Approche hybride pour résoudre le problème du Bin Packing

BACHI Yasmine, MIHOUBI LamiaZohra , MOUSSAOUI Meroua, NOUALI Sarah, SAADI Khaoula

Ecole nationale Supérieure d'Informatique -ESI-Alger

Abstract

L'objectif du problème du bin packing (BPP) est de trouver le nombre minimal de boîtes nécessaire pour ranger un ensemble de n objets ayant des tailles connues, en respectant la capacité de chaque boîte. Ce problème est parmis les problèmes NP-difficile. Dans cet article, on propose un algorithme génétique hybride utilisant le recuit simulé. Les résultats expérimentaux ont montré l'efficacité de notre hybridation dans l'amélioration de la qualité de solution de l'algorithme génétique pour les classes 1 et 2 du benchmark Scholl.

Keywords: Bin packing, hybridation, AG, recuit simulé, WOA, ILWOA

1. Introduction

Le problème du bin packing à une dimension (BPP) est défini comme suit, étant donné un nombre illimité de boîtes avec une capacité fixe C, et un ensemble de n objets, chacun ayant un poid spécifique $0 < wi \leqslant C$, on cherche à ranger les n objets dans un nombre minimal de boîte, tout en respectant la capacité C.

1.1. Algorithme génétique (AG)

L'algorithme génétique est une métaheuristique de recherche initialement proposée par Holland [7] qui imite le processus de sélection naturelle. Elle appartient à la plus grande classe d'algorithmes évolutionnaires (EA), qui génèrent des solutions en utilisant des techniques inspirées de l'évolution naturelle, telles que la mutation, la sélection et le croisement.

La figure suivante représente les étapes de fonctionnement de notre AG, avec la spécification des méthodes utilisées pour implémenter chaque étape.

Les algorithmes génétiques sont connus par leur rapidité et facilité d'utilisation. En effet, le AG est parmis les métaheuristiques les plus rapides, de plus, si la représentation vectorielle de l'individu est correcte, nous pouvons trouver une solution sans un travail d'analyse approfondi. Par contre, cette métaheuristique ne trouve pas toujours l'optimum et peut se retrouver avec le problème d'optimum local.

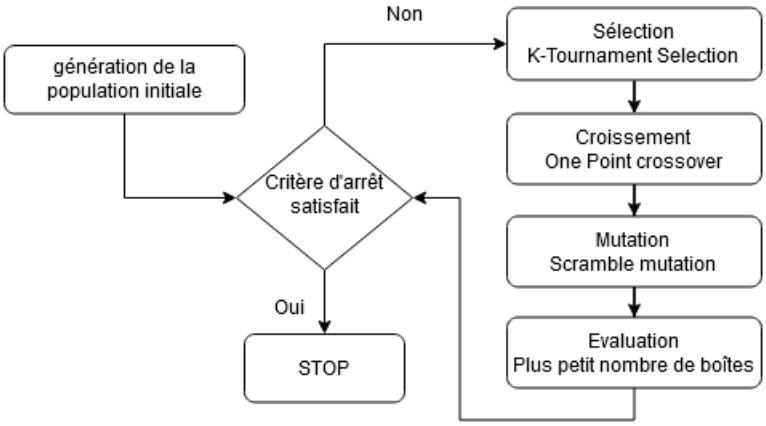


Figure 1: Processus de AG

1.2. Le Recuit Simulé (SA)

Le recuit simulé est un algorithme de recherche locale initialement introduit par Kirkpatrick et al [12] qui simule la fusion et le refroidissement dans le traitement des métaux. Le recuit simulé est généralement implémenté pour rechercher une solution optimale sur une petite zone, même s'il est parfois aussi performant que AG dans certains cas [3]. Cette métaheuristique sophistiquée empêche d'être piégé dans les minima locaux à l'aide d'un moteur de recherche aléatoire exprimé en termes de chaîne de Markov. Elle introduit des changements dans la solution pour améliorer la fonction objectif, mais conserve également des solutions qui, malgré les moins bonnes performances, répondent à certains critères. Mais l'un des inconvénients du recuit simulé est son temps d'exécution élevé.

