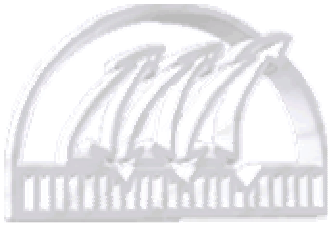
N° d'Ordre : **03/2004-M/IN**

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université des Sciences et de la Technologie
Houari Boumediene
« USTHB »



Faculté d'Electronique et d'Informatique
Département d'Informatique

Mémoire présenté pour l'obtention du diplôme
de **Magistère en Informatique**

Spécialité : **Intelligence Artificielle et Base des Données Avancées**

Par : Mr. **ABOUMSABAH Slimane**

Sujet :

La Résolution de Problème de Bin Packing Rectangulaire avec Contraintes En Deux Etapes Génétiques Hybrides

Soutenu publiquement le 01 juillet 2004, devant le jury suivant:

Mr. **S. LARABI,** Maitre de conférences, USTHB.	Président
Mr. **A. R. BABA-ALI,** Maitre de conférences, USTHB.	Directeur de thèse
Mme. **N. BENSAOU,** Maitre de conférences, USTHB.	Examinatrice
Mr. **S. BOUROUBI,** Dr. d'Etat, FM/USTHB.	Examinateur

Résumé.

Le problème de Bin Packing se retrouve dans plusieurs domaines d'application, essentiellement dans l'industrie de : tôle, bois, verre, papier etc. Dans cette thèse on s'intéresse au problème de découpe rectangulaire, avec la prise en charge de la contrainte de découpe de bout-en-bout.

L'application des algorithmes génétiques connaît des limites pour la résolution du problème de bin packing de grande taille. Pour résoudre ce problème nous proposons une méthode originale qui consiste à subdiviser le problème initial en deux sous problèmes. La première partie tente d'appliquer un algorithme génétique hybride, basé sur l'ordre d'apparition des pièces, pour agencer les pièces sur des niveaux dans une bande infinie en appliquant une heuristique puissante (BLF2G). La deuxième étape utilise les résultats de la première, à savoir les niveaux, et tente de les projeter sur les plaques en appliquant un deuxième algorithme génétique hybride. De plus nous avons proposé des améliorations à l'algorithme génétique classique pour améliorer les résultats. Les résultats sont comparés avec d'autres méthodes heuristiques sur des exemples aléatoires et réels.

Mots Clés : Bin Packing deux dimensionnel, découpe orthogonale, optimisation combinatoire, heuristique, algorithme génétique hybride.

Abstract.

The problem of Bin Packing is met in several domains of application, essentially in the industry of: sheet metal, wood, glass, paper etc. In this thesis we are interested in the problem of cuts up oblong, with the hold in charge of the constraint of cuts up tip - in - tip.

The application of genetic algorithms knows limits for the resolution of the bin-packing problem of large size. To solve this problem we propose an original method that consists in subdividing the initial problem in two sub problems. The first part attempts to apply a hybrid genetic algorithm, based on the order of piece apparition, to arrange pieces on levels in an infinite strip while applying a powerful heuristic (BLF2G). The second stage uses results of the first, to know levels, and try to project them on plates by applying a second hybrid genetic algorithm. Besides we proposed improvements to the classic genetic algorithm to improve results. Results are compared with other heuristic methods on uncertain and real examples.

Key words: Bin packing two dimensional, cut up orthogonal, optimization Combinatorial, heuristic, hybrid genetic algorithm.

Dédicace

Je dédie ce modeste travail à :

- mes parents
- ma femme et mon fils « yaser »
- ma sœur,
- mes frères,
- ma grande famille

Qu'ils trouvent tous en ce mots l'expression du profond amour que je leurs porte.

Remerciement.

Je remercie vivement Mr. S. LARABI, qui me fait l'honneur et le plaisir de présider ce jury.

Je remercie vivement Mr. Mahmoud SALIM le conseiller Culturel auprès de l'Ambassade de Palestine accréditée en Algérie pour son soutien et ces encouragements tout le long de ce travail.

J'exprime ma vive gratitude à mon directeur de thèse Mr. A. R. BABA-ALI, Maître de conférence à l'USTHB pour la confiance qu'il ma temoigné en me confiant ce travail. Je le remercie vivement pour son soutien et ses conseils qu'il ma constamment prodigués.

Je tiens à remercier Dr. Eva HOPPER docteur à l'université de Wales pour ses encouragements et pour son aide documentaire.

Toute ma reconnaissance et tous mes remerciements s'adressent à toutes les personnes ayant accepté de juger ce travail.

Préface.

Le problème de Bin Packing 2D se retrouve dans plusieurs domaines d'applications. Dans l'industrie, le problème d'agencement des pièces rectangulaires dans d'autres pièces plus larges, se présente dans plusieurs applications, à savoir la fabrication des armoires, où on cherche à couper des pièces rectangulaires à partir de la matière première tout en minimisant les chutes; dans la mise en page des journaux et des sites webs il s'agit d'agencer les articles et les photos sur les pages…

Dans toutes ces applications les objets manipulés ont une forme rectangulaire et on a comme objectif d'agencer ces items dans un nombre minimal d'objets, de forme rectangulaire également. Ce problème est connu dans la littérature sous le nom de « 2D Bin Packing » (2BP) ; dans un autre contexte ce problème peut être vu en utilisant des rouleaux (de papiers, de tissus, de tôle, …) dans ce cas il s'agit du problème « 2D Strip Packing » (2SP).

Ce problème est connu par son appartenance à la classe NP Complet, aussi généralement les travaux réalisés dans ce domaine sont basés sur des approches approximatives dites heuristiques. Récemment les algorithmes génétiques (AGs) ont été de plus en plus utilisés pour la résolution de ce problème[FAH93], [Hop98], [Rai99], [Val2001].

Une vue commune trouvée dans la plupart des algorithmes génétiques développés pour la résolution de problème d'agencement est qu'ils comportent deux étapes de résolution. L'algorithme génétique explore l'espace de recherche pour générer des individus. Une deuxième étape non génétique est nécessaire pour évaluer la performance des individus en appliquant une heuristique de placement. L'algorithme génétique offre des solutions basées sur l'ordre d'apparition des pièces pour le procédé d'agencement. Le format de découpe exact est donné par la routine de placement [Hop99].

Un problème commun rencontré par les AGs appliqués au problème de placement, est le problème du temps de résolution qui devient de plus en plus grand pour la résolution des problèmes de placement de grand taille. Il réduit

l'efficacité des AGs d'où la nécessité de limiter l'exploration de l'espace de recherche, ce qui influe directement sur les résultats [Hop2000], [Val2001].

Pour résoudre ce problème nous proposons une méthode originale qui consiste à subdiviser le problème de placement en deux sous problèmes, avec l'introduction d'une notion d'exploitation des niveaux (intra-niveau & inter-niveau). Il s'agit donc d'appliquer un algorithme génétique pour l'agencement des pièces sur une bande infinie (2SP) tout en exploitant l'espace libre dans les niveaux. Pour cela nous développons une politique de placement puissante nommée BLF2G (optimisation intra-niveau), qui respecte les contraintes du problème. Ensuite, on procède à l'agencement de ces niveaux sur les boites (1BP) en appliquant un autre algorithme génétique (optimisation inter-niveaux).

Notre travail utilise les techniques d'optimisation basées sur les métha-heuristiques et les algorithmes stochastiques, qui sont d'actualité, et on se propose d'enrichir les méthodes de résolution classiques du problème pour améliorer les résultats.

En premier lieu on propose de guider l'algorithme génétique par des heuristiques basées sur le tri. Il s'agit d'introduire ces heuristiques qui apportent de la matière génétique de qualité à la population initiale. Cette amélioration trouve son utilité pour les problèmes de moyenne et grande taille où les méthodes heuristiques basées sur le tri donnent de meilleurs résultats par rapport aux algorithmes génétiques.

Notre deuxième amélioration consiste à procéder à l'agencement des pièces en largeur (i.e. on fixe la longueur de la bande et on laisse la largeur infinie). Cette amélioration donne souvent des résultats meilleurs que l'agencement classique en longueur.

La troisième amélioration tire son utilité du fait que le processus génétique converge vers un optimal local après un certain nombre de générations. Notre amélioration consiste donc à ré-initialiser la population courante en générant aléatoirement une nouvelle population et en gardant le

meilleur individu issu des générations précédentes. Cette amélioration d'ordre technique apporte quelque fois des gains.

Le plan de notre thèse est comme suit :

Le chapitre 1 expose le problème, décrit ses contraintes et critères liés, et définit sa complexité et sa formulation.

Dans le chapitre 2 nous exposons d'une manière exhaustive les méthodes approximatives susceptibles d'être appliquées à la résolution de notre problème.

Les algorithmes génétiques sont exposés dans le chapitre 3.

Le chapitre 4 est consacré à notre contribution à la résolution du problème.

La réalisation de la méthode proposée et les résultats obtenus sont exposés au chapitre 5.

Enfin une conclusion générale récapitule le travail, expose et compare les résultats, et envisage des perspectives.

Sommaire.

Chapitre I.

Introduction générale.

1. Introduction.

Le problème de placement est un problème d'optimisation dont l'objectif est de chercher à trouver un bon arrangement d'un ensemble d'objets dans des objets plus larges. L'objectif principal est de maximiser l'exploitation de la matière première, donc de minimiser les pertes. Ceci est important pour les industries à production massive où l'optimisation de la matière première joue un rôle important dans la réduction de coût de fabrication.

Le développement d'un algorithme pour la résolution d'un problème de placement industriel doit prendre en considération la complexité du problème déterminée par la forme des objets manipulés, et des contraintes liées (imposées par le système de production).

Dans notre cas en prend en considération un problème de découpe orthogonal en manipulant des plaques brutes rectangulaires pour générer des pièces de forme rectangulaire également. La matière utilisée peut être de la tôle et les machine de production sont typiquement des cisailles-guillotine (découpe de bout-en-bout). Pour que notre algorithme ne se limite pas à ce type de problème, et pour qu'il puisse être étendu à d'autres problèmes vérifiant les contraintes imposées, l'orientation des objets n'est pas autorisée. Une telle contrainte est vérifiée dans plusieurs travaux et laisse l'algorithme utilisable pour un vaste nombre d'applications, tel que le bois, le verre, le tissu, … où on trouve des motifs ou des décorations, et même dans la mise en page des journaux, revue, ou page web, où l'orientation est fixée.

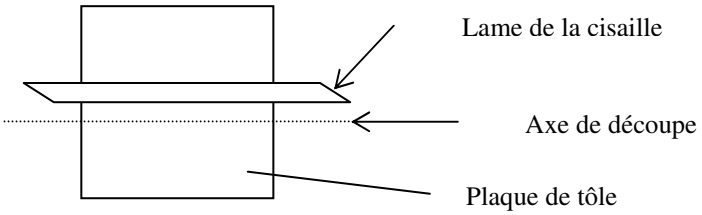
2. Contraintes du problème.

Selon les machines considérées à savoir les cisailles-guillotines et pour des raisons liées à la minimisation des coûts de fabrication, certaines contraintes sont imposées sur le procédé de découpe :

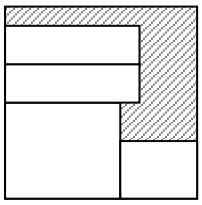
a) Les contraintes indispensables liées au système de production.

a1) La découpe de bout-en-bout.

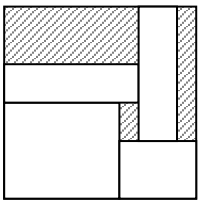
Les cisailles-guillotines ne permettent que la découpe de bout-en-bout



Découpe de bout-en-bout

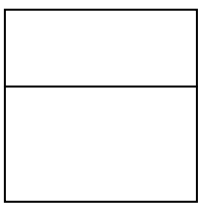


Forme réalisable par la cisaille

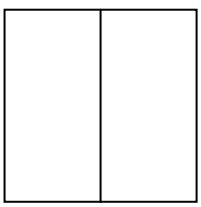


Forme irréalisable par la cisaille

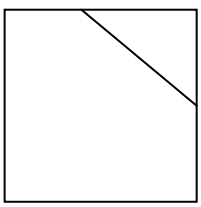
a2) La découpe Orthogonale.



Découpe en largeur
Réalisable



Découpe en longueur
Réalisable



Découpe diagonale
Irréalisable

b) Contraintes liées à la fabrication.

b1) Les pièces gardent leurs orientations d'origine, cette contrainte vise à assurer la faisabilité des formats de découpe réalisée sur des surfaces texturées ou décorer.

b2) L'agencement des pièces doit être optimal pour optimiser les chutes.

b3) La découpe doit se faire par série, pour améliorer le coût de fabrication (la découpe en série ne nécessite qu'un réglage pour découper toute la série).

3. Définition du problème. [Ghe92]

Un problème de découpe est un problème d'optimisation combinatoire qui consiste à trouver un agencement ou un placement optimal de certains éléments donnés, de valeurs ou coûts connus dans des domaines plus vastes, également donnés.

Cet agencement se traduit par l'optimisation de la fonction objective étudiée dépendant de la disposition adoptée et ce, tout en considérant les contraintes caractérisant le problème.

Un problème général de découpe est dit problème de Bin Packing (Bin=Boite, to Pack = Empaqueter).

4. Formulation du problème.

Etant données une liste L des pièces Pi (i=1,n) de forme rectangulaire et de dimension donnée.

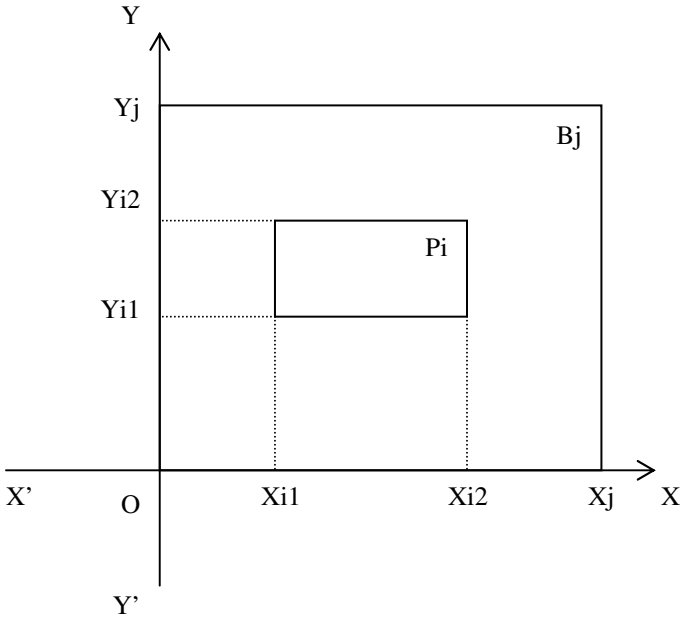
Etant donnée des boite Bj de forme rectangulaire et de dimension donnée, satisfaisant $\forall$Pi (i=1,n) Dim(Pi) <= Dim(Bj).

Objectif : Notre objectif et d'agencer les pièces Pi dans des boites Bj en réduisant le nombre des boites utilisées.

Contrainte de placements :

Soit le plan orthogonal (X'OX, Y'OY) borné par une boite Bj.

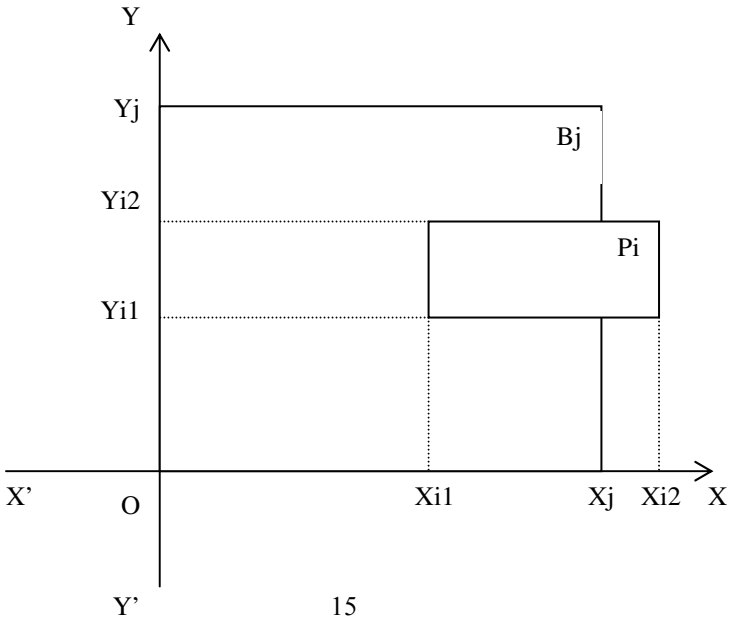
On définit le placement orthogonal (où un segment du rectangle Pi est orthogonal à l'une des axes du plan) d'une pièce Pi dans la boite Bj comme étant le quadruplé (X_{i1}, X_{i2}, Y_{i1}, Y_{i2}) issu de la projection des coins de Pi sur les axes du plan.



1) Contrainte de débordement

Une pièce ne doit pas excéder le domaine Bj

Xi1 >= 0 & Xi2 <= Xj & Yi1 >= 0 & Yi2 <= Yj



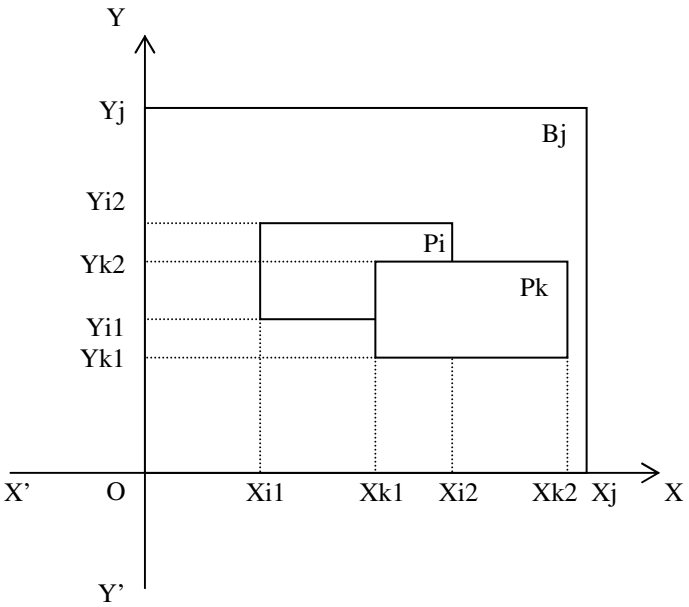
2) Contrainte de chevauchement.

Soit deux pièces P_i, P_k placées dans le domaine B_j (ou P_i et P_k satisfaisant la contrainte 1).

P_i s'intersecte avec P_k ssi l'une des inégalités suivantes et vérifiée :

$(Y_{i1} < Y_{k2} < Y_{i2}$ & $X_{i1} < X_{k1} < X_{i2})$ ou $(Y_{i1} < Y_{k2} < Y_{i2}$ & $X_{i1} < X_{k2} < X_{i2})$

ou $(Y_{i1} < Y_{k1} < Y_{i2}$ & $X_{i1} < X_{k1} < X_{i2})$ ou $(Y_{i1} < Y_{k1} < Y_{i2}$ & $X_{i1} < X_{k2} < X_{i2})$



5. Complexité du problème.

Notion de complexité [Dri92]

"De nos jours, l'informatique repose sur une notion fondamentale qui est la théorie de la complexité. Elle a été développée à partir des travaux de COOK et KARP. L'idée de base de cette notion est de mesurer théoriquement le temps d'exécution des algorithmes. Elle permet donc de distinguer les algorithmes de calcul efficaces des algorithmes dits «mauvais»."

Un algorithme est dit efficace si le nombre d'opérations nécessaires à la résolution du problème est borné par une fonction polynomiale.

On dit que f est O(g) s'il existe une constante C tel que f(n) <= Cg(n) pour tout n ∈ N.

Complexité de classe P.

C'est la classe des problèmes dont les algorithmes de résolution sont efficaces, où la fonction de résolution de problème f est O(g) et g est un polynôme en n.

Complexité de classe NP.

C'est la classe qui englobe les problèmes de décision qui sont résolus par des algorithmes non-déterministes en un temps polynomial, c.-à-d. la fonction de résolution du problème n'est pas polynomiale.

