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:
Author’s version before peer-review

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.

Download date: 26. May. 2021
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. Our MILP models can handle general supply chains and production processes that require multiple resources. Furthermore, they support semi-flexible capacity constraints and multiple production modes.

First, we introduce a model that assigns resources explicitly to release orders. Resulting in a second model, we introduce alternative capacity constraints, which assure that the available capacity in any subset of the planning horizon is sufficient. Since the number of these constraints is exponential we first solve the second model 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 model is significantly faster than solving the first model.

Keywords: Production, Integer Programming, Branch and Bound

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, Meyr & Wagner, 2005). 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, Rohde & Wagner, 2015). More specifically, we study the mathematical programming models that arise when a rolling scheduling approach is applied, which is common in both literature and practice (c.f. Silver, Pyke & Peterson, 1998).

There is a vast body of literature on mid-term production planning (c.f. Bertrand, Wortmann & Wijngaard (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, 2005). 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. First we introduce two models for the mid-term production planning problem for high-tech low-volume industries and, since the problem is NP-hard, we subsequently focus on solving these models.

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)). I.e. 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, Hurkens, de Kok, Lenstra & Negenman, 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 of which the order is critical.

Besides the semi-flexible capacity constraints we introduce another source of planning flexibility: production modes. These modes to produce an item can differ in lead time, resource requirements and assembly sequences. A longer lead time and different resource requirements give more flexibility in the resource allocation. Because of this the available capacity can be used more efficiently. In periods of lower resources availability (e.g. during holidays), inventory and backlog costs can thus be saved since fewer items have to be produced early or late. Similarly, in case of a material shortage an assembly sequence in which the missing component is required later saves a lot of time.

The models for high-volume production planning use continuous production quantity variables (c.f. Buschkühl, Sahling, Helber & Tempelmeier, 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 indus-
tries to be discrete and, considering the similarities between choosing which items to produce using the available capacity and the knapsack problem, even NP-hard. However, in practise 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.

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 and in section 4 we introduce alternative capacity constraints and a second model, which uses these constraints. Section 5 proves the equivalence of the models described in section 3 and 4. Section 6 describes an algorithm that solves the second model. In section 7 both models are compared against a set of test cases. Section 8 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 \( s = (s_1, s_2] \). \( D_{it} \) and \( G_{it} \) represent the independent and dependent 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 \( m \). 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 \( p_{itm}^{\text{total}} \) of resource \( u \) is required. In time slot \( q \) of the production of an item \( i \) in mode \( m \) at least \( p_{itm}^{\text{min}} \) of resource \( u \) is required and at most \( p_{itm}^{\text{max}} \) of resource \( u \) can be claimed for the production of this item. To summarize we define the following parameters for our models:

