8ème Congrès International de Génie Industriel – GI 2009, Bagnères de Bigorre, 10-12 juin, 2009.

Conception conjointe de familles de produits et chaîne logistique : Entre diversité et standardisation

RADWAN EL HADJ KHALAF[1], BRUNO AGARD[2], BERNARD PENZ[1]

[1] G-SCOP
GRENOBLE INP-CNRS-UJF
46, Avenue Félix Viallet, F-38031 Grenoble Cedex1
radwan.el-hadj-khalaf@g-scop.inpg.fr

[2] Département de Mathématiques et Génie Industriel
Ecole Polytechnique de Montréal
C.P. 6079, succ. Centre-Ville, Montréal (Québec), H3C 3A7, Canada
Bruno.agard@polymtl.ca

Résumé **- Dans cet article, nous étudions le problème de conception d'une famille de produits devant répondre, de part sa diversité, aux attentes des clients. Cette conception est de type modulaire et permet de générer la nomenclature des produits à partir des modules fabriqués. Conjointement à la définition des nomenclatures, le choix des sites de production est également décidé. Souvent, les stratégies adoptées sont extrêmes, et proposent une diversité totale (tous les produits sont possibles) ou au contraire une forte standardisation (très peu de produits sont proposés). L'objectif de cet article est d'explorer les cas intermédiaire pour mieux comprendre les gains que l'on pourrait espérer.**

Abstract **- In this paper, we study a product family design problem which has to satisfy diversified customer requirements. Modular design strategy allows generating bill of materials of finished products from a subset of modules. Furthermore, the selection of a production location is necessary. Often, adopted strategies are extreme, and propose a total diversity (manufacturing of all possible products) or on the contrary a strong standardization (few products are proposed). The objective of this paper is to investigate intermediate cases to better understand the possible profits.**

Mots clés **- famille de produits, chaîne logistique, conception, optimisation, tabou.**
Keywords **- product family, supply chain, design, optimization, taboo.**

1 INTRODUCTION

Depuis une dizaine d'années, l'industrie tente de proposer aux clients des produits de plus en plus diversifiés pour satisfaire la plupart des segments du marché. Cette approche marketing a conduit à ne plus concevoir des produits uniques, indépendants les uns des autres, mais au contraire des familles de produits, basées sur une plateforme commune, avec un jeu d'options qui permet d'obtenir la diversité voulue.

De ce fait, la conception même des produits a évolué, et un produit fini est maintenant vu comme une base fabriquée en grande série, sur laquelle s'ajoutent des options permettant la personnalisation. Du point de vue de la production, cette diversité est difficile à gérer. Si la base permet la mise en place de lignes de production répétitives, la ligne de finalisation du produit devient, elle, très complexe. Les produits ne peuvent plus être fabriqués pour stock, du fait du nombre important de produits finis possibles. Un assemblage à la commande est donc la seule alternative possible. De plus, la réactivité demandée face au client fait que, bien souvent, le produit ne peut être fini dans un temps suffisamment court ou que les coûts de main d'œuvre pour le faire sont trop élevés. Le regroupement des fonctions en modules assemblés à l'avance permet de résoudre ces difficultés. Ces modules, en nombre plus faible, peuvent alors être fabriqués dans des pays à bas coûts, pour stock, et acheminés sur un site d'assemblage final, proche du marché.

L'objectif de cette étude est alors de trouver un bon compromis entre le nombre de modules à fabriquer dans des sites délocalisés, à choisir ces sites en fonction des coûts de fabrication et des coûts logistiques, et de déterminer les nomenclatures des produits finis pour satisfaire aux contraintes de temps d'assemblage final. Le plus souvent, deux stratégies extrêmes s'opposent. La première consiste à définir un nombre limité de modules, qui serviront de base pour créer la nomenclature de chaque produit (éventuellement une fonction peut être incluse dans le produit même si le client ne la demande pas) ; c'est le principe de la standardisation. Ceci permet de réduire les coûts logistiques entrainés par la diversité, quitte à perdre sur certains produits finis. La deuxième stratégie consiste au contraire à créer des nomenclatures correspondant exactement à la composition voulue du produit fini (aucune fonction en trop). Dans ce cas, le gain sur le prix des composants est évident, mais le coût de la gestion de la diversité augmente.

Dans cet article, nous explorons les stratégies intermédiaires, c'est-à-dire que nous acceptons une standardisation, la présence de fonctions non requises (mais en nombre limité). A notre connaissance, aucun travail n'a été fait dans ce sens dans la littérature.

L'article, dans un premier temps, donnera un aperçu des travaux existant dans la littérature (section 2). Le problème sera ensuite décrit formellement dans la section 3, puis nous présenterons les modélisations envisagées en section 4. Les

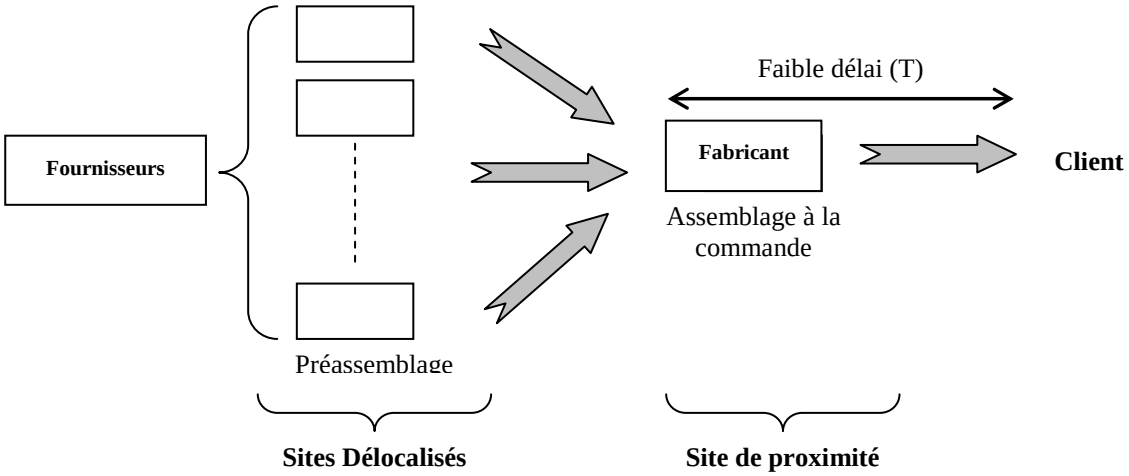

Figure 1. Structure de la chaine logistique

méthodes en cours de développement seront présentées dans la section 5. Quelques résultats seront présentés en section 6. Nous conclurons et donnerons quelques pistes de recherche en section 7.