La complexité du problème de découpe [Gar76][Lau87] [Hue81][Pre83]

- Le problème de découpe dans sa forme général (Bin Packing) ou plus précisément le problème de décision est posé comme suit:

« Etant donné m boites de capacité C chacune, peut-on emballer un ensemble de n objets dans un nombre minimum de boites ». Il a été prouvé dans la théorie de la complexité que ce problème appartient à la classe des problèmes difficile ou problèmes NP-Complets. »

- Notre problème qui consiste à :

« Disposer un ensemble de n pièces rectangulaires dans des plaques rectangulaires en utilisant le minimum de celles-ci tout en considérant certaines contraintes », est sans doutes plus difficile que le problème général de Bin Packing, puisqu'on considère en plus des contraintes propres.

La recherche de la solution optimale en un temps polynomial est par conséquent difficile pour ce problème, car il n'existe pas à jusqu'à présent un algorithme performant pour le résoudre efficacement, et tous les travaux menés actuellement se basent sur des techniques approximatives «near optimum techniques ».

6. Nouvelle formulation du problème.

Le problème de découpe est subdivisé en deux sous problèmes partant des données brutes pour engendrer des résultats intermédiaires d'agencement des pièces sur la bande infinie. Ceci engendre des niveaux, de telle sorte que la hauteur d'un niveau soit inférieure à la longueur des boites, puis on procède à la projection de ces niveaux sur les boites dans la deuxième étape pour avoir le résultat final.

Dans la première phase on envisage d'appliquer un algorithme génétique hybridé par une heuristique adéquate pour la minimisation des chutes (i.e. réduire la hauteur de la bande).

Dans la deuxième phase on cherche à appliquer un autre algorithme génétique hybridé par une heuristique adéquate pour la minimisation des chutes (i.e. réduire le nombre des boites utilisées).

Chapitre II

Etude bibliographique.

I. Introduction.

Selon la nouvelle définition du problème, nous avons orienté notre recherche bibliographique dans deux sens : le premier est celui des heuristiques destinées à être hybridées aux algorithmes génétiques. Le deuxième concerne la recherche bibliographique sur les algorithmes génétiques.

II. Travaux basés sur les heuristiques.

[Msa95] présente trois heuristiques extraites de l'article [Ber87], qui satisfaisant les contraintes intrinsèques (indispensable) de système (il s'agit des contraintes de découpe de bout-en-bout, et de découpe orthogonale) qui utilisent une bande infinie.

Hypothèses :

La liste L est triée selon la décroissance des longueurs.

La bande est subdivisée en niveaux (fig. 1).

On définit les niveaux par induction :

- Le 1er niveau est la bordure inférieure de la bande, de hauteur égale à 0.
- Le niveau i dans la bande est défini par le segment horizontal de hauteur égale à la somme de la hauteur de niveau i-1 et de la longueur de la 1ère pièce placée dans ce niveau (la 1ère pièce est celle de longueur la plus élevée du fait que la liste des pièces est triée par décroissance de longueur).

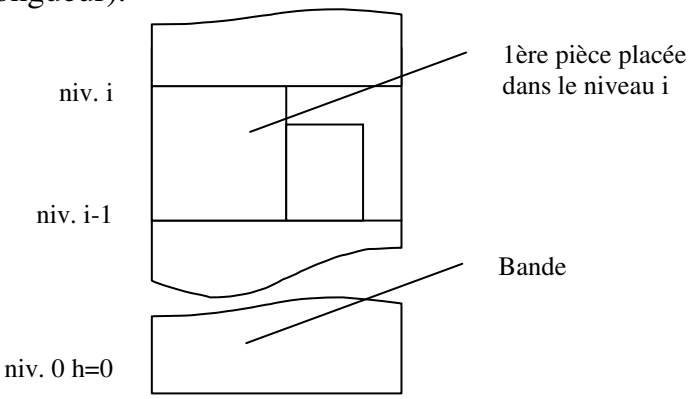


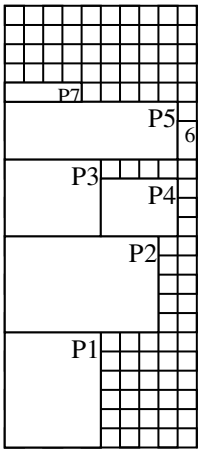
Fig. 1 : La bande est subdivisée en niveaux

- Le placement d'une pièce dans un niveau est orthogonal et le plus bas à gauche (Politique BL « Bottom Left ») .

- On complète la définition de chaque heuristique en appliquant l'algorithme approprié pour l'empaquettement des 7 pièces suivantes : P1(5x6), P2(8x5), P3(5x4), P4(4x3), P5(9x3), P6(1x2), P7(4x1), sur la bande de largeur 10 comme exemple.

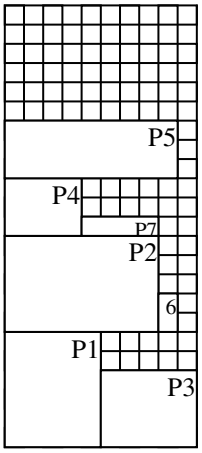
II.1. Finite Next Fit (FNF).

Autrement appelée « Next Fit Decreasing Height » NFDH. Cette heuristique est coûteuse en espace mais elle est rapide. Il s'agit de tenter d'agencer les pièces sur le niveau courant sans prendre en considération les résidus dans les niveaux précédents. Elle est de complexité linéaire en temps O(n).



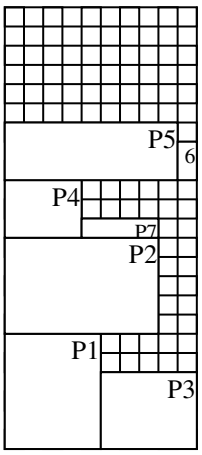
II.2. Finite First Fit (FFF).

Aussi appelée « First Fit Decreasing Height » FFDH. La politique appliquée ici consiste à placer la pièce courante dans le premier emplacement adéquat en se basant sur la politique BL. La complexité de cet algorithme est $O(n^2)$.



II.3. Finite Best Strip (FBS).

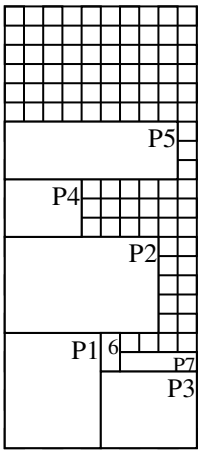
Appelée « Best Fit Decreasing Height » BFDH. Il s'agit d'agencer la liste triée des pièces dans l'emplacement le plus adéquat (qui offre le plus petit résidu). La complexité de cet algorithme est O(n log n).



[Hop99] : Présente deux routines de placement hybridées avec un algorithme génétique, il s'agit de la routine BL et d'une amélioration BLF qui vise à exploiter les résidus entre les niveaux. La liste des pièces n'est pas triée. L'ordre d'apparition des pièces, qui est l'élément déterministe de la qualité du placement, est laissé à l'algorithme génétique. Selon notre approche la politique BL est identique à la politique FFF, sauf que la liste n'est pas triée.

II.4. Bottom Left Fill.

Contrairement à la politique BL, cette politique tente d'exploiter les résidus entre les niveaux, L'application de cette heuristique sur une liste triée donne le résultat suivant:

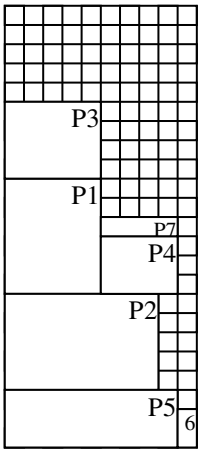


[Val01] : Regroupe les heuristiques dans deux catégories la première est celle des algorithmes orientés niveau dont il présente deux algorithmes NFDH et FFDH qui sont semblable resp. à FNF et FFF présenté ultérieurement.

La deuxième catégorie est définie par les algorithmes qui divisent.

II.4. Split Algorithme(SA) (Algorithme qui divise).

Les pièces sont orientées par décroissance de largeur, quand une pièce est placée dans la boite, elle sera subdivisée en deux par une ligne verticale passant par la largeur de la pièce placée, et ainsi de suite pour les autres pièces jusqu'à ce que la boite est subdivisée en sous bandes. Les autres pièces seront placées dans les sous bandes en appliquant la catégorie des algorithmes orientés niveau.

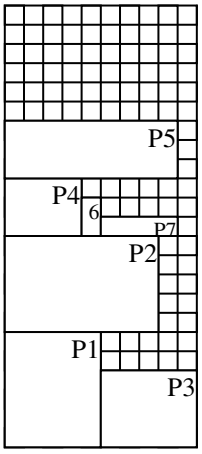


[Cof96],[Cof99] : Présente plus d'heuristiques ; en plus des routines Next Fit et First Fit et Best Fit déjà exposées, il expose quatre nouvelles heuristiques à savoir Worst Fit (WF), Almost Worst Fit (AWF), Any Fit (AF), et Almost Any Fit (AAF). Une nouvelle technique est référencée dans leurs travaux, il s'agit des algorithmes k-bornés en espace. Elle vise à limiter l'espace de recherche en appliquant une routine de placement à choix (tel que FF, BF WF..). Cette technique montre ces capacités pour les problèmes de grande taille, et elle sera présentée ultérieurement.

Notons que notre problème se place dans la catégorie des algorithmes OffLine, et non pas OnLine du fait que la liste des pièces est connue d'avance.

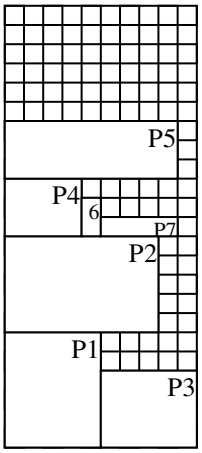
II.5. Worst Fit Algorithme (WF).

Contrairement à l'algorithme BF qui place la prochaine pièce dans l'emplacement le plus adéquat (le plus petit résidu), l'algorithme WF tente de placer la prochaine pièce dans l'emplacement le plus mauvais (le plus grand résidu).



II.6. Almost Worst Fit Algorithme (AWF).

L'algorithme AWF applique la même politique que WF mais il tente de placer la prochaine pièce dans le second mauvais emplacement si cet emplacement n'est pas suffisant (n'existe pas dans les niveaux ouverts) pour placer la pièce on revient à appliquer l'algorithme WF.



II.7. Any Fit Algorithme (AF).

C'est la classe d'algorithmes qui regroupe les heuristiques à choix qui parcourent les boites non vides (les niveaux dans notre cas) pour choisir l'emplacement dans lequel la pièce sera placée.

II.8. Almost Any Fit (AAF).

C'est la même classe que AF sauf que le placement des pièces ne s'effectue pas dans le 1er emplacement qui correspond à l'heuristique appliquée mais dans le 2ème emplacement, à moins que le 1er emplacement soit le seul qui correspond à la pièce.

II.9. Les algorithmes k-borné en espace.

On dit qu'un algorithme est k-borné en espace, si pour chaque pièce à placer, on restreint la recherche de l'emplacement qui correspond à la pièce selon l'heuristique appliqueé sur k objets (niveaux dans notre cas) et non pas sur tous les objets (niveaux) existants. On définit un niveau ouvert par le niveau qui a au moins une pièce. Un niveau fermé ne peut pas recevoir d'autres pièces.

L'heuristique NF utilise un algorithme 1-borné, les heuristiques FF, BF, et WF utilisent un algorithme non-borné.

Selon les règles de placement des pièces et selon la règle de fermeture des niveaux, on peut définir quatre algorithmes k-borné simples.

On peut placer la pièce dans le niveau le plus bas (politique First Fit) ou dans le niveau le mieux adapté (politique Best Fit), si une nouvelle pièce ne peut être placée dans aucun niveau on la place dans un nouveau niveau. Si e nombre des niveaux dépasse la borne k on doit fermer un niveau. Dans ce cas on peut appliquer deux règles pour choisir le niveau à fermer, à savoir le niveau le plus bas (First) ou le niveau le mieux saturé (Best). Ces quatre algorithmes peuvent être nommés comme suit : AFFk, AFBk, ABFk, et ABB, où A signifie

Algorithme, la deuxième lettre signifie la politique de placement des pièces, la troisième lettre dénote la politique appliquée pour le choix de niveau à fermer, et k dénote la borne qui détermine le nombre de niveaux ouverts.

II.10. Exact Fit Algorithm (pour le problème de bin packing à une dimension) [Dja98]

C'est une approche heuristique riche et complexe, proposée pour la résolution du problème de bin packing à une dimension. Son principe de résolution, inspiré de son nom, tente de remplir la boite courante en fonction de 3 possibilités : La première vise à placer la pièce qui a une taille exactement égale à celle de la boite, sinon il tente de placer deux pièces où la somme de leurs tailles est égale à celle de la boite, sinon il tente de placer trois pièces où la somme de leurs tailles est égale à celle de la boite. Si ces tentatives échouent, l'algorithme réduit la taille de la boite par une unité, et refait les trois tentatives. Ce processus est refait jusqu'à la dernière pièce. Notons que cette approche ne se base pas sur l'ordre d'apparition des pièces.

L'expérimentation montre que cette technique consomme nettement moins de temps par rapport aux algorithmes génétiques, et donne des résultats comparables, et pour certain cas, moins efficaces que ceux des algorithmes génétiques.

III. Travaux basés sur les algorithmes génétiques.

Les algorithmes génétiques ont récemment été appliqués au problème industriel de placement [Hop99]. Ils ont été appliqués avec succès dans un vaste nombre d'applications industrielles. Ils utilisent des procédures de recherche et d'optimisation appliquée d'une manière similaire au processus d'évolution observé dans la nature.

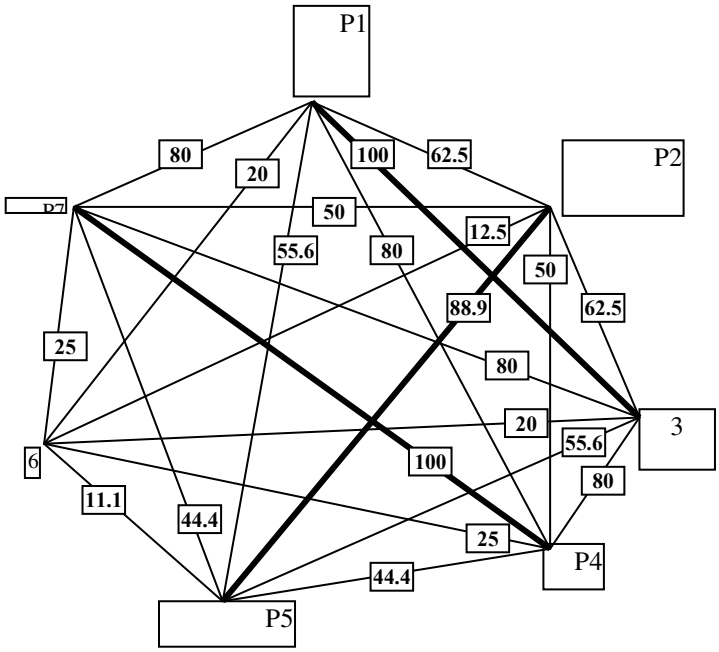
La recherche est guidée par la survie des meilleurs individus, et elle est basée sur l'extraction des caractéristiques des bons individus de la population et les combine entre eux pour générer la prochaine génération. La qualité de

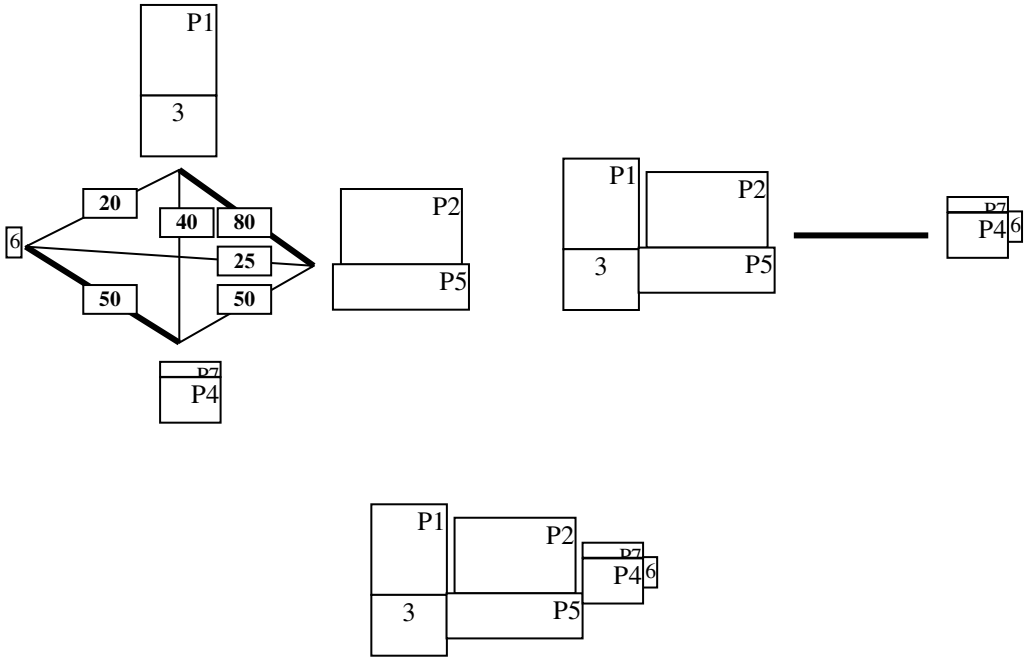
chaque solution est évaluée et les meilleurs individus sont sélectionnés pour le processus de reproduction. L'itération de ce processus sur un certain nombre de générations donne à la fin des solutions presque-optimale et dans certains cas optimales.

Dans cette partie de notre étude bibliographique nous étudions la recherche sur les algorithmes génétiques pour le problème de Bin Packing, puis on adaptera ces approches génétiques à notre approche de résolution.

[Sch98] : propose d'agencer les pièces en utilisant un graphe où les nœuds sont les pièces qui sont liées entre eux par des arrêtes où chaque arrête reçoit une valeur de ressemblance des deux pièces qui constituent l'extrémité de l'arrête.

Selon ces valeurs les pièces sont groupées deux à deux quand la valeur de l'arrête est maximale. Ces regroupements sont réutilisés à nouveau de la même manière pour former un graphe connexe, puis regroupés à nouveau et ainsi de suite jusqu'à la fin (i.e. le regroupement de toutes les pièces).





Enfin on aura un nouveau graphe dont les feuilles sont les pièces et les nœuds sont les regroupements. Un algorithme génétique est appliqué à ce graphe.

Cette approche est très coûteuse en espace et en temps et ne peut pas être appliquée à un problème de grand taille.

[Val01] : Montre dans son article que l'utilisation des algorithmes génétiques pour la résolution du problème d'agencement orthogonal donne des résultats optimaux par rapport aux méthodes heuristique. Cependant, pour les problèmes de grande taille le temps d'exécution devient de plus en plus grand de l'ordre d'heures et même de jours, donc un test d'arrêt doit être appliqué avant d'atteindre les résultats souhaités. Ensuite, il montre que pour les problèmes de moins de 100 pièces les algorithmes génétiques sont efficaces, au-delà de 100 pièces les approches heuristiques sont les mieux adaptées.

[Hop99] : Cet article présente deux algorithmes génétiques hybrides basés sur une approche génétique commune qui explore l'espace de recherche. La codification utilisée par l'algorithme génétique appliqué au problème de placement est basée sur l'ordre dans le quel les pièces rectangulaires sont placées dans les objets. L'emplacement exact des pièces sur les objets est déterminé par la routine de placement, qui réalise l'agencement effectif des pièces. Deux routines de placements sont combinées avec la séquence génétique des pièces.

La routine de placement :

La routine de placement BL, où chaque pièce est placée le plus bas dans l'objet puis le plus à gauche, présente pour certains cas de figure des inconvénients où des résidus vides sont présents dans l'objet. Une amélioration est proposée, elle consiste à placer les pièces dans ces résidus (i.e. le plus bas niveau possible) en appliquant la routine BLF (le plus bas à gauche possible).

L'algorithme génétique :

La qualité du placement utilisant les deux routines précédentes dépend essentiellement de la séquence dans la quelle les pièces sont présentées. Du fait que le nombre de combinaisons est énorme pour être exploré exhaustivement dans un temps raisonnable, l'utilisation d'un algorithme génétique comme stratégie de recherche est plus pratique.

[Hop00] : Eva Hopper dans sa thèse de doctorat confirme que pour les problèmes de petite et moyenne taille les métha-heuristiques, dont les algorithmes génétiques font partie, sont les mieux adaptées et offrent des résultats meilleurs en un temps raisonnable. Mais pour les problèmes de grande taille les techniques d'agencement standard offrent des résultats meilleurs en un temps raisonnable.

Conclusion.

La puissance des algorithmes génétiques, en matière d'exploration de l'espace de recherche, est réduite pour les problèmes de grand taille, alors que les heuristiques offrent des résultats satisfaisants quelque soit la taille de problème.