- \( n \): \( n \in \mathbb{N} \), the number of different items, which are labeled \( 1, ..., n \).
- \( k \): \( k \in \mathbb{N} \), the number of resources, which are labeled \( 1, ..., k \).
- \( M_i \): \( M_i \subset \mathbb{N} \), the set of production modes for item \( i \), \( i = 1, ..., n \).
- \( \tau_{im} \): \( \tau_{im} \in \mathbb{N} \), \( i = 1, ..., n \), 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 \( t = 0, ..., T \) and time slots \( s \), defined as \( (s_1, s_2] \), \( s = 1, ..., T \). The past decision points \( t = -\tau_{im}, ..., -1 \) and time slots \( s = -\tau_{im} + 1, ..., 0 \) are also considered.
- \( D_{it} \): \( D_{it} \in \mathbb{N} \), the independent demand for item \( i \) at time \( t \), \( i = 1, ..., n \), \( t = 0, ..., T \).
- \( h_{ij} \): \( h_{ij} \in \mathbb{N} \), the number of units of item \( i \) needed for the production of a single unit of item \( j \), \( i = 1, ..., n \), \( j = 1, ..., n \).
- \( \delta_{ijm} \): \( \delta_{ijm} \in \mathbb{N} \), the number of time slots after the release moment of item \( j \), component \( i \) is required for production of \( j \), in production mode \( m \), \( i = 1, ..., n \), \( j = 1, ..., n \), \( m \in M_j \).
\[ \alpha_{it} \geq 0, \text{ the costs of keeping one single unit of item } i \text{ on stock during } (t, t+1), \]
\[ i = 1, \ldots, n, t = 0, \ldots, T. \]
\[ \beta_{it} \geq 0, \text{ the costs of delaying the delivery of one single unit of item } i \text{ from time } t \text{ to time } t+1, i = 1, \ldots, n, t = 0, \ldots, T. \]
\[ \gamma_{itm} \geq 0, \text{ the extra costs occurring from releasing one single unit of item } i \text{ at time } t \]
\[ \text{with production mode } m, i = 1, \ldots, n, t = 1, \ldots, T - \tau_{im}, m \in M_i. \]
\[ \mathcal{R}_{im} : \text{ The set of resources that are used to produce item } i \text{ in mode } m, i = 1, \ldots, n, m \in M_i. \]
\[ \mathcal{J}_u : \text{ The set of items and their production modes } (i, m) \text{ that require resource group } u \]
\[ \text{during production, } u = 1, \ldots, k. \]
\[ c_{us} \geq 0, \text{ the maximum amount available of resource group } u \text{ in time slot } s, u = 1, \ldots, k, \]
\[ s = 1, \ldots, T. \]
\[ p_{iqum}^{\min} \geq 0, \text{ the minimum amount of resource } u \text{ required in time slot } t + q \text{ for the} \]
\[ \text{production order of one item } i \text{ released at time } t \text{ with production mode } m, i = 1, \ldots, n, \]
\[ m \in M_i, q = 1, \ldots, \tau_{im}, u \in \mathcal{R}_{im}. \]
\[ p_{iqum}^{\max} \geq 0, \text{ the maximum amount of resource } u \text{ that can be allocated in time slot } t + q \]
\[ \text{to the production order of one item } i \text{ released at time } t \text{ with production mode } m, \]
\[ i = 1, \ldots, n, m \in M_i, q = 1, \ldots, \tau_{im}, u \in \mathcal{R}_{im}. \]
\[ p_{iut}^{\max} \geq 0, \text{ the total amount of resource } u \text{ required to produce one item } i \text{ released} \]
\[ \text{with production mode } m, i = 1, \ldots, n, m \in M_i, u \in \mathcal{R}_{im}. \]
\[ \mathcal{R}_{it} : \mathcal{R}_{it} \in \mathbb{N}, \text{ the size of the work order in progress of item } i \text{ released at time } t \]
\[ \text{with production mode } m, i = 1, \ldots, n, m \in M_i, t = -\tau_{im}, \ldots, -1. \]
\[ T_{i-1} : T_{i-1} \in \mathbb{N}, \text{ the inventory level of item } i \text{ at time } -1, i = 1, \ldots, n. \]
\[ \mathcal{B}_{i-1} : \mathcal{B}_{i-1} \in \mathbb{N}, \text{ the size of the backlog of item } i \text{ at time } -1, i = 1, \ldots, n. \]

Furthermore, the decision variables used in both models are:
\[ I_{it} : \text{ The inventory of item } i \text{ during time interval } (t, t+1), i = 1, \ldots, n, t = 0, \ldots, T. \]
\[ B_{it} : \text{ The backlog of item } i \text{ at time } t, i = 1, \ldots, n, t = 0, \ldots, T. \]
\[ G_{it} : \text{ The dependent demand of item } i \text{ at time } t, i = 1, \ldots, n, t = 0, \ldots, T. \]
\[ R_{itm} : \text{ The size of the work order of item } i \text{ released at time } t \text{ with production mode } m, \]
\[ i = 1, \ldots, n, t = -\tau_{im}, \ldots, T - \tau_{im}, m \in M_i. \]