2 ÉTAT DE L'ART

La littérature est riche en approche de conception de famille de produit. Certains auteurs s'intéressent plus particulièrement à l'architecture des produits [Dahmus et al., 2001][Jiao et Tseng, 1999]. Ces travaux sont le plus souvent liés aux travaux sur la conception de modules [Hung et Kusiak, 1998] ou sur la standardisation des composants, des produits et même des process de production [Kota et al., 2000][Lee et Tang, 1997][Thonemann et Brandeau, 2000]. Des approches différentes s'intéressent plus particulièrement à la standardisation des process [Lee et Tang, 1997], voire au reséquencement des process [Lee, 1996] ou à la mise au point de procédures d'assemblage génériques [Gupta et Krishnan, 1998] [He et Kusiak, 1997]. Sur le plan logistique, les auteurs se concentrent principalement sur les méthodologies de conception de chaines logistiques. Pour cela, ils s'intéressent à l'environnement global de la chaine et particulièrement aux fournisseurs et clients et mettent l'accent sur le coté incertain de cet environnement [Davis, 1992]. D'autres étudient la répartition des stocks et leur variation le long de la chaine logistique [Lee et Billington, 1992]. Enfin, certains optimisent la localisation des sites de production [Van Roy, 1986]. Mais en règle générale, tous ces travaux sur les chaines logistiques considèrent que le produit (ou la famille de produits) est connu, parfaitement décrit et non modifiable.

Lorsqu'on veut intégrer dans une même décision la nomenclature du produit, le choix des modules à fabriquer et leurs lieux de production, la conception du produit, le process d'assemblage et la conception de la chaine logistique sont impactés. Les résultats fournis par la littérature traitent le plus souvent indépendamment ces trois points, au mieux, deux seulement d'entre eux. Lee [Lee, 1995] est le premier à pointer ce manque, et il montre que les choix lors de la conception du produit (conception modulaire, standardisation ou différentiation retardée) a une influence très forte sur la conception de la chaine logistique. Ainsi, il est crucial de déterminer, même grossièrement, la famille de produit, le process de production et la chaine logistique de manière intégrée.

Quelques travaux récents abordent le problème dans sa globalité. Lamothe et al. [Lamothe et al., 2006] utilisent une nomenclature générique pour identifier simultanément la meilleure nomenclature et la structure optimale de la chaine logistique. Agard et al. [Agard et al., 2006] propose un algorithme génétique pour minimiser le temps d'assemblage moyen des produits finis pour une demande donnée. Agard et Penz [Agard et Penz, 2008] ont développé un algorithme de recuit simulé basé sur une méthode de clustering pour déterminer le nombre minimum de modules à fabriquer.

3 DESCRIPTION DU PROBLEME

Le problème que nous traitons dans cet article est introduit dans [El Hadj Khalaf et al., 2008a]. Il consiste en un producteur qui reçoit les commandes de ces clients pour des produits finis contenant des options et variantes. Le producteur doit répondre dans un temps imparti (*T*), et pour cela, il assemble le produit fini à partir de sous-ensembles déjà fabriqués, les modules. Les modules sont fabriqués dans des sites délocalisés où la main d'œuvre est moins chère, puis ces modules sont acheminés sur le site d'assemblage final et stockés en attente d'utilisation. Le schéma général du fonctionnement de la chaine logistique est donné dans la figure (Figure 1).

Plus formellement, voici les différents éléments du problème, ainsi que les principaux coûts à prendre en compte. Dans un premier temps, nous introduisons les notions de fonctions, produits, modules et sites distants :

$F=\{F_1,...,F_q\}$ représente l'ensemble des q fonctions qui peuvent apparaître dans un produit fini;

$P=\{P_1,...,P_n\}$ représente l'ensemble des n produits qui peuvent être demandés par au moins un client;

$M=\{M_1,...,M_m\}$ représente l'ensemble des m modules possibles;

$S=\{S_1,...,S_s\}$ représente l'ensemble des s sites de production distants.

La demande des clients et les contraintes sur le problème s'expriment comme suit :

D_i est la demande estimée du produit P_i durant le cycle de vie de la famille. D_i peut-être obtenue par reconstruction à partir de l'estimation des demandes de fonctions;

q_i indique le nombre de fonctions présentes dans le produit P_i;

Cap_l est la capacité maximale du site de production S_l;