Trouver un compromis entre la puissance des algorithmes génétiques, et la simplicité des heuristiques de placement, fera l'objet de notre contribution. Ceci sera exposé après une présentation des algorithmes génétiques.

Chapitre III

Les Algorithmes génétiques.

I. Définition.

Les algorithmes génétiques (AG) représentent une famille assez riche et très intéressante d'algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique[Ler95]. Ils font partie des méthodes d'optimisation qui cherchent parmi un ensemble de solutions ou espace de recherche S la ou les solutions s* qui vont maximiser un critère mathématique (ou plus généralement un critère calculable) f de S dans R appelé fonction d'évaluation. Les éléments de l'espace de recherche S sont décrits par des caractéristiques nommés gènes. Un AG utilise une population d'individus $p(t)=(s1,\ldots,si,\ldots sN) \in S$ à la génération t qui réalise un échantillonnage de S et qui va servir à calculer la génération t+1 et à guider la recherche vers la ou les solutions optimales s*.

II. Le fonctionnement des AGs.

L'AG génère aléatoirement une population initiale d'individus, Une codification est attribuée aux individus, que se soit une codification binaire ou réelle (nommée chromosome). A chaque individu est affecté une qualité (fitness) calculée par le biais d'une fonction d'évaluation propre au problème. Une sélection est effectuée sur cette population, pour garder les meilleurs individus. Ces individus passent par une phase de croisement, puis de mutation. Ces deux phases génétiques modifient les individus, le meilleur individu risque d'être perdu. Un test est nécessaire après ces deux phases, pour vérifier si le meilleur individu de la population précédente est perdu. S'il est perdu, on peut remplacer le mauvais individu de cette génération par le meilleur individu de la génération précédente. Un certain nombre de générations est réalisé (en général 1000) pour extraire le meilleur individu.

II.1. La population initiale.

La population initiale est un ensemble fini d'individus (en général 100). Elle sert à lancer le processus génétique. La génération de cette population peut être :

- Faite d'une manière aléatoire, elle offre une diversification des solutions.
- Guidée par des heuristiques, elle offre des meilleurs résultats au départ mais elle risque de converger vers une solution optimale locale.
- Faite par un mélange des solutions aléatoire et des solutions guidées par des heuristiques.

II.2. Le codage des individus.

Un algorithme génétique de base utilise une représentation binaire des individus. Chaque individu représente une solution possible au problème. Cette représentation est indépendante du problème. Une phase de conversion et de reconversion est nécessaire pour évaluer chaque individu, cette phase est répétée pour les individus à chaque génération dans l'algorithme, par conséquent elle est très coûteuse en temps de calcul.

Les AGs utilisent aussi une codification réelle pour représenter les individus, cette codification dépend essentiellement du problème traité, elle rend facile l'évaluation des individus.

Pour chaque type de codification on associe une représentation, il existe deux types de représentations, la représentation basée sur des valeurs et la représentation basée sur l'ordre. Dans la première l'algorithme génétique génère des valeurs qui doivent être évaluées par la fonction d'évaluation, tandis que dans le second l'algorithme génère des séquences de nombres, où

l ‘évaluation de chaque séquence se fait selon l’ordre dans lequel les nombres sont présentés.

II.3. L’évaluation des individus.

L’évaluation des individus est souvent représentée par une fonction mathématique appliquée au chromosome, si la codification binaire est utilisé une phase de conversion est nécessaire. L’évaluation sert à déterminer la qualité (fitness) de chaque individu. Cette fitness est utilisée ultérieurement par les opérateurs génétiques de sélection, de croisement et de mutation.

Dans certain cas l’évaluation est représentée par un algorithme ou même une heuristique, en vu d’extraire la fitness de l’individu. On parle alors d’un algorithme génétique hybride.

II.4. Les opérateurs génétiques.

O1) La sélection (reproduction).

L’opérateur de reproduction est inspiré par le mécanisme de la sélection naturelle, il consiste à dupliquer chaque individu de la population proportionnellement à sa valeur d’évaluation. Chaque élément est tiré avec une probabilité liée à sa fitness, interprétée aussi comme sa probabilité de survie. Ainsi, les individus ayant une fitness élevée ont plus de chances d’être sélectionnés pour la génération suivante. Cet opérateur peut être mis en œuvre sous plusieurs formes :

- **La roue de loterie biaisée :**

Autrement appelé la roulette de casino, où chaque individu, occupe une portion dans la roue proportionnelle à sa valeur d’évaluation relative. Ainsi, un individu i aura une probabilité Pi d’être sélectionné qui est définie comme suit :

$$P_i = f(i) \Big/ \sum_{j=1}^{N} f(j)$$

L'inconvénient majeur de ce type de reproduction vient du fait qu'elle peut favoriser la domination d'un super-individu, qui engendre une perte de diversité, d'où le meilleur individu est local.

- **Une autre variante de la roue de loterie :**

Dans cette variante la probabilité de sélection est calculée à la base de fitness moyenne est non pas sur la somme des fitness. Elle est définit comme

$$P_i = f(i) \Big/ \left(\sum_{j=1}^{N} f(j) \Big/ N \right)$$

suit :

Ce type de sélection favorise aussi les meilleurs individus, et les faibles seront rarement sélectionnés.

- **N/2 élitisme :**

Il s'agit ici de trier les individus selon leurs fitness, et seule la moitié supérieure de la population correspondant aux meilleurs individus est sélectionnée. Cette méthode élimine la moitié faible de la population, d'où une perte de diversité qui est le point fort de l'algorithme génétique. Un mauvais individu peut contenir dans son gène de bon caractéristiques.

- **Par tournoi :**

Elle subdivise la méthode précédente en sous ensemble, le principe est de regrouper aléatoirement les individus de la population en sous ensemble et d'appliquer la méthode «N/2 élitisme » pour chaque sous ensemble.

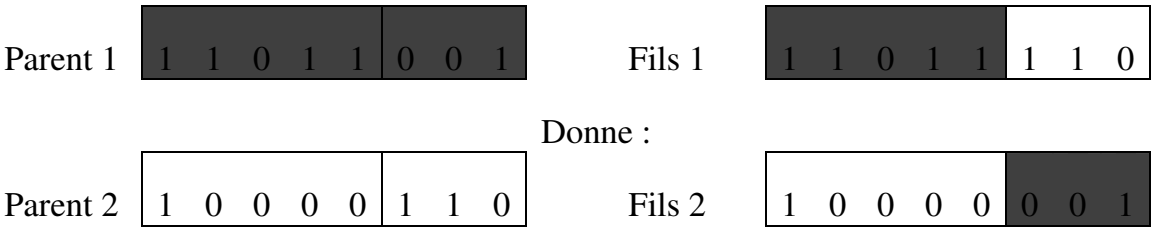
O2) Le croisement.

L'opérateur de croisement est appliqué sur deux individus «parents » choisis aléatoirement, le mécanisme de croisement est inspiré directement du processus de multiplication des chromosomes, il s'agit donc de générer deux

nouveaux individus « enfants » qui héritent des portions génétiques mélangés de leurs parents. En général il existe trois types de croisement.

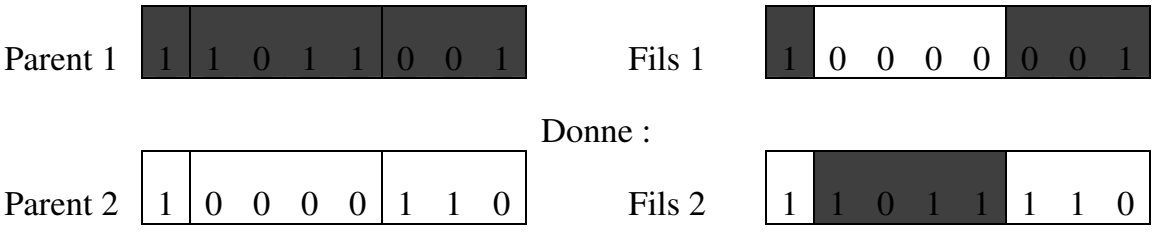
- **Croisement à un site :**

Un site de croisement est tiré aléatoirement sur l'individu, puis les segments finaux des deux parents sont échangés autour de ce site.



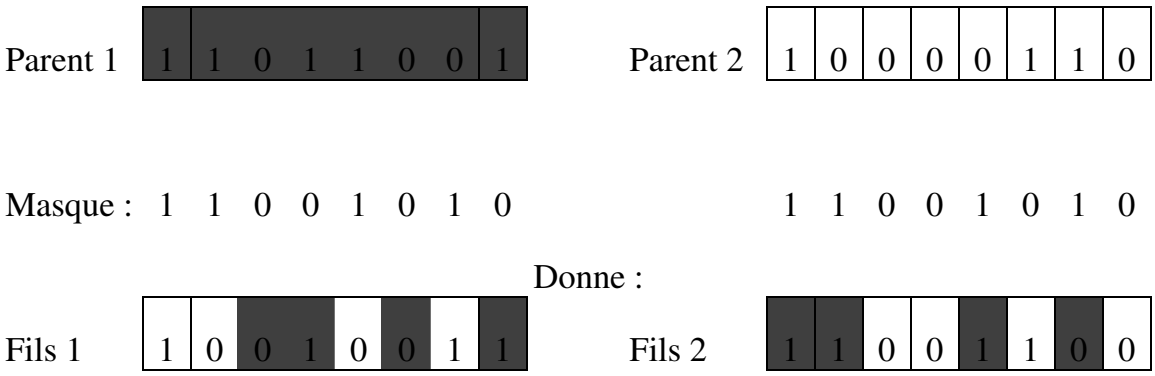
- **Croisement bipoint :**

Deux points de croisement sont tirés aléatoirement sur l'individu, la portion de gène située entre ces deux points est échangée pour les deux enfants.



- **Croisement uniforme :**

Ce croisement peut être vu comme un croisement multi-point. On utilise un masque de croisement binaire, de même taille que le génotype. Un « 0 » à la $n^{ème}$ position garde le gène du parent1, est un « 1 » déclenche un changement sur cette position avec le gène du parent2.



- **Autres types de croisement :**

Ces trois opérateurs de croisement sont les plus utilisés dans la littérature des AGs. Il en existe bien d'autres croisements spécialisés liés à la sémantique du problème traité et/ou à la structure du codage considéré.

O3) La mutation.

L'opérateur de mutation consiste à modifier de manière aléatoire, avec une probabilité généralement faible, la valeur d'un ou de plusieurs gènes d'un individu. La valeur du gène muté est remplacée par une autre appartenant au même alphabet.

Techniquement, la combinaison de la reproduction et du croisement est suffisante pour assurer l'évolution de la population, mais il peut arriver que certaines informations essentielles disparaissent de la population, et que celle-ci converge vers un optimum local. Le rôle de la mutation est donc d'assurer une diversité du matériel génétique des individus et d'assurer une exploration globale dans l'espace de recherche.

III. Le théorème fondamental des AGs.

Dans un premier temps, les AGs ont été largement exploités sans qu'aucune étude approfondie ne soit venue en valider le fonctionnement. Afin de bien comprendre le processus de fond qui guide l'algorithme génétique, introduisons la notion de schéma*.

III.1. Les schémata.

Une bonne partie des études menées sur les AGs repose sur la similarité entre les individus d'une population. La formalisation de cette recherche de similarité entre les individus utilise un méta-symbole, qui décrit un sous-ensemble particulier de l'espace de recherche, appelé schéma.

Définition 1. [Ler95]

Soit $X = X_1 X_2 \ldots X_n$ une séquence de longueur n sur l'alphabet binaire {0, 1, *} où * est une indéterminée booléenne susceptible de prendre l'une des deux valeurs logiques 0 ou 1. Une chaîne binaire $b = b_1 b_2 \ldots b_n$ est dite instance du motif X si et seulement si

$\forall i \in \{1 \ldots n\}\ b_i = X_i$ ou $b_i \in \{0, 1\}$ si $X_i = *$

Définition 2. [Ler95]

L'ensemble des chaînes binaires de longueur n qui sont des instances d'un motif spécifié S de longueur n est appelé schéma.

Par exemple, le schéma S = 10*10*0 représente l'ensemble des individus suivants :

$S^1 = 1001000$, $S^2 = 1001010$, $S^3 = 1011000$, $S^4 = 1011010$

* Au pluriel, schémata.

Relativement à la notion de schéma, introduisons deux caractéristiques couramment utilisées dans la théorie des AG.

- L'*ordre* d'un schéma S, noté o(S), correspond au nombre de positions instanciées dans le schéma S.

- La *longueur de définition*, notée l(S), désigne la distance entre la première et la dernière position instanciée du schéma S.

Pour l'exemple cité ci-dessus S = 10*10*0, o(S) = 5, et l(S) = 7.

L'adaptation ou l'évaluation d'un schéma, [Mic96] est définie comme étant la moyenne des adaptations de ses instances. On définit formellement

$$f(S)=\left(\sum_{i=1}^{2l(S)-o(S)} f(S_i)\right)\Big/ 2^{l(S)-o(S)}$$

l'adaptation d'un schéma comme suit :

IV. L'algorithme génétique.

Un AG peut être résumé comme suit :

1) **Générer** aléatoirement et **Evaluer** une population initiale P(0) de N individus t <— 0.

2) **Sélectionner** N individus dans P(t) pour former une population Ps(t).

3) **Combiner** les individus de Ps(t) et **appliquer** les opérateurs génétiques de croisement et de mutation pour obtenir une population P(t+1).

4) **Evaluer** les individus de P(t+1), t <— t +1

5) **Aller** en 2) ou **arrêter** selon un critère d'arrêt (qualité, temps, nombre de génération, etc.….)

Chapitre IV

Notre Contribution.

I. Introduction.

Notre problème consiste à agencer une liste L de n pièces de forme rectangulaire sur des plaques également rectangulaires. Ce problème est reconnu par sa complexité NP-Complet, et il est improbable qu'il soit résolu par une méthod e exacte, aussi l'utilisation des méthodes presque optimales est indispensable. L'utilisation des heuristiques a montré ses capacités pour la résolution du problème de bin packing pour obtenir des résultats proches de l'optimal en un temps raisonnable.

Les algorithmes stochastiques, dont les algorithmes génétiques font partie, sont de plus en plus utilisés pour la résolution de problème de Bin Packing [Fuj93], [Hop99], [Rai99], [Val2001].

« Une approche commune trouvée dans la plupart des algorithmes génétiques développée pour la résolution de problème d'agencement est qu'ils opèrent en deux étapes de résolution. L'algorithme génétique explore l'espace de recherche pour générer les individus. Une deuxième étape non génétique est nécessaire pour évaluer la performance des individus en appliquant une heuristique de placement. L'algorithme génétique trouve des solutions basées sur l'ordre d'apparition des pièces pour être appliquées au procédé d'agencement. Le format de découpe exact est donné par la routine de placement »[Hop99].

Certains travaux basés sur les algorithmes génétiques [Hop99], [Hop00], [Val01] montrent et confirment que pour les problèmes de grandes tailles les algorithmes génétiques sont moins efficaces comparés aux méthodes heuristiques classiques. Ceci est dû au fait que l'algorithme génétique doit donner des résultats en un temps raisonnable, aussi l'application de l'algorithme génétique pour un nombre limité de générations, l'arrête avant qu'il n'atteigne ces objectifs. Ceci a pour conséquence de limiter l'exploration de l'espace de recherche, ce qui influe directement sur les résultats. De plus, l'algorithme génétique risque d'être piégé dans un minimum local.

Notre travail consiste à subdiviser le problème initial en deux sous-problèmes, le premier tente d'agencer la liste L de n pièces sur une bande infinie de largeur égale à la largeur des plaques (a savoir 2SP) (nommé dans notre cas pré-traitement) pour engendrer des résultats intermédiaires sous forme d'une liste L' de m niveaux. Le deuxième sous-problème (nommé emboîtement) projette le résultat du pré-traitement sur les plaques. Ces niveaux sont caractérisés par une largeur identique et de longueur variable (1BP). Un algorithme génétique hybride est utilisé dans les deux étapes, pour offrir une exploration efficace de l'espace de recherche.

Notons ici que la liste des pièces n'est pas triée selon la décroissance des longueurs. L'affectation des pièces aux niveaux est basée seulement sur la largeur disponible, si elle est suffisante la pièce sera placée dans le niveau, donc la hauteur de niveau est définie dynamiquement selon la hauteur de la plus longue pièce placée.

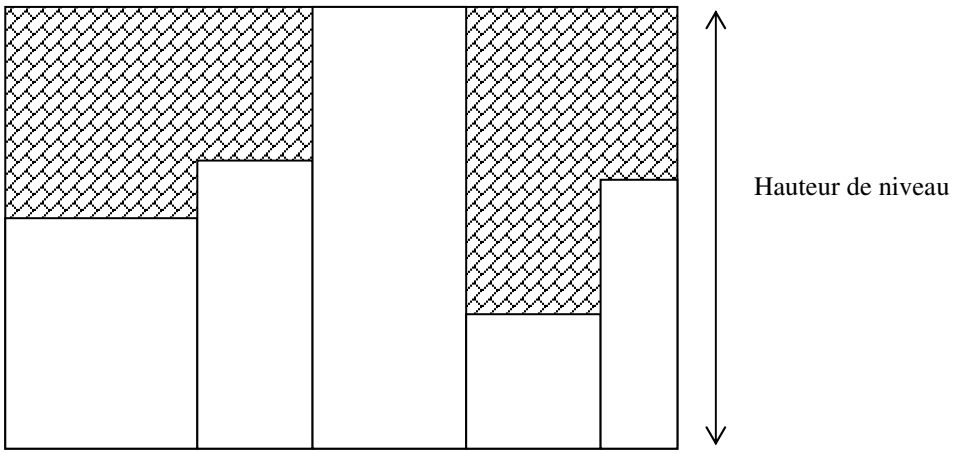


Figure 1: La hauteur du niveau égale à la longueur de la plus longue pièce placée dans le niveau.

II. L'algorithme génétique.

Dans la première étape on gère une liste de pièces et dans la deuxième étape on gère une liste de niveaux. L'algorithme génétique doit explorer un espace de recherche basé sur l'ordre dans lequel les pièces (niveaux) sont classé(e)s. Ce problème dans sa formulation est identique au problème de voyageur de commerce. L'application des algorithmes génétiques dans ce cas cherche à trouver le classement des villes à visiter (tournée), de façon à minimiser la charge de la distance totale parcourue.

II.1. Le codage des individus.

« Il y a un consensus dans la communauté des AGs pour dire que la représentation binaire des tournées n'est pas convenable pour le problème de voyageur de commerce »[Mic96].

« Une solution au problème de placement consiste en une séquence de nombre entier qui indiquent l'ordre dans lequel les rectangles sont placés dans les objets »[Hop99].

La représentation adoptée à notre problème est donc la représentation réelle basée sur l'ordre. Donc un chromosome 7426513 signifie que le premier objet (pièce ou niveau) qui doit être agencé est bien l'objet '7' suivie de l'objet '4' ainsi de suite.

II.2. L'évaluation des individus.

Il n'y a pas de fonction exacte qui nous permet de calculer la qualité (fitness) d'un individu. Il s'agit donc d'adopter une heuristique convenable au problème pour déduire la qualité de chaque individu. Rappelons qu'on procède à la résolution du problème en deux étapes (pré-traitement et emboîtement), aussi deux heuristiques sont nécessaires. Ces deux heuristiques seront abordées ultérieurement (voir paragraphe III.1 et III.2)..

II.3. Les opérateurs génétiques.

Pour cette codification spécifique des chromosomes (codification basée sur l'ordre), il faut utiliser des opérateurs génétiques spécifiques au problème.

O1) La sélection (reproduction).

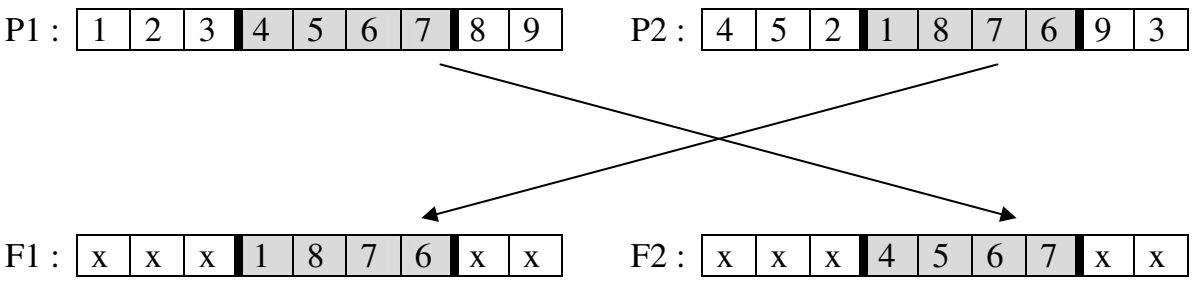
Du fait que la reproduction agit directement sur les individus, cette phase est indépendante du type de la représentation adoptée. Une reproduction classique peut être utilisée.

