An SKU Decomposition Algorithm for the Tactical Planning in the FMCG Industry

M.A.H. van Elzakker[a*], E. Zondervan[a], N.B. Raikar[b], H. Hoogland[b], I.E. Grossmann[c]

[a] Dept. Chem. Eng. And Chem., Eindhoven University of Technology, P.O. Box 513, 5600MB Eindhoven, the Netherlands

[b] Unilever R&D Vlaardingen, the Netherlands

[c] Dept. Chem. Eng., Carnegie Mellon University, Pittsburgh, USA

[*] Corresponding Author: M.A.H.v.elzakker@tue.nl, +31 402475807

Abstract

In this paper we address the optimization of the tactical planning for the Fast Moving Consumer Goods (FMCG) industry, in which numerous trade-offs need to be considered over possibly thousands of Stock-Keeping Units (SKUs). An MILP model for the optimization of this tactical planning problem is proposed. This model is demonstrated for a case containing 10 SKUs, but is intractable for realistically sized problems. Therefore, a decomposition algorithm based on decomposing the model into single-SKU submodels is proposed in this paper. To account for the interaction between SKUs, slack variables are introduced into the capacity constraints. These slack variables initially allow the capacity to be violated. In an iterative procedure the cost of violating the capacity is slowly increased, and eventually a feasible solution is obtained. Even for the relatively small 10 SKU case, the required CPU time could be reduced from 4427s to 472s using the algorithm. Moreover, the algorithm was used to optimize cases of up to 1000 SKUs, whereas the full model is intractable for cases of 25 or more SKUs. The solutions obtained with the algorithm are typically within a few percent of the global optimum.

Keywords: Tactical Planning, Optimization, MILP, Decomposition Algorithm, Fast Moving Consumer Goods

1. Introduction

The scale and complexity of enterprise-wide supply chains has increased significantly due to globalization. (Varma et al., 2007) Recently, the operation of enterprise-wide supply chains has attracted much interest. Grossmann (2005) and Varma et al. (2007) review the current research on Enterprise-wide Optimization (EWO), and they identify challenges and research opportunities. One of the main challenges is the integration of decision-making across various layers. This includes the integration of the various echelons of the supply chain and the integration of the various temporal decisions layers. The decisions on the various layers are often interconnected leading to trade-offs between these decisions. (Maravelias and Sung, 2009) Therefore, better solutions can be obtained if these decisions are optimized simultaneously.

Usually, three temporal decision layers are distinguished: strategic planning, tactical planning and operational planning. Strategic planning covers the long-term decisions regarding the design of the supply chain. Tactical planning covers the medium-term decisions regarding the allocation of capacity. Operational planning covers the short-term scheduling decisions.

Maravelias and Sung (2009) review the integration of short-term scheduling and tactical production planning. They identify two options for this integration. First, the detailed scheduling decisions can directly be included into the tactical planning model. While this would in theory yield optimal solutions, the resulting models are usually very large and difficult to solve.

Therefore, advanced solution strategies are often applied to solve larger problems. For example, Erdirik-Dogan and Grossmann (2007) developed a bi-level decomposition strategy to solve larger instances of their integrated scheduling and tactical planning model for a single plant. In addition, they modeled the sequence-dependent changeovers more efficiently by using constraints based on the traveling salesman problem. Terrazas-Moreno and Grossmann (2011) extended this model and bi-level decomposition method to a multi-site setting. In addition, they proposed a new hybrid decomposition method that combines bi-level and spatial Langragean decomposition. This hybrid method proved to be the most efficient for large-scale problems.

The second approach is to approximate the scheduling decisions by removing or relaxing part of the constraints or by aggregating some of the decisions. For example, Sung and Maravelias (2007) consider the restrictions found on the short-term scheduling level by incorporating linear surrogate constraints into the tactical planning model. These surrogate constraints are a convex approximation of the feasible region of the scheduling model projected in the space of production amounts of the products. They obtain these surrogate constraints by analyzing an MIP scheduling model off-line.

The tactical planning problem considered in this paper is already extremely large by itself. Therefore, we have chosen the second method and approximate the scheduling decisions as close as possible. While this will not give us a detailed weekly production schedule, the weekly production targets will be realistic.

In this paper, we consider the tactical planning for a Fast Moving Consumer Goods (FMCG) company. FMCG are products that are sold in large quantities, that have a relatively low profit margin and that, if not available, are quickly substituted by a competitor's product. Some examples of FMCG are ice cream, yogurt and shampoo. The production process in the FMCG industry is typically a make-and-pack production process. (Bilgen and Gunther, 2010)

Fast Moving Consumer Goods (FMCG) companies produce a wide range of products to satisfy an increasing demand for product variety. (Bilgen and Gunther, 2010) Even a single product category, such as ice cream, can consist of thousands of Stock-Keeping Units (SKUs). These SKUs are products that may vary, for example, in composition or packaging. The large number of SKUs adds an additional challenge to the tactical planning for FMCG industries.

Another challenge is the seasonality of ingredients and products. For example, the majority of the demand for ice cream occurs during the summer. To properly capture this seasonality, we must consider at least a one year horizon divided into weekly time periods.

An additional challenge is the generally large supply chain. A typical supply chain in the FMCG industry contains suppliers, factories, warehouses, distribution centers and retailers. There are many trade-offs between the decisions made in the various echelons, and therefore it is important to consider the complete supply chain when optimizing the tactical planning.

We will give a brief review of literature on tactical planning in the FMCG industry. We refer to Akkerman et al. (2010) for a detailed literature review on the optimization of operational, tactical and strategic planning in the food industry.

Duran (1987) considers the production and distribution network for a brewery by introducing separate capacities for the processing and packaging processes. The problem consisted of 17 breweries, 17 bottling factories, 40 agencies, 13 brands and 12 monthly periods. A combination of time decomposition and brand decomposition was used to obtain a solution. The proposed method reduced the total costs by 3.7% compared to the program that was being used at the production department.

Brown et al. (2001) discuss the operational and tactical planning LP models used at the FMCG company Kellogg. The supply chain they consider contains plants, co-packers and distribution centers. The tactical planning model contained over 600 SKUs, 27 locations and a 1-2 year horizon divided into 4-week periods. However, they did not consider set-up times, set-up costs and raw materials.

Li et al. (2009) optimize the capacity allocation decisions for a supply chain consisting of suppliers, factories and warehouses. They use two heuristic algorithms to be able to solve larger cases. Using the algorithms, they were able to optimize cases of up to 100 products and 4 time periods.

Bilgen and Gunther (2010) propose a flexible block planning approach for the short-term planning problem of a company producing fruit juices and soft drinks. A block planning approach is based on cyclically scheduling blocks that consist each of a pre-defined order of variable size production orders. They considered a planning problem containing 19 products, 4 weeks and a supply chain consisting of 3 factories and 3 warehouses with unlimited capacity. They showed that 5-15% cost savings can be obtained when using their flexible block planning approach instead of the more commonly used rigid block planning approach.

Kopanos et al. (2012) consider the optimization of production and logistics operations for a Greek dairy company. They use a discrete time representation to model the inventory and transportation decisions, and they use a continuous time representation to model the production and sequencing decisions. The sequencing and timing decisions are made for aggregated product families, whereas all other decisions are based on individual products. In the larger of the two problems they consider 93 products grouped into 23 families, 8 time periods, and a supply chain consisting of 2 factories and 5 distribution centers.

We can conclude from the above review that none of these papers have considered the optimization of a tactical planning problem consisting of thousands of SKUs for a 5-echelon supply chain over 52 weekly periods. The objective in this work is to develop an approach capable of optimizing such a case, which would be realistic for the FMCG industry. One of the main challenges is the size of this problem. We propose an algorithm based on SKU decomposition to be able to solve these extremely large problems.

The remainder of this paper is organized as follows. The problem definition is given in Section 2. The proposed MILP model is described in Section 3. The results obtained with this MILP model are discussed in Section 4. Next, Section 5 introduces the SKU decomposition algorithm that is used to optimize larger cases. The results obtained by the algorithm are discussed and compared with the results of the full model in Section 6. Finally, conclusions are drawn in Section 7, and the nomenclature is given at the end of the paper.

2. Problem Definition

Given is a set of SKUs that have to be produced and distributed through a supply chain network including suppliers, factories, warehouses, distribution centers and retailers. The location and capacity of all facilities is fixed. A schematic overview of the supply chain is given in Figure 1 where the arrows represent the possible flow of ingredients or SKUs from one facility to another. The procurement, production and distribution decisions have to be taken over a one year horizon divided into weekly time periods due to the seasonality of both SKUs and ingredients.

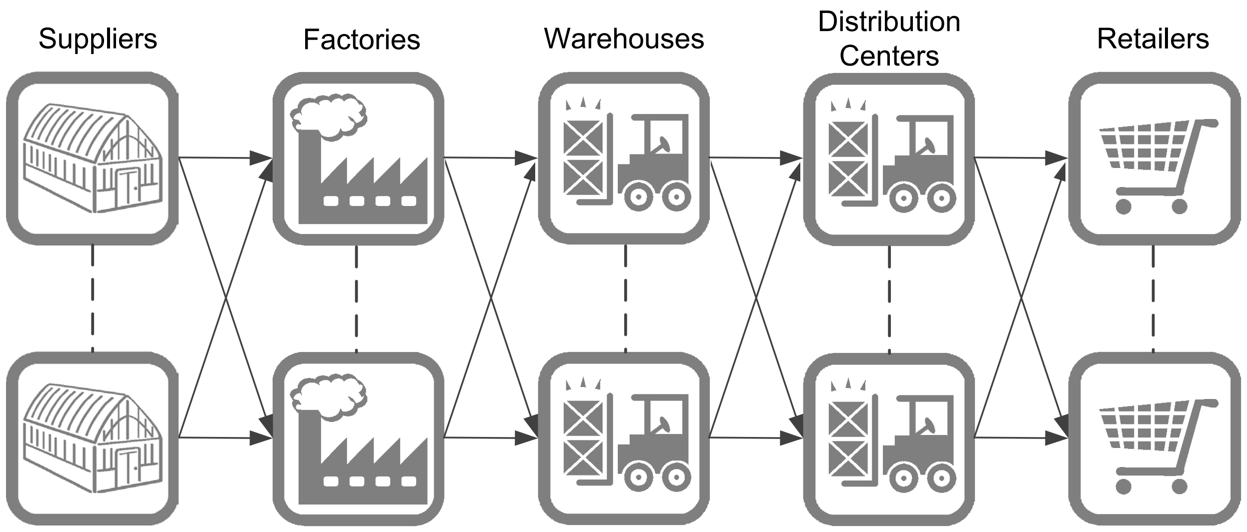

Figure 1. *Overview of the supply chain*

The unit transportation cost between any two consecutive facilities in the supply chain is known. The transportation times are typically considerably shorter than the period length of one week. Therefore, the lead times are assumed to be zero. Ingredients can be stored at the factories, and SKUs can be stored at the warehouses and distribution centers. For these facilities the initial and maximum inventories are given. In addition, the storage costs for each SKU or ingredient are known for each location. The minimum safety stock and the penalty for violating this minimum level are also given for all SKUs in all warehouses and distribution centers.

The maximum available supply and the procurement costs are known for all ingredients for every supplier for every week. The given recipes link the production of SKUs to the ingredient consumption. Each factory contains two production stages: a mixing stage and a packing stage. An SKU must be mixed and packed in the same factory in the same week. Factories contain various types of mixing and packing lines. Each type is dedicated to a subset of SKUs. The available production time on both stages is given as the aggregated amount per type of mixing or packing line. The mixing and packing rates of all SKUs are also known. Average SKU and SKU-family set-up times and costs are given for the packing stage.