C_{jl} est la charge induite par la production du module M_j dans le site S_l;

w_j est le temps nécessaire pour assembler le module M_j dans un produit fini;
T est le temps maximal d'assemblage autorisé pour la fabrication d'un produit.
Enfin, les différents coûts pris en compte sont les suivants :
CF_j est un coût fixe de gestion du module M_j sur le site d'assemblage;
CV_j est le coût d'un assemblage du module M_j dans un produit fini;
CF_{jl} est un coût fixe de gestion du module M_j dans le site délocalisé S_l;
CV_{jl} est le coût de fabrication du module M_j dans le site distant S_l.
Le problème d'optimisation est maintenant simple à exprimer. Il faut déterminer l'ensemble M' des modules que l'on doit fabriquer. Cet ensemble doit contenir tous les modules nécessaires à l'élaboration des nomenclatures de tous les produits finis potentiels. Lorsque toutes les nomenclatures (voir figure 2) sont connues, on peut en déduire aisément la demande en chacun des modules, et affecter ces productions de modules aux différents sites de production distants. Comme nous venons de l'exprimer, une démarche naturelle pourrait être, dans un premier temps, de déterminer l'ensemble M', pour ensuite déterminer les nomenclatures, et enfin, affecter les productions de modules aux sites. Sur les problèmes sans standardisation, les résultats montrent que cette approche n'est pas performante [El Hadj Khalaf et al., 2008a]. L'objectif est de résoudre globalement ce problème d'optimisation plutôt qu'une succession d'optimisation partielle.
Le problème que nous venons de décrire brièvement est un problème difficile à résoudre à l'optimal. Il est NP-difficile au sens fort car il contient comme sous problème, le problème classique de *set partitioning* [Garey et Johnson, 1979].

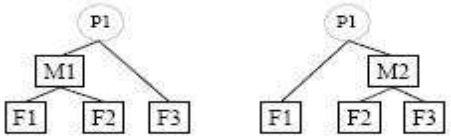

Figure 2. Nomenclatures alternatives

4 MODELISATIONS EN FONCTION DES SPECIFICITES DU PROBLEME

Dans un premier temps, nous présentons le modèle d'optimisation qui permet de rechercher des solutions optimales au problème de diversité totale, c'est-à-dire sans standardisation. Ensuite nous regarderons le modèle où l'on accepte un nombre limité de fonction en surplus, mais sans fonction en double. Le modèle avec fonction en surplus et fonctions en double sera alors présenté. Pour finir, une méthode de résolution sera proposée. Cette méthode a déjà fait ses preuves pour le cas sans standardisation et nous verrons comment l'adapter aux différents cas de standardisation.

4.1 Le modèle sans standardisation

Pour résoudre ce problème de conception à l'optimal, il est nécessaire de déterminer exactement la nomenclature de chaque produit. Pour cela, nous définissons la variable binaire X_{ij} qui prendra la valeur 1 si et seulement si le produit P_i possède le module M_j dans sa nomenclature. Si cela est le cas, la variable binaire Y_j, qui signifie que le module M_j est produit dans au moins un site S_l, prendra la valeur 1. La variable binaire Y_{jl} prendra la valeur 1 si et seulement si le module M_j est produit, au moins en partie, dans le site S_l. Pour finir, la variable Q_{jl} déterminera la quantité de module M_j produite dans le site S_l. Pour une écriture simplifiée du modèle, nous introduisons les paramètres δ_{ik} et λ_{jk}. Le paramètre binaire δ_{ik} vaut 1 si la fonction F_k est présente dans le produit P_i. De même, le paramètre λ_{jk} vaut 1 si la fonction F_k est présente dans le module M_j. Avec ces notations, nous pouvons maintenant écrire le programme linéaire en nombres entiers qui permet de déterminer la solution optimale de notre problème de conception conjointe de famille de produits et de chaine logistique.
La fonction objective s'écrit simplement comme une somme de coût :

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

$$+ \sum_{l=1}^{S} \sum_{j=1}^{m} CF_{jl} Y_{jl} + \sum_{l=1}^{S} \sum_{j=1}^{m} CV_{jl} Q_{jl}$$

Les contraintes linéaires sont les suivantes :

$$\sum_{j=1}^{m} \lambda_{jk} X_{ij} = \delta_k \quad \text{pour tout } P_i \text{ et } F_k \qquad (1)$$

$$\sum_{j=1}^{m} w_j X_{ij} \le T \quad \text{pour tout } P_i \qquad (2)$$

$$X_{ij} \le Y_j \quad \text{pour tout } P_i \text{ et } M_j \qquad (3)$$

$$\sum_{l=1}^{S} Q_{jl} = \sum_{i=1}^{n} D_i X_{ij} \quad \text{pour tout } Mj \qquad (4)$$

$$\sum_{j=1}^{m} C_{jl} Q_{jl} \le Cap_l \quad \text{pour tout } S_l \qquad (5)$$

$$Q_{jl} \le K_{jl} Y_{jl} \quad \text{pour tout } S_l \text{ et } M_j \qquad (6)$$

$$Q_{jl} \ge 0 \quad \text{pour tout } S_l \text{ et } M_j \qquad (7)$$

$$Xij, Yj, Yjl \in \{0,1\} \quad \text{pour tout } P_i, M_j \text{ et } S_l \qquad (8)$$

La contrainte (1) exprime le fait que dans un produit fini P_i, si la fonction n'est pas présente, alors elle ne doit figurer dans aucun module qui compose la nomenclature. En revanche, si la fonction est présente dans le produit P_i, alors un module et un seul doit la contenir. La contrainte (2) s'assure du respect du temps d'assemblage final du produit, et ce, pour chaque produit. La contrainte (3) lie les variables X_{ij} et Y_j. Lorsque Y_j vaut 1, alors le module peut entrer dans la composition de la nomenclature de P_i. Si Y_j vaut 0, alors il n'est pas possible de mettre le module M_j dans la nomenclature. La contrainte (4) garantit que les quantités produites du module M_j sur l'ensemble des sites de production S_l est bien la quantité totale de production demandée. La contrainte (5) garantit que la capacité des sites de production S_l est bien respectée. la contrainte (6) lie les variables Q_{jl} et Y_{jl}. Si Y_{jl} vaut 0, alors aucune quantité de M_j ne peut être produite sur le site S_l. Si Y_{jl} vaut 1, cela signifie que l'on peut produire M_j sur le site S_l. Le paramètre K_{jl} est une grande valeur représentant la quantité maximale de M_j que l'on peut produire sur le site S_l. Elle peut se calculer de la façon suivante :

$$K_{jl} = \min \left\{ Cap_l / C_{jl}, \sum_{i=1}^{n} D_i \right\} \quad \text{pour tout } M_j \text{ et } S_l \qquad (9)$$

Pour finir, les contraintes (7) et (8) permettent de garantir la positivité des quantités de modules produites et assurent que les variables de décision sont binaires.