O2) Le croisement :

Le croisement «PMX» est le plus efficace en application aux problèmes basés sur l'ordre [Mic96] [Hop99]. Le croisement égalé partiel (PMX) proposé par [Gol89] est donc le mieux adapté à notre codification. Son principe est exposé ci-après :

Le PMX construit les progénitures on utilisant le croisement bipoint, par le biais du choix d'une sous-séquence délimitée par deux points choisis aléatoirement dans les deux parents. Ces deux sous-séquences sont projetées selon leur ordre d'apparition et leur position, aux fils.

Exemple :



(où x représente un indéfini de l'alphabet à ce moment)

on remplace les x par les valeurs qui ne posent pas de conflit avec la sous-séquence ajoutée (éviter les doublons) selon leur ordre d'apparition dans les deux parents, on aura donc :

F1 : | x | 2 | 3 | 1 | 8 | 7 | 6 | x | 9 | F2 : | x | x | 2 | 4 | 5 | 6 | 7 | 9 | 3 |

les x restants, ceux qui posent des conflits entre leurs valeurs dans les parents et la sous-séquence ajoutée. Ils sont remplacés par les valeurs trouvées à la même position dans l'autre sous-séquence. Donc '1' est remplacé par '4' et '8' par '5' dans le premier enfant. Et '4' par '1' et '5' par '8' dans le deuxième enfant, on aura donc :

F1 : | 4 | 2 | 3 | 1 | 8 | 7 | 6 | 5 | 9 | F2 : | 1 | 8 | 2 | 4 | 5 | 6 | 7 | 9 | 3 |

O3) La mutation.

Deux sites sont choisis aléatoirement dans l'individu, une mutation est effectuée en permutant les valeurs de deux sites entre eux.

Exemple :

P : | 4 | 2 | 3 | 1 | 8 | 7 | 6 | 5 | 9 | P : | 4 | 7 | 3 | 1 | 8 | 2 | 6 | 5 | 9 |

Ces opérateurs sont utilisés dans les deux sous-problèmes définis ci-dessus.

III. La routine de placement.

Selon les caractéristiques, totalement différentes, des données manipulées dans les deux sous-problèmes, une routine de placement adéquate est discutée et adaptée pour chaque cas. Rappelons que l'heuristique adaptée doit vérifier la contrainte de découpe de bout-en-bout et que l'orientation des pièces n'est pas prise en charge dans notre cas, en vue de vérifier la découpe sur de la matière texturée ou décorée.

III.1. Le pré-traitement.

Dans la phase de pré-traitement (la première étape) nous utilisons un agencement en lignes formant des niveaux, qui sont les plus utilisés[Lod02] pour vérifier la contrainte de découpe de bout-en-bout.

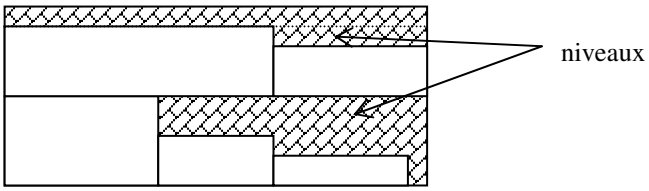


Figure 2 : **Agencement en lignes formant des niveaux.**

Le choix d'une routine de placement des pièces sur les niveaux est un point crucial. Il faut noter que cette routine doit être hybridée avec l'algorithme génétique pour évaluer la performance de chaque individu d'une population, et pour toutes les générations. Donc cette routine est sollicitée un très grand nombre de fois dans le pré-traitement. Une étude exhaustive à été effectuée dans le chapitre 2 sur les routines de placement.

La routine FNF n'a pas été retenue à cause de ces résultats nettement moins efficaces et à cause de sa politique abusive, en consommant des nouvelles plaques sans exploiter les résidus ultérieurs.

Les deux routines FBS et WF, appliquent une heuristique de choix respectivement le meilleur et le mauvais emplacement trouvé sur tous les emplacements possibles. Donc pour une pièce à placer, elles réalisent le choix de l'emplacement en balayant toute la bande. La pièce peut être placée dans un emplacement autre que le premier, et la pièce suivante aura la chance d'être placée sur le premier emplacement. On peut dire en quelque sorte que ces deux politiques altèrent l'ordre d'apparition des pièces, et elles nécessitent un temps supplémentaire pour effectuer le choix de l'emplacement. On voit bien que ces deux heuristiques sont pénalisantes en temps d'exécution. Par conséquent ces deux heuristiques n'ont pas été retenues.

La routine FFF (autrement appelée BL) tente de placer la pièce courante dans le premier emplacement disponible. Elle réalise un placement direct avec une exploitation des résidus ultérieurs. Donc elle est la plus convenable à notre cas. En plus son amélioration (BLF)[Hop99] offre encore plus d'avantages.

La classe des algorithmes k-bornés présentée au chapitre 2, peut être appliquée à l'heuristique BLF, il s'agit donc d'appliquer la politique BLF pour le placement des pièces, et la politique First pour la fermeture des niveaux. L'application des algorithmes k-bornés, trouve son utilité dans les problèmes de très grand taille, pour réduire le temps de calcul qui devient irraisonnable.

Donc la routine « Bottom Left » avec son amélioration BLF[Hop99] est utilisée pour l'agencement des pièces sur les niveaux. L'amélioration consiste à exploiter les résidus à l'intérieur des niveaux (intra-niveau) sans que cela influe sur la contrainte de découpe de bout-en-bout.

Notons que l'exploitation des résidus intra-niveau nécessite un temps de calcul supplémentaire. Pour cela on a défini trois nuances de la politique BLF, a savoir BLF0, BLF1, BLF2, définies ainsi :

- **BLF0 :** elle ne réalise aucune exploitation des résidus intra-niveau (identique à BL).

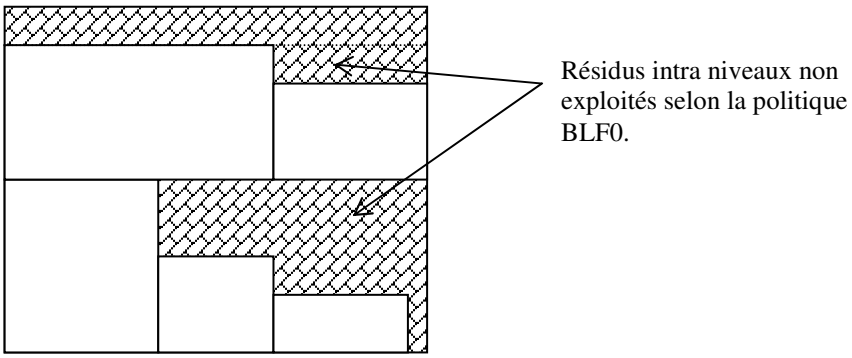


Figure 2 : Aucune exploitation des résidus selon la politique BLF0.

• **BLF1 :** elle réalise une exploitation verticale des résidus intra-niveau.

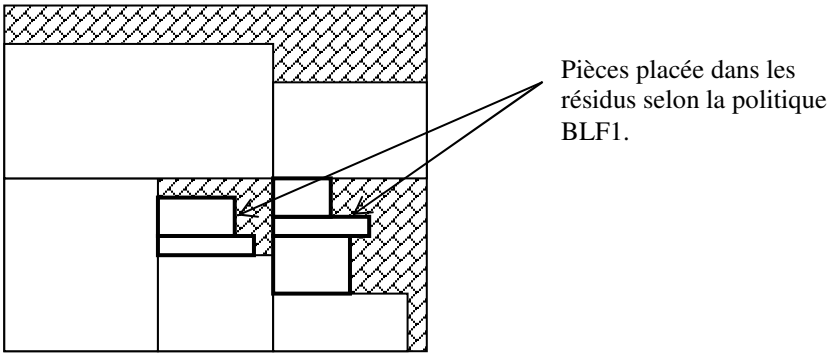


Figure 3 : Exploitation des résidus selon la politique BLF1

• **BLF2 :** elle réalise une exploitation verticale et horizontale des résidus intra-niveau.

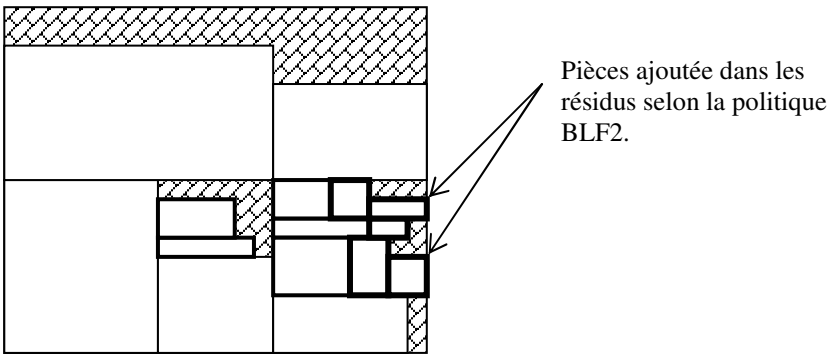


Figure 4 : Exploitation des résidus selon la politique BLF2

Le choix de la nuance de la politique BLF a être appliquée est paramètré dans le logiciel, ainsi l'utilisateur effectue son choix selon le jeu de pièces utilisé.

III.2. L'emboîtement.

De même, le choix d'une politique de placement des niveaux sur les boites est très importante dans cette phase. Rappelons qu'il s'agit d'agencer une liste L' de m niveaux de largeurs identiques sur des plaques de même largeur que ces niveaux. C'est un problème de placement à une dimension.

L'approche de résolution proposée par Djang [Dja98] consomme nettement moins de temps de calcul par rapport aux algorithmes génétiques, mais les algorithmes génétiques offrent des résultats meilleurs. Cette approche ne tient par compte de l'ordre d'apparition des pièces (niveaux dans cette phase) donc elle ne peut pas être hybridée à l'algorithme génétique.

L'heuristique BL suffit pour cette phase du fait que les niveaux ont la même largeur que les plaques.

IV. Améliorations.

Notre contribution, utilise, regroupe, et adapte des techniques d'optimisation trouvées dans la littérature pour la résolution de notre problème de placement. Nous proposons d'enrichir la méthode par les améliorations suivantes:

IV.1. Guider l'algorithme génétique.

Selon les résultats obtenus, en appliquant les algorithmes génétiques au problème de placement, on constate que pour les problèmes de moyenne et grande taille les méthodes heuristiques, basées sur le tri selon la décroissance des longueurs, offrent des résultats meilleurs que les algorithmes génétiques. Notre première amélioration consiste donc à guider la génération initiale en ajoutant, cet individu trié. De ce fait, notre méthode prend le relai des heuristiques. Donc notre contribution donne des résultats égaux ou même meilleurs que les méthodes heuristiques classiques indépendamment de la taille du problème.

IV.2. Optimisation en largeur.

Pour certains exemples où on a appliqué notre approche d'agencement sur la bande en fixant la largeur de la bande, on constate que l'écart entre les longueurs des pièces est élevé, de ce fait les résultats ne sont pas satisfaisants. Notre deuxième amélioration sera donc, de procéder à l'agencement des pièces sur la bande en largeur, i.e. en fixant la longueur de la bande, et on agence les pièces en largeurs.

On illustre l'agencement en largeur par l'exemple suivant :

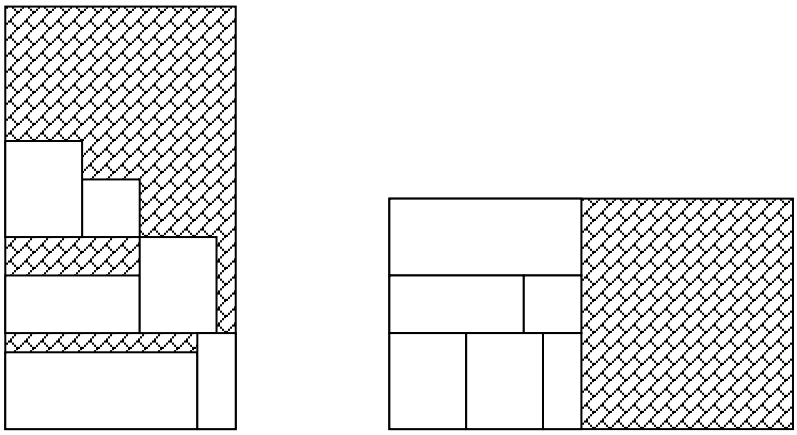


Figure 5 : Résultat d'agencement d'un jeu de pièces en longueur et en largeur.

On remarque nettement l'apport de cette amélioration pour ce jeu de pièces. Notons que les pièces gardent leurs orientations d'origine, seule la bande change de direction et l'agencement s'effectue en largeur.

Cette amélioration, donne plus de souplesse à notre approche, et l'utilisateur a le choix entre l'agencement en longueur et l'agencement en largeur selon le jeu de pièces utilisé.

Chapitre V.

Réalisation & Tests.

I. Introduction.

Dans ce chapitre, nous allons réaliser la méthode proposée, effectuer des tests, faire des comparaisons, et proposer des améliorations. Il s'agit donc d'implémenter deux algorithmes génétiques pour les deux étapes de traitement, à savoir le pré-traitement et l'emboîtement.

II. La réalisation.

II.1. Représentation des individus.

Une représentation interne d'un individu n'est rien qu'une liste dynamique chaînée. Une classe est utilisée pour coder les individus, en effet deux classes sont déclarées, à savoir GenoTypeLP pour coder les individus de Liste de Pièces pour la phase du pré-traitement ; et GenoTypeLN pour coder les individus de Liste de Niveaux pour la phase de l'emboîtement.

L'ordre d'apparition des pièces (niveaux) dans cette liste chaînée donne une solution possible (individu).

II.2. Le Chargement de la liste des pièces initiale.

Il s'agit de communiquer avec une base de données pour charger la liste des pièces initiale, par le biais de la fonction ChargerListePieces.

II.3. La population Initiale.

Il s'agit d'une fonction qui sert à générer une population initiale d'individus d'une manière aléatoire. Pour les deux phases de traitement deux fonctions sont définies, à savoir GenererPopInitLP et GenererPopInitLN.

La génération d'une population initiale dépend de deux paramètres contrôlés par l'utilisateur. Il s'agit de guider la population initiale en introduisant un individu trié selon la décroissance des longueurs. Le deuxième paramètre donne la possibilité à l'utilisateur d'interagir avec le logiciel en

imposant un individu ordonné à la population initiale. Ce deuxième paramètre trouve son utilité pour les exemples de tests où on connaît d'avance la solution optimale (qui est confectionné sur mesure selon la routine), et même si l'utilisateur arrive à avoir une solution meilleure que celle trouvée par le logiciel il peut l'intégrer à la population initiale. Ce paramètre donne une certaine interactivité au logiciel. Le reste de la population est généré aléatoirement.

II.4. La fonction de Tri.

Deux fonctions TrierGeneLP et TrierGeneLN servent à réaliser le tri selon la décroissance des longueurs. Elles seront appelées selon le choix de l'utilisateur pour guider la génération de la population initiale en triant le premier individu.

II.5. Evaluation des individus.

Deux fonctions d'évaluation sont définies pour les deux étapes de traitement à savoir EvaluerPopLP() et EvaluerPopLN(). Pour la phase de pré-traitement l'évaluation des individus dépend de la routine d'agencement choisie par l'utilisateur. Donc trois fonctions d'évaluation sont implémentées pour représenter les trois routines adoptées dans notre cas, il s'agit de EvaluerBLF0G(), EvaluerBLF1G(), et EvaluerBLF2G(), pour représenter respectivement les politiques BLF0, BLF1, et BLF2. Le G est ajouté pour exprimer la condition Guillotinable (i.e. la politique doit tenir compte de la contrainte de découpe de bout-en-bout). La fonction EvaluerPopLP() appelle une de ces trois nuances pour évaluer la population selon le choix de l'utilisateur.

Notons bien qu'une fonction d'évaluation et une vitesse sont choisis par l'utilisateur au lancement du processus, selon les paramètres « Qualité », « Temps » réglable au départ.

Pour la phase d'emboîtement une seule fonction d'évaluation est implémentée EvaluerPopLN() il s'agit d'agencer les niveaux sur les plaques selon la politique BL.

II.6. Les paramètres génétiques.

Ce sont des variables globales qui paramètrent le processus génétique, il s'agit de :

- MaxGens : le nombre maximal des générations.
- PopSize : la taille d'une population.
- PXOver : la probabilité de cross-over.
- Pmutation : la probabilité de mutation.

II.7. Les opérateurs génétiques.

Il s'agit de trois fonctions Select, CrossOver et Mutate qui servent à réaliser la phase génétique.

En plus une fonction Elitist est ajoutée au processus génétique, elle sert à conserver le meilleur individu après avoir exécuté un cross-over et une mutation, (on risque de perdre le meilleur individu après ces deux opérateurs génétiques).

II.8. La phase de pré-traitement.

C'est la première phase génétique dans le logiciel, elle réalise l'agencement des pièces sur des niveaux. La politique d'agencement appliquée dépend du choix de l'utilisateur, ainsi que des paramètres génétiques.

L'algorithme de pré-traitement est comme suit :

```
PreTraitement()
{
        GenererPopInitLP() ;
        generation = 0 ;
while (generation < MaxGens)
{
        EvaluerPopLP() ;
        SelectLP() ;
        CrossOverLP() ;
        MutateLP() ;
        ElitistLP() ;
        ++generation ;
}
Garder le meilleur individu.
}
```

II.9. La phase d'emboîtement.

C'est la deuxième phase génétique du logiciel, il s'agit de projeter la solution dite meilleure issue de la première phase génétique, sur les plaques.

L'algorithme d'emboîtement est ainsi défini :

```
Emboitement()
{
        GenererPopInitLN() ;
        generation = 0 ;
while (generation < MaxGens)
{
        EvaluerPopLN() ;
        SelectLN() ;
        CrossOverLN() ;
        MutateLN() ;
        ElitistLN() ;
        ++generation ;
}
Garder le meilleur individu.
}
```

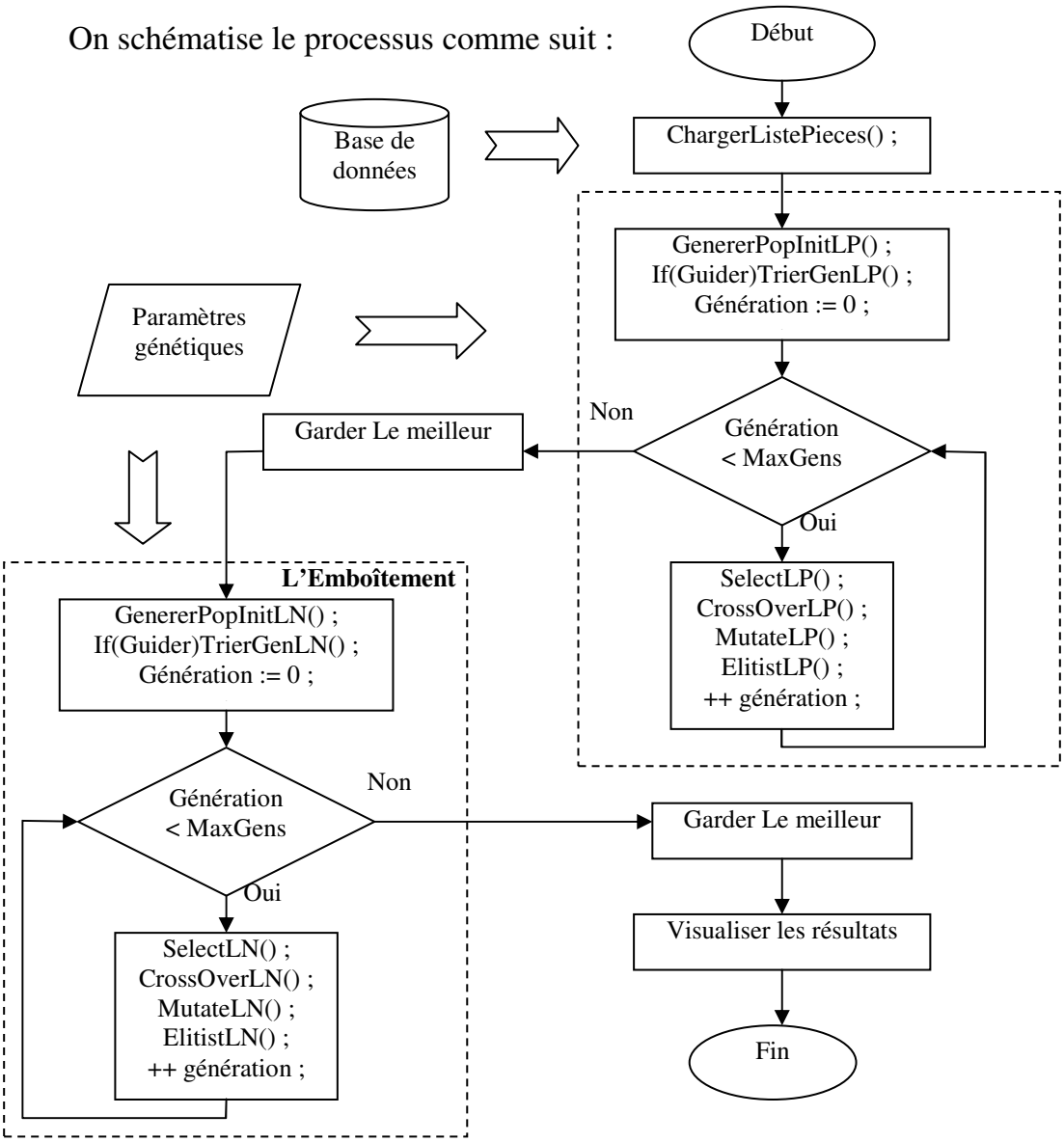
II.10. L'algorithme principal.

