1, \ldots, n$.
- $k$: $k \in \mathbb{N}$, the number of resources, which are labeled $u = 1, \ldots, k$.
- $M_i$: $M_i \subset \mathbb{N}$, the set of production modes for item $i$, $i = 1, \ldots, n$.
- $\tau_{im}$: $\tau_{im} \in \mathbb{N}$, $i = 1, \ldots, n$, the planned lead time for the production of item $i$ when production mode $m$ is used.
- $T$: $T \in \mathbb{N}$, the length of the planning horizon, which includes decision points $t = 0, \ldots, T$ and time slots $s$, defined as $(s - 1, s]$, $s = 1, \ldots, T$. The past decision points $t = -\tau_{im}, \ldots, -1$ and time slots $s = -\tau_{im} + 1, \ldots, 0$ are also considered.
- $D_{it}$: $D_{it} \in \mathbb{N}$, the independent demand for item $i$ at time $t$, $i = 1, \ldots, n$, $t = 0, \ldots, T$.
- $h_{ij}$: $h_{ij} \in \mathbb{N}$, the number of units of item $i$ needed for the production of a single unit of item $j$, $i = 1, \ldots, n$, $j = 1, \ldots, n$.
- $\delta_{ijm}$: $\delta_{ijm} \in \mathbb{N}$, the number of time slots after the release moment of item $j$, component $i$ is required for production of $j$, in production mode $m$, $i = 1, \ldots, n$, $j = 1, \ldots, n$, $m \in M_j$.
- $\alpha_{it}$: $\alpha_{it} > 0$, the costs of keeping one single unit of item $i$ on stock during $(t, t + 1)$, $i = 1, \ldots, n$, $t = 0, \ldots, T$.
- $\beta_{it}$: $\beta_{it} > 0$, the costs of delaying the delivery of one single unit of item $i$ from time $t$ to time $t + 1$, $i = 1, \ldots, n$, $t = 0, \ldots, T$.
- $\gamma_{itm}$: $\gamma_{itm} \geq 0$, the extra costs occurring from releasing one single unit of item $i$ at time $t$ with production mode $m$, $i = 1, \ldots, n$, $t = 1, \ldots, T - \tau_{im}$, $m \in M_i$.
- $R_{im}$: The set of resources that are used to produce item $i$ in mode $m$, $i = 1, \ldots, n$, $m \in M_i$.
- $J_u$: The set of items and their production modes $(i, m)$ that require resource group $u$ during production, $u = 1, \ldots, k$.
- $c_{us}$: $c_{us} \geq 0$, the maximum amount available of resource group $u$ in time slot $s$, $u = 1, \ldots, k$, $s = 1, \ldots, T$.
- $p_{iqum}^{\min}$: $p_{iqum}^{\min} \geq 0$, the minimum amount of resource $u$ required in time slot $t + q$ for the production order of one item $i$ released at time $t$ with production mode $m$, $i = 1, \ldots, n$, $m \in M_i$, $q = 1, \ldots, \tau_{im}$, $u \in R_{im}$.
- $p_{iqum}^{\max}$: $p_{iqum}^{\max} \geq 0$, the maximum amount of resource $u$ that can be allocated in time slot $t + q$ to the production order of one item $i$ released at time $t$ with production mode $m$, $i = 1, \ldots, n$, $m \in M_i$, $q = 1, \ldots, \tau_{im}$, $u \in R_{im}$.
- $p_{ium}^{\text{tot}}$: $p_{ium}^{\text{tot}} \geq 0$, the total amount of resource $u$ required to produce one item $i$ released with production mode $m$, $i = 1, \ldots, n$, $m \in M_i$, $u \in R_{im}$.
\( \bar{R}_{itm} \): \( R_{itm} \in \mathbb{N} \), the size of the work order of item \( i \) released in the past at time \( t \) with production mode \( m \), \( i = 1, \ldots, n \), \( m \in M_i \), \( t = -\tau_{im}, \ldots, -1 \).

\( T_{i,-1} \): \( T_{i,-1} \in \mathbb{N} \), the inventory level of item \( i \) at time \( -1 \), \( i = 1, \ldots, n \).

\( \bar{B}_{i,-1} \): \( B_{i,-1} \in \mathbb{N} \), the size of the backlog of item \( i \) at time \( -1 \), \( i = 1, \ldots, n \).

\( Z_{itsum} \): \( \bar{Z}_{itsum} \geq 0 \), the part of the planned work, related to the work order released in the past, \( \bar{R}_{itm} \), executed before the start of the planning horizon in time slot \( s \) on resource group \( u \), \( i = 1, \ldots, n \), \( m \in M_i \), \( t = -\tau_{im}, \ldots, T - \tau_{im} \), \( s = t + 1, \ldots, T \), \( u \in \mathcal{R}_{im} \).

Furthermore, the decision variables are:

\( I_{it} \): The inventory of item \( i \) during time interval \((t, t+1)\), \( i = 1, \ldots, n \), \( t = 0, \ldots, T \).

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

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

\( R_{itm} \): The size of the work order of item \( i \) released at time \( t \) with production mode \( m \), \( i = 1, \ldots, n \), \( m \in M_i \), \( t = -\tau_{im}, \ldots, T - \tau_{im} \), \( m \in M_i \).

\( Z_{itsum} \): The part of the planned work, related to work order release \( R_{itm} \), executed in time slot \( s \) on resource group \( u \), \( i = 1, \ldots, n \), \( m \in M_i \), \( t = -\tau_{im}, \ldots, T - \tau_{im} \), \( s = t + 1, \ldots, t + \tau_{im} \), \( u \in \mathcal{R}_{im} \).

Note that we are considering low-volume production, hence \( R_{itm} \) is an integer variable.

### 3. A mixed integer linear programming model

This model is based on the Extended model in Gort (2013) and explicitly assigns resources to releases. If the assignment of resources to releases is known, checking the constraints on the resource usage is straightforward. Given the notation introduced in the previous section, the mixed integer linear programming formulation, which we will call Formulation 1, is as follows:

\[
\min T \sum_{t=0}^{T} \sum_{i=1}^{n} \alpha_{it} I_{it} + T \sum_{t=0}^{T} \sum_{i=1}^{n} \beta_{it} B_{it} + \sum_{i=1}^{n} \sum_{m \in M_i} \sum_{t=0}^{T-\tau_{im}} \gamma_{itm} R_{itm}
\]

subject to

\[
I_{it} = I_{i,t-1} + \sum_{m \in M_i} R_{i,t-\tau_{im},m} - D_{it} - G_{it} + B_{it} - B_{i,t-1} \quad i = 1, \ldots, n, t = 0, \ldots, T \quad (1)
\]

\[
G_{it} = \sum_{j=1}^{n} \sum_{m \in M_j} h_{ij} R_{j,t-\delta_{ijm},m} \quad i = 1, \ldots, n, t = 0, \ldots, T \quad (2)
\]

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

\[
\bar{R}_{i,s-t,u,m} \leq Z_{itsum} \leq \bar{p}_{i,s-t,u,m} R_{itm} \quad i = 1, \ldots, n, m \in M_i, u \in \mathcal{R}_{im}, \quad t = -\tau_{im}, \ldots, T - \tau_{im}, \quad (4)
\]
The objective minimizes the costs of holding finished items on inventory, the costs of delivering products too late and the extra costs involved with using different production modes. Constraint (1) is the inventory balance equation, which models the flow of goods and enforces the satisfaction of demand. Constraint (2) determines the dependent demand. Constraint (3) ensures that backlog is only used for external demand. Constraint (4) ensures that resources assignments are between their upper and lower bounds. Constraint (5) enforces that in total enough resources are assigned to the production of an item. Constraint (6) ensures that the amount of work assigned to a resource in a time slot is not bigger than the capacity.

4. Alternative capacity constraints

Preliminary test results showed that solving Formulation 1 with a standard commercial solver is very time consuming for realistic instances. Therefore, we will present an alternative approach to solve this problem based on Benders’ decomposition (Benders (1962)).
The \( Z_{itsum} \) variables are only used to make sure enough resources are available for the planned production. Thus, indirectly, the constraints on \( Z_{itsum} \) are constraints on the releases \( R_{itm} \). The number of these \( Z_{itsum} \) variables, the number of constraints on these \( Z_{itsum} \) variables and the fact that these constraints are indirect constraints on the integer variables are complicating factors. Note that the problem is fairly easy to solve without capacity constraints. Furthermore, note that the problem is easy to solve if all variables are fixed except the \( Z_{itsum} \) variables, since they are continuous variables. Therefore, this problem might be suitable to solve using Benders’ decomposition (Benders (1962)). In the rest of the paper we will show that this is the case.