4.2 Le modèle avec standardisation totale

A partir du modèle précédent, nous pouvons maintenant voir comment le problème de standardisation totale peut être formulé. Pour cela, une simple transformation de la contrainte (1) est suffisante :

$$\sum_{j=1}^{m} \lambda_{jk} X_{ij} = 1 \quad \text{pour tout } P_i \text{ et k/} \delta_k = 1 \qquad (10)$$

$$\sum_{j=1}^{m} \lambda_{jk} X_{ij} \leq 1 \quad \text{pour tout } P_i \text{ et k/} \delta_k = 0 \qquad (11)$$

La contrainte (10) exprime le fait que si la fonction est présente, celle-ci doit figurer dans exactement un module de la nomenclature. La contrainte (11) indique que si la fonction F_k n'est pas présente dans le produit P_i, celle-ci peut quand même être ajoutée.

Ce modèle impose donc bien qu'une fonction demandée est présente en un seul exemplaire, et qu'une fonction qui ne l'est pas soit également présente en au plus un exemplaire dans la nomenclature du produit. Dans la suite, nous explorerons les variantes ou la standardisation est limitée.

4.3 Standardisation partielle sans redondance

Selon la stratégie industrielle voulue par l'entreprise, une limite peut être apportée à la standardisation. Cette limite peut venir d'un surcout de gestion apporté par les fonctions inutiles dans le produit, ou par des objectifs autres, tels que le fait de ne pas vouloir augmenter le poids du produit de base par des fonctions inutiles. Par exemple, il peut être préjudiciable d'ajouter une carte électronique inutile dans un ordinateur portable de base, si cette carte n'est pas utilisée.

Dans ce cas, il est nécessaire d'ajouter une contrainte au modèle pour limiter la standardisation. Celle-ci s'exprime de la façon suivante :

$$\sum_{k=1}^{q} \sum_{j=1}^{m} \lambda_{jk} X_{ij} \leq q_i + \alpha_i \quad \text{pour tout } P_i \qquad (12)$$

La contrainte (12) permet de compter le nombre total de fonctions apportées par la nomenclature. La valeur q_i donne le nombre de fonction minimal que doit comporter le produit (les fonctions demandées) et le paramètre α_i le nombre de fonctions supplémentaires acceptées pour le produit. On voit ici qu'il est possible de limiter la standardisation de manière spécifique pour chaque produit fini.

4.4 Le modèle avec redondance sans standardisation

Dans certaines applications, la redondance est parfois acceptée. La redondance est le fait de placer deux fois une fonction demandée (apportée par deux modules différents) dans le même produit fini. C'est le cas, par exemple, dans l'industrie de la micro-informatique. Supposons qu'un constructeur propose deux versions d'un ordinateur. La première version possède, entre autre, une carte mère et une carte graphique de base. La version plus sophistiquée possède la même carte mère mais une version plus puissante de carte graphique, demandée par seulement 5% de clients passionnés de jeux vidéos. Le constructeur à la possibilité de monter la carte mère et la bonne carte vidéo en fonction de la demande du client. Mais il a également la possibilité de concevoir une carte mère contenant déjà la carte graphique de base, qu'il installera sur tous les ordinateurs. Il ajoutera à cette carte mère incluant la carte graphique de base, la carte graphique plus puissante à la demande. Il n'aura ainsi que deux cartes à gérer, et dans seulement 5% des cas, il aura les composants de la carte graphique de base en surcout. Dans ce cas, une même fonction devrait apparaître plusieurs fois dans le produit (plusieurs indices k), et des contraintes supplémentaires devraient être ajoutées pour éviter que plusieurs options d'une même fonction soient présentes dans la nomenclature.

La redondance que nous traitons là est différente et concerne l'apparition d'une même fonction (identique) plusieurs fois. C'est le cas par exemple pour les faisceaux électriques où des fils inutiles peuvent appartenir à un faisceau, même s'ils ne sont pas utilisés. Dans notre modélisation, il suffit de modifier la contrainte (10) pour la remplacer par la suivante :

$$\sum_{j=1}^{m} \lambda_{jk} X_{ij} \leq 2 \quad \text{pour tout } P_i \text{ et k/} \delta_k = 1 \qquad (13)$$

$$\sum_{j=1}^{m} \lambda_{jk} X_{ij} \geq 1 \quad \text{pour tout } P_i \text{ et k/} \delta_k = 1 \qquad (14)$$

$$\sum_{j=1}^{m} \lambda_{jk} X_{ij} = 0 \quad \text{pour tout } P_i \text{ et k/} \delta_k = 0 \qquad (15)$$

La contrainte (13) permet une redondance sur les fonctions devant être présentes. En modifiant la valeur 2 par un paramètre, on pourrait facilement accepter que certaines fonctions apparaissent plus de deux fois dans un produit fini, mais cela ne semble pas très réaliste d'un point de vue industriel. La contrainte (14) permet de garantir que les fonctions demandées sont bien présentes au moins une fois. Enfin, la contrainte (15) empêche la standardisation.

Si l'on souhaite en plus limiter le nombre de redondances, il est nécessaire de compter le nombre total de fonctions présentes dans la nomenclature du produit et de la comparer avec le nombre de fonctions demandées. Cette contrainte est la suivante :

$$\sum_{k=1}^{q} \sum_{j=1}^{m} \lambda_{jk} X_{ij} \leq q_i + \beta_i \quad \text{pour tout } P_i \qquad (16)$$

Dans ce cas, le paramètre β_i donne le nombre de redondance en trop, et donc le nombre de fonctions en double dans le produit fini P_i.