La fonction principale interagit avec l'utilisateur, saisit les paramètres, lance les deux étapes génétiques, et visualise le résultat.

main()
{
Saisir le choix de l'utilisateur.
PreTraitement() ;
Emboitement() ;
Visualiser la meilleure solution.
}

III. Organigramme.

On schématise le processus comme suit :



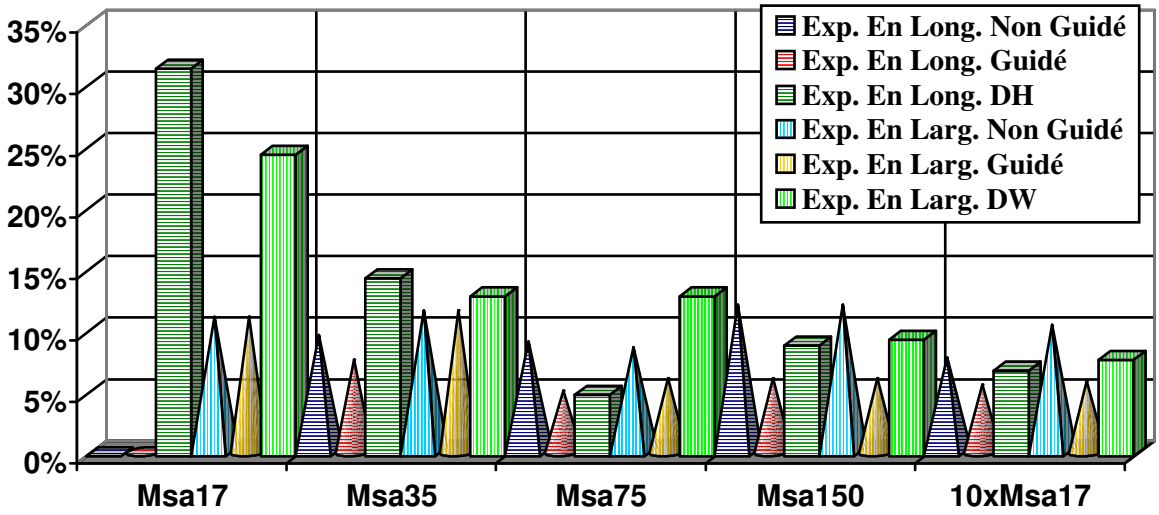
IV. Résultats.

Pour évaluer notre méthode nous avons confectionné des jeux de tests qui offrent une exploitation maximale de la matière (0% de chutes) en appliquant la politique BLF2G, et on gardant l'ordre initial d'apparition des pièces. Il s'agit de :

Nom	Taille	Optimal
Msa17	17	200 x 200
Msa35	35	200 x 200
Msa75	75	200 x 200
Msa150	150	200 x 200

Les résultats expérimentaux des quatre jeux de tests en appliquant notre approche avec une exploitation maximale de la matière sont récapitulés dans le tableau suivant. Notons que les jeux de tests ont été confectionnés en longueur (i.e. l'agencement en largeur ne garantit pas l'optimal).

Jeux de Test	Exploitation en longueur			Exploitation en largeur		
	Non Guidé	Guidé	DH	Non Guidé	Guidé	DW
Msa17	200x200	200x200	263x200	200x222	200x222	200x249
% de chute	0%	0%	31,5%	11%	11%	24,5%
Msa35	219x200	215x200	229x200	200x223	200x223	200x226
% de chute	9,5%	7,5%	14,5%	11,5%	11,5%	13%
Msa75	218x200	210x200	210x200	200x217	200x212	200x226
% de chute	9%	5%	5%	8,5%	6%	13%
Msa150	224x200	213x200	218x200	200x224	200x212	200x219
% de chute	12%	6%	9%	12%	6%	9,5%
10x Msa17	2154x200	2109x200	2139x200	200x2207	200x2115	200x2157
% de chute	7,7%	5,45%	6,95%	10,35%	5,75%	7,85%



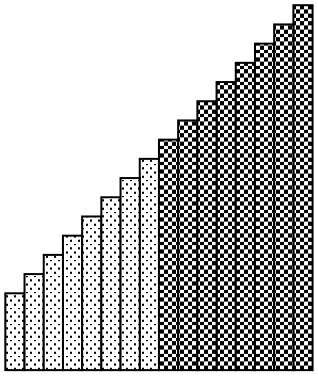
Les résultats montrent que notre méthode atteint l'optimal pour les petits jeux de test, et approche de l'optimal pour les jeux de test de moyenne taille. Et même donne des résultats meilleurs que l'heuristique basée sur le tri. En ce qui concerne les jeux de test de grande taille l'heuristique basée sur le tri donne des résultats meilleurs que notre algorithme. L'algorithme guidé trouve son utilité dans ce cas de figure.

V. Améliorations.

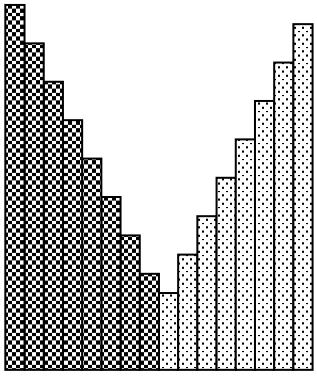
V.1. Guider l'algorithme génétique.

Vu les améliorations apportées par l'insertion d'un individu, trié selon la décroissance des longueurs, et en vue d'enrichir la méthode, nous avons proposé d'introduire certains heuristiques basées sur des politiques de tri à la population initiale. Il s'agit des quatre politiques suivantes:

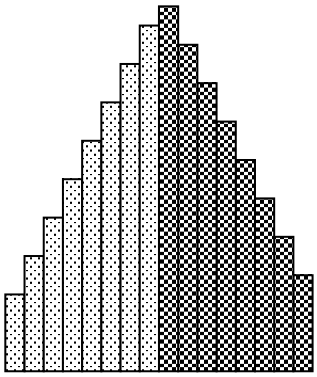
a) Trier selon la croissance des longueurs.



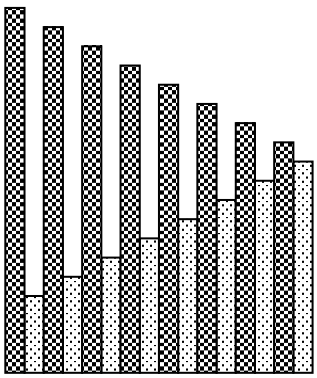
b) Trier selon la décroissant puis la croissance des longueurs.



c) Trier selon la croissance puis la décroissance des longueurs.



d) Trier alternativement selon la décroissance puis la croissance des longueurs (harmonique).



Les résultats obtenus montre l'apport de cette amélioration sur certains jeu de tests, qui sont représentés sur le tableau suivant :

Jeux de Test	Exploitation en longueur		Exploitation en largeur	
	Non Guidé	**Guidé**	**Non Guidé**	**Guidé**
Msa17	200x200	200x200	200x222	200x222
% de chute	0%	0%	11%	11%
Msa35	219x200	215x200	200x223	200x221
% de chute	9.5%	7,5%	11,5%	10,5%
Msa75	218x200	210x200	200x217	200x212
% de chute	9%	5%	8,5%	6%
Msa150	224x200	213x200	200x224	200x211
% de chute	12%	6,5%	12%	5,5%
10x Msa17	2154x200	2109x200	200x2207	200x2103
% de chute	7,7%	5,45%	10,35%	5,15%

Les cases en gris montrent les améliorations. Cette amélioration trouve son utilité pour certains cas.

IV.2. La ré-initialisation de la population.

D'après le suivi de l'exécution de traitement nous avons constaté que l'algorithme génétique converge vers un optimum local après un certain nombre de générations (l'algorithme est piégé dans cet optimum). Et qu'il n'y a plus d'améliorations à partir de ce rang. Pour remédier à cela nous avons introduit un nouveau paramètre à l'algorithme. Il sert à ré-initialiser la population courante en générant une nouvelle population aléatoire. Cette population doit prendre le relais des générations précédentes en gardant le meilleur individu issu des générations précédentes. Ce paramètre ne doit pas altérer le processus génétique. Une intervention fréquente de ce paramètre dégrade les résultats. Pour cela on définit un nouveau paramètre nommé Seuil de Stabilité qui sera défini par l'utilisateur. Quand l'algorithme atteint ce seuil le nouveau paramètre de ré-inisialisation sera opérationnel, ainsi des phases de ré-inisialisation seront déclenchées.

Le tableau suivant présente le résultat de notre approche sur les quatre jeux de tests en appliquant le nouveau paramètre de ré-initialisation de la population :

Jeux de Test	Exploitation en longueur		Exploitation en largeur	
	Non Guidé	**Guidé**	**Non Guidé**	**Guidé**
Msa17	200x200	200x200	200x213	200x213
% de chute	0%	0%	6,5%	6,5%
Msa35	215x200	215x200	200x223	200x221
% de chute	7,5%	7,5%	11,5%	10,5%
Msa75	213x200	210x200	200x212	200x212
% de chute	6,5%	5%	6%	6%
Msa150	219x200	213x200	200x222	200x211
% de chute	9,5%	6,5%	11%	5,5%
10x Msa17	2149x200	2109x200	200x2200	200x2103
% de chute	7,45%	5,45%	10%	5,15%

On remarque les améliorations apportées par ce nouveau paramètre.

V. Comparaisons.

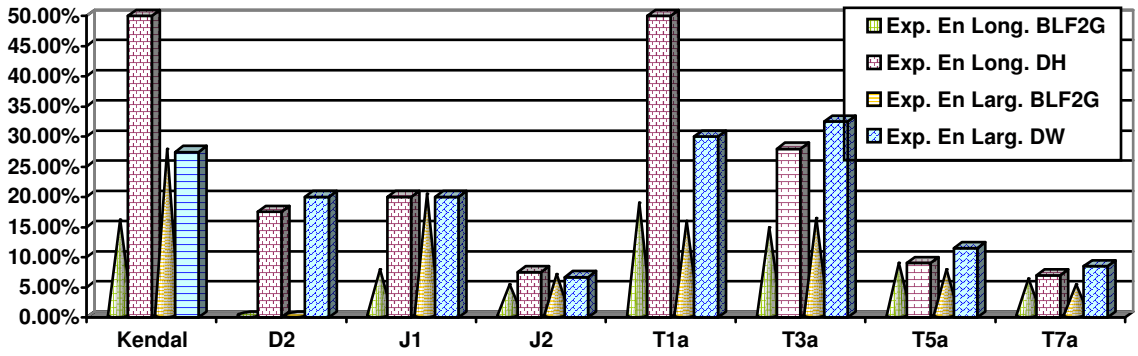
Nous avons choisi un vaste nombre de jeu de tests trouvés dans la littérature, dont on connaît la solution optimale. Ces exemples de tests sont présentés en Annexe.

Référence	Nom	Taille	Optimal
[Bur99a]	Kendall	13	140 x 80
[Rat97b]	D2	21	40 x 60
[Jak96]	J1	25	40 x 15
[Jak96]	J2	50	40 x 15
[Hop99]	T1a, T1b, T1c, T1d, T1e N1a, N1b, N1c, N1d, N1e	17	200 x 200
[Hop99]	T2a, T2b, T2c, T2d, T2e N2a, N2b, N2c, N2d, N2e	25	200 x 200

[Hop99]	T3a, T3b, T3c, T3d, T3e N3a, N3b, N3c, N3d, N3e	29	200 x 200
[Hop99]	T4a, T4b, T4c, T4d, T4e N4a, N4b, N4c, N4d, N4e	49	200 x 200
[Hop99]	T5a, T5b, T5c, T5d, T5e N5a, N5b, N5c, N5d, N5e	73	200 x 200
[Hop99]	T6a, T6b, T6c, T6d, T6e N6a, N6b, N6c, N6d, N6e	97	200 x 200
[Hop99]	T7a, T7b, T7c, T7d, T7e N7a, N7b, N7c, N7d, N7e	197	200 x 200

Notons bien que ces exemples de tests ne sont pas confectionnés pour vérifier la contrainte de découpe de bout-en-bout. Donc l'optimal risque de ne pas être atteint par notre méthode. Nous lançons notre méthode sur ces exemples juste pour situer notre position, et voir la différence.

Jeux de test	**Exploitation en longueur**			**Exploitation en largeur**		
	AG Classique	**Notre Approche**	**DH**	**AG Classique**	**Notre Approche**	**DW**
Kendall	162x80 16,63%	162x80 16,63%	210x80 50%	140x102 28,51%	140x102 28,51%	140x102 27,5%
D2	40x60 0%	40x60 0%	47x60 17.5%	40x60 0%	40x60 0%	40x72 20%
J1	43x15 7,5%	43x15 7,5%	48x15 20%	40x18 20%	40x18 20%	40x18 20%
J2	43x15 5,12%	42x15 5%	43x15 7,5%	40x17 7,08%	40x16 6,67%	40x16 6,67%
T1a	237x200 18,5%	237x200 18,5%	300x200 50%	200x246 23%	200x231 15,5%	200x260 30%
T3a	234x200 17%	229x200 14,5%	256x200 28%	200x239 19,5%	200x232 16%	200x265 32,5%
T5a	238x200 19%	217x200 8,5%	218x200 9%	200x239 19,5%	200x215 7,5%	200x223 11,5%
T7a	226x200 13%	212x200 6%	214x200 7%	200x227 13,5%	200x210 5%	200x217 8,5%



Ces résultats montrent la puissance de notre méthode en matière d'exploitation des résidus tout en vérifiant la contrainte de découpe de bout-en-bout. Notre méthode a nettement approché de l'optimal pour les exemples de Jakobs (J1 et J2) et elle atteint même l'optimal pour l'exemple de Dagli (D2), qui sont des jeux de test de petite taille. Le jeu de test Kendall, de petit taille, à donner des résultats moins efficaces. En ce qui concerne les jeux de test de Hopper nous avons choisi quatre exemples de petite moyenne et grand taille. Les résultats obtenus sont très satisfaisants du fait qu'on a approché de l'optimal malgré que ces jeux de test ne vérifient pas la contrainte de découpe de bout-en-bout.

Pour mieux situer notre approche nous avons utilisé un exemple réel [Msa95]. Il s'agit de produit D703 où les résultats obtenus sont 30 plaques de dimension 2500x1250.

Notre méthode à donnée le résultat suivant (en nombre de plaque):

Jeux de Test	**Exploitation en longueur**			**Exploitation en largeur**		
	Non Guidé	**Guidé**	**DH**	**Non Guidé**	**Guidé**	**DW**
MsaPFE95	29	29	29	28	28	28

On voit bien qu'il n'y à pas une grande différence entre les résultats. nous avons 29 plaques pour l'agencement en longueur et pour les trois nuances (non guidé, guidé, et trié) et de même pour l'agencement en largeur (28 plaques par tout). L'apport de notre méthode n'est pas remarquable dans ce cas, seulement 3 plaques de gain par rapport au résultat obtenu par [Msa95], une

telle chose est expliquée par l'absence des petites pièces qui font la puissance de notre méthode en matière d'exploitation des résidus.

VI. Conclusion.

Notre contribution, au problème de découpe rectangulaire, a montré son efficacité. En premier lieu nous avons développé une routine puissante en matière d'exploitation des résidus (BLF2G). Deuxièmement nous avons proposé l'hybridation de cette routine avec un algorithme génétique amélioré, qui surmonte les problèmes rencontrés par les algorithmes génétiques classiques. Pour les problèmes de petite taille notre méthode a atteint l'optimal. Pour les problèmes de moyenne taille notre méthode a approché nettement de l'optimal, et elle a donné des résultats meilleurs que celui de l'heuristique basée sur le tri. Pour les problèmes de grande taille notre méthode a divergé, (comme les autres méthodes génétiques classiques), et l'heuristique basée sur le tri a donné des résultats meilleurs. Mais avec l'option de guider l'algorithme génétique par l'individu trié, notre méthode a réussi de reprendre la main et améliore même le résultat trouvé par l'heuristique basée sur le tri.

La position de notre méthode par rapport aux autres méthodes est bonne. Malgré que notre approche tient compte de la contrainte de découpe de bout-en-bout, nous avons réussi à avoir des résultats comparables avec d'autres méthodes qui ne vérifient pas cette contrainte. En plus notre deuxième amélioration d'agencement en largeur a donné de la souplesse à notre méthode, et donne souvent des résultats meilleurs que l'agencement classique (en longueur). La dernière amélioration qui sert à réinitialisé la population courant après un certain nombre de stabilité, à apporter des bons résultats. Elle tente de diversifie la recherche quand l'algorithme génétique est piégé dans un minimum local.

Chapitre VI

Conclusion Générale.

I. Conclusion.

Dans ce travail une approche génétique hybride à été développée pour la résolution du problème de découpe rectangulaire avec contraintes. Du fait que le problème de découpe est un problème combinatoire, où l 'espace de recherche de la solution optimale est très large, les travaux récents encouragent l'utilisation des algorithmes génétiques. Un problème commun rencontré par les algorithmes génétiques est la divergence lorsque la taille des données est grande. Même les méthodes heuristiques donnent des résultats meilleurs que les algorithmes génétiques pour les problèmes de moyenne et grande taille. Notre objectif dans ce travail a consisté donc à trouver une solution originale au problème de découpe en apportant une solution au problème que rencontrent les algorithmes génétiques.

Notre premier apport a consisté donc à subdiviser le problème initial en deux sous-problèmes. Le premier consiste à agencer les pièces sur une bande infinie structurée en niveaux tout en respectant les contraintes du problème, en développant une routine de placement puissante baptisée « BLF2G » (Bottom Left Fit 2 Guillotineable) d'exploitation des résidus intra-niveaux, le '2' signifie que l'exploitation des résidus s'effectue en deux sens (verticale puis horizontal). Le deuxième sous-problème consiste à projeter les niveaux issus de la première étape sur les plaques «exploitation inter-niveaux », en utilisant une routine de placement simple, il s'agit de la routine BL vu que cette étape n'est rien qu'un agencement uni-dimensionnel. Par ce procédé on a réussi à diminuer la complexité du problème en le subdivisant en deux.

Le premier sous-problème garde une part importante de la complexité du problème initial ; l'application de l'algorithme génétique pour cette partie reste confrontée au même inconvénient. Les algorithmes génétiques échouent devant les heuristiques pour la résolution des problèmes de moyenne et grande taille. Notre deuxième amélioration a consisté donc à introduire un individu trié selon la décroissance des longueurs à la population initiale, introduisant ainsi de la matière génétique de qualité au processus génétique. Par cette amélioration notre méthode a surpassé les méthodes heuristiques indépendamment de la taille du problème.

Vu les résultats satisfaisants réalisés par l'introduction de l'individu trié selon la décroissance des longueurs à la population initiale on à développé cinq autres heuristiques basées sur le tri. Les individus obtenus sont injectés dans la population initiale. On a constaté certains améliorations pour certain cas, toute fois l'optimal n'est pas atteint pour les problèmes de moyenne et grande taille.

Une autre amélioration a été introduite à notre méthode. Il s'agit de procéder à l'agencement en largeur, en fixant la longueur de la bande et en laissant la largeur infinie. Cette méthode n'influe par sur la contrainte d'orientation des pièces. Avec cette amélioration l'utilisateur à plus de souplesse en choisissant l'une des deux méthodes d'agencement en longueur ou en largeur. L'agencement en largeur donne souvent des résultats meilleurs que l'agencement classique en longueur.

