38th International Conference on Computers and Industrial Engineering - 38CIE, Beijing, China, Oct. 31 - Nov. 2, 2008.

GREEDY HEURISTICS FOR DETERMINING A PRODUCT FAMILY BILL OF MATERIALS

Radwan El Hadj Khalaf
G-SCOP, Grenoble INP -- CNRS -- UJF
France
radwan.el-hadj-khalaf@g-scop.inpg.fr

Bruno Agard
CIRRELT, Département de Mathématiques et Génie Industriel
Ecole Polytechnique de Montréal
Canada
bruno.agard@polymtl.ca

Bernard Penz
G-SCOP, Grenoble INP -- CNRS -- UJF
France
bernard.penz@g-scop.inpg.fr

ABSTRACT

When designing a new product family, designers and manufacturers have to define both the product structure and its supply chain simultaneously. This leads to a complex optimization problem to solve in order to satisfy diversified customer requirements involving various options and variants. Furthermore, we must know the bill of materials for each product to evaluate production and transportation costs, which means that rapid methods for building bills of material are crucial. Our paper addresses this problem, which consists in selecting a set of modules to be manufactured at distant facilities and shipped to a nearby plant for a final assembly operation within a specified amount of time. The objective is a set of modules capable of defining the bill of materials for each finished product which will minimize production costs. Two fast heuristics are proposed to solve it. Experiments on small instances enable us to compare our results with optimal solutions, and experiments on larger instances enable us to compare the performance of the two heuristics.

KEYWORDS

Product family design, bill of materials, integer programming, heuristic

1. INTRODUCTION

Nowadays, the growing demand for customized products involves an increasing number of product variants and options, and, consequently, a complex product diversity to manage. This variety must be controlled in terms of product, process, and supply chain costs, as well as customer lead time. Moreover, when designing a new product family, a consistent approach is necessary in order to quickly define a set of variants and the relevant supply chain with a view to ensuring customer satisfaction and minimizing the total investment and operating costs of the global supply chain (Lamothe, J. et al., 2006).

A product family is composed of similar products which differ in some characteristics, such as options. For example, a basic car model may offer only a few options in order to minimize the base price. Options can then be added to this basic model, like air-conditioning, an automatic gear box, a diesel engine, and so on.

There are two extreme production strategies that a company can use to achieve this. The first is to stock all the various products, which involves selecting a minimum set of standardized products (Briant, O., Naddef, D., 2004), which could include supplementary options to meet diverse customer requirements. However, a product may contain options which are not required by the customer. The second strategy is to produce only when an order is received. In this case, the lead time may be higher, resulting in a failure to satisfy the customer. An intermediate strategy would be to manufacture pre-assembly components, called modules, for stock and then assemble them when an order is received. The advantage of such a strategy is to reduce the lead time and to permit broad product diversity for customers at limited cost to the company.

In this paper, we explore a production policy in which modules are manufactured at distant facilities to minimize costs. Those modules are then shipped and assembled at a nearby facility in order to ensure a short lead time for the customer. We present two heuristic strategies to define the bill of materials of each finished product: (1) look at the finished product set and determine the most suitable bill of materials for each product; and (2) look at the module set and select the components of most interest, inserting them into the bill of materials of compatible finished products.

This work will be helpful for exploring further research on the module assignment problem. In fact, rapidly determining the bill of materials for the product family will make it possible to investigate various procedures to influence what modules are stored at the distant facility. This is achieved either by improving the current heuristics in order to take into account the assignment constraints, or by introducing new, specific heuristics which take the resulting modules and assign them to minimize logistical costs.

The literature proposes various approaches to the design of product families. Some product design methodologies focus on product architecture (Dahmus, J. B. et al., 2001; Jiao, J. and Tseng, M., 1999). This approach is supported to advantage by modular design (Hung, C. C., Kusiak, A., 1998), component/product/process standardization (Kota, S. et al., 2000; Lee, H. L., Tang, C. S., 1997; Thonemann, U. W., Brandeau, M., 2000). Other methodologies concentrate on process standardization (Lee, H. L., Tang, C. S., 1997), process resequencing (Lee, H. L., 1996), or generic assembly routing (Gupta, S., Krishnan, V., 1998; He, D. W., Kusiak, A., 1997).