The demand of each retailer is given per SKU per week, and a penalty cost for missed sales is given as well. Demand can only be met in the week in which it occurs, and the amount sent to a retailer may not exceed the demand. The inventory at the retailers is not considered.

Given this information, the key decisions are the amount of each ingredient to buy from the suppliers, the amount of each SKU to produce in each of the factories, the inventory levels in the warehouses and distribution centers, the amount of each SKU to transport between the facilities and the amount of each SKU to be sent to each of the retailers. All decisions have to be taken for each week. The objective is to minimize the total costs. The total costs consist of the procurement costs, storage costs, transportation costs, set-up costs, safety stock violation costs and missed sales costs.

3. MILP model formulation

In this section we propose an MILP model for the tactical planning problem in the FMCG industry. We first discuss the concept behind the production capacity approximation used in the model. Afterwards the model constraints are discussed.

3.1. Production capacity approximation

The weekly production plans generated by the tactical planning model determine how much of each SKU should be produced by each factory in each week. Therefore, it is crucial that the capacity limitations in the tactical planning model closely represent the true capacity limitations. The capacity could be modeled accurately by incorporating the short-term scheduling decisions directly into the tactical planning model. However, optimizing these short-term scheduling decisions is already challenging for a single factory for a single week. (Kopanos et al., 2011; van Elzakker et al., 2012) Therefore, incorporating these decisions directly in the tactical planning model would render it intractable. Nevertheless, a close approximation is essential since underestimating the capacity would reduce the efficiency of the production facilities, while overestimating it would lead to infeasible weekly production targets.

We have to consider three important aspects of the production process in the capacity estimation. First of all, the production process is a two stage make-and-pack production process. The first stage contains the mixing lines and the second stage the packing lines. In general, the bottleneck stage is not known in advance because it depends on the selection of SKUs. Therefore, it is important that the capacity of the mixing stage and packing stage are both considered.

Second, each factory may contain various types of mixing lines. Each type can only produce a subset of the SKUs. The group of SKUs that can be produced on a certain mixing line is denoted a mixing family. Therefore, the mixing capacity must be tracked per mixing family. Similarly, the packing capacity should be considered per type of packing line. The SKUs that can be produced on a certain type of packing line are a packing family. For each type of mixing or packing line, we impose aggregated capacity constraints to ensure that the production plan is feasible.

Third, there are sequence-dependent changeovers on both the mixing and the packing lines. Including sequence-dependent changeovers would require including line allocation and sequencing decisions in the tactical planning model. Because this would lead to an intractable model, we instead approximate the sequence-dependent changeovers.

From previous work on the short-term production scheduling in the FMCG industry (van Elzakker et al., 2012) we know that, in general, single continuous packing campaigns are enforced on the packing lines. In other words, each SKU that is assigned to a factory in a week will be produced in a single continuous packing campaign. Therefore, each assigned SKU will only require a single changeover.

To approximate this sequence-dependent changeover we use the concept of SKU families. An SKU family is a group of SKUs that have similar processing characteristics. Changeovers between SKUs of the same family are relatively short, whereas changeovers between SKUs of different families are considerably longer. Changeovers between SKUs of the same family can be represented by a relatively small average set-up time. We then represent the longer changeover between SKUs of different families by adding an average set-up time for each SKU family.

In summary, on the packing line we include a short SKU set-up time for each assigned SKU, and for each SKU family for which at least one SKU is assigned an additional SKU-family set-up time is included. This approximation is shown in Figure 2. The accuracy of this approximation relies on the assumption that SKUs of the same family are packed consecutively. This is a reasonable assumption because it minimizes the changeover time.

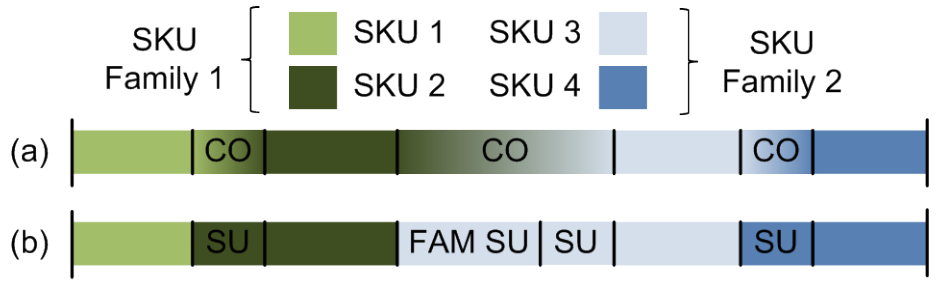

Figure 2. *The sequence-dependent changeover times (a) are approximated by SKU and SKU family set-up times (b).*

This SKU family approach is similar to the block planning approach by Gunther et al. (2006). A block is a predefined sequence of products which all have the same recipe. They typically account for a large set-up time for each block and a small set-up time for products within a block. They then allocate a single block to each period, and they only allow products that are part of this block to be produced in this period. The SKU families are also similar to the product families used for example by Shah et al. (1993). They introduced a required cleaning time when changing from products belonging to a "dark" family to products belonging to a "light" family.

However, this representation is not suitable for the mixing lines since the number of mixing line changeovers is much larger than the number of allocated SKUs. This is mainly because the throughput of mixing lines is higher than that of packing lines, as each factory contains more packing lines than mixing lines and because of the limited intermediate inventory. As a result, the mixing lines must switch frequently between SKUs to allow for the single continuous campaigns on the packing lines. This is explained in more detail in van Elzakker et al. (2012). Figure 3 shows a one week production schedule of a small FMCG factory generated by van Elzakker et al. (2012). It clearly demonstrates that the number of changeovers on the mixing lines is far greater than the number of allocated products.

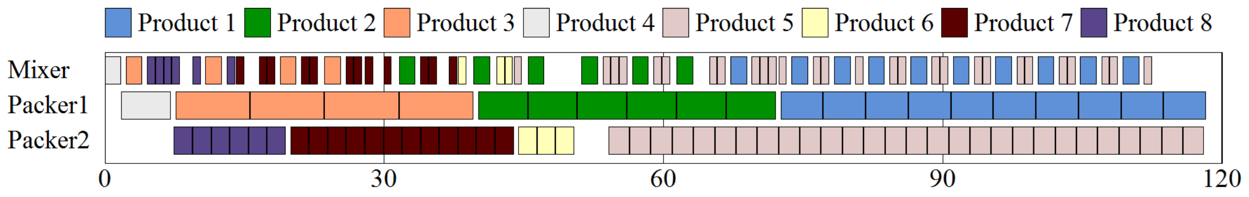

Figure 3. *Example of a one week production schedule for a small FMCG factory. (van Elzakker et al., 2012)*

The number of changeovers mainly depends on factory characteristics, such as the number of mixing and packing lines, the processing rates and the available intermediate storage. For example, a larger intermediate storage would allow for longer mixing runs and thus fewer changeovers. We propose to estimate the average total changeover time on the mixing lines based on historical factory data. While it should be noted that this is an approximation, it is far more accurate than linking it to the number of SKUs that are allocated.

3.2. Procurement

The total amount of ingredient *h* procured from supplier *s* in week *t* to all factories is limited by the available supply.

$$\sum_{f} TransIng_{h,f,s,t} \leq MaxSupply_{h,s,t} \quad \forall h,s,t \tag{1}$$

The total amount of ingredients in storage at factory *f* in week *t* cannot exceed the storage capacity.

$$\sum_{h} INVIng_{h,f,t} \leq INVIngCap_{f} \quad \forall f,t \tag{2}$$

The inventory of ingredient *h* in factory *f* in week *t* is equal to the inventory in the previous week, plus the amount procured from all suppliers minus the amount consumed in the production of all SKUs.

$$INVIng_{h,f,t} = INVIng_{h,f,t-1} + \sum_{s} TransIng_{h,f,s,t} - \sum_{i}\left(Recipe_{h,i} \cdot Prod_{i,f,t}\right) \quad \forall h,f,t \tag{3}$$

3.3. Production

The production time allocated to mixing all SKUs that are part of the same mixing family in factory *f* in week *t* cannot be larger than the available mixing time of this mixing family. The available mixing time has already been corrected for the estimated total weekly set-up time.

$$\sum_{(i \in IM_{mfam})} \frac{Prod_{i,f,t}}{MixRate_{i,f}} \leq MixTime_{mfam,f} \quad \forall mfam, f,t \tag{4}$$

The packing time allocated to the SKUs of the current packing family, plus the set up time of each SKU of this packing family that is produced, plus the set up time of the SKU families that are part of the packing family and of which at least one SKU is produced, must be less than the available packing time.

$$\begin{aligned}&\sum_{i \in IP_{pfam}} \left(\frac{Prod_{i,f,t}}{PackRate_{i,f}} + SUT_i \cdot WSU_{i,f,t} \right) \\ &+ \sum_{fam \in FAM_{pfam}} \left[FamSUT_{fam} \cdot YFamSU_{fam,f,t} \right] \\ &\qquad \leq PackTime_{pfam,f} \quad \forall pfam, f, t\end{aligned} \tag{5}$$

If SKU *i* is produced in factory *f* in week *t*, then there must be a set-up for this SKU in this factory in this week. The total available packing time for the packing family to which SKU *i* belongs is used as the upper bound for the packing time of SKU *i*.

$$\frac{Prod_{i,f,t}}{PackRate_{i,f}} \leq PackTime_{pfam,f} \cdot WSU_{i,f,t} \quad \forall i \in IP_{pfam}, pfam, f, t \tag{6}$$

If there is a set up for SKU *i*, there must also be a set up for the family to which this SKU belongs.

$$YFamSU_{fam,f,t} \geq WSU_{i,f,t} \quad \forall i \in IF_{fam}, fam, f, t \tag{7}$$

The total production of SKU *i* in factory *f* in week *t* must be transported to the warehouses because there is no product storage at the factories.

$$\sum_{w} TransFW_{i,f,w,t} = Prod_{i,f,t} \quad \forall i, f, t \tag{8}$$

3.4. Storage and Transport

The warehouse and distribution constraints are discussed together because they are very similar. The total inventory of all SKUs in a location may not exceed the storage capacity.

$$\sum_{i} INVWH_{i,w,t} \leq WHCap_{w} \quad \forall w, t \tag{9}$$

$$\sum_{i} INVDC_{i,dc,t} \leq DCCap_{dc} \quad \forall dc, t \tag{10}$$

The inventory of SKU *i* in warehouse *w* in week *t* is equal to the inventory in the previous week, plus the amount received from all factories, minus the amount sent to all distribution centers. For the first week, the inventory in the previous week is the initial inventory.

$$INVWH_{i,w,t} = INVWH_{i,w,t-1} + \sum_{f} TransFW_{i,f,w,t} - \sum_{dc} TransWDC_{i,w,dc,t} \quad \forall i, w, t \tag{11}$$

Similarly, the inventory of SKU *i* in distribution center *dc* in week *t* is equal to the inventory in the previous week, plus the amount received from all warehouses, minus the amount sent to all retailers. For the first week, the inventory in the previous week is the initial inventory.

$$INVDC_{i,dc,t} = INVDC_{i,dc,t-1} + \sum_{w} TransWDC_{i,w,dc,t} - \sum_{r} TransDCR_{i,dc,r,t} \quad \forall i, dc, t \tag{12}$$

