CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

An Optimization Method for the Simultaneous Design of a Product Family and its Related Supply Chain Using a Taboo Search Algorithm

Radwan El Hadj Khalaf
Bruno Agard
Bernard Penz

September 2009

CIRRELT-2009-35

Bureaux de Montréal :

Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :

Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Université Laval · UQÀM · HEC Montréal · École Polytechnique Montréal · Université de Montréal

An Optimization Method for the Simultaneous Design of a Product Family and its Related Supply Chain Using a Taboo Search Algorithm

Radwan El Hadj Khalaf[1], Bruno Agard[2,*] Bernard Penz[1]

[1] G-SCOP Laboratory, Grenoble INP-CNRS-UJF, 46, Avenue Félix-Viallet, 38031 Grenoble Cedex 1, France

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal, Canada H3C 3A7

Abstract. Product family design is currently facing a multitude of challenges, the main problem stemming from the diversity offered to consumers. To design a product family, designers have to define an efficient bill of materials which ensures product assembly within a predefined length of time in order to satisfy the synchronized delivery principle. In addition, the modules used to assemble the finished products have to be competitive in terms of logistical costs. The ability to anticipate the constraints associated with the production process and with transportation is consequently of great interest. In this paper, we focus on the process of identifying a set of modules to be used in the assembly of the finished product. The objective is to define the bill of materials for each product from the modules belonging to that set, and to assign these modules to distant facilities where they will be manufactured and then shipped to a nearby facility for final assembly within a specific time. We use a set partitioning formulation to represent the problem, and solve it by adapting a Taboo Search algorithm in which the assembly process and the supply chain design are considered at the same time.

Keywords. Product family, supply chain, set partitioning, taboo search.

---

* Corresponding author: Bruno.Agard@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec,
Bibliothèque et Archives Canada, 2009

1 Introduction

Faced with strong competition among the various products available on the market, customers are becoming more and more demanding. Every product sold needs to fit customer requirements exactly in terms of functionality and do so at the lowest possible price. To meet these customer expectations, manufacturers often produce a large number of different products, which may lead to excessive product diversity. To decrease production costs while at the same time ensuring the required diversification, manufacturers can standardize products and adopt an assemble-to-order production policy, which makes it possible for them to offer a wide range of final products from a limited number of semi-finished components, often called modules [23].

Lean production has brought about major changes in the supplier-retailer relationship. Today, lead time is a major issue in contract negotiations [6]. For the supplier, it is essential to fill an order within a specific time period, although delays are costly for both the retailer (through penalty costs charged for non respect of due dates) and the supplier (through production rescheduling costs stemming from component shortages).

In this context, new design strategies, like product family design, are being developed. Product family design must take into account not only product diversity, but also definition of the process and the supply chain [25]. A consistent approach is needed in order to guarantee customer satisfaction, as well as to minimize the total investment on the part of producers in the product and in the operating costs incurred by the global supply chain. Rai et al. [21] present a two-step approach to determine the optimal platform level for a selected set of product families and their variants. The first step employs a multi-objective optimization method using an agent-based framework to determine the Pareto design solutions for a given set of modules. In the second step, a post optimization analysis is performed to determine the optimal platform level. Montreuil et al. [19] present the various types of personalized manufacturing processes in a mass customization context to offer innovative, highly personalized products with short and reliable delivery times. He characterizes complementary types of personalization and highlights the key elements required for their successful implementation. Finally, he shows how personalization affects the design of the supply and demand network. Poulin et al. [20] present a framework comprising eight personalization options which can be combined to form a complete personalized offering. Then, using the analogy of golf clubs (irons) for illustration purposes, he contrasts the impact on that network.

Global design modeling has been studied recently by a number of authors. Agard et al. [1] propose a genetic algorithm to minimize the mean assembly time of a finished product for a given demand, and Agard and Penz [2] propose a model for minimizing module production costs and a solution based on simulated annealing. Da Cunha et al. [7] also propose a simulated annealing algorithm to determine the composition of the stock modules of a given size to minimize the mean assembly time of finished products. However, these models do not consider the variable costs arising from the number of modules to be

manufactured. Lamothe et al. [17] use a generic bill of materials representation to simultaneously identify the best bill of materials for each product and the optimal structure of the associated supply chain, although this approach requires that a predefined generic bill of materials be generated for the product family. Briant et al. [4] explore a Lagrangian relaxation method to resolve the diversity management problem, which consists of choosing an optimal set of some given number of configurations k that are produced, any non produced configuration being replaced by the cheapest one produced that is compatible with it. Electrical wiring in European car factories serves as an illustration of this.

The problem considered here consists of defining the best set of modules that permits a final assembly within a predefined length of time. An important specificity is to offer the products as demands that exactly match the needed functionality without extra options. We handle this policy using the set partitioning formulation.

Set Partitioning (SP) has numerous real life applications, and the literature offers SP solutions, as well as joint solutions, to many problems. Below, we provide an overview of the various problem-solving techniques that contain the SP formulation. Wu et al. [24] present an optimization allocation algorithm to solve a matching problem between the resources needed for multi-tasking and the M-dimensional resources offered by multi-processing optimization nodes. Ronnqvist et al. [22] develop a repeated matching algorithm to resolve a home care staff planning problem, the objective being to develop visiting schedules for care providers that incorporate some restrictions and soft objectives. Kotecha et al. [16] present a genetic algorithm using a new cost-based uniform crossover for solving an airline crew scheduling problem. Hanczar [15] proposes a new method for solving the vehicle routing problem using a very easily implemented branch-and-bound procedure, the efficiency of which is proved by solving a set of test problems. Akker et al. [3] use a combination of column generation and Lagrangian relaxation to tackle a single-machine common due date problem, where Lagrangian relaxation is exploited for early termination of the column generation algorithm and for speeding up the pricing algorithm. Bronmo et al. [] develop and experimentally compare policies for the control of a system of k elevators with a capacity of 1 in a transport environment with l floors. Friese et al. [10] present a multi-start local search heuristic for a typical ship scheduling problem, where a large number of initial solutions are generated by an insertion heuristic with random elements, the best initial ones being improved by a local search heuristic that is split into a quick and an extended version.