In all this work, product, process, and supply chain designs are integrated two by two. However, some recent work explores a global design modeling approach. Agard, B. et al., (2006), for example, propose a genetic algorithm to minimize the mean finished product assembly times for a given demand. Agard, B. and Penz, B. (2007) propose a model for minimizing module production costs and a solution approach based on simulated annealing. Lamothe, J. et al., (2006) use a generic bill of materials representation in order to identify the best bill of materials for each product and the optimal structure of the associated supply chain simultaneously.

In section 2, a more detailed description of the problem is given and an Integer Linear Program model is proposed. Section 3 is devoted to the description of the two heuristic strategies. Some computational experiments are presented and analyzed in section 4. Finally, concluding remarks and perspectives are provided in section 5.

2. THE PROBLEM

Consider the following industrial context: a producer receives customer orders for finished products containing options and variants. Each individual product is then manufactured from a set of modules, the components which come from a number of suppliers (El Hadj Khalaf, R. et al., 2008).

Consider now that the producer has only a short time (T) in which to respond to customer demands, and this is less than the time required to assemble products from elementary components. In addition to this, the producer has to provide the product precisely according to customer demands (without extra options). This constraint comes from technical considerations, or is simply a means to avoid supplementary costs.

To satisfy the customers, the producer brings in pre-assembled components, called modules, from many suppliers located at facilities around the world whose production costs are low. The modules are then assembled at the producer's facility, which is assumed to be close to the customers, and so be able to react very quickly and offer a reduced lead time. Our problem is to define a bill of materials for each finished product which will minimize total assembly costs.

The problem is based on the assumption that a product or a module is considered to embody the set of functions that represents the requirements of the customer, so we assume that:

- a function F_k, is a requirement that must be met by the finished product;
- a module M_j is an assembly of functions that could be added to other modules to make up 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:

- $F = \{F_1,...,F_q\}$: a set of q functions which can appear in both finished products and modules.
- $P = \{P_1,...,P_n\}$: a set of n possible finished products which may be demanded by at least one customer. We note D_i is the estimated demand of the product P_i during the life cycle of the product family.
- $M = \{M_1,...,M_m\}$: a set of m possible modules.
- CF_j: the fixed cost to management of module M_j at the nearby facility.
- CV_j: the variable cost of assembly of module M_j at the nearby facility.
- w_j: the time required to assemble module M_j in a finished product (which is fixed to 1 time unit for all modules).
- T: the time available to assemble a finished product from the modules.
- WhP_i, WhM_j: the weights of product P_i and module M_j respectively (which represent the number of existing functions in a product or module).

Under these assumptions, a product (or module) can be 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 M contains m modules. These may consist of either a selection of the modules defined at the engineering stage, or of the modules resulted from all the possible combinations of the function set.

The problem now is to determine the subset $M'' \in M$, of minimum cost, such that all products in P can be built in a constrained time window T with elements from M'. Concerning the products, the goal is to determine which bill of materials is the most suitable (Figure 1).

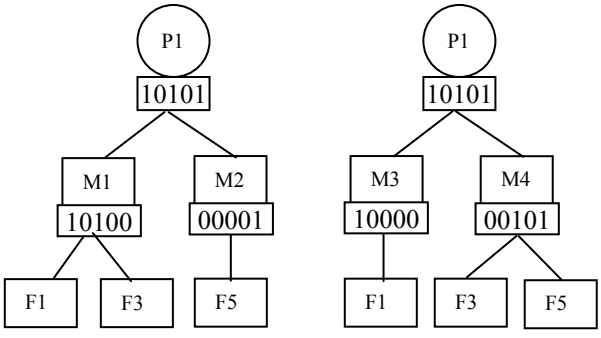

Figure 1 Alternative bills of materials

The following model uses an Integer Linear Program formulation. Our objective is to minimize the costs associated with the producer activity. These costs are as follows: fixed costs related to module management at the nearby facility and module assembly costs at the nearby facility.