If the inventory is less than the safety stock, the safety stock violation is the difference between the safety stock and the inventory. Otherwise the safety stock violation is zero. The safety stock violation is defined as a nonnegative continuous variable. Because the safety stock violation costs are added to the objective function, these costs will always take on the lowest possible value. These safety stock constraints are similar to those of McDonald and Karimi (1997).

$$SSVioWH_{i,w,t} \geq SSWH_{i,w,t} - INVWH_{i,w,t} \quad \forall i, w, t \tag{13}$$

$$SSVioDC_{i,dc,t} \geq SSDC_{i,dc,t} - INVDC_{i,dc,t} \quad \forall i, dc, t \tag{14}$$

The total amount of the SKU *i* transported to retailer *r* in week *t* cannot exceed the demand.

$$\sum_{dc} TransDCR_{i,dc,r,t} \le D_{i,r,t} \quad \forall i,r,t \tag{15}$$

3.5. Costs

The objective is to minimize the total costs. These costs consist of the purchasing plus transportation costs of the ingredients, the inventory costs of the ingredients at the factories, the inventory costs of the SKUs at the warehouses and distribution centers, the SKU transportation costs between the factory and warehouses, between the warehouses and distribution centers and between the distribution centers and retailers, the safety stock violation penalty costs in the warehouses and distribution centers, the set-up costs of the SKUs, the SKU family set-up costs, and the missed sales penalty costs.

$$\begin{aligned}
TotalCosts = & \sum_{h,f,s,t} TransIng_{h,f,s,t} \cdot \left(CostIng_{h,s,t} + TCSF_{f,s} \right) \\
& + \sum_{h,f,t} INVIng_{h,f,t} \cdot SCIng_{h,f} + \sum_{i,w,t} INVWH_{i,w,t} \cdot SCWH_{i,w} + \sum_{i,dc,t} INVDC_{i,dc,t} \cdot SCDC_{i,dc} \\
& + \sum_{i,f,w,t} TransFW_{i,f,w,t} \cdot TCFW_{f,w} + \sum_{i,w,dc,t} TransWDC_{i,w,dc,t} \cdot TCWH_{w,dc} \\
& + \sum_{i,dc,r,t} TransDCR_{i,dc,r,t} \cdot TCDCR_{dc,r} \\
& + \sum_{i,w,t} SSpenCost \cdot SSVioWH_{i,w,t} + \sum_{i,dc,t} SSpenCost \cdot SSVioDC_{i,dc,t} \\
& + \sum_{i,f,t} SUCost_{i} \cdot WSU_{i,f,t} + \sum_{fam,f,t} FAMSUCost_{fam} \cdot YFAMSU_{fam,f,t} \\
& + \sum_{i,r,t} MSpen_{i,r,t} \cdot \left(D_{i,r,t} - \sum_{dc} TransDCR_{i,dc,r,t} \right)
\end{aligned} \tag{16}$$

4. Full Model Results

This MILP model has been applied to several case studies. The supply chain in these cases studies consists of 10 suppliers, 4 factories, 5 warehouses, 10 distribution centers, and 20 retailers. The case studies contain 10 ingredients and between 10 and 1000 SKUs. Each SKU belongs to one of 2 different mixing families, 4 packing families and 12 SKU families. For all case studies the one year time horizon is divided into 52 weekly periods.

Due to the extremely large amount of data required, hypothetical data is used in this paper. For example, the transportation cost between factories and warehouses is generated from the uniform distribution $U(0.01,0.5)$. Most data is generated from uniform distributions. However, there are a few notable exceptions.

Since the retailers do not sell all SKUs, each SKU is only given a 33% chance to be allocated to a retailer. The average demand for an SKU that is allocated to a retailer is then generated from a uniform distribution. The weekly demand is then generated between 50% and 150% of the average demand. In addition, to account for seasonality, the demand during a 4 week period is increased. The total demand during these 4 weeks accounts for approximately 80% of the total demand.

Similarly, since not every supplier will sell all ingredients, each ingredient is only given a 25% chance to be sold at a supplier. The weekly supply is then generated in the same way as the weekly demand of SKUs. Each ingredient has a 33% chance that it is required in the production of an SKU, which gives an average of 3.3 ingredients per product. The amount required in the recipe is then generated from a uniform distribution. The available production time for both mixing and packing families is generated from a discrete uniform distribution, since each additional line would add 120 hours of production per week. The upper and lower bounds of distributions of the ingredient supply and production capacity are determined based on the demand of the products. All other data is generated from uniform distributions.

4.1. 10 SKU case study results

We first optimize the case study containing 10 SKUs. This case study will be used to discuss characteristics of the model and problem. All optimizations in this paper are performed using CPLEX 12.4 in AIMMS 3.12 on a computer with an Intel(R) Core(TM)2 Duo CPU P8700 2.53 GHz and with 4 GB of memory. All optimizations are performed with a one percent MIP optimality tolerance.

The model for the 10-SKU case study contains 41,809 constraints and 185,589 variables of which 2,080 are binary. The MILP model is already large for this small example case because of the size of the supply chain and the 52 weekly time periods. The required CPU time was 4427s. We will discuss the obtained solution by highlighting some of the key characteristics of the results.

The total inventory profile of SKU 10 is given in Figure 4. In the first part of the horizon there are a few small peaks followed by a slowly decreasing inventory. This indicates that producing a large batch and paying higher inventory costs is less expensive than producing small batches every week and incurring weekly set up costs. After 26 weeks the inventory starts to build up. This is necessary to cover the peak demand during weeks 45 to 48. This increase in production around week 26 is also clearly visible in Figure 5. It can also be seen from Figure 5 that typically only one product per packing family is produced in each week. This reduces the required set-up time and thus maximizes the available production time.

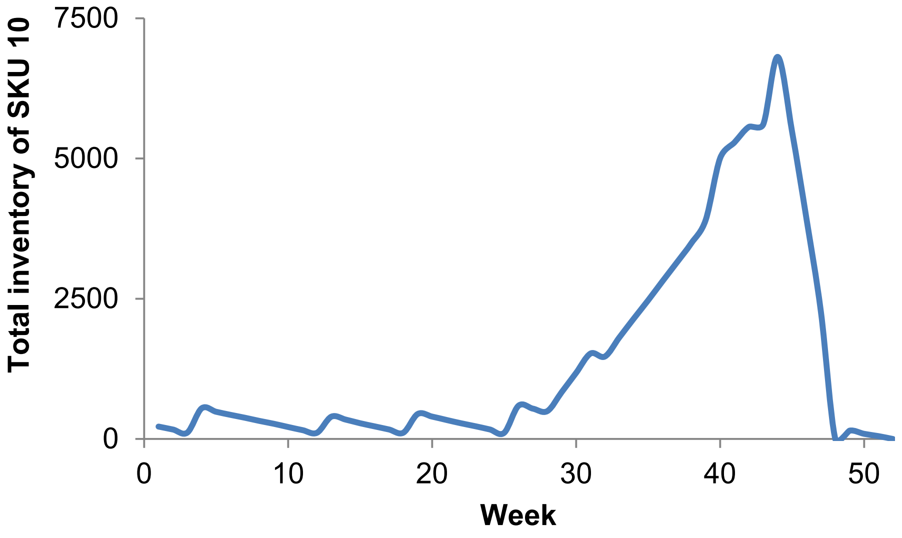

Figure 4. *Profile of the total inventory of SKU 10 in all storage facilities over the time horizon*

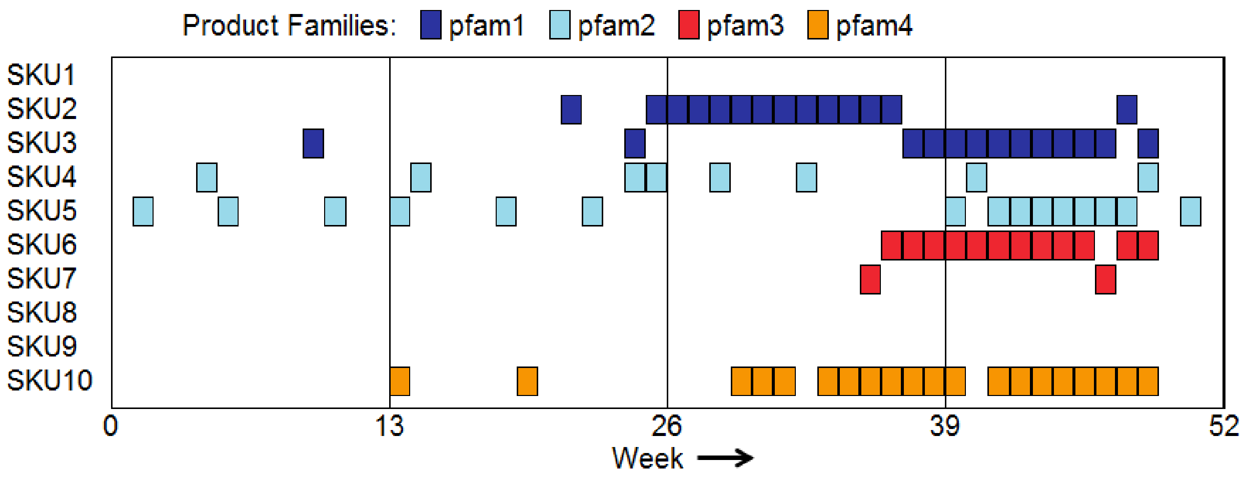

Figure 5. *Gantt Chart indicating which product is produced in each week in factory 1 in the MILP solution*

The modeling of the SKU and SKU family set-ups is an important part of the MILP model. While using these set-ups to approximate the changeovers is more efficient than directly including sequence-dependent changeovers, the binary set-up variables still make the model significantly harder to solve. To demonstrate the need of including the binary set-up variables, we have also optimized the same 10-SKU case study with the binary set up variables relaxed as 0-1 continuous variables.

The resulting LP problem was optimized in 124s. However, as can be seen in Figure 6, the number of products that are produced in each week increases drastically. In fact, the average number of products allocated to a factory in a week increases from 1.78 to 4.85. This results in a cost increase of 8.3% when accounting for set-up costs. Moreover, the solution obtained by the LP would be infeasible since it does not consider the set-up times. While this would only lead to a relatively small capacity violation for the 10-SKU case, the impact would be much greater in a more realistic case containing 1000 SKUs. In such a case, the LP might allocate hundreds of SKUs to the same factory in the same week. Therefore, the binary set-up variables are clearly necessary to obtain realistic solutions.