4.5 Standardisation limitée et redondance

Dans le cas le plus général, il est possible d'avoir en même temps de la redondance et des fonctions non demandées dans le produit fini. Le modèle doit alors contenir les contraintes (13) et (14) pour permettre la redondance, la contrainte (11) pour permettre la standardisation, et la contrainte (12) pour limiter le nombre de fonctions supplémentaires. Dans ce cas, le paramètre α_i représentera le nombre de fonction en trop, incluant à la fois la redondance et la standardisation.

Pour différencier les fonctions supplémentaires en fonction de leur origine, l'ajout des deux contraintes suivantes serait nécessaire :

$$\sum_{k\,/\,\delta_k=1} \sum_{j=1}^{m} \lambda_{jk} X_{ij} \leq q_i + \beta_i \quad \text{pour tout } P_i \qquad (17)$$

$$\sum_{k\,/\,\delta_k=0} \sum_{j=1}^{m} \lambda_{jk} X_{ij} \leq \gamma_i \quad \text{pour tout } P_i \qquad (18)$$

Ici, le paramètre β_i donne le nombre de redondance et le paramètre γ_i le nombre de fonction en trop dû à la standardisation.

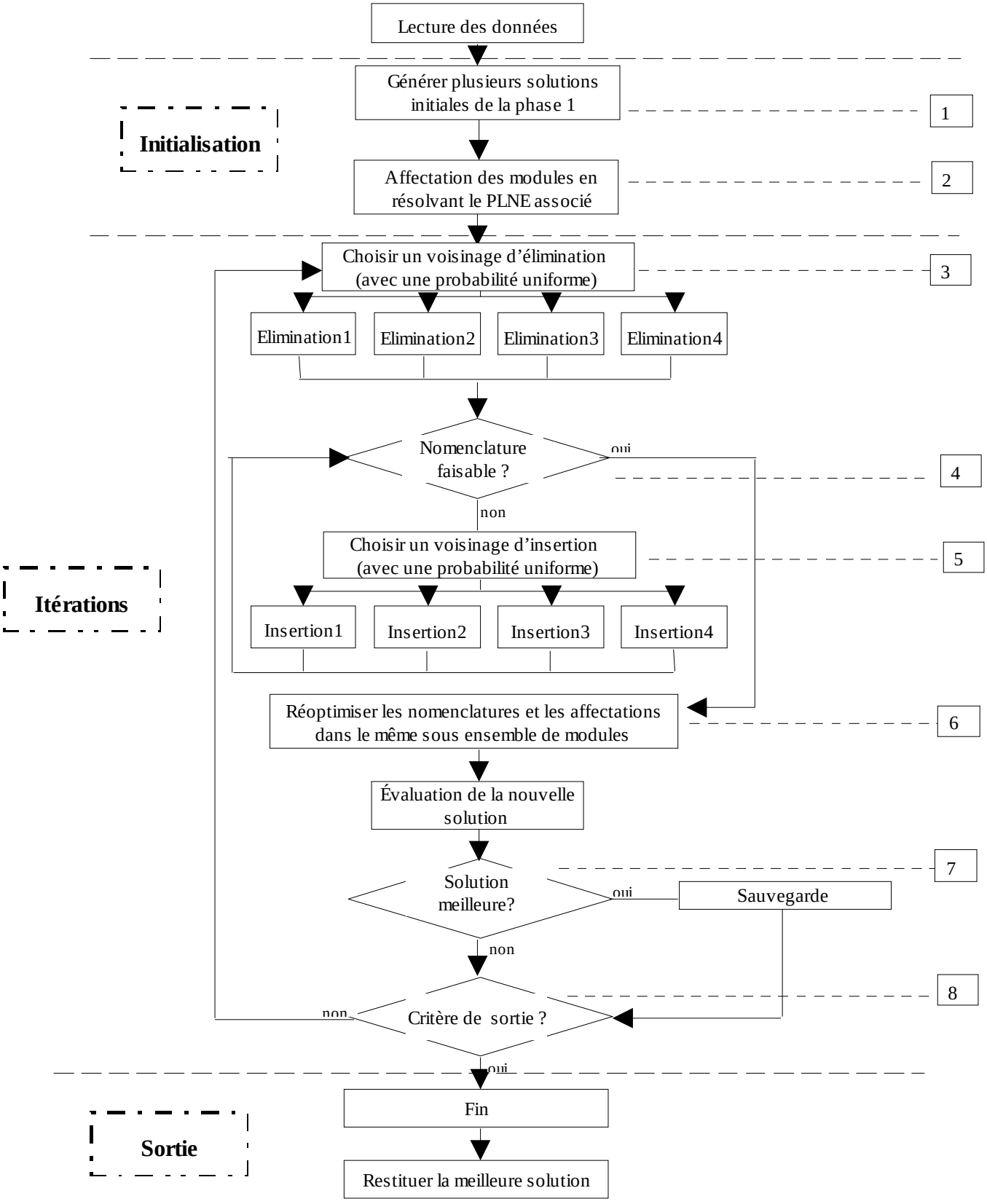

Figure 3. Algorithme tabou

4.6 Commentaires sur les sources d'approvisionnement

Nous avons jusqu'à présent regardé l'impact sur le modèle de la redondance et de la standardisation lors de la détermination de la nomenclature. Des variantes peuvent également apparaître dans la partie logistique. Il est effectivement possible de borner le nombre de sites de production où le module M_j sera produit. Pour cela, l'ajout des deux contraintes suivantes est nécessaire :

$$\sum_{l=1}^{S} Y_{jl} \leq \eta_j \quad \text{pour tout} \quad M_j \quad (19)$$

$$\sum_{l=1}^{S} Y_{jl} \geq \varepsilon_j \quad \text{pour tout} \quad M_j \quad (20)$$

