Hybridation des Métaheuristiques pour le Problème du Bin Packing

ABOUD Rayane, *AMEUR Samia,* BELLI Bilal, *BENABDELATIF Maya,*
BENDJENAHI Dihia, *SAFTA Abir*

***Abstract*—Dans cet article, nous avons adopté une structure bien organisée pour présenter notre solution. Nous commençons par la section de "Formulation du problème", où nous définissons clairement le problème étudié et proposons une formulation mathématique rigoureuse pour le décrire. Ensuite, nous décrivons en détail "Notre approche proposée", en fournissant une introduction qui présente l'architecture globale de notre approche hybride, ainsi que l'encodage de la solution utilisée et une explication détaillée de chaque étape spécifiquement adaptée au problème traité. La section suivante, "Tests et résultats", introduit le scénario des tests, y compris l'environnement expérimental et les caractéristiques de notre machine utilisée pour le benchmarking. Nous exposons ensuite les instances utilisées, le calibrage des paramètres, et réalisons une analyse approfondie des performances de notre approche en évaluant son comportement, son évolution en fonction des valeurs des paramètres et des différentes instances en entrée, en particulier avec et sans les composants d'hybridation ajoutés. Nous effectuons également une étude comparative avec d'autres méthodes, en termes de temps d'exécution et de qualité des solutions obtenues, en tenant compte de la littérature pertinente. Enfin, nous incluons une section de "Références bibliographiques" pour permettre aux lecteurs de consulter les sources que nous avons utilisées.**

I. Introduction

Le problème d'optimisation du bin packing est un défi classique en optimisation combinatoire qui vise à emballer de manière efficace un ensemble d'objets dans un nombre minimal de conteneurs, également appelés bacs. L'objectif est d'attribuer les objets aux bacs de manière à minimiser l'espace gaspillé et à maximiser l'utilisation de la capacité disponible. Chaque objet possède une taille ou un volume, et chaque bac a une capacité limitée. L'objectif est de trouver un arrangement qui remplit de manière optimale les bacs, en minimisant le nombre de bacs requis tout en garantissant que la taille totale des objets dans chaque bac ne dépasse pas sa capacité. Le problème du bin packing présente de nombreuses applications concrètes, telles que la logistique et le transport, l'allocation des ressources et la planification, où une utilisation efficace de l'espace est cruciale pour réduire les coûts et améliorer l'efficacité opérationnelle. Il convient de noter que le problème du bin packing est classé comme NP-complet, ce qui signifie qu'il est difficile à résoudre de manière exacte dans un temps raisonnable pour de grandes instances, et qu'il nécessite souvent l'utilisation d'approches approximatives ou heuristiques pour obtenir des solutions satisfaisantes.[1]

De nombreuses techniques ont été développées pour résoudre ce défi d'optimisation, allant des algorithmes exacts aux heuristiques et aux métaheuristiques. Les méthodes exactes, telles que la programmation linéaire en nombres entiers, permettent de trouver des solutions optimales, mais elles peuvent être limitées en termes de scalabilité pour de grandes instances en raison de la complexité exponentielle du problème. Les heuristiques, d'autre part, sont des approches basées sur des règles heuristiques qui cherchent à trouver des solutions de bonne qualité dans des délais raisonnables. Parmi les heuristiques couramment utilisées figurent la méthode du premier ajustement (first-fit), la méthode du meilleur ajustement (best-fit) et la méthode du prochain ajustement (next-fit). Les métaheuristiques, telles que la recherche tabou, les algorithmes génétiques et les essaims de particules, offrent des approches plus flexibles et exploratoires, permettant d'obtenir des solutions de qualité satisfaisante dans des contextes plus complexes. La revue de la littérature a souligné l'importance de combiner différentes techniques et d'adapter les approches en fonction des caractéristiques spécifiques du problème, du contexte et des contraintes. Cela permet d'obtenir des solutions efficaces et adaptées à des instances réelles du problème du bin packing. [2]

Les métaheuristiques sont des approches puissantes utilisées pour résoudre des problèmes d'optimisation NP-difficiles tels que le bin packing. Elles offrent une solution flexible et efficace en explorant l'espace des solutions de manière heuristique. Elles exploitent des mécanismes d'exploration et d'exploitation de l'espace des solutions pour converger vers des solutions prometteuses. Bien qu'elles puissent trouver des solutions satisfaisantes dans un temps raisonnable, elles ne garantissent pas l'optimalité en raison de leurs stratégies de recherche approximatives. Les métaheuristiques les plus couramment utilisées incluent la recherche tabou, les algorithmes génétiques, les essaims de particules, les colonies de fourmis, etc. Les métaheuristiques peuvent être sensibles aux paramètres et à la représentation de l'espace des solutions, et elles peuvent rencontrer des problèmes de stagnation. Pour surmonter ces limites, des recherches se concentrent sur le développement de métaheuristiques améliorées et l'hybridation avec différentes techniques afin de combiner leurs avantages respectifs. De plus, des approches parallèles et distribuées sont étudiées pour accélérer la recherche de solutions.

L'hybridation de la méthode du Water Wave Optimizer (WWO) avec le recuit simulé a été choisie afin de trouver un équilibre entre l'intensification et l'exploration dans la résolution du problème du bin packing. Cette décision repose sur les avantages complémentaires offerts par ces deux approches. Le WWO, basé sur les mouvements ondulatoires, est efficace pour explorer l'espace des solutions et découvrir de

nouvelles régions prometteuses. Cependant, il peut avoir des difficultés dans l'intensification rapide vers les optima locaux. En revanche, le recuit simulé, inspiré du refroidissement des métaux, permet des mouvements plus exploratoires et évite les pièges des optima locaux. L'hybridation de ces deux approches vise à exploiter leurs avantages respectifs, favorisant une meilleure exploration des solutions et une convergence plus rapide vers des solutions optimales. Cette approche hybride présente ainsi un fort potentiel pour résoudre efficacement le problème du bin packing en équilibrant l'intensification et l'exploration.

La littérature scientifique a examiné l'application d'approches hybrides de métaheuristiques dans divers problèmes d'optimisation, tels que le voyageur de commerce, l'ordonnancement de tâches, la planification de production, et l'optimisation des systèmes énergétiques. Par exemple, l'hybridation de l'algorithme génétique avec le recuit simulé a permis une meilleure exploration de l'espace des solutions et une convergence plus rapide vers des solutions de haute qualité pour le problème du TSP. Les approches hybrides offrent souvent la possibilité de surmonter les limitations inhérentes à chaque métaheuristique en combinant leurs forces complémentaires. Cela peut inclure l'ajout de stratégies de recherche locale à un algorithme génétique ou l'intégration d'une heuristique de construction dans une recherche tabou. Les méthodes hybrides varient en termes de combinaisons spécifiques et de techniques utilisées, allant des approches adaptatives dynamiques à l'utilisation de métaheuristiques parallèles et en cascade. En résumé, l'hybridation des métaheuristiques offre un potentiel prometteur pour améliorer la résolution de problèmes d'optimisation en exploitant les avantages de différentes techniques et en surmontant leurs limitations individuelles. Des recherches supplémentaires sont nécessaires pour développer et explorer de nouvelles approches hybrides adaptées à diverses problématiques d'optimisation.[3] [4]

L'approche proposée dans notre étude se démarque par son originalité en combinant le Water Wave Optimizer (WWO) avec le recuit simulé pour résoudre le problème d'optimisation considéré. Cette hybridation vise à exploiter les avantages complémentaires de ces deux métaheuristiques et à trouver un équilibre entre l'intensification et l'exploration lors de la recherche de solutions optimales. La singularité de notre approche réside dans l'utilisation du WWO, qui s'inspire des processus naturels des ondes d'eau, pour explorer efficacement l'espace des solutions. Les mouvements ondulatoires du WWO favorisent une exploration approfondie de l'espace de recherche. En combinant le WWO avec le recuit simulé, nous apportons une originalité supplémentaire à notre approche. Le recuit simulé, basé sur le refroidissement métallurgique, intensifie la recherche grâce à des mouvements plus exploratoires. Cette combinaison équilibre l'intensification en se concentrant sur les régions prometteuses et l'exploration en évitant les minimums locaux et en découvrant de nouvelles régions de recherche. L'originalité de notre approche hybride réside dans la combinaison spécifique du WWO et du recuit simulé, offrant une approche novatrice pour résoudre le problème considéré. Nous sommes convaincus que cette combinaison unique améliorera les performances de résolution en exploitant les avantages complémentaires de ces deux métaheuristiques pour trouver des solutions de haute qualité.

