International Journal of Production Research

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

Simultaneous design of a product family and its related supply chain using a Tabu Search algorithm

Radwan El Hadj Khalaf [a], Bruno Agard [b] & Bernard Penz [a]

[a] G-SCOP, Université de Grenoble, 46 avenue Félix-Viallet 38031 Grenoble Cedex 1, France

[b] CIRRELT, Département de Mathématiques et Génie Industriel, École Polytechnique de Montréal, C.P. 6079, succ. Centre-Ville, Montréal, Québec H3C 3A7, Canada

Available online: 13 Dec 2010

To cite this article: Radwan El Hadj Khalaf, Bruno Agard & Bernard Penz (2011): Simultaneous design of a product family and its related supply chain using a Tabu Search algorithm, International Journal of Production Research, 49:19, 5637-5656

To link to this article: http://dx.doi.org/10.1080/00207543.2010.519737

International Journal of Production Research
Vol. 49, No. 19, 1 October 2011, 5637–5656

Taylor & Francis
Taylor & Francis Group

RESEARCH ARTICLE

Simultaneous design of a product family and its related supply chain using a Tabu Search algorithm

Radwan El Hadj Khalaf[a]*, Bruno Agard[b] and Bernard Penz[a]

[a]*G-SCOP, Université de Grenoble, 46 avenue Félix-Viallet 38031 Grenoble Cedex 1, France;* [b]*CIRRELT, Département de Mathématiques et Génie Industriel, École Polytechnique de Montréal, C.P. 6079, succ. Centre-Ville, Montréal, Québec H3C 3A7, Canada*

(*Received 10 December 2009; final version received 4 August 2010*)

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 synchronised 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 formulations to represent the problem, and solve it by adapting a Tabu Search algorithm in which the assembly process and the supply chain design are considered at the same time.

Keywords: product platform; product development; product design; Tabu search; optimisation; supply chain management

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 standardise 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 (Starr 1965).

Lean production has brought about major changes in the supplier–retailer relationship. Today, lead time is a major issue in contract negotiations (Cachon 2003). For the supplier, it is essential to fill an order within a specific time period, although delays are costly for

*Corresponding author. Email: radwan.ed-hadj-khalaf@scop.inpg.fr

ISSN 0020–7543 print/ISSN 1366–588X online

DOI: 10.1080/00207543.2010.519737
http://www.informaworld.com

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, such as product family design, are being developed. Product family design must take into account not only product diversity, but also the definition of the process and the supply chain (Yang *et al*. 2007). A consistent approach is needed in order to guarantee customer satisfaction, as well as to minimise the total investment on the part of producers in the product and in the operating costs incurred by the global supply chain. Rai and Allada (2003) 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 optimisation method using an agent-based framework to determine the Pareto design solutions for a given set of modules. In the second step, a post-optimisation analysis is performed to determine the optimal platform level. Montreuil and Poulin (2005) present the various types of personalised manufacturing processes in a mass customisation context to offer innovative, highly personalised products with short and reliable delivery times. He characterises complementary types of personalisation and highlights the key elements required for their successful implementation. Finally, he shows how personalisation affects the design of the supply and demand network. Poulin *et al*. (2006) present a framework comprising eight personalisation options which can be combined to form a complete personalised offering. Then, using the analogy of golf clubs (irons) for illustration purposes, they contrast the impact on that network.

Delayed differentiation or postponement is a widely used concept in product families. The manufacturing process starts by making a generic product structure that is later differentiated into specific finished products. Feitzinger and Lee (1997) explain that the key of mass-customising is postponing the task of differentiating a product for a specific customer until the latest possible point in the supply network. They identify three organisational-design principles which form the basic building blocks of an effective mass-customisation program. They expose the case of the Hewlett-Packard Company. Garg and Tang (1997) develop two models to study products with more than one point of differentiation. In each model, they examine the benefits of delayed differentiation at each of these points, and derive the necessary conditions when one type of delayed differentiation is more beneficial than the other. Their analysis indicates that demand variabilities, correlations and the relative magnitudes of the lead times play an important role in determining which point of differentiation should be delayed. Yadav *et al*. (2008) formulate a multi-objective problem to select a product family and design its supply chain in order to maintain product differentiation and help trade-off the cost and price premium drawing capability. They use an Interactive Particle Swarm Optimisation (IPSO) approach. A case study for a wiring harness supplier of an Automated Guided Vehicle manufacturer is considered and IPSO is implemented to solve it.

Global design modelling has been studied recently by a number of authors. Agard *et al*. (2009) propose a genetic algorithm to minimise the mean assembly time of a finished product for a given demand, and Agard and Penz (2009) propose a model for minimising module production costs and a solution based on simulated annealing. Da Cunha and Agard (2005) also propose a simulated annealing algorithm to determine the composition of the stock modules of a given size to minimise 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*. (2006) 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 and Naddef (2004) 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 (Garey and Johnson 1979).

The originality of this work consists in the consideration of the whole supply chain in the product family design. Some studies have already treated this problem for a single product case, but few have considered a product family based on a modularity design.