A detailed description of the problem is provided in the following section. Notations are explained in subsection 2.1, and then a Mixed Integer Linear Program model is given in subsection 2.2. The Taboo Search (TS) algorithm is then presented in section 3. Computational experiments are given and analyzed in section 4. Finally, concluding remarks and perspectives are proposed in section 5.

2 Problem Presentation

Consider the following industrial context (Figure 1), which is similar to a problem treated in [9]. A producer receives customer orders for finished products containing options and variants. Each individual product is then manufactured from modules provided by various suppliers.

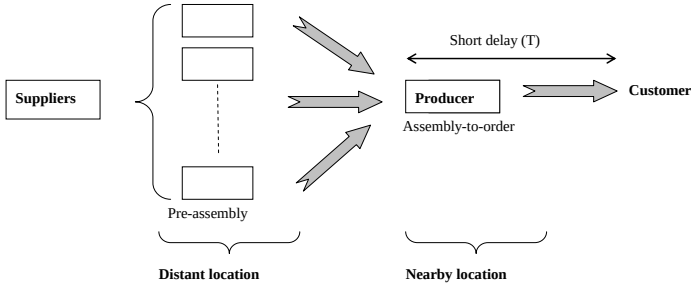

Figure 1: Structure of the supply chain

The producer has only a short time (T) in which to respond to each customer's order. This time is less than the time required to assemble the products from elementary components. In addition, the producer has to provide the product precisely according to the customers' requirements (without extra options). This constraint comes either from technical considerations or simply to avoid the supplementary cost of offering non requested options.

To satisfy customer orders, the producer brings in preassembled components, called modules, from many suppliers located at facilities around the world. The production costs incurred by these suppliers are low. The modules are then assembled at the producer's facility, which, we assume, is close to the customers, and thus is characterized by a rapid reaction time and a short lead time.

The strategic problem is, then, to design the product family, *i.e.* to determine the bill of materials for each product. A product will be made up of a set of modules. For modules that appear in at least one bill of materials, we have to determine where those modules must be produced in order to minimize production and transportation costs.

2.1 Notations

A product (or a module) is considered as the set of functions that it contains. It is currently modeled with a binary vector in which 1 means that the function is present in the product (or module) and 0 otherwise.

- A function F_k is a requirement that could be included in a finished product.
- A module M_j is an assembly of functions that could be added to other modules to make a finished product.

- A finished product P_i is an assembly of modules that corresponds exactly to at least one customer demand.

Let us introduce the following notations:

- $\mathcal{F} = \{F_1, ..., F_q\}$: the set of q functions that can appear in both finished products and modules;
- $\mathcal{P} = \{P_1, ..., P_n\}$: the set of n possible finished products that may be demanded by at least one customer (note that D_i is the estimated demand of product P_i during the life cycle of the product family);
- $\mathcal{M} = \{M_1, ..., M_m\}$: the set of m possible modules.
- $\mathcal{S} = \{S_1, ..., S_s\}$: the set of s distant production facilities where a site S_l has a production capacity W_l.
- F_j^A : the fixed cost of module M_j at the nearby facility (management costs);
- V_j^A : the variable cost of module M_j at the nearby facility (cost of assembly, storage, transportation, etc.);
- F_{jl}^P : the fixed cost of module M_j at the distant facility S_l (management)
- V_{jl}^P : the variable cost of module M_j at the distant facility S_l (cost of assembly, storage, etc.);
- t_j : the time required to assemble module M_j in a finished product;
- T : the maximum assembly time available;
- W_{jl} : the work load generated by producing one module M_j at facility S_l.
- W_l : the work load capacity available at facility S_l.

Under these assumptions, a product (or module) is represented by a binary vector of size q. Each element shows whether the corresponding function is required in the product (value = 1) or not (value = 0). The set $\mathcal{M}$ contains m modules. $\mathcal{M}$ may be all the possible modules in the whole combinatory, or a subset of those modules.

In terms of the manufacturing process: (1) the producer assembly line costs must be minimized; and (2) the final assembly time must be less than the available time, in order to respect the delivery time for the customers. In terms of supply chain design: (1) every distant facility cost is considered (with fixed and variable costs for each possible module); and (2) the total workload at each production facility must be less than its own production capacity.

The problem is now to determine the subset $\mathcal{M}' \in \mathcal{M}$, of minimum cost, such that all products in $\mathcal{P}$ can be built in a constrained time window T. Concerning the products, the goal is to determine which bill of materials is the most suitable (Figure 2).

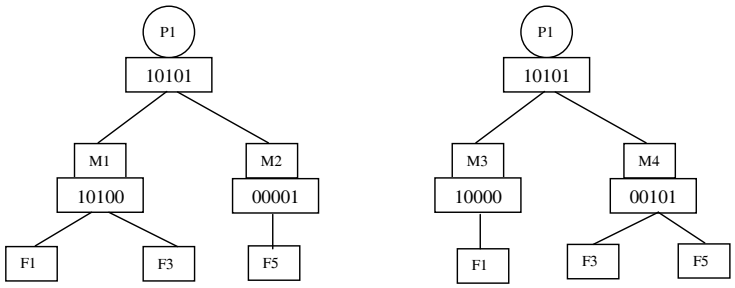

Figure 2: Alternative bills of materials

In terms of the manufacturing process: (1) the producer assembly line costs must be minimized; and (2) the final assembly time must be less than the available time, in order to respect the delivery time for the customers. In terms of supply chain design: (1) every distant facility cost is considered (with fixed and variable costs for each possible module); and (2) the total workload at each production facility must be less than its own production capacity.