$$\min\left(\sum_{j=1}^{m} CF_j Y_j + \sum_{j=1}^{m} CV_j(\sum_{i=1}^{n} D_i X_{ij})\right)$$

s.t.

$$AXi = Pi \qquad \forall i \in \{1,...,n\} \tag{1}$$

$$\sum_{j=1}^{m} w_j X_{ij} \le T \qquad \forall i \in \{1,...,n\} \tag{2}$$

$$X_{ij} \le Y_j \qquad \forall i \in \{1,...,n\} \forall j \in \{1,...,m\} \tag{3}$$

$$Y_j, X_{ij} \in \{0,1\} \qquad \forall i \in \{1,...,n\} \forall j \in \{1,...,m\} \tag{4}$$

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 (belongs to M', the set of produced modules), 0 otherwise; A is the binary matrix, column j of which is the vector M_j; X_j is the column vector composed of the variables X_{ij}.

The objective function minimizes the costs incurred at the nearby facility, where $(\sum_{i=1}^{n} D_i X_{ij})$ is the total need of module M_j.

Constraint (1) shows that a finished product P_i must be assembled precisely 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) translates the relation between the X_{ij} and Y_j variables. If a module is used in the bill of materials of some products, then it belongs to M'.

The problem described here involves the set-partitioning problem, which enables us to conclude that it is NP-hard in the strong sense of the term (Garey, M.R. and Johnson, D. S., 1979).

3. HEURISTIC DESCRIPTION

3.1. The product-building heuristic (PBH)

The main idea behind this heuristic is to assign an index to each module. The value of the index is high when the module is attractive, i.e. the module can be

included in many products and its cost is low. Then, we determine the bill of materials of the products one by one (by solving the MIP), in such a way as to maximize the sum of the indices of a product's components. At this point, the problem described above is solved at each iteration, but for one finished product at a time.

The indices are as follows:

$Coef_j = F1(CF_j) + F2(CV_j) + F3(WhM_j)$ where:

$$F1(CF_j) = 100\frac{\overline{CF}}{CF_j} \text{ with } \overline{CF} = \sum_{j=1}^{m}\frac{CF_j}{m}$$

$$F2(CVj) = 100\frac{CV_j\sum_{i\leftrightarrow j} D_i}{\overline{CV}\sum_{i=1}^{n} D_i} \text{ with } \overline{CV} = \sum_{j=1}^{m}\frac{CV_j}{m}$$

and $i \leftrightarrow j$ means that product P_i is compatible with module M_j (M_j does not contain a function which does not exist in P_i). In this case, the term $\sum_{i\leftrightarrow j} D_i$ represents the needs of module M_j if it is used in the remaining compatible products at iteration k.

As soon as a module is compatible with the finished products, *F2* becomes bigger, and so this index favors modules which are compatible with many finished products.

Two different functions *F3* are used:

$$F3_1(WhM_j) = 100 \text{ if } WhM_j \leq \left\lceil \frac{q}{T} \right\rceil, 0 \text{ otherwise.}$$

This function favors small modules, which is necessary when the cost configuration is such that the total fixed costs are higher than the total variable costs. In such a case, it is better to select small modules because they are compatible with many more products, and using them leads to a solution with a small number of modules. This reduces the fixed costs, which represent the greater part of the objective function (El Hadj Khalaf, R. et al., 2008).

$F3_2(WhM_j) = 100$ if $WhM_j \geq \overline{WhP}$, 0 otherwise, with $\overline{WhP} = \sum_{i=1}^{n}\frac{WhP_i}{n}$.

This function favors relatively large modules, which is necessary when the cost configuration is such that the total variable costs are higher than the total fixed costs. In such a case, using large modules makes it possible to have a small value of requirements (because large modules are not compatible with many finished products). This reduces the total variable costs, which represent the greater part of the objective function.

Since the ratio between fixed and variable costs is not known in advance, this heuristic is tested with both functions *F3*. Moreover, to improve this heuristic, finished products are sorted by increasing (and decreasing) order of their weights.