1.3. Hybridation et travaux relatifs

Pour surmonter les inconvénients de GA et SA, plusieurs études proposent une hybridation entre les deux. Junghans et Darde [8] comparent entre AG et AG hybride avec SA modifiée (MSA). Le SA utilisé dans leur expérience a été modifié pour contrôler la réduction de la température. Ils ont découvert que le GA-MSA hybride offre une fiabilité plus élevée que le AG. Une autre recherche menée par Chen et Shahandashti dans [5] qui compare également le GA, SA, un hybride de GA-SA et MSA, où ils ont constaté que le GA-SA hybride est plus performante que AG, SA et MSA.
Dans cet article, nous avons hybridé le GA-SA avec quatres scénarios, à savoir le AG-RS hybride, le AGH-RS, le AG-2RS et le AG-MIX, Le schéma hybride AG-RS consiste à obtenir la meilleure solution en AG et à l'utiliser comme population initiale en SA, le AGH-RS utilise le même processus sauf que AG à été initialisé par plusieurs heuristiques.Le schema AG-2RS consiste à inclure SA dans le AG après l'étape de mutation, et améliorer la solution de AG par le SA à nouveau. finalement le schéma AG-MIX utilise le même processus que

AG-2RS en initialisant le AG par plusieurs heuristiques. D'autres schémas ont été implémenté et serons inclus dans la comparaison, on cite le WOA-RS et le ILWOA-RS qui sont une hybridation de haut niveau des métaheuristiques WOA[9] et ILWOA[2] avec le recuit simulé.

1.4. Organisation du papier

On va tout d'abord donner la formulation mathématique du problème, par la suite on présentera nos schémas hybrides proposés. Finalement, des résultats expérimentaux sont donnés et une comparaison des schémas hybrides est effectuée dans (4) , suivi d'une conclusion.

2. Formulation du problème

2.1. Formulation mathématique

Etant donné m boites de capacité C et n articles de volume v_i chacun. Soient:

$$x_{ij} = \begin{cases} 1 & \text{article } j \text{ rangé dans la boîte } i \\ 0 & \text{sinon} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{boîte } i \text{ utilisée} \\ 0 & \text{sinon} \end{cases}$$

La formulation du problème donne ainsi le programme linéaire suivant

$$(PN) \begin{cases} Z(min) = \sum_{i=1}^{m} y_i \\ \sum_{i=1}^{m} x_{ij} = 1 \\ \sum_{j=1}^{n} v_j x_{ij} \leq C y_i \\ y_i \in \{0,1\} \\ x_{ij} \in \{0,1\} \end{cases}$$

La première contrainte signifie qu'un article j ne peut être placé qu'en une seule boite La deuxième fait qu'on ne dépasse pas la taille d'une boite lors du rangement

3. L'approche hybride proposée

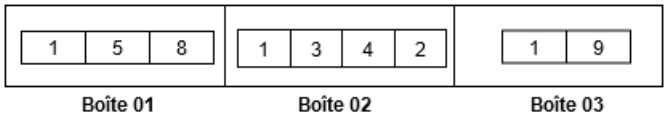
Pour le schéma de l'hybridation proposé dans cet article, il s'agit d'une hybridation classifiée selon la taxonomie de talbi[13] comme suit **HRH globale, généraliste** , et qui utilise l'algorithme génétique comme méthode révolutionnaire, suivie par un Recuit Simulé pour l'intensification des résultats de ce dernier.

3.1. Encodage de la solution

3.1.1. Solution du bin packing

La solution sera représentée par:

- une liste de boîtes, où la boîte est représentée par une liste des objets qui y sont rangés, et l'objet est représenté par un entier désignant son poids.