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

3. A mixed integer linear programming model

Our first model is based on the Extended model in Gort (2013) and explicitly assigns resources to releases, i.e. it 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 \( m \). This is also the approach of the model with balance equations in Spitter et al. (2005). If the assignment of resources to releases is known, checking the constraints on the resource usage is straightforward. For this purpose we introduce a variable and a corresponding parameter:
\[ Z_{its} : \text{ The part of the planned work, related to work order release } R_{itm}, \text{ executed in time slot} \]
\[ s \text{ on resource group } u, i = 1, \ldots, n, m \in M_i, t = -\tau_{im}, \ldots, T - \tau_{im}, s = t + 1, \ldots, t + \tau_{im}, \]
\[ u \in \mathcal{R}_{im}. \]
\[ \bar{Z}_{its} : \text{ The part of the planned work, related to work order release } \bar{R}_{itm}, \text{ already executed in} \]
\[ \text{time slot } s \text{ on resource group } u, i = 1, \ldots, n, m \in M_i, t = -\tau_{im}, \ldots, -1, s = t + 1, \ldots, 0, \]
\[ u \in \mathcal{R}_{im}. \]
Now the mixed integer linear programming formulation is as follows:

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

subject to

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

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

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

\[
\min \left\{ p_{i,u,m} R_{itim} : \sum_{s=t}^{t+\tau_{im}} Z_{itsum} \leq \sum_{s=t+1}^{t+\tau_{im}} R_{itim} \right\} \leq \max \left\{ p_{i,u,m} R_{itim} : \sum_{s=t}^{t+\tau_{im}} Z_{itsum} \leq \sum_{s=t+1}^{t+\tau_{im}} R_{itim} \right\} \quad i = 1, \ldots, n, m \in M_i, u \in \mathcal{R}_{im},
\]

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

\[
\sum_{i=1}^{n} \sum_{m \in M_i} \sum_{s=t}^{s-1} \sum_{t=s-\tau_{im}} Z_{itsum} \leq c_{us} \quad u = 1, \ldots, k, s = 1, \ldots, T \tag{5}
\]

\[
I_{i,-1} = I_{i,-1}, B_{i,-1} = B_{i,-1} \quad i = 1, \ldots, n
\]

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

\[
Z_{itsum} = Z_{itsum} \quad i = 1, \ldots, n, m \in M_i, u \in \mathcal{R}_{im},
\]

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

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

\[
Z_{itsum} \geq 0 \quad i = 1, \ldots, n, m \in M_i, u \in \mathcal{R}_{im},
\]

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 (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) guarantees 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

The capacity constraints of the first model explicitly assign the available resources to work orders. In this section we describe alternative capacity constraints, which where introduced in de Kruijff (2014) and are essentially Benders’ cuts (Benders, 1962). They are equivalent to the capacity constraints in the first model, but do not explicitly assign the available resources to work orders and thus require less variables. Since the goal of mid-term production planning is to determine a feasible release plan while the exact resource allocation is done at a lower planning level, this is sufficient. The alternative capacity constraints are knapsack constraints, which modern solvers use to eliminate non-integer solutions of the LP relaxation to find higher lower bounds faster.

The alternative capacity constraints compare, inspired by Hall’s marriage theorem (Hall, 1935), the available resources in a set of time slots with the minimum required amount of resources. Therefore, we define \( \chi_{itm u \sigma} \) 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 \( m \). Then

\[
\sum_{itm} \chi_{itm u \sigma} R_{itm} \leq \sum_{s \in \sigma_u} c_{us}
\]

should hold for all resources \( u \) and sets of time slots \( \sigma_u \), since the left side is the minimum amount of resource \( u \) in \( \sigma_u \) that is required for the production of all released orders and the right side is the amount of resource \( u \) available in \( \sigma_u \).

\( \chi_{itm u \sigma} \) is determined by the minimum amount of resources required in the time slots in \( \sigma_u \) or by the total amount of resources required and the maximal amount of resources that can be allocated to this release in time slots outside \( \sigma_u \). Thus

\[
\chi_{itm u \sigma} = \max \left( \sum_{q:t+q \in \sigma_u} p_{qum}^{\min} p_{i um}^{\text{total}} - \sum_{q:t+q \notin \sigma_u} p_{qum}^{\max} \right)
\]