3.2. The module-selecting heuristic (MSH)

This heuristic is quite simple, the idea behind it being to select the module having the smallest value of $CF_j + CV_j\sum_{i\leftrightarrow j} D_i$ and insert it into the bill of materials of compatible finished products.

To improve this heuristic, we first determine, at each iteration k, the ideal weight of the module to be selected by the formula $Wh_k = \left\lceil \frac{1}{n_k}\sum_i \frac{WhP_{ik}}{W_{ik}} \right\rceil$ with WhP_{ik} the weight of the remaining functions of product P_i at iteration k which are not yet covered, and with $W_{ik} = T - W0_{ik}$, where $W0_{ik}$ is the number of modules inserted into the bill of materials of product P_i up to iteration k, and n_k is the number of remaining products at iteration k (those that still have an incomplete bill of materials).

This operation is aimed at avoiding the selection of unitary modules (those with a weight of 1) at the beginning, which has been proved to damage the quality of the solution: if unitary modules are selected and T is small, then, at certain iterations (when $T-1$ modules have been inserted in previous iterations), it is necessary to complete the bill of materials for products having $WhP_i > T$ with a large module which is not necessarily compatible with other products. If T is small, then selecting unitary modules at early iterations leads to a solution which contains many modules, because many of them are large and they are used in only one or two bills of materials. Such solutions generate very high fixed costs, and consequently a larger objective function.

So, we can summarize this heuristic as follows at iteration k (note that P_{ik} are the remaining functions of product P_i that are not covered by the modules inserted in its bill of materials before iteration k):

1. Calculate the ideal weight Wh_k of the module to be selected, with $WhP_{i1} = WhP_i$ and $n_1 = n$, and $W_{i1} = T$ and $P_{i1} = P_i$
2. Select the module M_j, the weight of which is equal to Wh_k having the smallest value of $CF_j + CV_j \sum_{i \leftrightarrow j} D_i$ (i represents the products P_{ik} compatible with M_j).
3. Insert the module selected into the bill of materials of compatible finished products.
4. For these compatible products, update the product code: $P_{ik} = P_{i(k-1)} - M_j$
5. Also, update $W_{ik} = W_{i(k-1)} - 1$, which represents the maximum number of modules that can be inserted into product P_{ik}.
6. For products having $W_{ik} = 1$, complete their bill of materials with the only compatible module.
7. Repeat these steps until all the bills of materials have been constructed.

4. COMPUTATIONAL EXPERIMENTS

4.1. Datasets and experimental conditions

Ten randomly generated small instances for five different sizes are used; $q \in \{8, 10, 11, 12, 13\}$ on which the module set, the finished product set, the demand D_i, the assembly operating times w_j, the minimum number of functions per product *Minf*, and the maximum number of functions per product *Maxf* were fixed. Table (1) summarizes the various parameters for each instance size.

The demand D_i of a product P_i is a decreasing function on the product function number. This means that, as soon as the finished product contains more options, then the demand for it becomes less than if it had only a few options. The assembly operating times w_j were fixed to 1, so that constraint (2) results in a limitation on the number of modules in a bill of materials.

q	**8**	**10**	**11**	**12**	**13**
m	255	1023	2047	4095	8191
n	30	40	60	80	100
Minf	3	3	3	4	4
Maxf	6	7	8	8	9

Table 1 Instance parameters

In order to simplify the problem data, the following rules are imposed on the various costs:

- $CF_j = \alpha\left(f\left(q_j\right) + \lambda_1\right)$
- $CV_j = \beta\left(f\left(q_j\right) + \lambda_2\right)$

where q_j is the number of existing functions in module M_j; f is the square-root function $f(q_j) = \sqrt{q_j}$, this function representing the relation between costs and q_j better than the identity and square functions (El Hadj Khalaf, R. et al., 2008); α, β are coefficients used to scan different cost configurations; and λ_1, λ_2 are jamming factors.

For each instance size, three cost files are generated by the method described above. The aim is to scan different cases of the ratio between fixed and variable costs.

