RAIRO-Oper. Res. 43 (2009) 87–101
DOI: 10.1051/ro/2009006

RAIRO Operations Research
www.rairo-ro.org

OPTIMISATION HYBRIDE PAR COLONIES DE FOURMIS POUR LE PROBLÈME DE DÉCOUPE À DEUX DIMENSIONS

Alice Yalaoui[1] et Chengbin Chu[1]

Abstract. In this paper, we are interested in the guillotine bin packing problem (2BP/O/G). The aim of this problem is to cut a set of rectangular pieces (items) from a set of identical large rectangular pieces (stock sheets). We try to minimize the number of stock sheets used in order to satisfy the demand. This is done applying edge to edge cut, which is called the guillotine constraint. We propose in this paper a resolution approach using Ant Colony Optimization (ACO) and the SHF-FF heuristic of Ben Messaoud *et al.* [2] in order to solve this NP-hard problem. We improve the best known results.

Résumé. Nous nous intéressons dans cet article au problème de découpe guillotine en deux dimensions noté 2BP/O/G. Il s'agit de découper un certain nombre de pièces rectangulaires dans un ensemble de plaques de matière première, elles même rectangulaires et identiques. Celles-ci sont disponibles en quantité illimitée. L'objectif est de minimiser le nombre de plaques utilisées pour satisfaire la demande, en appliquant une succession de coupes, dites guillotines, allant de bout en bout. Nous proposons une approche de résolution combinant l'optimisation par colonies de fourmis (ACO) et l'heuristique SHF-FF de Ben Messaoud *et al.* [2] pour résoudre ce problème NP-difficile.

Mots Clés. Colonies de fourmis, découpe, guillotine, optimisation, *Bin packing*.

Classification Mathématique. 05B40, 52C15, 90C27.

Reçu le 6 février , 2006. Accepté le 16 septembre , 2008.

[1] ICD, Université de Technologie de Troyes, 12 rue Marie Curie, BP 2060, 10010 Troyes Cedex, France ; alice.yalaoui@utt.fr ; chengbin.chu@utt.fr

1. Introduction

On rencontre des problèmes de découpe dans de nombreux domaines tels que, l'industrie du papier, du verre, du textile, mais aussi lorsqu'il s'agit du remplissage de volumes tels que des wagons ou encore pour la gestion des zones mémoire. Dans un tel problème, on cherche à découper à partir de la matière première un ensemble de pièces, qui peuvent avoir des formes et dimensions variées, de telle sorte que l'on minimise un critère donné. Il s'agit le plus souvent de minimiser la quantité de matière première utilisée, ce qui revient à minimiser les chutes engendrées. En fonction de la nature du produit à découper et du processus de découpe en lui même, ces problèmes peuvent avoir différentes caractéristiques et comporter certaines contraintes. Dyckhoff [6] a proposé une classification de l'ensemble des problèmes de découpe et placement.

Nous nous intéressons ici aux problèmes de découpe à deux dimensions, et plus particulièrement aux problèmes de *Bin packing* et de *Strip packing* [9,10]. Dans le premier, on cherche à placer un ensemble de pièces dans plusieurs plaques rectangulaires (*bins*). L'objectif est de trouver un ensemble de grands rectangles (ou plaques) pouvant contenir toutes les pièces en minimisant la matière totale utilisée, c'est-à-dire le nombre de plaques. Ce problème est connu sous le nom de *Bin packing* à deux dimensions (2BP). En ce qui concerne le problème de *Strip packing*, la problématique est la même mais l'on dispose d'une bande de largeur fixe et de longueur infinie (*strip*) dans laquelle on doit découper les pièces. Pour ces deux problèmes, on peut considérer différentes contraintes supplémentaires comme l'impossibilité de rotation de 90° des pièces et la contrainte guillotine. Cette dernière est imposée par l'outil de coupe, lorsqu'une coupe doit être faite de bout en bout de la plaque. Les pièces doivent alors être disposées sur la plaque de telle sorte que l'on puisse les extraire par une succession de coupes dites guillotines. En ce qui concerne la non rotation des pièces, cette contrainte peut être rencontrée dans le cas de la découpe du papier ou du tissu avec des motifs par exemple. Ces problèmes sont NP-difficiles [9].