2.2 Mathematical Modeling

The problem is modeled using a Mixed Integer Linear Program formulation []. The objective is to minimize all the costs linked to the activities of the producer and the suppliers. The objective of the model proposed in this section is to determine the optimal bills of materials that minimize the assembly and production costs (fixed and variable) at the same time.

$$Z = min\left(\sum_{j=1}^{m} F_j^A Y_j + \sum_{j=1}^{m} V_j^A \left(\sum_{i=1}^{n} D_i X_{ij}\right)\right) + \left(\sum_{l=1}^{s}\sum_{j=1}^{m} F_{jl}^P Y_{jl} + \sum_{l=1}^{s}\sum_{j=1}^{m} V_{jl}^P Q_{jl}\right) \quad (1)$$

s.t.

$$AX_i = P_i \qquad \forall i \in \{1,\cdots,n\} \quad (2)$$

$$\sum_{j=1}^{m} t_j X_{ij} \leq T \qquad \forall i \in \{1,\cdots,n\} \quad (3)$$

$$X_{ij} \leq Y_j \qquad \forall i \in \{1,\cdots,n\}\ \forall j \in \{1,\cdots,m\} \quad (4)$$

$$\sum_{l=1}^{s} Q_{jl} = \sum_{l=1}^{s} D_i X_{ij} \qquad \forall j \in \{1,\cdots,m\} \quad (5)$$

$$\sum_{j=1}^{m} W_{jl} Q_{jl} \leq W_l \qquad \forall l \in \{1,\cdots,s\} \quad (6)$$

$$Q_{jl} \leq B Y_{jl} \qquad \forall j \in \{1,\cdots,m\}\ \forall l \in \{1,\cdots,s\} \quad (7)$$

$$Y_j, X_{ij} \in \{0,1\} \qquad \forall i \in \{1,\cdots,n\}\ \forall j \in \{1,\cdots,m\} \quad (8)$$

$$Q_{jl} \geq 0 \ \ integer \qquad \forall j \in \{1, \cdots, m\} \ \forall l \in \{1, \cdots, s\} \qquad (9)$$

$$Y_{jl} \in \{0, 1\} \qquad \forall j \in \{1, \cdots, m\} \ \forall l \in \{1, \cdots, s\} \qquad (10)$$

where: $X_{ij} = 1$, if module M_j is used in the bill of materials of product P_i, 0 otherwise; $Y_j = 1$ if module M_j is selected (then M_j belongs to $\mathcal{M}'$, the set of selected modules), 0 otherwise; A is the binary matrix, column j of which is the vector M_j; and X_i is the column vector composed of the variables X_{ij}; $Y_{jl} = 1$, if module M_j is produced at facility S_l, 0 otherwise; and Q_{jl} is the quantity of module M_j produced at facility S_l.

The objective function contains two major components:

$$Z^A = \left(\sum_{j=1}^{m} F_j^A Y_j + \sum_{j=1}^{m} V_j^A (\sum_{i=1}^{n} D_i X_{ij}) \right)$$

which represents the costs incurred at the nearby facility, where $\left(\sum_{i=1}^{n} D_i X_{ij} \right)$ corresponds to the total demand of module M_j and,

$$Z^P = \left(\sum_{l=1}^{s} \sum_{j=1}^{m} F_{jl}^P Y_{jl} + \sum_{l=1}^{s} \sum_{j=1}^{m} V_{jl}^P Q_{jl} \right)$$

which represents the costs occurring at all distant location facilities.

Constraint (2) shows that a finished product P_i must be assembled exactly according to customer requirements. Constraint (3) indicates that products must be assembled within the time window T, in order to respect the delivery time. Constraint (4) states that, if module M_j is used in the bill of materials of product P_i, then module M_j must be produced somewhere. Constraint(5) indicates that the production of a module M_j must satisfy the overall quantities required. Constraint (6) shows that total production at facility S_l must not exceed that facility's capacity. Constraint (7) expresses the relation between the variables Q_{jl} and Y_{jl} (B is a large constant). A module M_j can be produced at S_l only if M_j is assigned to S_l ($Y_{jl} = 1$).

The problem described here contains the set partitioning problem [11]. We conclude that it is NP-hard in the strong sense. For this, the following section explores the use of a TS algorithm to resolve large instances.

3 Resolution with a Taboo Search Algorithm

The Taboo Search (TS) is a metaheuristic approach designed to find a near optimal solution of combinatorial optimization problems [12], [13] and [14]. The algorithm can be sketched as follows. There is a set F of feasible solutions. A move is defined as an operation or a function which transforms a solution $x_1 \in F$ into another solution $x_2 \in F$. For any solution $x \in F$, a subset of moves

applicable to it is defined as its neighborhood $N(x) \subseteq F$. TS starts from an initial solution. At each step the neighborhood $N(x)$ of a given solution x is searched in order to find a neighbor x'. The move, which leads to the neighbor x', is performed. and the newly obtained solution is set as the origin for the next step. In order to prevent cycling, a structure, called the taboo list, of length L (fixed or variable) is introduced to prevent returning to a solution visited in the last L iteration. The taboo list is often interpreted as a limited queue of length L containing forbidden moves.

TS techniques have recently been used to treat set-partitioning problems. Lee et al. [18], for example, proposes? an efficient heuristic using TS and by solving set partitioning problems for determining the composition of a vehicle fleet and traveling routes. They perform an optimal vehicle allocation for the set of routes whenever a new feasible solution is obtained in an iteration of the TS. We use exactly the same technique in our problem: we perform an optimal module assignment to the distant facilities whenever a new feasible subset $\mathcal{M}^{'}$ is found in the TS iterations.

