الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université de 8 Mai 1945 – Guelma -
Faculté des Mathématiques, d'Informatique et des Sciences de la matière
Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Informatique Académique

Thème :

Résolution du problème de chargement des containers (Bin Packing) avec la métaheuristique Chemical Reaction Optimization

---

Encadré Par :

Mr. Zendaoui Zakaria

Présenté par :

Ladassi Amina

Boulakaab Amina

Juin 2017

Remerciements

Après Dieu le tout puissant, nous tenons à exprimer en tout premier lieu notre profonde gratitude à notre aimable encadreur, ***Mr. Zendaoui Zakaria*** pour nous avoir encadré, orienté et guidé afin de réaliser ce travail. Nous le remercie pour ses encouragements continuels qui ne cessaient de nous remonter le moral pendant les moments difficiles et pour l'énorme soutien scientifique et moral qu'il a su nous accorder pendant la période de ce travail.

Nous adressons aussi nos remerciements à nos chers parents pour leur encouragement et le soutien affectif et matériel qu'ils nous ont apporté tout au long de notre existence.

Nous remercie sincèrement les membres de jury d'avoir accepté d'évaluer notre travail.

Enfin, nos remerciements à toute les personnes qui ont contribué de près ou de loin à la réalisation de notre mémoire.

Amina

Résumé

Le problème de chargement de containers, bien connu sous le problème de bin packing, ce problème consiste à déterminer le nombre minimum des containers (bins) nécessaires pour ranger un ensemble d'objets de tailles connues afin qu'ils ne dépassent pas la capacité de chaque bin. Le problème de bin packing peut être appliqué à un grand nombre de secteurs industriels ou informatiques. Ce problème est connu pour être NP_Difficile.

De ce fait plusieurs méthodes de résolution exactes ou approchées tentent de le résoudre. Ces dernières se basent essentiellement sur les métaheuristiques et plus particulièrement sur les algorithmes évolutionnaires qui ont été révélés les plus efficaces pour résoudre les problèmes d'optimisation complexes à savoir le problème du bin packing. Parmi les métaheuristiques les plus récentes, nous nous choisissons l'algorithme d'Optimisation par la réaction chimique « Chemical Reaction Optimization CRO », cette métaheuristique est basée sur une population. Elle est inspirée par le phénomène d'interactions entre les molécules dans un processus de réaction chimique permettant la transformation des substances instables à d'autres plus stables. L'approche proposée nous permette de traiter des instances de grandes tailles en un temps raisonnable, et en obtenant des résultats très proches aux solutions optimales et dans d'autres cas, des solutions optimales.

Mots-clés: Bin Packing, Métaheuristique, Optimisation combinatoire, CRO.

Sommaire

Liste des figures

Liste des tableaux

Liste des Algorithmes

Liste des abréviations et acronymes

BPP: Problème de Bin Packing (Bin Packing Problem).

FF: First Fit.

FFD: First Fit Decreasing.

NF: Next Fit.

NFD: Next Fit Decreasing.

BF: Best Fit.

BFD: Best Fit Decreasing.

WF: Worst Fit.

AG: Algorithme Génétique.

CRO: Optimisation par la Réaction Chimique (Chemical Reaction Optimization).

PE : Energie Potentielle.

KE : Energie cinétique.

Introduction générale

L'optimisation combinatoire est une voie d'études qui fait l'objet de recherches intensives depuis plusieurs années. C'est un domaine qui couvre un large éventail de problèmes qui surgissent dans de nombreux domaines applicatifs comme l'industrie, le transport, la télécommunication, le traitement d'image, etc. Ces problèmes sont exprimés sous la forme générale d'un problème d'optimisation. Ce dernier est une classe qui regroupe les problèmes qui sont caractérisés par un espace de recherche fini et dénombrable, mais d'une très grande taille. La résolution d'un tel problème consiste à chercher la meilleure solution optimisant un critère ou encore une fonction donnée. L'objectif est de trouver une solution dite optimale ou parfois de bonne qualité, minimisant ou maximisant une ou plusieurs fonctions d'évaluation dites fonctions objectif.

Le problème de chargement des containers est l'un des problèmes de recherche les plus connus dans le domaine de l'optimisation combinatoire et son application est plus étendue. Le but principal des problèmes de ce genre est de l'utilisation des taux de l'espace et de réduire le coût. Dans tous les cas, un problème de chargement des containers peut être modélisé sous la forme d'un problème de ***Bin Packing***. Ce problème est une sorte de sujet de recherche scientifique, et son application est très répandue. Dans ce problème, on dispose d'une liste d'objets, et on cherche à minimiser le nombre de containers (bins) nécessaires pour placer l'ensemble d'objets, de telle manière que chacun des bins possède une capacité maximale. Cette capacité ne doit pas être dépassée. Pour ce problème, ils existent trois variantes qui sont : la variante unidimensionnel, la bidimensionnelle et la tridimensionnelle, la variante la plus basique se définie en une seule dimension, où on a une seule dimension à respecter tel que le poids.

Le problème du bin packing est classé parmi les problèmes NP_Difficile, de ce fait plusieurs méthodes de résolution exactes ou approchées tentent de le résoudre. Ces dernières se basent essentiellement sur les métaheuristiques, parmi les métaheuristiques les plus récentes, l'algorithme d'optimisation par la réaction chimique « ***CRO*** ». Elle est inspirée par le phénomène d'interactions entre les molécules dans un processus de réaction chimique permettant la transformation des substances instables à d'autres plus stables. En outre, l'algorithme CRO a démontré son efficacité dans plusieurs problèmes d'optimisation combinatoire.

Chapitre 01

Problème du Bin Packing et méthodes de résolution

« Ce qui ne nous tue pas nous rend plus fort »

Friedrich Nietzsche

Introduction :

Dans le domaine d'optimisation combinatoire et de recherche opérationnelle, le problème de Bin packing (BPP) est l'un des problèmes difficiles rencontrés dans divers applications réelles. Il est défini depuis plus de 50 ans et il a été introduit par D. Johnson.

Les problèmes de bin packing ont fait l'objet de nombreuses études dans la littérature, les raisons à cela sont multiples. Tout d'abord, il existe de nombreuses variantes à ces problèmes, qui peuvent générer des modélisations et des méthodes de résolution différentes. Enfin, Les enjeux industriels et financiers ont été les moteurs initiaux de cette recherche.

Dans ce chapitre nous présentons le problème de « Bin Packing ». En premier temps nous donnons une définition générale à ce problème, et après nous déterminons ses variantes. Enfin, nous décrivons les différentes méthodes de résolution du problème associé (exactes et approchées).

1. *Qu'est ce qu'un problème de bin packing ?*

On considère un ensemble des bins (boites) identiques et un ensemble d'objets dont on connaît le poids. Sachant que les bins ne peuvent supporter qu'un poids maximum, combien faudra-t-il au minimum de bins pour y ranger l'ensemble des objets considérés ? Ce problème d'optimisation, connu sous le nom de « *bin packing problem* ». Le problème du bin packing, ou problème du rangement/placement, est une problématique majeure dans le domaine de la recherche, puisqu'on trouve des applications concrètes dans de très nombreux domaines, telles que les problèmes d'affectation avec contraintes ainsi que certains problèmes d'ordonnancement et d'allocation de ressources. Le bin packing est un problème algorithmique, il s'agit de ranger des objets de taille connu avec un nombre minimum de bins [1].

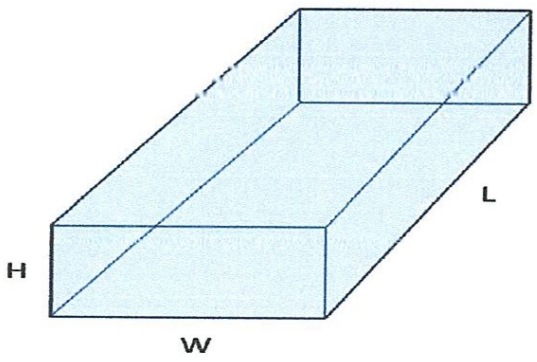

Figure 1.1 : *un bin.*

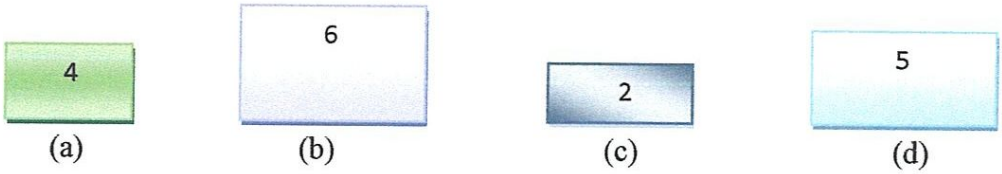

Figure1.2 : *Objets de différentes tailles.*

2. ***Objectif :***

L'objectif de ce problème est mettre un ensemble d'objets fixés dans des bins de même capacité ou des capacités différentes en utilisant le moins possible de bin.

3. ***Les variantes de problème de bin packing :***

Il existe un grand nombre de variantes pour le problème de bin packing. Cependant chaque problème réel présente ses propres spécificités telles que [2]:

➢ *Les caractéristiques propres aux objets :*

- objets de formes homogènes ou non homogènes.
- objets de tailles uniformes ou différentes.
- objets déformables ou non déformables.

➢ *Les spécificités propres au problème :*

- le nombre de dimensions du problème.
- disposer d'un seul bin (problème de décision ou problème de maximisation).
- minimiser le nombre de bins à utiliser.
- minimiser la surface ou le volume global des objets à placer.

➢ *Les contraintes propres au problème :*
- l'équilibre entre les objets.
- l'ordre dans lequel les objets doivent être retirés du bin.
- l'orientation d'un objet.
- le poids (par exemple, le poids total dans un bin ne peut pas excéder une limite donnée).
- le placement : certains objets très lourds doivent être placés en bas, d'autres fragiles doivent être placés en haut.

Selon la dimension du problème, trois variantes de Bin packing sont connus dans la littérature. Le problème classique se définit en une dimension 1D, mais il existe de nombreuses variantes en deux dimensions 2D et en trois dimensions 3D.

3.1. Bin Packing uni- dimensionnel (1BPP) :

a) Définition :

Une instance I de Bin packing en une dimension peut être définit comme suite : Etant donnés n objets de poids $w_1, ..., w_n$, et des bins (boîtes) de capacité C, un 1BP s'agit de ranger tous les objets dans un nombre minimal de bins, la somme des poids des objets d'un même bin ne pouvant pas dépasser C [3].

➢ *Exemple* :

Soient un ensemble d'objet n ($n = 6$) de poids wi, et des bins de capacité maximale de 10 ($C = 10$), nous avons par exemple les données suivantes :

Objets	1	2	3	4	5	6
w_i	7	6	3	8	2	4

La solution optimale est :

Bin 1 : w_1 , w_3.

Bin 2 : w_2 , w_6.

Bin 3 : w_4 , w_5.

b) modèle mathématique pour 1BPP :

Dans le problème classique, les données sont :

- un nombre infini de bins de taille C ;
- une liste $1,2, ..., n$ d'objets i de taille c_i.

On cherche à trouver le rangement valide pour tous ces objets qui minimise le nombre de bins utilisées. Pour qu'un rangement soit valide, la somme des tailles des objets affectés à un bin doit être inférieure ou égale à C.

Pour décrire une solution, on peut utiliser un codage binaire pour indiquer dans quel bin j chaque objet i est rangé.

- La variable x_{ij} vaudra 1 si l'objet i est rangé dans le bin j, et 0 sinon.
- La variable binaire y_i est égale à 1 si le bin j est utilisée, 0 sinon.

On cherche donc à minimiser le nombre de bins utilisées :

$$min \sum_{j=1}^{n} yj$$

Sous les contraintes suivantes :

$$\sum_{i=1}^{n} c_i x_{ij} \leq C_{y_j}, \qquad j = 1, \ldots, n \ldots(1)$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, \ldots, n \ldots(2)$$

$$x_{ij} \in \{0,1\}, \qquad i = 1, \ldots, n ; j = 1, \ldots, n$$

$$y_i \in \{0,1\}, \qquad j = 1, \ldots, n$$

La première inégalité (1) signifie qu'on ne peut pas dépasser la taille d'un bin pour un rangement. À noter que la partie droite de l'inégalité oblige y_i à prendre la valeur **1** dès qu'un objet est rangé dans le bin ***j***. La deuxième inégalité (2) impose à tous les objets d'être rangés dans un bin et un seul. Toute solution pour laquelle la famille d'équations précédente est vérifiée est dite *réalisable* [4].

3.2. Bin packing Bi-dimensionnel (2BPP) :

a) Définition :

Le *2BPP* s'agit de minimiser le nombre de grands rectangles (ou grandes boîtes, ou bins) identiques pour ranger une liste d'objets rectangulaires. Les dimensions du bin sont notées W et H et les objets ont une orientation fixe (i.e., pas de rotation possible). Chaque objet i a une largeur $w_i \leq W$ et une hauteur $h_i \leq H$ (w_i et h_i entiers) [5].

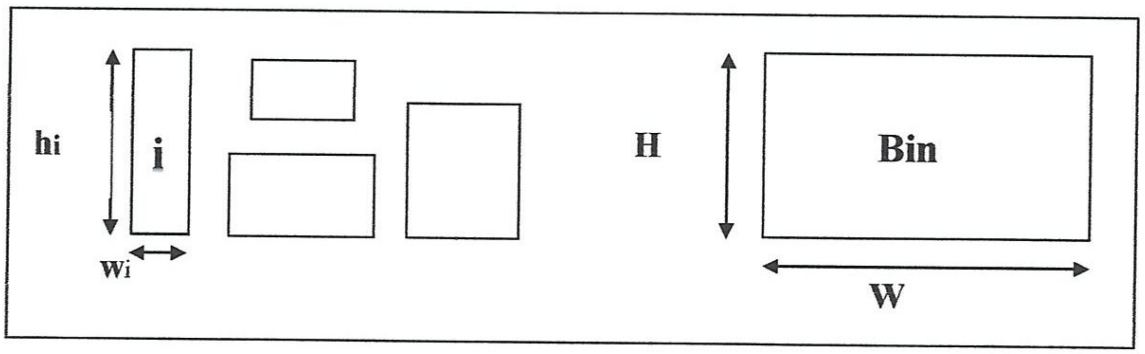

Figure 1.3 : *Description d'un problème de bin-packing en 2 dimensions (2D).*

➢ *Exemple :*

Un objet de largeur *wi* et de hauteur *hi* est noté (*wi, hi*). Le bin est de taille (10,10), et six objets de tailles {(4,4), (4,4), (6,4), (8,4), (8,4), (4,8)}.