for \( t \geq 0 \). Figure 1 depicts three examples of the determination of \( \chi_{itm u \sigma} \). 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 slot 4 and 5, only the required 4 units have to be claimed in time slot 3. This leads that \( \chi_{i,2,1,u,\sigma_u} = 4 \). For \( t < 0 \), \( \chi_{itm u \sigma} \) should be determined based on the progress of the production process, i.e. the total amount of resources required, \( p_{i um}^{\text{total}} \) in the above formula, should be adjusted to the amount of work that still has to be carried out.
Figure 1: Examples of the calculation of $\chi_{i,t,u,\sigma_u}$, with $\sigma_u = \{1, 2, 3\}$ and $p_{iqum}^{\text{total}} = 20$, $p_{iqum}^{\text{max}} = 10$ and $p_{iqum}^{\text{min}} = 10$ or $p_{iqum}^{\text{min}} = 4$ for each time slot $q$. 

- For $R_{i,1,0}$, $p_{iqum}^{\text{total}} = 20$, $p_{iqum}^{\text{min}} = 10$, $p_{iqum}^{\text{max}} = 10$, and $\chi_{i,1,0,u,\sigma_u} = 20$.
- For $R_{i,1,1}$, $p_{iqum}^{\text{total}} = 20$, $p_{iqum}^{\text{min}} = 4$, $p_{iqum}^{\text{max}} = 10$, and $\chi_{i,1,1,u,\sigma_u} = 10$.
- For $R_{i,2,1}$, $p_{iqum}^{\text{total}} = 20$, $p_{iqum}^{\text{min}} = 4$, $p_{iqum}^{\text{max}} = 10$, and $\chi_{i,2,1,u,\sigma_u} = 4$. 


We use these inequalities as capacity constraints and combine them with the inventory balance equations to create Model 2. Besides the parameters we introduced earlier, Model 2 uses the following parameters 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 set of sets \( \sigma_u \) of time slots in \([1, T]\), used to make sure that there is enough capacity of resource group \( u \), \( u = 1, \ldots, k \).

\( \chi_{itmu\sigma_u} \): The minimal amount of resource \( u \), required in \( \sigma_u \) by the release of one item \( i \) at time \( t \) with production mode \( m \), \( i = 1, \ldots, n \), \( m \in M_i \), \( t = -\tau_{im}, \ldots, T - \tau_{im} \), \( u \in R_{im} \), \( \sigma_u \in \Sigma_u \).

The mixed integer linear programming formulation of Model 2 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

\[ I_{it} = I_{i,t-1} + \sum_{m \in M_i} R_{i,t-\tau_{im},m} - D_{it} - G_{it} + B_{it} - B_{i,t-1} \quad i = 1, \ldots, n, t = 0, \ldots, T \quad (7) \]

\[ G_{it} = \sum_{j=1}^{n} \sum_{m \in M_j} h_{ij} R_{j,t-\delta_{ijm},m} \quad i = 1, \ldots, n, t = 0, \ldots, T \quad (8) \]

\[ B_{it} - B_{i,t-1} \leq D_{it} \quad i = 1, \ldots, n, t = 0, \ldots, T \quad (9) \]

\[ \sum_{(i,m) \in \mathcal{I} u, t: [t+1, t+\tau_{im}] \cap \sigma_u \neq \emptyset} \chi_{itmu\sigma_u} R_{itm} \leq \sum_{s \in \sigma_u} c_{us} \quad \forall \sigma_u \in \Sigma_u, u = 1, \ldots, k \quad (10) \]

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

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

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

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

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 (7) is the inventory balance equation, which models the flow of goods and enforces the satisfaction of demand. Constraint (8) determines the dependent demand. Constraint (9) ensures that backlog is only used for external demand. Constraint (10) 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.

In the next section we prove that the capacity constraints are equivalent if \( \Sigma_u \) would contain all subsets of \([1, T]\) and thus that Model 2 is equivalent to Model 1. Note that in this case the number of constraints in Model 2 is exponential in \( T \). However, in general not all constraints are necessary to find feasible (and optimal) solutions. In section 6 we present an algorithm to find the necessary constraints and solve Model 2.
5. Equivalence of the models

This section proves the equivalence of Model 1 and Model 2, i.e. the equivalence of their capacity constraints. First we introduce the Resource assignment graph, which is a directed graph that can be used to determine if the capacity constraints of Model 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 models.

