TACTICAL PLANNING IN THE FAST MOVING CONSUMER GOODS INDUSTRY: AN SKU DECOMPOSITION ALGORITHM

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

[a] Dept. Chem. And Chem. Eng., 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

[d] Hoogewerff Chair for Product-driven Process Technology, Dept. Chem. And Chem. Eng., Eindhoven University of Technology

Abstract

Tactical planning models for the Fast Moving Consumer Goods (FMCG) industry can quickly become intractable due to the extremely large number of Stock Keeping Units (SKUs). We propose an SKU decomposition algorithm that is aimed at being able to solve cases containing thousands of SKUs. The full tactical planning model is decomposed into a set of single SKU models. These models are then solved sequentially. The capacity used by other SKUs is removed from the available capacity and, at a certain penalty cost, a violation of the capacity is initially allowed. By slowly increasing the penalty cost, the capacity violations are decreased until a feasible solution is obtained. Using the algorithm it was possible to obtain solutions within a few percent of optimality for example cases containing 10 and 25 SKUs. It was also possible to solve a larger 100 SKU case for which the full space model was intractable. The main advantage of the algorithm is that the required CPU time scales approximately linearly with the number of SKUs.

Keywords

Tactical Planning, Fast Moving Consumer Goods, Decomposition, Supply Chain Optimization

Introduction

Customers of Fast Moving Consumer Goods (FMCG) companies desire a large variety of products (Bilgen and Günther, 2010). Therefore, FMCG companies produce a wide range of products, differing both in composition and in packaging. Even within a single product category, such as ice cream, there can be as many as a few thousand different SKUs. Developing a tactical planning model for a FMCG company can be challenging due to this large product variety.

The seasonality of the products and ingredients is another important challenge for a FMCG company. For example, the ice cream demand is far higher during summer than during winter. As a result, weekly time periods over a one year horizon are required in a tactical planning model to properly represent this seasonality. In addition, a tactical planning model should cover the complete supply chain to be able to consider all trade-offs. A schematic overview of a typical FMCG supply chain is given in Figure 1.

In the FMCG industry, products are often produced in make-and-pack production (Bilgen and Günther, 2010). In such a production environment, products are processed

* To whom all correspondence should be addressed

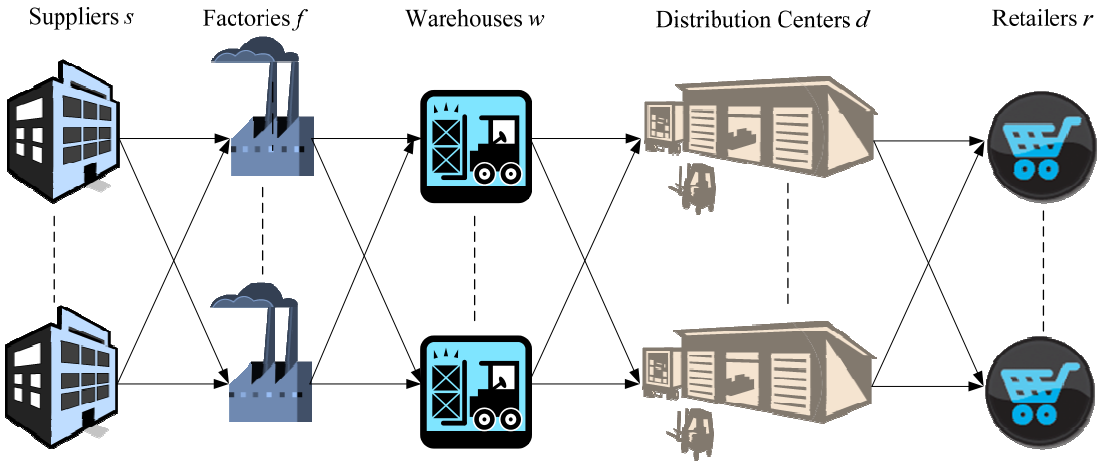

Figure 1. Schematic overview of the supply chain.

in a first stage and subsequently packed in a second stage in the same factory. Most factories contain different types of processing and packing lines that can each only process a subset of all SKUs.

Because of the large number of products, time periods and facilities in the supply chain, an MILP model for the tactical planning of a FMCG company would most likely be prohibitively large. This problem is not unique to FMCG companies. The tactical planning models for other industries tend to become large as well. One approach that is often used to decrease the model size is to aggregate products into product families. For example, Omar and Teo (2007) developed a tactical planning model using aggregated product families for a chemical multiproduct batch plant. Usually, these product families consist of products that can be produced on the same equipment. In addition, the changeover times between products of the same family are reasonably small. Therefore, the capacity required to produce the individual products can be estimated fairly well based on the aggregated amount.