Article	*1*	*2*	*3*	*4*	*5*	*6*
(largeur, hauteur)	*(4,4)*	*(4,4)*	*(6,4)*	*(8,4)*	*(8,4)*	*(4,8)*

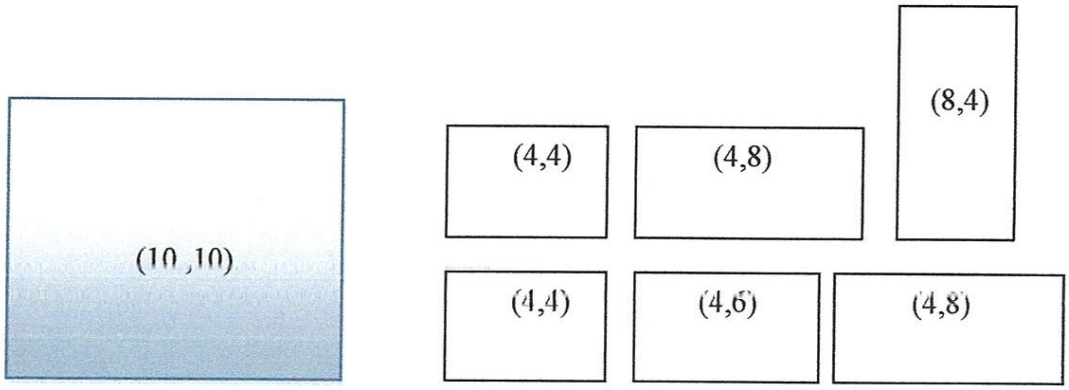

Figure 1.4 **:** *Une instance d'un Bin-packing en deux dimensions (2D).*

La solution optimale est :

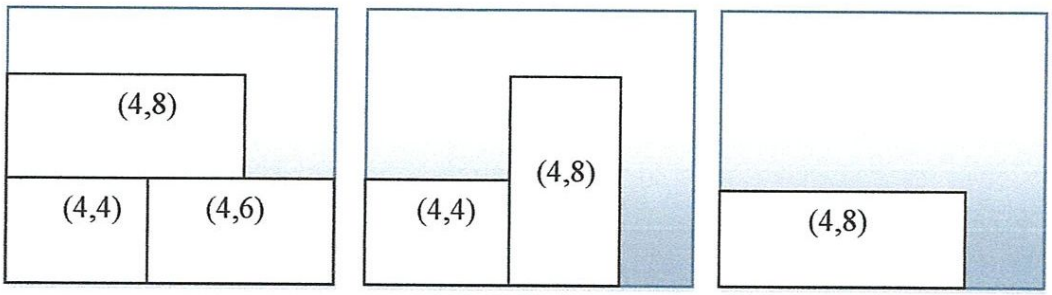

Figure 1.5: *Solution optimale de l'exemple.*

3.3. ***Bin packing tri-dimensionnel (3BPP)*** :

Le bin packing tridimensionnel est une généralisation du problème bidimensionnel avec un nombre donné d'objets 3D ainsi que des bins tridimensionnels. L'objectif est de déterminer le nombre minimum de bins pouvant contenir l'ensemble de ces objets identiques en forme (3D). Sachant que Les dimensions des bins sont notées *W*, H et L

et chaque objet i a 3 dimensions tel que $wi \leq W$, $hi \leq H$ *et* $li \leq L$ (wi et hi *et* li entiers) [6].

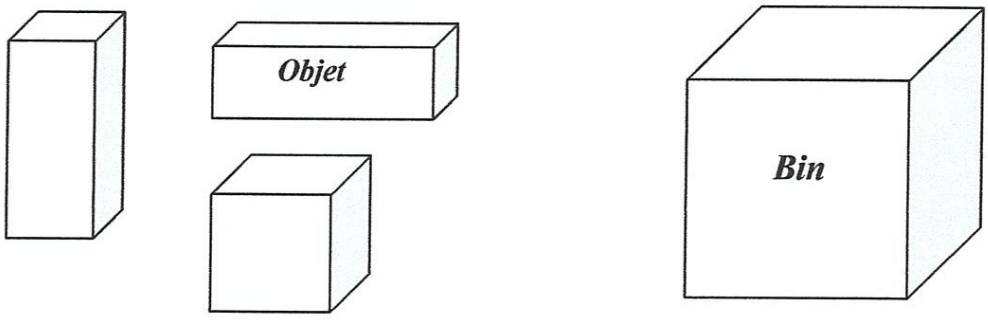

Figure 1.6 : *Une instance d'un problème de bin-packing en trois dimensions.*

Dans un problème de bin packing en 3 dimensions en tenant compte les contraintes pratiques suivantes:

- Contrainte d'équilibre.
- Rotation des objets.
- Façon dont arrivent les objets.
- Contraintes de voisinage.

4. Domaines d'application des problèmes de bin packing :

Le problème de Bin packing peut être appliqué à un grand nombre de domaine, il a fait l'objet depuis quelque années d'une attention croissante pour deux raisons : outre son intérêt théorique, il a de nombreuse applications dans le domaine industriel, de la logistique, de l'informatique et même de l'édition. On retrouve le problème notamment dans l'industrie du tissu, du métal, mais aussi dans le cadre de la découpe de bois, de verre ou même de placement des spots publicitaires dans les journaux…etc.

5. Des exemples pratiques des problèmes de bin packing :

Dans la vie courante (ou presque), on est parfois amené à résoudre un problème de bin packing sans le savoir [7].

• Je souhaite sauvegarder le contenu de mon disque dur sur des CD-Rom de 640Mo. Chaque fichier a une taille, et je ne souhaite pas découper les fichiers. Comme il ne me reste pas beaucoup de CD, je cherche le minimum de disques que je peux utiliser. Notez bien que dans cette version du problème, je ne cherche pas à regrouper spécialement des fichiers par genre. J'ai un problème de bin packing à résoudre.

• J'ai acheté des grandes planches de bois. Je voudrais construire une armoire avec ce bois. Vu le coût du bois, je voudrais acheter le moins de planches possibles pour découper les rectangles nécessaires pour fabriquer mon armoire. C'est un problème de bin packing en deux dimensions.

• Je vais déménager, et je me demande si le camion que j'ai loué va être suffisant pour ranger tous mes cartons. Cette fois, c'est un problème de bin packing en 3D.

6. Méthodes de résolutions de problème du Bin Packing :

Le problème de bin packing a été largement étudié dans la communauté de recherche opérationnelle. Comme dans tous les problèmes combinatoires, des méthodes exactes et des méthodes approchées ont été proposées pour la résolution du problème de bin-packing. Les méthodes exactes permettent d'obtenir la solution optimale à chaque fois, mais si le problème est compliqué le temps de calcul peut être long. Ce pendant les méthodes approchées, encore appelées heuristiques, permettent d'obtenir rapidement une solution approchée, donc pas nécessairement optimale. Nous nous concentrons uniquement sur le problème de bin packing en une dimension, on va alors étudier que les méthodes de résolution de ce problème. Et donc nous allons détailler un exemple d'algorithme de résolution de chaque catégorie.

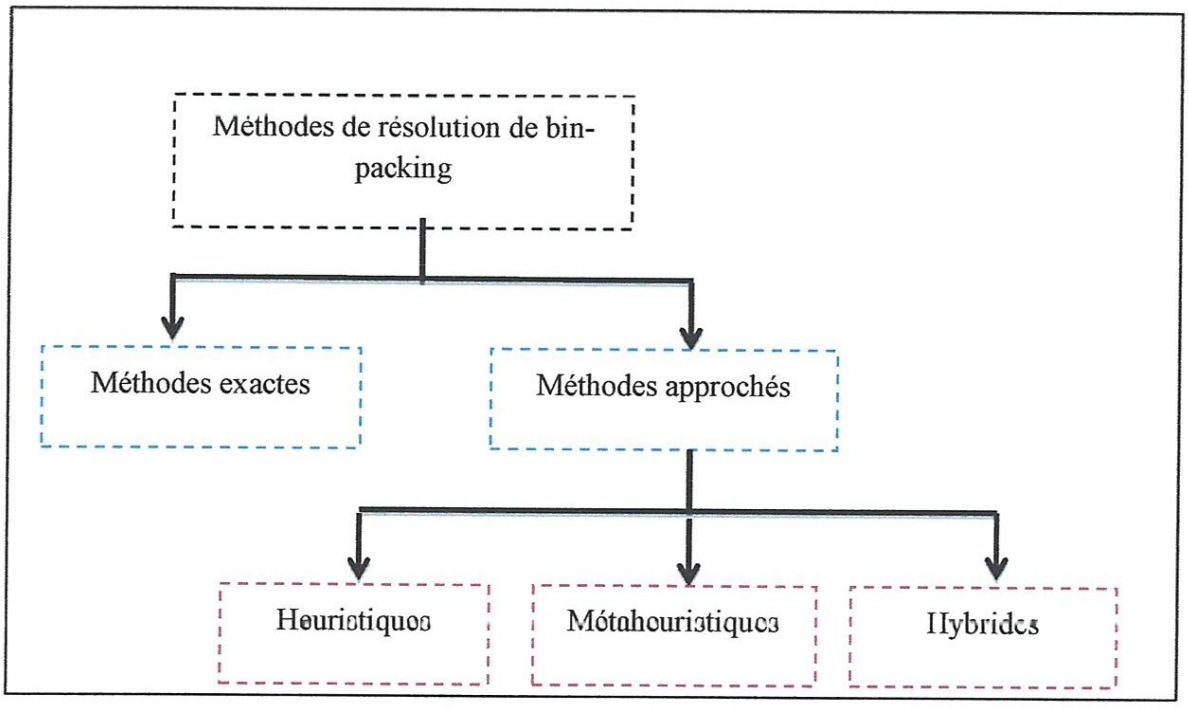

Figure 1.7 : *Classification des méthodes de résolution du bin-packing.*

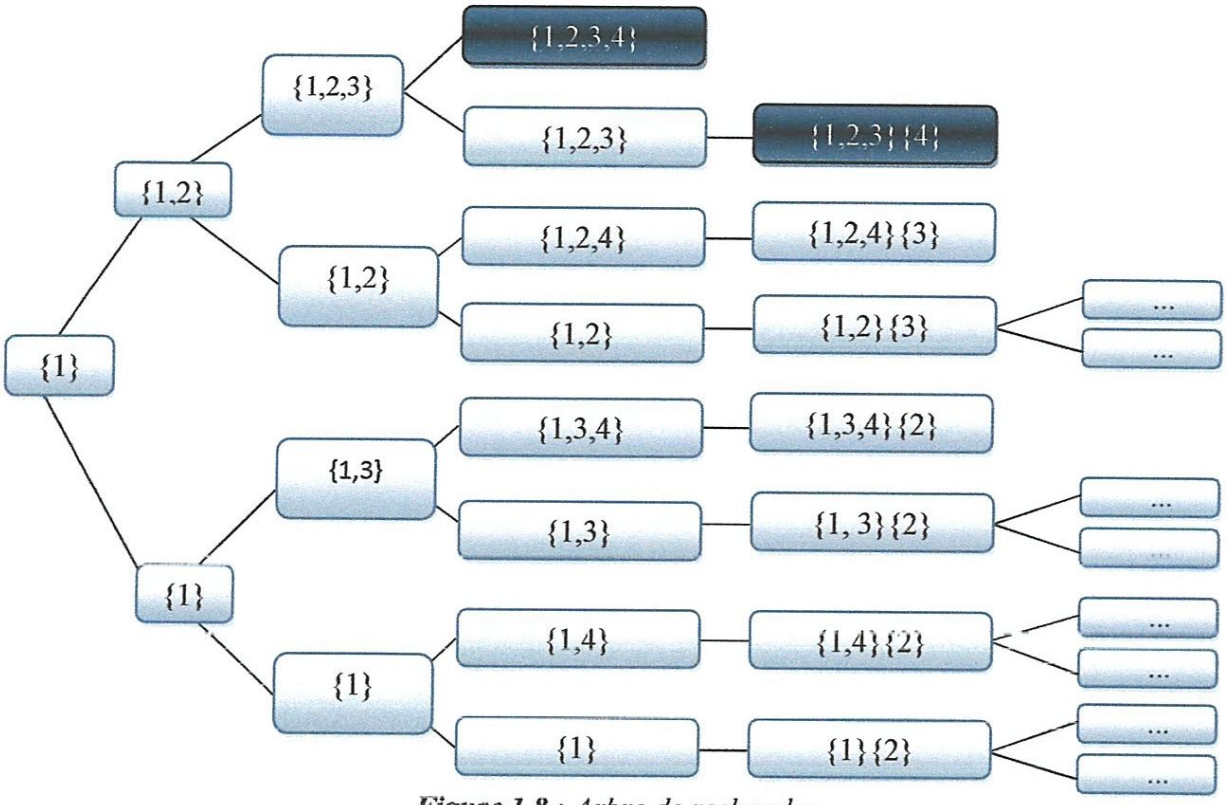

***Figure 1.8** : Arbre de recherche.*

Au départ, on cherche à remplir le premier bin. Tout d'abord on lui affecte l'objet 1 : bien que cet objet soit contenu dans un bin, on décide en quelque sorte de nommer 1 le bin qui le contient. La racine de l'arbre, dont la profondeur est par définition 0, contiendra donc l'ensemble {1} qui représente le contenu du 1er bin. Puis pour un nœud de profondeur i on construit deux fils : celui du haut où l'on ajoute l'objet $i + 1$ dans le bin et celui du bas où le bin reste tel quel. Lorsque on a considéré tous les objets pour un bin donné, on poursuit la construction de l'arbre en passant au bin suivant et en lui affectant initialement l'objet non sélectionné dont l'indice est le plus petit. On poursuit ainsi la construction de l'arbre jusqu'à avoir rangé tous les objets. Dans l'arbre de recherche achevé, chaque feuille représente une solution potentielle mais pas forcément réalisable.

Dans le schéma, les feuilles en noire représentent les propositions irréalisables, car elles sont supérieures au poids maximal à ne pas dépasser.

Pour déterminer la solution, il suffit de calculer le nombre de bins pour chaque feuille acceptable et de prendre la solution ayant la plus petite valeur.

La procédure B&B permette d'élaguer encore plus cet arbre en utilisant des bornes inférieures et supérieures de la fonction objectif [10] :

- Une borne inférieure est une valeur minimale de la fonction objectif. Autrement dit, c'est une valeur qui est nécessairement inférieure à la valeur de la meilleure solution possible. Dans notre cas, ça peut être la somme des poids des objets divisés par la capacité des bins.
- Une borne supérieure est une valeur maximale de la fonction objectif. Autrement dit, nous savons que la meilleure solution a nécessairement une valeur plus petite. Dans notre cas, nous savons que la valeur de la meilleure solution est nécessairement inférieure au nombre d'objets (comme si on les mettait chacun dans un bin) ou mieux la meilleure solution doit être strictement inférieure à la valeur trouvée par l'algorithme BFD car nous cherchons à améliorer le meilleur résultat connu.

6.2. *Les méthodes approchées :*

Les méthodes approchées ont été proposées pour le problème de bin-packing dans le but de trouver une solution avec un bon compromis entre la qualité de la solution et le temps de calcul. Ils peuvent être classés en deux groupes : les algorithmes heuristiques et les métaheuristiques.

6.2.1. *Les heuristiques :*

Par définition, une heuristique est un moyen de guider les choix que doit faire un algorithme pour réduire sa complexité. Une heuristique est spécifique à un problème et ne peut pas être généralisée. Les heuristiques ne garantissent pas l'obtention d'une solution optimale mais fournissent en général, en un temps raisonnable et à un coût acceptable, une solution dont les performances sont assez bonnes [3].