A detailed description of the problem is provided in the following section. The notation is explained in Subsection 1, and then a Mixed Integer Linear Program model is given in Subsection 2.2. The Tabu Search (TS) algorithm is then presented in Section 3. Computational experiments are given and analysed 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 by El Hadj Khalaf *et al.* (2009a). A producer receives customer orders for finished products containing options and variants. Each individual product is then manufactured from modules provided by various suppliers.

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.

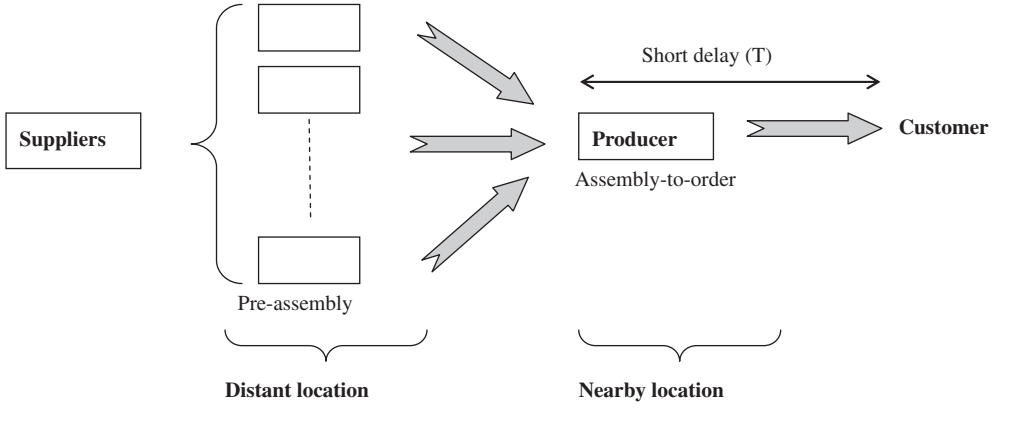

Figure 1. Structure of the supply chain.

To satisfy customer orders, the producer brings in pre-assembled components, called modules, from many suppliers located at facilities around the world. We will call these facilities the distant location facilities. Generally, these facilities are located at a distant location from the nearby facility where they take advantage of smaller production costs; this also implies some transportation costs to bring modules from a distant location facility to the nearby location facility. The modules are then assembled at the nearby location facility, which, we assume, is close to the customers, and thus is characterised by a rapid reaction time and a short lead time. The modules are manufactured in advance, from provisions, and then delivered to the nearby facility for stock. When designing the product family and supply chain we do not consider transportation delays of modules, which should be included at the operational level.

This problem occurs in the car industry, and more specifically for the manufacturing of electrical beams. The electrical beam is one of the first parts assembled in a car, just after the vehicle has entered the assembly line. The time needed to build the requested beam from scratch is longer than the time constraint. The electrical beam must then be built from subparts called modules. The challenge for these companies is to decide on the set of modules that have to be produced in order to respect the time delay for all possible orders. As the modules are made to stock, the strategy used is to produce the modules in low cost countries and transport them to nearby assembly facilities (Lamothe *et al.* 2006).

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 minimise production and transportation costs. The model considered here is proposed to take strategic decisions (modules selected for the bill of materials and production sites) for a long-term period, and not to solve planning issues. Consequently, all the costs are estimations of the total costs during the time horizon of the application of the decisions. For example, the transportation costs are estimated means considering that the transportation policy is fixed. At this level, tactical concerns such as transportation or storage capacity, inventory management, and lead time, for example, are not taken into account.

In this study, we do not consider the problem of the interface between modules. A function is simply present or not in a module. Due to this assumption, we avoid the problem of an interface between modules. We also consider that the platform is managed before the definition of the module set, i.e. the common part of the products of the family is decided previously.

2.1 *Notation*

A product (or a module) is considered as the set of functions that it contains. It is currently modelled with a binary vector in which 1 indicates 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.

Table 1. Notation.

$\mathcal{F} = \{F_1, \ldots, F_q\}$	Set of q functions that can appear in both finished products and modules
$\mathcal{P} = \{P_1, \ldots, P_n\}$	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)
$\mathcal{M} = \{M_1, \ldots, M_m\}$	Set of m possible modules
$\mathcal{S} = \{S_1, \ldots, S_s\}$	Set of s distant production facilities where site S_l has production capacity W_l
F_j^A, V_j^A	Two components of the *Assembly Cost* of module M_j at the nearby facility
F_j^A	Fixed cost of module M_j at the nearby facility (management costs)
V_j^A	Variable cost of module M_j at the nearby facility (cost of assembly, storage, etc.)
F_j^P, V_j^P	Two components of the *Supply Cost* of module M_j at distant facility S_l
F_{jl}^P	Fixed cost of module M_j at distant facility S_l (management)
V_{jl}^P	Variable cost of module M_j at distant facility S_l (cost of assembly, storage, transportation, etc.)
t_j	Time required to assemble module M_j in a finished product
T	Maximum assembly time available
W_{jl}	Workload generated by producing one module M_j at facility S_l
W_l	Workload capacity available at facility S_l

Let us introduce the notation given in Table 1. 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. Even when $\mathcal{M}$ does not contain the whole combinatory of modules, the problem is NP-hard because it contains set partitioning constraints. For our tests we use all possible modules to guarantee problem feasibility.