- Le nombre de boîtes utilisées, ce qui représente le coût de la solution, et nous permettra d'évaluer la performance de l'algorithme en terme de la qualité de la solution.

3.1.2. Représentation chromosomique

Pour cette implémentation de l'AG, nous allons utiliser une nouvelle représentation chromosomique, proposée dans [10], On représente une solution du Bin packing comme suit:

- On suppose qu'on a n objets à ranger donc on utilisera n boîtes au maximum.
- Chaque boîte est composée de n cellules, où chaque cellule ne peut contenir qu'un seul objet.
- Chaque cellule a un numéro unique dans la solution.
- Si la cellule de l'ordre i de la boite j est remplie par un objet, alors on aura plus le droit de ranger un objet dans toutes les cellules de l'ordre i des autres boites.
- La cellule *0* contient le nombre de boîtes utilisées dans cette solution.

Dans le chromosome , on ne garde trace que des numéros de cellules où les objets sont stockés, donc la taille du chromosome sera de *n+1*.

3.2. L'hybridation AG-RS

Une hybridation entre AG et RS permet à AG d'explorer un énorme espace de recherche et à RS d'exploiter des zones de recherche locales. Le RS commence généralement par une solution initiale aléatoire, dans la solution proposée, la meilleure solution de AG est utilisée comme configuration initiale, et le Recuit simulé va se charger de l'intensification de cette dernière, où il utilise un processus de recherche d'une solution optimale globale dans l'espace de la solution, grâce à son aspect aléatoire, et qui a été prouvé de guider l'algorithme vers l'optimum global. Donc, le AG est puissant pour obtenir une solution presque optimale sur la zone de recherche large tandis que RS est utile pour rechercher une solution dans la zone de recherche étroite.[6]

4. Tests et résultats

Les différents algorithmes ont été développés en utilisant python, et exécutés sur un DELL Inspiron15 [Intel® Core™ i7-8550U CPU @ 1.80GHz×8, 8Go] en utilisant Visual Studio Code.
les tests ont été effectué sur le benchmark Scholl [1], en prenant des instances de petites, moyenne, et grandes taille, ont utilisera par la suite l'encodage suivant

Code	**N1**	**N2**	**N3**	**N4**	**HARD**
Valeur	50	120	200	500	HARD

Table 1: encodage des instances

4.1. Calibrage des paramètres

Les paramètres des métaheuristiques utilisées dans cette article (AG, WOA, ILWOA) ont été fixé en utilisant un outil de calibrage automatique Irace , son implémentation est disponible dans un package R au près de CRAN dans [11], ce package implémente une extension de la course F itérée (I / F-Race) expliquée dans [4]. Les configurations optimales des paramètres sont données dans le tableau suivant :

Méthode	**Paramètre**	**Valeur**
AG	le nombre de populations générées (nbrGen)	200
	paramètre de "k-tournament selection" (k)	20
	taille d'une population (popSize)	10
	probabilité de croisement	0.85
	probabilité de mutation	0.1
RS	le nombre d'itérations (R)	1000
	Le facteur de diminution de la température (α)	0.925
WOA	taille de la population (nb-whales)	30
	nombre d'itérations (max-iter)	117
	constante de l'encerclement spirale (b)	8.96
	constante d'exploration (a)	10
ILWOA	taille de la population (nb-whales)	10
	nombre d'itérations (max-iter)	30(hard),10
	constante de l'encerclement spirale (b)	1.5
	constante d'exploration (a)	2
	beta	1.5

4.2. Etude des performances

Dans le tableau suivant on donne le temps d'exécution moyen de AG-RS en fonction de la taille de l'instance ainsi que l'écart par rapport à solution optimale en séparant la classe difficile (HARD) dans ce cas.