La contrainte (19) impose que le nombre de sites ne peut excéder η_j pour le module M_j, pour éviter une trop grande

dispersion des fournisseurs. La contrainte (20) oblige à produire dans au moins ε_j sites. Ce dernier peut prendre la valeur 1, ce qui garantit qu'au moins un fournisseur est nécessaire, mais également une valeur plus grande imposant de ce fait la multiplication des sources d'approvisionnement.
Dans ces dernier cas, il est également possible de contraindre chaque fournisseur à produire au moins un certain pourcentage du nombre de modules total pour le module M_j. Cela est souvent utilisé pour garantir pour chaque fournisseur un effet de production de masse permettant de réduire les coûts de production. La contrainte s'écrit alors :

$$Q_{jl} \geq \tau_{jl} \sum_{i=1}^{n} D_i X_{ij} \quad \text{pour tout } M_j \text{ et } S_l \quad (21)$$

Le paramètre τ_{jl} indique le pourcentage minimum de modules M_j que doit produire le site S_l.

5 APPROCHE DE RESOLUTION PROPOSEE

Dans la suite, nous proposons une approche taboue pour résoudre ce problème. Comme nous l'avons dit dans la section 3, ce problème est difficile à résoudre, et les solveurs de programmation linéaire en nombres entiers peinent à trouver des solutions en temps raisonnable, même pour le problème sans standardisation avec des produits contenant 8 fonctions au maximum [El Hadj Khalaf et al., 2008b].
La solution en cours de développement pour traiter ce problème difficile et une méthode taboue. Cette méthode généralise celle déjà proposée pour le cas sans standardisation, et qui a montrée qu'elle pouvait résoudre des problèmes avec plus de 15 fonctions dans un produit fini [El Hadj Khalaf et al., 2008b].
Les principales étapes de la méthode sont les suivantes. Dans un premier temps (étape 1 & 2), une solution initiale est générée soit au hasard, soit à l'aide d'une heuristique (donnée plus loin) et une nomenclature pour chaque produit est calculée par la résolution d'un programme linéaire en nombres entiers (PLNE) basé sur les contraintes (1), (2), (3) et (8). Les nomenclatures sont calculées pour chaque produit, l'un après l'autre, en fixant l'indice i dans chaque PLNE. Avec 15 fonctions, ce PLNE est résolu rapidement par un solveur. Les modules alors choisis sont ensuite affectés aux sites de production par la résolution d'un autre PLNE. Celui-ci est basé sur les contraintes (4), (5), (6), (7) et (8). Là encore, la résolution à l'optimal est rapide car le nombre de modules choisis est faible au regard de la combinatoire total du nombre de modules, et le nombre de sites de production plutôt faible dans les cas industriels. Les étapes 3 à 8 constituent le cœur de l'algorithme tabou. Elles consistent en la construction progressive de nouveaux ensembles de modules permettant de trouver les nomenclatures des produits. Dans l'étape 3, des modules sont éliminés du sous-ensemble de modules choisis, et l'étape 5 permet l'ajout de nouveaux modules tant que la faisabilité des nomenclatures n'est pas garantie. Lorsqu'un nouveau sous-ensemble de modules est généré et que les nomenclatures sont calculées, l'étape 4 permet de vérifier les produits infaisables et de mettre à jour les nouvelles nomenclatures. La nouvelle solution est ensuite évaluée et gardée si elle est meilleure que les précédentes (étape 7). La meilleure solution trouvée est enfin donnée lorsqu'un des critères d'arrêt est rencontré (étape 8).
Lors des phases de suppression ou d'ajout de modules dans le sous-ensemble courant, le principe de la recherche taboue est appliqué. Il consiste ici à mémoriser les suppressions et ajouts effectués dans les itérations précédentes dans la liste taboue, et des les interdire pendant un certain nombre d'itérations.
L'algorithme est donné dans la figure 3.

5.1 Heuristiques de calcul d'une solution initiale

Pour calculer une solution initiale, nous avons mis au point les heuristiques suivantes. La première (H1) consiste à générer aléatoirement un sous-ensemble de modules. La deuxième (H2) calcule pour chaque produit une nomenclature optimisée en fonction des coûts d'assemblage final, mais sans tenir compte des coûts de production dans les sites. Enfin, la troisième (H3) fonctionne comme la deuxième, mais intègre une estimation des coûts logistiques, c'est-à-dire des coûts de production dans les sites distants. Ces heuristiques ont été expliquées en détails dans [El Hadj Khalaf, 2008c]
Les tests préliminaires ont permis de montrer que la vitesse de convergence de l'algorithme tabou était améliorée par l'utilisation de l'heuristique H3.

5.2 Contrôle de la faisabilité des nomenclatures

Après élimination ou ajout de modules dans le sous-ensemble des modules choisis, la nomenclature de chaque produit fini est calculée par la résolution du PLNE basé sur les contraintes (1), (2), (3) et (8). Si le produit fini admet une nomenclature avec le nouveau sous-ensemble, elle est alors mise à jour. Sinon, elle reste indiquée comme non faisable pour la suite de l'algorithme, et elle sera évaluée de nouveau à la prochaine itération. Bien sûr, seule les nomenclatures impactées par un changement de modules sélectionnés seront évaluées, celles qui étaient faisables avec des modules restant dans le sous-ensemble ne seront pas vérifiées.

5.3 Procédure de réoptimisation

Cette procédure permet un nettoyage du sous-ensemble M'. Elle permet de supprimer des modules inutilisés dans les nomenclatures. Elle permet également un calcul de la nouvelle affectation des modules aux sites par la résolution du PLNE basé sous les contraintes (4), (5), (6), (7) et (8), avec comme objectif uniquement les coûts logistiques.

5.4 Voisinage d'élimination

Quatre voisinage ont été testés pour l'élimination de modules de M'.