Table 2 gives the values assigned to α and β for each cost. For cost 1, the fixed costs predominate over the variable costs, while for cost 2, the two costs are almost equivalent, and for cost 3, the variable costs predominate.

Cost	**1**	**2**	**3**
α	1000	240	100
β	0.10	0.40	0.50

Table 2 Cost Configurations

Also, $8\% \leq \lambda_1, \lambda_2 \leq 12\%$. For the tests, T was varied from *Minf* to *Maxf*. The heuristics were then tested on all instances and all costs. Finally, the mean of the ten instance objective values was recorded for the comparison analyses.

4.2. Result analyses

The first objective was to compare the heuristic results with optimal solutions. The heuristics were tested on an instance of size eight (q = 8), for which the optimal solutions are known (El Hadj Khalaf, R. et al., 2008). Table 3 shows the gap rate between the heuristic results and the optimal solutions for the three cost structures.

T		3	4	5	6
Cost 1	PBH	54%	28%	8%	0%
	MSH	29%	9%	11.5%	0%
Cost 2	PBH	35%	19%	7%	0%
	MSH	15%	2.2%	2.4%	0%
Cost 3	PBH	25%	16%	12%	7%
	MSH	9%	2.8%	4.3%	7%

Table 3 Gap rate between heuristic results and optimal solutions for $q = 8$

A first reading of these results shows that the MSH heuristic is better than the PBH one. However, it can be noted that both heuristics run well for medium and high values of *T*, and especially for cost 2 and cost 3, that is, when fixed assembly costs are not greater than the variable assembly costs. This indicates that these heuristics optimize variable costs well.

Our second objective was to test the two heuristics on relatively large instances, in order to compare their performance for larger problems. We achieved this with the tests on sizes 10, 11, 12, and 13.

Figures 2, 3, and 4 show the gap rate (as a percentage) between the results of the product-building heuristic and the module -selecting heuristic (the objective MSH value is considered as the reference value).