On peut classer les algorithmes de résolution de ces problèmes en deux catégories : ceux à une phase et ceux à deux phases. Dans le premier cas, les objets sont placés directement dans les plaques et dans le second cas, on commence par les placer dans une bande (*strip*) de même largeur que les plaques, puis on utilise la solution obtenue pour construire une solution pour le problème initial en utilisant une heuristique dédiée au problème de *Bin Packing* à une dimension. Beaucoup de ces approches sont dites par couches. Dans ces approches, le rangement des objets est obtenu en plaçant les objets de gauche à droite, dans des rangées formant des couches. Les niveaux sont définis en fonction de la taille de l'objet le plus grand dans chaque couche. Dans ce genre d'algorithmes, les objets sont initialement rangés dans l'ordre décroissant de leur taille. On rencontre ensuite plusieurs stratégies de placement comme *Next-Fit Decreasing Height* (NFDH), *First-Fit Decreasing Height* (FFDH) ou encore *Best-Fit Decreasing Height* (BFDH) [10]. Pour le problème du *Bin-packing* à deux dimensions, citons, en ce qui concerne

TABLEAU 1. Différentes stratégies de placement.

Abréviation	Nom	Placement de l'objet courant
NFDH	Next-Fit Decreasing Height	Le plus à gauche possible dans la couche courante si possible, sinon, la couche est fermée et une nouvelle est créée
FFDH	First-Fit Decreasing Height	Le plus à gauche possible dans la première couche où il rentre sinon, une nouvelle est créée
BFDH	Best-Fit Decreasing Height	Le plus à gauche possible dans la couche où l'espace restant horizontalement est le plus petit sinon, une nouvelle couche est créée
FNF	Finite Next-Fit	Dans la couche courante du *bin* courant , le plus à gauche possible, sinon, une nouvelle couche est initialisée dans le *bin* courant ou dans un nouveau *bin*
FFF	Finite First-Fit	Le plus à gauche possible dans la couche la plus basse du premier *bin* possible, sinon, une nouvelle couche est créée dans le dernier *bin* ou un nouveau est initialisé
HFF	Hybrid First-Fit	Exécution de FFDH puis résolution d'un *Bin-packing* à une dimension
FBS	Finite Best Strip	Exécution de BFDH puis résolution d'un *Bin-packing* à une dimension

les méthodes à une phase, les algorithmes *Finite Next-Fit* (FNF) et *Finite First-Fit* (FFF) [3]. Pour ce qui est de l'approche en deux phases, on rencontre les algorithmes *Hybrid First Fit* (HFF) [4], *Finite Best Strip* (FBS) [3]. Dans l'ensemble de ces méthodes (Tab. 1), les pièces sont posées sur le sol de la couche ainsi définie et placées suivant une stratégie de placement définie. Il reste alors un espace non utilisé au dessus des pièces les moins hautes. D'autres algorithmes ont été développés pour exploiter cet espace et ainsi tenter de réduire le nombre de plaques nécessaires. Citons ici les heuristiques *Floor-Ceiling (rotation and guillotine)* (FC_{RG}) [8] et *Shelf Heuristic Filling-First Fit* (SHF-FF) [1,2]. L'ensemble de ces méthodes fournit des configurations guillotines.

En 2004, Levine et Ducatelle [7] ont proposé une approche par colonies de fourmis, combinée avec une recherche locale simple, pour le problème de *Bin-packing* à une dimension. L'optimisation par colonies de fourmis est une meta-heuristique, utilisée entre autres, pour les problèmes d'optimisation combinatoire. Cette technique est inspirée du comportement des fourmis qui coopèrent dans la recherche de nourriture. Le premier algorithme d'optimisation par colonies de fourmis a été proposé par Dorigo *et al.* en 1996 [5] pour le problème du voyageur de commerce. Levine et Ducatelle sont les premiers à avoir utilisé les colonies de fourmis pour les problèmes de *Bin packing*.

Nous proposons dans cet article un algorithme hybride d'optimisation par colonies de fourmis pour le problème de *Bin packing* à deux dimensions, avec prise en compte de la contrainte guillotine (2BP/O/G). En effet, nous proposons de coupler l'optimisation par colonies de fourmis avec la technique de placement de l'heuristique SHF-FF [2]. Dans la Section 2, nous présentons le problème et les notations utilisées, la Section 3 est consacrée à la présentation du principe de l'optimisation par colonies de fourmis (ACO). Nous rappelons dans la Section 4 le principe de l'heuristique SHF-FF et présentons à la Section 5 l'approche de résolution ACO-SHF-FF pour le problème 2BP/O/G. La Section 6 est consacrée au réglage des paramètres et les résultats numériques sont quant à eux présentés Section 7, suivis de la conclusion.

2. Description du problème et notations

Soit un ensemble J de n objets rectangulaires $j \in J = 1, ...n$. Chaque objet j est défini par sa largeur w_j et sa hauteur h_j. On suppose que l'on dispose d'un ensemble de grands rectangles identiques de largeur W et hauteur H. On cherche donc à placer les pièces de J dans le plus petit nombre de rectangles possibles. La rotation de 90° des pièces n'étant pas permise, celles-ci doivent être placées de telle sorte que leur côté représentant la largeur (w_j) soit parallèle avec le côté de largeur W de la plaque. On suppose donc, sans perte de généralité, que $w_j \leq W$ et que $h_j \leq H \ \forall j \in J$.