Notre dernière amélioration trouve son utilité en constatant que le processus génétique converge vers un optimal local. Donc nous avons introduit un nouveau paramètre qui sert à ré-initialiser la population courante, dès quelle converge, par une nouvelle population aléatoire en

gardant le meilleur individu issu des générations précédentes. Cette amélioration donne dans certain cas des améliorations.

Selon le jeu de pièces utilisé, une des améliorations précédentes trouve son utilité. Donc on a mis à la disposition de l'utilisateur une méthode d'agencement puissante basée sur les algorithmes génétiques, augmentée de certains paramètres avec les quels il peut trouver une configuration qui lui offre le meilleur résultat.

En conclusion notre méthode à réussi à atteindre l'optimal pour les problèmes de petite taille. Pour les problèmes de moyenne et grande taille l'optimal n'est pas toujours atteint. En comparaison avec les autres méthodes, notre méthode est bien placée, puisqu'elle donne toujours des résultats meilleurs que les méthodes heuristiques quelque soit la taille du problème contrairement aux autres méthodes génétiques classiques.

II. Travaux futurs.

L'utilisation des algorithmes génétiques qui sont des méthodes puissantes en matière d'exploration de l'espace de recherche connaît ces limites dans l'application au problème de découpe de moyenne et grande taille. En perspective, et vu la nette amélioration apportée par l'introduction de l'individu trié à la population initiale, le développement d'une méthode heuristique qui sert à guider l'algorithme génétique tout le long de son processus génétique depuis la population initiale jusqu'aux opérateurs génétiques, reste à explorer. De plus ce travail peut être étendu à d'autres formes géométriques, en vue d'être appliqué pour la découpe de cuir, de tissus, etc. ...

En plus la parallélisation des algorithmes génétiques offre des avantages en temps de résolution et en qualité des solutions qui reste aussi à explorer.

Références Bibliographique.

1. [Ama01] R. Amara, S. R. Semghouni, La Résolution des Problèmes d'Emplois du Temps Scolaires en utilisant les Algorithmes Génétiques en Mono-critère et Multicritères, Mémoire du Projet de fin d'études pour l'obtention du diplôme d'ingénieur d'état en Informatique, Faculté du Génie Electrique, Département d'Informatique, USTHB, Novembre 2001.
2. [Amr01] F. M. Amrane, Détection de primitives géométriques dans une image par un algorithme génétique, Mémoire du Projet de fin d'études pour l'obtention du diplôme d'ingénieur d'état en Informatique, Faculté du Génie Electrique, Département d'Informatique, USTHB, Novembre 2001.
3. [Bab01] S. Baba-ali, Une approche génétique parallèle pour la résolution du problème de dimensionnement de portes des circuits intégrés VLSI, Thèse de Magistère, Faculté du Génie Electrique, Département d'Electronique, USTHB, 2001.
4. [Ben03] W. Benfold, M. Manfrin, A. M. R. Manso, S. Spinella, A Genetic Algorithm for 2 D Glass Cutting Problem, University of Southampton, Great Britain, 2003.
5. [Benn00] D. A. Bennett, M. P. Armstrong, Development of Two-dimensional Genetic Algorithms to Evolve Optimal Landscapes, Department of Geography, University of lowa City, IA, 2000.
6. [Ber] G. Bernard, J. Feat, Introduction aux Algorithmes Génétiques, Application aux problèmes d'optimisation combinatoire.
7. [Bor99] A. Bortfeldt, H. Gehring, Two Meta-heuristics for Strip Packing Problems, FernUniversität Hagen, Profilstr. 8, 58084 Hagen, Germany, 1999.
8. [Bou01] B. Bouchenak, A, Amoura, Application des Algorithmes Génétiques Séquentiels et Parallèles aux Problème de Routage de Véhicules Multicritère avec Fenêtre Horaires, Mémoire du Projet de fin d'études pour l'obtention du diplôme d'ingénieur d'état en Informatique, Faculté du Génie Electrique, Département d'Informatique, USTHB, Novembre 2001.
9. [Bur99a] E. Burke and G. Kendall, Comparison of Meta-Heuristic Algorithms for Clustering Rectangles. Computers in Engineering 37, 383-386, 1999a.

10. [Car88] J. Carlier, Problème d'ordonnancement, modélisation, complexité, algorithmes, Edition Masson, pp. Chap. III, VI, VIII, 1988.
11. [Cha00] D. Chapman, Visual C++ 6.0, CampusPress, 2000.
12. [Cof96] E. G. Coffman Jr., M. R. Garey, D. S. Johnson, Approximation Algorithms for Bin Packing : A Survey, Appears in Approximation Algorithms for NP-Hard Problems, D. Hochbaum (ed.), PWS Publishing, Boston (1996), 46-93.
13. [Cof99] E. G. Coffman Jr., J. Csirik, G. J. Woeginger, Approximate Solutions to Bin Packing Problems, March 15, 1999.
14. [Cor93] A. L. Corcoran, R. L. Wainwright, LIBGA: A User-Friendly Workbench for Order-Based Genetic Algorithm Research, Proceedings of the 1993ACM/SIGAPP Symposium on Applied Computing, ACM Press: New York, 1993.
15. [DeJ90] K. A. De Jong, W. M. Spears, Using Genetic Algorithms to Solve NP-Complete Problems, George Mason University, 1990.
16. [Dja98] P. A. Djang, P. R. Finch, Solving One Dimension Bin Packing Problems, submitted to Journal of Heuristics, June 1998.
17. [Dri92] H. Drias, Approche probabiliste du dénombrement des solutions du problème de satisfiabilité, Thèse de doctorat d'état en informatique, Faculté du Génie Electrique, Département d'Informatique, USTHB, 1992.
18. [Fuj93] K. Fujita, S. Akagi, N. Hirokawa, Hybrid Approach for Optimal Nesting Using a Genetic Algorithm and a Local Minimization Algorithm, Department of Mechanical Engineering for Industrial Machinery and Systems, Osaka University Suita, Osaka, JAPAN, 1993.
19. [Gar76] M. R. Garey, Computers and intractability, A guide to theory of NP-completeness, 1976.
20. [Ghe92] G. Ghezali, Etude d'un problème de découpe avec conception d'un logiciel, Mémoire du Projet de fin d'études pour l'obtention du diplôme d'ingénieur d'état en Recherche Opérationnel, Département R.O. USTHB, 1992.
21. [Gol89] D. E. Goldberg, Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley, 1989.
22. [Gol92] D. E. Goldberg, Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking, Department of General Engineering University of Illinois at Urbana-Champaign Urbana, IL 61801, 1992.

23. [Gué92] C. Guéret, N. Jussien, O. Lhomme, C. Prins, Chargement d'avions dans le cadre d'une projection de forces, Ecole des Mines de Nantes, 1992.

24. [Hol75] J. H. Holland, Adaptation in natural and artificial systems, Ann Arbor : The University of Michigan Press.

25. [Hop97] E. Hopper, B. Turton, Application of Genetic Algorithms to Packing Problems – A review, In : Proceedings of the 2nd On-line World Conference on Soft Computing in Engineering Design and Manufacturing, Springer Verlag, London, pp279-288, 1997.

26. [Hop99] E. Hopper, B. Turton, A Genetic Algorithm for a 2D Industrial Packing Problem, Computers and Industrial Engineering, vol. 37/1-2, pp. 375-378, 1999.

27. [Hop00] Eva Hopper, Two-dimensional Packing utilizing Evolutionary Algorithms and other Meta-Heuristic Methods, A Thesis submitted to the University of Wales for the Degree of Doctor of Philosophy, May 2000.

28. [Hue81] Hue, Combinatorial algorithms, 1981.

29. [Jak96] S. Jakobs, On genetic algorithms for the packing of polygons. European Journal of Operations, Research 88, 165-181, 1996

30. [Khe01] A. Khelladi, R. Ouafi, M. Hifi, The A* Algorithm for Orthogonal Rectangular Packing and Strip Packing Problems, Fifth International Symposium On Programming system, ISPF'2001.

31. [Khu96] S. Khuri, M. Schütz, J. Heitkötter, Evolutionary Heuristics for the Bin Packing Problem, Department of Mathematics & Computer Science, San José State University One Washington Square, San José, CA 95192-0103, USA, 1996.

32. [Lau87] J. L. Laurière, Intelligence Artificielle, résolution de problèmes par l'homme et la machine, 3ème édition, 1987.

33. [Ler95] I. C. Lerman, R. F. Ngouenet, Algorithmes génétiques séquentiels et parallèles pour une représentation affine des proximités, Rapport de Recherche N°2570, INRIA, Janvier 1995.

34. [Liu97] D. Liu, H. Teng, An improved BL-Algorithm for genetic algorithm of the orthogonal packing of rectabgles, European Journal Of Operational Research vol. 112 (2) pp. 413-420, 1997.

35. [Lod01] A. Lodi, S. Martello, M. Monaci, Two-dimensional packing problems : A survey, Dipartimento di Elettronica, Informatica e Sistemistica, University of Bologna Viale Risorgimento, 2 – 40136 – Bologna (Italy), 2001.

36. [Lod02] A. Lodi, S. Martello, M. Monaci, Two-dimensional packing problems : A survey, Invited Review, European Journal of Operational Research 141 (2002) 241-252, Elsevier, 2002.

37. [Man99] K. F. Man, K. S. Kwong, Genetic Algorithms: Concepts and Designs, University of Hong Kong Tat Chee Avenue, Kowloon, Hong Kong, 1999.

38. [Mic96] Z. Michalewics, Genetic Algorithms + Data Structures = Evolution Programs, Third, Revised and Extended Edition, Springer 1996.

39. [Msa95] S. Abou Msabah, M. Izzerouken, Conception et Réalisation d'un logiciel d'optimisation des chutes de tôles, Mémoire du Projet de fin d'études pour l'obtention du diplôme d'ingénieur d'état en Informatique, Institue d'Informatique, USTHB, 1995.

40. [Oli02] A. Oliver, O. Regragui, N. Monmarché, G. Venturini, Optimisation génétique et interactive de sites web, RSTI-TSI Volume 21, n°2/2002, pp. 965–984, 2002.

41. [Pre83] F. Premti, méthodes stochastiques dans les problèmes du placement, Doctorat 3ème cycle R . D. université scientifique et médicale de Grenoble, 1983.

42. [Rai99] G. R. Raidl, G. Kodydek, Genetic Algorithms for the Multiple Container Packing Problem, Department of computer graphics, Vienna University of Technology, Karlsplatz 13/1861, 1040 Vienna, Austria, 1999.

43. [Rat97a] Ratanapan K. and Dagli C. H. An object-based evolutionary algorithm for solving rectangular piece nesting problems. In: IEEE (eds.), Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, IEEE, Piscataway, NJ, USA, pp. 989-994, 1997.

44. [Ren95] J. M. Renders, Algorithmes Génétiques et réseaux de neurones, applications à la commande de processus, Ed. Hermès, Paris, 1995.

45. [Sak84] M. Sakarovitch, Optimisation combinatoire, méthode mathématiques et algorithmiques, Edition Herman, 1984.

46. [Sch98] V. Schnecke, O. Vornberger, Hybrid Genetic Algorithms for Constrained Placement Problems, appeared in IEEE Transactions on Evolutionary Computation, Vol. 1, N°. 4, November 1997, pp. 266-277.

47. [Spe93] W. M. Spears, K. A. De Jong, T. Bäck, D. B. Fogel, H. De Garis, An Overview of Evolutionary Computation, in the proceedings of the European Conference on Machine Learning, 1993.

48. [Sri99] A. Srivastav, P. Stangier, Tight Approximations for Resource Constrained scheduling and Bin Packing, Institut Für Informatik, Humboldt Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany, January 1999.

49. [Val01] C. L. Valensuela, P. Y. Wang, Heuristics for Large Strip Packing Problems with Guillotine Patterns : an Empirical Study, MIC'2001, 4th Meta-heuristics International Conference, 2001

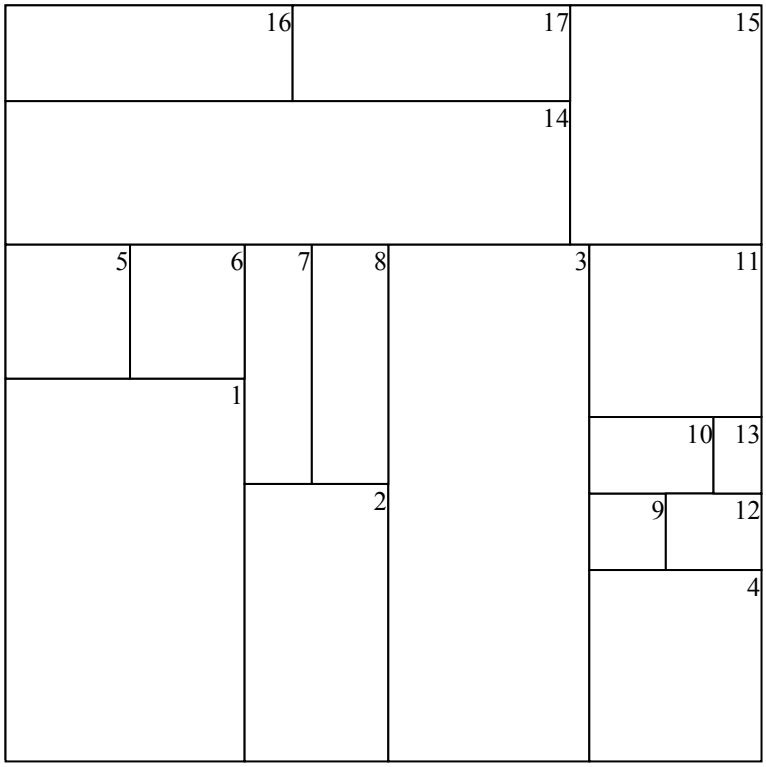
50. [Vall01] T. Vallée, M. Yildizoglu, Présentation des algorithmes génétiques et de leurs applications en économie, Université de Nantes, LEA-CIL, F-44312 Nantes, Septembre 2001.

Annexe A :

Jeux de tests développés pour la réalisation de la méthode :

Nom	Taille	Optimal
Msa17	**17**	**200 x 200**
Msa35	**35**	**200 x 200**
Msa75	**75**	**200 x 200**
Msa150	**150**	**200 x 200**

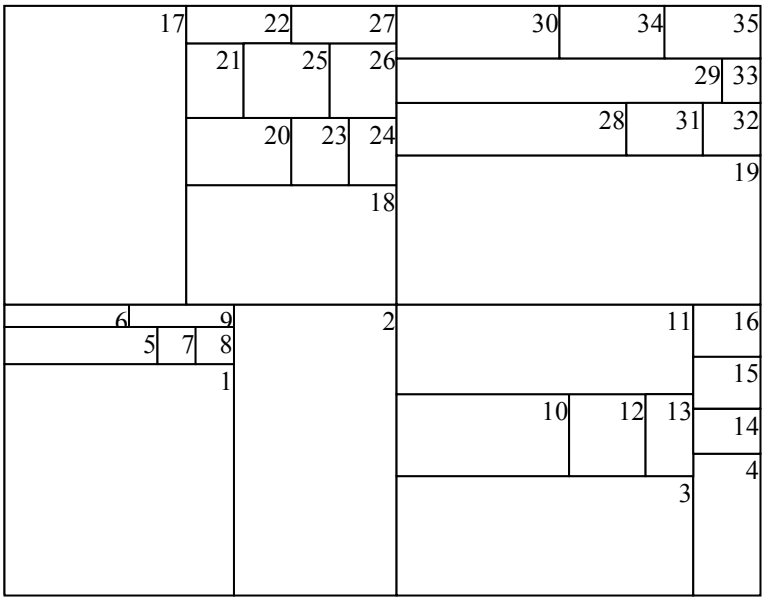
	Nom:	**Msa17**
	Taille:	17
	Optimum:	200 x 200
no	**Largeur**	**Longueur**
1	63	101
2	38	74
3	53	137
4	46	51
5	33	36
6	30	36
7	19	63
8	19	63
9	20	20
10	33	20
11	46	46
12	26	20
13	13	20
14	150	38
15	50	63
16	76	25
17	74	25



Msa17 : Agencement selon la politique BLF2G.
L'ordre des pièces donné par S. Abou Msabah

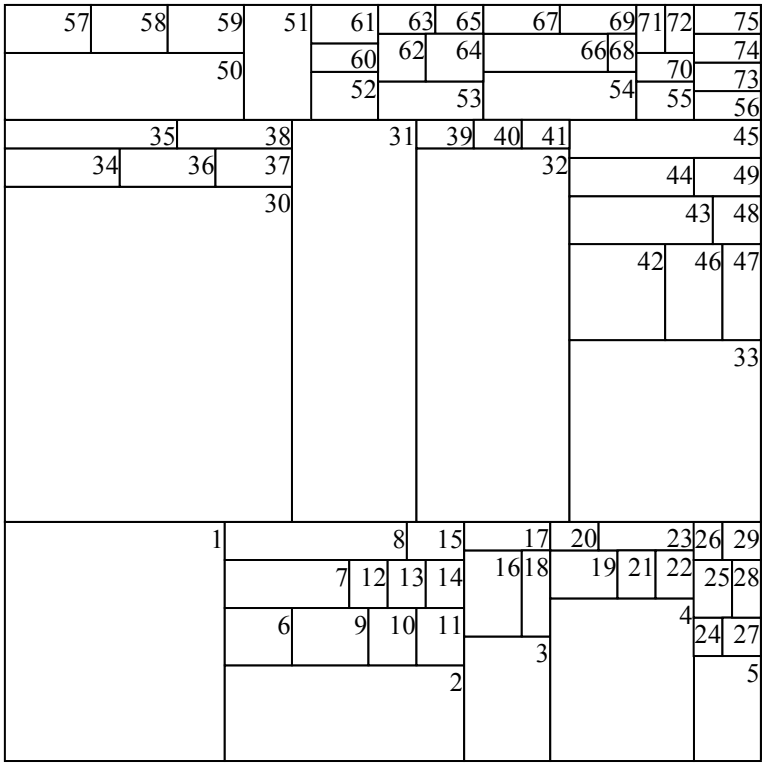
	Nom:	**Msa35**
	Taille:	35
	Optimum:	200 x 200

no	**Largeur**	**Longueur**
1	61	79
2	42	99
3	79	40
4	18	48
5	41	13
6	33	7
7	10	13
8	10	13
9	28	7
10	46	28
11	79	31
12	20	28
13	13	28
14	18	15
15	18	18
16	18	18
17	48	101
18	56	40
19	96	50
20	28	23
21	15	25
22	28	13
23	15	23
24	13	23
25	23	25
26	18	25
27	28	13
28	61	18
29	86	15
30	43	18
31	20	18
32	15	18
33	10	15
34	28	18
35	25	18



