Le Whale Optimization Algorithm est une nouvelle métaheuristique introduite en 2016 par Mirjalili et Lewis basée sur l'intelligence en essaim. Cet algorithme est inspiré d'une stratégie d'alimentation des baleines à bosse connue sous le nom de *L'alimentation au filet à bulles* . Une tactique qui leur permet d'attraper le plus de poissons possibles en un seul coup. Après avoir détecté ses proies, la baleine libère des bulles en nageant dans un mouvement spirale vers la surface pour encercler la proie pour la capturer. Les bulles libérées peuvent prendre 2 formes : une forme de cercles rétrécissants ***[figure 01]*** ou une forme spirale ***[figure 02]***. Dans cet algorithme, La recherche des proies représente l'exploration de l'espace de recherche et La libération des bulles représente l'exploitation.

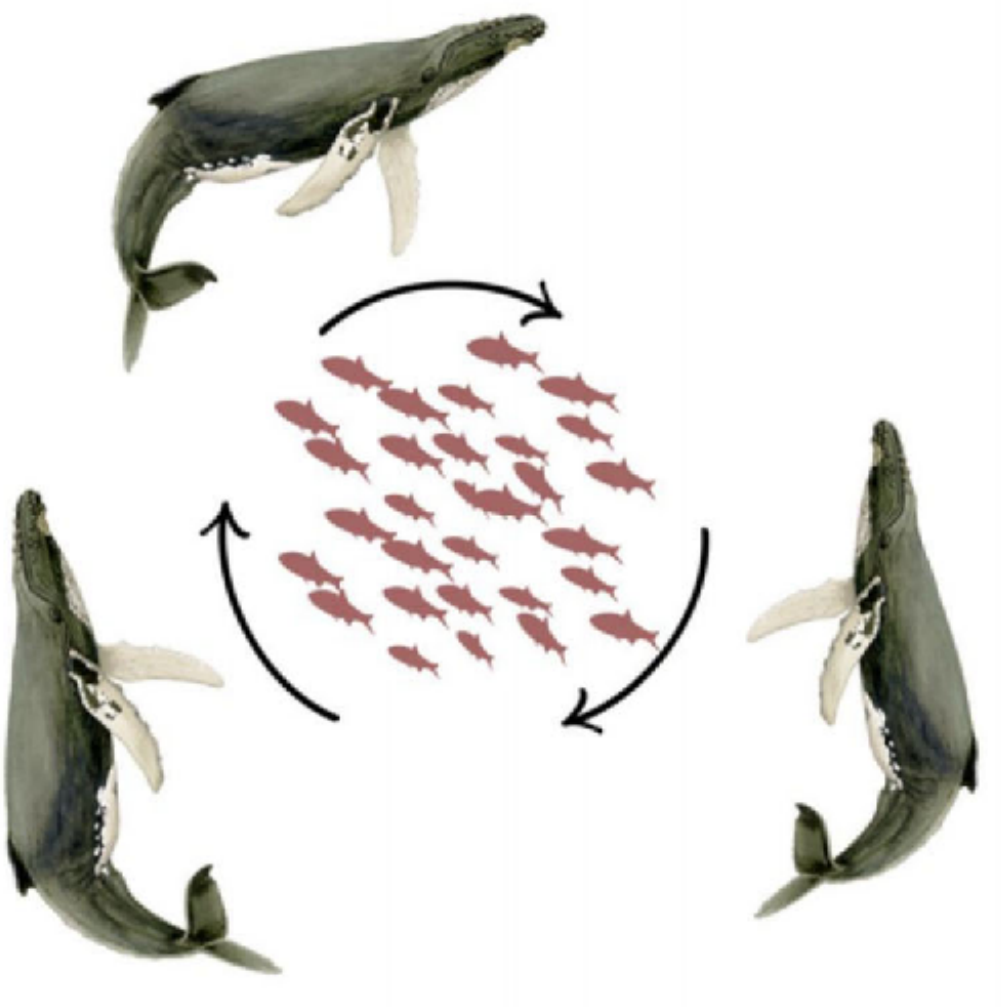

Figure 1: Libération des bulles en cercles rétrecissants