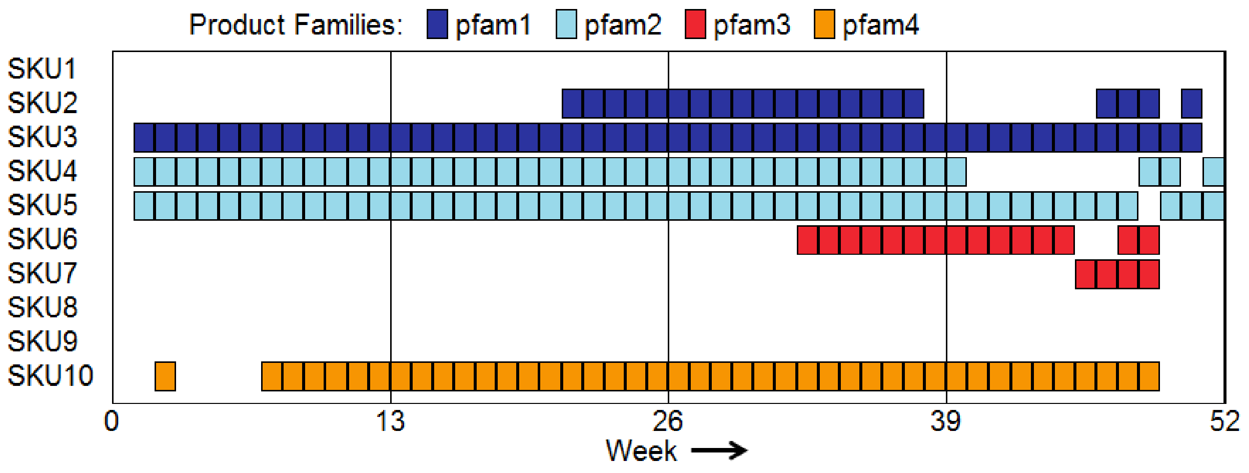

Figure 6. *Gantt Chart indicating which product is produced in each week in factory 1 in the LP solution*

5. SKU Decomposition Algorithm

As mentioned in the previous section, the MILP model for the 10 SKU case is already relatively large. For larger cases, the MILP becomes prohibitively large. For the 50 SKU case, the model contains 170,769 constraints and 826,229 variables of which 10,400 are binary. No feasible solution could be obtained for this case within 12 hours. A realistic case would contain at least a thousand SKUs. For such a case, the model would contain more than 2 million constraints and more than 10 million variables of which 208,000 would be binary. Because even the far smaller 50 SKU case could not be optimized, the model is intractable for realistic cases.

Therefore, we must reduce the model size to be able to optimize realistic cases. One approach would be to aggregate SKUs into families. For example, Omar and Teo(2007) reduce the size of their tactical planning model for chemical multiproduct batch plants by aggregating the products into product families. However, in the FMCG industry, the SKUs within a family may require different ingredients. Therefore, if we aggregate SKUs into SKU families, we would not be able to accurately determine the demand of ingredients based on the production. As a result, we would not be able to optimize the entire supply chain simultaneously because we cannot include the procurement decisions.

5.1. Overview of decomposition methods

An alternative approach to reduce the size of the model is to decompose the model into several smaller submodels. Sousa et al. (2011) and Terrazas-Moreno et al. (2011) give an overview of decomposition methods. The most common decompositions are spatial or temporal decompositions. In a spatial decomposition, the subproblems can describe either different echelons of the supply chain or different physical locations. However, for our problem a spatial decomposition would not give a sufficient reduction in model size because each submodel would still contain thousands of SKUs over 52 time periods.

In a temporal decomposition, the problem is decomposed into submodels covering a single time period each. Temporal decomposition does not seem promising for our problem because of the high seasonality of products and ingredients. In addition, the resulting subproblems would still be very large since they would contain thousands of SKUs and a relatively large supply chain.

Castro et al. (2009) proposed an order decomposition algorithm for the scheduling of multiproduct plants. The main idea behind their algorithm is to start with a couple orders and allocate them to a unit. Then the next few orders are allocated while the allocation decisions for the first few orders are fixed. However, the timing decisions of the first orders are still variable. A few more orders are then added while the allocation decisions of all previous orders are fixed. This continues until all orders have been allocated. Finally, the schedule is improved in a rescheduling step. In this step, a few orders may be rescheduled while the allocation decisions for all other orders are still fixed. This step can be repeated several times.

5.2. SKU Decomposition Algorithm

The concept of decomposition based on SKUs is promising for our problem. However, the problem size will not be reduced significantly if we use the method of Castro et al. (2009) and fix only the allocation decisions. Therefore, we propose the following SKU decomposition algorithm.

In the algorithm, the tactical planning MILP model is decomposed into single SKU MILP submodels. In each submodel, the domain of the constraints and variables is limited to a single SKU. The updated constraints for the submodels are given in Appendix A.

A solution to the full problem could be obtained by optimizing these submodels incrementally. In other words, the decisions for the various SKUs are optimized sequentially, and the decisions of the previous SKUs are fixed. As a result, the available capacity will decrease after each SKU is optimized. Because this procedure does not include any interaction between the SKUs, the capacity would most likely be used inefficiently, and the initial solution would most likely be poor.

We have modified the capacity constraints to improve the capacity utilization. We have added a slack variable to each capacity constraint to allow the maximum capacity to be violated. The capacity constraints are constraints (2), (4), (5), (9) and (10). The slack variables are added to the objective function with a penalty costs. Therefore, a penalty cost is incurred when the capacity is violated. This approach is similar to the classical penalty function method introduced by Courant (1943) that replaces constraints with penalty terms in the objective function.

The SKU decomposition algorithm consists of two steps. In the first step an initial solution is obtained. This initial solution is most likely infeasible. In the second step this initial solution is used as a starting point, and in several iterations it is driven towards a feasible solution.

In the first step, all submodels are optimized incrementally with the penalty costs set to zero and with relaxed set-up variables. The zero penalty costs in essence represent an optimization for unlimited capacity. Because of this unlimited capacity, the solution will most likely be infeasible for the problem with limited capacity. Therefore, the binary variables are relaxed to obtain this initial solution faster.

In the second step, the initial solution obtained in step one is used as a starting point. All decisions except for those relating to SKU 1 are fixed, and then SKU 1 is re-optimized. In this second step the optimal decisions will change because integrality is enforced for the binary variables and because the penalty for capacity violation is set to a non-zero value. Then these updated decisions are fixed, and the decisions for SKU 2 are re-optimized. This entire procedure is repeated for each SKU. In each optimization in step 2, the decisions for all SKUs but the current SKU are frozen, and the decisions for the current SKU are re-optimized using the MILP submodel. Afterwards, these decisions are frozen, and the next SKU is updated. In each iteration in step 2, all SKUs are re-optimized once.

Because the penalty is initially set to a low value, violating the capacity will be relatively inexpensive. As a result, for most SKUs it will be less expensive to pay the capacity violation penalty costs than it would be to reallocate them to a different facility. The algorithm continuous to iterate until all slack variables are zero, and a feasible solution is thus obtained.

To ensure that the slack variables will eventually become zero, the penalty costs are increased slightly after each optimization. Therefore, it will continuously become more expensive to exceed the capacity. For some SKUs it will become less expensive to be reallocated to a different facility than it would be to pay the penalty costs. Eventually, the penalty costs will become sufficiently high, and enough products will be reallocated to obtain a feasible solution. The algorithm is terminated once a feasible solution is obtained.

For the classical penalty function method, the solution of the unconstrained problem converges to the solution of the constrained problem if the penalty is selected to be sufficiently large. (Luenberger, 1971) For our problem a feasible solution can be guaranteed within a finite number of iterations because of the missed sales costs. Eventually, the penalty costs per unit of capacity violation will be higher than the missed sales costs. At that point, a feasible solution will be obtained because any remaining capacity violations will become missed sales. An overview of the algorithm is given in Figure 7.

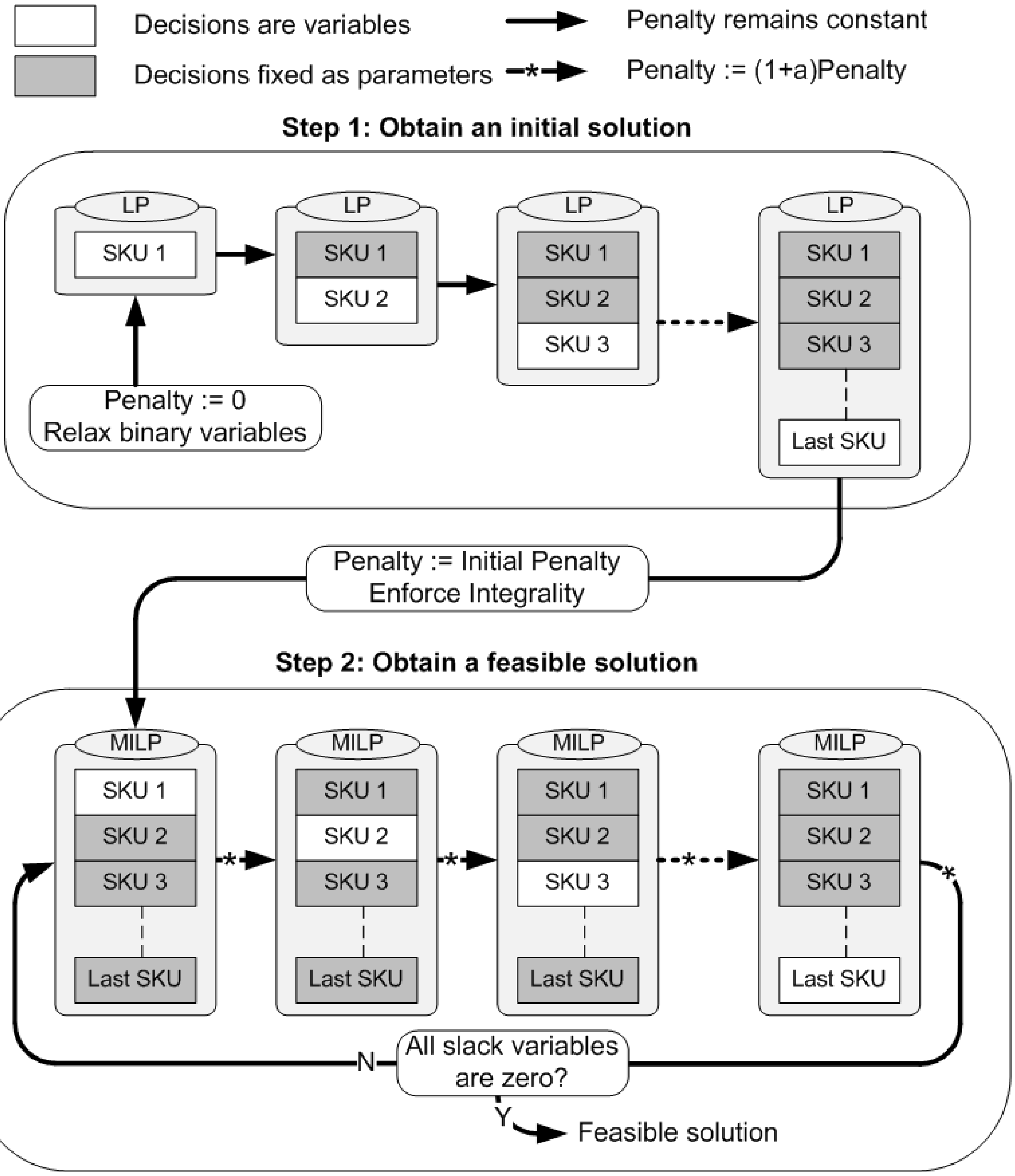

Figure 7: *SKU Decomposition algorithm*

5.3. Illustrative Example

The SKU decomposition algorithm will be demonstrated using a small illustrative problem. This illustrative example contains 4 time periods, 4 ingredients, 4 SKUs and a supply chain consisting of 2 suppliers, 2 factories, 2 warehouses and 2 retailers. Dimensionless data is used in this illustrative example. The ingredient availability at the suppliers is given in Table 1, and the procurement costs for all ingredients is 1/unit. The transportation costs are given in Table 2, and the demand is given in Table 3. The weekly storage costs are 0.5/unit. For all SKUs, the production rate on mixing and packing lines is 1 unit/hr, the set-up time is 0.5 hr, the set-up costs is 15, and the missed sales costs are 25/unit. All products belong to the same mixing, packing and SKU family, and therefore, family set-up times or costs are not considered.