II. Formulation du Problème

Le problème du bin packing, également connu sous le nom de problème d'emballage dans des bacs, est un problème d'optimisation combinatoire. Il consiste à emballer un ensemble d'objets de tailles variables dans un nombre minimal de bacs de capacité fixe. L'objectif est de minimiser le nombre total de bacs utilisés tout en respectant la contrainte de capacité de chaque bac. Ce problème est considéré comme NP-complet, ce qui signifie qu'il n'existe pas d'algorithme efficace pour trouver la solution optimale en un temps polynomial. Il a de nombreuses applications pratiques dans des domaines tels que la logistique, l'ordonnancement des tâches et la gestion des ressources. La recherche de stratégies efficaces pour résoudre le problème du bin packing a suscité l'intérêt de la communauté scientifique et a donné lieu à de nombreuses approches heuristiques et métaheuristiques.

A. *Bin Packing*

Le problème de bin packing peut être formulé comme un problème de programmation mathématique, comme suite :

$\forall i = 1, 2, ..., n \quad \forall j = 1, 2, ..., m$

$$\begin{cases} Z : \text{Le nombre total de conteneurs utilisés} \\ W : \text{La capacité maximale d'un conteneur} \\ W_i : \text{La taille de l'objet i} \\ x_{ij} \in \{0, 1\} : 1 \text{ si l'objet i est placé dans le conteneur j, 0 sinon} \\ y_j \in \{0, 1\} : 1 \text{ si le conteneur j contient au moins un objet, 0 sinon} \end{cases}$$

L'objectif est de minimiser Z, c'est-à-dire le nombre total de conteneurs utilisés. donc on peut écrire la fonction objectif comme suite :

Min $Z = \sum_{j=1}^{m} y_j$

Sous les 3 contraintes C1, C2 et C3 :

$$\begin{cases} \sum_{j=1}^{m} x_{ij} = 1 & \forall i = 1, 2, ..., n \\ (C1) & \\ \sum_{i=1}^{n} W_i \times x_{ij} \leqslant W & \forall j = 1, 2, ..., m \\ (C2) & \\ x_{ij} \leqslant y_j & \forall j = 1, 2, ..., m \\ (C3) & \end{cases}$$

$(C1)$: Chaque objet doit être placé dans un conteneur
$(C2)$: La capacité maximale de chaque conteneur n'est pas dépassée
$(C3)$: Chaque objet i est soit dans ce conteneur j , soit ne l'est pas

B. *Water Wave Optimization*

Voici une représentation générale des principales équations utilisées dans le WWO :
Soit X un ensemble de solutions potentielles, la fonction objectif à minimiser est notée f(x).

a) Propagation de la vague:

$$\boxed{x'(d) = x(d) + \text{Rand}(-1, 1) \cdot \lambda \cdot L(d)} \quad (1)$$

Où :

- x'(d) : la nouvelle position de la vague dans la dimension d
- x(d) : la position actuelle de la vague dans la dimension d
- Rand(-1,1) génère un nombre aléatoire uniformément distribué entre -1 et 1
- λ : un paramètre de contrôle qui régule l'amplitude du mouvement de propagation
- L(d) : une valeur caractéristique de la dimension d

L'objectif de cette propagation est d'explorer de nouvelles régions de l'espace de recherche en permettant des mouvements stochastiques.

b) Réfraction de la vague:

$$\boxed{x'(d) = \mathcal{N}\left(\frac{x^*(d) + x(d)}{2}, \frac{|x^*(d) - x(d)|}{2}\right)} \quad (2)$$

- $\mathcal{N}(\mu, \sigma)$ représente une distribution normale avec une moyenne μ et un écart-type σ.
- $x^*(d)$: la meilleure position trouvée jusqu'à présent dans la dimension d.
- x(d) : la position actuelle de la vague dans la dimension d.

Cette formule permet d'introduire un mouvement de la vague en direction de la meilleure position trouvée $x^*(d)$, tout en conservant une certaine variation aléatoire basée sur la différence entre $x^*(d)$ et x(d).

c) Rupture des vagues:

$$\boxed{x'(d) = x(d) + \mathcal{N}(0, 1) \cdot \beta \cdot L(d)} \quad (3)$$

- $\mathcal{N}(0, 1)$ est un nombre aléatoire entre 0 et 1 qui suit la loi normale.
- β est un coefficient qui contrôle l'amplitude de la perturbation.
- La longueur d'onde L(d) détermine la magnitude de la perturbation dans cette dimension.

Il est important de noter que les équations précédentes sont une représentation générale du WWO, et on va par la suite les adapter en fonction du problème du bin packing et des objectifs recherchés.

C. *Recuit Simulé*

Voici les formules principales utilisées dans l'algorithme du recuit simulé :

- x : Solution courante (représentation d'une solution partielle ou complète du problème).
- f(x) : Valeur de la fonction objectif associée à la solution x.
- T : Paramètre de température, qui contrôle l'exploration et l'acceptation des solutions voisines.
- α' : Facteur de diminution de la température, généralement compris entre 0 et 1.

a) Génération d'une solution voisine:

$$\boxed{x' = N(x)} \quad (4)$$

Où : N(x) est une fonction qui génère une solution voisine à partir de la solution courante x.

b) Calcul de la différence de coût:

$$\boxed{\Delta E = f(x') - f(x)} \quad (5)$$

ΔE représente la différence de la valeur de la fonction objectif entre la nouvelle solution x' et la solution courante x.

c) Probabilité d'acceptation de la nouvelle solution:

$$\boxed{P(\text{accepter}) = e^{-\frac{\Delta E}{T}}} \quad (6)$$

d) Mise à jour de la température:

$$\boxed{T = \alpha \cdot T} \quad (7)$$

Ces formules sont utilisées pour déterminer si une nouvelle solution doit être acceptée ou rejetée lors de l'exploration de l'espace des solutions.

III. DESCRIPTION DE L'APPROCHE PROPOSÉE

Dans cette section, nous présentons brièvement les deux métaheuristiques que nous avons choisi d'hybrider, à savoir le Recuit Simulé (RS) et le Water Wave Optimization (WWO).

A. *Recuit Simulé*

Le Recuit Simulé (RS) est une métaheuristique inspirée du processus de refroidissement d'un matériau pour atteindre un état d'équilibre thermodynamique. Le principe du RS repose sur la simulation d'un processus de chauffage et de refroidissement progressif de la solution courante. Cela permet d'explorer l'espace des solutions en acceptant également des solutions moins performantes, afin d'éviter les minima locaux. Le RS est largement utilisé pour résoudre des problèmes d'optimisation combinatoire et se caractérise par sa capacité à effectuer des mouvements de grande amplitude et à éviter de rester piégé dans des régions de recherche peu prometteuses.