For this problem, it is difficult to define moves that transform one feasible solution into another, new feasible solution. In order to do so, we use a group of moves (Figure 3) which are used consecutively until that new feasible solution is found. These moves are described in the elimination and insertion neighborhoods. Before introducing the description of the TS algorithm, let us introduce the following terminology:

- A module M_j is compatible with a product P_i if it does not contain extra functions for this product. Then, a finished product can only be assembled from compatible modules (because we require an exact assembly of each product demanded).
- The degree of the module M_j is the number of finished products compatible with the module M_j.
- $\mathcal{M}^{'}$ is the subset of modules selected in the current solution.
- A product P_i is not feasible if it does not admit a bill of materials from modules in $\mathcal{M}^{'}$.
- The logistical costs of a module are costs (fixed and variable) generated by the production of the module at the distant facilities.

3.1 Taboo Search algorithm

The algorithm (Figure 3) begins with an initialization phase (Steps 1 and 2) which generates an initial solution. In Step (1), the bill of materials for each product is determined. This is ensured by solving the integer linear program (ILP) formed by constraints (2), (3), (4) and (8). We note here that we determine the bill of materials of products one by one. We then solve the above ILP by freezing the variable i (which represents the finished products) each time.

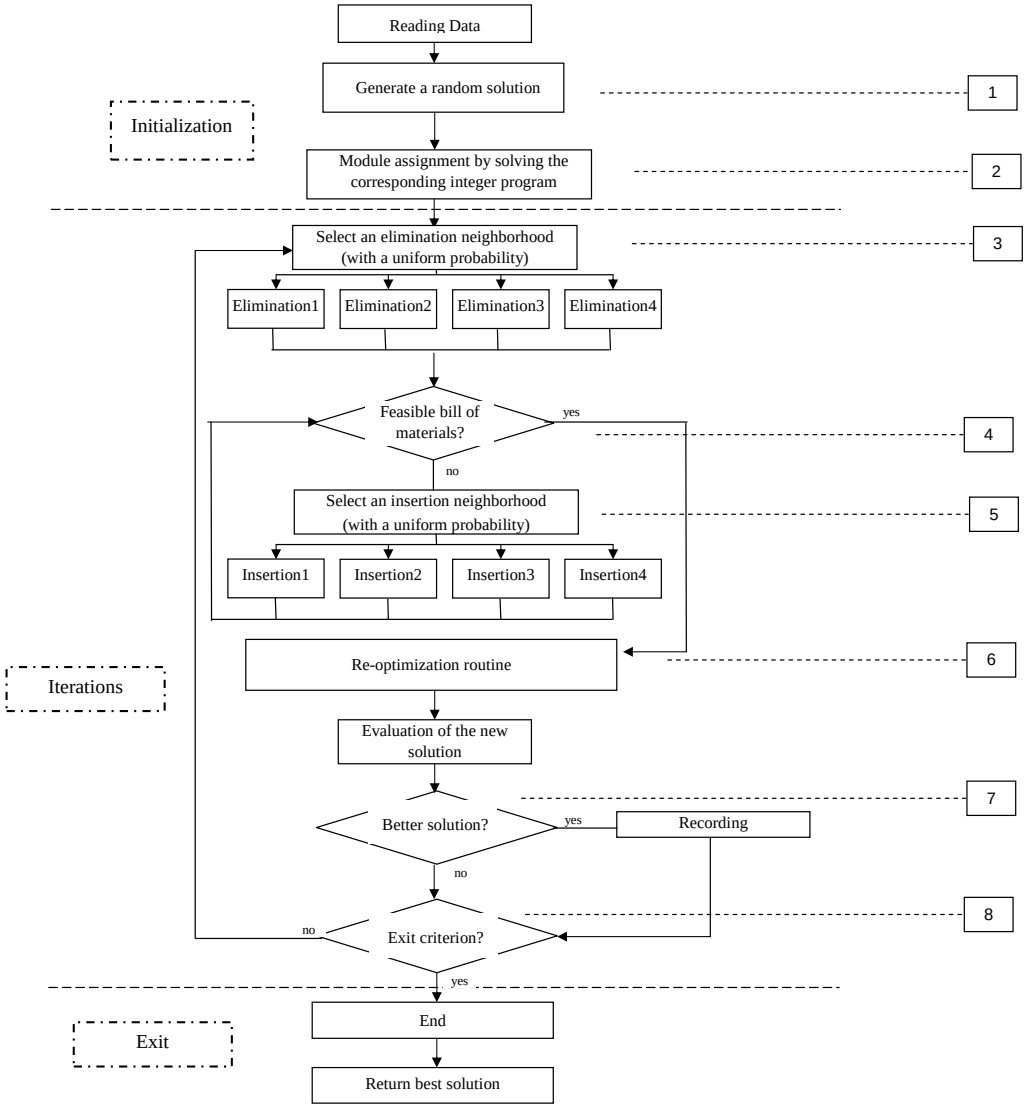

Figure 3: The Taboo Search algorithm

Owing to the size of the problem, it is possible to do so exactly with an efficient ILP solver. The chosen modules are assigned to distant facilities (Step 2) by solving the ILP formed by constraints (5), (6), (7), (9)and (10). This program is easily solved by the ILP solver, because the number of chosen modules is always very small compared with the whole combinatory, and also because the number of distant facilities is generally limited in real supply chains.

The iteration phase (Steps 3 to 8) consists of constructing a new subset of modules, which allows the bill of materials of all finished products to be defined from the current subset. This can be done by eliminating a module (Step 3) from the current subset $\mathcal{M}'$ and then inserting new modules (Step 5) until the feasibility of all finished products is proved.