Table 1: *Weekly available supply*

	Ingredient 1	Ingredient 2	Ingredient 3	Ingredient 4
Supplier 1	20	-	-	12
Supplier 2	-	5	12	-

Table 2: *Transportation costs per unit between suppliers(S), factories(F), warehouses(W) and retailers(R)*

	F1	F2
S1	0.2	0.6
S2	0.6	0.2

	W1	W2
F1	0.3	0.5
F2	0.5	0.3

	R1	R2
W1	0.3	0.5
W2	0.5	0.3

Table 3: *Demand in the retailers. The demand is the same in both retailers.*

	Period 1	Period 2	Period 3	Period 4
SKU 1	1	1	5	1
SKU 2	0.5	0.5	2.5	0.5
SKU 3	1.5	1.5	6.5	1.5
SKU 4	3	3	3	3

The production of one unit of SKU 1 requires one unit of ingredient 1. Similarly, the production of SKUs 2-4 require one unit of ingredients 2-4. Since the illustrative example is used to demonstrate the algorithm based on the production decisions, it does not include the storage capacity constraints or safety stock constraints. The initial penalty costs are set at 0.1, and a penalty increase of 100% per iteration is used. Because the illustrative example contains 4 SKUs, each iteration contains 4 parts, and the penalty increase is thus approximately 18.9% after each SKU.

In the first step of the algorithm, the set-up variables are relaxed, and the penalty costs are set to 0. This leads to the production plan shown on the left side of Figure 8. Because of the relaxed set-up costs and the zero capacity violation costs, the optimal production plan exactly meets the demand in each week. In addition, to minimize the transportation costs, SKU 1 and 4 are produced in factory 1, and SKU 2 and 3 are produced in factory 2. The solution obtained in the first step of the algorithm is infeasible since the production capacity of factory 2 is violated in the third week.

Figure 8 shows the production plans obtained in the first iteration of the second step of the algorithm. In this figure, iteration i.k refers to SKU k at iteration i. In iteration 1.1, the decisions for SKU 1 are re-optimized. In the new production plan, the complete 4-week demand of SKU 1 is produced in the first week. This minimizes the set-up costs, which are no longer relaxed in step 2 of the algorithm. Similarly, the production of SKU 2 is moved to the first week in iteration 1.2. After iterations 1.3 and 1.4, SKU 3 and 4 are produced in weeks 1 and 3. Producing all of SKU 3 and 4 in the first week would reduce the set-up costs even further. However, the increased ingredient and production capacity penalty costs combined with the increased inventory costs would outweigh the reduction in set-up costs. The solution obtained after the first iteration of the second step is still infeasible as the production capacity is violated in week 1 for factory 1 and in weeks 1 and 3 for factory 2.

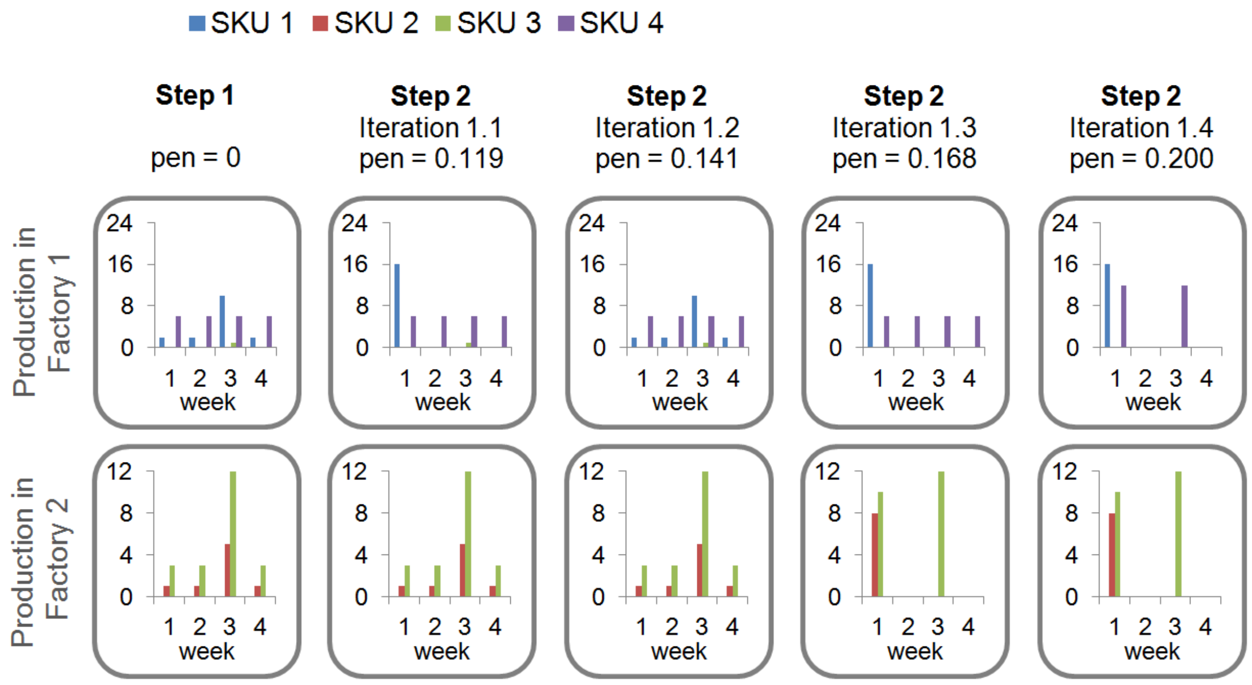

Figure 8: *Production plans for the illustrative example obtained in step 1 and the first iteration of step 2 of the algorithm*

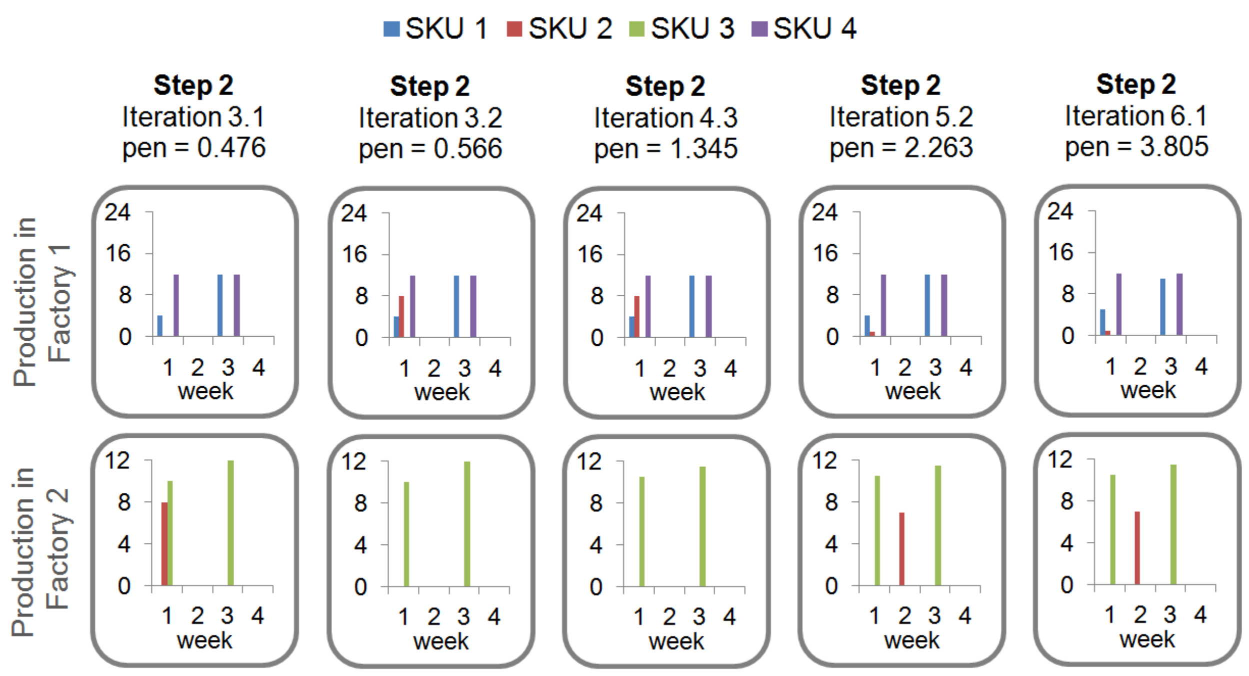

Figure 9: *Production plans for the illustrative example obtained in the various iterations of step 2 of the algorithm. Only those iterations where the production plan changed are included.*

In each iteration of step 2, each SKU is re-optimized once, and the penalty costs increases after each optimization. Only those optimizations that resulted in changes in the production plan will be discussed, and these production plans are given in Figure 9. The first change occurs in the iteration 3.1. In this optimization, the penalty costs are sufficiently high to force part of the production of SKU 1 to be reallocated to the third week. The additional set-up costs are less than the penalty and inventory costs would have been otherwise. The amount produced in the first week is exactly enough to meet the demand in the first two weeks. It should be noted that this leads to a small capacity violation in week 3, where the total required time is now 24 hours of production and 1 hour of set-ups. However, for the current penalty costs, this small capacity violation is less expensive than the alternatives.

In iteration 3.2, the penalty costs are sufficiently high to force SKU 2 to be reallocated. Interestingly, it is reallocated to the first factory. While this increases the transportation costs, it prevents an additional set-up. It should again be noted that this leads to a small 1.5 hour capacity violation at the first factory. In iteration 4.3, half a unit of SKU 3 is moved from week 3 to week 1. This removes the capacity violation in factory 2 in week 3. The increase in inventory costs is less than the penalty costs would have been.

The next change occurs in iteration 5.2. In this optimization the capacity violation in factory 1 week 1 is removed by moving most of the production of SKU 2 to week 2 factory 2. Not all of the production of SKU 2 is moved since that would lead to missed sales in the first week. While some production capacity is available in factory 2 week 1, the available capacity is not sufficient to meet all week 1 SKU 2 demand. Therefore, a small amount of SKU 2 is still produced in factory 1 in week 1. Finally, in iteration 6.1 the small capacity violation in factory 1 week 3 is resolved by moving 1 unit of SKU 1 to factory 1 week 1. At this point a feasible solution is obtained, and the algorithm terminates

The solution obtained with the algorithm for the illustrative example is identical to the solution that would be obtained with the full model. Therefore, for this problem the algorithm obtains the optimal solution. However, it should be noted that the algorithm offers no guarantee of global optimality.

6. SKU Decomposition Algorithm Results

6.1. Penalty Settings

The quality of the solution obtained by the algorithm depends on the selection of the penalty settings. These initial penalty settings also influence the required CPU time. A feasible solution can be obtained more quickly by using high initial penalty costs. Using high initial penalty costs, most infeasibilities will already be resolved in the first iteration since the costs of capacity violations will be high. However, this may cause the "wrong" SKUs to be reallocated since the penalty costs could be sufficiently high such that any SKU would be reallocated to prevent incurring capacity violation penalty costs. In this case the algorithm will reallocate the first few SKUs that are considered, whereas it might be less expensive to reallocate some of the other SKUs. Alternatively, a low initial penalty cost will yield better solutions but at a higher computational costs.

Similarly, better solutions can also be obtained by using a low penalty increase, although again at a computational costs. A feasible solution can be obtained faster by using a high penalty increase because fewer iterations are required to reach a sufficiently high penalty cost to remove the infeasibilities. Nevertheless, this higher penalty increase may lead to worse solutions.