3. L'optimisation par colonies de fourmis (ACO)

L'optimisation par colonies de fourmis est une méthode meta-heuristique inspirée par la capacité des fourmis réelles à trouver le plus court chemin entre leur nid et la source de nourriture. Le premier algorithme d'optimisation par colonies de fourmis (ACO) a été proposé par Dorigo [5] pour résoudre le problème du voyageur de commerce. Cette technique a depuis été utilisée pour résoudre une multitude de problèmes d'optimisation combinatoire.

La capacité des fourmis à déterminer le plus court chemin vient du fait qu'elles déposent des traces chimiques (phéromones) sur le sol. Plus un chemin est utilisé par des fourmis, plus il y a de phéromones de déposées et plus il devient attractif pour les fourmis suivantes. On définit donc $\tau(i, j)$, la quantité de phéromones

associée à la connexion entre 2 villes i et j, ainsi qu'une probabilité $p_k(i,j)$ qu'a une fourmi k de se déplacer d'une ville i à une ville j.

Chaque fourmi est placée aléatoirement dans une ville de départ et construit une solution en allant de ville en ville. Quand toutes les fourmis ont construit un tour, les phéromones sont mises à jour. Cette mise à jour comporte deux aspects. D'une part, les phéromones s'évaporent, et diminuent avec une vitesse d'affaiblissement ρ. D'autre part, elles sont plus importantes pour ce qui est du plus court chemin. Il faut donc augmenter les phéromones entre certaines villes. La technique utilisée pour faire cette augmentation dépend de la définition donnée aux phéromones en fonction du problème traité et prend en compte l'ordre des villes visitées dans la ou les meilleure(s) solution(s) obtenue(s) dans la population de fourmis.

Prenons par exemple le cas du *Bin packing* à 1 dimension traité par Levine et Ducatelle [7]. La traînée de phéromone $\tau(i,j)$ représente *l'avantage d'avoir un objet i et un objet j dans le même bin.* La probabilité pour qu'une fourmi k choisisse un objet j comme prochain objet pour le *bin* courant b, dans une solution partielle s est :

$$p_k(s,b,j) = \begin{cases} \frac{[\tau_b(j)][\eta(j)]^\beta}{\sum_{g \in J_k(s,b)} [\tau_b(g)][\eta(g)]^\beta} & j \in J_k(s,b) \\ 0 \quad \text{sinon} \end{cases} \tag{1}$$

avec

$$\tau_b(j) = \begin{cases} \frac{\sum_{i \in b} \tau(i,j)}{|b|} & \text{si } b \neq \{\} \\ 0 \quad \text{sinon} \end{cases} \tag{2}$$

où $J_k(s,b)$ est l'ensemble des objets candidats (ceux dont la taille permet de les y placer) pour aller dans le *bin* actuel, $\eta(j)$ est le poids de l'objet j donné par une heuristique qui guide les fourmis, et $\tau_b(j)$ est la valeur de la phéromone d'un objet j pour le *bin* b. β définit l'importance relative de l'heuristique. La mise à jour des traces de phéromones se fait de la manière suivante : seules les meilleures fourmis sont autorisées à placer des phéromones après chaque itération. Les phéromones sont augmentées à chaque fois que i et j sont combinés dans un *bin*. Dans l'hypothèse où il peut y avoir plusieurs objets identiques, ils définissent $t(i,j)$ le nombre de fois où i et j vont ensemble dans la meilleure solution s^{best}.

$$\tau_{i,j} = \rho\tau(i,j) + t(i,j) f(s^{\text{best}}). \tag{3}$$

La fonction f permet une évaluation de la qualité des solutions.

4. L'heuristique SHF-FF

L'heuristique SHF a été proposée en 2003 [1] pour le problème de *Strip-packing* avec contrainte guillotine. Les auteurs ont ensuite proposé une extension, nommée

SHF-FF, de cette approche pour le problème de *Bin-packing* avec contrainte guillotine [2]. Cette heuristique permet le placement des objets en couches, tout en cherchant à utiliser l'espace disponible au dessus des pièces les plus basses. Elle repose sur la notion de *rectangle disponible*. Un tel rectangle est un rectangle vide (ne contenant pas de pièce) dont le coin inférieur gauche correspond à un *point disponible*. Un tel point correspond soit au coin inférieur droit, soit au coin supérieur gauche, d'une pièce déjà placée. L'ensemble des points disponibles représente tous les emplacements possibles d'une nouvelle pièce. Un rectangle disponible R_i est caractérisé par les coordonnées (x_i, y_i) du point disponible qui le définit et par sa largeur w_i et sa hauteur h_i. Le placement d'une pièce j dans un rectangle disponible R_i se fait de telle sorte que le coin gauche bas de la pièce coïncide avec le coin gauche bas du rectangle R_i. Les pièces sont placées en fonction de l'ordre non croissant de la hauteur. L'heuristique SHF-FF se résume comme suit :