To apply Benders’ decomposition we split our variables into \((R_{itm}, I_{it}, B_{it}, G_{it})\) and \((Z_{itsum})\). Constraints (1), (2) and (3) only contain variables from \((R_{itm}, I_{it}, B_{it}, G_{it})\), while constraints (4), (5) and (6) also contain variables from \((Z_{itsum})\). Note that the objective function does not depend on \( Z_{itsum} \). Then the subproblem is:

\[
\begin{align*}
\text{min} & \quad 0 \\
\text{subject to} & \\
\sum_{s = t+1}^{t+\tau_{im}} Z_{itsum} = p_{it}^\text{tot} R_{itm} & \quad i = 1, \ldots, n, m \in M_i, u \in \mathcal{R}_i, \quad (7) \\
Z_{itsum} \geq p_{i,s-t,u,m}^\text{min} R_{itm} & \quad i = 1, \ldots, n, m \in M_i, u \in \mathcal{R}_i, \quad t = -\tau_{im}, \ldots, T - \tau_{im} \quad (8) \\
Z_{itsum} \leq p_{i,s-t,u,m}^\text{max} R_{itm} & \quad i = 1, \ldots, n, m \in M_i, u \in \mathcal{R}_i, \quad t = -\tau_{im}, \ldots, T - \tau_{im}, s = t + 1, \ldots, t + \tau_{im} \quad (9) \\
\sum_{i=1}^{n} \sum_{m \in M_i} \sum_{t=s-\tau_{im}}^{t+\tau_{im}} Z_{itsum} \leq c_{us} & \quad u = 1, \ldots, k, s = 1, \ldots, T \quad (10)
\end{align*}
\]

Again, the objective function does not depend on \( Z_{itsum} \). Hence, the subproblem will only generate feasibility cuts and no optimality cuts. Note that since \( p_{itsum}^\text{min} \geq 0 \) constraint (8) implies \( Z_{itsum} \geq 0 \). If we introduce dual variables \( \pi_{itum}^\text{tot}, \pi_{itum}^\text{min}, \pi_{itum}^\text{max} \) and \( \pi_{us}^\text{cap} \) for constraints (7), (8), (9) and (10) respectively, the dual of the subproblem is:

\[
\begin{align*}
\max & \quad \sum_{(i,t,u,m)} R_{itm} p_{itum}^\text{tot} + \sum_{(i,t,s,u,m)} (p_{i,s-t,u,m}^\text{min} R_{itm} p_{itum}^\text{min} + p_{i,s-t,u,m}^\text{max} R_{itm} p_{itum}^\text{max}) + \sum_{(u,s)} c_{us} \pi_{us}^\text{cap} \\
\text{subject to} & \\
\pi_{itum}^\text{tot} + \pi_{itum}^\text{min} + \pi_{itum}^\text{max} + \pi_{us}^\text{cap} = 0 & \quad i = 1, \ldots, n, m \in M_i, u \in \mathcal{R}_i
\end{align*}
\]
\[ \begin{align*} 
\pi_{\text{itsum}}^{\text{min}} &\geq 0 
\pi_{\text{itsum}}^{\text{max}} &\leq 0 
\pi_{\text{us}}^{\text{cap}} &\leq 0 
\end{align*} \]

Each extreme ray \((\pi_{\text{itum}}^{\text{tot}}, \pi_{\text{itsum}}^{\text{min}}, \pi_{\text{itsum}}^{\text{max}}, \pi_{\text{us}}^{\text{cap}})\) of the feasible region of the dual gives the following constraint for the master problem:

\[ \sum_{(i,t,u,m)} R_{itm} p_{itum} \pi_{\text{itum}}^{\text{tot}} + \sum_{(i,t,u,m)} (p_{i,s-t,u,m}^{\text{min}} R_{itm} \pi_{\text{itsum}}^{\text{min}} + p_{i,s-t,u,m}^{\text{max}} R_{itm} \pi_{\text{itsum}}^{\text{max}}) + \sum_{(u,s)} c_{us} \pi_{\text{us}}^{\text{cap}} \leq 0 \]

Given the limitations on the coefficients in this constraint, constraints corresponding to some extreme rays are dominated by others. We will now classify the extreme rays for which the constraints are not dominated. Note that the dual problem decomposes over the resources, so we will fix \(u\).

It is clear that at least one of \(\pi_{\text{itum}}^{\text{tot}}\) and \(\pi_{\text{itsum}}^{\text{min}}\) should be positive to get a positive left hand side. Thus, because of constraint (11), at least one of \(\pi_{\text{itum}}^{\text{tot}}, \pi_{\text{itsum}}^{\text{max}}\) and \(\pi_{\text{us}}^{\text{cap}}\) should be negative. Since \(p_{i,t,u,m}^{\text{max}} \geq p_{i,t,u,m}^{\text{min}}, \pi_{\text{itsum}}^{\text{max}}\) will not be negative if \(\pi_{\text{itsum}}^{\text{min}}\) is positive. Since \(\sum_{q} p_{i,q,u,m}^{\text{min}} \geq p_{i,t,u,m}^{\text{tot}}, \pi_{\text{itsum}}^{\text{max}}\) will not be negative for all \(s\) if \(\pi_{\text{itum}}^{\text{tot}}\) is positive. Since \(\sum_{q} p_{i,q,u,m}^{\text{min}} \leq p_{i,t,u,m}^{\text{tot}}, \pi_{\text{itsum}}^{\text{min}}\) will not be negative if \(\pi_{\text{itsum}}^{\text{min}}\) is positive. Therefore, \(\pi_{\text{us}}^{\text{cap}}\) will be negative for at least one time slot \(s\).

Define \(\sigma_u\) as the set of all time slots \(s\) for which \(\pi_{\text{us}}^{\text{cap}}\) is negative. Note that each point in the feasible region of the dual for which \(\pi_{\text{us}}^{\text{cap}}\) takes different values in \(\sigma_u\), can be written as the sum of a feasible point for which \(\pi_{\text{us}}^{\text{cap}}\) is constant in \(\sigma_u\) and another feasible point. It is quite straightforward to show that a point \(\pi\) with \(\pi_{\text{us}}^{\text{cap}} = \max_{s \in \sigma_u} \pi_{\text{us}}^{\text{cap}}\) for all \(s \in \sigma_u\), \(\pi_{\text{itum}}^{\text{tot}} = \min(-\max_{s \in \sigma_u} \pi_{\text{us}}^{\text{cap}}, \pi_{\text{itum}}^{\text{tot}}), \pi_{\text{itsum}}^{\text{max}} = -\pi_{\text{itum}}^{\text{tot}}\) for all \(s \notin \sigma_u\), \(\pi_{\text{itsum}}^{\text{min}} = -\pi_{\text{us}}^{\text{cap}} - \pi_{\text{itum}}^{\text{tot}}\) for all \(s \in \sigma_u\) and all other dual variables 0, is suitable for this. Hence, an extreme ray has \(\pi_{\text{us}}^{\text{cap}}\) constant over all \(s \in \sigma_u\).

For each \((i,t,m)\) such that \(u \in \mathcal{R}_im\) and there exists a \(s \in \sigma_u\) such that \(s > t\) and \(s \leq t + \tau_{im}\) there are two options: \(\sum_{s \in \sigma_u} p_{i,s-t,u,m}^{\text{min}} \geq p_{i,t,u,m}^{\text{tot}} - \sum_{s \notin \sigma_u} p_{i,s-t,u,m}^{\text{max}}\) or \(\sum_{s \in \sigma_u} p_{i,s-t,u,m}^{\text{min}} < p_{i,t,u,m}^{\text{tot}} - \sum_{s \notin \sigma_u} p_{i,s-t,u,m}^{\text{max}}\). In the first case the strongest constraint corresponds to the extreme ray with \(\pi_{\text{itsum}}^{\text{min}} = -\pi_{\text{us}}^{\text{cap}} > 0\) for \(s \in \sigma_u\) and the other dual variables zero for this \((i,t,m)\). In the second case the strongest constraint corresponds to the extreme ray with \(\pi_{\text{itum}}^{\text{tot}} = -\pi_{\text{us}}^{\text{cap}} > 0\) with \(s \in \sigma_u\), \(\pi_{\text{itsum}}^{\text{max}} = -\pi_{\text{itum}}^{\text{tot}} < 0\) for \(s \notin \sigma_u\) and the other dual variables zero for this \((i,t,m)\). This implies that all extreme rays for which the constraint is not dominated give constraints of the following form:

\[ \sum_{(i,t,m)} 
\chi_{itmu} \sigma_u R_{itm} \leq \sum_{s \in \sigma_u} c_{us} \]
where, for $t \geq 0$:

$$\chi_{itm\sigma_u} = \max \left( \sum_{q:t+q \in \sigma_u} p_{iqum}^{\min}, p_{ium}^{\text{tot}} - \sum_{q:t+q \notin \sigma_u} p_{iqum}^{\max} \right).$$

$\chi_{itm\sigma_u}$ can be interpreted as the minimal amount of resource $u$ required in a set of time slots, $\sigma_u$, for the release of one item $i$ at time $t$ with production mode $m$. Figure 1 depicts three examples of the determination of $\chi_{itm\sigma_u}$. The first example releases an item $i$ at time 1 with production mode 0, for which the lead time is 2. It requires exactly 10 units of resource $u$ in time slots 2 and 3. Hence $\chi_{i,1,0,u,\sigma_u} = 20$. The second example releases the same item $i$ at time 1 using production mode 1, which has lead time 3. It requires in total 20, and in time slots 2, 3 and 4 at least 4 and at most 10 units of resource $u$. Thus at most 10 units of resource $u$ can be claimed outside $\sigma_u$. Hence $\chi_{i,1,1,u,\sigma_u} = 20 - 10 = 10$. The third example releases the same item $i$ at time 2 with with production mode 1, which has lead time 3. It requires in total 20, and in time slots 3, 4 and 5 at least 4 and at most 10 units of resource $u$. Since it is possible to claim 16 units in time slots 4 and 5, only the required 4 units have to be claimed in time slot 3. Thus $\chi_{i,2,1,u,\sigma_u} = 4$. For $t < 0$, $\chi_{itm\sigma_u}$ should be determined based on the progress of the production process, i.e. the total amount of resources required, $p_{ium}^{\text{tot}}$ in the above formula, should be adjusted to the amount of work that still has to be carried out. Thus, for $t < 0$,

$$\chi_{itm\sigma_u} = \max \left( \sum_{q:t+q \in \sigma_u} p_{iqum}^{\min}, p_{ium}^{\text{tot}} - \sum_{s=t+1}^{0} Z_{itsum} - \sum_{q:t+q \notin \sigma_u \land t+q > 0} p_{iqum}^{\max} \right).$$