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 $R_{itm}^{p_{\text{total}}}$ 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 $R_{itm}^{p_{\text{max}}}_{i,s-t,u,m}$ and the minimum flow is of size $R_{itm}^{p_{\text{min}}}_{i,s-t,u,m}$.

![Resource assignment graph](image)

Figure 2: A Resource assignment graph. The maximum flow in this graph is of size $\sum_{itm} R_{itm}^{p_{\text{total}}}$ 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 & Fulkerson (1955)) in this graph is of size $\sum_{itm} R_{itm}^{p_{\text{total}}}$ 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 $\sum_{itm} R_{itm}^{p_{\text{total}}}$ if and only if the capacity constraints of Model 1 hold for resource $u$.

**Proof.** Suppose there is a flow of size $\sum_{itm} R_{itm}^{p_{\text{total}}}$ and let $Z_{\text{itsum}}$ 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 $R_{itm}^{p_{\text{total}}}_{i,t,u,m}$. Thus the constraint $\sum_{s=t+1}^{t+\tau_{im}} Z_{\text{itsum}} = p_{\text{total}}^{\text{sum}} R_{itm}$ is satisfied, by the flow conservation in node $(i, t, m)$. By the flow conservation in node $s$ constraint $\sum_{i=1}^{n} \sum_{m \in M_i} \sum_{t=s-\tau_{im}}^{s-1} Z_{\text{itsum}} \leq c_{us}$ 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 $\sum_{i=1}^{n} \sum_{m \in M_i} \sum_{t=s-\tau_{im}}^{s-1} Z_{\text{itsum}} \leq c_{us}$ is satisfied.
satisfied. Thus the capacity constraints of Model 1 for resource \( u \) hold if the maximum flow is of size \( \sum_{itm} R_{itmu}^{total} \).

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

We use this lemma to prove that Model 1 and Model 2 are equivalent.

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

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

If there is no feasible flow in the corresponding Resource assignment graph, then \( \sum_{itm} R_{itmu}^{total} \) is not empty. Thus the constraint \( \sum_{s/\in\sigma_u} F_{itms} \leq \sum_{s/\in\sigma_u} c_{us} \) is violated for \( \sigma_u = \{ s \} \). If there is a feasible flow, but the maximum flow is not of size \( \sum_{itm} R_{itmu}^{total} \), define \( F \) as the maximum flow, \( F_{itms} \) as the flow from the source to \( (i,t,m) \), \( F_{itmu} \) 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 \( \sum_{itm} R_{itmu}^{total} \), there is at least one \((i,t,m)\) such that \( F_{itms} < R_{itmu}^{total} \) and for this \((i,t,m)\) there is at least one \( s \) such that \( F_{itms} < R_{itmu}^{max} \), hence \( \sigma_u \) is not empty.

Now we prove that \( \sum_{itm} \chi_{itmu} R_{itms} \leq \sum_{s/\in\sigma_u} c_{us} \) is violated for this \( \sigma_u \). To do this we show \( \sum_{itm} \sum_{s/\in\sigma_u} R_{itmu} \leq \sum_{s/\in\sigma_u} c_{us} \) is violated for this \( \sigma_u \). This is enough since \( \sum_{itm} \sum_{s/\in\sigma_u} F_{itms} = \sum_{s/\in\sigma_u} c_{us} \), 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 \( F_{itms} < R_{itmu}^{total} \) holds and such \( \sum_{s/\in\sigma_u} F_{itms} < R_{itmu}^{total} \). Note that \( s \notin \sigma_u \) implies \( F_{itms} = p_{i,s-t,u,m}^{max} \) and thus \( \sum_{s/\in\sigma_u} F_{itms} < R_{itmu}^{total} \).

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

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

The number of capacity constraints of Model 2 is exponential in the size of the planning horizon. Since mid-term production planning requires a considerable planning horizon, solving Model 2 with all its capacity constraints is not viable. Hence we developed an algorithm that adds violated capacity constraints while solving the model without all its capacity constraints. First we describe how to find a violated constraint given a set of release orders and then present the whole algorithm.