- Initialisation de la liste des rectangles disponibles au rectangle $R_1 = (0, 0, W, H)$.
- Initialisation du nombre de plaques $m = 1$.
- Pour chaque pièce j
  - Placer la pièce j dans le premier rectangle disponible qui peut la contenir (stratégie de placement *First-Fit*).
  - Si un tel rectangle n'existe pas, placer j dans une nouvelle plaque et $m = m + 1$.
  - Mettre à jour la liste des rectangles disponibles en retirant de celle-ci le rectangle disponible utilisé et en créant ceux générés par les points disponibles de la pièce j placée.
  - Mettre à jour la liste des rectangles disponibles en modifiant les dimensions des rectangles pour éviter d'une part tous chevauchements avec la pièce j et pour conserver d'autre part l'aspect guillotine de la solution.
  - Classer les rectangles disponibles par ordre croissant des ordonnées et pour des ordonnées égales, par ordre croissant des abscisses.

Un exemple d'application de cette méthode est donné à la Section 7.2 (Fig. 2).

5. ACO-SHF pour le 2BP/O/G

Nous proposons un algorithme hybride d'optimisation par colonies de fourmis utilisant le mode de placement de l'heuristique SHF-FF. En entrée de SHF-FF, les pièces sont toujours classées de la même manière : dans l'ordre décroissant de leur hauteur. Nous exploitons ici une colonie de fourmis afin de faire varier l'ordre dans lequel les pièces sont placées dans les plaques et trouver celui qui permet de minimiser le nombre de celles-ci. La colonie de fourmis permet alors d'optimiser l'ordre de placement des pièces dans l'heuristique SHF-FF.

5.1. Les phéromones

Les phéromones $\tau(i,j)$ déposées par les fourmis représentent *l'avantage de choisir de placer la pièce j après avoir placé la pièce i*. Afin de les mettre à jour à la fin de chaque itération, nous proposons de considérer les α meilleures solutions de l'itération actuelle. La mise à jour se fait alors avec la formule suivante :

$$\tau_{i,j} = \rho\tau(i,j) + \sum_{l=1}^{\alpha} u_{(i,j)}(l) f(s_l) \tag{4}$$

avec

$$u_{(i,j)}(l) = \begin{cases} 1 & \text{si } j \text{ choisie après } i \text{ dans } l \\ 0 & \text{sinon.} \end{cases} \tag{5}$$

Nous avons souhaité prendre en compte les α meilleures solutions afin de pouvoir faire varier ce paramètre et rendre ainsi l'algorithme plus agressif dans sa recherche. La fonction f est la fonction d'évaluation de la qualité des solutions.

5.2. Principe général de construction d'une solution

On considère une population de K fourmis. A chaque itération de l'algorithme, chaque fourmi k ($k \in K$) va commencer avec un ensemble de n objets à placer et un *bin* vide. Chacune va construire une solution s, en choisissant les pièces à placer une par une de manière aléatoire. Le choix des objets se fait en prenant en compte la probabilité $p_k(i,j)$ qu'à une fourmi k de choisir la pièce j sachant qu'elle vient de choisir la pièce i.

$$p_k(i,j) = \begin{cases} \dfrac{\tau(i,j)[\eta(j)]^{\beta}}{\sum_{g \in J_k} \tau(i,j)[\eta(j)]^{\beta}} & j \in J_k(i) \\ 0 \quad \text{sinon} \end{cases} \tag{6}$$

où $J_k(i)$ est l'ensemble des pièces non encore prises par k, après avoir choisi la pièce i, et $\eta(j)$ est la valeur donnée par l'heuristique qui guide les fourmis. Dans la majorité des heuristiques pour le problème 2BP/O/G, les pièces sont classées et numérotées en fonction de l'ordre décroissant (*Decreasing height*) de leur hauteur. Elles sont ensuite placées suivant cet ordre. En supposant ici que les pièces sont numérotées dans l'ordre croissant de leur hauteur (si $i < j$ alors $h_i < h_j$), plus son indice est important, plus elle aurait de chance d'être placée dans les premières suivant l'ordre *Decreasing height*. Nous avons choisi ici de prendre $\eta(j) = j$. Le paramètre β permet, quant à lui, de jouer sur l'importance du rôle joué par cette heuristique.

Une fois l'ordre de traitement des pièces ainsi établi, les *bins* sont remplis en appliquant l'algorithme SHF-FF [2].