B. *World Water Optimization*

Le World Water Optimization (WWO) est une métaheuristique qui s'inspire des phénomènes de propagation, de réfraction et de rupture observés dans le comportement des molécules d'eau. Ces phénomènes font référence à la manière dont l'eau se propage, se déforme et se redistribue dans les écosystèmes naturels. Le WWO tire parti de ces mécanismes

pour guider sa recherche de solutions de haute qualité.

- **La propagation (Propagation)** : C'est la phase où les solutions sont explorées et diffusées dans l'espace de recherche. Cela permet d'explorer différentes régions de l'espace des solutions afin de détecter les meilleures solutions locales.
- **La réfraction (Refraction)** : elle représente l'adaptation et la transformation des solutions en fonction des contraintes et des dépendances spécifiques du problème.
- **La rupture (Breaking)** : C'est le processus par lequel les solutions sont modifiées ou fragmentées pour atteindre des améliorations locales significatives.

En combinant ces phénomènes et en utilisant une recherche locale intelligente, le WWO vise à améliorer progressivement la qualité des solutions en se concentrant sur les meilleurs éléments locaux. Cette approche est particulièrement adaptée aux problèmes complexes avec des contraintes et des dépendances importantes entre les variables, où une amélioration progressive de la solution est essentielle.[5]

C. *Pourquoi hybrider WWO avec RS*

Nous avons choisi d'hybrider le RS et le WWO pour résoudre le problème de bin packing en raison de leurs avantages respectifs et de leurs complémentarités.

- Le Recuit Simulé (RS) est connu pour sa capacité de faire une recherche intensive tout en donnant une chance à explorer l'espace des solutions de manière globale, en effectuant des mouvements de grande amplitude. Il est efficace pour éviter de rester piégé dans des minima locaux et peut fournir des solutions de haute qualité.

- Le WWO, en revanche,est une métaheuristique qui manipule une population tout en améliorant la solution locale tout en améliorant la solution de chaque individu de cette population dans certains cas.

En combinant le RS et le WWO, nous pouvons tirer parti de leurs forces respectives. Le RS peut fournir une exploration globale de l'espace des solutions pour éviter les minima locaux, tandis que le WWO peut exploiter les meilleures solutions locales pour les améliorer progressivement. Cette combinaison peut potentiellement améliorer les performances de résolution du problème de bin packing, en fournissant une recherche plus efficace et une meilleure qualité de solution.

D. *Type d'hybridation et Schéma global*

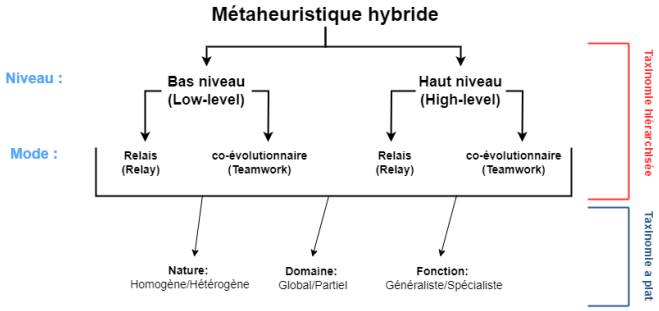

Fig. 1. Diagramme de classification des méthodes hybrides

[6] L'hybridation réalisée entre le RS et le WWO dans le cadre de cette étude suit le type LRH (Low-level Relay Hybrid) selon la taxinomie de Talbi. Dans cette approche, les composants des deux métaheuristiques sont combinés de manière où le RS est utilisé en tant que composant interne dans WWO pour explorer l'espace des solutions, et le WWO est utilisé comme un composant global pour exploiter les solutions locales.

E. *Détail de l'hybridation*

Dans l'algorithme WWO, une partie spécifique a été remplacée par un segment de l'algorithme RS. Ce remplacement intervient lors de l'évaluation d'un nouveau candidat solution, noté x'. Dans l'algorithme WWO d'origine, si x' surpasse la meilleure solution actuelle (x*), x' est décomposé selon l'équation (2) et x* est mis à jour en conséquence. Cependant, dans la version hybride, si x' ne surpasse pas x* mais satisfait une certaine condition, il est accepté comme nouvelle solution avec une probabilité déterminée par le paramètre de température T. Ce processus de prise de décision est basé sur le critère d'acceptation probabiliste de l'algorithme RS. En incorporant cette composante du RS, l'algorithme hybride acquiert la capacité d'explorer des solutions potentiellement meilleures même lorsque le critère d'amélioration immédiate n'est pas satisfait.

Voici les schémas respectifs de WWO, RS et hybridation WWO-RS, les parties encadrées sont ceux utilisés pour l'hybridation.