Once the new subset is constructed and new bills of materials for non feasible products are determined (the bill of materials of finished products can be checked and updated in Step 4), the new solution is evaluated and recorded if it is better than the current best solution (Step 7). This process is iterative until the exit criterion is reached. The algorithm then returns the best solution recorded (Exit phase), that is, the best subset $\mathcal{M}'$, the bills of materials of the corresponding product, and the related module assignments.

Elimination and insertion operations are controlled by taboo lists of a specific length. Thus, we keep in memory the latest modules that have been eliminated (or inserted) and we prohibit their insertion (or elimination) within a specific number of iterations.

Below, we provide a more detailed description of the various routines and neighborhoods used in the algorithm.

3.2 Bill of materials feasibility control (Step 4)

After elimination (or insertion) of a module from (or into) $\mathcal{M}'$, this routine takes the non feasible products (those that contain the eliminated module in their current bill of materials) one by one, and checks the feasibility of their bill of material by solving the ILP formed by constraints (2), (3), (4) and (8). If the product admits a bill of materials by the new subset $\mathcal{M}'$ then its bill of materials will be updated, otherwise it will still be considered as an non feasible product and its feasibility will be checked again in the next iteration.

After elimination (resp. insertion) of a module from (resp. in) $\mathcal{M}'$, this routine takes the infeasible products (those which contain the eliminated module in their current bill of materials) one by one and check their bill of material feasibility by solving the integer linear problem formed by constraints (2), (3), (4) and (8). If the product admits a bill of materials by the new subset $\mathcal{M}'$ then its bill of materials will be updated, otherwise it will be still considered as an infeasible product and its feasibility will be checked again in the next iteration.

3.3 Re-optimization routine (Step 6)

The purpose of this routine is to clean the new subset $\mathcal{M}'$ by eliminating the modules that do not belong to any bill of materials (inserted in the previous iterations and not used). It also updates some data used by the other routines (such as module degrees and quantities needed for each module), and assigns the modules of $\mathcal{M}'$ to the distant facilities by optimally solving the ILP formed by constraints (5), (6), (7), (9) and (10). The objective function here is formed only from the logistical costs Z^P.

3.4 Elimination Neighborhoods (Step 3)

Four elimination neighborhoods are considered:

- Elimination 1: Eliminates a low-degree module. A module belonging to a small number of bills of materials is randomly selected and eliminated from the subset $\mathcal{M}^{'}$.
- Elimination 2: Eliminates a module that generates high logistical costs. As with the previous move, the logistical costs of modules in $\mathcal{M}^{'}$ are calculated and one of them, with high logistical costs, is randomly selected.
- Elimination 3: Eliminates a large-degree module (because it will certainly generate high variable costs).
- Elimination 4: Random elimination.

3.5 Insertion Neighborhoods (Step 5)

Four insertion neighborhoods are considered:

- Insertion 1: Inserts a module that generates low logistical costs. This marks such modules, and one of them is randomly inserted into $\mathcal{M}^{'}$.
- Insertion 2: Inserts a large-degree module. The selection criterion here is the absolute degree, which is the number of finished products compatible with the module, regardless of the feasibility of the finished products.
- Insertion 3: Inserts a large-degree module (by looking only at non feasible products). The selection criterion here, unlike that in the previous move, is the relative degree, which is the number of non feasible finished products compatible with the module.
- Insertion 4: Inserts a module that allows the bill of materials feasibility for an non feasible product. This can be done by taking one non feasible finished product P_i and solving the ILP formed by constraints (2), (3), (4), and (8), and for which the objective function is the minimization of the sum of X_{ij} variables of modules that do not belong to $\mathcal{M}^{'}$. $(min \sum_{j \in \mathcal{M} \setminus \mathcal{M}'} X_{ij})$

3.6 The Exit Criterion (Step 8)

Various exit criteria are considered simultaneously: computational time and number of iterations without improvement of the objective function.

4 Computational Experiments

The objective in this section is to define the experimentation framework and to analyze the solution quality and algorithm efficiency on the generated instances. For this, five instances have been generated on which the module set, the finished product set, the distant facility set, the demand D_i, the assembly operating times w_j, the distant facility capacities, the module production loads, the distant facility costs, and the nearby facility costs are fixed.

The experiments were conducted on a model with one assembly site (nearby location) and four production sites (distant locations) which compete for the production of the modules. The distant sites have limited production capacity, while the nearby site is assumed to have an unlimited production capacity.This guarantees solutions for any instance.

The problem data were fixed as follows: $q = 15$, $n = 500$, where each product has at least 5 functions and at most 10, $m = 30826$ (all modules compatible with the finished products). The assembly operating times t_j were fixed to 1, and T was varied from 3 to 5.

Three cost files were randomly generated. For the "Cost 1" problem, the assembly costs are greater than the logistical costs; for the "Cost 2" problem, the two costs are almost equivalent; and for the "Cost 3" problem, the logistical costs are the highest.

The tests were carried in C++ with the Ilog Cplex9.0 library. They were solved on a DELL station / 2.8 GHZ/ 1Go RAM. The computational time was fixed to six hours for the TS algorithm.

4.1 Influence of the initial solution

The objective in this section is to analyze the TS convergence speed and the final solution quality after six hours of computational time, depending on the initial solution used. For this, we compare the algorithm's efficiency by starting with three different initial solutions: a randomly generated solution, the MSH1 solution, and the MSH2 solution.

A randomly generated solution This initial solution can be obtained by freezing the index i and solving n ILPs formed by constraints (2), (3), (4) and (8). The objective function could, for example, be the maximization of the number of modules per bill of materials: $max \sum_{j=1}^{m} X_{ij}$.

After determining an initial bill of materials for each product, we then assign the resulting modules to the distant facilities by solving the ILP formed by constraints (5), (6), (7), (9) and (10). The objective function remains, of course, the minimization of the logistical costs Z^P.