5.3. Fonction f d'évaluation de la qualité des solutions

L'objectif étant la minimisation du nombre de plaques utilisées, on pourrait prendre comme fonction f d'évaluation l'inverse du nombre de *bins*, mais cela ne permet pas de différencier des solutions avec le même nombre de *bins*. Nous proposons alors de prendre en compte, comme Levine et Ducatelle [7] le nombre d'objets N_i dans une plaque. On utilise alors la fonction suivante pour évaluer une solution s :

$$f(s) = \frac{\sum_{i=1}^{N} (N_i/N_{\max})^{\gamma}}{N} \tag{7}$$

où N est le nombre de plaques, N_i le nombre d'objets dans la plaque i et $N_{\max}$ le nombre maximum d'objets par plaque dans l'ensemble des solutions de l'itération. γ est un paramètre qui définit l'importance du nominateur.

5.4. Exploitation de l'ordre de SHF-FF

Afin d'améliorer les performances de l'ACO, nous décidons d'introduire après 20 itérations, sans amélioration de la meilleure solution, la solution donnée par l'ordre décroissant des hauteurs (ordre utilisé dans l'heuristique SHF-FF).

5.5. Algorithme ACO-SHF

Nous résumons dans le pseudo-code suivant l'apporche ACO-SHF, avec K la taille de la population, $\emptyset$ le point de départ et $U[0,1]$ la loi uniforme sur [0,1].

- compteur = 0
- Faire
  - pour ($k = 1$ à K) faire
    - $J_k(\emptyset) = \{1, 2, 3, ...n\}$
    - $i = \emptyset$
    - pour ($t = 1$ à n) faire
      - R = U[0,1]
      - choisir de placer $j_2 \in J_k(i)$, selon SHF-FF, telle que

$$\sum_{l=j_{\min}}^{j_1} p_k(i,l) < R < \sum_{l=j_{\min}}^{j_2} p_k(i,l)$$

        avec $J_k(i) = \{j_{\min}, ..., j_1, j_2, ...\}$ tel que $j_{\min} < j_1 < j_2$
      - $J_k(j_2) = J_k(i) - \{j_2\}$
      - $i = j_2$
    - fin pour
  - fin pour
  - évaluer les solutions avec $f(s)$
  - Evaluation de la meilleure solution
  - si (meilleure solution améliorée) alors compteur = 0, sinon compteur = compteur +1

TABLEAU 2. Paramètres de ACO-SHF.

paramètre	K	α	ρ	γ	β
réglage pour $20 \leq n \leq 40$	20	3	0.7	3	2 à 5
réglage pour $50 \leq n \leq 80$	40	5	0.7	3	2 à 5

  - mise à jour des phéromones
  - mise à jour des probabilités
  - si (compteur = 20) et (meilleure solution moins bonne que SHF-FF), alors
    - meilleure solution = SHF-FF
    - compteur = 0
  - fin si
- tant que (compteur < 10)

6. Réglage des paramètres

Les différents paramètres de l'algorithme ACO-SHF sont résumés dans le tableau suivant. Pour chacun des paramètres, nous présentons les réglages par défaut à utiliser, en fonction du nombre de pièces à placer n. Nous expliquons ensuite la démarche qui nous a conduit à proposer ces réglages.

K est la taille de la population, α est le nombre de meilleures solutions à prendre en compte dans la mise à jour des phéromones, ρ est le paramètre d'affaiblissement des phéromones, le paramètre γ intervient dans le calcul de la note donnée à chacune des solutions et le paramètre β intervient quant à lui dans le calcul des probabilités.

Afin de proposer des réglages par défaut, nous avons effectué une campagne de tests suivant le protocole que nous décrivons maintenant. Nous avons utilisé 20 problèmes tests pour chaque taille (nous considérons que la taille est le nombre de pièces à placer) $n = 20, 30, 40, 50, 60, 70, 80$, pour lesquels nous avons également obtenu la solution optimale par énumération. Ainsi, nous avons testé l'influence de la variation de chaque paramètre sur le convergence, de la solution obtenue, vers l'optimum. Pour ce faire, nous n'en avons fait varier qu'un à la fois, en fixant les autres à la valeur la plus favorable connue. Une fois tous les paramètres traités, nous avons recommencé en réactualisant les valeurs des différents paramètres. Pour chaque taille de population, nous avons ainsi quantifié la rapidité de convergence vers l'optimum et retenu les réglages qui permettent de converger le plus rapidement possible.

Il ressort de cette campagne de tests qu'il vaut mieux générer des populations de $K = 20$ solutions pour des nombres de pièces de 20 et 40. L'augmentation de la taille de la population n'améliore pas la solution obtenue par la suite. Pour les problèmes de 50 à 80 pièces, la population doit être de $K = 40$ solutions, le nombre de pièces étant plus grand, le nombre de combinaisons aussi. Il est donc intéressant

de disposer d'une diversité suffisamment grande à chaque itération avant de mettre à jour les phéromones.