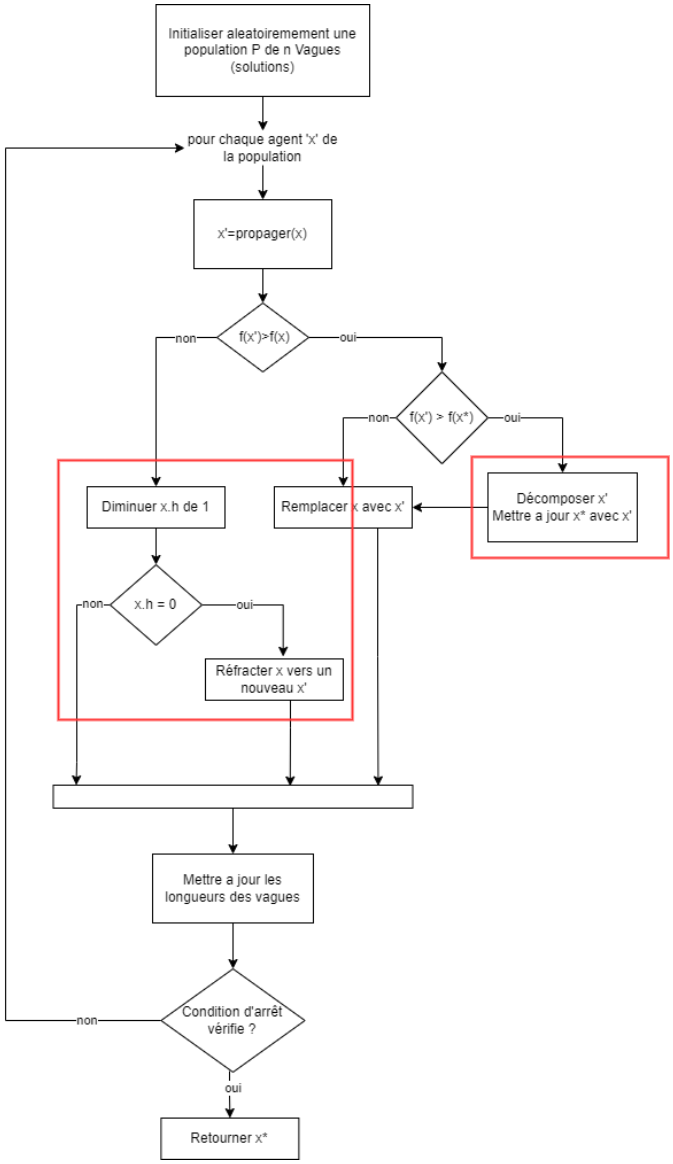

Fig. 2. Schéma expliquant déroulement de l'algorithme du WWO

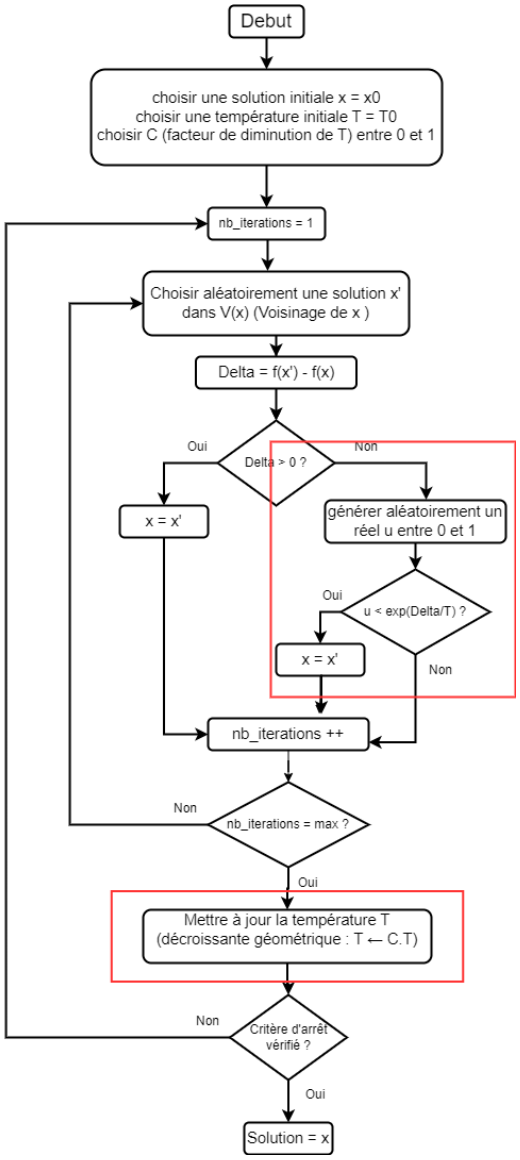

Fig. 3. Schéma expliquant déroulement de l'algorithme du RS

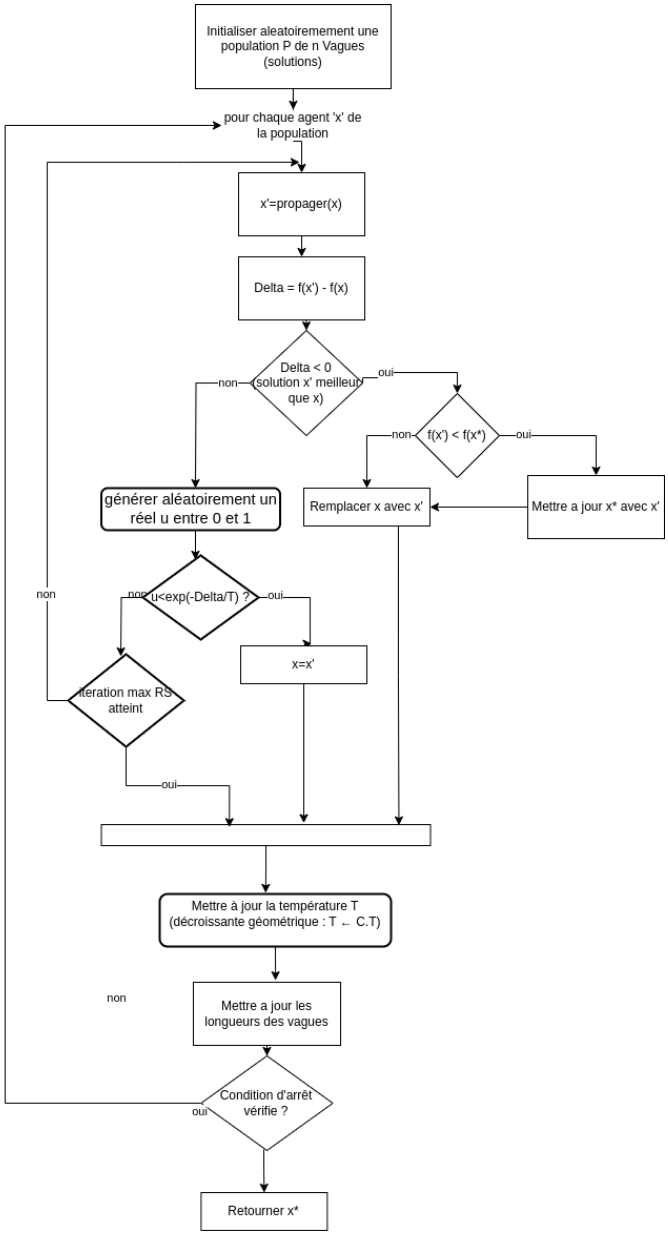

Fig. 4. Schéma expliquant l'hybridation entre WWO et RS

Ci-contre, voici les deux algorithmes relatifs à la méthode du recuit simulé [7] et du water waves optimizer :

Algorithm 1 Recuit Simulé (RS)

Choisir une solution initiale $x_0 \in S$
Choisir une température initiale T_0
Choisir $\alpha \in (0, 1)$ (facteur de diminution de T)
tant que *Critère d'arrêt non vérifié* **faire**
  tant que *il reste des x à visiter* **faire**
    Choisir aléatoirement $x' \in V(x)$ (une solution voisine de x)
    $\Delta = f(x') - f(x)$
    si $\Delta > 0$ **alors**
      $x \leftarrow x'$
    sinon
      Générer aléatoirement un réel $u \in [0, 1]$
      si $u < e^{\Delta/T}$ **alors**
        $x \leftarrow x'$
      fin
    fin
  fin
  MàJ la température (décroissante géométrique : $T \leftarrow \alpha T$)
fin

Algorithm 2 Water Waves Optimizer (WWO)

Initialiser aléatoirement une population P de n vagues (n solutions)
tant que *Critère d'arrêt non vérifié* **faire**
  tant que *il reste des x à visiter* **faire**
    Propager x vers un nouveau x' en utilisant l'équation (1)
    si $f(x') > f(x)$ **alors**
      si $f(x') > f(x^*)$ **alors**
        Décomposer x' en fonction de l'équation (2)
        Mettre à jour x^* avec x'
      fin
      Remplacer x avec x'
    sinon
      Diminuer $x.h$ de un
      si $x.h = 0$ **alors**
        Réfracter x vers un nouveau x' en utilisant les équations (3) et (4)
      fin
    fin
  fin
  Màj les longueurs d'onde en utilisant l'équation (5)
fin
Retourner x^*

F. Encodage de la solution

Pour notre cas (le problème du bin packing), on a décidé de présenter la solution comme une matrice de nxn où chaque ligne représente un bin et chaque colonne est un bit (prend la valeur 0 ou 1) pour spécifier l'existence d'un objet j dans un bin i.
si un objet existe dans un bin alors il n'existerai pas dans les autres.

l'objet 2 et 3 sont dans le même sont dans le même bin 2 l'objet 1 est dans 2 bins au même temps :

TABLE I
POSITIONNEMENT ACCEPTABLE

1	0	0	0	0	0
0	1	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	1
0	0	0	0	0	0
0	0	0	0	0	0

TABLE II
POSITIONNEMENT ACCEPTABLE

1	0	0	0	0	0
1	1	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	1
0	0	0	0	0	0
0	0	0	0	0	0

TABLE III
PIRE POSITIONNEMENT ACCEPTABLE UN OBJET DANS CHAQUE BIN)

1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

G. Description détaillée de chaque étape de l'approche proposée appliquée au problème traité

1) Génération d'une population initiale à partir des heuristiques. Cette population est considérée comme un ensemble de vagues.
2) Initialisation de la solution optimale.
3) Pour chaque vague x de la population, faire :
   a) Propager x vers x'.
   b) Calculer la différence $\Delta = f(x') - f(x)$.
   c) Si $\Delta < 0$:
      i) Vérifier si x' est une meilleure solution que la solution optimale. Si c'est le cas, $x^* \leftarrow x'$.
      ii) Mettre à jour $x : x \leftarrow x'$.
   d) Sinon :
      i) Générer un nombre aléatoire entre 0 et 1.
      ii) Si $u < \exp(\Delta/T)$, mettre à jour $x : x \leftarrow x'$ (permettre un certain degré de diversification).
      iii) Sinon :
         A) Si cptRS > nb_max_RS, aller à g.
         B) Aller à a.
   e) Mettre à jour cptRS.
   f) Mettre à jour la température $T : T \leftarrow C \cdot T$.
   g) Mettre à jour les longueurs des vagues.
   h) Si le critère d'arrêt n'est pas satisfait, aller à l'étape 3.
4) Retourner x^*.