Il existe plusieurs heuristiques pour le problème de Bin packing, dans cette section, nous avons présenté quelques heuristiques les plus populaires dans ce domaine.

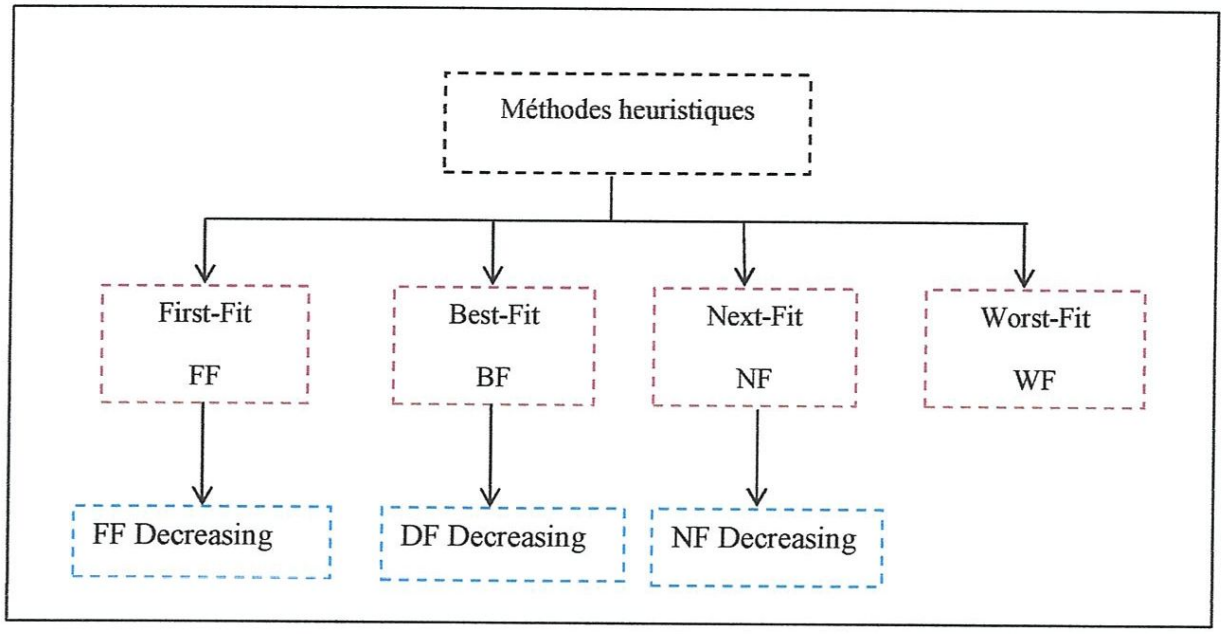

Figure 1.9 *: Méthodes heuristiques pour la résolution du bin-packing.*

a) ***Stratégie First Fit (FF)***

Initialement un seul bin est considéré, et les objets sont traités selon un ordre donné. Quand il n'y a plus de la place dans le premier bin pour ranger l'objet en cours, un deuxième bin est alors ouvert mais sans fermer le premier. Dans une étape intermédiaire où on dispose de k bins ouverts numérotés de 1 à k selon l'ordre de leur première utilisation, un objet a_i en cours est rangé dans le bin du plus faible numéro qui peut le contenir. Dans le cas où aucun bin ne peut contenir l'objet a_i un nouveau bin k + 1 est alors utilisé sans fermer les autres. L'ordre selon lequel on traite les objets est crucial pour la qualité de la solution [7].

✓ *Algorithme First Fit* :

---

1 . Pour tous les objets i = 1, 2, . . ., n faire

2: Pour tous les bins j = 1, 2, . . ., faire

3 : Si l'objet i S'inscrit dans le bin j, alors

4 : Mettre l'objet i dans le bin j.

5 : Fin Si.

6 : Fin Pour.

7 : Si l'objet i, ne pas entrer dans un bin disponible.

8 : Créer un nouveau bin, et mettre l'objet i.

9 : Fin Si.

10 : Fin Pour.

***Algorithme 1.1** : First Fit.*

➢ *First Fit Decreasing (FFD) :*

Consiste à trier les objets par ordre décroissant de poids et appliquer la stratégie FF pour les ranger.

b) *Stratégie Next Fit (NF) :*

Dans cette stratégie on ne considère qu'un bin ouvert à la fois. Les objets sont traités selon un ordre donné. Les objets sont rangés successivement dans le bin ouvert tant qu'il y a de la place pour l'objet a_i en cours, sinon le bin en cours est fermé et un nouveau bin est ouvert [7].

✓ *Algorithme Next Fit :*

---

1 : Pour tous les objets i = 1, 2, . . ., n faire

2 : Si l'objet i S'inscrit dans le bin actuel, alors

3 : Mettre l'objet i dans le bin actuel.

4 : Sinon

5 : Créer un nouveau bin, et mettre l'objet i.

6 : Fin Si.

7 : Fin Pour.

***Algorithme 1.2:** Next Fit.*

➢ *Next Fit Decreasing (NFD) :*

Consiste à trier les objets par ordre décroissant de poids et appliquer la stratégie NF pour les ranger.

c) *Stratégie Best Fit (BF) :*

Comme dans la stratégie FF, les algorithmes BF laissent les bins toujours ouverts. Cependant, le choix du bin dans lequel l'objet a_i en cours va être placé dépend des valeurs des gaps (hauteurs non utilisées) présentes dans les bins. Ainsi, a_i sera placé dans le bin qui présente le moindre gap parmi les bins qui peuvent le contenir [11].

✓ *Algorithme Best Fit* :

---

1 : Pour tous les objets i = 1, 2, . . ., n faire
2: Pour tous les bins j = 1, 2, . . ., faire
3 : Si l'objet i S'inscrit dans le bin j, alors
4 : Calculer la capacité restante après l'ajout de l'objet.
5 : Fin Si.
6 : Fin Pour.
7 : mettre objet i dans bin j, où j est le bin avec la capacité minimale restante après avoir ajouté l'objet (c'est-à-dire l'objet "correspond au mieux")
8 : Si aucun tel bin n'existe, ouvrez un nouveau et ajoutez l'objet
10 . Fin Pour

Algorithme 1.3: *Best Fit.*

➢ *Best Fit Decreasing (BFD) :*

On parle de BFD quand il s'agit de trier les objets à ranger dans l'ordre décroissant de poids avant de les ranger suivant une stratégie BF.

d) Stratégie Worst Fit (WF)

La stratégie Worst Fit (WF), est une variante du BF, suivant laquelle l'objet en cours est placé dans le bin qui peut le contenir et qui présente le plus grand gap. Cette stratégie peut produire de meilleures solutions que BF sur certaines instances [11].

- Dans une stratégie dite *Any Fit (AF),* l'objet en cours est affecté à un bin choisi aléatoirement parmi l'ensemble de bins ouverts qui peuvent le contenir.

➢ *Exemple :*

Soit l'ensemble des objets $S = \{4, 8, 2, 6, 3, 2\}$ et bins de taille 10, Insérer les objets dans autant de bins que possible en utilisant les heuristiques simple NF,FF,BF.

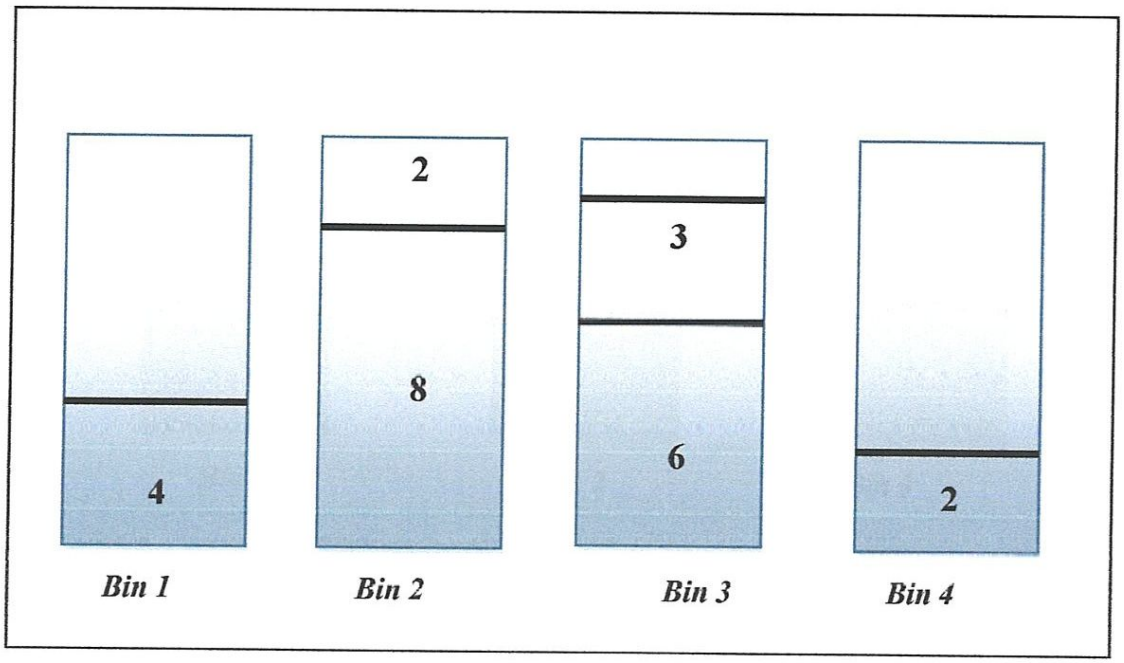

***Figure 1.10** : Exemple de stratégie Next Fit.*

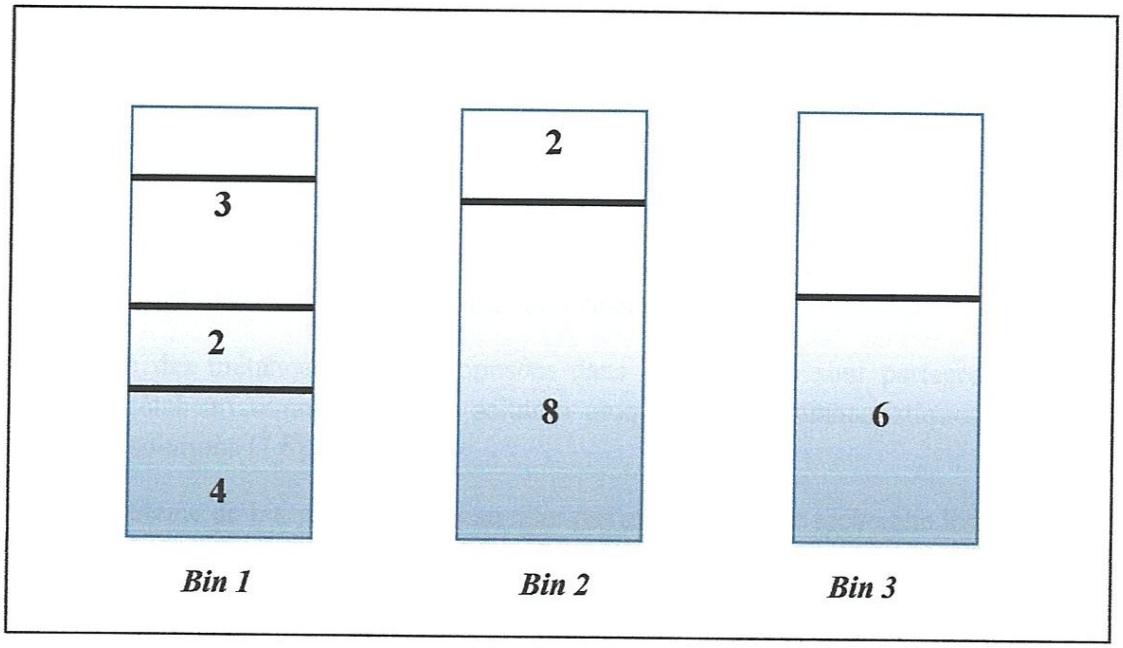

***Figure 1.11** : Exemple de stratégie First Fit.*

➢ *Exemple :*

Cette représentation indique que, l'objet 1 est placé dans le bin 2, l'objet 2 est placé dans le bin 3, et ainsi de suite.

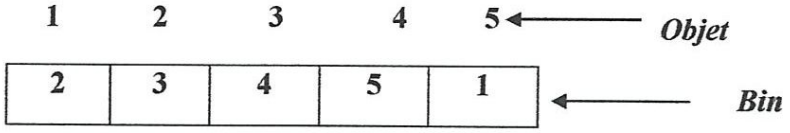

Figure 1.14 : *Représentation à base de bin.*

A.2. Représentation en fonction d'objets

Dans ce cas, la liste est divisée selon le nombre d'objets placés dans les bins.

➢ *Exemple :*

Pour les objets 1, 2, 3, 4, 5,6. Une partition de la liste pourrait être 1 2 | 3 4 5 | 6.

Cela indique que les objets 1 et 2 sont placés dans le bin 1, les objets 3, 4, et 5 sont placés dans le bin 2, et l'objet 6 est placé dans le bin 3.

A.3. Représentation en fonction des groupes

Cette représentation est proposée par Falkenauer, elle se compose de deux parties : La première partie est similaire à la représentation en fonction de bin, tandis que la deuxième partie fournit un codage des bins utilisés.

➢ *Exemple :*

La solution présentée dans la figure 1.15 signifie que, l'objet 1 est placé dans le bin 4, l'objet 2 dans le bin 1, les objets 3 et 5 dans le bin 2, l'objet 4 dans le bin 5, et l'objet 6 dans le bin 3.

4	1	2	5	2	3

Figure 1.15 : *Représentation à base de bin d'une solution.*

La représentation à base des groupes de cette solution, sert à utiliser des lettres pour représenter les bins.

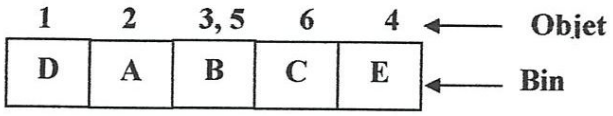

Figure 1.16 : *Représentation en fonction des Groupes d'une solution.*

Où D = {1}, A = {2}, B = {3,5}, C = {6}, et E = {4}.

L'avantage de ce système est que les cases de la liste représentent à la fois des objets et des groupes. Cependant son inconvénient, est que les listes ont une longueur variable.

B. les algorithmes génétiques

1. Définition :

L'algorithme génétique représente une célèbre métaheuristique évolutionnaire. Il a été proposé par Jhon Holland en 1975. L'algorithme génétique s'inspire des mécanismes biologiques tels que les lois de Mendel et la théorie de l'évolution proposée par Charles Darwin en 1859. Son processus de recherche de solutions à un problème donné imite celui des êtres vivants dans leur évolution. Il utilise le même vocabulaire que celui de la biologie et la génétique classique, on parle donc de: gène, chromosome, individu, population et génération [13].