MSH1 This is a greedy heuristic, in which the objective is to determine efficient bills of materials of finished products so as to minimize the assembly costs Z^A. Its principle is quite simple, the idea being to select interesting

modules and insert them into the bills of materials of compatible finished products. A module is deemed interesting if it has attractive (low) costs. We introduce the following criterion to calculate the attraction level of a module: $ind1_j = F_j^A + (V_j^A \sum_{i \leftrightarrow j} D_i)/Deg_j$, where $i \leftrightarrow j$ means that we have to sum demands D_i of products P_i that are compatible with the module M_j for which we are calculating the criterion. Deg_j represents the degree of M_j. This criterion takes into account the module costs (fixed and variable) at the nearby facility and the number of products with which the module is compatible. A detailed description of this heuristic is given in [8]. As with the previous method, we then solve the linear subprogram to assign the resulting modules.

MSH2 With this heuristic, we improve the module selection criterion so as to take into account the logistical costs, and at the same time proceed with module selection and assignment of the module to the appropriate distant facility. So, $ind2_j = ind1_j + C_j^P$ where $C_j^P = min_{l=1}^{s} F_{jl}^P + (V_{jl}^P \sum_{i \leftrightarrow j} D_i)$ such that $W_{jl} \sum_{i \leftrightarrow j} D_i \leq W_l$ is the cost of assigning the module M_j to the facility with the lowest costs (if there is capacity available). With this criterion, we can simultaneously select the most interesting module (in terms of costs), construct the bills of materials around the selected module, and assign it to the lowest-cost facility. Of course, at each iteration, the production capacities at the distant facility are updated according to the modules assigned.

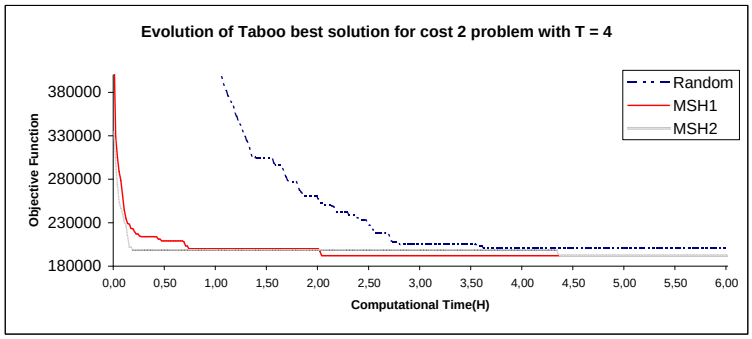

Figure 4: Evolution of the best TS solution taking into account the various starting points for the cost 2 problem

Figure 4 shows that the MSH2 heuristic gives the best initial solution and the best final one (this is true for the three cost configurations). We also note that using heuristics to determine an initial solution makes it possible to accelerate the TS convergence speed. In fact, the gap between the random initial solution curve and the heuristic curves is very large at the first iterations. As the TS proceeds, this gap tightens and becomes small at the end of execution. However, the gap remains significant, even at the end of execution, so all the following tests will be carried out with the TS that uses the initial MSH2 solution.

4.2 Influence of neighborhoods

In this section, we analyze TS efficiency when we freeze the elimination and insertion neighborhoods. For this, we compare the solution of the standard algorithm (with a uniform random selection of elimination and insertion neighborhoods) with the solution of each pair of neighborhoods (elimination i, insertion j).

Table 1 shows that in all the cases the standard algorithm gives the best results, except when the pair (elimination 1, insertion 4) is used. This result has been confirmed for the three cost configurations.

	Solution of the standard taboo algorithm: 192573			
	Ins1	Ins2	Ins3	Ins4
Elm1	192857	196415	205180	183822
Elm2	209770	200848	199353	199426
Elm3	220083	221880	210907	249036
Elm4	233244	225745	234116	227718

Table 1: Standard TS solution compared with TS solutions after freeze of elimination and insertion neighborhoods for the instance 1 and cost 2 problem

All the following tests will be conducted with the TS algorithm for which the initial solution is generated by the MSH2 heuristic, and the elimination neighborhood is frozen at elimination 1 (elimination of a module of small degree), and the insertion neighborhood is frozen at insertion 4 (insertion of a module to guarantee the feasibility of a non feasible finished product).

4.3 Evolution of the best TS solution value

Figure 5 represents the evolution of the best solution found and the local solution with the computational time for the instance 1 and cost 2 problem with $T = 4$. For the other instances, costs, and delays, the curve shapes are almost the same.

This figure shows a major fall in the value of the best solution objective function at the first iterations, which indicates that the TS algorithm very quickly eliminates from the initial solution the modules that are not of interest. Then, the TS algorithm needs more computational time to improve the objective function because those interesting modules have been already detected. We also note the large fluctuation of the local solution in some areas and its small fluctuation in another. The large fluctuation comes from the taboo lists, which make it possible to explore a large solution space, and the small fluctuation indicates that the TS algorithm is stuck in a local minimum. However, we can also note the TS algorithm's capacity to reduce the objective function of the local solution quickly when it is high, which confirms the efficiency of the neighborhoods used.