However, for the tactical planning of FMCG companies this aggregation of products into families is not straightforward. While products that are produced on the same type of processing and packing lines can be grouped into a family, these products may require completely different ingredients. Consequently, while the required capacity could be estimated accurately, it would not be possible to consider the suppliers when using aggregated product families. However, these suppliers clearly play an important role in determining the optimal product allocation.

Decomposition methods are another commonly used approach to solve large problem instances. Decomposition methods are based on reducing the model size by dividing the model into several smaller submodels. Sousa et al. (2011) and Terrazas et al. (2011) give a review of decomposition methods used in literature. Many of the decompositions they discuss are spatial or temporal decompositions. The submodels in a spatial decomposition can either relate to different echelons of the supply chain or to different physical locations within an echelon. Spatial decomposition is unsuitable for our tactical planning model since each of these submodels would still be very large as they would contain thousands of SKUs over 52 periods.

In temporal decomposition, the problem is decomposed into several independent submodels for each time period. Even though this would significantly reduce the size of our tactical planning model, the seasonality of the products and ingredients make temporal decomposition an unsuitable approach.

Castro et al. (2009) developed an order decomposition approach for a scheduling problem. Instead of trying to solve the often intractable full scale scheduling problem, they optimized the schedule for a subset of orders at a time. During each optimization, the allocation decisions of all previous orders are fixed. The timing and possibly the ordering decisions of the other orders can still be changed in their approach.

We have developed a method based on decomposing the problem into single SKU models. In each of these SKU models the decisions of all other SKUs are completely fixed. In the remainder of this paper, we will first give a problem description, then describe the tactical planning MILP model, subsequently explain the SKU decomposition method used to increase the tractability of this model and finally discuss the results.

Problem Description

The tactical planning problem for the FMCG industry addressed in this paper can be stated as follows. Given is a set of SKUs that have to be produced and distributed in a supply chain network over a one year horizon that is divided into weekly periods. This supply chain network consists of suppliers, factories, warehouses, distribution centers and retailers, whose locations are all fixed.

The availability and procurement costs of all ingredients at all suppliers are given for every period. The

ingredient unit transportation costs for every supplier-factory combination are also known. The ingredient inventory costs and the maximum ingredient inventory capacity at all factories are given as well. The ingredient consumption is linked to the production using given production recipes. For each factory, the aggregated available production time per week is given per type of processing and packing line. The processing and packing rates of all SKUs are also known. An SKU must be processed and packed in the same factory in the same period. An average changeover time and cost are given for each SKU.

The SKU unit transportation costs for all factory-warehouse, warehouse-distribution center and distribution center-retailer combinations are known. In addition, the initial SKU inventory, desired SKU safety stock, safety stock penalty, unit inventory cost and maximum total storage capacity are given for all warehouses and distribution centers. It is assumed that no stock is kept at the retailers, that the amount sent to a retailer may not exceed the demand and that demand can only be met in the period in which it occurs. The demand is given per retailer per SKU per week and the penalty cost for missed sales is given as well. Given this problem, the objective is to find a procurement, production and distribution plan that minimizes the total costs.

Model

This problem can be described by the MILP model that will be explained in this section. The objective of the model is to minimize the total costs. The total costs contain the procurement cost, transportation cost, changeover cost, storage cost, safety stock penalty cost and missed sales penalty cost.

The total amount of an ingredient transported from a supplier to all factories is restricted by the maximum available in each time period. The total amount of all ingredients stored at a factory may not exceed the total storage capacity. The total ingredient amount stored at a factory is equal to the amount stored in the previous period plus the amount received from all suppliers minus the ingredient consumption. The ingredient consumption at a factory is determined based on the recipes and the amount of each SKU produced at this factory.

The total amount of all SKUs that can be produced in a factory is limited by the available processing and packing time per type of SKU. An average changeover time is subtracted from the available packing time for every SKU that is assigned to a factory in each period. The assignment of SKUs to factories and periods is modeled by binary variables and no production is allowed if an SKU is not assigned.