IV. TESTS ET RÉSULTATS

L' environnement et les caractéristiques de nos machines ont été soigneusement sélectionnés pour assurer une exécution efficace et précise des tests, ainsi qu'une analyse approfondie des résultats obtenus lors de l'optimisation du problème du bin packing.

Environnement et langages de programmation: **Python** est un langage polyvalent et largement utilisé dans le domaine de l'optimisation.
VSCode est un environnement de développement intégré (IDE) populaire, offrant une interface conviviale pour écrire, déboguer et exécuter du code Python.

Bibliothèques utilisées: **Numpy** : Une bibliothèque Python puissante pour le calcul numérique. Numpy fournit des structures de données efficaces et des fonctions de calcul avancées pour manipuler des tableaux multidimensionnels.
Pandas : Une bibliothèque Python utilisée pour la manipulation et l'analyse des données. Pandas offre des structures de données flexibles pour le traitement et la manipulation des jeux de données, notamment les DataFrames.

Système d'exploitation: **Windows** : Le système d'exploitation utilisé pour le développement et les tests. Windows offre une interface utilisateur conviviale et une compatibilité étendue avec de nombreuses bibliothèques Python.

Caractéristiques de la machine: **RAM** : 16 Go de mémoire vive. Une quantité de RAM suffisante pour gérer les tâches de calcul ainsi que les jeux de données volumineux.
CPU : Intel Core i7 de 10e génération. Un processeur puissant et performant, capable de traiter efficacement les calculs complexes et de gérer plusieurs tâches simultanément.

L'utilisation de Python comme langage de programmation, en combinaison avec les bibliothèques Numpy et Pandas, offre un environnement flexible et puissant pour la manipulation des données et l'optimisation. Le choix de Windows comme système d'exploitation, associé à une configuration matérielle solide comprenant 16 Go de RAM et un processeur Intel Core i7 de 10e génération, garantit des performances optimales lors de l'exécution des différents tests envisagés.

A. *Présentation des instances utilisées*

Nous avons pris soin de sélectionner une variété d'instances pour évaluer les performances de notre approche. Ces instances étaient composées de 18 benchmarks différents , et nous avons fait varier la capacité des bins ainsi que le nombre d'objets à stocker à chaque itération des tests. Cette approche nous a permis d'explorer un large éventail de scénarios réalistes et de recueillir des données significatives sur l'efficacité de notre méthode d'optimisation.

Le tableau des benchmarks choisis ainsi que leurs caractéristiques est décrit au dessous:

TABLE IV
INSTANCES UTILISÉES

Benchmarks	Nombre Objets	Capacité Bacs
BPP_50_50_0.1_0.7_7	50	50
BPP_50_400_0.2_0.8_4	50	400
BPP_100_750_0.2_0.7_0	100	750
BPP_200_1000_0.2_0.8_8	200	1000
BPP_300_50_0.1_0.7_0	300	50
BPP_500_50_0.1_0.7_2	500	50
BPP_750_500_0.2_0.8_8	750	500
BPP_750_750_0.2_0.8_9	750	750
BPP_750_1000_0.2_0.8_9	750	1000
BPP_1000_50_0.1_0.7_1	1000	50
BPP_1000_125_0.2_0.8_9	1000	125
BPP_1000_150_0.2_0.8_9	1000	150
BPP_1000_200_0.2_0.8_9	1000	200
BPP_1000_300_0.2_0.8_9	1000	300
BPP_1000_400_0.2_0.8_9	1000	400
BPP_1000_500_0.2_0.8_9	1000	500
BPP_1000_750_0.2_0.8_9	1000	750
BPP_1000_1000_0.1_0.7_0	1000	1000

B. *Calibrage des paramètres*

Lors de nos tests d'optimisation du problème du bin packing, nous avons accordé une attention particulière au calibrage des paramètres. En effet ,le calibrage des paramètres joue un rôle crucial dans l'efficacité et la performance de notre méthode d'optimisation. Il consiste à ajuster les valeurs des paramètres utilisés dans notre approche afin d'obtenir les meilleurs résultats possibles.

1) Paramètres du WWO:

- Instance du type de problème
  - Nombre d'objets
  - Liste d'objets
  - Capacité du conteneur (Bin)
- La largeur maximale d'onde Hmax
- Le coefficient de réduction de la longueur d'onde α

2) Paramètres du Récuit Simulé:

- Température initiale T_0.
- Le coefficient de réduction de la température α'.
- La fonction de réduction de la température (géométrique).

Nous avons pris en compte plusieurs critères lors de la sélection des valeurs de paramètres, tels que l'équilibre entre l'exploration et l'exploitation, la diversité des solutions explorées, ainsi que la capacité à éviter les pièges locaux. Nous avons également tenu compte des caractéristiques spécifiques des benchmarks utilisés, en ajustant les paramètres pour optimiser les performances sur chaque instance.Nos choix finaux sont illustrés ci-contre :

TABLE V
PARAMÈTRES UTILISÉS

Paramètre	Choix
Largeur maximale d'onde	0.5
Coefficient de réduction de la longueur d'onde α	1,01
Largeur maximale d'onde	0,5
Coefficient de réduction de la température α'	0.9

C. *Étude des performances*

L'étude de performance que nous avons réalisée vise à évaluer l'efficacité et l'efficience de notre méthode d'optimisation hybride du problème du bin packing. Cette étude nous permet de mesurer les performances de notre approche dans différents aspects, tels que la qualité des solutions obtenues, le temps de calcul nécessaire et la consommation de ressources. Elle nous offre un aperçu précis de la capacité de notre méthode à résoudre efficacement le problème du bin packing.