(Table 2) shows the mean values of the initial and final solution objective function of the five instance files, for a given cost file configuration and delay value. This table indicates that:

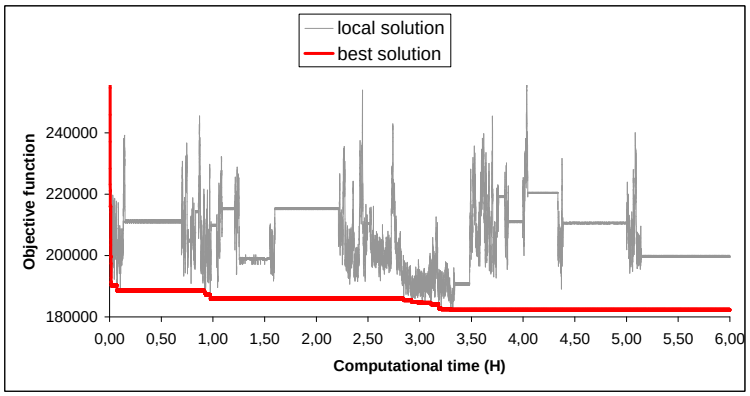

Figure 5: The objective function evolution of the local and best TS solutions

Cost Configuration	T=3			T=4			T=5		
	Initial Solution	Final Solution	Gap(%)	Initial Solution	Final Solution	Gap(%)	Initial Solution	Final Solution	Gap(%)
Cost1	546574	268974	51%	334476	146580	56%	140475	101285	28%
Cost2	503874	278593	45%	329231	182232	45%	190097	149832	21%
Cost3	1174150	628569	46%	739725	424932	43%	477969	353133	26%

Table 2: Initial and final TS solutions for different tests

- For a given cost configuration, MSH2 provides a better initial solution as T increases.
- For a given cost configuration, the objective function of the best solution found by TS decreases as T increases.
- TS optimizes low T value problems well.
- " The taboo optimization mechanism is stable with respect to the cost configurations.

All the above elements show TS efficiency, and also that the quality of the MSH2 solution is better when T increases.

4.4 Evolution of the size of the best TS solution

The solution size is the number of distinct modules in $\mathcal{M}'$ (which is the subset of modules used in the product bills of materials).

The solution size follows approximately the same evolution shape as the objective function value (Figure 6). In fact, many modules in the initial solution are generally used in a small bill of materials and greatly increase the fixed costs (of the assembly phase or of the production phase, or both). The TS

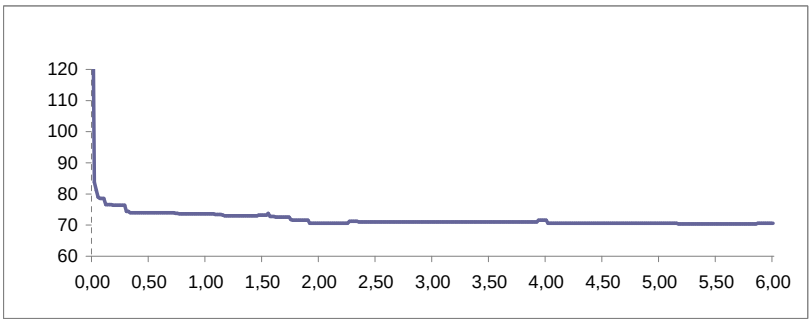

Figure 6: Evolution of the best solution size for the cost 3 problem with T =4

algorithm detects them very quickly through the first elimination neighborhood, and consequently the solution size decreases greatly at the first iterations.

We also note in the figure a little fluctuation in the evolution of the solution size, and can therefore find better solutions for larger sizes. This means that optimizing costs does not involve consistently reducing the solution size. By reducing the solution size, fixed costs are optimized and variable costs will increase (because a module will be used in many more bills of materials). So, optimizing the whole cost consists in finding a trade-off between fixed and variable costs.

4.5 Evolution of the mean degree of the best TS solutio

The solution mean degree is the mean degree of a module in $\mathcal{M}'$. It gives an idea about how many bills of materials the modules are used.

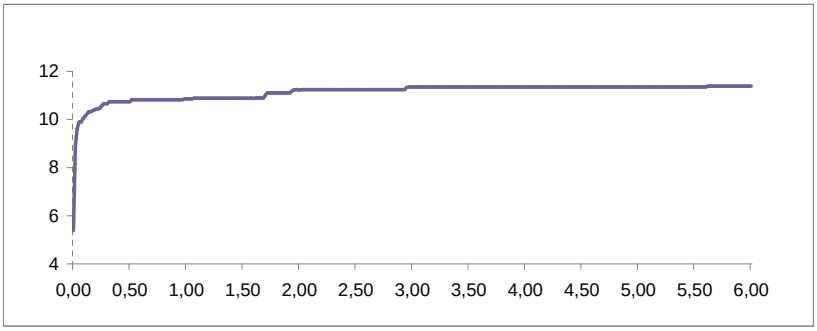

Figure 7: Evolution of the best solution mean degree for the cost 2 problem with T =3

Unlike the solution size, the evolution shape of the solution's mean degree is opposite to that of the objective function's evolution shape (Figure 7). In fact,

as the algorithm proceeds, the solution size decreases, leading to an increase in the number of module degrees (because modules will be used in more bills of materials), and so the solution's mean degree increases. We also note that, for a given cost configuration, the solution degree increases when T increases because it will be easier to find more shared modules.

5 Conclusion

This paper has been devoted to the difficult industrial problem that arises when companies try to offer a wide variety of products to consumers. The growing demand for personalized products makes fulfilling this need a matter of survival and prosperity for numerous firms. Customers become harder and harder to please in terms of the options and variants they choose. Consequently, product functionalities are increasing rapidly in size, and the need for decision support tools to determine bills of materials for large-scale problems is crucial. Moreover, an efficient choice of components (modules) has to be made. These modules are produced for stock, and used in the last stage, which is the assembly line. Several authors have considered this problem, using different assumptions - a function can appear twice in a final product, or one final product can be substituted for another containing more functions - but few papers consider the problem in which each final product must correspond exactly to the demand.

We have presented a TS algorithm with interesting neighborhoods. Tests on small instances and comparisons with optimal values showed the performance of our algorithm and encouraged us to develop more tests in order to improve its performance. In this paper, more complete tests are conducted. Large instances are generated, and the TS results were very efficient and initial solutions greatly improved.