The total amount of an SKU transported in a period from a factory to all warehouses must be equal to the total amount produced in this factory in this period. The total amount of all SKUs stored at a warehouse or distribution center may not exceed the inventory capacity. A unit penalty cost is incurred if the inventory of an SKU at a warehouse or distribution center is lower than the safety stock. The total amount of an SKU stored at a warehouse is equal to the amount stored in the previous period plus the amount received from all factories minus the amount sent to all distribution centers. The total amount of an SKU stored at a distribution center is equal to the amount stored in the previous period plus the amount received from all distribution centers minus the amount sent to all retailers. The total amount of an SKU sent to each retailer in each period may not exceed the demand. If the demand is not fully met, a missed sales penalty cost is incurred.

Algorithm

For realistic cases, containing thousands of SKUs and 52 time periods, this model would become extremely large. It would be in the order of tens of millions of continuous variables and hundreds of thousands binary variables. Therefore, we propose the following decomposition method based on single SKU models. These SKU models are the same as the full space model described above. However, the domain of all constraints and variables only includes a single SKU. These SKU models are solved sequentially for all SKUs.

If these SKU models are solved separately, they would not consider the capacity required by the other SKUs and we would obtain infeasible solutions. Therefore, in each model we consider the capacity required by the other SKUs by adding a parameter to all capacity constraints. For example, the storage capacity constraint in warehouses is given by the inequality:

$$WHinv_{i,w,t} + \sum_{i' \neq i} WHinvP_{i',w,t} \leq WHcap_w \quad \forall i, w, t \qquad (1)$$

where $WHinv_{i,w,t}$ is the amount of SKU i stored in warehouse w at the end of week t, $WHinvP_{i',w,t}$ is a parameter representing the amount of the other SKUs stored in warehouse w at the end of week t, and $WHcap_w$ is the capacity of warehouse w. After optimizing the model for SKU i, the parameter $WHinvP_{i,w,t}$ is set equal to the variable $WHinv_{i,w,t}$ for this SKU.

Even though this formulation yields feasible solutions, the quality of the solutions could be very poor. The reason is that the early SKUs could be produced anywhere without any real restrictions on the capacity, whereas the later SKUs could only be produced in those locations that have not been fully occupied by earlier SKUs. To account for the cost of moving SKUs to alternative locations or periods, we add a nonnegative slack variable to the capacity constraints:

$$WHinv_{i,w,t} + \sum_{i' \neq i} WHinvP_{i',w,t} \leq WHcap_w + \beta_{w,t} \quad \forall i, w, t \qquad (2)$$

With this slack variable β, we initially allow the total production to exceed the capacity. This ensures that the

later SKUs will be produced in those locations and periods that are optimal for them, rather than in the only available locations. A positive slack variable indicates that the capacity is exceeded and it thus indicates infeasibility. We add these slack variables with a penalty cost to the objective function. We then start with a penalty cost of zero and find the optimal plan for unlimited capacity. Next we slowly increase the penalty cost to force SKUs to be produced or stored in different locations until all slack variables are zero and we find a feasible solution. Basically, we continue to increase the penalty cost and solve the model for the next SKU until a feasible solution is obtained. Each SKU model only updates the decisions of the current SKU while keeping all other decisions fixed.

A summary of this SKU decomposition algorithm is given in Figure 2. Pen is the penalty cost applied to the objective function per unit that the capacity is exceeded, CO variables are the changeover variables, NSKU is the total number of SKUs and α is the fraction by which the penalty cost are increased after every optimization. The SKU decomposition algorithm requires the specification of an initial value of the penalty cost and the increase in penalty cost per optimization. These two parameters can be used to fine-tune the performance of the algorithm.