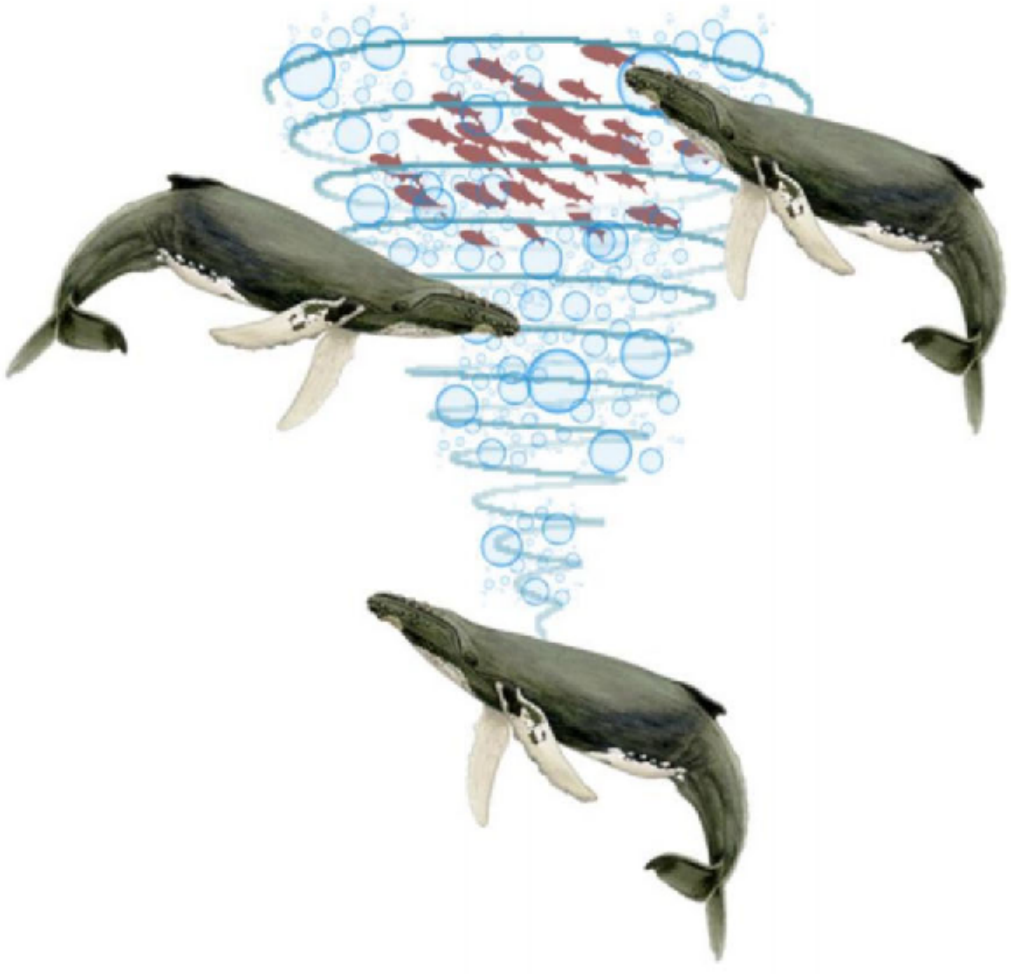

Figure 2: Libération des bulles en spirale

1 Représentation mathématique :

1.1 Encerclement avec bulles en cercles rétrécissants :

Soient:

- $x^*(t)$ la meilleure solution courante.
- t le numéro de l'itération courante.

- $\vec{D}$ indique la distance du ième baleine (ième solution candidate) à la proie (la meilleure solution actuelle).
- $x(t)$ une solution de la population à l'itération t, et $x(t+1)$ le résultat de sa mise à jour.
- $x_r(t)$ une solution choisie aléatoirement de la population courante.
- A et C des coefficients calculés par les formules suivantes:

$$A = 2ar - a$$

$$C = 2r$$

Avec r un nombre aléatoire appartenant à $[0, 1]$.
et a un nombre décrémenté linéairement à chaque itération de 2 à 0, de la façon suivante: $a = a_{init} - a_{init} * i/max\_iter$ avec i numéro de l'itération courante, max_iter le nombre maximal des itération et a_{init} la valeur initiale de a.