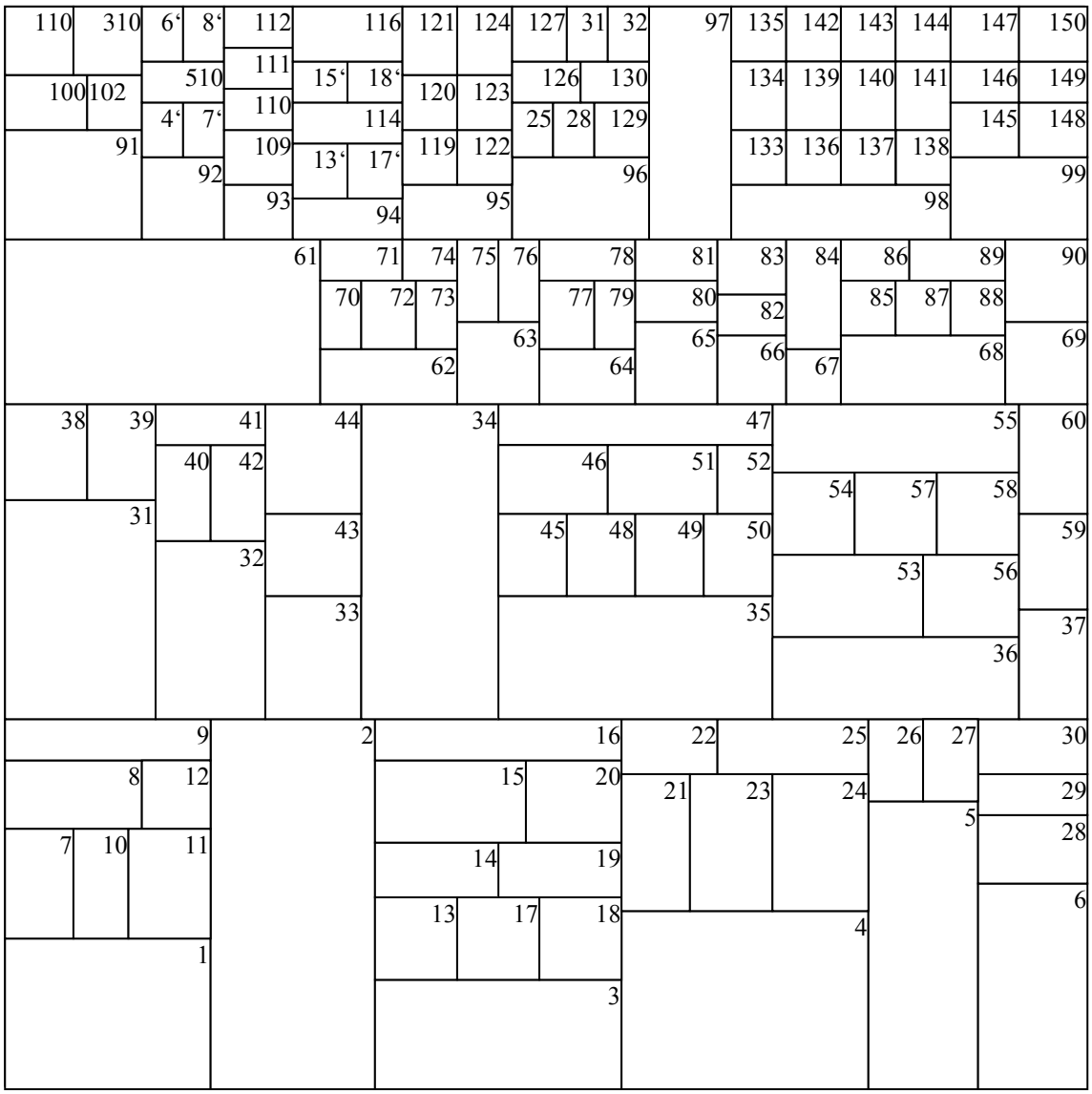
Msa35 : Agencement selon la politique BLF2G.
L'ordre des pièces donné par S. Abou Msabah

	Nom:	**Msa75**
	Taille:	75
	Optimum:	200 x 200
no	**Largeur**	**Longueur**
1	58	63
2	63	25
3	23	33
4	38	43
5	18	28
6	17	15
7	33	13
8	48	10
9	20	15
10	13	15
11	13	15
12	10	13
13	10	13
14	10	13
15	15	10
16	15	23
17	23	7
18	8	23
19	18	13
20	13	7
21	10	13
22	10	13
23	25	7
24	7	10
25	10	15
26	6	10
27	11	10
28	8	15
29	12	10
30	76	90
31	33	107
32	40	100
33	51	49
34	31	10
35	46	7
36	25	10
37	20	10
38	30	7
39	15	7
40	12	7
41	13	7
42	26	25
43	38	13
44	33	10
45	51	10
46	15	25
47	10	25
48	13	13
49	18	10
54	40	13
55	15	10
56	18	7
57	23	13
58	20	13
59	20	13
60	18	7
61	18	10
62	13	13
63	15	7
64	15	13
65	13	7
66	33	10
67	20	7
68	7	10
69	20	7
70	15	7
71	8	13
72	7	13
73	18	7
74	18	9
75	18	7
50	63	17
51	18	30
52	18	13
53	28	10



Msa75 : Agencement selon la politique BLF2G.
L'ordre des pièces donné par S. Abou Msabah

	Nom:	**Msa150**
	Taille:	150
	Optimum:	200 x 200
no	**Largeur**	**Longueur**
1	38	28
2	30	68
3	46	20
4	46	33
5	20	53
6	20	38
7	13	20
8	25	13
9	38	7
10	10	20
11	15	20
12	13	13
13	15	15
14	23	10
15	28	15
16	46	8
17	16	15
18	15	15
19	23	10
20	18	15
21	13	25
22	18	10
23	15	25
24	18	25
25	28	10
26	10	15
27	10	15
28	20	13
29	20	7
30	20	10
31	28	40
32	20	33
33	18	23
34	24	58
35	51	23
36	46	15
37	13	20
38	15	18
39	13	18
40	10	18
41	20	7
42	10	18
43	18	15
44	18	20
45	12	15
46	21	13
47	51	7
48	13	15
49	13	15
50	13	15
51	20	13
52	10	13
53	28	15
54	15	15
55	46	13
56	18	15
57	16	15
58	15	15
59	13	18
60	13	20
61	59	30
62	25	10
63	15	15
64	18	10
65	15	15
66	13	13
67	10	10
68	30	13
69	15	15
70	7	13
71	15	7
72	11	13
73	7	13
74	10	7
75	7	15
76	8	15
77	10	13
78	18	7
79	8	13
80	15	7
81	15	8
82	13	7
83	13	10
84	10	20
85	10	10
86	12	7
87	10	10
88	10	10
89	18	7
90	15	15
91	25	21
92	15	17
93	13	13
94	20	10
95	20	11
96	25	17
97	17	44
98	40	11
99	25	17
100	15	10
101	12	13
102	10	10
103	13	13
104	8	10
105	15	7
106	8	10
107	7	10
108	7	10
109	13	10
110	13	7
111	13	7
112	13	7
113	10	10
114	20	7
115	10	7
116	20	10
117	10	10
118	10	7
119	10	10
120	10	10
121	10	13
122	10	10
123	10	10
124	10	13
125	8	10
126	12	7
127	10	10
128	7	10
129	10	10
130	13	7
131	8	10
132	7	10
133	10	10
134	10	13
135	10	10
136	10	10
137	10	10
138	10	10
139	10	13
140	10	13
141	10	13
142	10	10
143	10	10
144	10	10
145	12	10
146	13	7
147	12	10
148	13	10
149	12	7
150	13	10



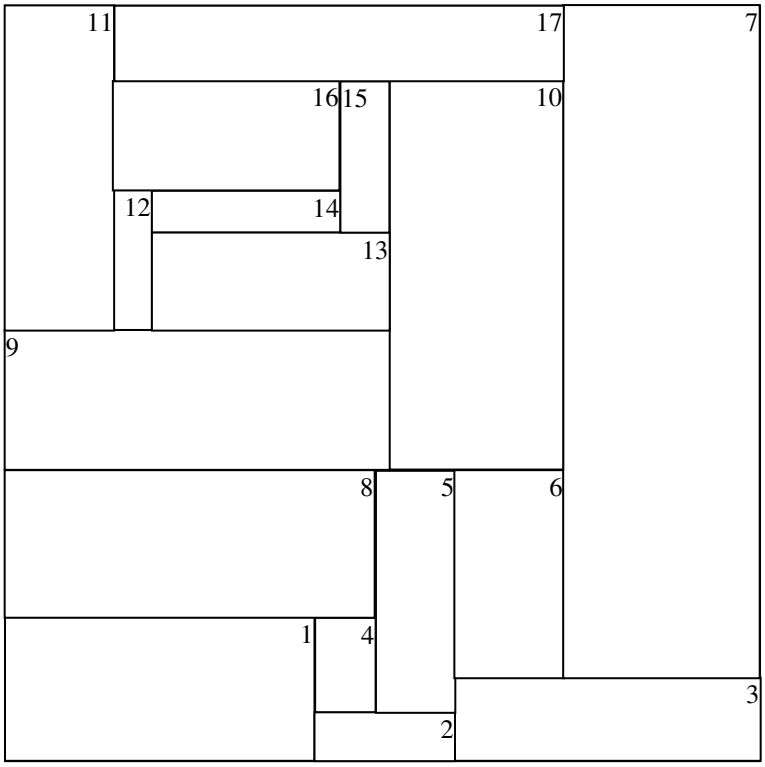
Msa150 : Agencement selon la politique BLF2G.
L’ordre des pièces donné par S. Abou Msabah

Annexe B :

Jeux de tests trouvés dans la littérature pour le problème de Bin Packing rectangulaire :

Référence	Nom	Taille	Optimal
[Bur99a]	**Kendall**	**13**	**140 x 80**
[Rat97b]	**D2**	**21**	**40 x 60**
[Jak96]	**J1**	**25**	**40 x 15**
[Jak96]	**J2**	**50**	**40 x 15**
[Hop99]	**T1a, T1b, T1c, T1d, T1e** **N1a, N1b, N1c, N1d, N1e**	**17**	**200 x 200**
[Hop99]	**T2a, T2b, T2c, T2d, T2e** **N2a, N2b, N2c, N2d, N2e**	**25**	**200 x 200**
[Hop99]	**T3a, T3b, T3c, T3d, T3e** **N3a, N3b, N3c, N3d, N3e**	**29**	**200 x 200**
[Hop99]	**T4a, T4b, T4c, T4d, T4e** **N4a, N4b, N4c, N4d, N4e**	**49**	**200 x 200**
[Hop99]	**T5a, T5b, T5c, T5d, T5e** **N5a, N5b, N5c, N5d, N5e**	**73**	**200 x 200**
[Hop99]	**T6a, T6b, T6c, T6d, T6e** **N6a, N6b, N6c, N6d, N6e**	**97**	**200 x 200**
[Hop99]	**T7a, T7b, T7c, T7d, T7e** **N7a, N7b, N7c, N7d, N7e**	**197**	**200 x 200**

	name:	**T1a**	**name:**	**T1b**	**name:**	**T1c**	**name:**	**T1d**	**name:**	**T1e**
	size:	17	size:	17	size:	17	size:	17	size:	17
Optimum:	200	200	200	200	200	200	200	200	200	200
no	**width**	**height**	**width**	**height**	**width**	**height**	**width**	**height**	**width**	**height**
1	82	38	41	12	167	40	178	58	28	139
2	37	13	25	34	33	125	22	153	151	25
3	81	22	19	44	20	70	46	142	21	44
4	16	25	115	22	111	84	59	62	44	60
5	21	64	25	51	7	20	73	44	107	19
6	29	55	16	22	29	18	20	18	121	21
7	52	178	71	22	8	2	32	20	7	32
8	98	39	44	109	21	67	21	34	95	20
9	102	37	41	29	15	65	74	11	26	11
10	46	103	90	87	13	90	5	19	33	9
11	29	86	35	137	7	14	22	31	78	54
12	10	37	31	68	118	76	10	14	31	78
13	63	26	129	44	37	6	46	22	63	83
14	50	11	36	15	32	11	28	8	106	61
15	13	40	17	29	18	69	31	17	11	37
16	60	29	19	54	19	5	33	14	20	5
17	119	20	146	25	51	64	154	47	83	32



T1a : Agencement selon la politique BLF non guillotinable
L'ordre des pièces donné par E. Hopper

	name:	**T2a**	**name:**	**T2b**	**name:**	**T2c**	**name:**	**T2d**	**name:**	**T2e**
	size:	25	size:	25	size:	25	size:	25	size:	25
Optimum:	200	200	200	200	200	200	200	200	200	200
no	width	height	width	height	width	height	width	height	width	height
1	114	36	82	39	11	76	14	50	37	71
2	22	75	45	56	58	71	67	13	61	28
3	64	150	73	20	42	34	51	57	90	32
4	13	55	21	35	89	21	68	44	12	48
5	81	31	17	9	14	13	38	37	7	43
6	16	3	35	180	75	44	29	96	37	10
7	4	24	8	26	56	31	22	13	17	33
8	12	36	9	44	20	45	46	75	64	94
9	4	21	22	34	29	6	52	59	26	16
10	8	15	60	17	11	21	33	40	26	112
11	44	24	29	18	17	112	40	22	11	23
12	37	46	105	17	54	69	21	18	38	78
13	21	31	86	33	24	5	19	40	28	89
14	21	6	28	17	34	98	54	22	44	79
15	5	25	51	84	11	27	47	24	50	24
16	16	32	11	16	18	15	21	67	43	44
17	57	22	17	41	35	93	13	10	9	50
18	26	7	97	25	29	12	97	36	46	18
19	56	87	15	49	18	12	35	67	102	32
20	80	37	30	43	22	71	17	67	29	32
21	27	34	69	26	38	59	30	43	17	14
22	117	17	120	43	40	43	81	45	26	30
23	55	17	23	6	14	66	16	31	17	6
24	62	33	7	26	129	31	51	24	26	24
25	82	16	38	20	57	23	51	14	119	18

	name:	**T3a**	**name:**	**T3b**	**name:**	**T3c**	**name:**	**T3d**	**name:**	**T3e**
	size:	29	size:	29	size:	29	size:	29	size:	29
Optimum:	200	200	200	200	200	200	200	200	200	200
no	**Width**	**height**	**width**	**height**	**width**	**height**	**width**	**height**	**width**	**height**
1	58	12	27	154	112	22	37	71	8	47
2	22	20	159	23	18	47	61	28	143	26
3	20	101	14	35	61	25	90	32	16	107
4	54	14	78	32	9	38	12	48	33	95
5	14	40	81	12	41	8	7	43	5	21
6	32	63	95	20	50	12	37	10	54	40
7	30	22	85	39	21	25	17	33	84	15
8	28	8	32	53	26	57	64	94	32	25
9	33	54	10	59	35	13	26	16	52	45
10	21	26	23	52	35	33	26	112	13	39
11	50	14	23	95	6	4	11	23	86	20
12	20	132	17	60	56	29	38	78	50	77
13	16	48	38	29	44	44	28	89	77	31
14	44	16	30	18	39	16	44	79	24	74
15	35	28	15	93	24	94	50	24	11	12
16	33	32	8	43	42	60	43	44	22	65
17	11	51	10	25	64	19	9	50	27	53
18	19	5	22	6	86	60	46	18	14	46
19	13	48	17	11	17	11	102	32	17	9
20	87	43	13	21	31	5	29	32	15	21
21	49	19	32	19	21	6	17	14	31	6
22	64	21	55	10	10	41	7	15	8	38
23	16	33	110	21	38	35	19	7	23	77
24	65	55	10	41	14	34	17	6	12	35
25	35	89	21	22	28	77	26	24	5	12
26	17	44	154	46	25	14	8	8	20	23
27	47	12	7	19	23	72	11	23	73	40
28	63	32	14	28	111	58	119	18	40	39
29	165	34	17	9	38	43	15	15	64	37

	name:	**T4a**	**name:**	**T4b**	**name:**	**T4c**	**name:**	**T4d**	**name:**	**T4e**
	size:	49	size:	49	size:	49	size:	49	size:	49
Optimum:	200	200	200	200	200	200	200	200	200	200
no	**width**	**height**	**width**	**height**	**width**	**height**	**width**	**height**	**width**	**height**
1	26	25	54	27	25	105	37	57	110	32
2	9	28	109	16	57	24	16	63	12	33
3	47	23	37	102	27	18	48	23	78	15
4	18	42	33	11	13	25	50	41	62	18
5	13	52	76	35	20	15	49	86	16	53
6	45	94	46	15	58	20	27	53	37	76
7	8	40	41	6	14	10	21	18	14	97
8	34	12	14	9	6	28	64	14	28	43
9	11	28	27	18	14	59	7	18	16	9
10	23	63	60	9	13	25	38	21	15	22
11	28	5	35	46	17	57	26	4	64	19
12	19	17	56	18	41	82	25	77	10	13
13	7	33	22	92	32	81	12	6	4	27
14	19	15	45	19	25	102	33	17	12	13
15	37	12	5	30	27	18	28	71	3	6
16	75	18	20	28	46	34	93	20	7	22
17	19	35	36	56	77	25	5	10	67	16
18	6	37	15	32	8	15	54	10	27	14
19	12	10	30	11	25	18	23	18	13	41
20	25	27	17	13	64	13	10	30	78	35
21	56	41	18	11	7	19	58	29	30	115
22	4	14	5	2	14	74	14	41	8	54
23	22	6	13	10	57	21	24	13	20	91
24	8	8	22	8	58	11	18	74	6	6
25	14	15	55	28	6	6	15	72	72	12
26	12	7	16	51	4	9	9	61	15	92
27	32	19	71	38	4	3	18	20	22	58
28	10	68	51	75	29	6	5	12	19	6
29	57	20	40	18	13	5	27	23	38	48
30	52	35	32	13	105	22	31	12	21	61
31	25	26	7	42	10	45	15	8	10	38
32	34	101	27	23	39	74	20	20	22	47
33	5	29	5	26	6	31	32	9	22	37
34	36	9	50	37	26	21	34	51	5	30
35	38	20	12	45	13	10	45	11	5	9
36	6	20	11	37	13	29	6	11	3	15
37	30	22	37	32	51	52	26	31	24	10
38	20	11	10	8	54	23	55	52	20	37
39	18	24	17	3	19	19	17	33	18	13
40	10	9	22	5	44	8	26	20	23	19
41	15	23	25	11	11	11	16	3	41	27
42	11	2	7	29	9	9	11	7	18	5
43	62	14	31	20	46	24	4	8	6	11
44	61	13	19	8	36	21	12	4	39	24
45	104	18	32	18	8	8	29	26	21	6
46	62	20	25	5	7	2	23	11	62	17
47	91	39	12	15	2	7	23	4	20	15
48	13	2	31	12	18	5	59	19	3	8
49	75	37	36	10	28	13	57	15	44	7

	name:	**T5a**	**name:**	**T5b**	**name:**	**T5c**	**name:**	**T5d**	**name:**	**T5e**
	size:	73	size:	73	size:	73	size:	73	size:	73
Optimum:	200	200	200	200	200	200	200	200	200	200
no	**width**	**height**	**width**	**height**	**width**	**height**	**width**	**height**	**width**	**height**
1	54	12	36	24	112	22	18	43	66	25
2	39	23	26	4	18	47	55	12	14	30
3	17	58	14	3	61	25	16	41	18	14
4	90	30	31	26	9	38	11	23	43	34
5	12	32	23	83	41	8	29	20	59	30
6	42	11	52	10	50	12	15	81	10	16
7	81	21	18	37	21	25	30	27	8	33
8	58	20	6	1	26	57	17	23	22	47
9	4	23	8	11	35	13	9	26	18	16
10	25	9	32	10	35	33	15	31	26	13
11	3	13	9	38	6	4	40	73	24	17
12	14	35	43	27	56	29	1	3	26	4
13	11	4	4	10	12	13	28	14	7	28
14	9	15	36	103	32	9	12	11	26	14
15	5	23	23	29	39	16	15	7	30	51
16	15	21	17	35	10	4	2	3	39	25
17	18	10	18	100	22	35	11	4	7	3
18	13	88	13	75	22	31	18	22	19	9
19	5	19	61	11	24	94	12	3	25	6
20	42	10	7	35	13	11	38	19	8	14
21	49	8	32	23	29	8	36	7	18	78
22	9	3	11	4	64	19	4	21	14	17
23	13	5	20	7	5	3	52	14	62	12
24	4	11	9	9	24	52	6	10	15	64
25	14	60	2	3	18	49	7	11	16	5
26	37	9	14	64	34	23	20	8	46	13
27	5	20	9	6	52	43	32	32	27	26
28	79	16	22	6	5	7	24	76	12	63
29	5	16	26	58	12	2	12	15	30	8
30	4	8	10	10	31	5	8	34	36	91
31	42	11	11	17	8	5	2	13	23	31
32	19	49	10	35	4	9	4	1	8	13
33	9	8	6	8	21	6	11	12	22	9
34	33	22	36	21	10	41	9	30	9	50
35	34	6	4	23	13	4	13	39	14	4
36	27	12	7	18	38	35	34	9	8	16
37	7	23	4	22	29	37	20	30	22	12
38	4	21	2	13	5	20	14	35	57	37
39	26	7	30	18	14	34	25	19	7	6
40	23	4	38	9	8	38	31	17	12	36
41	9	3	17	5	20	28	16	30	11	29
42	14	59	106	25	57	17	82	14	30	56
43	18	16	14	83	25	14	33	5	50	62
44	16	24	62	9	23	72	4	27	61	13
45	27	49	88	26	51	17	27	13	4	7
46	8	47	8	35	60	28	97	16	7	33
47	8	23	90	30	2	10	20	56	16	26
48	19	14	60	21	8	17	12	26	12	46