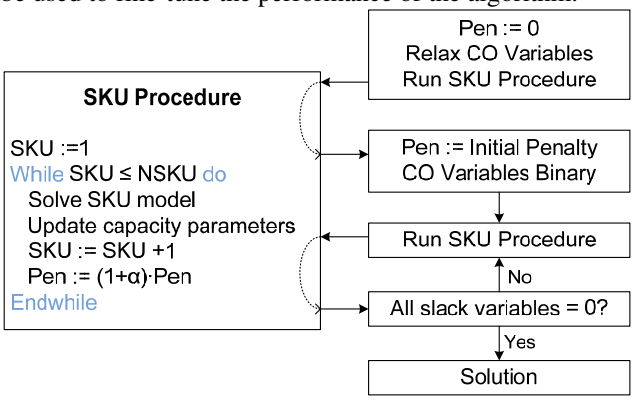

Figure 2. SKU decomposition algorithm

If the initial penalty cost is set to a relatively large number, a feasible solution will be found quickly because the penalty cost will soon be large enough to prevent any infeasible capacity allocation. However, this is also the reason why the solution quality could be poor. Already in the first iteration, the penalty cost will be high enough to prevent most over-allocation. Since the SKUs are optimized sequentially, the first few SKUs will be heavily reallocated to reduce the high penalty cost. On the other hand, the last few SKUs may hardly see any reallocation since the capacity might no longer be exceeded when these SKUs are optimized. Therefore, even though reallocating part of the later SKUs may be less expensive, the high initial penalty cost will force the reallocation mostly on the first SKUs. Alternatively, when starting with a low penalty cost, this phenomenon is greatly reduced since the initial cost of exceeding the capacity is very low. Therefore, the total reallocation in the first iteration will be low.

Similarly, a large increase in penalty cost per optimization would yield a feasible solution quickly but the quality could be worse. A feasible solution would be found quickly because the penalty cost would soon be large enough to prevent any infeasible allocation. However, quickly increasing the penalty cost may again lead to the wrong SKUs being reallocated. For example, consider a case with 10 SKUs and an α of 1%. After the optimization of the first SKU, the penalty cost is thus increased by a factor of (1+0.01). After a single iteration, all 10 SKUs have been optimized once and the penalty cost has increased by a factor of $(1+0.01)^{10}=1.1046$. Therefore, in the worst case, if SKU 10 is reallocated, reallocating SKU 1 could have been 10.46% less expensive. A larger number of SKUs thus requires a smaller α to give an equivalent increase in penalty cost per iteration.

Results

We will compare the SKU decomposition algorithm to the full space model for five example cases. In all cases, the supply chain consists of 10 suppliers, 4 factories, 5 warehouses, 10 distribution centers and 20 retailers. Cases 1 to 5 contain 10, 25, 50, 75 and 100 SKUs respectively. All cases contain 10 different ingredients and each SKU requires 2-3 ingredients. The data for these cases was randomly generated.

We performed all optimizations using Gurobi 4.0 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 were performed with a 1% MIP relative optimality tolerance. The computational results of the full space model are given in Table 1 and those of the algorithm are given in Table 2. Pen is the initial penalty cost, α is the increase in penalty cost, Time is the required CPU time and Deviation is the percentage increase in costs compared to the best obtained lower bound (LB). For case 1, this was the LB after the optimization of the full space model terminated once a solution within the 1% optimality gap was found. For cases 2 and 3, this was the LB after the optimization of the full space model was terminated after 6 hours. For cases 4 and 5, the optimization of the full space model did not yield a solution within 6 hours. For these cases, the LB was taken as the best solution found using the algorithm. A greater than sign is added to the deviation for these cases to indicate that the real deviation may be larger.

Table 1. Computational results full space models

Case	*Time*	*Deviation*
1	870s	0.31%
2	>6hr	1.50%
3	>6hr	83.8%
4	>6hr	No solution
5	1220s	Memory Error

Table 2. Computational results SKU algorithm

Case	*Pen*	*α[%]*	*Time [s]*	*Deviation*
	1	1	49	8.12%
	0.1	5	111	2.22%
1	0.1	1	418	1.71%
	0.1	0.5	803	1.65%
	0.01	1	1052	1.47%
	1	0.4	174	3.9%
	0.1	2	299	2.7%
2	0.1	0.4	1322	2.9%
	0.1	0.1	4589	2.3%
	0.01	0.4	2866	2.7%
	1	0.2	278	5.6%
	0.1	1	605	4.2%
3	0.1	0.2	2500	3.7%
	0.1	0.1	5156	3.3%
	0.01	0.2	5700	3.5%
	1	0.13	248	>6.9%
	0.1	0.65	1354	>1.0%
4	0.1	0.13	5548	>0.3%
	0.1	0.07	11323	>0%
	0.01	0.13	12464	>0.2%
	1	0.1	290	>8.2%
	0.1	0.5	1836	>1.0%
5	0.1	0.1	6700	>0.7%
	0.1	0.05	11570	>0%
	0.01	0.1	12621	>0.3%

The first thing that should be noted is that the optimization of the full space models for example cases 4 and 5 did not yield a feasible solution within 6 hours. The optimization of example case 5 was even terminated early due to a memory error. This confirms the need for a decomposition approach since a real case would be even larger than case 5. In fact, even in the optimization of the full space model for case 3, containing only 50 SKUs, the optimality gap was over 80% after 6 hours.