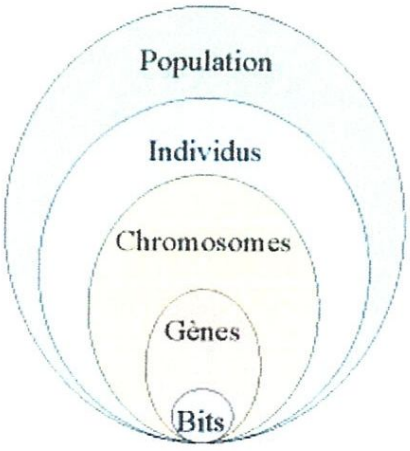

Figure 1.17*: les cinq niveaux d'organisation d'un algorithme génétique.*

✓ *Les éléments des algorithmes génétiques :*

Puisque les algorithmes génétiques sont inspirés de la génétique classique, il convient de présenter leurs vocabulaires informatiques corresponds aux vocabulaires biologiques, des algorithmes génétiques :

Gène **:** Un gène est une suite de bases azotées qui contient le code d'une protéine donnée. On appellera gène la suite de symboles qui codent la valeur d'une variable. Dans le cas général, un gène correspond à un seul symbole (0 ou 1 dans le cas binaire). Une mutation changera donc systématiquement l'expression du gène muté.

Chromosome **:** Un chromosome est constitué d'une séquence finie de gènes qui peuvent prendre des valeurs appelées allèles qui sont prises dans un alphabet qui doit être judicieusement choisi pour convenir du problème étudié.

Individu **:** On appellera individu une des solutions potentielles. Dans la plupart des cas un individu sera représenté par un seul chromosome.

Population **:** On appellera population l'ensemble des solutions potentielles qu'utilise l'AG.

Génération : On appellera génération l'ensemble des opérations qui permettent de passer d'une population P_i à une population P_j. Ces opérations seront généralement : sélection des individus de la population courante, application des opérateurs génétiques, évaluation des individus de la nouvelle population.

Fonction d'évaluation (Fitness) : Elle quantifie la qualité de chaque chromosome pour la reproduction. Les chromosomes ayant une bonne qualité ont plus de chance pour la reproduction et donc plus de chance pour que la population suivante hérite leur matériel génétique.

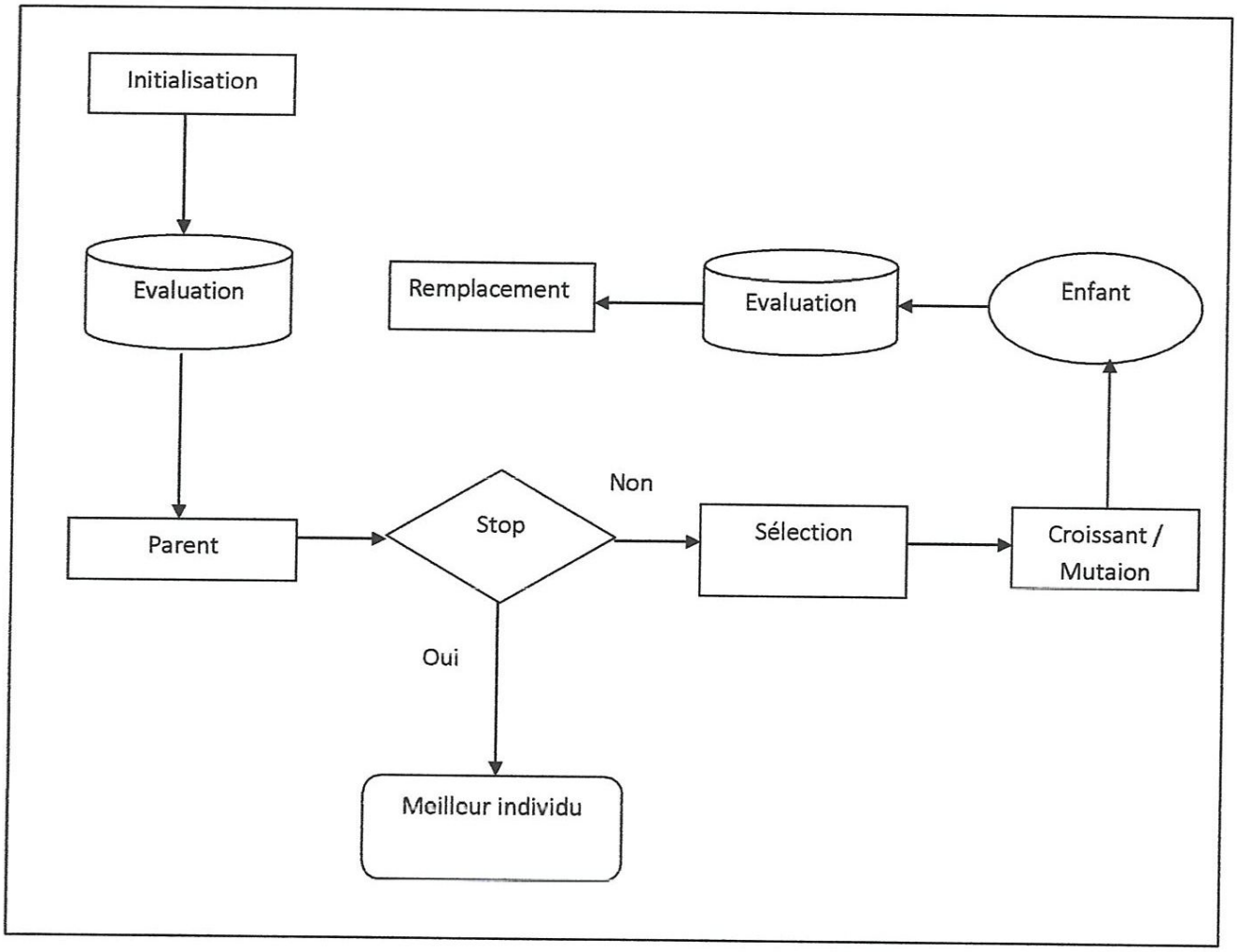

Figure 1 .18 : *Démarche d'un algorithme génétique.*

➢ *Principe générale* :

Un algorithme génétique recherche les extrema d'une fonction définie sur un espace de données appelé population initiale. Par analogie avec la génétique, chaque individu de cette population est un chromosome et chaque caractéristique de l'individu est un gène On part avec une population de solutions potentielles initiales, on évalue leur performance (fitness).Sur la base des ces performance on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples (la sélection, croisement et la mutation).On recommence ce cycle jusqu'à ce que l'on trouve une solution optimale [15].

➢ *Principe de croisement*

1. Sélectionner aléatoirement deux sites de croisement, et limiter les sections de croisement des deux parents.

2. Insérer le contenu de section de croisement du premier parent, au début de celle de deuxième parent.

3. Eliminer tous les bins du deuxième parent, qui contiennent des objets existants dans les autres bins qui composent le premier parent.

4. A l'aide d'une stratégie First Fit, réinsérer les objets qui apparaitre uniquement dans les bins supprimés.

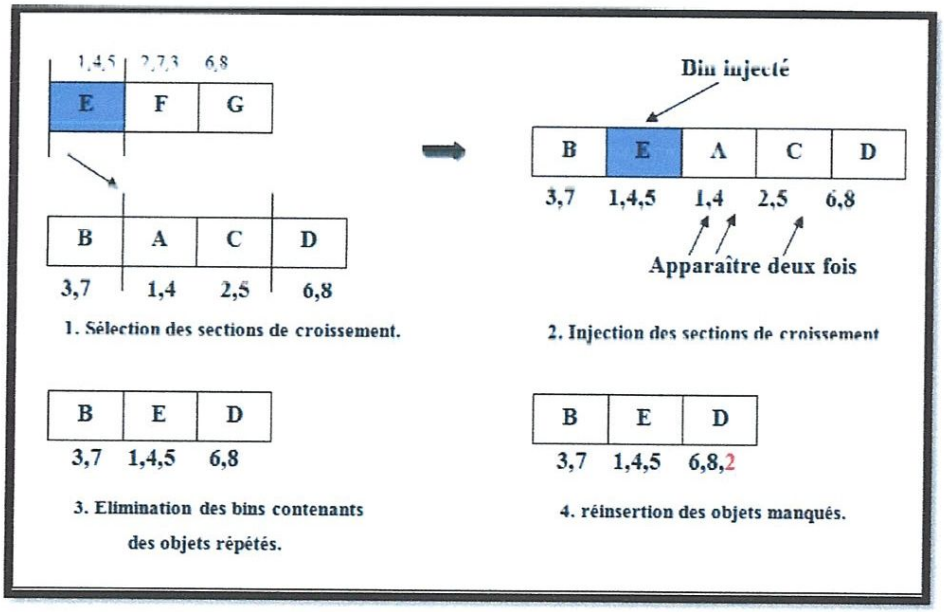

Figure 1.19 : *Exemple de croissement.*

b. opérateur de mutation :

Soit l'instance suivante d'un problème de bin packing :

Objet	1	2	3	4	5	6	7	8
Poids	5	4	2	7	4	3	5	2
Bin	Bin 1	Bin 2	Bin 1	Bin 3	Bin 2	Bin 1	Bin 4	Bin 3

1. Sélection d'une section pour la mutation.

1, 3, 6 | 2, 5, 8, 6 |

Section sélectionné

2. Insertion des objets sélectionnés par l'utilisation de l'heuristique FF.

Objet	1	2	3	4	5	6	7	8
Poids	5	4	2	7	4	3	5	2
Bin	Bin 1		Bin 1	Bin 3		Bin 1	Bin 4	Bin 3

- Insérer l'objet 2 dans le bin 4.
- Insérer l'objet 5 dans le bin 2.

Cet opérateur dispose de 4 grands avantages :

a. Il garantit la diversité de la population, ce qui est primordial pour les algorithmes génétiques.
b. Il permet d'éviter un phénomène connu sous le nom de ***dérive génétique***.
c. Il permet de limiter les risques d'une convergence prématurée causée par exemple par une méthode de sélection élitiste imposant à la population une pression sélective trop forte.

6.2.3. Méthodes hybrides :

Le principe général de l'hybridation consiste à combiner un algorithme avec une autre méthode, dont l'objectif est d'améliorer ses performances. Les algorithmes hybrides sont considérés parmi les méthodes les plus puissantes pour résoudre les problèmes d'optimisation combinatoires. Cette puissance réside dans la combinaison des deux principes de recherche de

meilleure solution, fondamentalement différents. Il existe plusieurs manières pour faire l'hybridation (par exemple: utilisant BF ou ses variantes avec AG, RT, RS). Ces approches d'hybridation ont permis de produire beaucoup de travaux dans la littérature [16].

Conclusion :

Nous avons traité dans ce chapitre le problème de bin-packing standard .Ce problème appartient à la classe des problèmes NP-difficiles, et pour le résoudre, on utilise souvent des méthodes et des algorithmes simples. Nous avons effectué une revue de littérature concernant les méthodes de résolution du BPP uni-dimensionnel, Ces méthodes sont réparties en deux grandes catégories à savoir, les méthodes exactes, les méthodes approchés (les heuristiques, les métaheuristiques et l'hybrides).

Dans le chapitre prochain, Nous avons opté pour la métaheuristique d'optimisation par la réaction chimique (ou Chemical Reaction Optimization (CRO)).

CHAPITRE 02

La métaheuristique CRO

« Ce qui ne nous tue pas nous rend plus fort »

Friedrich Nietzsche

Introduction

La plupart des méthodes que nous avons citées dans la partie précédente pour la résolution du problème de Bin Packing et généralement dans le domaine d'optimisation sont inspirées de la nature, par exemple l'algorithme génétique est inspiré du système biologique. Récemment, une nouvelle métaheuristique a été introduite en s'inspirant du phénomène de réactions chimiques. Cette métaheuristique est sous le nom d'Optimisation par la réaction chimique « Chemical Reaction Optimization ». Ce chapitre comporte une description de CRO, la similarité entre la réaction chimique et l'optimisation, les différents concepts de CRO. Ainsi son algorithme, ses différentes applications et finalement, ses avantages.

1. *Définition et Inspiration*

Il s'agit d'une métaheuristique qui a été développée récemment par Lam et Li en 2010. Elle est basée sur une population. Elle est inspirée par le phénomène d'interactions entre les molécules dans un processus de réaction chimique permettant la transformation des substances instables à d'autres plus stables. Le CRO imite les interactions des molécules dans une réaction chimique pour atteindre un état stable de basse énergie. Ainsi, le CRO peut être impliqué pour résoudre les problèmes dans les deux domaines discrets et continus [18]

La figure 2.1 présente bien le processus des réactions chimiques.

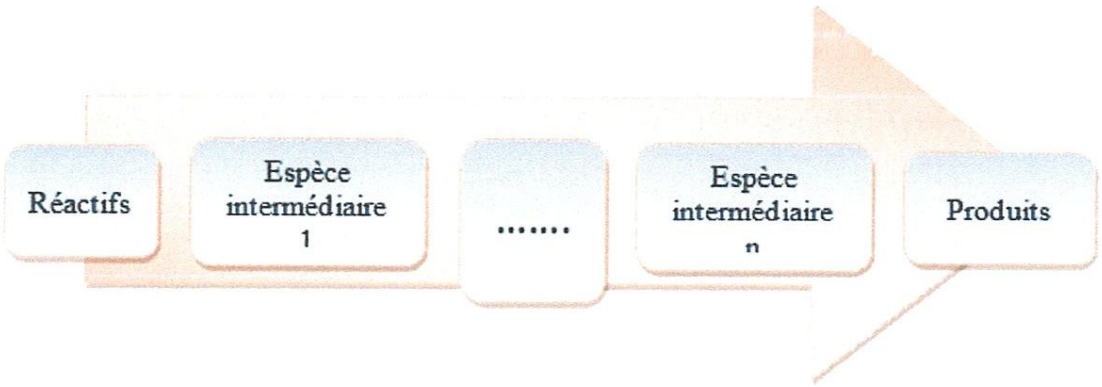

***Figure 2.1**: Processus des réactions chimiques.*

Il est à noter que chaque système de réaction cherche à réaliser un minimum d'énergie libre qui signifie que les réactions chimiques ont tendance à libérer de l'énergie. Donc les produits obtenus ont généralement moins d'énergie que les réactifs de départ. Le processus d'une réaction chimique est basé sur les deux lois de la thermodynamique [19] :

1. La loi de conservation de l'énergie :

Qui signifie que l'énergie ne peut pas être créée ou détruite. L'énergie peut se transformer d'une forme à une autre et se transférer d'une entité à une autre. Pratiquement, toutes les réactions chimiques sont accompagnées par l'absorption ou libération d'énergie. Dans une réaction chimique, chaque molécule possède deux types d'énergie: énergie potentielle (PE) et d'énergie cinétique (KE).

Pendant le processus de transformation moléculaire, une nouvelle structure moléculaire est obtenue par un transfert de l'énergie cinétique et potentielle.