1. Élimination d'un module de faible degré (un module ne pouvant pas entrer dans beaucoup de nomenclatures)
2. Élimination d'un module générant de forts coûts logistiques
3. Élimination d'un module de haut degré (car susceptible de générer des coûts variables élevés)
4. Élimination aléatoire

Dans la première version de nos tests, ces voisinages ont été utilisés aléatoirement au cours des itérations. Dans les tests à venir, une attention particulière sera mise sur un choix opportun de ces voisinages.

5.5 Voisinages d'insertion

Là encore, quatre voisinages ont été proposés:

1. Insertion d'un module à faible coûts logistiques
2. Insertion d'un module à fort degré
3. Insertion d'un module à fort degré uniquement sur la base des produits infaisables
4. Insertion d'un module permettant de rendre un produit infaisable

A nouveau, ces voisinages ont été utilisés aléatoirement au cours de l'algorithme. Une nouvelle fois, il serait intéressant de regarder la qualité de ces voisinages indépendamment les uns des autres.

5.6 Critères de sortie

Différents critères de sortie sont utilisés simultanément. Il s'agit du temps total de calcul et du nombre d'itérations sans amélioration de la meilleure solution. L'algorithme s'arrête lorsque le premier de ces critères est rencontré.

5.7 Quelques éléments sur la qualité de la méthode taboue

Cette méthode taboue a été testée sur des instances où ni la standardisation, ni la redondance ne sont acceptées. Les premiers résultats indiquent que cette méthode est performante.

Sur les instances à 8 fonctions, les tests ont été effectués sur cinq instances de produits finis (et demandes) différentes et trois configurations de coûts. Le tabou tourne pendant 30 minutes et les solutions trouvées sont comparées avec les solutions optimales (possibles sur des petites instances). La comparaison révèle un écart de 7% au plus par rapport à l'optimal. De plus, les tests ont montré que l'application est assez stable par rapport aux instances et aux coûts ce qui prouve son bon paramétrage.

Sur les instances à 15 fonctions, le tabou tourne pendant six heures en commençant avec trois solutions initiales différentes. Les heuristiques utilisées pour déterminer la solution initiale sont détaillées dans la section 5.1. Là également, les tests ont été effectués sur trois configurations de coûts. A la fin de l'exécution, les écarts entre les valeurs obtenues avec les trois points de départ différents ne dépassent pas au pire des cas 5%. En revanche les solutions initiales heuristiques permettent de converger plus vite que la solution aléatoire.

5.8 Comment adapter cet algorithme à la standardisation

L'adaptation du tabou pour la résolution du problème avec standardisation serait très simple. En effet, il suffit de modifier la procédure de contrôle de la faisabilité des nomenclatures afin qu'elle tienne compte des nouvelles contraintes ((10)…(18)). Il faut adapter aussi les indicateurs utilisés pour l'exploration des voisinages. Par exemple, dans le cas de standardisation, le degré d'un module va certainement augmenter (par rapport à sa valeur dans le cas de non standardisation) car il deviendra candidat dans beaucoup plus de nomenclatures puisque une redondance ou une standardisation des fonctions seront permises sur l'assemblage du produit fini.

6 QUELQUES RESULTATS

Nous avons effectué quelques tests sur une petite instance afin de comparer les différentes stratégies d'assemblage proposées dans cette étude. Une petite instance de huit fonctions par produit fini a été aléatoirement générée dans laquelle trente produits finis ont été générés comportant chacun entre 3 et 6 fonctions. Également, des coûts fixes et variables ont été générés aléatoirement dans le site d'assemblage et les sites de production pour chaque module. Deux sites délocalisés sont considérés dans cette expérimentation.

La figure 4 montre l'évolution de la valeur de la fonction objective (optimale par résolution du PLNE) en fonction du temps d'assemblage disponible T pour les différents scénarios explorés dans ce papier. Elle compare aussi ces scénarios avec le problème de référence « Opt » qui est l'assemblage sans standardisation et sans redondance. « Red » représente les résultats du modèle redondance sans standardisation (sans limitation sur le nombre de fonctions redondantes), « St1 » représente le modèle standardisation partielle sans redondance avec α_i =1 pour tout Pi, « St2 » représente le même modèle avec α_i =2 pour tout P_i, « St » représente le modèle standardisation totale et enfin

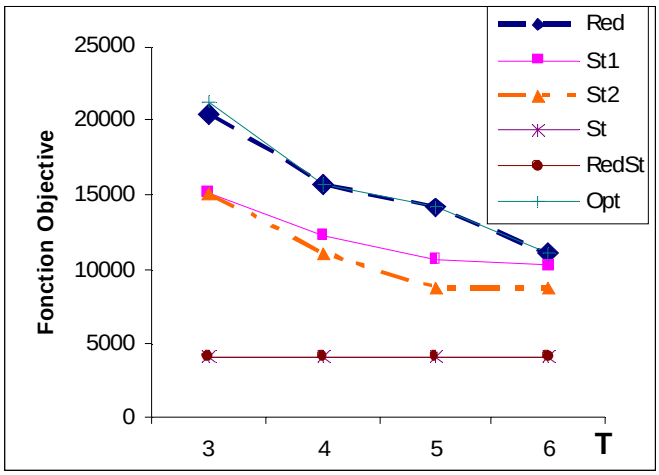

Figure 4. Évolution de la fonction objective en fonction de T et des différentes stratégies d'assemblage

« RedSt » représente le modèle standardisation totale avec autorisation de redondance.

On constate que le modèle « Red » ne permet pas d'améliorer considérablement la valeur de la fonction objective, nous avons particulièrement constaté que ceci est dû à la structure des coûts pour laquelle les coûts engendrés dans le site d'assemblage sont plus importants que ceux engendrés dans les sites délocalisés (c'est la structure des coûts utilisée dans les tests). D'autres tests ont montré que cette stratégie peut apporter des améliorations dans le cas contraire.