In terms of the manufacturing process: (1) the producer assembly line costs must be minimised; 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 shows two different ways to assemble product P_1 from two different modules. The first strategy is to build product P_1 from modules M_1 and M_2. The second is to build product P_1 from modules M_3 and M_4. Those modules are different because they do not contain the same set of functions. Then the problem is to determine, for each product, the modules to use in bills of materials in order to minimise globally the assembly and production costs of the supply chain.

2.2 *Mathematical modelling*

The problem is modelled using a Mixed Integer Linear Program formulation. The objective is to minimise all the costs linked to the activities of the producer and the suppliers.

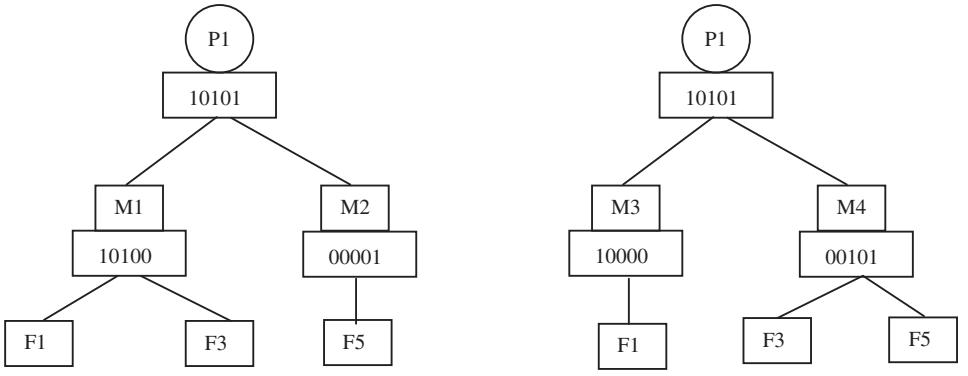

Figure 2. Alternative bills of materials.

The objective of the model proposed in this section is to determine the optimal bills of materials that minimise the assembly and supplying costs (fixed and variable) at the same time:

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

s.t.

$$AX_i = P_i^T, \quad \forall i \in \{1, \ldots, n\}, \tag{1}$$

$$\sum_{j=1}^{m} t_j X_{ij} \leq T, \quad \forall i \in \{1, \ldots, n\}, \tag{2}$$

$$X_{ij} \leq Y_j, \quad \forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, m\}, \tag{3}$$

$$\sum_{l=1}^{s} Q_{jl} = \sum_{i=1}^{n} D_i X_{ij}, \quad \forall j \in \{1, \ldots, m\}, \tag{4}$$

$$\sum_{j=1}^{m} W_{jl} Q_{jl} \leq W_l, \quad \forall l \in \{1, \ldots, s\}, \tag{5}$$

$$Q_{jl} \leq B Y_{jl}, \quad \forall j \in \{1, \ldots, m\}, \forall l \in \{1, \ldots, s\}, \tag{6}$$

$$Y_j, X_{ij} \in \{0, 1\}, \quad \forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, m\}, \tag{7}$$

$$Q_{jl} \geq 0 \text{ integer}, \quad \forall j \in \{1, \ldots, m\}, \forall l \in \{1, \ldots, s\}, \tag{8}$$

$$Y_{jl} \in \{0, 1\}, \quad \forall j \in \{1, \ldots, m\}, \forall l \in \{1, \ldots, s\}, \tag{9}$$

where $X_{ij}=1$ if module M_j is used in the bill of materials of product P_i and 0 otherwise, $Y_j=1$ if module M_j is selected (then M_j belongs to $\mathcal{M}'$, the set of selected modules) and 0 otherwise, A is the binary matrix, column j of which is the vector M_j, X_i is the column vector composed of the variables X_{ij} (P_i^T is the column vector representing the product P_i), $Y_{jl}=1$ if module M_j is produced at facility S_l and 0 otherwise, and Q_{jl} is the quantity of module M_j produced at facility S_l.

The objective function contains two major components,

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

which represents the costs incurred at the nearby facility, where $(\sum_{i=1}^{n} D_i X_{ij})$ 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 (1) shows that a finished product P_i must be assembled exactly according to customer requirements. Constraint (2) indicates that products must be assembled within the time window T, in order to respect the delivery time. Constraint (3) states that if module M_j is used in the bill of materials of product P_i, then module M_j must be produced. Constraint (4) indicates that the production of module M_j must satisfy the overall quantities required. Constraint (5) shows that the total production at facility S_l must not exceed that facility's capacity. Constraint (6) expresses the relation between the variables Q_{jl} and Y_{jl} (B is a large constant). 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 (Garey and Johnson 1979). We conclude that it is NP-hard in the strong sense. Even if we fix the bills of materials, the logistical problem is still NP-hard in the strong sense. This can be easily proved by a reduction from the 3-Partition problem (Garey and Johnson 1979). Consequently, the design of product families using the concept of modularity is a difficult optimisation problem with or without the logistical part. For this, the following section explores the use of a TS algorithm to resolve large instances.