2. Le principe d'évolution :

Qui énonce que l'entropie d'un système isolé ne diminue jamais, parce que les systèmes isolés évoluent spontanément vers l'équilibre thermodynamique (l'état d'entropie maximale), là où l'entropie est la mesure du degré de trouble. Comme nous le savons, l'énergie cinétique d'un objet est l'énergie qu'il contient en raison du mouvement. Si un objet est au repos, il n'a pas d'énergie cinétique, Contrairement à un objet en déplacement. Cependant, l'énergie potentielle est stockée dans un objet. Le travail se fait quand une force déplace un objet à une nouvelle position. En d'autres termes, l'énergie potentielle est stockée dans un système. Par conséquence, lorsque le PE est convertie en une autre forme (KE), le système devient plus désordonné et il se déplace rapidement, de sorte que son entropie augmente jusqu'à trouver l'équilibre thermodynamique, qui se caractérise par un minimum de PE. Donc pour bien comprendre comment la réaction chimique a lieu:

- Les substances initialement impliquées dans une réaction chimique sont appelés réactifs.
- Ces réactifs sont représentés par un ensemble de molécules, qui sont classés en différentes espèces.

- Pour qu'une réaction chimique ait lieu, les réactifs doivent entrer en collision. Les particules de la substance sont décomposées, et les atomes sont réarrangés dans de nouvelles particules, formant une nouvelle substance.

CRO a été appliqué pour résoudre des problèmes d'optimisation discrets.

Notons qu'une réaction chimique se traduit toujours par des produits plus stables (Avec un minimum d'énergie). Il est un processus de recherche pour le point optimal par étapes.

2. Similarité entre la réaction chimique et l'optimisation

Il existe deux points communs entre la réaction chimique et l'optimisation tels que [18]:
- Les deux visent à chercher le minimum tel qu'une réaction chimique se traduit toujours par des produits plus stables (avec un minimum d'énergie), également un problème d'optimisation vise l'optimum global qui correspond à la valeur minimal d'une fonction objectif.

• Les deux subissent une série d'événements étape par étape tel qu'une réaction chimique, des états moléculaires changent dans des collisions, également, dans un problème d'optimisation, différentes solutions sont obtenues dans les itérations.
A l'aide de cette correspondance, nous pouvons bien présenter l'idée de CRO en commençant par la modélisation de réactions chimiques dans un cadre d'optimisation, qui est illustré dans le tableau 2.1:

Réaction chimique	***Optimisation Combinatoire***
La molécule	**Solution du problème:** Une molécule est caractérisée par différents attributs comme suit : le type de l'atome, longueur de la liaison… Deux molécules ont des structures moléculaires différentes correspondant réellement à une solution dans le domaine mathématique. Autrement dit, deux structures moléculaires dans CRO représentent deux solutions possibles pour un problème d'optimisation.
Une collision	**Fonctionnement de base pour trouver la nouvelle solution:** Tout changement dans la structure moléculaire équivaut à passer à une autre solution possible, ce changement est appelé collision.
Energie potentielle	**Valeur de la fonction objectif:** Une molécule possède une énergie qui quantifie la structure moléculaire en termes d'énergies qui correspond à la valeur de la fonction objectif.

***Tableau 2.1**: Correspondance entre une réaction chimique et l'optimisation combinatoire.*

3. Concepts du CRO

3.1. Molécule :

Les agents manipulés en CRO sont des molécules, dont chacune se caractérise par un profil contenant certaines propriétés. Le tableau 2.2 représente les principaux attributs utilisés pour définir le mécanisme de base de CRO.

5. Réactions élémentaires du CRO :

Comme nous avons expliqué précédemment qu'il existe quatre réactions élémentaires du CRO tel que: La collision inefficace sur le mur, la collision de décomposition, la collision inefficace intermoléculaire et la collision de synthèse. Ces quatre réactions ont des propriétés différentes qui sont présentés ci-après [22].

1. Collision inefficace sur le mur

Elle représente l'opérateur de recherche locale de base pour CRO. Cet opérateur se produit lorsqu'une molécule entre en collision avec le mur du récipient, puis elle rebondit à une distance de la production d'une seule unité qui représente le voisin de la molécule d'origine. Ce type de réaction ne consomme pas beaucoup d'énergie. Par conséquent, la structure de la molécule ne change pas beaucoup par rapport à l'origine, seulement certains attributs moléculaires changent.

En supposant que w est la structure moléculaire de cette molécule, nous pouvons chercher w' du voisinage de w si et seulement si: $PEw + KEw \geq PEw'$.
Par conséquent, l'énergie cinétique de la structure de la molécule résultante w' peut être obtenue comme suit: $KEw' = (PEw + KEw - PEw') \times q$

Où $q \in [KELossRate]$ et le KELossRate est un paramètre du système qui limite la quantité maximale de KE perdue à la fois. Dans cette réaction, KE est divisée en deux parties de manière aléatoire, mais sous le contrôle du paramètre KELossRate. Une partie de l'énergie est affectée à la KE de la molécule résultante (($KEw' = (PEw + KEw - PEw') \times q$) et la partie restante est stockée dans la mémoire du tampon central ($PEw + KEw - PEw') \times (1 - q)$.

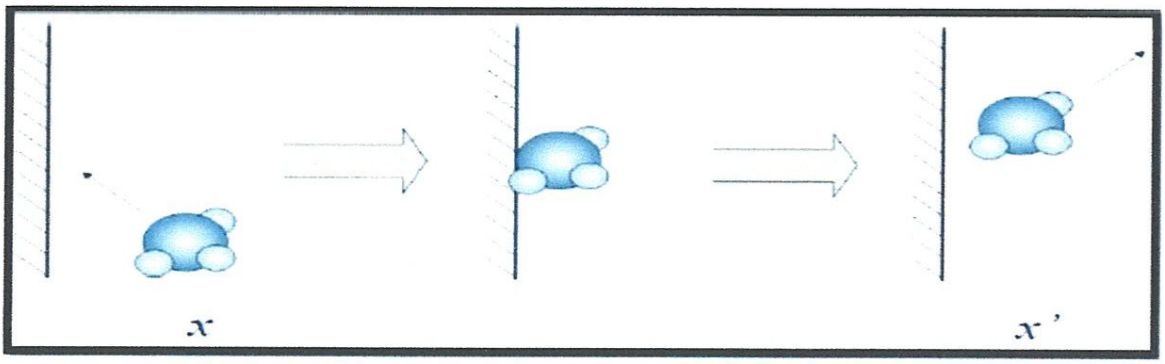

Figure 2.3: *Collision inefficace sur le mur.*

Le pseudo code de collision inefficace sur le mur est représenté ci-dessous:

Algorithme Collision-inefficace-sur-le-mur
Entrée : Une molécule M avec son profile et le tampon central de l'énergie *Buffer* Sortie : M et *Buffer* 01. Begin 02. Obtenir w' = voisinage (w) ; 03. Calculer PEw' ; 04. Si $PEw + KEw \geq PEw'$ Alors 05. Obtenir q au hasard dans [KELossRate, 1] 06. $KEw' = (PEw + KEw - PEw') \times q$; 07. Mettre à jour $Buffer = Buffer + (PEw + KEw - PEw') \times (1 - q)$ 08. Mettre à jour le profil de M par w = w', $PEw = PEw'$ et $KEw = KEw'$ 09. Fin Si 10. Fin

Algorithme 2.2: *Algorithme de Collision Inefficace sur le mur.*

2. Collision de décomposition

Au cours de cette réaction, une molécule frappe le mur et se divise en deux parties ou plus. Ce type de réaction consomme plus d'énergie que la précédente. Les molécules résultantes possèdent des structures moléculaires très différentes de l'originale. L'idée principale de décomposition est de permettre au système d'explorer d'autres régions de l'espace de solutions.
En supposant que w est la structure moléculaire d'une telle molécule et celles des molécules résultantes sont w1 et w2. La modification est permise dans deux solutions:

*) Quand la molécule originale a l'énergie suffisante (PE et KE), pour doter le PE de ceux résultants : $PEw + KEw \geq PEw1 + PEw2$

Dans ce cas, l'énergie est divisée au hasard en deux parties, dont chacune est affectée à la KE de l'une des deux nouvelles molécules comme suit: $temp1 = PEw + KEw - PEw1 - PEw2$
Nous obtenons alors : $KEw1 = temp1 \times k$ et $KEw2 = temp1 \times (1-k)$,où k est un nombre aléatoire généré à partir de l'intervalle [0.1].

*) La deuxième situation lorsque la molécule d'origine n'a pas suffisamment d'énergie (PE et KE), pour alimenter le PE des solutions résultantes. Dans de ce cas, nous prenons le tampon central d'énergie en considération pour combler le manque d'énergie. Ainsi la collision de décomposition sera autorisée: $PEw + KEw + Buffer \geq PEw1 + PEw2$
Pour faciliter la décomposition, nous exploitons l'énergie stockée dans le tampon. Dans ce cas, le tampon KE + *Buffer* sera divisé en trois parties comme suit :

- Deux parties pour l'énergie des deux nouvelles molécules,

$KEw1 = (temp1 + Buffer) \times m1 \times m2$, $KEw2 = (temp1 + Buffer - KEw1) \times m3 \times m4$.

- La partie restante est retournée dans le nouveau tampon central d'énergie de sorte Qu'il sera mis à jour, $Buffer = temp1 + Buffer - KEw1' - KEw2'$
  Notons que m1, m2, m3 et m4 sont indépendamment des nombres générés d'une manière aléatoire dans l'intervalle [0.1].

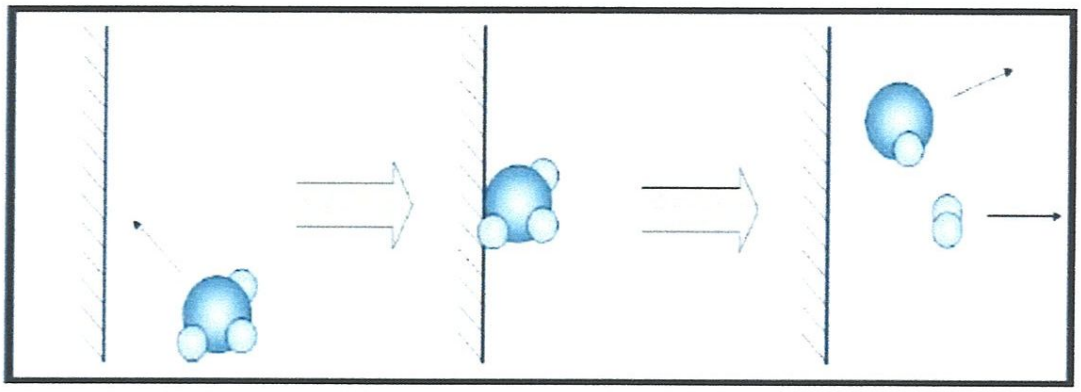

Figure 2.4: *Collision de décomposition.*

Le pseudo code de la collision de décomposition est présenté ci-dessous :

Algorithme collision de décomposition
Entrée : une molécule M avec son profil et le tampon centrale d'énergie *Buffer* Sortie : M_1 *et* M_2, *Buffer et Succes ;* 01.Begin 02. Obtenir w_1 et w_2 à partir de w 03. Calculer PEw_1 et PEw_2 04. $temp1 = PEw + KEw - PEw1 - PEw2$ 05. Créer une variable boolènne *Succes* 06. Si $temp1 \geq 0$ Alors 07. *Succes* = Vrai 08. Soit k une valeur aléatoire dans [0.1] 09. $KEw1 = temp1 \times k$ 10. $KEw2 = temp1 \times (1 - k)$ 11. Créer de nouvelles molécules M_1 et M_2 12. Attribuer w_1, PEw_1 et KEw_1 au profile de M_1, et w_2, PEw_2 et KEw_2 au profil de M_2 13. Sinon Si $temp1 + Buffer \geq 0$ Alors 14. *Succes* Vrai 15. Soit m_1, m_2, m_3 et m_4 sont des nombres aléatoires dans [0.1] 16. $KEw1 = (temp1 + Buffer) \times m1 \times m2$ 17. $KEw2 = (temp1 + Buffer - KEw1) \times m3 \times m4$ 18. Mettre à jour $Buffer = temp1 + Buffer$- KEw_1- KEw_2 19. Attribuer w_1, PEw_1 et KEw_1 au profile de M_1, et w_2, PEw_2 et PKw_2 au profile de M_2 20. Sinon

21. *Succes* = Faux
22. Fin Si
23. Fin

Algorithme 2.3: *Algorithme de Décomposition.*

Dans l'algorithme de CRO, la collision de décomposition est exécutée si le critère de décomposition qui est défini ci-dessous est vérifié.

- ***Critère de décomposition :***

Pendant l'exécution de l'algorithme CRO, une série de différentes collisions se produisent, donc les solutions tentent à savoir moins d'énergie cinétique et aboutir à des solutions avec d'énergies potentielles inférieures. Les minimums locaux peuvent alors être atteints. Lorsqu'il est estimé que la solution correspondant à une molécule est un minimum local, la molécule essaie de passer à une nouvelle région de l'espace de recherche par la décomposition. On peut décider si un minimum local a été atteint selon le nombre de collisions qui se sont produites sans entrainant une diminution de la valeur de la fonction. Ceci représente le critère de décomposition. Le critère de décomposition est défini par : **NumHit – MinHit ≻ α** (Avec α représente le nombre de fois qu'une molécule a subi sans localiser un meilleur minimum local).

3. Collision inefficace inter-moléculaire

Cette réaction illustre le cas de deux molécules ou plus qui entrent en collision et puis rebondissent. Dans cette réaction, il n'y a aucune quantité d'énergie KE perdue ni stockée dans la mémoire de tampon. Cette dernière peut être considérée comme deux collisions inefficaces sur le mur qui se produisent au même temps. La différence par rapport à la collision inefficace sur le mur se trouve au niveau du processus de traitement d'énergie tel que dans une collision inefficace inter-moléculaire, l'énergie excessive des deux molécules est combinée. Donc, la possibilité qu'une molécule ait une plus grande valeur de f (w') est plus élevée dans une collision inefficace intermoléculaire que dans une collision inefficace sur le mur. En conséquence, cette collision ne sera acceptée que si:

$$temp2 = PEw1 + KEw1 + PEw2 + KEw2 \; - PEw1' - PEw2'$$

L'énergie KE des molécules résultantes est alors définie :

$$KEw1' = temp2 \times p \text{ et } KEw2' = temp2 \times (1 - p)$$

où p est un nombre généré d'une façon aléatoire dans [0.1].