In the previous section we proved that the maximum flow over a Resource assignment graph is of size $\sum_{itm} R_{itm} p_{itm}^{total}$ 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 $\sum_{itm} \chi_{itmu \sigma} R_{itm} \leq \sum_{s \in \sigma_u} c_{us}$ 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.

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 $m$ $\chi_{itmu \sigma_1} = \chi_{itmu \sigma}$ for a subset $\sigma_1$. 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 $\sigma_u$.

Algorithm 2 solves Model 2 by using Algorithm 1 to find missing capacity constraints. It starts with Model 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.

**Input:** Model 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 $k$ do
    Use Algorithm 1 to find $\sigma_u$ given $u$ and the solution of the LP relaxation;
    if $\sigma_u \neq \emptyset$ then
      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 $k$ do
        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
  end
  for $u = 1$ to $k$ do
    if $\sigma_u \neq \emptyset$ then
      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 Model 2;

Algorithm 2: An algorithm to solve Model 2.

To add the missing constraints as global cuts and continue the branch and bound process might seem a more natural approach. However, we implemented and tested this approach and it was
clearly slower than Algorithm 2. Apparently, a good solution that violates a capacity constraint is not close to the optimal solution in the branch tree. Furthermore, the branch and bound process makes notably better decisions when one important missing constraint is added.

7. Computational comparison

In sections 3 and 4 we introduced two models for the mid-term production planning problem for high-tech low-volume supply chains. Model 1 can be solved using branch and bound. For Model 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 inspired by an instance of the supply chain planning at a high tech company. The instance contains 104 time slots and 32 items divided over 3 levels. The supply chain is converging and contains 4 end items. Each end item has one component at level 2 and these components each have 6 components at level 3. The number of resources varies per test case. An overview of the main characteristics of the test cases can be found in Table 1.

<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>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of items</td>
<td>4</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>Number of components per item</td>
<td>1</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Total demand</td>
<td>272</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Number of production modes</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead time</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>Demand per time slot</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Total demand</td>
<td>12</td>
<td>140</td>
</tr>
<tr>
<td>Total required of resource 1</td>
<td>208</td>
<td>945</td>
</tr>
<tr>
<td>Total required of resource 2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total required of resource 3</td>
<td>150</td>
<td>300</td>
</tr>
<tr>
<td>Total required of resource 4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: Main characteristics of the test cases.

The end items and their components each have 4 production modes, which have different lead times. The first production mode has the shortest lead time and the fourth production mode the longest. The end item with the shortest lead time has a lead time of 5 time slots in the first production mode and of 8 time slots in the fourth production mode. Similarly, the lead time of the end item with the longest lead time varies from 12 to 15, the lead time of the level 2 item with the longest lead time from 6 to 9 and with the shortest lead time from 3 to 6. The items on the third level do not have different production modes and have lead times of 1 or 2 time slots.

The costs of inventory and backlog depend on the value of the item and the cost of backlog is 10 times the cost of inventory. Having the item with the lowest value in inventory for one week costs for example 1 and the value of an item is higher than the value of its components. The cost
of producing in a certain mode depends on the lead time and the value of the item, i.e. the costs of producing with a certain lead time $\tau_{im}$ is equal to producing with lead time $\tau_{im} - 1$ plus the cost of having this item 1 time slot in inventory. The demand follows the pattern of a real demand scenario and in total there is demand for 272 end-items.

In the first test case we included one resource, which is required for the production of end items and their components. In the first production mode there is no flexibility in the assignment of this resource to the time slots of production. However, in the other production modes, which have a longer lead time, there is. In the second test case we added a second resource, which is required for three of the items at level 3. In the third test case we added a third resource, which is required for the production of end items. And in the forth test case we added a forth resource, which is required for the production of the items at level 2. Of these resources a fixed amount is required, i.e. there is no flexibility in the assignment of these resources to time slots of production. An example of the resource usage parameters of an end item and its component can be found in Table 2.