So far in this section, we found alterative capacity constraints using Benders’ decomposition. They compare the available resources in a set of time slots with the minimum required amount of resources. Note that the same capacity constraints were found in de Kruijff (2014), where the inspiration was Hall’s marriage theorem (Hall (1935)). Note furthermore that these constraints are knapsack constraints, which modern solvers use to eliminate non-integer solutions of the LP relaxation to find higher lower bounds faster. If we replace the capacity constraints in our model with these alternative capacity constraints for all resources $u$ and all subsets of time slots $\sigma_u$, we get an alternative formulation of the model, which we call Formulation 2. For this formulation the following parameters are used as input:

- $\sigma_u$: 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 collection of sets $\sigma_u$ of time slots in $[1, T]$, used to make sure that there is enough capacity of resource group $u$, $u = 1, \ldots, k$.
- $\chi_{itm\sigma_u}$: The minimal amount of resource $u$, required in $\sigma_u$ by the release of one item $i$ at time $t$ with production mode $m$, $i = 1, \ldots, n$, $m \in M_i$, $t = -\tau_{im}, \ldots, T - \tau_{im}$, $u \in R_{im}$, $\sigma_u \in \Sigma_u$. 

10
Figure 1: Examples of the calculation of $\chi_{i,t,\sigma_u}$, with $\sigma_u = \{1, 2, 3\}$ and $p_{i,um}^{\text{tot}} = 20$, $p_{i,qu}^{\text{max}} = 10$ and $p_{i,qu}^{\text{min}} = 10$ or $p_{i,qu}^{\text{min}} = 4$ for each time slot $q$. 

$R_{i,1,0}$

$p_{i,um}^{\text{tot}} = 20$

$p_{i,qu}^{\text{min}} = 10$

$p_{i,qu}^{\text{max}} = 10$

$\chi_{i,1,0,u,\sigma_u} = 20$

$R_{i,1,1}$

$p_{i,um}^{\text{tot}} = 20$

$p_{i,qu}^{\text{min}} = 4$

$p_{i,qu}^{\text{max}} = 10$

$\chi_{i,1,1,u,\sigma_u} = 10$

$R_{i,2,1}$

$p_{i,um}^{\text{tot}} = 20$

$p_{i,qu}^{\text{min}} = 4$

$p_{i,qu}^{\text{max}} = 10$

$\chi_{i,2,1,u,\sigma_u} = 4$
Then, the second mixed integer linear programming formulation of our model is:

\[
\min \sum_{t=0}^{T} \sum_{i=1}^{n} \alpha_{it} I_{it} + \sum_{t=0}^{T} \sum_{i=1}^{n} \beta_{it} B_{it} + \sum_{i=1}^{n} \sum_{m \in M_i} \sum_{t=0}^{T-\tau_{im}} \gamma_{itm} R_{itm}
\]

subject to

\[
I_{it} = I_{i,t-1} + \sum_{m \in M_i} R_{i,t-\tau_{im},m} - D_{it} - G_{it} + B_{it} - B_{i,t-1} \quad i = 1, \ldots, n, t = 0, \ldots, T \quad (12)
\]

\[
G_{it} = \sum_{j=1}^{n} \sum_{m \in M_j} h_{ij} R_{j,t-b_{ijm},m} \quad i = 1, \ldots, n, t = 0, \ldots, T \quad (13)
\]

\[
B_{it} - B_{i,t-1} \leq D_{it} \quad i = 1, \ldots, n, t = 0, \ldots, T \quad (14)
\]

\[
\sum_{(i,m) \in I_u : [t+1,t+\tau_{im}] \cap \sigma_u \neq \emptyset} \chi_{itm} \sigma_u R_{itm} \leq \sum_{s \in \sigma_u} c_{us} \quad \forall \sigma_u \in \Sigma_u, u = 1, \ldots, k \quad (15)
\]

\[
I_{i,-1} = \overline{I}_{i,-1}, B_{i,-1} = \overline{B}_{i,-1} \quad i = 1, \ldots, n
\]

\[
R_{itm} = \overline{R}_{itm} \quad i = 1, \ldots, n, m \in M_i, t = -\tau_{im}, \ldots, -1
\]

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

\[
R_{itm} \in \mathbb{N} \quad i = 1, \ldots, n, m \in M_i, t = 0, \ldots, T - \tau_{im}
\]

The objective minimizes the costs of holding finished items on inventory, the costs of delivering products too late and the extra costs involved with using different production modes. Constraint (12) is the inventory balance equation, which models the flow of goods and enforces the satisfaction of demand. Constraint (13) determines the dependent demand. Constraint (14) ensures that backlog is only used for external demand. Constraint (15) ensures for each resource and subset of time slots that the amount of resources needed to execute all release orders is not bigger than the capacity.

Note that there are an exponential number of capacity constraints in Formulation 2. However, the idea of Benders’ decomposition is not to add all constraints corresponding to extreme rays of the dual of the subproblem at once, but to solve the master problem without these constraints and to add them when needed. We will describe an algorithm to do this in section 6.

5. Equivalence of the formulations

Although the way we constructed our second formulation implies that it is equivalent to our first formulation, in this section we will give an alternative proof. First we introduce the Resource
assignment graph, which is a directed graph that can be used to determine if the capacity constraints of Formulation 1 hold, i.e. if enough resources are available to execute the released orders. We prove this statement in a lemma, which we use to prove the equivalence of the formulations. Furthermore, this Resource assignment graph will be used in the algorithm in section 6 to find missing capacity constraints for a given release plan, as an alternative to creating the dual of the subproblem and searching for extreme rays.

Given a set of release orders and a resource $u$, we define a Resource assignment graph as follows. There is a source, a sink, a node for each release order $(i, t, m)$ and a node for each time slot $s$. There is an arc from the source to each release order $(i, t, m)$ with capacity $R_{itm} p_{t, i, u, m}^{\text{tot}}$ and from each time slot $s$ to the sink with capacity $c_{us}$. Furthermore, there is an arc from each release order $(i, t, m)$ to each time slot $s$ in which resource $u$ can be used for the execution of release order $(i, t, m)$. On these arcs the capacity is $R_{itm} p_{t, s - t, u, m}^{\text{max}}$ and the minimum flow is of size $R_{itm} p_{t, s - t, u, m}^{\text{min}}$. An example can be found in Figure 2.

![Resource assignment graph](image)

**Figure 2:** A Resource assignment graph. The maximum flow in this graph is of size $\sum_{itm} R_{itm} p_{t, i, u, m}^{\text{tot}}$ if and only if there is enough capacity of resource $u$ available to execute the release plan.

Having defined the Resource assignment graph we show that the maximum flow (c.f. Ford Jr & Fulkerson (1955)) in this graph is of size $\sum_{itm} R_{itm} p_{t, i, u, m}^{\text{tot}}$ if and only if there is enough capacity of resource $u$ available to execute the release plan.

**Lemma.** The maximum flow in a Resource assignment graph is of size $\sum_{itm} R_{itm} p_{t, i, u, m}^{\text{tot}}$ if and only if the capacity constraints of Formulation 1 hold for resource $u$.

**Proof.** Suppose there is a flow of size $\sum_{itm} R_{itm} p_{t, i, u, m}^{\text{tot}}$ and let $Z_{itsum}$ be the size of the flow on the arc from $(i, t, m)$ to $s$. The size of the flow implies that on each arc from the source to $(i, t, m)$ the flow
is \( R_{itm}^{\text{flat}} \). Thus the constraint \( \sum_{s=t+1}^{t+\tau_{im}} Z_{itsum} = p_{itm}^{\text{flat}} R_{itm} \) is satisfied, by the flow conservation in node \((i, t, m)\). By the flow conservation in node \(s\) constraint \( \sum_{i=1}^{n} \sum_{m \in M_i} \sum_{t=s-\tau_{im}}^{s-1} Z_{itsum} \leq c_{us} \) is satisfied, since the flow from \(s\) to the sink is at most \(c_{us}\). Because of the capacity and minimal flow constraints on the arcs from \((i, t, m)\) to \(s\) also constraint \( p_{i,s-t,u,m}^{\text{min}} R_{itm} \leq Z_{itsum} \leq p_{i,s-t,u,m}^{\text{max}} R_{itm} \) is satisfied. Thus the capacity constraints of Formulation 1 for resource \(u\) hold if the maximum flow is of size \( \sum_{itm} R_{itm}^{\text{flat}} \).