Pour le paramètre α, nous avons testé différentes valeurs allant de 1 à 7 pour l'ensemble des problèmes de tailles différentes. Il ressort de cette étude que l'augmentation de ce paramètre n'améliore pas de manière très significative la qualité des résultats. Nous avons finalement choisi de prendre $\alpha = 3$ pour une population de 20 solutions et $\alpha = 5$ pour une population de 40 solutions, en se basant sur les légères fluctuations observées au niveau de la convergence des solutions dans l'intervalle $[1, 7]$. Prendre des valeurs plus importantes pénalise la convergence des solutions et la qualité de celles-ci.

Pour le paramètre γ, nous avons testé les valeurs 1, 2, 3 et 4 et avons pu noter que les meilleurs résultats ont été obtenus pour $\gamma = 3$, quelque soit la taille du problème.

Pour ce qui est du paramètre ρ, les meilleurs résultats on été obtenus pour $\rho = 0.7$.

En ce qui concerne le paramètre β, nous avons testé divers valeurs entre 2 et 10 et il apparaît qu'il joue un rôle important dans la qualité de la solution, mais l'on ne peut pas déterminer pour un ensemble de problèmes ou une classe toute entière une valeur optimale. Pour chaque instance, nous avons testé les valeurs 2, 3, 4 et 5 et retenu la meilleure solution à chaque fois.

7. Résultats numériques

7.1. Protocole expérimental

L'algorithme ACO-SHF a été programmé en langage C et testé sur un pentium 4. Nous l'avons testé suivant les mêmes protocoles de test que Ben Messaoud *et al.* [2] en utilisant les classes de Berkey et Wang [3] très largement utilisées pour tester les algorithmes pour le 2BP/O/G. Celles-ci sont définies de telle sorte que les hauteurs h_j et les largeurs w_j des pièces sont générées aléatoirement suivant une loi uniforme dans divers intervalles (Tab. 1).

Pour chacune de ces classes, nous avons généré des problèmes avec différents nombres de pièces n. Nous avons choisi de prendre $n = 20, 40, 60$ ou 80. Pour chaque valeur de n, 10 instances ont été générées aléatoirement et comparées avec le résultat donné par l'heuristique SHF-FF.

7.2. Exemple d'application

Considérons le problème suivant issu de la classe I, pour $n = 20$. Les caractéristiques (w_j et h_j) des pièces $j = 1, ...20$ sont données dans le tableau suivant.

En appliquant l'heuristique SHF-FF, on obtient une solution avec 5 plaques. La répartition des pièces (Fig. 2) est donnée dans le Tableau 3, ainsi que les coordonnées du coin inférieur gauche de chacune dans la plaque où elle est placée.

L'application de l'ACO-SHF nous donne une solution avec 4 plaques (Fig. 1) dont la répartition des pièces est donnée dans le Tableau 4.

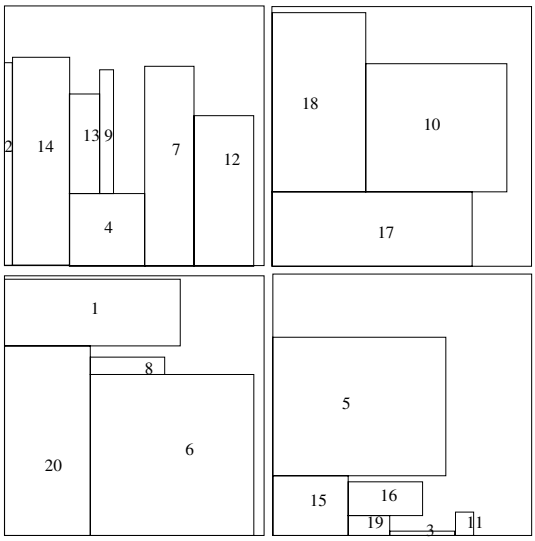

FIGURE 1. Placement avec l'ACO-SHF.

TABLEAU 3. Classes de Berkey et Wang.