<table>
<thead>
<tr>
<th>Time slot</th>
<th>Example end item in mode 1</th>
<th>Example component in mode 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Resource 1 minimum</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Resource 1 maximum</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>Resource 2 total</td>
<td>208</td>
<td></td>
</tr>
<tr>
<td>Resource 3 minimum</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Resource 3 maximum</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Resource 3 total</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>Resource 4 minimum</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Resource 4 maximum</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Resource 4 total</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Example of resource usage of an end item and its component.

Each of the four test cases contains 15 scenarios, which differ in the available capacity of the first resource. In all time slots the available capacity is the same, except in four holiday periods. Two of the holiday periods span 1 time slot and two of the holiday periods span 6 time slots. Both the normal capacity and the holiday capacity of the first resource are varied. The capacity of the other resources is constant over time and over all scenarios. The capacity of the first resource is chosen such that it is a bottleneck during the holidays. The capacity of the other resources is chosen such that they are only bottlenecks if the need for these resources is clearly higher than average. During the test cases we saw that the impact of the fourth resource was significant, while the impact of the third resource was rather small.

The models and algorithms are implemented using AIMMS 4.15. 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 Model 2 with all sets of length one for the second, third and fourth resource and no sets for the first resource. Note that for the second, third and fourth 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. Model 1 was solved with CPLEX 12.6.3 and Model 2 was solved using Algorithm 2 and CPLEX 12.6.3. Both models were given a maximal run time of 900 seconds. The results of the test scenarios can be found in table 4, 5, 6 and 7. An summary of these results is given in table 3.

<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>Model 1</td>
<td>Model 2</td>
<td>Model 1</td>
<td>Model 2</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>34</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimal solution</th>
<th>Average run time</th>
<th>Average gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>Model 2</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>7</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>19</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>34</td>
</tr>
</tbody>
</table>

Table 3: Summary of the test results.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Model 1</th>
<th>Model 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Run time (seconds)</td>
<td>Best found solution</td>
</tr>
<tr>
<td>Normal capacity</td>
<td>Holiday capacity</td>
<td>32</td>
</tr>
<tr>
<td>3300</td>
<td>2200</td>
<td>900</td>
</tr>
<tr>
<td>3150</td>
<td>2100</td>
<td>360</td>
</tr>
<tr>
<td>3100</td>
<td>1575</td>
<td>900</td>
</tr>
<tr>
<td>3000</td>
<td>2000</td>
<td>900</td>
</tr>
<tr>
<td>3000</td>
<td>1500</td>
<td>900</td>
</tr>
<tr>
<td>3000</td>
<td>1000</td>
<td>380</td>
</tr>
<tr>
<td>2850</td>
<td>1900</td>
<td>900</td>
</tr>
<tr>
<td>2850</td>
<td>1425</td>
<td>900</td>
</tr>
<tr>
<td>2850</td>
<td>950</td>
<td>900</td>
</tr>
<tr>
<td>2700</td>
<td>1800</td>
<td>900</td>
</tr>
<tr>
<td>2700</td>
<td>1350</td>
<td>900</td>
</tr>
<tr>
<td>2700</td>
<td>900</td>
<td>900</td>
</tr>
</tbody>
</table>

Table 4: Comparison of the solution time of Model 1 and Model 2 on a test case with one resource.