3. Resolution with a Tabu Search algorithm

The Tabu Search (TS) is a metaheuristic approach designed to find a near-optimal solution of combinatorial optimisation problems (Glover *et al.* 1985, Glover 1989, 1990). 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 neighbourhood $N(x) \subseteq F$. The TS starts from an initial solution. At each step the neighbourhood $N(x)$ of a given solution x is searched in order to find a neighbour x'. The move which leads to the neighbour 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 Tabu list, of length L (fixed or variable), is introduced to prevent returning to a solution visited

in the last L iterations. The Tabu 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.* (2008), for example, propose an efficient heuristic using TS and by solving set-partitioning problems to determine 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 the new feasible solution is found. These moves are described in the elimination and insertion neighbourhoods. Before introducing a 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 for $\mathcal{M}'$ if it does not admit a bill of materials from modules in $\mathcal{M}'$.
- The supplying costs of a module are costs (fixed and variable) generated by the production (and transportation) of the module at the distant facilities.

3.1 *Tabu Search algorithm*

The algorithm (Figure 3) begins with an initialisation 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 (1), (2), (3) and (7). 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. 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 (4), (5), (6), (8) and (9). 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. In order to ensure the feasibility of this step regarding the capacity constraints, a fictive distant facility l^* is considered for which the workload W_{jl^*} of each module M_j is null while its supply costs are very high. This facility allows us to avoid the infeasibility of the modules' assignment step and also helps to balance the non-fictive facilities' capacities whenever the manufacturing capacities are insufficient.

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. In order to avoid a cycling search, the elimination (or insertion) of modules must take into

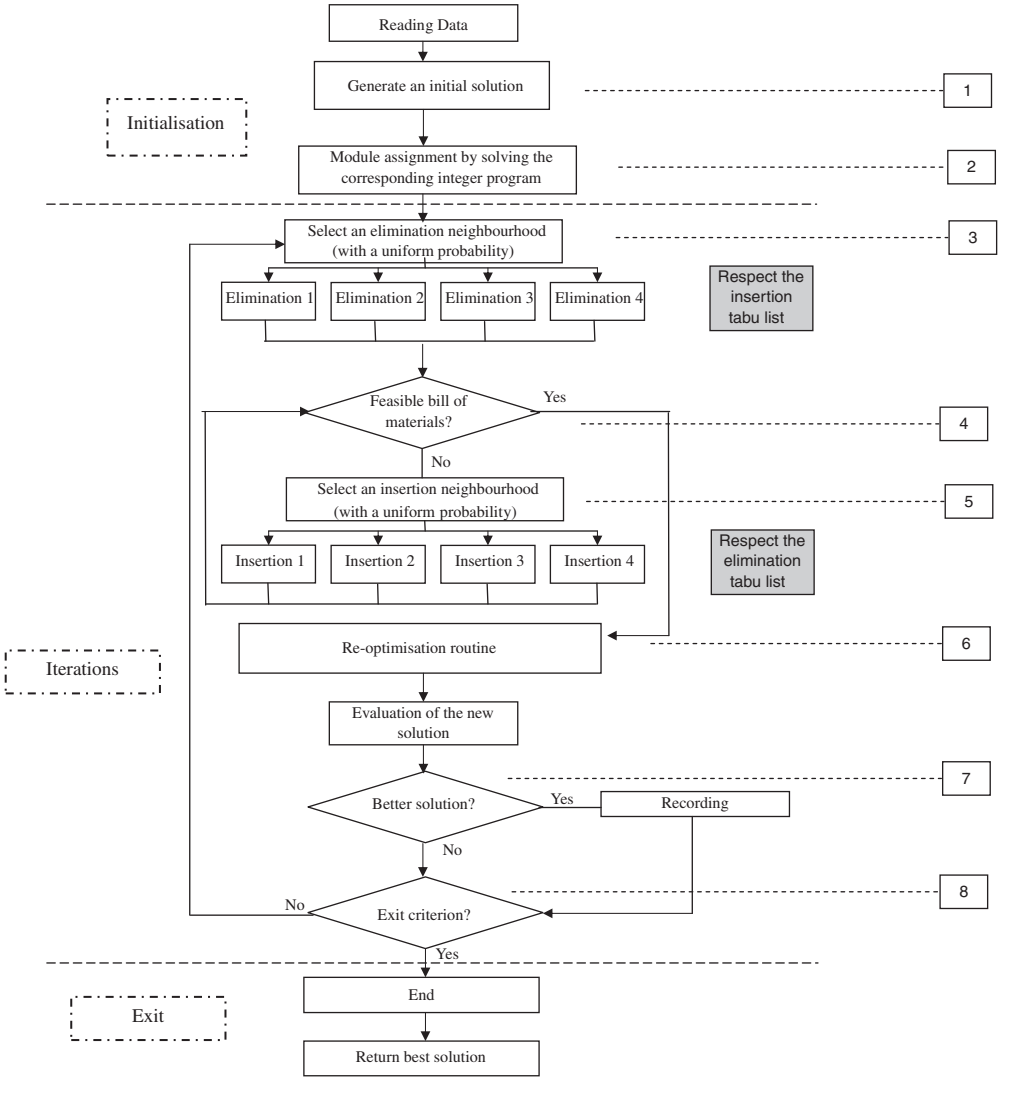

Figure 3. The Tabu Search algorithm.

consideration the Tabu lists which means that a module should not be eliminated (or inserted) if it appears in the insertion (elimination) Tabu list. If a module has recently been eliminated (or inserted) in the l previous iterations, then it should not be inserted (or eliminated) in the current iteration.