Classe	Intervalle	W, H
I	$[1, 10]$	10
II	$[1, 10]$	30
III	$[1, 35]$	40
IV	$[1, 35]$	100
V	$[1, 100]$	100

TABLEAU 4. Caractéristiques des pièces.

j	w_j	h_j	j	w_j	h_j
1	6.91	2.58	11	0.77	0.86
2	0.36	7.82	12	2.28	5.84
3	2.49	0.02	13	1.24	3.97
4	2.87	2.80	14	2.13	8.04
5	6.78	5.30	15	3.15	2.36
6	6.44	6.12	16	2.94	1.34
7	1.89	7.75	17	7.75	2.93
8	2.87	0.64	18	3.86	6.84
9	0.54	4.89	19	1.69	0.97
10	5.47	4.95	20	3.33	7.35

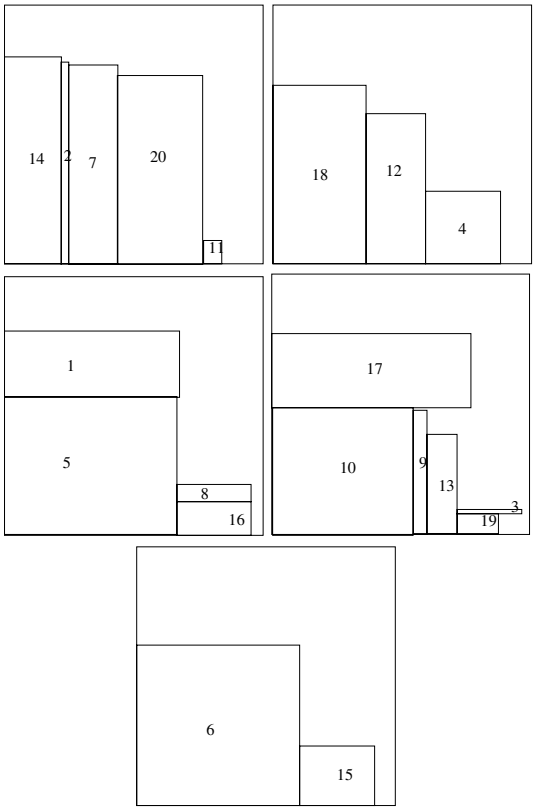

FIGURE 2. Placement avec l'heuristique SHF-FF.

TABLEAU 5. Répartition des pièces par SHF-FF.

Plaque	pièces
1	14(0-0), 2(2.13-0), 7(2.49-0), 20(4.39-0), 11(7.71-0)
2	18(0-0), 12(3.86-0), 4(6.14-0)
3	6(0-0), 15(6.44-0)
4	5(0-0), 1(0-5.3), 16(6.78-0), 8(6.78-1.34)
5	10(0-0), 9(5.47-0), 13(6.01-0), 17(0-4.95), 19(7.25-0), 3(7.25-0.97)

Cet exemple illustre le fait que le traitement des pièces dans l'ordre décroissant des hauteurs n'est pas forcément le meilleur pour toutes les instances, avec le mode de placement de l'heuristique SHF-FF.

TABLEAU 6. Répartition des pièces par ACO-SHF.

Plaque	pièces
1	2(0-0),14(0.36-0), 4(2.49-0), 7(5.36-0), 12(7.25-0), 13(2.49-2.8), 9(3.73-2.8)
2	17(0-0), 18(0-2.93), 10(3.86-2.93)
3	20(0-0), 6(3.33-0), 1(0-7.35), 8(3.33-6,12)
4	15(0-0), 19(3.15-0), 5(0-2.36), 3(4.84-0) 16(3.15-0.9), 11(7.33-0)

TABLEAU 7. Résultats pour la Classe I.

n	Pam	PMD	TM
20	40	23.75	35.39
40	30	8.84	52.72
60	30	4.76	163.46
80	10	4.76	226.46

TABLEAU 8. Résultats pour la Classe II.

n	Pam	PMD	TM
20	0*	-	0.40
40	30	50	14.60
60	0*	-	90.81
80	0*	-	316.50

TABLEAU 9. Résultats pour la Classe III.

n	Pam	PMD	TM
20	20	16.6	30.00
40	30	9.14	114,43
60	10	5,5	132,82
80			

7.3. EXPÉRIMENTATIONS NUMÉRIQUES

Nous présentons maintenant les résultats obtenus en appliquant le protocole expérimental défini précédemment. Pour chaque classe de problèmes, nous présentons dans un tableau le pourcentage Pam de problèmes dont la solution est améliorée. Le pourcentage de problèmes dont le nombre de *bins* est identique est alors $Pid = 100 - Pam$, du fait de l'introduction de l'ordre de traitement des pièces de SHF-FF. Sont également indiqués les pourcentages PMD moyens de diminution du nombre *de bins* et les temps moyens TM d'exécution en seconde CPU pour l'ensemble des 10 problèmes pour chaque valeur de n.

Tableau 10. Résultats pour la Classe IV.

n	Pam	PMD	TM
20	0	–	0.47
40	10	50	31.71
60	0	–	189.98
80	0	–	320.70

Tableau 11. Résultats pour la Classe V.

n	Pam	PMD	TM
20	60	19.72	30.09
40	40	8.19	42.84
60	20	5.26	173.59
80	0	–	330.70

La marque * indique que pour l'ensemble des problèmes, le nombre de plaques n'a pas pu être diminué, celui-ci étant de 1 pour $n = 20$, de 2 pour $n = 60$ et 3 pour $n = 80$.