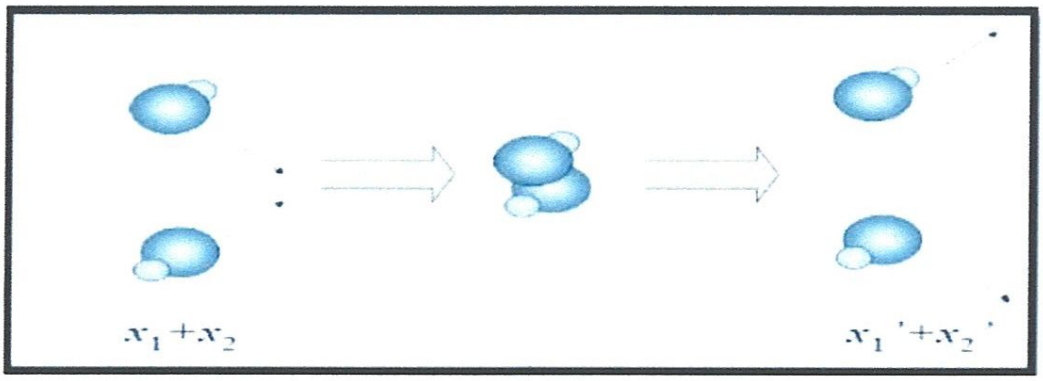

***Figure 2.5**: Collision inefficace intermoléculaire.*

Le pseudo code de la collision inefficace inter-moléculaire est représenté ci-dessous :

Algorithme Collision ineffectif-inter-moléculaire
Entrée : Molécules M1, M2 avec leurs profils Sortie : M1 et M2 01. Begin 02. Obtenir w1 = voisinage (w1) et w2 = voisinage (w2) 03. Calculer PEw1' et PEw2' 04. Soit $temp2 = (PEw1 + PEw2 + KEw1 + KEw2) - (PEw1'+PEw2')$ 05. Si $temp2 \geq 0$ Alors 06. Soit p une valeur aléatoire dans [0.1] 07. $KEw1' = temp2 \times p$ 08. $KEw2' = temp2 \times (1-p)$ 09. Mettre à jour le profile de M1 par w1 = w1', PEw1 = PEw1' et KEw1 = KEw1', et le profile de M2 par w2 = w2', PEw2 = PEw2' et KEw2 = KEw2' 10. Fin Si 11. Fin

***Algorithme 2.4**: Algorithme de Collision Inefficace inter-moléculaire.*

1. Collision de synthèse

Cet opérateur implique deux ou plusieurs molécules qui sont combinées ensemble pour former une nouvelle structure moléculaire. Cette collision ne comporte pas de tampon car elle combine l'énergie des molécules impliquées. La structure moléculaire w résultante est très différente de structures originales. Ce type de collision contrôle principalement la taille de la population et la diversité de la population.

En supposant que les structures moléculaires des deux molécules originales sont w1 et w2, Une nouvelle molécule est générée avec la structure moléculaire w'. La collision de synthèse

sera acceptée si: $PEw1 + KEw1 + PEw2 + KEw2 \geq PEw$.
Nous obtenons alors :$KEw = PEw1 + PEw2 + KEw1 + KEw2 - PEw1$.
Ainsi, KEw' est très grande en comparaison avec KEw1 et KEw2 de sorte que la nouvelle molécule a une grande chance d'échapper à un optimum local. En fait, quand une molécule a peu d'énergie, elle risque de rester coincée dans un minimum local. Par conséquent, nous pouvons dire que la synthèse permet l'exploration de nouvelles régions de l'espace de solution.

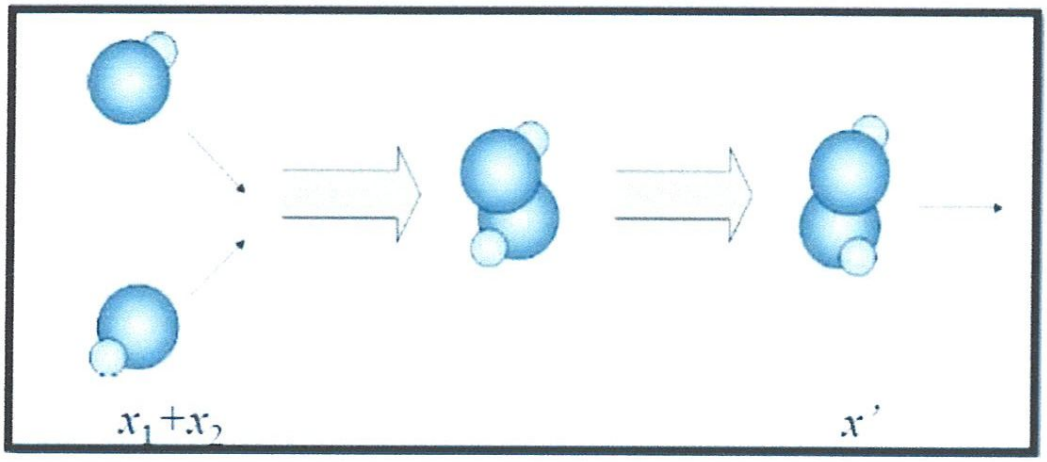

***Figure 2.6**: Collision de synthèse.*

Le pseudo code de la collision de synthèse est représenté ci-dessous:

Algorithme collision de synthèse
Entrée : Molécule M_1, M_2 avec leurs profils Sortie : M' et *Succes* 01. Begin 02. Obtenir w' à partir de $w1$ et $w2$ 03. Calculer PEw' 04. Créer une variable booléenne 05. Créer une nouvelle Molécule M' 06. Si $PEw1 + PEw2 + KEw1 + KEw2 \geq PEw'$ Alors 07. *Succes* = Vrai 08. $KEw' = PEw1 + PEw2 + KEw1 + KEw2 - PEw'$ 09. Attribuer w', PEw' et KEw' au profil de M' 10. Sinon 11. *Succes* = Faux 12. Fin Si 13. Fin

***Algorithme 2.5 :** Algorithme de collision de synthèse.*

Dans l'algorithme CRO, la collision de synthèse est exécutée si le critère de synthèse qui est défini ci dessous est vérifié.

- ***Critère de synthèse :***

La collision de synthèse permet de générer une nouvelle molécule avec solution w' à partir de deux molécules existantes w1 et w2 en appliquant un opérateur pour obtenir cette nouvelle solution. Le critère de synthèse est utilisé pour tester si deux molécules, dans le même temps n'ont pas d'énergie cinétique (KE) suffisamment, c'est-à-dire KEw1 $< \beta$ et KEw2 $< \beta$ où β définit le moins de l'énergie cinétique (KE) qu'une molécule doit posséder. Quand une molécule a trop peu de KE, elle perd potentiellement sa capacité à échapper à un minimum local. Lorsque deux de ces molécules entrent en collision, elles se combinent et forment une nouvelle molécule dont KE devient grand encore. Puis la molécule résultante peut continuer à explorer différentes parties avec la possibilité d'éviter de se retrouver dans un minimum local. Le critère de synthèse est défini par : KE $\leq \beta$.

6. Classification des réactions élémentaires :

Nous pouvons classer les opérateurs de CRO selon le nombre de molécules impliquées. Le tableau 2.3 présente la classification des opérateurs de CRO. Nous distinguons les collisions uni-moléculaires et les collisions inter-moléculaires [23].

	Collision uni moléculaire	*Collision inter moléculaire*
Collision avec population de taille fixe	Collision inefficace sur le mur	Collision inefficace inter moléculaire
Collision avec population de taille variable	Décomposition	Synthèse

Tableau 2.3: *Classification des réactions élémentaires selon le critère de molécularité.*

Pour les collisions inefficaces uni-moléculaire et inter-moléculaire, l'opérateur de CRO ne peut pas modifier le nombre total de molécules dans le système. Tandis que, les collisions de décomposition et de synthèse peuvent agir sur la taille de la population en la diminuant ou l'augmentant.
Les structures moléculaires des molécules impliquées ne seront changées que dans les réactions réussies. Le CRO permet de modifier les structures moléculaires des molécules pour explorer l'espace des solutions. Le degré de diversification et d'intensification sont soumis aux mécanismes réels utilisés dans les collisions et donc ce critère peut classer les opérateurs.

Les critères de décomposition et de synthèse, et ainsi les mécanismes utilisés dans la collision de décomposition et la collision de synthèse pour générer des nouvelles solutions peuvent être modifiées et ajustés en fonction de la nature du problème. Le tableau 2.4 présente la classification des réactions élémentaires selon le critère d'intensification et la diversification

Réaction élémentaire	***Intensification***	***Diversification***
Inefficace sur le mur	√	
Décomposition		√
Inter moléculaire	√	
Synthèse	√	√

Tableau 2.4: *Classification des réactions élémentaires selon le critère d'intensification et de l'exploration.*

- ***Collision inefficace sur le mur*** **:** La structure moléculaire courante a l'intention d'obtenir une nouvelle structure dans son voisinage.
- ***Collision de décomposition*** **:** Elle donne un moyen d'explorer d'autres régions, si une solution tente de passer à une nouvelle zone de l'espace de recherche à travers la décomposition.
- ***Collision inefficace inter-moléculaire*** **:** Deux structures moléculaires originales ont tendance à obtenir deux nouvelles structures moléculaires de leurs voisinages.
- ***Collision de synthèse*** **:** L'effet de synthèse peut être régler selon les besoins. Il peut être considéré soit un opérateur d'intensification, depuis, il produit une solution unique à partir de deux ou plusieurs solutions, soit un opérateur de diversification les énergies des molécules, de sorte que l'augmentation de la capacité permet d'explorer d'autres régions.

7. Applications du CRO :

Dans le monde réel, la science moderne et l'industrie sont en effet riche en problèmes d'optimisation. Bien que le CRO soit un algorithme nouvellement proposé, il a été appliqué à des nombreux problèmes avec succès. Voici quelques applications [24]:

- ***Problème d'affectation quadratique :***

Problème d'affectation quadratique est un problème combinatoire fondamental dans la recherche opérationnelle. Son objectif est de minimiser le coût de transport en faisant varie les emplacements des installations. Considérer l'affectation des installations n à n emplacements. La distance est entre chaque paire d'emplacements et le flux humain entre chaque paire d'installations. Le problème est de minimiser le coût total en disposant les emplacements des installations. Il existe une version parallèle de CRO grâce à une stratégie de communication synchrone est proposée pour résoudre ce problème [19]. Dans cette version, le temps de calcul et la qualité de la mise en place de solution de la version parallèle sont améliorées par rapport à ceux de la version séquentielle.

- ***Problème d'ordonnancement des projets aux ressources limitées :***

Ce problème est l'un des problèmes d'optimisation les plus difficiles à résoudre dans la recherche opérationnelle, lié à la gestion, l'affectation des ressources du projet, et le processus de fabrication. On considère qu'il y a des activités sont programmées pour un projet et le temps est divisé en intervalles. Chaque activité nécessite des ressources pour être traitées et elles peuvent couvrir plus d'un intervalle de temps. Les ressources sont limitées, donc il faut décider quelles activités doivent être prises en charge dans un certain intervalle de temps. Il y a aussi des contraintes de priorité entre les activités. C'est un problème à un seul objectif qui est la minimisation de la durée de projet. Pour la résolution de ce problème, il existe une version de CRO qui est adoptée par.

- ***Problème d'affectation de canal dans les réseaux maillés sans fil :***

Ce problème est NP_Difficile. Un réseau maillé sans fil est composé de certains routeurs maillés sans fil fixes, dont chacun est équipé de certaines interfaces radio et plusieurs canaux sont disponibles pour la communication. L'objectif de ce problème est de minimiser l'interférence globale du réseau. Comme le nombre de radios sur un nœud peut être inférieur au nombre de canaux disponibles, l'affectation de canal doit obéir à la contrainte que le nombre de différents canaux affectés à l'incident de liens sur tout nœud est au plus le nombre d'interfaces radio sur ce nœud. Le CRO peut améliorer les solutions existantes à ce problème.

- ***Problème de transition de la population en streaming peer to peer :***

  Le principe de ce problème est qu'une source de flux fournissant des données en streaming,

ainsi que des pairs recevant les données. En raison des conditions de réseau hétérogène, les pairs subissent des retards différents de transmission des données de la source et elles peuvent être regroupées en fonction des colonies retards. Le système est en streaming universelle où tous les pairs sont servis avec des flux de données suffisantes. Les pairs peuvent rejoindre et quitter le système et passer à une autre colonie alors que le système peut imposer des règles pour guider les pairs pour rejoindre les colonies. L'objectif de ce problème est de maximiser la probabilité de diffusion universelle par l'attribution des probabilités de transition de la population parmi toutes les colonies. L'approche CRO est bien exécutée pour ce problème par rapport aux autres stratégies pratiques.

8. Les Avantages du CRO :

Les avantages de CRO sont mis en évidence comme suit [23]:

- CRO est un cadre de conception qui permet de déployer différents opérateurs pour s'adapter à différents problèmes.
- Sa taille de population variable permet au système de s'adapter automatiquement aux problèmes.
- La conversion d'énergie et le transfert d'énergie dans différentes entités et sous des formes différentes rendent CRO unique parmi les métaheuristiques.
- CRO a le potentiel pour aborder les problèmes qui n'ont pas été résolus avec succès par d'autres métaheuristiques.
- CRO peut être facilement programmé dans le langage de programmation orienté objet, où une classe définit une molécule et les méthodes définissent les types de réaction élémentaires.
- Il est facile de modifier CRO pour fonctionner en parallèle, comme la taille de la population n'a pas besoin d'être synchronisé entre les unités de calcul.

Conclusion :

Dans ce chapitre, nous nous identifions la métaheuristique CRO, analysons les différents concepts et son algorithme et nous avons également fourni quelques exemples des applications.

Dans le chapitre suivant, une étude détaillée des différentes réactions élémentaires de la métaheuristique CRO appliquée pour la résolution du BPP avec des exemples illustratifs.

Chapitre 03

La métaheuristique CRO pour le problème de Bin Packing

« Ce qui ne nous tue pas nous rend plus fort »

Friedrich Nietzsche

Introduction :

L'objectif de ce mémoire est d'implémenter cette nouvelle méthode (CRO) pour résoudre le problème de bin packing uni-dimensionnel. Rappelons que ce problème qui a fait l'objet de plusieurs travaux de recherche étant donné sa complexité de résolution (problème NP-Difficile).

Dans ce chapitre, nous décrivons la métaheuristique CRO pour la résolution du BPP. Nous nous concentrons sur ces différentes étapes. Nous allons dans ce qui suit présentés la métaheuristique CRO pour le cas du BPP, partant de la représentation d'une solution, fonction d'évaluation, passant par les réactions élémentaires, l'arrêt de l'algorithme. Ensuite, nous l'avons exécuté sur des problèmes Benchmark en vue d'une étude comparative des solutions générées par rapport aux solutions existantes.

1. *Représentation d'une solution :*

Dans la métaheuristique CRO, la représentation est conçue avec les bins à l'esprit, et non des objets individuels, de sorte que chaque molécule représente une liste des bins, chaque bin est un groupe d'objets (non un objet individuel). Notre but est de construire des molécules (solutions) minimisent le nombre du bins utilisés par chaque molécule en respectant les contraintes de capacité. La figure 3.1 qui est représentée ci-dessous illustre la représentation moléculaire d'une solution dans la métaheuristique CRO.

- *Exemple :*
  On a la liste des objets suivante (La capacité de chaque bin est 10).