49	3	2	28	9	10	36	43	14	21	27
50	1	5	5	17	19	25	64	17	28	65
51	51	8	28	22	19	10	25	74	4	4
52	10	3	57	11	19	41	8	40	9	19
53	30	9	36	12	32	11	24	10	40	13
54	28	96	30	11	10	39	7	45	6	19
55	9	31	11	9	6	6	52	17	15	22
56	30	19	16	28	7	2	47	68	19	31
57	18	55	23	4	6	33	17	57	21	17
58	20	28	20	7	5	32	11	53	5	6
59	23	86	58	11	3	19	13	35	5	15
60	10	18	5	24	4	4	25	4	30	7
61	9	12	6	19	3	13	7	31	6	19
62	21	41	17	10	24	21	20	58	8	12
63	4	2	6	3	68	11	9	42	11	8
64	4	9	17	12	10	9	16	27	19	21
65	31	7	41	9	24	10	17	11	18	14
66	23	68	26	7	44	19	11	23	3	20
67	18	29	10	3	26	7	10	34	18	3
68	69	27	48	16	6	11	7	12	83	16
69	91	14	53	13	13	13	20	18	16	13
70	35	23	8	3	2	11	23	15	2	7
71	61	22	33	13	24	4	18	22	6	10
72	30	9	25	10	48	9	63	12	37	6
73	65	13	22	5	30	7	42	11	8	3

	name:	**T6a**	**Name:**	**T6b**	**name:**	**T6c**	**name:**	**T6d**	**name:**	**T6e**
	size:	97	size:	97	size:	97	size:	97	size:	97
Optimum:	200	200	200	200	200	200	200	200	200	200
no	**width**	**height**	**width**	**height**	**width**	**height**	**width**	**height**	**width**	**height**
1	101	37	28	7	18	90	13	9	15	72
2	99	25	9	8	15	63	22	7	15	47
3	10	12	8	6	23	89	22	64	67	11
4	14	80	59	22	108	19	11	12	56	39
5	55	14	23	10	17	45	35	11	6	35
6	20	19	7	5	19	106	36	14	11	17
7	50	10	27	8	12	30	21	55	8	20
8	9	11	10	45	22	14	18	116	22	8
9	15	18	29	32	74	26	22	56	8	12
10	14	17	3	14	5	16	10	2	3	16
11	23	14	4	3	17	41	12	27	11	9
12	31	19	4	2	9	16	23	25	9	61
13	24	5	4	14	82	12	5	1	47	7
14	8	10	8	21	17	25	30	42	11	11
15	36	9	20	13	3	4	16	41	5	18
16	22	59	13	12	79	17	16	68	6	22
17	28	14	31	11	12	13	7	20	5	7
18	2	13	18	41	7	27	13	6	6	14
19	7	7	5	9	8	32	4	14	25	21
20	9	3	26	8	33	5	9	35	22	4
21	14	45	13	18	4	17	27	94	16	11
22	4	1	37	8	20	5	8	46	33	17
23	32	5	20	6	48	13	11	21	8	7
24	23	42	30	10	10	20	7	11	4	8
25	12	4	9	31	10	32	31	20	26	110
26	22	6	19	24	25	22	8	2	11	4
27	15	23	7	10	8	12	4	18	58	12
28	16	16	65	29	9	12	4	25	56	21
29	16	40	24	25	3	20	41	27	21	102
30	10	56	6	21	8	36	10	60	6	25
31	18	94	4	13	14	9	12	75	9	51
32	15	40	25	57	26	7	29	16	16	11
33	37	35	20	14	8	7	35	7	30	5
34	7	7	11	18	9	6	12	7	12	9
35	9	31	3	19	14	73	7	11	4	6
36	22	24	12	59	25	5	4	24	26	16
37	1	5	23	70	12	10	47	12	68	12
38	36	10	22	38	5	10	6	20	20	10
39	22	10	15	4	4	8	34	102	21	26
40	15	104	23	70	9	2	23	49	20	12
41	12	21	19	57	17	11	8	11	49	16
42	4	16	13	3	18	11	4	4	6	28
43	16	5	29	8	35	11	11	7	11	3
44	23	42	20	18	7	29	14	10	37	12
45	22	49	6	12	23	9	6	5	9	18
46	25	11	7	5	7	2	27	8	2	9
47	44	12	7	26	49	14	4	5	5	14

48	16	40	4	1	30	10	2	10	15	4
49	11	48	7	25	28	40	5	21	39	9
50	14	5	36	7	21	48	14	6	64	10
51	17	6	23	59	11	44	33	7	32	7
52	2	10	18	50	3	18	18	5	16	26
53	6	1	1	3	13	13	18	15	99	21
54	50	11	14	40	16	16	5	9	38	19
55	15	81	7	46	87	30	28	5	23	32
56	11	33	20	40	5	26	20	21	6	37
57	8	6	37	7	8	3	13	4	22	54
58	9	4	4	23	24	23	15	7	25	36
59	11	2	32	27	10	38	18	3	50	27
60	22	65	5	16	22	6	28	15	27	22
61	13	35	5	30	23	9	56	9	8	41
62	18	47	7	11	67	16	14	4	66	23
63	16	54	9	11	32	24	4	25	8	5
64	10	37	5	13	1	4	56	19	15	3
65	6	8	14	86	10	12	16	14	18	12
66	20	53	8	25	6	11	13	56	9	4
67	3	7	6	6	16	3	13	34	6	2
68	2	10	9	7	22	8	50	24	9	6
69	19	4	4	14	39	8	7	33	5	8
70	8	35	16	9	8	20	7	21	4	18
71	17	29	14	5	4	26	7	2	15	19
72	25	46	2	9	55	8	2	4	8	5
73	8	6	16	18	63	18	7	22	14	4
74	11	12	15	6	18	5	5	3	48	13
75	10	6	26	17	20	4	2	2	14	14
76	5	16	8	4	4	1	4	1	23	10
77	8	12	6	14	6	13	65	17	5	18
78	8	32	28	73	10	12	11	12	10	23
79	13	19	7	12	22	22	13	10	51	11
80	26	4	2	7	49	15	37	29	58	14
81	4	14	35	11	35	11	6	1	7	12
82	16	3	24	10	3	13	55	15	10	9
83	11	12	25	9	1	1	8	5	16	12
84	3	17	6	5	9	11	3	15	13	3
85	13	9	62	22	7	10	24	35	35	9
86	34	6	49	8	35	20	14	11	34	12
87	9	5	47	19	28	9	12	8	17	25
88	21	13	7	14	31	11	6	22	3	3
89	5	1	24	20	4	2	2	10	7	23
90	4	12	18	25	7	19	8	3	10	20
91	39	11	6	21	1	6	4	11	109	18
92	24	8	41	41	7	9	22	8	7	7
93	6	4	69	18	47	16	19	21	6	6
94	20	3	2	12	2	7	36	12	44	13
95	5	1	22	5	5	13	5	12	3	1
96	15	3	40	7	66	11	110	16	3	12
97	11	2	117	20	40	9	47	9	10	11

	name: T7a		name: T7b		name: T7c		name: T7d		name: T7e	
	size: 199		size: 199		size: 199		size: 199		size: 199	
Optimum :	200	200	200	200	200	200	200	200	200	200
No	**width**	**height**	**width**	**height**	**width**	**height**	**width**	**height**	**width**	**height**
1	18	11	19	50	11	3	14	46	15	69
2	23	11	8	50	4	3	28	46	13	28
3	17	35	8	8	8	11	29	7	6	28
4	21	35	9	8	44	9	34	7	11	27
5	25	10	8	2	3	9	10	33	5	27
6	5	1	10	2	19	20	23	49	59	17
7	7	1	10	29	6	20	20	86	29	9
8	76	12	17	22	50	8	19	31	11	9
9	8	12	32	22	2	2	23	31	9	4
10	5	9	9	22	2	2	3	6	15	4
11	7	9	6	9	51	16	26	6	4	24
12	9	13	21	9	2	6	21	19	5	24
13	16	13	3	15	2	6	4	19	18	51
14	12	22	10	5	11	8	9	22	9	58
15	5	4	30	5	4	8	3	20	15	58
16	13	4	8	11	50	8	21	14	29	8
17	16	19	10	11	4	8	5	14	11	8
18	7	19	10	10	44	8	21	3	30	14
19	46	8	30	10	3	8	4	3	29	14
20	30	8	8	5	15	6	21	6	10	66
21	8	20	9	5	8	6	5	6	2	4
22	5	31	6	13	54	8	25	4	9	4
23	13	31	21	13	51	8	9	4	3	10
24	23	7	17	64	5	22	8	14	6	19
25	23	7	10	26	6	22	9	6	10	19
26	30	12	8	26	12	46	2	6	2	6
27	9	9	3	18	24	8	11	20	9	6
28	16	9	40	18	23	8	8	11	4	27
29	23	5	17	7	19	4	4	11	5	27
30	23	5	32	7	6	4	63	16	11	8
31	16	16	9	11	8	17	7	9	5	8
32	7	16	27	11	17	17	3	9	11	20
33	21	12	4	23	4	11	9	8	3	20
34	10	12	6	23	19	11	2	8	13	7
35	6	19	14	7	10	14	7	7	6	7
36	32	39	35	7	20	11	3	7	9	44
37	9	39	10	6	52	11	8	9	3	21
38	43	45	3	6	6	21	4	9	9	21
39	17	11	5	11	7	12	8	41	9	35
40	5	7	14	15	11	12	11	41	12	38
41	16	7	4	15	7	13	10	12	7	38
42	5	4	12	74	2	13	4	12	10	53
43	16	4	12	29	4	16	21	15	9	3
44	21	7	19	29	10	23	7	15	10	3
45	10	7	14	41	4	3	73	18	7	15
46	13	19	22	8	19	3	23	18	9	15
47	11	11	13	8	20	9	11	35	6	11

48	17	11	10	38	52	9	12	35	10	11
49	6	18	8	38	7	9	10	9	9	31
50	32	18	10	5	11	9	4	9	10	31
51	19	28	3	5	7	3	21	6	4	21
52	9	9	22	33	2	3	7	6	10	21
53	3	9	5	13	23	6	5	10	5	8
54	6	35	8	13	10	6	16	10	11	8
55	11	8	8	10	5	24	6	6	7	19
56	17	8	3	9	6	24	15	6	9	19
57	9	19	2	9	9	7	42	6	9	11
58	3	19	5	31	4	7	28	6	18	11
59	6	22	14	27	8	42	10	8	3	14
60	19	5	4	27	11	18	7	3	9	14
61	13	5	10	41	6	18	9	3	5	28
62	13	21	9	41	20	5	7	5	11	28
63	6	13	8	52	13	5	9	5	16	24
64	5	6	4	25	12	12	6	9	8	24
65	5	4	6	25	18	12	15	9	6	31
66	12	4	3	1	26	14	20	5	4	3
67	5	2	2	1	9	11	22	5	17	3
68	12	2	8	21	7	11	7	5	4	28
69	19	17	5	21	6	37	21	5	9	13
70	13	17	5	20	18	37	10	11	8	13
71	5	7	8	20	13	35	7	7	12	9
72	17	7	12	45	10	35	9	7	9	9
73	32	6	19	45	20	34	5	5	4	15
74	9	6	18	41	13	34	16	5	10	15
75	41	9	18	41	9	3	20	8	15	71
76	43	9	7	12	7	3	22	8	9	37
77	6	8	28	12	12	9	7	8	26	37
78	22	8	6	13	18	9	21	8	12	15
79	19	7	25	9	26	7	5	19	7	15
80	12	7	28	9	16	7	16	19	9	26
81	60	17	25	4	11	24	21	49	9	9
82	19	17	28	4	6	24	7	4	5	9
83	10	50	3	4	8	10	9	4	7	14
84	24	30	4	4	3	10	26	35	9	15
85	5	30	7	9	12	20	31	11	8	15
86	28	34	21	9	12	13	13	11	10	18
87	27	8	6	12	18	13	4	15	30	18
88	27	8	5	3	42	18	3	15	9	5
89	9	4	17	3	8	10	10	5	5	5
90	18	4	31	7	3	10	9	5	19	17
91	7	18	10	11	12	5	10	13	10	17
92	3	10	9	11	18	5	10	13	16	15
93	12	10	3	9	44	11	15	6	8	15
94	5	12	4	9	11	11	27	6	14	12
95	9	62	5	4	12	20	10	15	7	12
96	9	47	17	4	3	20	9	15	6	8
97	9	47	22	5	8	21	7	16	21	8
98	11	7	31	5	6	21	8	16	24	10
99	29	7	7	4	18	13	10	10	17	10
100	4	22	21	4	49	13	5	5	50	8
101	2	11	27	14	13	26	12	5	8	8

102	5	11	94	14	27	26	31	24	14	16
103	9	16	12	9	9	36	7	11	3	5
104	19	97	31	9	13	13	6	11	14	5
105	3	2	4	4	31	13	5	5	9	13
106	12	2	8	4	11	18	12	5	2	13
107	15	6	17	9	18	11	10	32	7	12
108	5	6	25	12	49	11	5	12	2	12
109	11	15	24	12	12	9	5	12	9	34
110	29	15	4	10	3	9	5	30	11	21
111	7	48	20	10	8	3	16	30	15	21
112	8	41	19	16	6	3	4	5	3	11
113	12	41	29	45	13	5	3	5	14	11
114	2	5	28	5	31	5	4	18	21	8
115	5	5	15	5	14	21	6	18	29	8
116	24	4	7	32	7	7	17	29	8	21
117	5	4	4	5	35	7	7	25	24	19
118	3	6	8	5	11	8	5	14	12	17
119	3	3	5	15	14	8	14	14	5	17
120	1	3	23	15	13	10	4	5	7	4
121	9	14	6	18	27	10	3	5	2	4
122	22	4	9	18	10	9	6	13	9	3
123	7	4	12	10	29	9	7	23	2	3
124	19	3	17	10	6	31	8	23	11	10
125	9	3	4	6	10	31	5	20	10	10
126	3	3	20	6	15	88	5	20	29	13
127	1	3	25	7	7	8	4	8	14	9
128	17	4	24	7	35	8	3	8	17	9
129	2	4	6	19	11	7	5	11	11	9
130	9	13	9	19	14	7	14	11	9	9
131	18	4	9	21	40	9	4	11	6	6
132	4	4	19	38	9	9	6	11	25	6
133	7	12	7	19	10	40	26	10	20	14
134	40	8	10	3	29	40	19	4	4	7
135	4	8	12	3	42	6	25	4	7	7
136	3	8	28	27	25	6	19	6	15	13
137	4	8	9	18	22	38	25	6	11	3
138	12	4	12	18	28	15	12	24	10	3
139	5	4	5	12	5	6	22	10	12	2
140	2	9	23	12	26	6	35	10	5	2
141	18	8	10	16	7	2	6	20	41	10
142	4	8	12	16	6	2	5	20	58	10
143	12	5	6	9	2	4	15	8	6	8
144	5	5	9	9	13	8	22	8	25	8
145	13	44	43	13	21	8	21	16	4	6
146	13	37	7	13	7	2	19	15	7	6
147	7	11	6	2	6	2	16	15	37	7
148	11	11	9	2	13	8	8	10	13	7
149	16	67	9	9	2	8	4	4	60	11
150	4	14	12	9	5	9	15	4	15	11
151	6	14	15	17	26	9	4	6	11	20
152	39	6	9	17	13	4	15	6	18	9
153	18	6	7	16	21	4	15	8	9	9
154	39	18	6	13	5	11	22	8	28	5
155	18	18	8	3	2	11	22	14	9	5

156	7	26	8	3	8	15	35	14	28	13
157	11	26	8	10	8	41	8	8	9	13
158	9	15	8	10	10	19	19	8	21	36
159	9	15	4	27	16	19	19	3	10	26
160	4	10	25	27	6	18	16	3	6	26
161	6	10	4	15	7	3	37	35	13	53
162	8	7	3	15	3	3	11	18	18	9
163	12	7	20	24	28	23	3	16	9	9
164	23	7	9	30	31	23	7	16	60	9
165	17	7	4	24	7	15	17	24	15	9
166	27	14	10	24	3	15	18	24	27	19
167	21	21	28	8	5	4	13	35	33	14
168	28	16	20	4	2	4	14	35	4	14
169	5	16	1	4	7	45	6	21	75	17
170	23	7	20	4	8	45	5	21	11	17
171	17	7	1	4	10	22	8	15	33	5
172	13	7	6	3	16	22	4	15	4	5
173	18	7	16	3	30	6	57	17	10	10
174	33	6	3	19	9	6	3	2	6	10
175	7	6	27	8	12	22	7	2	62	10
176	27	26	48	8	4	22	11	17	24	10
177	28	5	17	17	15	28	10	17	12	12
178	5	5	26	17	7	28	8	2	19	12
179	5	10	4	9	10	6	4	2	17	7
180	8	10	3	9	7	6	69	10	5	3
181	31	23	27	11	42	19	9	4	11	3
182	6	7	48	11	15	15	2	4	5	4
183	3	7	7	10	15	15	17	27	11	4
184	24	8	20	10	9	33	11	17	21	17
185	7	20	4	6	10	13	7	17	16	17
186	16	14	10	6	7	13	9	6	17	8
187	5	14	17	13	8	19	2	6	12	2
188	21	3	26	13	26	19	80	18	4	2
189	12	3	4	12	15	18	37	8	12	6
190	21	16	13	8	15	18	21	8	4	6
191	12	16	12	8	12	17	13	16	62	13
192	5	13	3	11	4	17	14	16	24	13
193	8	13	75	11	17	15	18	4	12	3
194	6	1	9	4	40	7	19	4	19	3
195	3	1	14	4	2	7	21	10	15	4
196	9	12	27	5	40	8	11	10	16	4
197	24	12	23	5	2	8	7	10	33	8
198	16	5	13	4	15	6	18	6	15	4
199	5	5	12	4	7	6	19	6	16	4

	name:	**MsaPFE95**	
	size:		
	Optimum:	not know	
Bin Size:	1250	2500	
no	**width**	**height**	**Quantity**
3025/A	172	2342	1
3026/A	472	2342	1
3025	142	2342	1
3026	142	2342	2
3092	96	2310	1
3045	66	2290	1
3001	320	2180	3
3004	205	2180	6
3037	374	2062	1
3017	177	2062	1
3046	66	2010	2
3060	61	2000	4
3061	61	2000	4
3082	1000	1620	2
3080	947	1620	10
3081	924	1620	6
3083	754	1620	2
3029	164	1562	17
3013	320	1320	3
3014	320	1320	3
3015	205	1320	1
3016	205	1320	1
3040'	205	1320	2
3041	66	1280	2
3087	947	1230	2
3088	924	1230	1
3089	777	1230	1
3021/A	172	1192	1
3022/A	142	1192	1
3022	142	1192	2
3021	142	1192	1
3041'	66	1136	1
3084	947	1100	2
3084	924	1100	4
3086	777	1100	2
3019/A	172	1062	2
3020/A	172	1062	2
3019	142	1062	2
3020	142	1062	4
3040	66	1006	2
3058	61	994	4
3059	61	994	4
3052	61	994	8
3046	66	636	2
30110	250	590	2
30111	205	590	2
3039	432	586	2
3045	66	556	1
30120	66	535	2
3037/A	322	512	1
3093	142	512	2
3079	16	200	8
30120'	70	134	2
30106	30	60	2

Nom: **J1**
Taille: 25
Optimum: 15 X 40

no	Largeur	Longueur
1	12	6
2	4	7
3	6	7
4	10	2
5	2	5
6	6	4
7	4	2
8	4	6
9	7	9
10	4	5
11	6	4
12	4	6
13	6	3
14	4	5
15	2	4
16	8	4
17	8	6
18	8	3
19	6	3
20	2	6
21	8	2
22	3	5
23	2	5
24	3	4
25	2	4

Nom: **J2**
Taille: 50
Optimum: 15 X 40

no	Largeur	Longueur
1	5	6
2	7	6
3	4	3
4	4	4
5	6	4
6	6	3
7	4	2
8	6	2
9	3	4
10	3	4
11	2	5
12	4	2
13	3	3
14	3	6
15	4	3
16	4	6
17	4	3
18	4	3
19	4	2
20	4	4
21	4	2
22	4	3
23	3	4
24	3	4
25	2	5
26	2	5
27	2	4
28	3	6
29	5	2
30	5	4
31	3	3
32	5	3
33	2	3
34	4	3
35	2	3
36	4	3
37	2	2
38	2	4
39	3	4
40	3	4
41	2	4
42	3	2
43	3	2
44	2	2
45	3	2
46	2	2
47	3	3
48	2	3
49	3	4
50	2	4

	name:	**D2**	
	size:	21	
	Optimum:	40 x 60	
no.	**width**	**height**	**quantity**
1	12	12	4
2	12	10	5
3	12	9	6
4	12	8	6

	name:	**Kendall**	
	size:	13	
	Optimum:	140 x 80	
no.	**width**	**height**	**quantity**
1	24	16	1
2	28	16	2
3	60	14	2
4	20	28	1
5	22	26	2
6	42	44	1
7	18	70	1
8	62	26	1
9	18	48	2