These tests reveal that the TS algorithm always converges to small solutions. This means that large-degree modules are generally kept in the solution subset $\mathcal{M}'$, which leads to a reduction in the total number of modules used to assemble the finished products. We also conclude that the product assembly phase is influenced more by the algorithm, in terms of solution quality and computational time, than is the module assignment phase. Besides the fact that the assembly phase, which contains set partitioning constraints, is more difficult than the assignment phase both in size and complexity, analysis of the assignment details shows that, generally, a module is assigned to the most readily available lowest cost distant facility. We exploited the possibility of optimal resolution of the assignment phase to implement our algorithm.

There are several interesting future research areas for the product family and its related supply chain design problem. A Dantzig-Wolfe decomposition approach with a column generation would be an interesting direction. However, we think that it could be more expensive in terms of computational time and CPU resources. Moreover, it would not be able to handle problems as large as ours can - up to 20 functions per product - which is a very convenient capability for industrial applications.

We also suggest testing neighborhoods for the TS algorithm which are able to modify some product bills of materials for a given subset $\mathcal{M}'$. This would be an efficient way to optimize the assembly costs whenever the module subset $\mathcal{M}'$ is fixed.

Finally, it would be of interest to treat a problem with more than one assembly site, which would involve defining the costs of transportation from a distant facility to an assembly facility. Doing so would result in more complete modeling and a better understanding of the influence of transportation costs on supply chain design.

References

[1] Agard B., Cheung B., Da Cunha C. Composition of module stock for final assembly using enhanced genetic algorithm. *International Journal of Production Research*. doi:10.1080/00207540802161030.

[2] Agard B. and Penz B. A simulated annealing method based on a clustering approach to determine bills of materials for a large product family. *International Journal of Production Research*. 2009, 117(2), 389–401

[3] Akker v. d., Hoogeven H., Velde S. Combining column generation and Lagrangean relaxation to solve a single-machine common due date problem. *INFORMS Journal on Computing*. 2002, 14, 55–67.

[4] Briant O., Naddef D. The optimal diversity management problem. *Operation Research*. 2004, 52(4), 515–526.

[5] Bronmo G., Christiansen M., Fagerholt K., Nygreen B. A multi-start local search heuristic for ship scheduling - a computational study. *Computers & Operations Research*. 2007, 34, 900–917.

[6] Cachon G. Supply chain management: Design, coordination and operation. *Handbooks in Operations Research and Management Science*. 2003, 229–340.

[7] Da Cunha C., Agard B. Composition of modules' stock using Simulated Annealing. *IEEE International Symposium on Assembly and Task Planning-ISATP*. Montreal, Canada, July 19-21 2005.

[8] El Hadj Khalaf R., Agard B., Penz B. Greedy heuristics for determining a product familly bill of materials. *38th International Conference on Computers and Industrial Engineering*, Beijing, China, Oct 31- Nov 2, 2008.

[9] El Hadj Khalaf R., Agard B., Penz B. An experimental study of a product and supply chain design problem for mass customization. *Journal of Intelligent Manufacturing* , 2009. DOI : 10.1007/s10845-009-0247-0.

[10] Friese P., Rambau J. Online-optimization of multi-elevator transport systems with reoptimization algorithms based on set-partitioning models. *Discrete Applied Mathematics*. 2006, 154, 1908–1931.

[11] M.R. Garey and D.S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[12] Glover F., McMillan C., Novick B. Interactive Decision Software and Computer Graphics for Architectural and Space Planning. *Annals of Operations Research*, 1985, 5, 557–573.

[13] Glover F. Tabu Search - PartI. *ORSA Journal on Computing*, 1989, 1(3), 190–206.

[14] Glover F. Tabu Search - PartII. *ORSA Journal on Computing*, 1990, 2(1), 4–32.

[15] Hanczar P. Solving the vehicle routing problem based on branch and bound method. *Badania Operacyjne i Decyzje*. 2002, 3-4(1), 37–51.

[16] Kotecha K., Shanghani G., Gambhava N. Genetic algorithm for airline crew scheduling problem using cost-based uniform crossover. *Proceedings Lecture Notes in Computer Science*. 2004, 3285, 84–91.

[17] Lamothe J., Hadj-Hamou K., and Aldanondo M. An optimization model for selecting a product family and designing its supply chain. *European Journal of Operational Research*, 169(1):1030–1047, 2006.

[18] Lee YH, Kim JI, Kang KH, Kim KH A heuristic for vehicle fleet mix problem using tabu search and set partitioning. *Journal of the Operational Research Society*, June 2008,59, 833–41.

[19] Montreuil B., Poulin M. Demand and supply network design scope for personalized manufacturing. *Production Planning & Control*, July 2005, 16(5), 454–469.

[20] Poulin M., Montreuil B., Martel A. Implications of personalization offers on demand and supply network design: A case from the golf club industry. *European Journal of Operational Research*, 2006, 169, 996–1009.

[21] Rai R., Allada V. Modular product family design: agent-based Pareto-optimization and quality loss function-based post-optimal analysis. *International Journal of Production Research*, 2003, 41(17), 4075–4098.

[22] Ronnqvist M., Eveborn P., Flisberg P. Laps Care-an operational system for staff planning of home care. *European Journal of Operational Research*. June 16 2006, 171(3), 962–976.

[23] Starr M. Modular production - a new concept. *Hatvard business review*. nov.-dec. 1965, 131–142.

[24] Wu-Zhen d., Xiang-Sheng j., Zeng-De S. A multiple dimension set partitioning load balancing resource optimization allocation algorithm. *Journal of Computer Applications*. May 2007, 27(5), 1208–1213.

[25] Yang Y., Wang J.X., Sang S.J. Research on product family design for mass customization. *Modular Machine Tool and Automatic Manufacturing Technique*, 2006,10, 9–13.