Once the new subset is constructed and new bills of materials for infeasible products are determined, the bill of materials of finished products can be checked and updated in Step 4. During this phase the set of modules could be infeasible and, consequently, new modules are added in order to converge to a feasible set. The new solution is evaluated and recorded if it is better than the current best solution (Step 7). This process is iterated 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 Tabu 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 neighbourhoods used in the algorithm.

3.2 *Bill of materials feasibility control (Step 4)*

After elimination (insertion) of a module from (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 checks their bill of material feasibility by solving the Integer Linear Program formed by constraints (1), (2), (3) and (7). 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 infeasible product and its feasibility will be checked again in the next iteration.

When infeasible products appear, the set of modules must be updated in order to obtain a feasible set. To do that, insertion neighbourhoods are used (see Section 3.5.) until feasibility is attained.

3.3 *Re-optimisation 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 (4), (5), (6), (8) and (9). The objective function here is formed only from the supplying costs Z^P.

3.4 *Elimination neighbourhoods (Step 3)*

Four elimination neighbourhoods 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 supplying costs. As with the previous move, the supplying costs of modules in $\mathcal{M}'$ are calculated and one of them, with high supplying costs, is randomly selected.
- *Elimination 3*: Eliminates a large-degree module (because it will certainly generate high variable costs).
- *Elimination 4*: Random elimination.

We insist here on the probabilistic aspect of each elimination neighbourhood. For example, in 'elimination 1', a procedure is used to sort modules by their degree values, then a bounded number k is randomly drawn and finally the module having the kth smaller degree value is eliminated.

3.5 *Insertion neighbourhoods (Step 5)*

Four insertion neighbourhoods are considered.

- *Insertion 1*: Inserts a module that generates low supplying 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 bill of materials feasibility for a non-feasible product. This can be done by taking one non-feasible finished product P_i and solving the ILP formed by constraints (1), (2), (3), and (7), and for which the objective function is the minimisation of the sum of X_{ij} variables of modules that do not belong to $\mathcal{M}'$ (min $\sum_{j\in\mathcal{M}\setminus M'} X_{ij}$).

As explained in the previous section, the probabilistic aspect is also present in the above insertion neighbourhoods. The combination of the probabilistic aspect in both elimination and insertion neighbourhoods gives the Tabu algorithm the desired ability to explore the entire solution space.

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 analyse the solution quality and algorithm efficiency for the generated instances. In previous work, El Hadj Khalaf *et al.* (2009b) compared the algorithm efficiency with the optimal solution for small instances. The Tabu results were very close to optimal with a mean gap of 4.85% and a gap of 9.82% in the worst case. In this study we experiment on larger instances. For this, five instances were randomly generated for which the module set, the finished product set, the distant facility set, the demand D_i, the assembly operating times t_j, the distant facility capacities, the module production loads, the distant facility costs, and the nearby facility costs are fixed.

The problem data were fixed as follows: $q = 15$ (the number of functions in a product or a module), $n = 500$ (the number of finished products that have to be assembled), where each product has at least five functions (Min f) and at most 10 (Max f), which represents the number of non-zero functions, $m = 30{,}826$ (all modules compatible with the finished products), $s = 4$ (the number of distant production facilities), and for each site the manufacturing costs and the production load of each module are randomly generated as soon as the site production capacity. The assembly operating times t_j were fixed to 1, and T was varied from 3 to 5. The method works for all t_j and T but we set these parameters to integer values in order to simplify the understanding of the results.

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.

Three cost scenarios were randomly generated. For the 'Cost 1' problem, the assembly costs are greater than the supplying costs. For the 'Cost 2' problem, the two costs are almost equivalent. For the 'Cost 3' problem, the supplying costs are the highest.

Table 2 shows the different cost values (of the Tabu final solution) for the cost scenarios with $T=4$ in order to give an idea of the ratio of these values.

The tests were carried out in C++ with the Ilog Cplex9.0 library. They were solved on a DELL station/2.8 GHz/2.5 Go RAM. The computational time was fixed to six hours for the TS algorithm.

4.1 *Influence of the initial solution*

The objective of this section is to analyse 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 efficiency by starting with three different initial solutions: a randomly generated solution, the MSH1 solution, and the MSH2 solution (described in the following).

4.1.1 *A randomly generated solution*

This initial solution can be obtained by freezing the index i and solving n ILPs formed by constraints (1), (2), (3) and (7). The objective function could, for example, be the maximisation 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 (4), (5), (6), (8) and (9). The objective function remains, of course, the minimisation of the supplying costs Z^P.

4.1.2 *MSH1*

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

Table 2. Cost scenarios.