The SKU decomposition algorithm was able to find solutions for all five cases within reasonable computational time. As can be seen in Figure 3, the required computational time increases roughly linearly with the number of SKUs. This is promising since a real case would contain thousands of SKUs. This linear relationship between the number of SKUs and the required CPU time is also consistent with the model sizes. For all five cases, each SKU model contains approximately 7k constraints, 24k continuous variables and 208 binary variables. Per iteration, one model for each SKU has to be solved. Therefore, the number of models that have to be solved per iteration increases linearly with the number of SKUs. Assuming that the required number of iterations remains constant, the required CPU time will increase linearly with the number of SKUs.

The data points in Figure 3 deviate slightly from the linear trend lines because of differences in data set. The minimum penalty cost required to obtain a feasible solution is dependent on the data. For example, a higher capacity utilization would most likely lead to a higher final penalty cost. Therefore, depending on the data, the number of iterations required to reach this penalty cost may vary, even if the initial penalty cost and the penalty increase are kept constant. In addition, the time required to solve the individual SKU models can deviate depending on the data. Nevertheless, the linear trend is clearly visible in Figure 3.

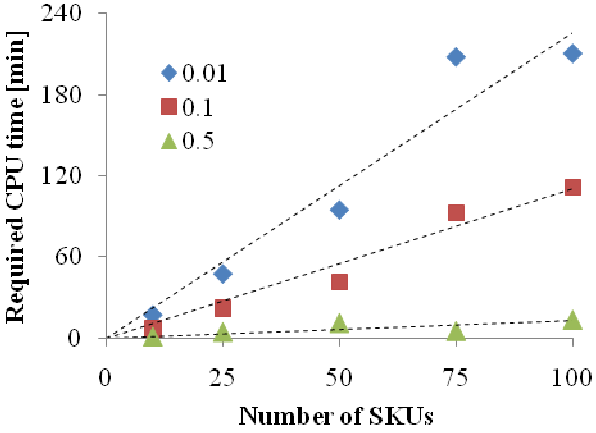

Figure 3. Required CPU time versus the number of SKUs for three different initial penalty values. In all optimizations, the α was equivalent to an α of 1% for 10 SKUs. Linear trend lines are added.

On the other hand, for the full space model the size increases from 42k constraints, 185k continuous variables and 2080 binary for case 1 to 235k constraints, 1.5M continuous variables and 21k binary variables for case 5. This explains why it is difficult to obtain solutions for the full space model for larger cases.

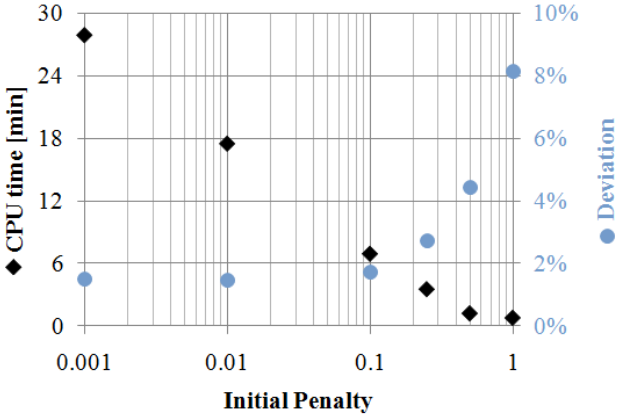

Figure 4. Deviation from optimality and required CPU time of case 1 optimized using the algorithm for different initial penalty values. In all optimizations α was 1%.

The optimization of the full space models for cases 1 and 2 yielded solutions with lower costs than those that could be obtained with the decomposition algorithm. Nevertheless, in both cases the algorithm was able to

obtain solutions within a few percent of the optimal solution. As expected, a lower initial penalty cost or penalty increase leads to better solutions. Although, as can be seen in Figure 4 and 5, decreasing the initial penalty cost or penalty increase beyond a certain threshold has little impact on the solution quality. Decreasing these values increases the required computational time as the number of required iterations before the penalty cost is high enough to prevent all infeasibilities increases.