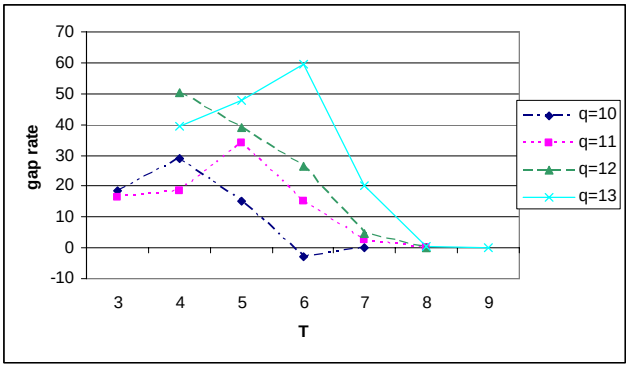

Figure 2 Gap rate between PBH & MSH for cost 1

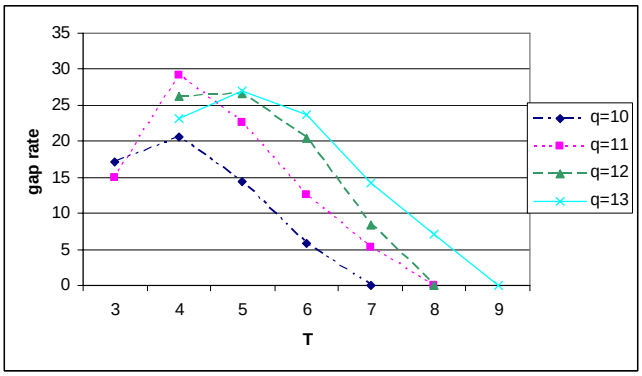

Figure 3 Gap rate between PBH & MSH for cost 2

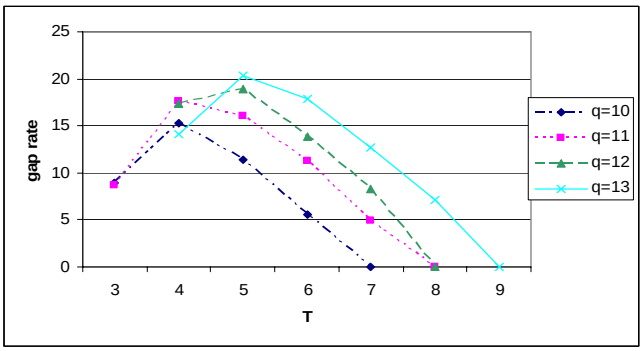

Figure 4 Gap rate between PBH & MSH for cost 3

These figures confirm the result obtained for *q=8*, which is that the MSH is much better than the PBH, and this is true for all sizes, delays, and costs. Also note that the gap rate increases as the problem size increases. In such a case, the gap rate for *q=13* is generally greater than the gap rate for *q=12*, and so on.

Furthermore, the gap rates are generally larger for cost 1 than for cost 2, and for cost 2 than for cost 3. This indicates that the MSH optimizes fixed costs more efficiently than the PBH.

Finally, Figure 5 shows the computational time of the PBH for *q=13*. This figure indicates that the PBH is much more time-consuming than the MSH, which takes an average of one second per instance for all sizes (which is very quick). Now, the PBH needs more computational time as the size of the problem increases (an average of 200 seconds for *q=12*). This is expected because: (1) the PBH tests two indicators with two sorting methods (four combinations), and especially because (2) it uses Cplex to determine the

product bill of materials, which requires much more time.

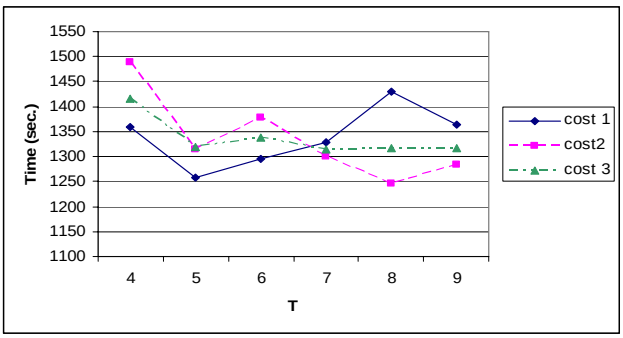

Figure 5 Computational time of the PBH for $q = 13$

Finally, Table 4 shows the gap rate (as a percentage) between the MSH results and the problem's linear relaxation results for each problem size, cost file, and delivery time.

This shows that the gap rate is quite reasonable for medium and large values of T, possibly reaching 0% in some cases (when $T = Maxf$). However, for small values of T, it becomes huge. With small values of T, finding an optimal bill of materials for each product is much more difficult, because of constraint (2). It takes much more time for Cplex to reach the optimal solution.

Again, the gap rate is greater (for small delays) for cost 1 than it is for the other costs. We can hope that the MSH will provide relatively good solutions for cost configurations on which fixed assembly costs do not predominate over variable assembly costs.

5. CONCLUSION

This paper has been devoted to a difficult industrial problem which arises when companies try to offer a large variety of products to consumers. In this problem, the process of selection of components (modules) has to be efficient. The modules are produced for stock and used at the final stage of production, which is on the assembly line. Several authors have considered this problem based on different assumptions: for example, a function can appear twice in a final product, or a final product can be substituted by another one containing more functions - but few papers consider the problem in

		T						
q	**Cost**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
10	**1**	130	65	28	17	0		
	2	65	24	8	4	1.4		
	3	36	16.7	12.8	15.7	19.4		
11	**1**	197	105	39	19	11	0	
	2	89	33.7	12	6.4	5.9	6	
	3	47	22.6	16.3	17.9	22.7	26.9	
12	**1**		141	73	24.4	15.1	0	
	2		44.8	18.4	9.3	9.3	11.5	
	3		28.5	19.8	21	25.4	33.8	
13	**1**		202	110	31.3	20.4	16.5	0
	2		64.2	27.2	11.8	10.3	11.2	15.1
	3		36.1	23.4	22.2	25.8	30.6	38.8

Table 4 Gap rate between MSH results and linear relaxation results

which a final product must correspond exactly to the product ordered by the customer.