Le processus d'encerclement de la proie peut être représenté par les équations suivantes :

$$\vec{D} = |Cx^*(t) - x(t)|$$

Si $|A| < 1$:

$$x(t+1) = x^*(t) - A\vec{D} \tag{1}$$

Sinon:

$$x(t+1) = x_r(t) - A\vec{D} \tag{2}$$

Le comportement de l'encerclement avec bulles en cercle rétrécissant est obtenu en diminuant la valeur de a de a_{init} à 0 au cours des itérations. La variation de A peut être utilisée pour rechercher des proies, c'est-à-dire la phase d'exploration. Par conséquent, A peut être utilisée avec des valeurs aléatoires supérieures à 1 ou inférieures à -1 pour forcer les solutions à s'éloigner de la solutions de référence (la meilleure solution $|A| < 1$ ou une solution aléatoire sinon).

1.2 Encerclement avec bulles sous forme spirale :

Il est modélisé par les équations suivantes :

$$\vec{D} = |x^*(t) - x(t)| \tag{3}$$

$$x(t+1) = \vec{D}e^{bl}\cos(2\pi l) + x^*(t) \tag{4}$$

Où l est un nombre aléatoire appartenant à $[-1, 1]$ Et b une constante qui définit la forme de la spirale

Ces deux types définissent deux mécanismes d'exploration de l'espace de recherche, ce qui permet une meilleure diversification de l'espace de recherche. Dans chaque itération du WOA, un de ces deux mécanismes est choisi avec une probabilité p égale à 50% pour mettre à jour la population des solutions candidates comme suit:

$$x(t+1) = \begin{cases} x^*(t) - A\vec{D} & r < p \\ \vec{D}e^{bl}\cos(2\pi l) + x^*(t) & r \geq p \end{cases}$$

avec $x*(t)$ est la meilleure solution actuelle au temps t, p est égale à $0,5$ et r est un nombre aléatoire entre $[0, 1]$

1.3 Pseudocode:

Algorithm 1 Improved Lévy Whale Optimization Algorithm

Initialiser la population de baleines (l'ensemble initial des solutions candidates): Xi (i = 1, 2 ,..., N)
Évaluer les solutions de la population initiale
X* = la meilleure solution actuelle
while $t < max\_iter$ **do**
  for solution ∈ population **do**
    Mettre à jour a, A, C avec le vol de levy, l et p avec la fonction logistique
    if $r < 0,5$ **then**
      if $|A| < 1$ **then**
        Mettre à jour la solution par Eq.(1)
      else
        Sélectionnez une solution aléatoire X_r
        Mettre à jour la solution par Eq.(2)
      end if
    else
      Mettre à jour la solution par l'Eq.(3)
    end if
  end for
  Vérifier si une solution dans la population dépasse l'espace de recherche et la modifier
  Appliquer la discretisation par LOV
  Évaluer la nouvelle solution
  Mettre à jour X* s'il existe une meilleure solution Sinon lancer la phase de mutation
  Evaluer la nouvelle solution,màj de la meilleure solution
  t = t + 1
end while

1.4 Ingrédients du WOA :

- Population initiale des solutions.
- Fonction d'évaluation qui permet de choisir la meilleure solution x*(t).

- Un mécanisme d'évolution:
  - Encerclement avec bulles en cercle rétrécissant.
  - Encerclement avec bulles sous forme de spirale.
- Critère d'arrêt du WOA: nombre maximal d'itérations.

1.5 Paramètres du WOA :

- La taille de la population des solutions candidates à évaluer dans chaque itération.
- La constante b qui définit la forme de la spirale.
- Le nombre maximal d'itérations: max_iter.
- Le nombre a qui détermine le degré de diversification (exploration).