As an example for seeing the effect of the penalty increase, consider the situation where SKU 1 and 2 would ideally both be allocated to the same factory. However, allocating both would exceed the production capacity, and the optimal decision would be to allocate SKU 1 to this factory and SKU 2 to another.

The basis of the algorithm is that initially both are allocated to the factory. Then by slowly increasing the penalty costs, they eventually become sufficiently high to reallocate one of the two. If a small penalty increase is used, then at a certain iteration the penalty costs is sufficiently high to force SKU 2 to be reallocated but not high enough to force SKU 1 to be reallocated. However, if a large penalty increase is used, it is possible that at one iteration neither of the two would be forced to be reallocated, while in the next iteration both would be forced to be reallocated. In that scenario SKU 1 would be reallocated because it is considered first.

Therefore, it is important to carefully select the penalty settings to obtain a good balance between the total required CPU time and the solution quality. The 10 SKU case has been optimized using various penalty settings. The initial penalty was varied between 0.05 and 5, and the penalty increase was varied between 5% and 500% per iteration. The results are summarized in Figure 10. Cost increase denotes the increase in costs for the solution obtained with the algorithm compared to the solution obtained with the full model. The CPU time is the total required CPU time until a feasible solution was obtained.

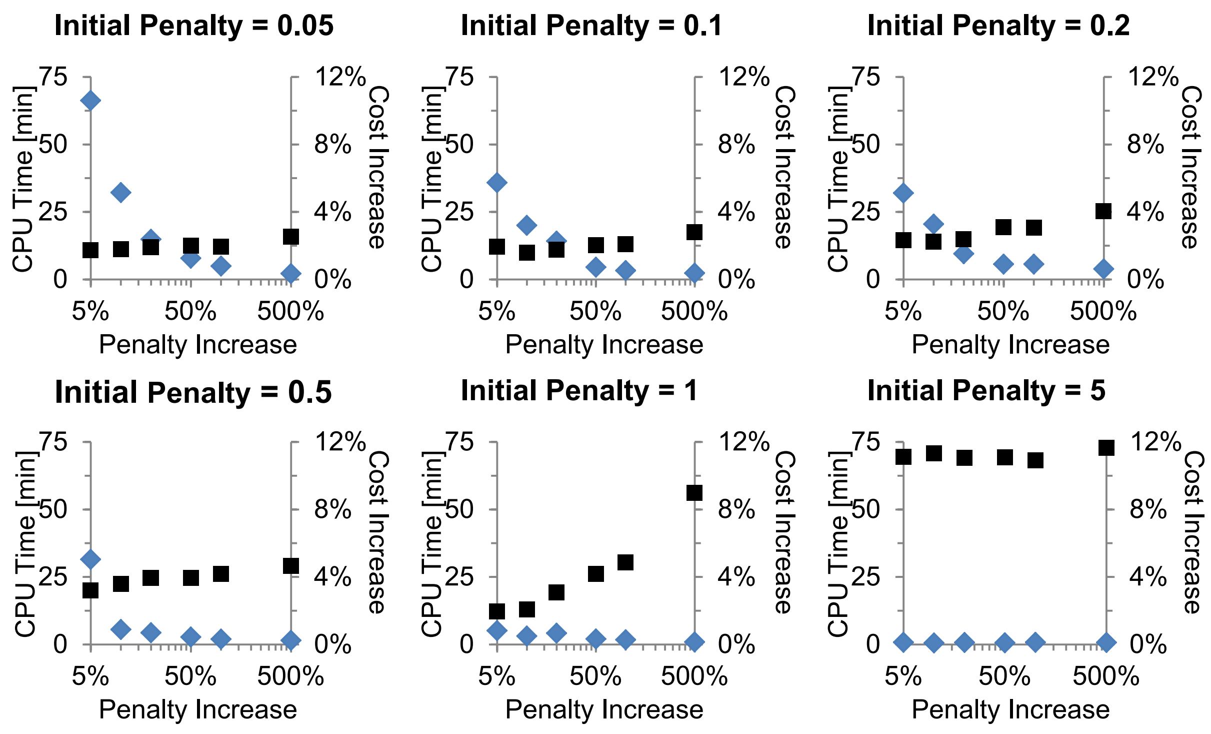

Figure 10: *Total CPU time (◆) and increase in costs (■) of the algorithm compared to the full model for various penalty settings*

It can be seen that all solutions obtained with the algorithm have a higher costs than the solution obtained with the full model. However, with the right penalty settings good solutions can be obtained with the algorithm. In fact, the best solution obtained with the algorithm had a cost increase of only 1.59%. On the other hand, for almost all penalty settings the CPU time required by the algorithm is far less than the 74 minutes required by the full model.

A very high initial penalty of 5 leads to poor solutions for any penalty increase. This is because the high initial penalty costs force all infeasibilities to be removed in the first iteration. Even though these solutions can be obtained within a minute, such a high initial penalty is a poor choice because the total costs increase by approximately 12%.

On the other hand, good solutions can still be obtained with a very high penalty increase, as long as the initial penalty is small. For example, the solution obtained with an initial penalty of 0.05 and a penalty increase of 500% has a cost increase of 2.5%. Nevertheless, with smaller penalty increases even better solutions can be obtained at only slightly increased computational costs.

Especially for small initial penalty values, a penalty increase of 50% offers a good trade-off between solution quality and required computational time. With a penalty increase of 50%, the required computational time is relatively constant for initial penalty values between 0.05 and 0.2. Therefore, the combination of an initial penalty value of 0.05 and a penalty increase of 50% is a suitable penalty setting. For the 10-SKU case, the algorithm obtained a solution with a cost increase of 1.99% in 472 seconds with these penalty settings.

It should be noted that the best initial penalty depends on the data. However, we found that as long as the data is in the same range, the best penalty settings remain reasonably constant. Because the data in all of our cases is generated between the same upper and lower bounds, we have used an initial penalty of 0.05 and a penalty increase of 50% for all cases. When using those penalty settings in the optimization of the 10-SKU case, the algorithm spent 10s in the first step. The second step required 8 iterations for a total of 462s. Because the penalty is set to zero in the first step, the time spent in the first step is independent of the penalty settings. Therefore, the differences in required CPU time between the various penalty settings originate from the second step of the algorithm.

Each submodel in the algorithm contains approximately 6917 constraints, 20593 continuous variables and 208 binary variables. The exact number of continuous variables and constraints varies slightly between the submodels. For example, the number of constraints describing the availability of ingredients varies because only those ingredients that are used in the production of the current SKU are included in the submodel.

6.2. Solution Quality

As discussed in the previous section, the solutions obtained with the algorithm have slightly higher total costs than the solution obtained with the full model. In this section, we will compare the characteristics of both solutions in more detail. The solution quality of larger cases will be discussed at the end of this section.

Figure 11 shows which SKUs are produced in factory 1 in each week in the solution obtained by the algorithm. When this figure is compared with Figure 5, it can be seen that the same SKUs are allocated to factory 1 in the full model. However, it can also be seen that the exact timing of the allocation decisions varies. Nevertheless, most differences in these timing decisions have a limited impact on the costs. For example, only the storage costs differ between producing SKU 2 in week 33 and SKU 3 in week 37 and producing them the other way around. Because the differences in unit storage costs between the various SKUs are very small, the impact on the total costs is very small as well.

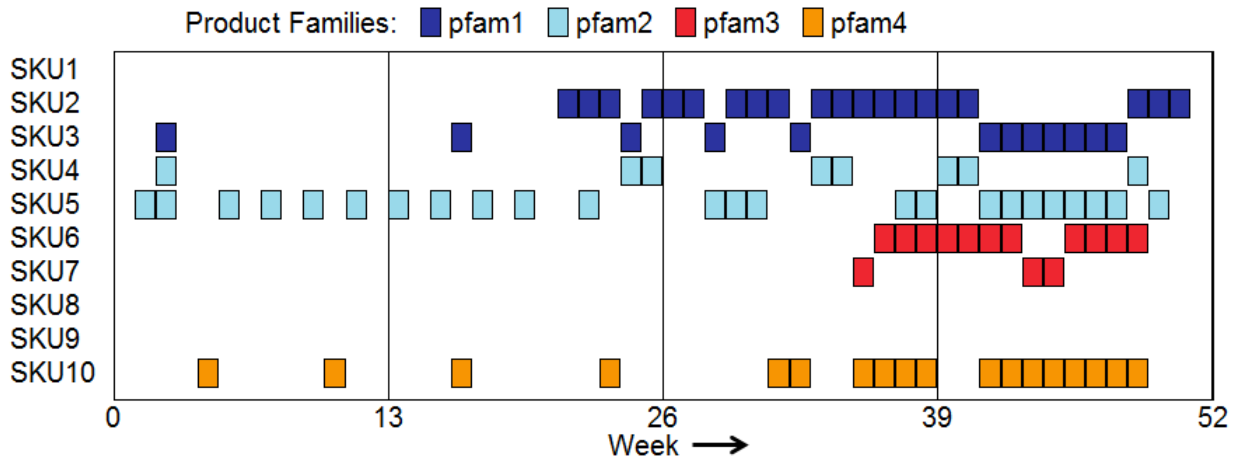

Figure 11: *Gantt Chart indicating which product is produced in each week in factory 1 in the algorithm solution*

Even though the timing of the individual SKUs varies, it can be seen in Figure 12 that the total inventory buildup is very similar. As a result, the total inventory costs are only slightly higher in the solution obtained with the algorithm; they account for only 8% of the increase in total costs. The majority of the total cost increase is caused by the set-up costs (47%) and the transportation costs (39%). The safety stock penalty contributes 3% and the procurement and missed sales 1% each.

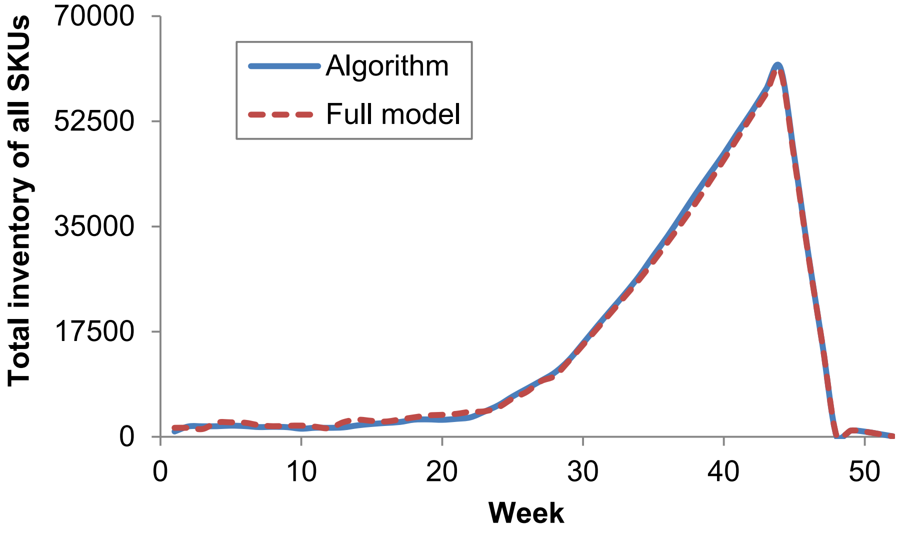

Figure 12: *Profile of the total inventory of all SKUs in all storage facilities over the time horizon*

It is more difficult to determine the quality of the solutions obtained by the algorithm for cases containing 25 or more SKUs since the full model is intractable for these cases. However, when the set-up variables are relaxed, the full model is able to optimize cases containing up to 100 SKUs. Therefore, despite some limitations in this comparison, LP relaxations of both the full model and the algorithm were used to optimize cases containing between 10 and 100 SKUs. An overview of the increase in costs when using the algorithm instead of the full model is given in Table 4. It should be noted that the submodel constraints and variables given in Table 4 are the typical number of constraints and variables in a submodel for an SKU produced from 3 ingredients. The exact number may vary depending on data such as the number of ingredients used in the production of the current SKU.