We have focused on the assembly operation and tried to determine an efficient set of modules which would allow all products to be assembled, while at the same time avoiding function redundancy and respecting the delivery time. The objective function consists in optimizing the costs incurred during the production activity.

Two heuristics were analyzed: the first determining the bill of materials of finished products one by one, by fixing attractive indices for each module; and the other determining the ideal module verifying the selection criteria and inserting that module into the bill of materials of compatible finished products.

Next to computational speed, the MSH strategy is revealed by the tests to be the one that gives by far the best results. This can be explained by the difficulty in finding good module indices, and especially by the problem structure, in which solutions must take into account the interactions between finished products. This constraint makes it very difficult to determine one product's bill of materials independently of those of other products.

Finally, the MSH results encourage us to use this heuristic as an initial solution for a metaheuristic resolution in the future. It would also be interesting to improve this heuristic to take into account the logistical chain phase.

REFERENCES

Agard, B., Cheung, B., da Cunha, C., (2006), " Selection of a module stock composition using genetic algorithm", in 12th IFAC Symposium on Information Control Problems in Manufacturing – INCOM, Saint-Etienne, France, May 17 – 19.

Agard, B., Penz, B., (2007), "A simulated annealing method based on a clustering approach to determine bills of materials for a large product family", personal communication.

Briant, O., Naddef, D., (2004), "The optimal diversity management problem", Operations Research, Vol 52-4, pp. 515-526.

Dahmus, J. B., Gonzalez-Zugasti, J. P., Otto, K., (2001), "Modular product architecture", Design studies, Vol 22, pp 409-424.

Davis, E. W., (1992), "Global outsourcing: have U.S. managers thrown the baby out with the bath water?", Business Horizons, pp 58-65.

Garey, M. R., Johnson, D. S., (1979), "Computers and intractability. A guide to the theory of NP-completeness", pp 210, Freeman , Oxford, UK.

Gupta, S., Krishnan, V.,(1998), "Product family-based assembly sequence design methodology", IIE Transactions, Vol. 30, pp 933-945.

El Hadj Khalaf, R., Agard, B., Penz, B., (2008), "Product and supply chain design using a two-phases optimization approach", International Conference on Information Systems, Logistics and Supply Chain, Madison, USA, May 27-30.

He, D. W., Kusiak, A., (1997), "Design of assembly systems for modular products", IEEE Transactions on Robotics and Automation, Vol. 13-5, pp. 646-655.

Hung, C. C., Kusiak, A., (1998), "Modularity in design of products and systems", IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, Vol. 28-1, pp. 66-77.

Jiao, J., Tseng, M., (1999), "A methodology of developing product family architecture for mass customization", Journal of Intelligent Manufacturing, Vol. 10, pp. 3-20.

Kota, S., Sethuraman, K., and Miller, R., (2000), "A metric for evaluating design commonality in product families", Journal of Mechanical Design, Vol. 122, pp. 403-410.

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

Lee, H. L., (1995), "Product universality and design for supply chain management", Production Planning and Control, Vol. 6-3, pp. 270-277.

Lee, H. L., (1996), "Effective inventory and service management through product and process redesign", Operations Research, Vol. 44-1, pp. 151-159.

Lee, H. L., Billington, C., (1992), "Managing supply chain inventory: Pitfalls and opportunities", Sloan Management Review, Vol. 33-3, pp. 65-73.

Lee, H. L., Tang, C. S., (1997), "Modelling the costs and benefits of delayed product differentiation", Management Science, Vol. 43-1, pp. 40-53.

Van Roy, T., (1986), "Cross decomposition algorithm for capacity facility location", Operations Research, Vol. 34, pp. 145-163.

Pine B. J., (1993), "II. Mass Customization: The new Frontier in Business Competition", Harvard Business School Press, Boston.

Thonemann, U. W., Brandeau, M. (2000), "Optimal commonality in component design", Operations Research, Vol. 48-1, pp. 1-19.