Nous observons tout d'abord que les temps d'exécution augmentent avec la taille des problèmes, tout en restant raisonnables. Cette augmentation est due à la taille n elle même mais aussi au fait que la taille de la population de solutions est plus importante pour des problèmes de grande taille.

L'utilisation des colonies de fourmis nous a permis de montrer qu'il peut exister un ordre de traitement des pièces meilleur que celui des hauteurs décroissantes pour l'application du mode de placement de l'heuristique SHF-FF.

Pour les problèmes de la classe I, nous avons diminué le nombre de plaques de 23,75 % en moyenne pour 40 % des solutions. Ce pourcentage de solutions améliorées diminue avec la taille des problèmes. Notons toutefois que l'ordre obtenu d'une part par ACO-SHF et d'autre part par SHF-FF pour un nombre de plaques égale n'est pas toujours le même du fait de l'introduction de la fonction f dans l'évaluation des solutions.

Pour les problèmes de la classe II et de la classe IV, peu de solutions ont pu être améliorées du fait du petit nombre de plaques nécessaires obtenu par SHF-FF.

Pour les problèmes des classes III et V, nous avons également amélioré des solutions. Dans une classe, plus la taille des problèmes augmente, moins il semble simple d'améliorer la solution donnée par SHF-FF. Ceci semble dû en partie au jeu des différents paramètres qui interviennent dans l'algorithme d'optimisation par colonies de fourmis.

Afin d'améliorer les performances de notre méthode, nous envisageons d'établir d'autres fonctions d'évaluation des solutions et d'introduire une recherche locale sur l'ensembles des α meilleures solutions retenues pour chaque itération.

8. Conclusion

Nous avons traité dans cet article le problème de *Bin-packing* à deux dimensions prenant en compte la contrainte guillotine (2BP/O/G). Nous avons développé l'algorithme ACO-SHF combinant l'optimisation par colonies de fourmis et la technique de placement de l'heuristique SHF. Nous avons amélioré les résultats donnés par l'heuristique SHF sur de nombreux problèmes. Cette approche peut également être étendue à la prise en compte de la rotation des pièces (2BP/R/G) ainsi qu'aux problèmes de *Strip-packing* à deux dimensions équivalents.

Afin d'améliorer cet algorithme, nous envisageons de tester une approche en deux phases. En effet, on peut résoudre de manière générale le problème de *strip-packing* directement avec les colonies de fourmis et construire la solution pour le problème de *Bin-packing* ensuite. Il est également envisageable de générer parallèlement des solutions qui ne respectent pas la contrainte guillotine avec une seconde colonie de fourmis, et d'utiliser celle-ci pour rendre plus agressive la recherche dans le cas avec contrainte guillotine.

Remerciements. Nous tenons à remercier les éditeurs en chef de la revue et les rapporteurs pour leurs conseils.

Références

[1] S. Ben messaoud, C. Chu and M.L. Espinouse, Une nouvelle heuristique pour le problème de découpe guillotine en 2D, in *Proc.MOSIM'03*, Toulouse, France (2003) 116–121.
[2] S. Ben messaoud, C. Chu and M.L. Espinouse, New concept of the classic shelf algorithm. *Proc. IEPM'03*, Porto, Portugal (2003) 465–471.
[3] J.O. Berkey and P.Y. Wang, Two dimensional finite bin-packing algorithms. *J. Oper. Res. Soc.* **38** (1987) 423–429.
[4] F. Chung, M. Garey and D. Johnson, On packing two-dimensional bins. *SIAM J. Algebr. Discrete Methods* **3** (1982) 66–76.
[5] M. Dorigo, V. Maniezzo and A. Colorni, The ant system : Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. B* **26** (1996) 29–71.
[6] H. Dyckhoff, A typology of cutting and packing problems. *Eur. J. Oper. Res.* **44** (1990) 145–159.
[7] J. Levine and F. Ducatelle, Ant Colony optimization and local search for bin packing and cutting stock problems. *J. Oper. Res. Soc.* **55** (2004) 705–716.
[8] A. Lodi, S. Martello and D. Vigo, Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem, in *Meta-heuristics : Advances and Trends in Local Search Paradigms for Optimization*, S. Voss, S. Martello, I.H. Osman, C. Roucairol, Kluwer academic Publishers, Boston (1998) 125–139.
[9] A. Lodi, S. Martello and D. Vigo, Recent advances on two-dimensional bin packing problems. *Discrete Appl. Math.* **123** (2002) 379–396.
[10] A. Lodi, S. Martello and D. Vigo, Two-dimensional packing problems : A survey. *Eur. J. Oper. Res.* **141** (2002) 241–252.