	Cost scenario		
	C1	C2	C3
Instance no.	1	3	5
Total assembly costs	135,580	92,368	89,868
Total production costs	10,999	89,863	335,063

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 $i \leftrightarrow j$ indicates that we have to sum the 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 by El Hadj Khalaf *et al.* (2008). As with the previous method, we then solve the linear subprogram to assign the resulting modules.

4.1.3 *MSH2*

With this heuristic, we improve the module selection criterion so as to take into account the supplying costs, and at the same time proceed with module selection and assignment of the module to the appropriate distant facility. Therefore, $ind2_j = ind1_j + C_j^P$, 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 shows that the MSH2 heuristic gives the best initial solution and the best final solution (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.

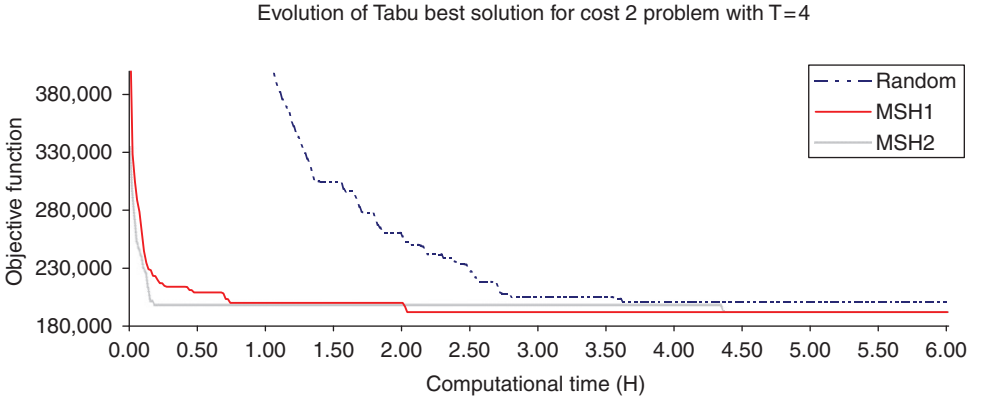

Figure 4. Evolution of the best TS solution taking into account the various starting points for the cost 2 problem.

Table 3. Standard TS solution compared with TS solutions after freezing elimination and insertion neighbourhoods for the instance 1 and cost 2 problem.

	Solution of the standard Tabu algorithm: 192,573			
	Ins1	Ins2	Ins3	Ins4
Elm1	192,857	196,415	205,180	183,822
Elm2	209,770	200,848	199,353	199,426
Elm3	220,083	221,880	210,907	249,036
Elm4	233,244	225,745	234,116	227,718

4.2 *Influence of neighbourhoods*

In this section, we analyse the TS efficiency when we freeze the elimination and insertion neighbourhoods. For this, we compare the solution of the standard algorithm (with a uniform random selection of elimination and insertion neighbourhoods) with the solution of each pair of neighbourhoods (elimination i, insertion j). Table 3 shows that, in all 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.

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 neighbourhood is frozen at elimination 1 (elimination of a module of small degree), and the insertion neighbourhood 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 presents 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 already been 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 Tabu 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 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 neighbourhoods used.

Table 4 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 the following.

- For a given cost configuration, MSH2 provides a better initial solution as T increases.

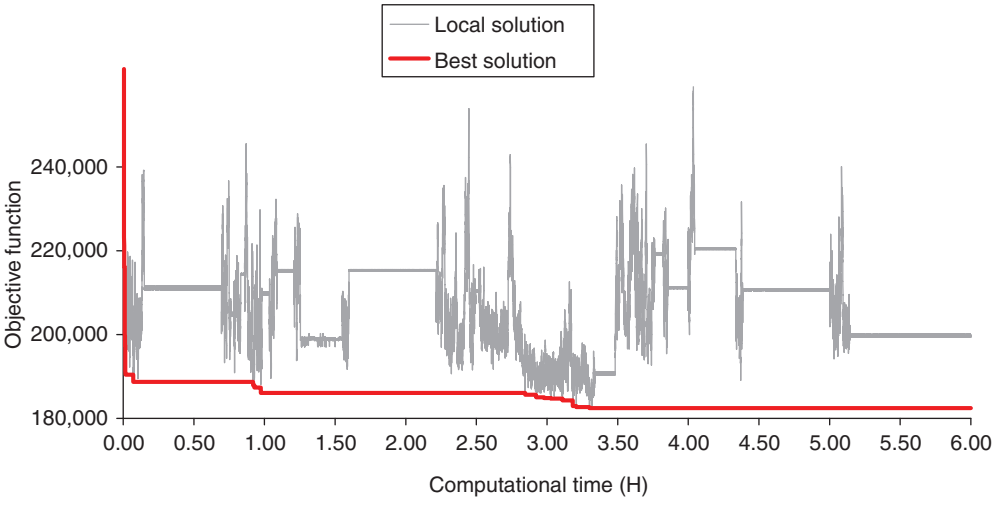

Figure 5. The objective function evolution of the local and best TS solutions.

Table 4. Initial and final TS solutions for different tests.