On the other hand, if the capacity constraints of Model 1 hold for resource \(u\), the variables \(Z_{itsum}\) have values such that the constraints hold. Then a feasible flow of size \( \sum_{itm} R_{itm}^{\text{flat}} \) can be obtained by putting a flow of size \( Z_{itsum} \) on the edges from \((i, t, m)\) to \(s\). This clearly is the maximum flow.

We use this lemma to prove that both our formulations are equivalent.

**Theorem.** Formulation 1 and Formulation 2 are equivalent.

**Proof.** Since the difference between both formulations is the type of capacity constraints, we prove the equivalence of the capacity constraints. By the definition of \( \chi_{itmsu} \), the capacity constraints of Formulation 2 are necessary conditions for any set of releases that satisfies the capacity constraints of Formulation 1. Hence we prove that these constraints are also sufficient, i.e. that the capacity constraints of Formulation 2 are not satisfied if the capacity constraints of Formulation 1 are not satisfied. For this we use the lemma and prove that the capacity constraints of Formulation 2 are not satisfied if the maximum flow in the corresponding Resource assignment graph is not of size \( \sum_{itm} R_{itm}^{\text{flat}} \).

If there is no feasible flow in the corresponding Resource assignment graph, then \( \sum_{itm} R_{itm}^{\text{min}} c_{us} > c_{us} \) for some \(s\). Thus the constraint \( \sum_{itm} \chi_{itmsu} R_{itm} \leq \sum_{s \in \sigma_u} c_{us} \) is violated for \(\sigma_u = \{s\} \). If there is a feasible flow, but the maximum flow is not of size \( \sum_{itm} R_{itm}^{\text{flat}} \), define \(F\) as the maximum flow, \(F_{itm}\) as the flow from the source to \((i, t, m)\), \(F_{itms}\) as the flow from \((i, t, m)\) to \(s\) and \(\sigma_u\) as the set of time slots that can be reached from the source in the residual graph of the maximum flow. Note that, since \(F\) is not of size \( \sum_{itm} R_{itm}^{\text{flat}} \), there is at least one \((i, t, m)\) such that \(F_{itm} < R_{itm}^{\text{flat}}\) and for this \((i, t, m)\) there is at least one \(s\) such that \(F_{itms} = R_{itm} \) hence \(\sigma_u\) is not empty.

Now we prove that \( \sum_{itm} \chi_{itmsu} R_{itm} \leq \sum_{s \in \sigma_u} c_{us} \) is violated for this \(\sigma_u\). To do this we show \( \sum_{itm} \sum_{s \in \sigma_u} F_{itms} < \sum_{itm} \sum_{s \in \sigma_u} \chi_{itmsu} R_{itm} \), which is enough since \( \sum_{itm} \sum_{s \in \sigma_u} F_{itms} = \sum_{s \in \sigma_u} \sum_{s \in \sigma_u} \chi_{itmsu} \leq \sum_{s \in \sigma_u} \sum_{s \in \sigma_u} \chi_{itmsu} R_{itm} \) by definition of \(F\) and \(\sigma_u\). We prove this by distinguishing two cases. For the first case, let \((i, t, m)\) be one of the release orders for which \(F_{itm} < R_{itm}^{\text{flat}}\) and thus \(\sum_{s \in \sigma_u} F_{itms} < R_{itm}^{\text{flat}} - \sum_{s \notin \sigma_u} \chi_{itmsu} R_{itm} \). Note that \(s \notin \sigma_u\) implies \(F_{itms} = p_{i,s-t,u,m}^{\text{max}} R_{itm} \) and thus \(\sum_{s \in \sigma_u} F_{itms} < R_{itm}^{\text{flat}} - \sum_{s \notin \sigma_u} \chi_{itmsu} R_{itm} \).

For the second case, let \((i, t, m)\) be a release order for which \(F_{itm} = R_{itm}^{\text{flat}}\) and thus \(\sum_{s \in \sigma_u} F_{itms} = R_{itm}^{\text{flat}} - \sum_{s \notin \sigma_u} \chi_{itmsu} R_{itm} \). If for all \(s \in \sigma_u\), \(F_{itms} = p_{i,s-t,u,m}^{\text{max}} R_{itm}\), \(\sum_{s \in \sigma_u} F_{itms} = \sum_{s \in \sigma_u} R_{itm}^{\text{flat}} - \chi_{itmsu} R_{itm} \). On the other hand, if there is a time slot \(s\) for which \(F_{itms} > p_{i,s-t,u,m}^{\text{max}} R_{itm}\), \((i, t, m)\) can be reached from the source in the residual graph. Thus \(s \notin \sigma_u\) implies \(F_{itms} = p_{i,s-t,u,m}^{\text{max}} R_{itm} \) and thus \(\sum_{s \in \sigma_u} F_{itms} = R_{itm}^{\text{flat}} - \sum_{s \notin \sigma_u} \chi_{itmsu} R_{itm} \). Summing over these cases yields \( \sum_{itm} \sum_{s \in \sigma_u} F_{itms} < \sum_{itm} \sum_{s \in \sigma_u} \chi_{itmsu} R_{itm} \) and thus \( \sum_{itm} \sum_{s \in \sigma_u} \chi_{itmsu} R_{itm} \leq \sum_{s \in \sigma_u} c_{us} \) is violated for this \(\sigma_u\).

Hence the capacity constraints of Formulation 2 are not satisfied if the maximum flow is not of size \( \sum_{itm} R_{itm}^{\text{flat}} \). From the lemma follows that the capacity constraints of Formulation 2 form a sufficient condition and thus that Formulation 1 and Formulation 2 are equivalent. \(\square\)
6. Algorithm

When Benders’ decomposition is applied, the master problem is solved without the constraints of the subproblem and these missing constraints are added when needed. Since Formulation 2 has an exponential number of capacity constraints, this will also be our approach. In this section we describe our algorithm to solve Formulation 2, but first we explain how we use the Resource assignment graph to find the missing capacity constraints instead of searching for extreme rays in the dual problem of the subproblem. The idea for using this Resource assignment graph came from the fact that the subproblem looks like a flow problem. Furthermore, we believe it is more intuitive than searching for extreme rays in the dual problem of the subproblem.

In the previous section we proved that the maximum flow over a Resource assignment graph is of size \( \sum \text{time} R_{\text{time}}^{\text{tot}} \) if and only if there is enough capacity of resource \( u \) available to execute the release plan. So, by constructing the Resource assignment graph for each resource given a set of release orders and finding the maximum flow over this graph we can check if this solution violates any missing capacity constraints. Furthermore, we showed that the constraint \( \sum \text{time} \chi_{\text{time}} R_{\text{time}} \leq \sum_{s \in \sigma u} c_{us} \) is a violated constraint if \( \sigma u \) is the set of time slots in the Resource assignment graph that can be reached from the source in the residual graph. So, by finding this set in the residual graph we can find a violated constraint. The method to find a missing constraint is outlined in Algorithm 1. Note that the most time consuming part of Algorithm 1 is solving the maximum flow, which is done only once. Since the maximum flow problem can be solved in polynomial time, Algorithm 1 can be executed in polynomial time.

In some cases it is beneficial to partition the \( \sigma u \) found by Algorithm 1 into two or more subsets. This is the case if there is no possible release order that requires resource \( u \) in more than one of these subsets, i.e. if for any given combination of \( i, t \) and \( m \) \( \chi_{\text{time}} = \chi_{\text{time}}^{\sigma u} \) for a subset \( \sigma u \). In that case partitioning \( \sigma u \) into subsets and adding one capacity constraint for each subset results in stronger capacity constraints than adding the capacity constraint for \( \sigma u \).

In classical Benders’ decomposition the master problem is solved and then a subproblem is used to check if any constraint is violated (c.f. Rahmaniani et al. (2017)). If a constraint is violated, it is added to the master problem, which is solved again. This is repeated until no more violated constraints are found and the problem is solved. In our case this is very time consuming, because it requires solving mixed integer linear programs to optimality until sufficient capacity constraints are found. To speed up the process of finding sufficient capacity constraints we apply two-phase Benders (c.f. Rahmaniani et al. (2017)), i.e. we first solve the LP-relaxation with classical Benders’ decomposition. Then we add the constraints found while solving the LP-relaxation to the master problem and start solving it with branch and bound. However, we do not solve the master problem to optimality before checking for violated capacity constraints, but we check for violated
**Input:** Release orders $R_{itm}$ and resource group $u$;
Construct the Resource assignment graph given the input;
Check if there is a feasible flow in this graph;

*if there is no feasible flow then*

  - Find $s$ for which $\sum_{itm} \chi_{itmus} R_{itm} > c_{us}$;
  - $\sigma_u = s;$

*else*

  - Find the maximum flow;
  - *if maximum flow is of size $\sum_{itm} R_{itm}p_{i.am}$ then*
    - $\sigma_u = \emptyset$;
  - *else*
    - Construct the residual graph of the maximum flow;
    - $\sigma_u = \{ s \mid s$ can be reached from the source in the residual graph$\}$;
  - *end*
*end*

**Output:** $\sigma_u$;

Algorithm 1: Method to find a set of time slots $\sigma_u$ belonging to a violated capacity constraint.

constraints for each integer solution found during the branch and bound. Our approach is described in Algorithm 2 and could be called truncated two-phase Benders’ decomposition.