We see that in 7 scenarios both models solve to optimality. In these scenarios the average solution time for Model 1 is 237 seconds and for Model 2, 84 seconds. There are 34 scenarios for which no optimal solution is found for both models. In these scenarios the average optimality gap was 0.80% for Model 1 and 0.53% for Model 2. Furthermore, there are 19 scenarios in which Model 2 was solved within 900 seconds, but Model 1 was not. The average solution time for Model 1 is thus bigger than 900 seconds, where the average solution time for Model 2 is only 206 seconds in these scenarios. Furthermore we see that for Model 2 in 13 scenarios a better solution was found.
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Model 1</th>
<th>Model 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Run time</td>
<td>Best found solution</td>
</tr>
<tr>
<td>Normal Holiday capacity</td>
<td>(seconds)</td>
<td>(seconds)</td>
</tr>
<tr>
<td>3300</td>
<td>23</td>
<td>276</td>
</tr>
<tr>
<td>3300</td>
<td>900</td>
<td>7642</td>
</tr>
<tr>
<td>3300</td>
<td>900</td>
<td>16614</td>
</tr>
<tr>
<td>3150</td>
<td>900</td>
<td>1270</td>
</tr>
<tr>
<td>3150</td>
<td>900</td>
<td>8764</td>
</tr>
<tr>
<td>3150</td>
<td>900</td>
<td>17495</td>
</tr>
<tr>
<td>3000</td>
<td>900</td>
<td>2618</td>
</tr>
<tr>
<td>3000</td>
<td>900</td>
<td>9914</td>
</tr>
<tr>
<td>3000</td>
<td>900</td>
<td>18404</td>
</tr>
<tr>
<td>2850</td>
<td>900</td>
<td>3970</td>
</tr>
<tr>
<td>2850</td>
<td>900</td>
<td>11179</td>
</tr>
<tr>
<td>2850</td>
<td>900</td>
<td>19668</td>
</tr>
<tr>
<td>2700</td>
<td>900</td>
<td>5374</td>
</tr>
<tr>
<td>2700</td>
<td>900</td>
<td>12589</td>
</tr>
<tr>
<td>2700</td>
<td>900</td>
<td>22334</td>
</tr>
</tbody>
</table>

Table 5: Comparison of the solution time of Model 1 and Model 2 on a test case with two resources.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Model 1</th>
<th>Model 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Run time</td>
<td>Best found solution</td>
</tr>
<tr>
<td>Normal Holiday capacity</td>
<td>(seconds)</td>
<td>(seconds)</td>
</tr>
<tr>
<td>3300</td>
<td>25</td>
<td>276</td>
</tr>
<tr>
<td>3300</td>
<td>900</td>
<td>7639</td>
</tr>
<tr>
<td>3300</td>
<td>900</td>
<td>16614</td>
</tr>
<tr>
<td>3150</td>
<td>900</td>
<td>1270</td>
</tr>
<tr>
<td>3150</td>
<td>900</td>
<td>8764</td>
</tr>
<tr>
<td>3150</td>
<td>900</td>
<td>17490</td>
</tr>
<tr>
<td>3000</td>
<td>900</td>
<td>2597</td>
</tr>
<tr>
<td>3000</td>
<td>900</td>
<td>9912</td>
</tr>
<tr>
<td>3000</td>
<td>900</td>
<td>18404</td>
</tr>
<tr>
<td>2850</td>
<td>900</td>
<td>3968</td>
</tr>
<tr>
<td>2850</td>
<td>900</td>
<td>11179</td>
</tr>
<tr>
<td>2850</td>
<td>900</td>
<td>19668</td>
</tr>
<tr>
<td>2700</td>
<td>900</td>
<td>5374</td>
</tr>
<tr>
<td>2700</td>
<td>900</td>
<td>12591</td>
</tr>
<tr>
<td>2700</td>
<td>900</td>
<td>22332</td>
</tr>
</tbody>
</table>

Table 6: Comparison of the solution time of Model 1 and Model 2 on a test case with three resources.
Table 7: Comparison of the solution time of Model 1 and Model 2 on a test case with four resources.

than for Model 1, while vice versa was the case in only 7 scenarios. We conclude that for these test cases solving Model 2 using Algorithm 2 is substantially faster than solving Model 1.

An observation is that the difference in performance of the two models is less clear in the test cases with more resources. This can be explained by the fact that there is no flexibility in the usage of resources 2, 3, and 4, which implies that the capacity constraints of both models are exactly the same for these resources. Furthermore, Model 2 performs relatively poor in the last scenario of all test cases. For this scenario we observed that the algorithm adds by far the most constraints.

8. Conclusions and further research

In this paper we introduced two equivalent models of the mid-term production planning problem for the high-tech low-volume industry. The models have 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 models allow 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.

Model 1 assigns resources explicitly to release orders. For Model 2 we developed alternative capacity constraints. Since this model contains an exponential number of constraints, 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 Model 2 is significantly faster than solving Model 1 with standard branch and bound.

Furthermore, the test instance 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 first model and giving valuable input from practice to this research. 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..

References