Les résultats d'exécution obtenus pour les **18** benchmarks, avec un nombre d'itérations égal à **100**, sont résumés dans la table ci-dessous :

Résultats des Benchmarks

Benchmarks	Solution Exacte	Solution WWO sans Hybridation	Solution Hybridation	T_ex WWO sans Hybridation	T_ex Hybridation
BPP_50_50_0.1_0.7_7	23	31	29	0,42s	0,47s
BPP_50_400_0.2_0.8_4	36	41	40	0,52s	0,4s
BPP_100_750_0.2_0.7_0	45	64	59	1,41s	1,5s
BPP_200_1000_0.2_0.8_8	110	145	136	4,86s	4,7s
BPP_300_50_0.1_0.7_0	129	181	173	9,93s	9,8s
BPP_500_50_0.1_0.7_2	198	276	257	26,15s	28s
BPP_750_500_0.2_0.8_8	385	529	527	63,69s	67s
BPP_750_750_0.2_0.8_9	385	532	530	64s	64,3s
BPP_750_1000_0.2_0.8_9	394	540	525	65s	67s
BPP_1000_50_0.1_0.7_1	391	544	517	121s	115s
BPP_1000_125_0.2_0.8_9	498	695	683	129s	131s
BPP_1000_150_0.2_0.8_9	525	712	699	120s	121s
BPP_1000_200_0.2_0.8_9	508	697	680	118s	123s
BPP_1000_300_0.2_0.8_9	516	700	685	116s	117s
BPP_1000_400_0.2_0.8_9	504	702	669	137s	135s
BPP_1000_500_0.2_0.8_9	510	707	687	116s	126s
BPP_1000_750_0.2_0.8_9	523	718	702	116s	131s
BPP_1000_1000_0.1_0.7_0	397	563	528	116s	125s

Analyse générale des résultats

Les résultats obtenus précédemment ont conduit aux conclusions suivantes :

- Les performances sont améliorées grâce à l'hybridation : Dans la majorité des cas, la solution hybride, qui combine l'algorithme du WWO avec celui du recuit simulé, produit de meilleurs résultats que la solution WWO seule. Cette amélioration est observée en comparant les valeurs des solutions WWO sans hybridation avec celles de la solution hybride dans chaque cas de test.
- Les temps d'exécution des solutions hybrides sont comparables : Malgré l'ajout du recuit simulé, l'hybridation ne semble pas avoir un impact significatif sur le temps d'exécution global. Les temps d'exécution des solutions hybrides sont similaires, voire légèrement inférieurs, à ceux des solutions WWO sans hybridation.
- Amélioration des résultats variables : L'amélioration des résultats obtenus avec l'hybridation varie d'un cas de test à l'autre. Dans certains cas, la différence est modérée, tandis que dans d'autres, elle est plus marquée. Par exemple, pour le cas de test "BPP_50_50_0.1_0.7_7", la solution hybride a une valeur de 31, tandis que la solution WWO sans hybridation a une valeur de 23. En revanche, pour le cas de test "BPP_1000_50_0.1_0.7_1", la différence est plus faible, avec des valeurs respectives de 544 et 391.
- Impact de la taille du problème : Les différences entre les solutions WWO sans hybridation et les solutions hybrides semblent s'intensifier avec l'augmentation de la taille du problème. Pour les cas de test de plus grande taille, tels que "BPP_1000_500_0.2_0.8_9" et "BPP_1000_750_0.2_0.8_9", les valeurs des solutions hybrides sont nettement meilleurs à celles des solutions WWO sans hybridation.

En général, nous pouvons observer que l'hybridation de Water Wave Optimizer avec le Recuit Simulé permet d'améliorer les résultats obtenus en termes de fitness. L'utilisation du triage contribue également à améliorer la qualité des solutions, bien que cela entraîne une légère augmentation du temps d'exécution.

Il est important de noter que l'impact des paramètres spécifiques utilisés dans ces tests (tels que : la largeur maximale d'onde, le coefficient de réduction de la longueur d'onde α , le coefficient de rupture β etc ..) peut être observé dans les résultats obtenus, ainsi les critères choisis se sont révélés appropriés en fonction de l'évolution de l'algorithme.

D. Etude Comparative

Pour évaluer les performances de notre approche hybride, nous avons effectué des tests sur benchmarks de paramètres différents. Ces benchmarks ont été soigneusement sélectionnés pour représenter des scénarios réalistes du problème du bin packing. Notre évaluation s'est principalement concentrée sur deux aspects clés : le temps d'exécution et la fitness des solutions obtenues.

Dans cette étude comparative, nous visualisons les résultats de nos tests sur 18 benchmarks, en mettant l'accent sur le temps d'exécution et la fitness des solutions obtenues. Ces résultats nous permettront d'évaluer l'efficacité et l'efficience de notre approche hybride, ainsi que de comparer ses performances avec celles des métaheuristiques individuelles; WWO avec RT, WWO sans RT et les autres littératures.

1) ***Temps d'exécution****:* L'histogramme ci-dessus représente les variations des temps d'exécution pour les 18 benchmarks étudiés. Il offre une visualisation claire des résultats du tests.

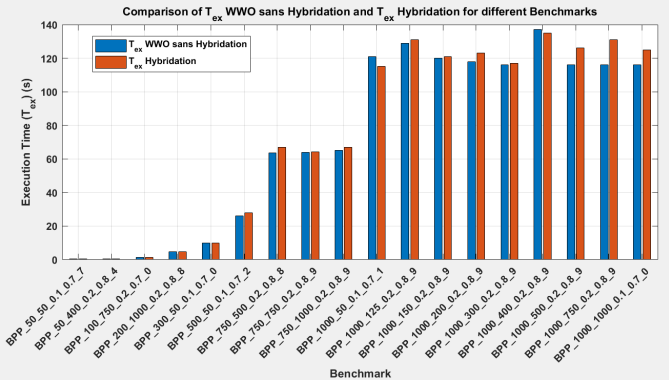

Fig. 5. Histogramme de variation de temps d'exécution pour les benchmarks

En observant le graphe, nous pouvons constater que les temps d'exécution varient en fonction du benchmark (nombre des objets et la taille maximal du conteneurs).

- Temps d'exécution cohérent : Globalement, les temps d'exécution des solutions hybrides sont comparables, voire légèrement inférieurs, à ceux des solutions WWO sans hybridation. Cela indique que l'ajout du recuit simulé n'a pas entraîné une augmentation significative du temps de calcul.

- Consistance des performances : Malgré des différences dans les temps d'exécution, les solutions hybrides montrent généralement des performances similaires ou supérieures par rapport aux solutions WWO sans hybridation. Cela suggère que l'hybridation a permis d'améliorer la qualité des solutions tout en maintenant des temps d'exécution raisonnables.

- Évolutivité du temps d'exécution : Il est intéressant de noter que, dans certains cas, le temps d'exécution de la solution hybride est inférieur à celui de la solution WWO sans hybridation, malgré des améliorations des performances. Cela indique que l'hybridation peut offrir une meilleure efficacité en termes de temps de calcul, ce qui est particulièrement bénéfique pour les problèmes plus grands.

En résumé, le graphe met en évidence les variations des temps d'exécution pour les 18 benchmarks et permet de visualiser les différences entre les deux approches (sans et avec hybridation). Il confirme nos conclusions antérieures selon lesquelles le triage peut réduire le temps d'exécution dans certains cas, mais son impact peut être mitigé en fonction de la complexité du problème et surtout en fonction des nombres d'objets utilisés dans notre problème de Bin packing; tel que, plus le nombre d'objet augmente plus la solution avec triage devient plus coûteuse en temps d'exécution.

2) ***Fitness****:* Ce schéma présente les évolutions des fitness pour les benchmarks examinés, en comparant les deux approches (WWO sans et avec hybridation) avec une approche exacte conçue en utilisant la méthode branch & bound :

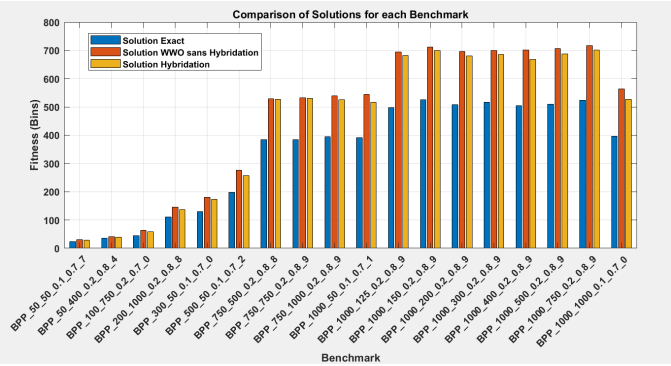

Fig. 6. Histogramme de variation de fitness pour les benchmarks

En observant le graphe, nous pouvons constater que les fitness varient en fonction du benchmark et de l'effet d'hybridation.