1.6 Application de l'algorithme au problème du bin packing :

1.6.1 Discrétisation de l'espace de recherche :

Cet algorithme a été proposé pour la résolution de problèmes à espace de recherche continu. Afin de l'adapter à notre problème discret nous allons utiliser une méthode appelée LOV qui permet de passer d'une solution continue à une solution discrète:

Algorithm 2 Discrétisation de l'espace de recherche par LOV

Résultat en sortie: Solution discrète $X = (x_1, x_2, \ldots, x_n)$ obtenue à partir de la solution continue $\tilde{X} = (\tilde{x_1}, \tilde{x_2}, \ldots, \tilde{x_n})$
Ordonner les valeurs du vecteur de $\tilde{X}$ par ordre décroissant
L'indice d'ordre de chaque valeur est stocké dans un vecteur $\theta = \{\theta_i =$ ordre de l'élément $\tilde{x_i}\}$

1.6.2 Représentation d'une solution

Une solution est exprimée par un vecteur $x(t) = (a_1, a_2, \ldots, a_n)$ représentant la distribution des articles dans les boîtes au moment t, avec n le nombre d'articles et a_i est un article d'ordre i. Les articles sont ordonnés selon les boîtes dans lesquelles ils sont rangés, i.e. l'article a_1 est rangé dans la première boîte, s'il y'en a de l'espace dans cette boîte alors l'article a_2 est y rangé, sinon a_2 est rangé dan la deuxième boîte...ainsi de suite. (principe du Next Fit) Une solution $x(t)$ appartient au domaine de recherche tant qu'elle respecte les contraintes du problème: un objet ne peut pas être rangé dans plus d'une boîte, donc pas de doublons dans x(t), l'autre contrainte concernant le respect de la capacité d'une boîte est vérifiée trivialement par définition du vecteur x(t).

1.6.3 La fonction objective :

Afin de pouvoir évaluer les solutions, Nous avons opté pour la fonction suivante proposée par *Hyde et al* [add ref here], au lieu du nombre de boîtes utilisées, parce que avec cette dernière pour plusieurs solutions on peut avoir la même évaluation ce qui peut engendrer la stagnation de l'algorithme.

$$F_{min} = 1 - \frac{\sum_1^n (occup_i/c)^k}{n}$$

Avec:

- n nombre de boites utilisées.
- $ocup_i$ total des poids des objets rangés dans l'ième.
- c capacité des boites.

2 Test et Résultats:

Dans cette partie nous allons tester les performances de la métaheuristique WOA sur les instances du benchmark Scholl. Ensuite nous allons effectuer une comparaison des résultats obtenus avec les résultats optimaux. Nous utiliserons pour chacune des trois classes du benchmark Scholl deux configurations de paramètres; la première configuration est obtenue par tâtonnement après plusieurs essais manuels, la deuxième par calibrage automatiques des

paramètres utilisant le package *irace* qui implémente l'algorithme I/F-Race (voir chapitre XX). Pour pouvoir étudier les performances de WOA, notre étude se comportera 2 axes:

- Temps d'exécution.
- Qualité de la solution (ratio et comparaisons avec les solutions optimales)

Remarque: Les algorithmes ont été développés en utilisant le langage de programmation Python, et exécutés sur un **HP probook [Intel Core i7-6500U CPU @2.50GHz, 8Go RAM]** en utilisant l'IDE Intellij pycharm Le générateur d'instances utilise la fonction random() de la bibliothèque random de Python, cette fonction utilise le Mersenne Twister qui est un générateur de nombres pseudo-aléatoires, réputé pour sa qualité.

2.1 Rappel des paramètres de WOA:

- La taille de la population: nb_whales.
- Le nombre maximal d'itérations: max_iter.
- La constante de l'encerclement spirale b.
- La constante a qui détermine le degré d'exploration de l'espace de recherche.

2.2 Temps d'exécution:

2.3 Qualité de solution:

2.3.1 Ratio:

2.3.2 Comparaison avec la solution optimale:

2.4 Analyses des résultats:

- WOA permet d'obtenir un ratio inférieur à 1.4 pour les benchmark Scholl, et un ratio de 1 pour les instances de la classe 2 de taille N1.

- Parmis les trois classes WOA est le plus performant dans le cas des instances difficiles (classe 3: C3) et la classe 2.
- Le calibrage automatique permet généralement d'améliorer la qualité de la solution très légèrement sauf pour le cas des instances de la classe C2 de tailles N1 et N4 ou la configuration manuelle donne de meilleurs résultats.
- Le calibrage automatique permet de réduire le temps d'exécution significativement surtout pour les instances de grande taille (N4) et les instances de la classe 3 (instances difficiles).
- Avec une configuration optimale des paramètres le temps de résolution des instances les plus difficiles et/ou volumineuses ne dépasse pas 8s.