	$T=3$			$T=4$			$T=5$		
Cost configuration	Initial solution	Final solution	Gap (%)	Initial solution	Final solution	Gap (%)	Initial solution	Final solution	Gap (%)
Cost 1	546,574	268,974	51	334,476	146,580	56	140,475	101,285	28
Cost 2	503,874	278,593	45	329,231	182,232	45	190,097	149,832	21
Cost 3	1,174,150	628,569	46	739,725	424,932	43	477,969	353,133	26

- For a given cost configuration, the objective function of the best solution found by TS decreases as T increases.
- TS optimises low-T-value problems well.
- The Tabu optimisation mechanism is stable with respect to the cost configurations.

All the above elements show the efficiency of the TS 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 algorithm

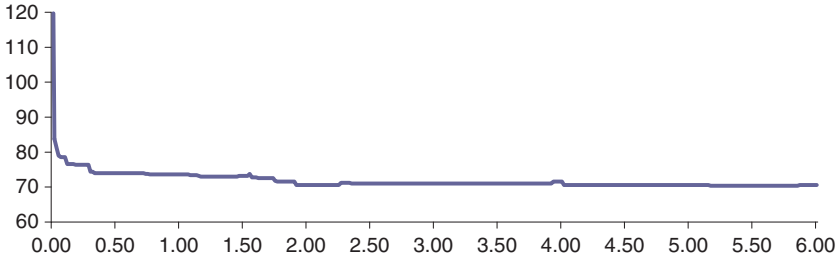

Figure 6. Evolution of the best solution size for the cost 3 problem with $T=4$.

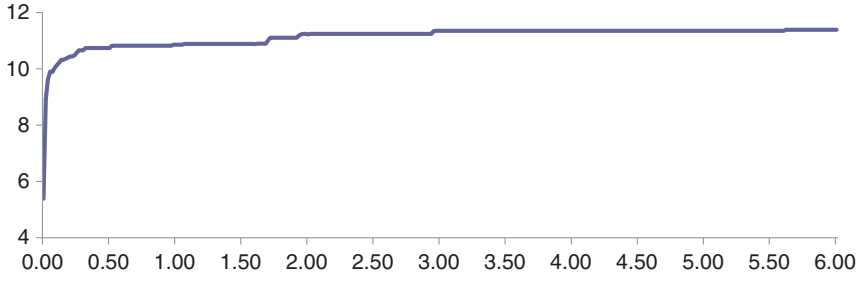

Figure 7. Evolution of the best solution mean degree for the cost 2 problem with $T=3$.

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

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

4.5 *Evolution of the mean degree of the best TS solution*

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

4.6 *Tests on large instances*

The aim of this section is to verify the efficiency of the Tabu algorithm on larger instances. For this, two large instances are generated for which the total number of functions in a

Table 5. Parameters of the generated instances.

Instance size	q	18	21
No. of finished products	n	700	1000
Min f		8	10
Max f		14	16
No. of modules	m	31,179	27,895
No. of distant facilities	s	5	6
T		4	5

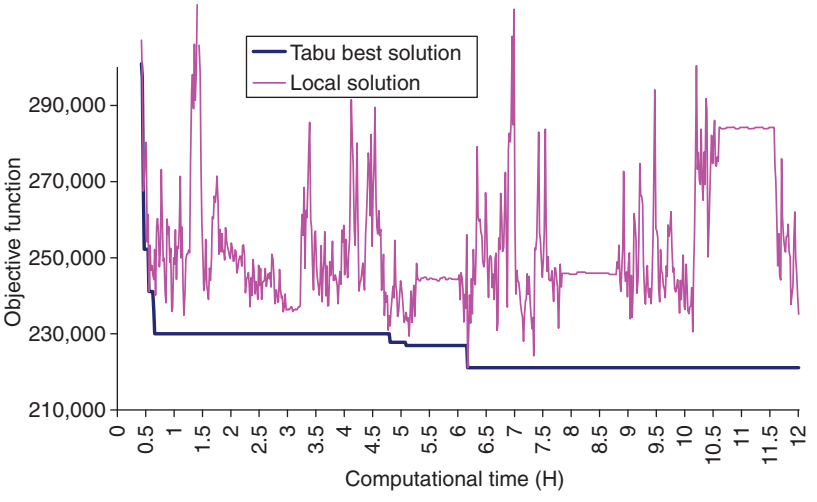

Figure 8. Evolution of the Tabu best solution for instance 18.

product or a module is, respectively, 18 and 21. Table 5 summarises the instance parameters. We remember here that Min f and Max f are respectively the minimum and the maximum number of functions that a finished product contains in our instances. We also note that the number of modules generated in the initial set $\mathcal{M}$ is very small compared with the whole possible combinatory, so a limited initial set of modules $\mathcal{M}$ is used in order to avoid memory problems when running the algorithm. From a practical point of view, $\mathcal{M}$ is frequently smaller than the complete combination of all possible functions due to technical (or commercial) constraints, for example if function A is present (or absent), then function B is present (or absent). This is supported in our modelling by simply removing non-supported modules (non-admissible combinations of functions) from the set $\mathcal{M}$.