- Performances de la solution hybride : Les solutions hybrides semblent généralement produire des valeurs de fitness meilleures que celles du WWO sans hybridation. Cependant, sans les valeurs optimales, il est difficile de déterminer si les solutions hybrides se rapprochent de la solution exacte ou optimale.

- En se basant sur les valeurs de fitness disponibles, nous pouvons constater que la méthode hybride, qui combine le WWO avec le recuit simulé, semble produire des résultats prometteurs. Les solutions hybrides affichent généralement des valeurs de fitness meilleures ou équivalentes à celles du WWO sans hybridation.Cette observation suggère que l'incorporation du recuit simulé dans le processus d'optimisation permet d'améliorer la qualité des solutions obtenues. L'hybridation offre une perspective intéressante en apportant une certaine diversification et une exploration plus efficace de l'espace de recherche.

- Impact de la taille du problème : Il est important de noter que les valeurs de fitness peuvent être influencées par la taille du problème. Les cas de test de plus grande taille peuvent présenter des écarts plus importants entre les solutions WWO sans hybridation et les solutions hybrides.

En résumé, les résultats montrent que l'approche exacte atteint les solutions optimales pour les benchmarks étudiés. L'approche avec hybridation tend à améliorer la fitness par rapport à l'approche sans hybridation par rapport au problème de bin packing. L'approche avec triage parvient à optimiser plus efficacement l'utilisation de l'espace puisqu'en intégrant le processus de triage des objets avant leur affectation aux bacs, l'approche avec triage parvient à mieux organiser les objets et à réduire davantage l'espace gaspillé, se rapprochant ainsi de la solution optimale obtenue par l'approche exacte.

3) ***Comparaison avec des méthodes déja existantes****:* Voici une comparaison théorique entre cette approche hybride et d'autres algorithmes métaheuristiques:

Optimiseur dépendant de la fitness (FDO)

Le FDO a été développé en 2019 par Abdullah et Ahmed [8]. Il s'agit d'un algorithme d'intelligence de swarm qui modélise les caractéristiques du processus de reproduction des essaims d'abeilles, ainsi que leur comportement de prise de décision collective. Bien que le FDO soit considéré comme un algorithme basé sur le PSO, il calcule la vitesse d'une particule différemment.

Il utilise la valeur de la fonction de fitness du problème pour produire des poids. Ces poids guident ensuite les agents de recherche de l'algorithme lors des phases d'exploitation et d'exploration. De plus, l'algorithme FDO nécessite moins de calculs que l'algorithme PSO pour mettre à jour la vitesse et les positions des particules [7]. Les abeilles sont des insectes sociaux qui vivent dans de petites grottes ou des arbres creux. Elles travaillent en colonies généralement appelées ruches. Les trois types d'abeilles dans une colonie comprennent la reine, les ouvrières et les abeilles éclaireuses. Chacune de ces abeilles a des fonctions spécifiques en fonction de leurs caractéristiques. L'algorithme FDO se concentre sur la fonction des abeilles éclaireuses. Ces abeilles explorent leur environnement pour trouver un endroit approprié pour construire une ruche (autrement dit, elles exploitent des ruches préférables). Une fois cela trouvé, les abeilles effectuent une sorte de "danse" pour communiquer avec l'essaim [9]. Dans l'algorithme, chaque ruche exploitée par une abeille éclaireuse est considérée comme une solution candidate, et la meilleure ruche représente une solution optimale globale, selon le poids de fitness relatif.

Les avantages de cet algorithme sont qu'il est stable à la fois dans les phases d'exploitation et d'exploration, et qu'il nécessite moins de calculs que d'autres algorithmes. L'inconvénient est qu'il utilise un facteur de poids pour contrôler les poids de fitness qui sont, dans la plupart des cas, ignorés [10]. Abdul-Minaam et al. [8] ont tenu compte de ces forces et faiblesses. Ils ont adapté le FDO pour résoudre le problème du bin packing 1D en remplaçant la

méthode originale de génération de la première population par une génération aléatoire de la population initiale à l'aide d'une heuristique First Fit améliorée, et en mettant à jour les stratégies opérationnelles de l'algorithme original pour améliorer les phases d'exploration et d'exploitation.

Recherche de coucou via les vols de Lévy En 2009, Yang et Deb ont développé l'algorithme

Métaheuristique de recherche de coucou basé sur une population via les vols de Lévy (CS) [11]. Cet algorithme s'inspire de la stratégie de reproduction de certaines espèces de coucous, qui ont tendance à pondre leurs œufs dans les nids d'oiseaux d'autres espèces ou d'oiseaux hôtes. Souvent, ces coucous choisissent un nid qui contient des œufs récemment pondus. Un oiseau hôte peut réagir à la présence d'un œuf étranger dans son nid de deux manières - il peut se débarrasser de l'œuf étranger en le jetant, ou il peut abandonner tout le nid et le reconstruire ailleurs. Certains coucous ont évolué de telle manière que les femelles parasites peuvent imiter les couleurs et les motifs des œufs des oiseaux hôtes, réduisant ainsi la probabilité que leurs œufs soient abandonnés. Cette technique conduit à une augmentation de la reproductivité [11]. Ces œufs de coucou ont tendance à éclore avant les autres œufs présents dans le nid. Le premier réflexe du poussin est d'expulser les œufs de l'hôte en les propulsant aveuglément hors du nid. Cette action entraîne une augmentation de la quantité de nourriture fournie par l'oiseau hôte au poussin coucou. Le mécanisme des vols de Lévy remplace la simple marche aléatoire pour améliorer les performances de l'algorithme [22]. Ce mécanisme de vol est décrit comme une "marche aléatoire dans laquelle les longueurs des pas sont calculées avec une distribution de probabilité à queue lourde" [12].

Les avantages de cet algorithme sont qu'il utilise beaucoup moins de paramètres que la plupart des métaheuristiques ; par conséquent, peu de réglages sont nécessaires, il est facile à mettre en œuvre et peut être facilement hybridé avec d'autres algorithmes. Un inconvénient majeur est que la convergence rapide ne peut pas être garantie car elle dépend des marches aléatoires [13]. Zendaoui et Layeb [14] ont utilisé ces forces pour aider leur tentative d'adapter l'algorithme CS pour résoudre le problème du bin packing 1D en introduisant un mécanisme intéressant, la valeur de classement ordonné (ROV), afin de convertir les résultats de l'algorithme original d'un espace continu à un espace discret.

The Whale Optimization Algorithm (WOA)

Le WOA a été développé en 2016 par Mirjalili et Lewis [15]. Cet algorithme est une métaheuristique basée sur l'intelligence de swarm, et il imite la stratégie de chasse des baleines à bosse. Ces baleines sont connues pour leur capacité à utiliser une stratégie de chasse inhabituelle, également appelée stratégie du "filet de bulles". Les baleines parviennent à piéger leurs proies, les poissons, en créant un cercle d'air. Les baleines y parviennent en émettant un flux de bulles. Ces bulles prennent deux formes différentes : la première est une spirale, et la seconde est un cercle rétrécissant. Après avoir piégé leurs proies dans ces bulles, les poissons sont avalés par les baleines à bosse. Dans l'algorithme, la phase d'exploration est représentée par la recherche de proies, tandis que la stratégie de chasse utilisant les formes rétrécissantes et en spirale représente la phase d'exploitation. Les positions des proies sont initialisées de manière aléatoire - cela représente la population initiale, qui est évaluée pour trouver la meilleure solution actuelle.