Le temps d'exécution de AG-RS augmente avec la taille de l'instance, on voit un plus grand écart au niveau de N4 =500, où le temps moyen de résolution

N/Instances	**Temps d'execution (s)**	**Ecart(%)**
N1=50	1.216188	0.606618
N2=120	3.213707	0.446429
N3=200	12.083458	0.071839
N4=500	47.665262	1.320662
HARD		6.059809

Table 2: temps d'execution et écart de AG-RS

d'une instance est de 48 secondes, qui reste toujours un temps acceptable. Le AG-RS converge vers la solution assez rapidement.
Pour toutes les instances du benchmark, AG-RS arrive à donner des solutions très proches des solutions optimales avec un écart qui ne dépasse pas 2%, sauf le cas des instances HARD (classe 3) où il atteint une valeur de 6%.

4.3. Etude comparative

Dans cette partie nous allons comparer AG-RS avec d'autres algorithmes de résolution implémentés et présentés dans *(1.3)*, dont les trois schémas à base de AG (AGH-RS,AG-MIX-RS, AG-RS2), ainsi que deux autres schémas HRH-WOA-RS, HRH-ILWOA-RS, à base des métaheuristiques WOA, ILWOA respectivement. La comparaison va inclure les métaheuristiques utilisées ainsi que l'heuristique FF qui est la plus performante parmis les heuristiques du bin packing.
Les figure 1 et 2 représentent le temps d'exécution et l'écart à la solution optimale en % (resp) pour chaque méthode implémentée, en fonction de la taille de l'instance et sa difficulté.