En revanche la standardisation permet un meilleur gain que la redondance et ce gain augmente quand le paramètre α_i augmente. On constate aussi que pour un α_i donné, l'augmentation de T permet aussi de diminuer les coûts totaux (la fonction objective).

Finalement, on voit que les deux stratégies « St » et « RedSt » aboutissent pratiquement au même résultat ce qui signifie que la stratégie « St » est la plus dominante ce qui est logique car « Red » n'a pas de grandes améliorations par rapport à « Opt ».

On constate également que T n'a pas d'influence pour ce cas de figure car cette stratégie extrême (standardisation totale) aboutit à une solution pour laquelle chaque produit finis possède dans sa nomenclature un seul module (qui comporte certainement beaucoup de fonctions supplémentaires), et c'est pour cette raison que l'on obtient une valeur indépendante de T.

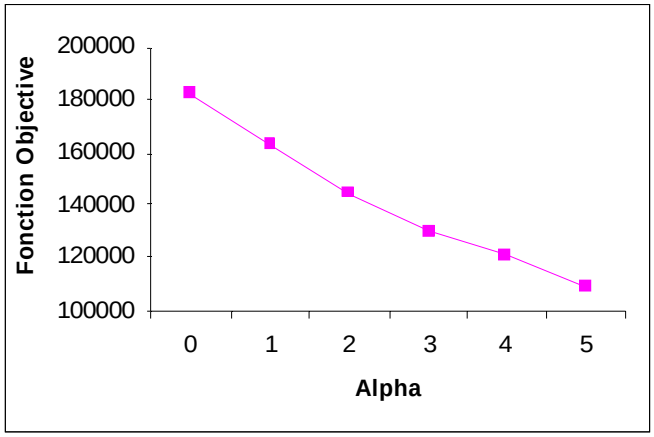

Figure 5. Évolution de la fonction objective en fonction de α pour T = 4

Ensuite, nous avons utilisé notre algorithme tabou pour effectuer des tests sur une grande instance de 15 fonctions par produit fini, Nous nous sommes particulièrement intéressé à l'évolution de la fonction objective en fonction de α_i (nombre de fonctions ajoutées par produit fini) du modèle « standardisation partielle sans redondance ». Pour ces tests T est figé, et α_i est le même pour tout les produits finis.

La figure 5 montre bien que la stratégie de standardisation partielle est rentable aussi pour des problèmes de grande taille et que notre application permet bien d'optimiser ce modèle là (ainsi que le modèle initial qui correspond à $\alpha = 0$), et on constate que la fonction objective décroît évidement quand α augmente.

En revanche, pour le moment, le tabou n'est pas capable d'optimiser le modèle de la redondance, probablement car la structure de coût n'est pas compatible avec ce modèle. Nous sommes en train d'élaborer des structures de coûts pour lesquelles nous sommes certains que la redondance peut être rentable et dans ce cas nous allons perfectionner le tabou afin qu'il puisse optimiser ce modèle.

7 CONCLUSION ET PERSPECTIVES

Cet article avait pour objectif de proposer des modélisations générales pour la résolution de problèmes de conception conjointe de famille de produit et de chaines logistiques. Après un rappel d'un modèle déjà traité où chaque produit fini doit contenir exactement les fonctions dont il a besoin, sans fonction apparaissant en double, nous avons proposé des modèles permettant une standardisation contrôlée.

Ces modèles mathématiques sont difficiles à résoudre (difficile au sens de la théorie de la complexité) et de ce fait, presque impossibles à résoudre pour des problèmes de taille industrielle. C'est pourquoi une approche heuristique a été envisagée, celle-ci étant une extension d'une méthode ayant été développée, testée et ayant montré de bonnes performances sur le cas sans standardisation. Cette approche semble donc réaliste pour le nouveau problème étudié.

Mais que peut réellement apporter une standardisation partielle ? Cette question est l'enjeu des tests numériques à venir. En effet, le problème d'optimisation est plus complexe à traiter et le gain obtenu par cet assouplissement des contraintes n'est pas garanti. Il faudra donc voir dans quelles configurations de coûts il y a un intérêt ou pas.

L'approche modulaire présentée ici considère implicitement une nomenclature en râteau, c'est-à-dire une nomenclature où le produit fini est composé directement d'un ensemble de modules indépendants. Une piste qui serait intéressante à suivre est de traiter le problème avec des nomenclatures de profondeur supérieure à 1, ou autrement dit, des nomenclatures où les modules peuvent être eux eux-mêmes des assemblages de modules plus petits, avec possibilité de dédier des sites à l'assemblage de « petits » modules, et d'autres à l'assemble de « gros » modules (voire de certains produits finis) à partir des petits modules.

8 RÉFÉRENCES

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

Agard, B., 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), pp. 389-401.

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

Dahmus, J.B., Gonzalez-Zugasti, J.P., Otto, K., (2001) Modular product architecture. *Design studies*, 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.

El Hadj Khalaf, R., Agard, B., Penz, B., (2008a) 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

El Hadj Khalaf, R., Agard, B., Penz, B., (2008b) Product and supply chain design using a taboo search. soumis à publication.

El Hadj Khalaf, R., Agard, B., Penz, B., (2008c) Greedy heuristics for determining a product family bill of materials. *38th International Conference on Computers and Industrial Engineering*, Beijing, China.

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

Gupta, S., Krishnan, V., (1998) Product familly-based assembly sequence design methodology. *IIE Transaction*, 30, pp. 933-945.

He, D.W., Kusiak, A., (1997) Design of assembly systems for modular products. *IEEE Transactions on Robotics and Automation*, 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, 28(1), pp. 66-77.

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

Kota, S., Sethuraman, K., Miller, R., (2000) A metric for evaluating design commonality in product families. *Journal of Mechanical Design*, 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*, 169, pp. 1030-1047.

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

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

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

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

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

Pine, B.J., (1993) 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*, 48(1), pp. 1-19.