Table 4. *Computational results for cases containing between 10 and 100 SKUs. In all cases the set-up variables were relaxed in both the full model and the algorithm. The cost increase is for the solution obtained with the algorithm compared to the solution obtained with the full model.*

Number of SKUs	Full Model			Algorithm			
	Constraints	Variables	CPU Time	Submodel Constraints	Submodel Variables	Total CPU time	Cost Increase
10	41,809	185,589	28s	6,917	20,593	70s	0.95%
25	90,169	425,829	153s	6,917	20,593	366s	1.56%
40	138,529	666,069	384s	6,917	20,593	452s	1.32%
50	170,769	826,229	611s	6,917	20,593	648s	1.40%
60	203,009	986,389	1507s	6,917	20,593	946s	0.93%
75	251,369	1,226,629	9278s	6,917	20,593	1618s	3.09%
90	299,729	1,466,869	4959s	6,917	20,593	2030s	0.71%
100	331,969	1,627,029	3782s	6,917	20,593	1188s	0.78%

Clearly, the data set influences the optimality gap of the solution obtained by the algorithm. This optimality gap is the difference in costs between the solution obtained by the algorithm and the full model. However, for all cases, the optimality gap is within a few percent. Moreover, there does not seem to be a relation between the number of SKUs and the optimality gap. This is particularly important because a realistic case could contain thousands of SKUs. While it should be noted that the relaxation of binary variables may influence the optimality gap, it seems unlikely that including binary variables would introduce a strong correlation between the number of SKUs and the optimality gap of the algorithm. Therefore, we conclude that while the algorithm cannot guarantee global optimality, the solutions it obtains are within a few percent of optimality.

6.3. Required CPU time

The advantage of the algorithm is that it is computationally much more efficient than the full model. It was shown in Section 6.1 that the algorithm is already more efficient than the full model for a small case containing only 10 SKUs. However, the main advantage of the algorithm is that it can solve cases that are far larger than those that can be solved with the full model. While the full model is intractable for cases containing 25 or more SKUs, we have used the algorithm to solve cases of up to 1000 SKUs.

Not only is the algorithm capable of solving these large cases, but the required computational time scales well with the number of SKUs because the size of the submodels is independent of the number of SKUs. Whether the problem contains 10 or 100 SKUs, each submodel contains approximately 6917 constraints, 20,593 continuous variables and 208 binary variables. As a result, the only difference between cases with 10 and 100 SKUs is that the number of submodels increases by a factor 10. Consequently, the duration of each iteration is approximately 10 times longer, and thus the total required computational time also increases by a factor 10. However, for extremely large cases containing 500 or 1000 SKUs, the required computational time increases more than linearly. While the time spent optimizing each submodel remains constant, there is a significant time loss in between the optimizations of submodels. Nevertheless, cases containing 500 and 1000 SKUs could still be solved with the algorithm.

Table 5. *Required CPU time of the algorithm for cases containing between 10 and 1000 SKUs.*

Problem Size	Required CPU time
10 SKU	8 minutes
25 SKU	22 minutes
50 SKU	28 minutes
75 SKU	60 minutes
100 SKU	102 minutes
150 SKU	162 minutes
500 SKU	916 minutes
1000 SKU	3426 minutes

7. Conclusions

An MILP model was developed for the tactical planning in the FMCG industry. This MILP model was used to optimize a case containing 10 SKUs. However, a realistic case could contain thousands of SKUs, and for such a case the MILP model is prohibitively large.

Therefore, an SKU-decomposition algorithm was proposed. In this algorithm, submodels containing a single SKU are optimized sequentially while a penalty cost is introduced for violating the capacity. This penalty cost is increased after each optimization, and eventually it becomes sufficiently high to obtain a feasible solution. While there is no guarantee of global optimality, this feasible solution is typically within a few percent of the global optimum. Moreover, the algorithm is computationally efficient. Even for the small 10 SKU case the required CPU time could be reduced by a factor 9 by using the algorithm instead of the full model. Furthermore, the algorithm was able to optimize cases of a realistic size containing up to 1000 SKUs.

Acknowledgements
This research is supported by Unilever, which is gratefully acknowledged by the authors.

Nomenclature

Indices

dc	Distribution centers
f	Factories
fam	SKU families
h	Ingredients
i	SKUs
$mfam$	Mixing families
$pfam$	Packing Families
r	Retailers
SKU	Current SKU
t	Weeks
w	Warehouses

Subsets

FAM_{pfam}	SKU families belonging to packing family *pfam*
$FAMI_i$	SKU family to which SKU *i* belongs.
HI_i	Ingredients that are required for the production of SKU *i*.
IF_{fam}	SKUs belonging to SKU family *fam*
IM_{mfam}	SKUs belonging to mixing family *mfam*
IP_{pfam}	SKUs belonging to packing family *pfam*
MI_i	Mixing family to which SKU *i* belongs

Variables

$INVDC_{i,dc,t}$	Amount of SKU *i* stored in distribution center *dc* in week *t*
$INVIng_{h,f,t}$	Inventory of ingredient *h* at factory *f* in week *t*
$INVWH_{i,w,t}$	Amount of SKU *i* stored in warehouse *w* in week *t*
$Prod_{i,f,t}$	Amount of SKU *i* produced in factory *f* in week *t*
$SSVioDC_{i,dc,t}$	Amount of SKU *i* short of the safety stock in distribution center *dc* in week *t*
$SSVioWH_{i,w,t}$	Amount of SKU *i* short of the safety stock in warehouse *w* in week *t*

$TransDCR_{i,dc,r,t}$	Amount of SKU *i* transported from distribution center *dc* to retailer *r* in week *t*
$TransFW_{i,f,w,t}$	Amount of SKU *i* transported from factory *f* to warehouse *w* in week *t*
$TransIng_{h,f,s,t}$	Amount of ingredient *h* procured from supplier *s* to factory *f* in week *t*
$TransWDC_{i,w,dc,t}$	Amount of SKU *i* transported from warehouse *w* to distribution center *dc* in week *t*
$WSU_{i,f,t}$	Binary variable, indicates a set-up of SKU *i* in factory *f* in week *t*
$YFAMSU_{fam,f,t}$	0-1 continuous variable, indicates if there is a set-up of SKU family *fam* in factory *f* in week *t*
$\gamma_{h,s,t}$	Slack variable, represents the procurement amount that exceeds the available capacity of ingredient *h* at supply *s* in time period *t.*
$\gamma 2_{mfam,f,t}$	Slack variable, represents the production amount that exceeds the available capacity of mixing family *mfam* at factory *f* in time period *t.*
$\gamma 3_{pfam,f,t}$	Slack variable, represents the production amount that exceeds the available capacity of packing family *pfam* at factory *f* in time period *t.*
$\gamma 4_{w,t}$	Slack variable, represents the inventory amount that exceeds the available capacity of warehouse *w* in time period *t.*
$\gamma 5_{dc,t}$	Slack variable, represents the inventory amount that exceeds the available capacity of warehouse *dc* in time period *t.*

Parameters

$CostIng_{h,s,t}$	Unit cost of ingredient *h* at supplier *s* in week *t*
$D_{i,r,t}$	Demand of SKU *i* at retailer *r* in week *t*
$DCCap_{dc}$	Available storage capacity in distribution center *dc*
$FAMSUCost_{fam}$	Average set up cost for SKU family *fam*
$FAMSUT_{fam}$	Average set up time for SKU family *fam*
$INVDCP_{i,dc,t}$	Amount of SKU *i* stored in distribution center *dc* in week *t*. This parameter is used when the decisions for SKU *i* are frozen in the current optimization.
$INVIngCAP_{f}$	Available storage capacity for ingredients at factory *f*
$INVIngP_{h,f,t}$	Inventory of ingredient *h* at factory *f* in week *t*. This parameter is used when the decisions for ingredient *h* are frozen in the current optimization.
$INVWHP_{i,w,t}$	Amount of SKU *i* stored in warehouse *w* in week *t*. This parameter is used when the decisions for SKU *i* are frozen in the current optimization.
$MaxSupply_{h,s,t}$	Available supply of ingredient *h* at supplier *s* in week *t*
$MixTime_{mfam,f}$	Available mixing time at factory *f* for SKUs that are part of mixing family *mfam*
$MixRate_{i,f}$	Mixing rate of SKU *i* in factory *f*
$MSpen_{i,r,t}$	Penalty costs per unit of missed sales of SKU *i* at retailer *r* in week *t*
$PackRate_{i,f}$	Packing rate of SKU *i* in factory *f*
$PackTime_{pfam,f}$	Available packing time at factory *f* for SKUs that are part of packing family *pfam*
$ProdP_{i,f,t}$	Amount of SKU *i* produced in factory *f* in week *t*. This parameter is used when the decisions for SKU *i* are frozen in the current optimization.
$Recipe_{h,i}$	Amount of ingredient *h* consumed per unit produced of SKU *i*
$SCIng_{h,f}$	Storage costs of ingredient *h* at factory *f*
$SCDC_{i,dc}$	Storage costs of SKU *i* at distribution center *dc*

$SCWH_{i,w}$	Storage costs of SKU *i* at warehouse *w*
$SSDC_{i,dc,t}$	Minimum safety stock of SKU *i* in distribution center *dc* in week *t*
$SSWH_{i,w,t}$	Minimum safety stock of SKU *i* in warehouse *w* in week *t*
$SSpenCost$	Safety stock violation penalty cost
$SUCost_i$	Average set-up cost for SKU *i*
SUT_i	Average set-up time for SKU *i*
$TCDCR_{dc,r}$	Transportation cost between distribution center *dc* and retailer *r*
$TCFW_{f,w}$	Transportation cost between factory *f* and warehouse *w*
$TCSF_{f,s}$	Transportation cost between supplier *s* and factory *f*
$TCWDC_{w,dc}$	Transportation cost between warehouse *w* and distribution center *dc*
$WHCap_w$	Available storage capacity in warehouse *w*
$WSUP_{i,f,t}$	Binary parameter, indicates a set-up of SKU *i* in factory *f* in week *t.* This parameter is used when the decisions for SKU *i* are frozen in the current optimization.
$YFAMSUP_{fam,f,t}$	Binary parameter, indicates if there is a set-up of SKU family *fam* in factory *f* in week *t.* This parameter is used to indicate a required set up for one of the SKUs of SKU family *fam* that are frozen in the current optimization.