Objets	1	2	3	4	5	6
Poids	2	4	5	6	7	9

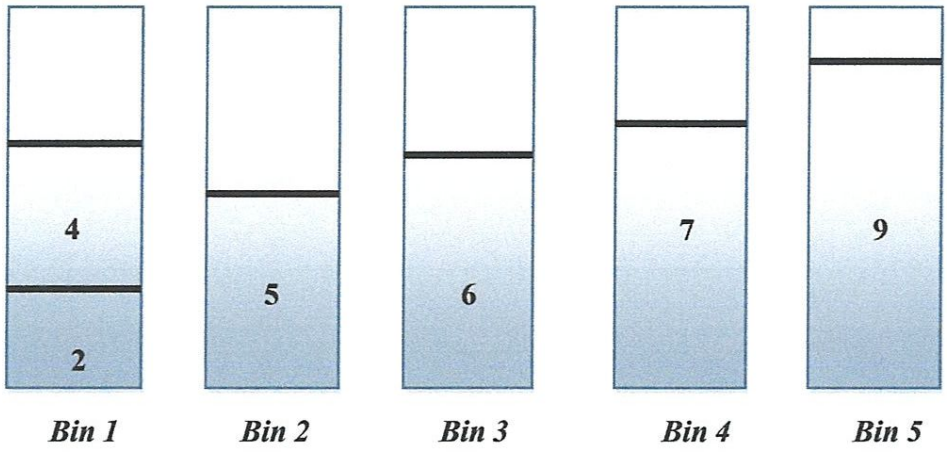

Figure 3.1 : Représentation *d'une solution.*

2. ***Fonction d'évaluation (Fitness) :***

Un facteur crucial pour toute résolution d'un problème d'optimisation combinatoire est la fonction objectif (fitness), une bonne fonction objectif aide énormément dans la recherche de bonne solution. Dans le contexte du problème du bin packing, choisir la fonction objectif classique, qui consiste à minimiser le nombre de bins, n'est pas pertinente étant donné que plusieurs différentes solutions peuvent avoir le même nombre de bins sans avoir le même rangement d'objets. Néanmoins, cette information est intéressante à condition qu'elle soit couplée avec d'autres informations. Pour cela nous avons choisi une fonction d'évaluation F (proposée par Falkenauer) qui contient des informations sur les poids des objets et la capacité des bins, notre objectif est minimiser la valeur de cette fonction d'évaluation:

$$F = 1 - \left(\frac{\sum_{i=1}^{n} \left(\frac{f_i}{c}\right)^k}{n} \right)$$

Tels que :

F : la fitness de la solution. n: nombre du bins. f_i : Remplissage du ième bin. C : la capacité du bin. k : Constant plus grand que 1 (Supposons que k = 4).

Nous avons écrit la formule de F sous cette forme $\left(1 - \left(\frac{\sum_{i=1}^{n} \left(\frac{f_i}{c}\right)^k}{n} \right)\right)$ Afin de la convertir de fonction de maximisation à fonction de minimisation. Un problème de maximisation d'une fonction f est équivalent au problème de minimisation $-f$, on a donc :

$$\max f(x) = \min -f(x)_{x \in X} \text{ (X : ensemble de solution).}$$

La métaheuristique CRO devrait minimiser cette valeur (F).

➢ *Exemple :*

Pour bien comprendre cette opération nous l'avons appliqué sur l'exemple qui a été mentionné précédemment :

$$\left(\frac{f_1}{c}\right)^4 = \left(\frac{6}{10}\right)^4 = 0{,}1296.$$

$$\left(\frac{f_2}{c}\right)^4 = \left(\frac{5}{10}\right)^4 = 0{,}0625.$$

$$\left(\frac{f_3}{c}\right)^4 = \left(\frac{6}{10}\right)^4 = 0{,}1296.$$

$$\left(\frac{f_4}{c}\right)^4 = \left(\frac{7}{10}\right)^4 = 0{,}2401.$$

$$\left(\frac{f_5}{c}\right)^1 = \left(\frac{9}{10}\right)^1 = 0{,}6561.$$

$$F - 1 - \left(\frac{\sum_{i=1}^{5}\left(\frac{f_i}{c}\right)^4}{5}\right)$$

$$= 1 - \left(\frac{0{,}1296 + 0{,}0625 + 0{,}1296 + 0{,}2401 + 0{,}6561}{5}\right) = 0{,}75642.$$

3. ***Processus d'optimisation :***

Dans cette section, nous allons illustrer l'algorithme CRO tel qu'il a été adapté au problème de bin packing uni-dimensionnel.

3.1. Etape d'initialisation :

Durant cette étape, les paramètres PopSize,FELimit, KELossRate, MoleColl et InitialKE sont fixés. Pop est une liste qui représente l'ensemble des molécules dont chacune représente une solution. La taille de cette liste est fixe.

3.2. Etape d'itération :

Une fois que toutes les données sont définies, nous pouvons commencer la première itération de l'algorithme. Cette étape est entamée par la génération d'une nouvelle solution qui sera construite selon les quatre réactions élémentaires présentés précédemment: Collision inefficace sur le mur, collision inefficace inter-moléculaire, décomposition et synthèse.

a. Collision inefficace sur le mur :

Nous considérons l'exemple de la liste Pop du tableau 3.1. Nous supposons que la solution sélectionnée à partir de cette liste Pop est la deuxième solution nommée x ayant la structure moléculaire suivante :

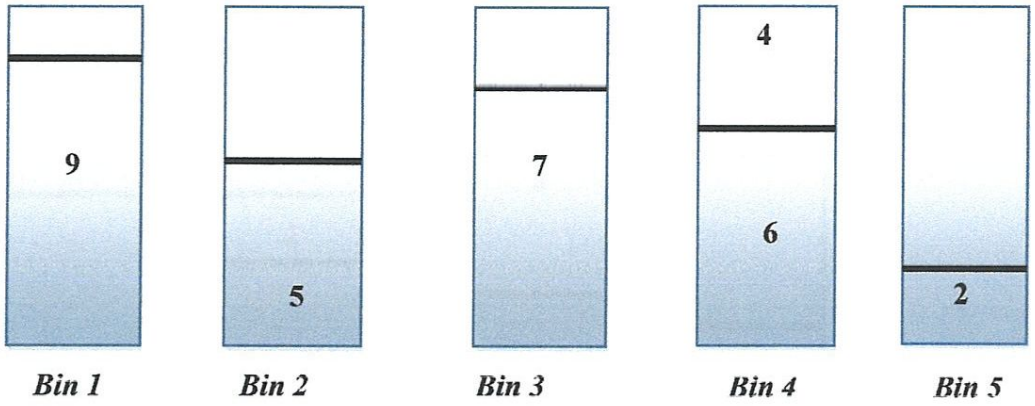

Figure 3.2 : *Structure moléculaire de la solution choisie.*

Avec PE_x =0,60794 et $KE_x = 1 - PE_x = 0{,}39206$.

Comme nous l'avons expliqué dans le chapitre 2, le principe de cette collision est qu'une molécule entre en collision avec le mur de récipient, puis elle rebondit à une distance de la production d'une seule molécule qui représente le voisin de la molécule d'origine.

Pour générer la nouvelle solution x', deux bins sont sélectionnés au hasard et supprimer. La suppression des bins permettra de réarranger les objets qui se trouvaient dans les bins supprimés après réinsertion. Cela, nous l'espérons, entraînera des améliorations.

x'' → y'' :

Nous commençons par la deuxième solution sélectionnée x'' pour aboutir à la première nouvelle solution nommée y''.Nous choisissons aléatoirement deux bins de la solution x', ensuite nous supprimons les objets des bins choisis de x'' et les mettre dans la liste des objets non affectés.

	Bin 1	***Bin 2***	***Bin 3***	***Bin 4***	***Bin 5***
Solution x'	2,4	5	6	7	9

Bins choisi

Supprimer donc les objets du bin 1 et 2 de la solution x'' et les mettre dans une liste :

2	4	5	← Objets non affectés

Réinsérer ces objets dans x'' avec l'heuristique simple BFD, nous obtenons une nouvelle solution y''.

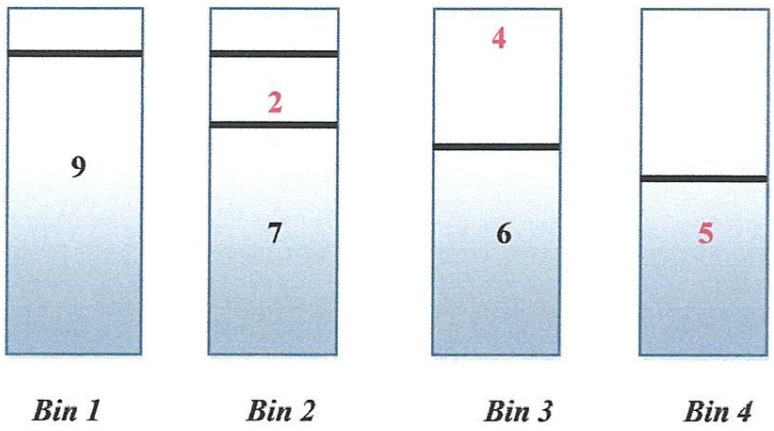

Figure 3.6 : *Nouvelle Solution y''.*

Avec $PE_{y''}$ =0,406325.

x' → y'

Ensuite, à partir de la première solution sélectionnée x', nous pouvons générer la deuxième nouvelle solution nommée y'.Supposons que les deux bins choisi sont bin 1 et bin 4 de x'', Ils contiennent trois objets du poids 5, 4 et 6, supprimer ces objets de la solution x', les mettre dans la liste des objets non affectés et après les réinséré dans la solution x' avec BFD.

Nous obtenons la deuxième nouvelle solution y' :

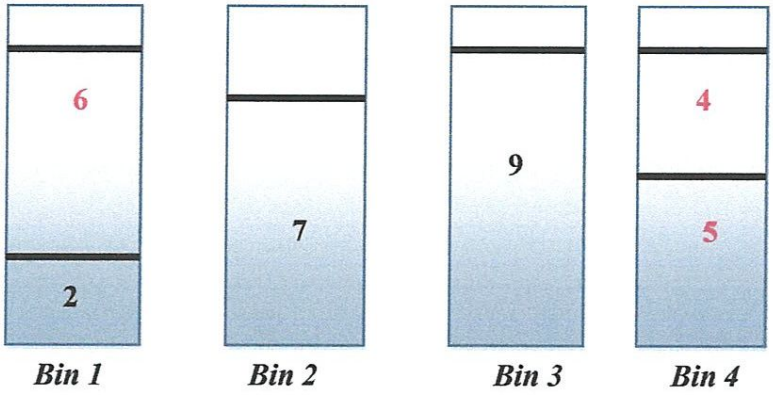

Figure 3.7 : *Nouvelle Solution y'.*

Avec $PE_{y'} = 0{,}509525$.

Après avoir construit les deux nouvelles solutions, il faut vérifier s'il est possible d'appliquer la collision inefficace inter-moléculaire en vérifiant que la somme d'énergie de deux solutions d'origine x' et x'' est suffisante pour produire les deux nouvelles solutions y' et y''. Alors, selon la formule présentée dans l'algorithme de collision inefficace inter-moléculaire (dans le chapitre 02), nous supposons une variable nommée temp2 égale à :

$$\text{Temp2} = (PE_{x'} + PE_{x''} + KE_{x'} + KE_{x''}) - (PE_{y'} + PE_{y''}) = (0{,}75642 + 0{,}475525 + 0{,}39206 + 0{,}524475) - (0{,}509525 + 0{,}406325) = 1{,}23263 \geq 0.$$

Après la satisfaction de cette condition, nous calculons les énergies cinétiques de deux nouvelles solutions selon les deux formules présentées dans l'algorithme de collision inefficace inter-moléculaire. Nous supposons une variable p dans la plage de [0.1] dont

p = 0.4, alors :

$$KE_{y'} = \text{Temp2} * p = 1{,}23263 * 0{,}4 = 0{,}493052.$$
$$KE_{y''} = \text{Temp2} * (1 - p) = 1{,}23263 * (1 - 0{,}4) = 0{,}739578.$$

Donc:

Solution y' avec $PE_{y'} = 0{,}509525$, $KE_{y'} = 0{,}493052$.

Solution y'' avec $PE_{y''} = 0{,}406325$, $KE_{y''} = 0{,}739578$.

La dernière instruction de cette collision est la mise à jour de la liste Pop, en intégrant les deux nouvelles solutions générées y' et y'' pour remplacer les deux solutions x' et x'' respectivement. Alors la liste Pop devient :

	Bin 1	***Bin 2***	***Bin 3***	***Bin 4***	***Bin 5***	***Fitness(PE)***
Solution 1	2,6	7	9	5,4	/	0,509525
Solution 2	9	7,2	6,4	5	/	0,406325
Solution 3	5	9	7,2	6,4	/	0,406325

Tableau 3.3: *liste Pop après une collision inefficace intermoléculaire.*

✓ *Algorithme d'inter moléculaire*

Entrée : Deux molécule x', x''.

Sortie : Deux nouvelles molécules y' et y''.

1: Dupliquer x' pour générer y'
2: Dupliquer x'' pour générer y''
3: Sélectionner aléatoirement deux bins de y'
4: Sélectionner aléatoirement deux bins de y''

5 : Supprimer de y' les objets des bins sélectionnés de y''
6 : Supprimer de y'' les objets des bins sélectionnés de y'

7 : Réinsérer les objets supprimés de y'dans y' selon la stratégie BFD
8 : Réinsérer les objets supprimés de y'' dans y''selon la stratégie BFD

9 : Retourner y' et y''.

Algorithme 3.3 : *Collision inefficace inter moléculaire.*

c. Collision de décomposition

Dans cette collision, une molécule x frappe le mur et se divise en deux parties ou plus (x' et x'').

Au début de cette collision, une solution est sélectionnée à partir de la liste Pop d'une façon aléatoire. Nous supposons que cette solution est la première solution ayant la structure moléculaire suivante :

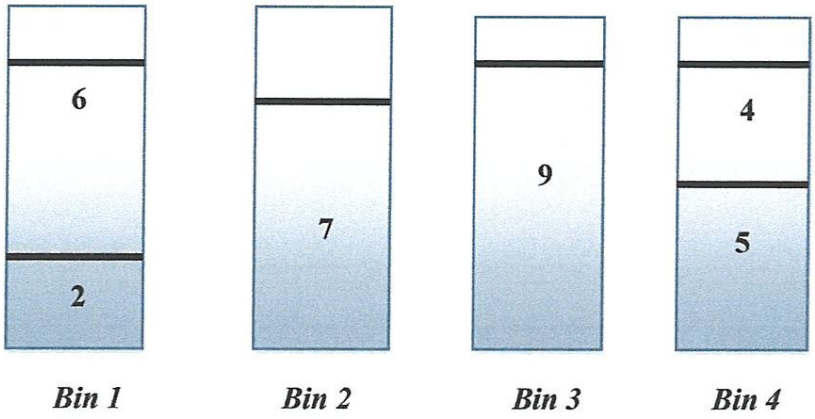

Figure 3.8 : *Structure de solution choisie x.*

Ensuite, nous avons affecté aléatoirement les bins de la solution choisie x à x' et x'', nous supposons que :

x' contient un seul bin (Bin 2).

x'' contient trois bin (Bin 1, Bin 3, Bin 4).