In other words, Algorithm 2 solves Model 2 by using Algorithm 1 to find missing capacity constraints. It starts with Formulation 2 without capacity constraints or with only a small number of its capacity constraints. It solves the LP relaxation of this model, adds any missing capacity constraints and repeats this process until the solution of the LP relaxation satisfies all capacity constraints. This is a fast way to find some important missing capacity constraints, since a period with too few resources in the LP relaxation is also a period with too few resources in the original problem. After this the mixed integer linear program is solved with branch and bound. However, for each new incumbent we check if it violates any missing capacity constraints. If this is the case we add these constraints and restart the branch and bound. Otherwise, we continue the branch and bound process.

To add the missing constraints as global cuts and continue the branch and bound process might seem a more natural approach, which is sometimes called modern Benders or the single-search-tree strategy (c.f. Rahmaniani et al. (2017)). However, in this case CPLEX does not use the new constraint in its presolving procedure and thus cannot use this knapsack constraint to strengthen other constraints and bounds. Still, we implemented and tested this approach and it was clearly slower than Algorithm 2.
**Input:** Formulation 2 with only a subset of its capacity constraints; Consider the LP relaxation of the model;

**repeat**
- Solve the LP relaxation;
  - for $u = 1$ to $k$
    - Use Algorithm 1 to find $\sigma_u$ given $u$ and the solution of the LP relaxation;
    - if $\sigma_u \neq \emptyset$
      - Partition $\sigma_u$ if possible;
      - Add a capacity constraint for each part of $\sigma_u$ to the LP relaxation;
    - end
  - end
until $\sigma_u = \emptyset, \forall u$;

Consider the original problem of the LP relaxation;

**repeat**
- Start solving the model with branch and bound;
  - while solving do
    - if new incumbent is found then
      - for $u = 1$ to $k$
        - Use Algorithm 1 to find $\sigma_u$ given $u$ and the new incumbent;
      - end
    - if $\sigma_u = \emptyset, \forall u$ then
      - Continue solving;
    - else
      - Stop solving;
    - end
  - end
  - for $u = 1$ to $k$
    - if $\sigma_u \neq \emptyset$
      - Partition $\sigma_u$ if possible;
      - Add a capacity constraint for each part of $\sigma_u$ to the model;
    - end
  - end
until $\sigma_u = \emptyset, \forall u$;

**Output:** Optimal solution of Formulation 2;

Algorithm 2: An algorithm to solve Formulation 2.
7. Computational comparison of the two formulations

In Sections 3 and 4 we introduced two formulations for the mid-term production planning problem for high-tech low-volume supply chains. Formulation 1 can be solved using branch and bound. For Formulation 2 this is not viable, thus we developed Algorithm 2 to solve this model. In this section we compare these two methods to solve the mid-term production planning problem for high-tech low-volume supply chains. We first describe the test cases and then the results.

The test cases are loosely based on a mid-term production planning instance at ASML, one of world’s leading providers of semiconductor equipment. The data is simplified, disguised and made up in such a way that the tradeoffs to be addressed by their production planning are preserved. At ASML, machines are assembled from modules and modules are assembled from components, which are ordered at various suppliers. Besides machines ASML produces upgrade packages and service parts for systems in the field. Their supply chain can be characterized by enormous bills of materials, long lead times, expensive materials, low-volume production, fluctuating demand, numerous engineering changes and customer configurations. For the mid-term planning, each 4 weeks they plan their production for the next 18 to 30 months in week buckets, based on a demand forecast. Two of the most important challenges are the timing of the orders for expensive components with long lead times and the lower resource availability during vacations.

The test cases consider the production of 4 machines and their 6 main modules during 104 weeks. The demand for each machine is based on a real life demand pattern and is on average 0.65 per week. The bill of materials of one machine in our test cases can be found in Figure 3. The production of a machine starts with the assembly, followed by an extensive testing procedure and packing the machine for shipping. Testing takes most of the time, but each of these phases takes at least one week. Note that the order of these phases cannot be changed and that for example the testing phase consists of different tasks the order of which is important. Teams of employees are specifically trained for one of these phases. Note that the production of a machine and the testing procedure are split into a generic part and a client specific part. The modules are machine specific and have a lead time of 1 or 2 weeks. We only consider the resource that is required for the most critical module.

During vacation periods less capacity is available at ASML, therefore machines are planned with longer lead times such that the resource claims per week are smaller during the vacations. This way, less inventory is required to fulfill the demand during vacations. Since most of the production time is spent on the testing procedure, testing activities are the most challenging to schedule around the vacation periods. Therefore, the testing resource is the only resource with less capacity availability during vacations in our test cases. The vacation periods cover 6 weeks of summer vacations and one week of Christmas holidays in each year. Furthermore, only items that require the testing resource
can be produced in different modes, i.e. the generic and the client specific parts of the machines can be produced in 4 different production modes, which differ in lead time. This extra lead time can only be used for testing. The parameter setting relevant to the resource usage and modes of one machine can be found in Table 1. The parameters for the other 3 machines are slightly different, but have a similar structure.

![Diagram](Diagram.png)

Table 1: The parameter setting for one machine.

Table 2 gives an overview of the characteristics of the test cases. We created 60 test cases by varying the number of resources and the available capacity of the test resource in both normal and vacation periods. In the first set of 15 test cases we only included the test resource, which

![Diagram](Diagram.png)
is required for the production of all generic and client specific parts of the machines. The normal capacity of the test resource is varied from 2700 to 3300 in steps of 150. The vacation capacity is varied between $\frac{2}{3}$, $\frac{1}{2}$, and $\frac{1}{3}$ of the normal capacity. For the second set of 15 test cases we added the resource that is required for the most critical module of three of the machines. For the third set of test cases we added the packing resource, which is required for the client specific part of the machines. And for the last set of test cases we added the assembly resource, which is required for the generic part of the machines.

<table>
<thead>
<tr>
<th>Number of items</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of resources</td>
<td>1-4</td>
</tr>
<tr>
<td>Number of time slots</td>
<td>104</td>
</tr>
<tr>
<td>Level 1 Client</td>
<td>Level 2 Generic</td>
</tr>
<tr>
<td>Number of items</td>
<td>4</td>
</tr>
<tr>
<td>Number of components per item</td>
<td>1</td>
</tr>
<tr>
<td>Total demand</td>
<td>272</td>
</tr>
<tr>
<td>Number of production modes</td>
<td>4</td>
</tr>
<tr>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>Lead time</td>
<td>5</td>
</tr>
<tr>
<td>Demand per time slot</td>
<td>0</td>
</tr>
<tr>
<td>Total demand</td>
<td>12</td>
</tr>
<tr>
<td>Total required of resource test</td>
<td>208</td>
</tr>
<tr>
<td>Total required of resource mod</td>
<td>0</td>
</tr>
<tr>
<td>Total required of resource pack</td>
<td>150</td>
</tr>
<tr>
<td>Total required of resource assy</td>
<td>0</td>
</tr>
<tr>
<td>Regular test capacity</td>
<td>2700-3300</td>
</tr>
<tr>
<td>Vacation test capacity</td>
<td>900-2200</td>
</tr>
<tr>
<td>Mod capacity</td>
<td>14</td>
</tr>
<tr>
<td>Pack capacity</td>
<td>1000</td>
</tr>
<tr>
<td>Assy capacity</td>
<td>1250</td>
</tr>
</tbody>
</table>

Table 2: Characteristics of the test cases.

The models and algorithms are implemented and tested using AIMMS 4.24. Instead of the maximum flow algorithm used in Algorithm 1 we implemented a LP formulation of the corresponding minimum cut problem, for practical reasons. Note that only a small part of the running time is spent on Algorithm 1. Furthermore, we started Algorithm 2 with as input Formulation 2 with all sets of length one for the packing, assembly and module resource and no sets for the test resource. Note that for the packing, assembly and module resource the sets of length one are sufficient since there is no flexibility in the assignment of these resources to time slots of production. Hence, it is not required to check if there are any violated capacity constraint for these resources during Algorithm 2.

The tests were performed on a HP EliteBook 8570w laptop with an Intel Core i7-3520M CPU 2.90GHz processor and 8 GB RAM. Formulation 1 was solved with CPLEX 12.6.3 and Formulation 2 was solved using Algorithm 2 and CPLEX 12.6.3. CPLEX used 4 threads and its standard settings. Thus, by default, preprocessing is used to eliminate rows and columns, probing to tighten bounds, heuristics to find integer solutions and cuts to strengthen lower bounds. Both models were given
a maximal run time of 900 seconds. The results of the test cases can be found in Table 4, 5, 6 and 7. A summary of these results is given in Table 3. Furthermore, the number of variables and constraints per set of test cases is given in Table 8.

<table>
<thead>
<tr>
<th>Optimal solution</th>
<th>Shorter run time</th>
<th>Better solution</th>
<th>Smaller gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form. 1 Form. 2</td>
<td>#</td>
<td>Form. 1 Form. 2</td>
<td>Form. 1 Form. 2</td>
</tr>
<tr>
<td>Yes Yes</td>
<td>7</td>
<td>2 5</td>
<td>- -</td>
</tr>
<tr>
<td>No Yes</td>
<td>20</td>
<td>- 20</td>
<td>- 3 20</td>
</tr>
<tr>
<td>No No</td>
<td>33</td>
<td>- -</td>
<td>4 9</td>
</tr>
</tbody>
</table>

Table 3: Summary of the test results.