Appendix A

In this Appendix the constraints of the submodels of the SKU decomposition algorithm are given. Note that the domain of the constraints is limited to the current SKU (constraints (24), (27)-(33)), to the product/mixing/packing family to which the current SKU belongs (constraints (20)-(23)), or to the ingredients which are consumed in the production of the current SKU ((17) and (19)). In the same way, the domain of all variables is limited. Note also that $\gamma_{h,s,t}$, $\gamma 2_{mfam,f,t}$, $\gamma 3_{pfam,f,t}$, $\gamma 4_{w,t}$ and $\gamma 5_{dc,t}$ are the slack variables. In constraints (20) and (21) the slack variables are divided by the average mixing/packing rate. Because of this the unit of each slack variable is equal and we can apply the same penalty cost to all slack variables.

$$\sum_f TransIng_{h,f,s,t} \le MaxSupply_{h,s,t} + \gamma_{h,s,t} \qquad \forall h \in HI_{SKU}, s, t \tag{17}$$

$$\sum_{h \in HI_{SKU}} INVIng_{h,f,t} + \sum_{h \notin HI_{SKU}} INVIngP_{h,f,t} \le INVIngCap_f \quad \forall f,t \tag{18}$$

$$\begin{aligned} INVIng_{h,f,t} = {} & INVIng_{h,f,t-1} + \sum_s TransIng_{h,f,s,t} \\ & - \sum_{i=SKU} \left(Recipe_{h,i} \cdot Prod_{i,f,t} \right) - \sum_{i \ne SKU} \left(Recipe_{h,i} \cdot ProdP_{i,f,t} \right) \quad \forall h \in HI_{SKU}, f, t \end{aligned} \tag{19}$$

$$\begin{aligned} & \sum_{((i=SKU) \in IM_{mfam})} \frac{Prod_{i,f,t}}{MixRate_{i,f}} + \sum_{((i \ne SKU) \in IM_{mfam})} \frac{ProdP_{i,f,t}}{MixRate_{i,f}} \le MixTime_{mfam,f} \\ & \qquad + \frac{\gamma 2_{mfam,f,t}}{\text{average}_{i \in IM_{mfam}} \left[MixRate_{i,f} \right]} \quad \forall mfam \in MI_{SKU}, f,t \end{aligned} \tag{20}$$

$$\begin{aligned}
&\sum_{(i=SKU)\in IP_{pfam}} \left(\frac{Prod_{i,f,t}}{PackRate_{i,f}} + SUT_i \cdot WSU_{i,f,t} \right) \\
&+ \sum_{(i\neq SKU)\in IP_{pfam}} \left(\frac{ProdP_{i,f,t}}{PackRate_{i,f}} + SUT_i \cdot WSUP_{i,f,t} \right) \\
&+ \sum_{fam \in FAMI_{SKU} \cap FAM_{pfam}} \left[FamSUT_{fam} \cdot YFamSU_{fam,f,t} \right] \\
&+ \sum_{(fam \notin FAMI_{SKU}) \in FAM_{pfam}} \left[FamSUT_{fam} \cdot YFamSUP_{fam,f,t} \right] \\
&\qquad \le PackTime_{pfam,f} + \frac{\gamma 3_{pfam,f,t}}{\text{average}_{i\in IP_{pfam}} \left[PackRate_{i,f} \right]} \quad \forall pfam, f, t
\end{aligned} \tag{21}$$

$$\frac{Prod_{i,f,t}}{PackRate_{i,f}} \le PackTime_{pfam,f} \cdot WSU_{i,f,t} \quad \forall (i = SKU) \in IP_{pfam}, pfam, f, t \tag{22}$$

$$YFamSU_{fam,f,t} \ge WSU_{i,f,t} \text{ - } \max_{(i\neq SKU)\in IFAM_{fam}} \left[WSUP_{i,f,t} \right] \quad \forall (i = SKU) \in IF_{fam}, fam, f,t \tag{23}$$

$$\sum_{w} TransFW_{i,f,w,t} = Prod_{i,f,t} \quad \forall i = SKU, f, t \tag{24}$$

$$\sum_{i=SKU} INVWH_{i,w,t} + \sum_{i\neq SKU} INVWHP_{i,w,t} \le WHCap_w + \gamma 4_{w,t} \quad \forall w,t \tag{25}$$

$$\sum_{i} INVDC_{i,dc,t} \le DCCap_{dc} \quad \forall dc,t \tag{26}$$

$$INVWH_{i,w,t} = INVWH_{i,w,t-1} + \sum_{f} TransFW_{i,f,w,t} - \sum_{dc} TransWDC_{i,w,dc,t} \quad \forall i = SKU, w, t \tag{27}$$

$$INVDC_{i,dc,t} = INVDC_{i,dc,t-1} + \sum_{w} TransWDC_{i,w,dc,t} - \sum_{r} TransDCR_{i,dc,r,t} \quad \forall i = SKU, dc, t \tag{28}$$

$$SSVioWH_{i,w,t} \ge SSWH_{i,w,t} - INVWH_{i,w,t} \quad \forall i = SKU, w, t \tag{29}$$

$$SSVioDC_{i,dc,t} \ge SSDC_{i,dc,t} - INVDC_{i,dc,t} \quad \forall i = SKU, dc, t \tag{30}$$

$$\sum_{dc} TransDCR_{i,dc,r,t} \le D_{i,r,t} \quad \forall i = SKU, r, t \tag{31}$$

$$
\begin{aligned}
TotalCosts = & \sum_{h\in HI_{SKU},f,s,t} TransIng_{h,f,s,t}\cdot\left(CostIng_{h,s,t}+TCSF_{f,s}\right)\\
&+\sum_{h\in HI_{sku},f,t} INVIng_{h,f,t}\cdot SCIng_{h,f}+\sum_{i=SKU,w,t} INVWH_{i,w,t}\cdot SCWH_{i,w}+\sum_{i=SKU,dc,t} INVDC_{i,dc,t}\cdot SCDC_{i,dc}\\
&+\sum_{i=SKU,f,w,t} TransFW_{i,f,w,t}\cdot TCFW_{f,w}+\sum_{i=SKU,w,dc,t} TransWDC_{i,w,dc,t}\cdot TCWH_{w,dc}\\
&+\sum_{i=SKU,dc,r,t} TransDCR_{i,dc,r,t}\cdot TCDCR_{dc,r}\\
&+\sum_{i=SKU,w,t} SSpenCost\cdot SSVioWH_{i,w,t}+\sum_{i=SKU,dc,t} SSpenCost\cdot SSVioDC_{i,dc,t}\\
&+\sum_{i=SKU,f,t} SUCost_{i}\cdot WSU_{i,f,t}+\sum_{fam\in FAMI_{SKU},f,t} FAMSUCost_{fam}\cdot YFAMSU_{fam,f,t}\\
&+\sum_{i=SKU,r,t} MSpen_{i,r,t}\cdot\left(D_{i,r,t}-\sum_{dc} TransDCR_{i,dc,r,t}\right)\\
&+pen\cdot\left(\sum_{h,s,t}\gamma_{h,s,t}+\sum_{mfam,f,t}\gamma 2_{mfam,f,t}+\sum_{pfam,f,t}\gamma 3_{pfam,f,t}+\sum_{w,t}\gamma 4_{w,t}+\sum_{dc,t}\gamma 5_{dc,t}\right)
\end{aligned}
\tag{32}
$$

References

1. Akkerman,R., Farahani,P., & Grunow,M.(2010). Quality, safety and sustainability in food distribution: a review of quantitative operations management approaches and challenges. *Or Spectrum*, *32*, 863-904.

2. Bilgen,B. & Gunther,H.O.(2010). Integrated production and distribution planning in the fast moving consumer goods industry: a block planning application. *Or Spectrum*, *32*, 927-955.

3. Brown,G., Keegan,J., Vigus,B., & Wood,K.(2001). The Kellogg Company optimizes production, inventory, and distribution. *Interfaces*, *31*, 1-15.

4. Castro,P.M., Hariunkoski,I., & Grossmann,I.E.(2009). Optimal Short-Term Scheduling of Large-Scale Multistage Batch Plants. *Industrial & Engineering Chemistry Research*, *48*, 11002-11016.

5. Courant,R.(1943). Variational methods for the solution of problems of equilibrium and vibrations. *Bull.Amer.Math.Soc.*, *49*, 1-23.

6. Duran,F.(1987). A Large Mixed Integer Production and Distribution Program. *European Journal of Operational Research*, *28*, 207-217.

7. Erdirik-Dogan,M. & Grossmann,I.E.(2007). Planning models for parallel batch reactors with sequence-dependent changeovers. *Aiche Journal*, *53*, 2284-2300.

8. Grossmann,I.(2005). Enterprise-wide optimization: A new frontier in process systems engineering. *Aiche Journal*, *51*, 1846-1857.

9. Gunther,H.O., Grunow,M., & Neuhaus,U.(2006). Realizing block planning concepts in make-and-pack production using MILP modelling and SAP APO. *International Journal of Production Research*, *44*, 3711-3726.

10. Kopanos,G.M., Puigjaner,L., & Georgiadis,M.C.(2011). Production Scheduling in Multiproduct Multistage Semicontinuous Food Processes. *Industrial & Engineering Chemistry Research*, *50*, 6316-6324.

11. Kopanos,G.M., Puigjaner,L., & Georgiadis,M.C.(2012). Simultaneous production and logistics operations planning in semicontinuous food industries. *Omega-International Journal of Management Science*, *40*, 634-650.

12. Li,H.Y., Hendry,L., & Teunter,R.(2009). A strategic capacity allocation model for a complex supply chain: Formulation and solution approach comparison. *International Journal of Production Economics*, *121*, 505-518.

13. Luenberger,D.G.(1971). Convergence rate of a penalty-function scheme. *Journal of Optimization Theory and Applications*, *7*, 39-51.

14. Maravelias,C.T. & Sung,C.(2009). Integration of production planning and scheduling: Overview, challenges and opportunities. *Computers & Chemical Engineering*, *33*, 1919-1930.

15. McDonald,C.M. & Karimi,I.A.(1997). Planning and scheduling of parallel semicontinuous processes .1. Production planning. *Industrial & Engineering Chemistry Research*, *36*, 2691-2700.

16. Omar,M.K. & Teo,S.C.(2007). Hierarchical production planning and scheduling in a multi-product, batch process environment. *International Journal of Production Research*, *45*, 1029-1047.

17. Shah,N., Pantelides,C.C., & Sargent,R.W.H.(1993). A General Algorithm for Short-Term Scheduling of Batch-Operations .2. Computational Issues. *Computers & Chemical Engineering*, *17*, 229-244.

18. Sousa,R.T., Liu,S.S., Papageorgiou,L.G., & Shah,N.(2011). Global supply chain planning for pharmaceuticals. *Chemical Engineering Research & Design*, *89*, 2396-2409.

19. Sung,C. & Maravelias,C.T.(2007). An attainable region approach for production planning of multiproduct processes. *Aiche Journal*, *53*, 1298-1315.

20. Terrazas-Moreno,S. & Grossmann,I.E.(2011). A multiscale decomposition method for the optimal planning and scheduling of multi-site continuous multiproduct plants. *Chemical Engineering Science*, *66*, 4307-4318.

21. Terrazas-Moreno,S., Trotter,P.A., & Grossmann,I.E.(2011). Temporal and spatial Lagrangean decompositions in multi-site, multi-period production planning problems with sequence-dependent changeovers. *Computers & Chemical Engineering*, *35*, 2913-2928.

22. van Elzakker,M.A.H., Zondervan,E., Raikar,N.B., Grossmann,I.E., & Bongers,P.M.M.(2012). Scheduling in the FMCG Industry: An Industrial Case Study. *Industrial & Engineering Chemistry Research*, *51*, 7800-7815.

23. Varma,V.A., Reklaitis,G.V., Blau,G.E., & Pekny,J.F.(2007). Enterprise-wide modeling & optimization - An overview of emerging research challenges and opportunities. *Computers & Chemical Engineering*, *31*, 692-711.