Les avantages de cet algorithme comprennent la simplicité de mise en œuvre et le faible coût computationnel. En revanche, l'inconvénient est que l'algorithme peut parfois conduire à une stagnation dans les optima locaux, et que la phase d'exploitation serait plus bénéfique si elle était améliorée [16]. Abdel-Basset et al. [17] ont utilisé ces avantages pour adapter le WOA à la résolution du problème du bin packing 1D, appelé Improved Lévy-based Whale Optimization Algorithm (ILWOA), développé pour résoudre le problème du bin packing. Cela a été réalisé en intégrant divers mécanismes, notamment la technique de la valeur d'ordre la plus élevée (LOV) (afin de convertir les résultats continus en résultats discrets), les vols de Lévy et les cartes chaotiques pour surmonter les faiblesses du WOA original.

Algorithme de recherche d'écureuils

L'algorithme d'optimisation inspiré de la recherche d'écureuils (SSA - Squirrel Search Algorithm) a été développé en 2018 par Jain et al. [17]. Cet algorithme imite le comportement des écureuils volants et leur déplacement efficace. Les écureuils volants ne volent pas, mais utilisent plutôt une technique spéciale appelée le planeur. Cette technique est considérée comme très efficace car elle est économe en énergie et permet aux petits animaux de parcourir de grandes distances rapidement [18]. Il y a trois hypothèses principales dans le SSA [17]. Premièrement, il y a un certain nombre d'écureuils volants dans la forêt. Deuxièmement, chacun de ces écureuils cherche de la nourriture et utilise les ressources alimentaires disponibles de manière dynamique. Enfin, il y a trois types d'arbres dans la forêt mentionnée précédemment les arbres normaux, les arbres à glands et les arbres à noix de pécan.

Un avantage de cet algorithme est qu'il permet une exploration efficace de l'espace de recherche. Les inconvénients, en revanche, sont qu'il souffre d'une convergence prématurée et que, lors de la résolution de problèmes complexes, il est possible que l'algorithme se retrouve piégé dans un optimum local [19]. El-Ashmawi et Abd Elminaam [20] ont modifié le SSA pour résoudre le problème du bin packing 1D en tenant compte de ces faiblesses, dans le but de les améliorer. L'algorithme modifié a intégré une randomisation dans la population initiale et des ajustements des stratégies opérationnelles pour améliorer les nouvelles générations pendant l'exploration.

Algorithme génétique (GA)

Le GA a été initialement proposé par Holland [21] et a été appliqué à une vaste gamme de problèmes d'optimisation. Cet algorithme est basé sur le processus biologique de l'évolution et de la sélection naturelle. Le GA utilise une population de solutions candidates qui évoluent au fil des générations grâce à des opérations génétiques telles que la sélection, le croisement et la mutation. Les individus les plus adaptés (avec les meilleurs scores de fitness) ont une plus grande probabilité d'être sélectionnés pour la reproduction, ce qui conduit à une amélioration globale de la population au fil du temps.

Les avantages de cet algorithme incluent sa simplicité de mise en œuvre et sa capacité à trouver des solutions de haute qualité. Cependant, il peut parfois converger prématurément vers un optimum local et peut nécessiter des paramètres spécifiques pour des problèmes spécifiques [22]. Wang et al. [23] ont adapté l'algorithme génétique pour résoudre le problème du bin packing 1D en utilisant une représentation de chromosome améliorée et en introduisant des opérations génétiques spécifiques pour gérer les contraintes du problème.

References

[1] Natalia Hartono, Asrul Harun Ismail, Sultan Zeybek, Mario Caterino, Kaiwen Jiang, Murat Şahin, and Christine Natalia. The 1-d bin packing problem optimisation using bees algorithm. *Journal Industrial Services*, 2022.

[2] J. Hayek. Le problème de bin-packing en deux-dimensions, le cas non-orienté : résolution approchée et bornes inférieures. 2006.

[3] Jean C. Creput. Hybridation de métaheuristiques pour la résolution distribuée de problèmes d'optimisation spatialisés. 2008.

[4] Johann Dréo, Alain Petrowski, Patrick Siarry, and Éric D. Taillard. Métaheuristiques pour l'optimisation difficile. 2003.

[5] Yu-Jun Zheng. Water wave optimization: A new nature-inspired metaheuristic. *Computers Operations Research*, 55:1–11, 2015.

[6] Jhon Amaya, Carlos Cotta, and Antonio Fernández-Leiva. Memetic and hybrid evolutionary algorithms. pages 1047–1060, 01 2015.

[7] Malika Bessedik. Chapitre 4 méthodes d'optimisation approchées. page 11, 2023.

[8] Awad MA El-Ashmawi WH. Abdul-Minaam DS, Al-Mutairi WMES. An adaptive fitness-dependent optimizer for the one-dimensional bin packing problem. ieee access.

[9] Seeley Schultz KM, Passino KM. The mechanism of flight guidance in honeybee swarms: subtle guides or streaker bees.

[10] Rashid TA. Muhammed DA, Saeed SAM. Improved fitness-dependent optimizer algorithm. ieee access. 2020;8:19074–88.

[11] S Deb Yang X-S. Cuckoo search via levy flights in 2009 world congress on nature biologically inspired computing (nabic). ieee; 2009. p. 210–14.

[12] Kakandikar G Nandedkar V. Joshi AS, Kulkarni O. Cuckoo search optimization-a review. mat today. 2017 jan;4:7262–9.

[13] Al-Betar M. Shehab M, Khader AT. A survey on applications and variants of the cuckoo search algorithm. appl soft comput. 2017;61:1041–59.

[14] Layeb A. Zendaoui Z. M. a survey on applications and variants of the cuckoo search algorithm.

[15] Lewis A. Mirjalili S. The whale optimization algorithm. adv eng softw. 2016;95:51–67.

[16] Rashid TA. Mohammed HM, Umar SU. A systematic and meta-analysis survey of whale optimization algorithm. comput intell neurosci. 2019;2019:8718571.

[17] Rani A. Jain M, Singh V. A novel nature-inspired algorithm for optimization: Squirrel search algorithm. swarm evol comput. 2019;44:148–75.

[18] Vernes K. Gliding performance of the northern flying squirrel (glaucomys sabrinus) in mature mixed forest of eastern canada. j mammal. 2001 nov;82:1026–33.

[19] Luo W. Zheng T. An improved squirrel search algorithm for optimization. complexity.

[20] Abd Elminaam DS El-Ashmawi WH. A modified squirrel search algorithm based on improved best fit heuristic and operator strategy for bin packing problem. appl soft comput. 2019;82:105565.

[21] Holland JH. Adaptation in natural and artificial systems. 2nd ed., ann arbor, mi: University of michigan press; 1975. p. 1992.

[22] Yang XS. Nature-inspired metaheuristic algorithms. united kingdom: Luniver press; 2010.

[23] Hu R Xiang F-H Qian B, Zhou H-B. Hybrid differential evolution optimization for no-wait flow-shop scheduling with sequence-dependent setup times and release dates. in: Huang de-s, gan y, bevilacqua v, figueroa jc, editors. advanced intelligent computing. berlin, heidelberg: Springer berlin heidelberg; 2012. p. 600–11.