In Table 3 we see that in 7 cases both models found the optimal solution. In these cases the average solution time for Formulation 1 is 241 seconds and for Formulation 2, 83 seconds. There are 33 cases for which no optimal solution is found for both models. In these cases the average optimality gap was 0.82% for Formulation 1 and 0.55% for Formulation 2. Furthermore, there are 20 cases in which Formulation 2 was solved within 900 seconds, but Formulation 1 was not. The average solution time for Formulation 1 is thus bigger than 900 seconds, while the average solution time for Formulation 2 is only 201 seconds in these cases. Additionally, we see that for Formulation 2 in 12 cases a better solution was found than for Formulation 1, while vice versa was the true for only 4 cases. In Table 4, 5, 6 and 7 we see that for cases in which Formulation 1 outperforms Formulation 2 the differences are small, while the differences are often significant when Formulation 2 outperforms Formulation 1. We conclude that for these test cases solving Formulation 2 using Algorithm 2 is substantially faster than solving Formulation 1.
<table>
<thead>
<tr>
<th>Cases with $k = 1$</th>
<th>Formulation 1</th>
<th>Formulation 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Run time</td>
<td>Best Gap</td>
</tr>
<tr>
<td></td>
<td>(seconds)</td>
<td>solution</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regular Vacation LP relax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3300 2000 197</td>
<td>26 275 0.60%</td>
<td>2358 0</td>
</tr>
<tr>
<td>3300 1650 7502</td>
<td>900 7573 0.03%</td>
<td>503270 23178</td>
</tr>
<tr>
<td>3300 1100 1442</td>
<td>900 16538 0.22%</td>
<td>744891 46299</td>
</tr>
<tr>
<td>3150 2100 1213</td>
<td>354 1261 0.60%</td>
<td>255105 0</td>
</tr>
<tr>
<td>3150 1575 8644</td>
<td>900 8609 0.36%</td>
<td>1218022 901118</td>
</tr>
<tr>
<td>3150 1050 17302</td>
<td>900 17387 0.18%</td>
<td>803151 447755</td>
</tr>
<tr>
<td>3000 2000 2948</td>
<td>900 2593 2.33%</td>
<td>392203 255026</td>
</tr>
<tr>
<td>3000 1500 9798</td>
<td>900 9836 0.26%</td>
<td>1155623 251518</td>
</tr>
<tr>
<td>3000 1000 18205</td>
<td>417 18300 0.00%</td>
<td>1534631 0</td>
</tr>
<tr>
<td>2850 1900 3844</td>
<td>900 3921 1.41%</td>
<td>4386835 301188</td>
</tr>
<tr>
<td>2850 1425 11001</td>
<td>900 11160 0.45%</td>
<td>407862 254230</td>
</tr>
<tr>
<td>2850 950 19425</td>
<td>900 19584 0.16%</td>
<td>249814 89546</td>
</tr>
<tr>
<td>2700 1800 5271</td>
<td>900 5307 0.00%</td>
<td>825789 573674</td>
</tr>
<tr>
<td>2700 1350 12411</td>
<td>900 12524 0.55%</td>
<td>268388 146964</td>
</tr>
<tr>
<td>2700 900 22162</td>
<td>900 22288 0.55%</td>
<td>52203 19184</td>
</tr>
</tbody>
</table>

Table 4: Results of test cases with 1 resource.
<table>
<thead>
<tr>
<th>Cases with k = 3</th>
<th>Formulation 1</th>
<th>Formulation 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Regular</td>
<td>Vacation</td>
</tr>
<tr>
<td></td>
<td>capacity</td>
<td>capacity</td>
</tr>
<tr>
<td>3300</td>
<td>2200</td>
<td>198</td>
</tr>
<tr>
<td>3300</td>
<td>1650</td>
<td>756</td>
</tr>
<tr>
<td>3300</td>
<td>1100</td>
<td>1652</td>
</tr>
<tr>
<td>3150</td>
<td>2100</td>
<td>1215</td>
</tr>
<tr>
<td>3150</td>
<td>1575</td>
<td>8725</td>
</tr>
<tr>
<td>3150</td>
<td>1050</td>
<td>17386</td>
</tr>
<tr>
<td>3000</td>
<td>2000</td>
<td>2516</td>
</tr>
<tr>
<td>3000</td>
<td>1500</td>
<td>9879</td>
</tr>
<tr>
<td>3000</td>
<td>1000</td>
<td>162890</td>
</tr>
<tr>
<td>2850</td>
<td>1900</td>
<td>3896</td>
</tr>
<tr>
<td>2850</td>
<td>1425</td>
<td>11083</td>
</tr>
<tr>
<td>2850</td>
<td>950</td>
<td>19517</td>
</tr>
<tr>
<td>2700</td>
<td>1800</td>
<td>5353</td>
</tr>
<tr>
<td>2700</td>
<td>1350</td>
<td>12493</td>
</tr>
<tr>
<td>2700</td>
<td>900</td>
<td>22153</td>
</tr>
</tbody>
</table>

Table 6: Results of test cases with 3 resources.
Besides the performance of the models, Table 4, 5, 6 and 7 also contain information on the behavior of the two solution methods. For both approaches the number of processed and unexplored nodes in the branch and bound tree are reported. For Formulation 2 these are split in the number of nodes in total and in the last run, i.e. summed over all (re)starts and since the last restart, respectively. Furthermore, the number of constraints added using the LP relaxation and in total are reported. We see that most of the constraints are added using the LP relaxation and that the number of added constraints is in general small. Additionally, the number of times the branching process was restarted and the time stamp of the last restart are reported. In most test cases the branching process is restarted less than 10 times and the last restart is after less than 60 seconds. We conclude that our approach of finding missing constraints is quite effective. Lastly, the number of incumbent callbacks is reported, i.e. the number of times a better integer solution was found and a minimum cut problem was solved to check if no new capacity constraint were violated. The average number of seconds required to solve this minimum cut problem can be found in Table 8.

<table>
<thead>
<tr>
<th>Set of cases</th>
<th>Constraints</th>
<th>Variables</th>
<th>Int. vars</th>
<th>Nonzeros</th>
<th>Constraints (avg)</th>
<th>Variables</th>
<th>Int. vars</th>
<th>Nonzeros (avg)</th>
<th>avg mincut time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 resource</td>
<td>60625</td>
<td>37185</td>
<td>6456</td>
<td>172833</td>
<td>7655</td>
<td>13241</td>
<td>6456</td>
<td>56439</td>
<td>0.25</td>
</tr>
<tr>
<td>2 resources</td>
<td>62334</td>
<td>38409</td>
<td>6456</td>
<td>176745</td>
<td>7757</td>
<td>13241</td>
<td>6456</td>
<td>56140</td>
<td>0.35</td>
</tr>
<tr>
<td>3 resources</td>
<td>91798</td>
<td>52137</td>
<td>6456</td>
<td>237597</td>
<td>7861</td>
<td>13241</td>
<td>6456</td>
<td>58585</td>
<td>0.46</td>
</tr>
<tr>
<td>4 resources</td>
<td>115438</td>
<td>62953</td>
<td>6456</td>
<td>289013</td>
<td>7952</td>
<td>13241</td>
<td>6456</td>
<td>57225</td>
<td>0.58</td>
</tr>
</tbody>
</table>

Table 8: Size of models for different sets of test cases.

Observe that in cases with more resources the time limit is reached more often. In many of these cases the same solution value is found by both models. However, the optimality gap of Formulation 2 is in general smaller. Furthermore, note that the number of processed nodes for Formulation 2 is much bigger than for Formulation 1 in cases that the time limit was reached. This can be explained by the smaller number of constraints and variables of Formulation 2.

8. Comparison to Spitter et al. (2005)

This section will compare our model to the model in Spitter et al. (2005), which is, to the best of our knowledge, the model in the literature that is the closest to our model. Spitter et al. (2005) consider capacity constrained Supply Chain Operations Planning for arbitrary supply chain structures. Supply Chain Operations Planning aims to coordinate the release of materials and resources in the supply chain such that customer service goals are met at minimal costs. In their paper two alternative linear programming formulations of this planning problem are proposed. We will focus on the formulation with balance equations. To compare both models, we will discuss the differences in assumptions, introduce their notation and model, introduce a test case that fits in both models and compare the computational times of both approaches.
The most important difference in assumptions is that we assume a low-volume production environment, while they do not. This allows them to use continuous variables, while for low-volume environments integer release variables are required. We cope with this difference by adding an integrality constraint on the release variables in their model, such that it can be used for low-volume production planning.

Spitter et al. (2005) assume that items can be produced on several resources, i.e. that there are a number of resources that can produce a certain item and it is even possible to produce an order for an item partly on one of those resources and partly on another. In the high-tech low-volume industries we consider, resources are often specialized in a certain task and switching resources during the production process is very undesirable because of the complexity of the production process, if it is even physically possible at all. Furthermore, in these industries the production of some items requires multiple resources.