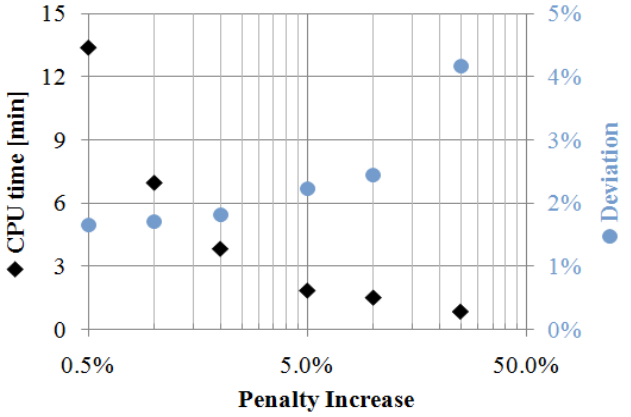

Figure 5. Deviation from optimality and required CPU time of case 1 optimized using the algorithm for different penalty increase values. In all optimizations the initial penalty value was 0.1.

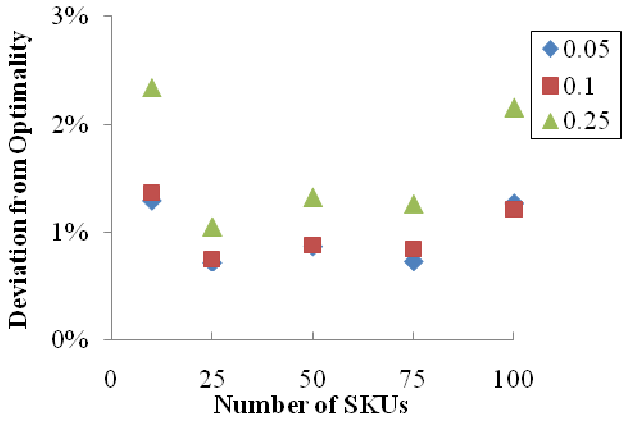

Figure 6. LP relaxation: Deviation from Optimality when using the algorithm for different initial penalty values. In all optimizations, the α was equivalent to an α of 1% for 10 SKUs.

It is difficult to compare the degree of optimality of the algorithm for various numbers of SKUs since the full space models for 50,75 and 100 SKUs could not be solved to optimality. Therefore, an LP relaxation of all five cases was optimized using both the full space model and the algorithm. The deviation from optimality for various initial penalty values is shown in Figure 6. Clearly, different data sets lead to different deviations from optimality. However, increasing the number of SKUs does not seem to significantly increase or decrease the deviation from optimality. Most importantly, all five cases could be optimized within a 1.5% deviation from optimality. While the binary variables may influence the optimality gap of the algorithm, it seems unlikely that they would introduce a strong correlation between the number of SKUs and the optimality gap of the algorithm.

Conclusions

We have developed an SKU decomposition algorithm for a tactical planning model in the FMCG industry. For two small scale cases, containing 10 and 25 SKUs, this algorithm was able to obtain solutions within a few percent of the optimal solution. Even for these small cases, the algorithm required considerably less CPU time than the full space models to obtain these solutions. The algorithm was also able to optimize a larger 100 SKU case for which the full space model could not be optimized.

The main advantage of the SKU decomposition algorithm is that it scales approximately linearly with the number of SKUs. This should allow the algorithm to be used in real cases containing thousands of SKUs.

It is difficult to confirm whether there is a correlation between the number of SKUs and the degree of optimality of the algorithm because the full space model for even relatively small cases could not be solved to optimality. However, no direct correlation between the number of SKUs and the degree of optimality of the algorithm for the LP relaxation of the model was observed. Therefore, we conclude that the SKU decomposition algorithm seems a suitable approach for solving cases of a realistic size.

Acknowledgements

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

References

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

Castro, P.M., Harjunkoski, I., and Grossmann, I.E. (2009). Optimal Short-Term Scheduling of Large-Scale Multistage Batch Plants. *Ind. Eng. Chem. Res., 48(24), 11002-11016*

Omar, M. K. and Teo, S. C. (2007). Hierarchical production planning and scheduling in a multi-product, batch process environment. *Int. J. Prod. Res.*, *45*, *5, 1029-1047*.

Sousa, R.T., Liu, S., Papageorgiou, L.G., and Shah, N. (2011). Global supply chain planning for pharmaceuticals. *Chem. Eng. Res. Des., doi:10.1016/j.cherd.2011.04.005*

Terrazas-Moreno, S., Trotter, P., and Grossmann, I.E. (2011) "Temporal and spatial Lagrangean decompositions in multi-site, multi-period production planning problems with sequence-dependent changeovers," to appear in *Computers and Chemical Engineering* (2011).