Figures 8 and 9 show the evolution of the Tabu best solution (as well as the local one) with the computational time (which is fixed at 12 hours) for instances 18 and 21. We clearly see that the curves respect perfectly the same appearance as the curve of Figure 5. We can conclude that the Tabu algorithm is able to run easily on larger instances, with the unique constraint of respecting the memory capacity. This constraint can be respected by controlling the size of the initial set of modules $\mathcal{M}$. Furthermore, we remark that the method converges rapidly (in one hour approximately). The method then finds better solutions, but the improvements are low in percentage terms.

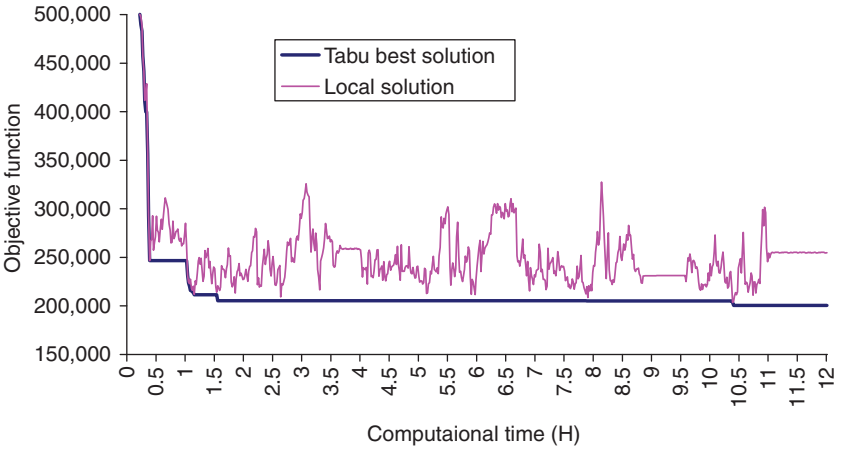

Figure 9. Evolution of the Tabu best solution for instance 21.

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 personalised 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.

Tests on small instances and comparisons with optimal values have demonstrated 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 column generation would be an interesting direction. However, we believe 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 neighbourhoods 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 optimise 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 modelling and a better understanding of the influence of transportation costs on supply chain design.

References

Agard, B., da Cunha, C., and Cheung, B., 2009. Composition of module stock for final assembly using an enhanced genetic algorithm. *International Journal of Production Research*, 47 (20), 5829–5842.

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

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

Cachon, G., 2003. Supply chain management with contracts. In: *Handbooks in operations research and management science: supply chain management: design, coordination and operation*. Elsevier. 229–340.

Da Cunha, C. and Agard, B., 2005. Composition of modules' stock using simulated an-nealing. *In: IEEE international symposium on assembly and task planning – ISATP*, 19–21 July 2005, Montreal, Canada.

El Hadj Khalaf, R., Agard, B., and Penz, B., 2008. Greedy heuristics for determining a product familly bill of materials. *In: 38th international conference on computers and industrial engineering*, 31 October–2 November 2008, Beijing, China.

El Hadj Khalaf, R., Agard, B., and Penz, B., 2009a. An experimental study for the selection of modules and facilities in a mass customization context. *Journal of Intelligent Manufacturing*, DOI 10.1007/s10845-009-0247-0.

El Hadj Khalaf, R., Agard, B., and Penz, B., 2009b. Product and supply chain design using a taboo search. *In: International conference on industrial engineering and systems management IESM*, 13–15 May 2009, Montreal, Canada.

Feitzinger, E. and Lee, H.L., 1997. Mass customization at Hewlett-Packard: the power of postponement. *Harvard Business Review*, 75 (1), 116–121.

Garey, M.R. and Johnson, D.S., 1979. *Computers and intractability, a guide to the theory of NP-completeness*. New York: W.H. Freeman.

Garg, A. and Tang, C.S., 1997. On postponement strategies for product families with multiple points of differentiation. *IIE Transactions*, 29 (8), 641–650.

Glover, F., 1989. Tabu search — Part I. *ORSA Journal on Computing*, 1 (3), 190–206.

Glover, F., 1990. Tabu search – Part II. *ORSA Journal on Computing*, 2 (1), 4–32.

Glover, F., McMillan, C., and Novick, B., 1985. Interactive decision software and computer graphics for architectural and space planning. *Annals of Operations Research*, 5 (3), 557–573.

Lamothe, J., Hadj-Hamou, K., and Aldanondo, A., 2006. An optimization model for selecting a product family and designing its supply chain. *European Journal of Operational Research*, 169 (3), 1030–1047.

Lee, Y.H., *et al*., 2008. A heuristic for vehicle fleet mix problem using tabu search and set partitioning. *Journal of the Operational Research Society*, 59 (6), 833–841.

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

Poulin, M., Montreuil, B., and Martel, A., 2006. Implications of personalization offers on demand and supply network design: a case from the golf club industry. *European Journal of Operational Research*, 169 (3), 996–1009.

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

Starr, M., 1965. Modular production – A new concept. *Harvard Business Review*, 131–142.

Yadav, S.R., *et al*., 2008. An interactive particle swarm optimization for selecting a product family and designing its supply chain. *International Journal of Computer Applications in Technology*, 31 (3/4), 168–186.

Yang, Y., Wang, J.W., and Sang, S.J., 2007. Research on product family design for mass customization. *Modular Machine Tool and Automatic Manufacturing Technique*, 10, 9–13.