In the model of Spitter et al. (2005) capacity can be allocated at any point in time during the lead time, i.e. the resources needed for the production of one item could all be claimed in one time slot or could be spread out over different time slots within the lead time. This decouples the material release from the resource consumption. In an high-tech low-volume environment the production of an item consists of different tasks that often cannot be executed in parallel and sometimes cannot be interrupted. This limits the resource allocation per time slot within the lead time.

The model of Spitter et al. (2005) does not consider different production modes nor allows for components arriving during the lead time. These assumptions are in line with the very flexible allocation of resources and fit high-volume production settings. This concludes our discussion of the differences in assumptions.

Now, we will introduce the notation used in Spitter et al. (2005). Since many of the parameters and variables are the same as in our model, we only describe the differences. Since they assume that an item can be produced on multiple resources, they use $\mathcal{R}_i$ as the set of resources which can produce item $i$ and $V_{iut}$ as a variable that specifies how much of resource $u$ in time slot $t$ is allocated to the production of items $i$. This allows the $Z_{its}$ variable to be independent of the resource. Other differences and exact definitions can be found in the following overview of parameters:
\( \tau_i : \) \( \tau_i \in \mathbb{N} \), \( i = 1, \ldots, n \), the planned lead time for the production of item \( i \).

\( \mathcal{R}_i : \) the set of resources that could be used for the production of item \( i \), \( i = 1, \ldots, n \).

\( p_i : \) \( p_i \geq 0 \), the total amount of a resource required to produce one item \( i \), here is assumed that this is equal for each \( u \in \mathcal{R}_i \).

\( \bar{R}_{it} : \) \( \bar{R}_{it} \in \mathbb{N} \), the size of the work order of item \( i \) released in the past at time \( t \), \( i = 1, \ldots, n \), \( t = -\tau_i, \ldots, -1 \).

\( \nabla_{iut} : \) \( \nabla_{iut} \geq 0 \), the capacity of resource \( u \) already allocated in time slot \( t \) to the production of items \( i \), \( i = 1, \ldots, n \), \( u \in \mathcal{R}_i \), \( t = -\tau_i + 1, \ldots, 0 \).

\( \bar{Z}_{its} : \) \( \bar{Z}_{its} \geq 0 \), the part of the work, related to the work order released in the past, \( \bar{R}_{it} \), already executed in time slot \( s \), \( i = 1, \ldots, n \), \( t = -\tau_i, \ldots, -1 \), \( s = t + 1, \ldots 0 \).

As mentioned before we assume that \( R_{it} \) is an integer variable, while this is not the case in Spitter et al. (2005). Their linear programming formulation with balance equations is as follows:

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

subject to

\[
I_{it} = I_{i,t-1} + R_{i,t-\tau_i} - D_{it} - G_{it} + B_{it} - B_{i,t-1} \quad i = 1, \ldots, n, t = 0, \ldots, T \tag{16}
\]

\[
G_{it} = \sum_{j=1}^{n} h_{ij} R_{jt} \quad i = 1, \ldots, n, t = 0, \ldots, T - 1 \tag{17}
\]

\[
B_{it} - B_{i,t-1} \leq D_{it} \quad i = 1, \ldots, n, t = 0, \ldots, T \tag{18}
\]

\[
p_i R_{it} = \min_{t'=t+\tau_i,T} \sum_{s=t+1}^{t'-1} \bar{Z}_{its} \quad i = 1, \ldots, n, t = -\tau_i, \ldots, T - 1 \tag{19}
\]

\[
\sum_{s=t-\tau_i}^{t-1} \bar{Z}_{ists} = \sum_{u \in \mathcal{R}_i} V_{iut} \quad i = 1, \ldots, n, t = 1, \ldots, T \tag{20}
\]

\[
\sum_{i : u \in \mathcal{R}_i} V_{iut} \leq c_{ut} \quad u = 1, \ldots, k, t = 1, \ldots, T \tag{21}
\]
\[ I_{i,-1} = \overline{I}_{i,-1}, B_{i,-1} = \overline{B}_{i,-1} \quad i = 1, \ldots, n \]
\[ R_{it} = \overline{R}_{it} \quad i = 1, \ldots, n, t = -\tau_i, \ldots, -1 \]
\[ Z_{ists} = \overline{Z}_{ists} \quad i = 1, \ldots, n, t = -\tau_i, \ldots, -1, s = t + 1, \ldots, 0 \]
\[ I_{it}, B_{it} \geq 0 \quad i = 1, \ldots, n, t = 0, \ldots, T \]
\[ R_{it} \in \mathbb{N} \quad i = 1, \ldots, n, t = 0, \ldots, T - 1 \]
\[ V_{iut} \geq 0 \quad i = 1, \ldots, n, u \in \mathcal{R}_i, t = 0, \ldots, T - 1 \]
\[ Z_{ists} \geq 0 \quad i = 1, \ldots, n, t = -\tau_i, \ldots, T - 1, s = t + 1, \ldots, t + \tau_i \]

To create a test case that fits both models we made a number of assumptions. We assume that each item can be produced by, and requires, at most one resource, i.e. \(|\mathcal{R}_i| = 1\) for the model of Spitter et al. (2005) and \(|\mathcal{R}_{im}| \leq 1\) for our model. Furthermore, we do not consider different production modes, thus \(|\mathcal{M}_i| = 1\) and \(\gamma_{itm} = 0\), nor allow component claims during lead time, i.e. \(\delta_{ijm} = 0\). There are also no minimum or maximal capacity claims per time slot of production, thus \(p_{qum}^{\text{min}} = 0\) and \(p_{qum}^{\text{max}} = p_{qum}^{\text{tot}}\).

Under the first assumption constraint (20) and (21) simplify to \(\sum_{i: u \in \mathcal{R}_i} \sum_{s = i - \tau_i}^{t - 1} Z_{ist} \leq c_{ut}\) for the one \(u \in \mathcal{R}_i\), which is equal to constraint (6) under the assumption that there are not multiple modes. Also, the other constraints of the model of Spitter et al. (2005) are equal to the constraints of the first formulation of our model under the assumptions of this test case. Hence, the formulation with balance equations of the model of Spitter et al. (2005) is equal to the first formulation of our model for this test case.

The test case is similar to the test case in Spitter et al. (2005) and considers a fictitious supply chain with 263 items, of which 20 end items. The supply chain is converging and consists of 5 levels, i.e. end items are on level 1 and do not have any successors and each other item has exactly one successor on one level lower. There are 20 items on level 1, 30 items on level 2, 45 items on level 3, 67 items on level 4 and 101 items on level 5. Except the items on level 5, each item has at least one and on average 1.5 predecessors.

The test case considers 52 time slots. The lead times of the items on level 5 are arbitrarily set to 4 or 5 time slots, while the lead times of the other items are randomly set to 1 or 2. The inventory and backlog costs depend on the value of the items and backlog costs 10 times more than inventory. The value of the items on level 5 are arbitrarily set to an integer between 1 and 5 and the value of the other items is equal to the sum of the values of the predecessors plus one.

The test case contains 84 resources: 10 are used for the production of end items, 23 for items on level 3 and 51 for items on level 5. Each item on level 1, 3 and 5 requires exactly one resource during production. Each resource is used for the production of at least one and on average two items. The amount of a required resource needed for the production of one item is randomly set.
to an integer between 1 and 5.

To create different test scenarios we varied the demand and the available capacity. For each end item and each time slot, the demand follows a binomial distribution with a chance of success of 0.5 and the number of tries varies per test case. Test cases with the same number of tries have exactly the same demand pattern. Past releases are equal to the expected (dependent or independent) demand per time slot and past resource allocations are enough to fully produce each item released in the past. The expected capacity utilization varies over the test cases. The capacity is equal to the amount of resources required to fulfill the expected demand divided by the capacity utilization. For half the test cases the capacity is constant over time. For the other test cases there are two holiday periods of 4 time slots in which the expected utilization is 200%. In these test cases the expected utilization of the other time slots is chosen such that the expected utilization over the whole planning horizon is equal to the expected utilization in the other test cases.

This test case was used to compare the formulation with balance equations of Spitter et al. (2005) with the second formulation of our model. Both models are implemented and tested using AIMMS 4.24. Like in section 7, instead of the maximum flow algorithm used in Algorithm 1, we implemented a LP formulation of the corresponding minimum cut problem. Furthermore, we started Algorithm 2 with as input Formulation 2 with all sets of length one for all resources. The tests were performed on a HP EliteBook 8570w laptop with an Intel Core i7-3520M CPU 2.90GHz processor and 8 GB RAM. The model of Spitter et al. (2005) was solved with CPLEX 12.6.3 and Formulation 2 was solved using Algorithm 2 and CPLEX 12.6.3. CPLEX used 1 thread and its standard settings. Thus, by default, preprocessing is used to eliminate rows and columns, probing to tighten bounds, heuristics to find integer solutions and cuts to strengthen lower bounds. Both models were given a maximal run time of 900 seconds.

The results of the test cases can be found in Table 9 and 10. Similar to the previous section, more details about the behavior of the solving methods can also be found in these tables. Note that shorter run times, better best solutions and smaller gaps are highlighted. However, in our view, the difference in performance is not significant if both methods did not solve the case within 900 seconds and the difference in gap is less than 0.02%. So, in these cases the gap is not highlighted.