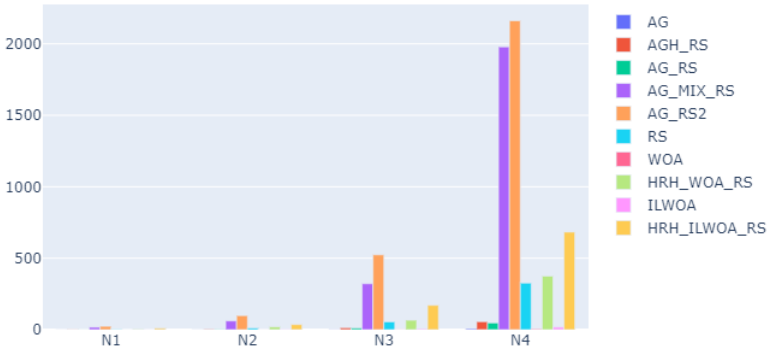


Figure 2: Temps d'execution moyen en fonction des tailles d'instances

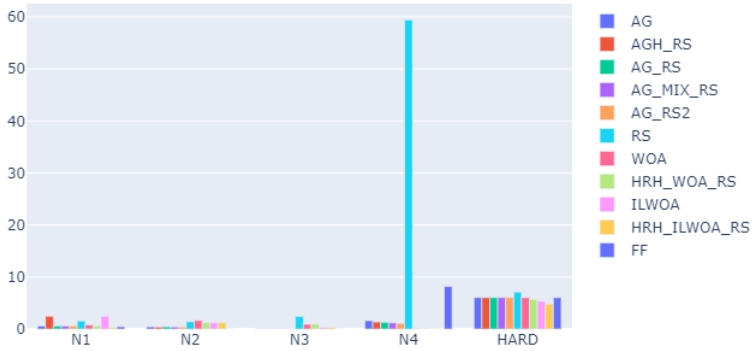


Figure 3: Ecart en % en fonction des tailles d'instances

A partir des deux graphes, on voit que l'algorithme AG-RS permet d'avoir une bonne qualité de solution en un temps d'exécution assez rapide. Il est à noter que les schémas à base de WOA et son amélioration ILWOA permettent d'avoir une légère amélioration de la qualité de solution pour les instances difficiles (HARD), cependant la lenteur de leur exécution les rend moins intéressants. Les schémas où les heuristiques sont utilisées dans l'initialisation de AG (AGH-RS, AG-MIX-RS) ont une qualité dégradée par rapport à leurs versions sans heuristiques (AG-RS , AG-2RS resp), on justifie ce résultat par leur coincement dans un optimum local, en effet une meilleure amélioration est d'utiliser les heuristiques dans l'initialisation d'une petite partie de la population et garder l'aspect aléatoire dans le reste pour éviter ce problème. Il est clair que le recuit simulé consomme un temps d'exécution très grand, mais offre une qualité de solution très bonne, sauf dans le cas N4 où un écart de 60% est trouvé, ceci est dû à la limite de temps imposée dans l'implémentation de l'algorithme, qui n'as pas permi à RS d'explorer correctement le voisinage. Les schémas utilisant le RS dans chaque itération de AG (AG-2RS, AG-MIX-RS) ont un temps d'exécution inacceptable avec l'une des meilleures qualités de solutions (AG-2RS) , une meilleure version est d'utiliser une recherche locale moins sophistiqué que RS et l'appliquer chaque lot d'itération. Finalement, l'heuristique FF donne une qualité assez bonne dans les instances de petites taille (N1,N2) avec un temps d'exécution négligeable, l'apport des MH et l'hybridation est plus visible et intéressant à partir des instances moyenne et grande taille.

5. Conclusion

Dans cet article, on s'est intéressé au problème du bin packing, où on a proposé un schéma hybride AG-RS. Pour montrer les performances de cet algorithme, on l'a comparé avec d'autres schémas proposées à base des métaheuristiques AG, WOA, et ILWOA. Les tests ont été effectué sur le benchmark scholl et les résultats ont été donné en temps d'exécution moyen et écart à la solution optimale. Notre schéma AG-RS est celui qui assure un bon compromis qualité coût.

Pour pousser l'étude, on propose l'utilisation d'une heuristique de recherche locale dans quelques itérations de AG et ILWOA.

References

[1] *Scholl Benchmark*. `https://www2.wiwi.uni-jena.de/Entscheidung/binpp/index.html`.

[2] M. Abdel-Basset, G. Manogaran, L. Abdel-Fatah, and S. Mirjalili. An improved nature inspired meta-heuristic algorithm for 1-d bin packing problems. *Personal and Ubiquitous Computing*, 22(5-6):1117–1132, 2018.

[3] T. W. Al-Khateeb B. Solving 8-queens problem by using genetic algorithms, simulated annealing, and randomization method. *International Conference on Developments in eSystems Engineering (DeSE)*, page 187–191, 2013. doi: 10.1109/DeSE.2013.41.

[4] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle. *Experimental Methods for the Analysis of Optimization 311 Algorithms*, chapter 13, pages 312–333. Springer-Verlag Berlin Heidelberg, 2010. doi: 10.1007/978-3-642-02538-9_13,.

[5] S. S. Chen PH. Hybrid of genetic algorithm and simulated annealing for multiple project scheduling with multiple resource constraints. page 18:434–443, 2009. doi: 10.1016/j.autcon.2008.10.007.

[6] Gusti, Andreas, Tirana, M.Shochibul, and F.Wayan. Hybrid genetic algorithm and simulated annealing for function optimization. 2017. doi: 10.25126/jitecs.20161215.

[7] H. J. Adaptation in natural and artificial systems. (English). *MIT press, Cambridge*, (1975,1992).

[8] D. N. Junghans L. Hybrid single objective genetic algorithm coupled with the simulated annealing optimization method for building optimization. page 187–191, 2015. doi: 10.1016/j.enbuild.2014.10.039.

[9] S. Mirjalili and A. Lewis. The whale optimization algorithm. *Advances in engineering software*, 95:51–67, 2016.

[10] N. Mohamadi. Application of genetic algorithm for the bin packingproblem with a new representation scheme. 2010.

[11] C. R. package. Irace. `https://cran.r-project.org/package=irace`.

[12] J. M. S.Kirkpatrick, C.D.Gelatt. Optimization by simulated annealing. 1983.

[13] E.-G. Talbi. A taxonomy of hybrid metaheuristics. *Journal of heuristics*, 8(5):541–564, 2002.