The formulation with balance equations of Spitter et al. (2005) outperforms our second formulation for most of the test cases. Apparently, removing the high-tech-specific aspects, such as different production modes and more complex resource requirements, reduces the computational complexity of the problem.
<table>
<thead>
<tr>
<th>Cases</th>
<th>Spitter et al. (2005)</th>
<th>Formulation 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Expected demand</td>
<td>Expected LP relax</td>
</tr>
<tr>
<td>5</td>
<td>0.85 6210.575</td>
<td>5 6332 0.00% 150 0</td>
</tr>
<tr>
<td>5</td>
<td>0.90 7798.375</td>
<td>8 1457.9 0.00% 490 0</td>
</tr>
<tr>
<td>5</td>
<td>0.95 19975.6</td>
<td>13 2322.1 0.00% 7399 0</td>
</tr>
<tr>
<td>4</td>
<td>0.85 11482.29</td>
<td>74 3722.4 0.00% 4536 0</td>
</tr>
<tr>
<td>4</td>
<td>0.90 24105.68</td>
<td>168 39469.5 0.00% 10621 0</td>
</tr>
<tr>
<td>4</td>
<td>0.95 39929.75</td>
<td>900 45472.1 0.00% 6332 0</td>
</tr>
<tr>
<td>3</td>
<td>0.85 6994.467</td>
<td>3 15622.7 0.00% 14 0</td>
</tr>
<tr>
<td>3</td>
<td>0.90 16357.19</td>
<td>506 18709.4 0.00% 29587 0</td>
</tr>
<tr>
<td>3</td>
<td>0.95 33793.81</td>
<td>900 36402.7 0.35% 46890 24572 0</td>
</tr>
<tr>
<td>2</td>
<td>0.85 34330.1</td>
<td>168 41027.7 0.00% 6743 0</td>
</tr>
<tr>
<td>2</td>
<td>0.90 36537.24</td>
<td>900 47339.5 0.00% 6332 0</td>
</tr>
<tr>
<td>2</td>
<td>0.95 48990.98</td>
<td>900 54364.7 0.62% 60891 27926 0</td>
</tr>
</tbody>
</table>

Table 9: Results of test cases without holiday periods.

<table>
<thead>
<tr>
<th>Cases</th>
<th>Spitter et al. (2005)</th>
<th>Formulation 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Expected demand</td>
<td>Expected LP relax</td>
</tr>
<tr>
<td>6</td>
<td>0.85 3976.89</td>
<td>563 860.8 0.00% 44123 0</td>
</tr>
<tr>
<td>5</td>
<td>0.90 8764.393</td>
<td>900 9763.5 0.07% 59166 39254 0</td>
</tr>
<tr>
<td>5</td>
<td>0.95 10490.63</td>
<td>900 13621.7 0.07% 53849 36591 0</td>
</tr>
<tr>
<td>4</td>
<td>0.85 13692.01</td>
<td>900 13745.7 0.01% 49883 13122 0</td>
</tr>
<tr>
<td>4</td>
<td>0.90 15378.88</td>
<td>900 18268.5 0.00% 54161 10256 0</td>
</tr>
<tr>
<td>4</td>
<td>0.95 18188.6</td>
<td>900 20914.4 0.15% 46154 23679 0</td>
</tr>
<tr>
<td>3</td>
<td>0.85 6355.205</td>
<td>900 810.3 0.10% 46185 13944 0</td>
</tr>
<tr>
<td>3</td>
<td>0.90 6854.115</td>
<td>900 8562.7 0.07% 50420 16050 0</td>
</tr>
<tr>
<td>3</td>
<td>0.95 8378.095</td>
<td>900 9622.7 1.05% 59135 19484 0</td>
</tr>
<tr>
<td>2</td>
<td>0.85 1264.33</td>
<td>900 12775.5 0.02% 59135 19484 0</td>
</tr>
<tr>
<td>2</td>
<td>0.90 14090.06</td>
<td>900 3255.8 0.02% 46660 29179 0</td>
</tr>
<tr>
<td>2</td>
<td>0.95 23534.79</td>
<td>900 6783.4 0.01% 45146 32701 0</td>
</tr>
</tbody>
</table>

Table 10: Results of test cases with two holiday periods.
Note that the concept of capacity utilization might have different implications in a low-volume setting than in a high-volume setting. Consider a resource that is only used for the production of one item. Suppose the expected demand of this item is 5 and the lead time 1. Then an expected capacity utilization of 0.95 implies that the capacity of this resource will allow the production of 5.26 items. Since it is impossible to create 0.26 items, 5.26 capacity means that at most 5 items can be produced, which is equal to the expected demand of this item. This happens more often when the average demand decreases or the expected capacity utilization increases. We see that in both cases the problem becomes more difficult to solve.

Based on the first five rows of table 10, our second formulation seems to outperform the formulation of Spitter et al. (2005) in the test cases with lower resource availability during the vacation periods and higher expected demand. The lower resource availability during the vacations means that the capacity problems are focused around the holidays. This benefits our second formulation, since the algorithm only needs to add constraints around the holidays. From the performance details in Table 9 and Table 10 we observe that Algorithm 2 adds less capacity constraints during the branch and bound and spends less time on restarts for the test cases with two vacation periods. Furthermore, we observe that higher expected utilization and lower average demand increase the number of constraints added during the branch and bound and time spend on restarts.

In conclusion, the addition of resource complexity typical for high-tech industries is best dealt with with Formulation 2, as the Benders-like approach enables to identify these complexities and adds the necessary resource constraints. This complexity includes difference in availability of resources over time. In case capacity constraints are stable over time and there is a one-to-one relationship between items and resources, Formulation 1 solves faster. As these problem characteristics can be identified directly from the problem formulation, it is possible to create an algorithm that combines the strength of both formulations.

9. Conclusions and further research

In this paper we introduced a model of the mid-term production planning problem for the high-tech low-volume industry. The model has semi-flexible capacity constraints, which give extra planning flexibility and force the division of the capacity claims over multiple time slots if necessary. Furthermore, the model allows different production modes for the production of an item, which also increases the planning flexibility. These modes differ in lead time, resource requirement or assembly sequence.

Because of the integer production variables, size of realistic instances and complexity of the model, regular branch and bound methods have trouble solving this model. We applied Benders'
decomposition to get an alternative formulation of the problem. Formulation 2 contains an exponential number of capacity constraints. So, we developed Algorithm 2 to solve this model without adding all constraints. The algorithm checks for each new incumbent if any capacity constraints are violated. If this is the case the constraints are added and the solver is restarted. A maximum flow algorithm is used to find the missing constraints. Results from a realistic test instance show that utilizing Algorithm 2 to solve Formulation 2 is significantly faster than solving Formulation 1 with standard branch and bound.

Furthermore, we compared our methods with the approach in Spitter et al. (2005). Formulation 1 and the model of Spitter et al. (2005) are identical under some restrictive assumptions, e.g. simplified resource usage and no different production modes. We showed that under these assumptions the model of Spitter et al. (2005) outperforms Formulation 2 when the capacity availability is constant over time or the capacity utilization is high.

The computational study shows that there are still many realistic instances that cannot be solved to optimality fast. Hence, further research should focus on further reducing the solution time of these models. Another remaining question is how to set the parameters, e.g. lead times and resource usage parameters, to maximize the performance of the rolling horizon approach for high-tech low-volume supply chains.

Acknowledgements

We are grateful to H.J.H. Gort and M.H.E. Welters for developing the basis of the model and giving valuable input from practice to this research. Additionally, we are thankful to two anonymous reviewers, whose critical and detailed comments enabled us to significantly improve this paper. This research was funded by ASMLs Research and Development Framework, which is an agreement between Eindhoven University of Technology, Delft University of Technology, University of Twente and ASML Netherlands B.V..

Appendix: Proof of NP-hardness

**Theorem.** Formulation 1 describes a NP-hard problem.

**Proof.** Consider an instance of the Multi-dimensional knapsack problem. Thus we have a set of items \( i \), which have a value \( v_i \) and a size in \( D \) dimensions: \( \bar{w}_i = (w_{i1},...,w_{iD}) \). Furthermore, we have a \( D \)-dimensional knapsack with a capacity of \( \bar{W} = (W_1,...,W_D) \). We want to choose a subset of the items such that the sum of the values of these items is maximized and the sum of the sizes is at most the capacity of the knapsack in each dimension.

Construct an instance of Formulation 1 with the same items, \( D \) resources and two time slots. The lead time of each item is one. The demand at time 1 is one for each item and the cost for
having backlog of an item $i$ is $v_i$. Note that you have to pay this backlog cost in the second time slot if you do not produce this item in the first time slot. Resource $u$ has capacity $W_u$ in the first time slot and item $i$ requires exactly $w_{iu}$ of resource $u$ in the first and only time slot of production. All other parameters are zero.

Clearly, the items Formulation 1 will choose to produce correspond with the items you would choose in your knapsack. Hence, if it would be possible to solve Formulation 1 in polynomial time it would also be possible to solve the Multi-dimensional knapsack problem in polynomial time. Since the Multi-dimensional knapsack problem problem is NP-hard (Chekuri & Khanna (2004)), we conclude that Formulation 1 describes a NP-hard problem.

References