L'étape suivante consiste à compléter les deux solutions x' et x'' avec une heuristique simple (Que ce soit Next Fit).Nous obtenons :

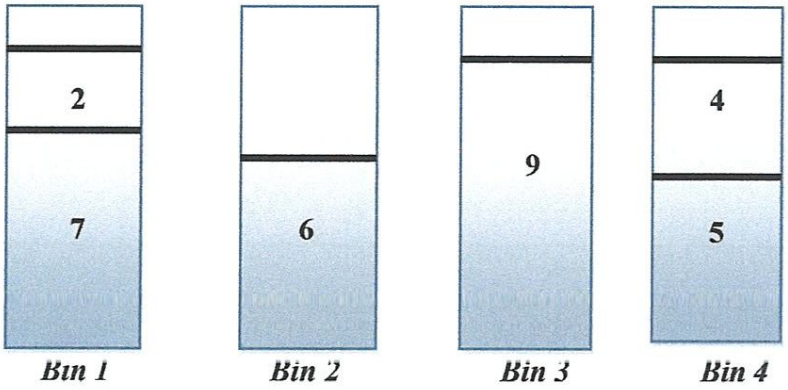

Figure 3.9 : *Nouvelle solution x'.*

Avec $PE_{x'} = 0{,}475525$.

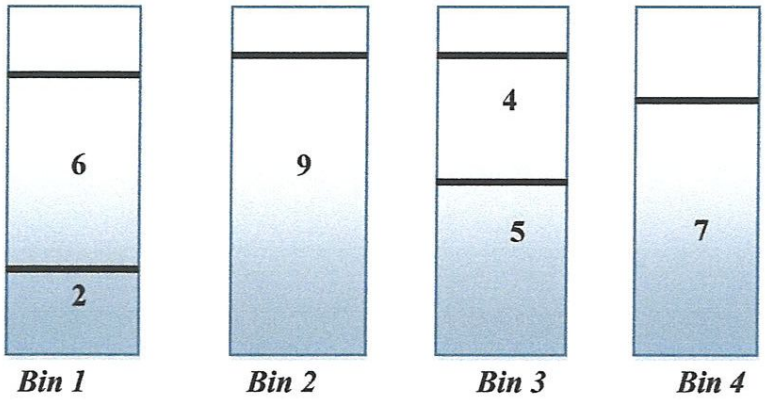

Figure 3.10 : *Nouvelle solution x''.*

Avec $PE_{x''} = 0{,}509525$

Dans notre exemple, la solution d'origine a suffisamment d'énergie pour doter l'énergie de ceux résultantes :

$$PE_x + KE_x = 0{,}509525 + 0{,}493052 \geq PE_{x'} + PE_{x''} = 0{,}475525 + 0{,}509525 .$$

Après la satisfaction de cette condition, nous calculons les énergies cinétiques de deux nouvelles solutions :

Nous obtenons alors :

$$temp1 = PE_x + KE_x - PE_{x'} - PE_{x''} = 0{,}509525 + 0{,}493052 - 0{,}475525 - 0{,}509525$$

$$= 0{,}017527.$$

$KE_{x'} = temp1 \times k = 0{,}017527 * 0{,}8 = 0{,}0140216.$

$KE_{x''} = temp1 \times (1 - k) = 0{,}017527 * (1 - 0{,}8) = 0{,}0035054.$

k est un nombre aléatoire généré à partir de l'intervalle [0.1].

La dernière instruction de cette collision est la mise à jour de la liste Pop. La liste Pop devient alors la structure illustrée dans le tableau 3.4.

	Bin 1	***Bin 2***	***Bin 3***	***Bin 4***	***Bin 5***	***Fitness(PE)***
Solution 1	7,2	6	9	5,4	/	0,475525
Solution 2	2,6	9	5,4	7		0,509525
Solution 3	9	7,2	6,4	5	/	0,406325
Solution 4	5	9	7,2	6,4	/	0,406325

Tableau 3.4 : *La liste Pop après une collision de décomposition.*

✓ *Algorithme de décomposition*

Entrée : Molécule x.
Sortie : deux nouvelle molécule x' et x''.

```
1 : Pour chaque bin de x.
2 : Générer p aléatoirement entre 0 et 1.
3 : Si p >= 0,5 alors
4 :     Affecter bin à x'.
5 : Sinon
6 :     Affecter bin à x''
7 : Fin Pour
8 : Compléter x' avec une heuristique aléatoire.
9 : Compléter x'' avec une heuristique aléatoire.
10 : Retourner x' et x''.
```

Algorithme 3.4 : *Algorithme de décomposition.*

d. Collision de synthèse

C'est la dernière collision présentée dans l'algorithme de CRO dont le principe est de combiner deux ou plusieurs molécules ensemble pour former une nouvelle structure moléculaire (on va combiner deux molécule). Le début de cette collision est la sélection de deux solutions d'une façon aléatoire.

La capacité de chaque bin = 100.

Bin	Espace	Liste d'objets
1	0	21, 40, 39
2	5	47, 48
3	5	67, 28
4	5	41, 54
5	5	82, 13
6	5	25, 70
7	9	35, 56
8	15	85
9	30	70
10	69	12, 19
11	75	25

Tableau 3.5 : *Structure moléculaire de la solution* x_1

Avec $PE_{x1} = 0{,}40604082$, $KE_{x1} = 1 - PE_{x1} = 0{,}59395918$

Bin	Espace	Liste d'objets
1	3	85, 12
2	3	56, 41
3	5	25, 70
4	5	67,28
5	5	82, 13
6	20	21, 19, 40
7	30	70
8	46	54
9	52	48
10	53	47
11	61	39
12	65	35
13	75	25

Tableau 3.6 : *Structure moléculaire de la solution* x_2

Avec $PE_{x2} = 0{,}60824811$, $KE_{x2} = 0{,}39175189$

La première étape consiste à trié les bins selon le remplissage de chaque bin :

Bin	Espace	Liste d'objets
1	0	21, 40, 39
2	3	85, 12
3	3	56, 41
4	5	47, 48
5	5	67, 28
	5	41, 54
6	5	82, 13
7	5	25, 70
8	5	25, 70
	5	67, 28
	5	82, 13
	9	35, 56
	15	85
	20	21, 19, 40
	30	70
9	46	54
	52	48
	53	47
	61	39
	65	35
	69	12, 19
	75	25
	75	25

Tableau 3.7 : *bins trié selon leur remplissage.*

Ensuite, tout les bins mutuellement exclusif sont ajoutés à la nouvelle solution x', et tout les objets restants sont ajoutés à x' selon une heuristique :

Bin	Espace	Liste d'objet
1	0	21, 40, 39
2	3	85, 12
3	3	56, 41
4	5	47, 48
5	5	67, 28
6	5	82, 13
7	5	25, 70
8	5	25, 70
9	2	54, 19, 25

Tableau 3.8 : *Nouvelle solution x'.*

Avec $PE_{x'} = 0{,}13716622$

Selon la formule présentée dans l'algorithme de collision de synthèse, on a :

$$PE_{x1} + KE_{x1} + PE_{x2} + KE_{x2} = 0{,}40604082 + 0{,}59395918 + 0{,}60824811 + 0{,}39175189 = 2 \geq PE_{x'} = 0{,}13716622$$

Puis, un calcul de l'énergie cinétique de cette nouvelle solution est effectué en appliquant la formule présentée dans l'algorithme de collision de synthèse :

$KE_{x'} = (PE_{x1} + KE_{x1} + PE_{x2} + KE_{x2}) - PE_{x'}$

$= 0{,}40604082 + 0{,}59395918 + 0{,}60824811 + 0{,}39175189 - 0{,}13716622 = 1{,}862834.$

✓ *Algorithme de synthèse*

Entrée : Deux molécules x_1 et x_2

Sortie : une nouvelle molécule x'

1 : Mettre tout les bins de x_1 et x_2 dans une liste descendante selon le Remplissage de chaque bin.

2 : Mettre les bins mutuellement exclusifs dans x' selon l'ordre descendant de la liste.

3 : Ajouter les objets restants à x' selon une heuristique.

Algorithme 3.5 : *Algorithme de synthèse*

3.3. Etape d'arrêt

Le processus décrit précédemment dans l'étape d'itération est répété jusqu'à ce qu'une condition d'arrêt soit vérifiée. Dans notre cas, cette condition d'arrêt est définie par un nombre maximum d'itérations.

4. Etude expérimentale:

L'implémentation la métaheuristique CRO sur notre problème Bin Packing uni-dimensionnel est effectuée en Java sous l'environnement de développement Eclipse. Java est un langage qui bénéficie d'une excellente portabilité. L'utilisateur peut ainsi développer un programme sous Windows et l'exécuter dans des environnements et des architectures diverses tels que MacOs, Linux…

Afin de mesurer l'efficacité de CRO pour résoudre des problèmes de type Bin packing en une dimension, nous avons testé notre approche sur des benchmarks proposés dans la littérature qu'on peut trouver sur le site suivant :

(http://www.wiwi.unijena.de/Entscheidung/binpp/). Nous avons comparé notre approche avec les résultats exacts. Pour chaque algorithme, le nombre minimum de bins nécessaire pour le chargement a été enregistré. De plus, Les résultats trouvés dans la littérature pour ces mêmes instances sont aussi pris en compte. Finalement, on a utilisé des tests de Friedman pour comparer statistiquement les résultats trouvés.

4.1.Description des instances utilisées

Notre ensemble d'instances tests fut conçu à partir de celles de Martello and Toth (1990) [25], qui contiennent un nombre d'objets compris entre 50 et 500. Ces instances originales furent utilisées régulièrement afin d'évaluer les méthodes traitantes le problème de bin packing.

Trois classes d'instances sont disponibles, ayant chacune leurs caractéristiques particulières. Notant que les poids d'objets pour toutes les instances, sont dans un ordre décroissant.

- La première classe (Easy) : regroupant 720 instances de degré de difficulté facile.
- La deuxième classe (Medium): regroupant 480 instances de degré de difficulté moyen.
- La troisième classe (Hard): regroupant 10 instances de degré de difficulté difficile.

4.2.Paramètre des expérimentations

Comme nous l'avons décrit dans le chapitre 2, l'algorithme de CRO possède 7 paramètres à initialisés :

- **FELimit :** Nombre maximal d'évaluation de la fonction objectif (Condition d'arrêt).
- **PopSize :** La taille de la liste Pop.
- **KELossRate :** Le pourcentage limité de réduction des KE dans les collisions inefficaces sur le mur.
- **MoleColl :** La fraction de toutes les réactions élémentaires correspondantes à des collisions inefficaces inter-moléculaire.
- **InitialKE :** L'énergie initialisée pour chaque solution (Elle est initialisée à 1- *PE*).
- **α :** Il est utilisé pour le critère de décomposition.
- **β :** Il est utilisé pour le critère de synthèse.

4.3.Résultats et discussion

Afin d'évaluer le potentiel de notre approche, les résultats obtenus ont été comparés aux meilleurs résultats trouvés dans la littérature (Best known). Ces solutions optimales sont rapportées dans [26].

Les valeurs appropriées des paramètres de PopSize, FELimit, KELossRate, MoleColl, alpha et beta sont fixés selon un test préliminaire.

A travers les tests que nous avons fais, notons que les meilleures valeurs adoptées et retenus sont les suivants :

FELimit= 5000

PopSize = 3.

MoleColl = 0,8.

KELossRate = 0,8.

InitKE = 1- *PE*.

α = 1000

β = 0,00001.

Nous avons réalisé des expériences sur ces 3 classes de problèmes existants, les caractéristiques numériques prises en compte sont le nombre de variables, et de contraintes des problèmes. Le nombre de variables désigne le nombre d'objets disponibles, cependant les contraintes désignent les capacités maximums des bins. Les résultats obtenus (le nombre minimum de bins nécessaire pour le chargement) sont récapitulés dans les tableaux suivant où chacun est approprié à une classe de tests. Pour chaque tableau, la première colonne est pour le nom de l'instance, la deuxième colonne contient le nombre d'objets pour cette instance, la troisième contient la capacité du bin, la quatrième colonne contient les résultats obtenus en appliquant l'heuristique (BFD), la cinquième colonne contient les résultats de notre approche CRO, et la dernière colonne contient les meilleurs résultats connus dans la littérature.

4.3.1. Les résultats pour la classe facile de tests :

Le tableau 3.9 ci-dessous résume les résultats trouvés au cours de notre expérience en appliquant l'approche CRO sur la première classe d'instances ayant un degré de difficulté facile.

Ce tableau montre que les résultats obtenus en appliquant notre algorithme CRO sur la classe de test de type facile et les résultats de BFD sont complètement identiques aux meilleures solutions obtenues avec l'application de la méthode exacte.

Instance	*N*	*Cap*	*BFD*	*CRO*	*Best known*
N1C1W1_A	50	100	25	25	25
N1C1W1_D	50	100	28	28	28
N1C1W1_G	50	100	25	25	25
N1C1W1_B	50	100	31	31	31
N1C1W1_E	50	100	26	26	26
N1C1W1_F	50	100	27	27	27
N1C1W1_I	50	100	25	25	25
N2C1W2_P	100	100	68	68	68
N2C1W2_N	100	100	64	64	64
N2C1W2_O	100	100	64	64	64
N2C1W2_R	100	100	67	67	67
N4C1W2_T	500	100	323	323	323
N4C1W4_C	500	100	365	365	365
N4C1W4_A	500	100	368	368	368
N4C1W4_D	500	100	359	359	359
N4C1W4_B	500	100	349	349	349

Tableau 3.9 *. Les résultats obtenus pour la première classe de test.*

4.3.2. Les résultats pour la classe moyenne de test

Dans la deuxième classe de tests moyens, le tableau 3.10 montre que les résultats obtenus avec notre approche sont très proche à des solutions optimales, et sont meilleurs que ceux obtenus en appliquant l'heuristique BFD.

Instance	*N*	*Cap*	*BFD*	*CRO*	*Best Known*
N4W1B1R0	500	1000	184	167	167
N4W1B1R3	500	1000	187	167	167
N4W1B1R6	500	1000	188	169	167
N4W1B1R9	500	1000	188	170	168
N4W1B2R2	500	1000	172	166	164
N4W1B2R5	500	1000	169	163	161
N4W1B2R8	500	1000	175	169	167

N4W1B3R1	500	1000	172	172	171
N4W1B3R4	500	1000	158	158	158
N4W1B3R7	500	1000	163	163	163
N4W2B1R0	500	1000	109	102	101
N4W2B1R3	500	1000	109	101	100
N4W2B1R6	500	1000	109	102	102
N4W2B1R9	500	1000	109	101	101
N4W2B2R2	500	1000	104	103	102
N4W2B2R5	500	1000	104	103	102
N4W2B2R8	500	1000	101	100	100
N4W2B3R1	500	1000	102	102	101
N4W2B3R4	500	1000	100	100	100
N4W2B3R7	500	1000	101	101	101
N4W3B1R0	500	1000	74	71	71
N4W3B1R3	500	1000	74	71	71
N4W3B1R6	500	1000	74	71	71
N4W3B1R9	500	1000	75	72	71
N4W3B2R2	500	1000	72	71	71
N4W3B2R5	500	1000	73	72	72
N4W3B2R8	500	1000	73	72	72
N4W3B3R1	500	1000	71	71	71
N4W3B3R4	500	1000	73	73	73
N4W3B3R7	500	1000	74	74	74
N4W4B1R0	500	1000	58	56	56
N4W4B1R3	500	1000	58	56	56
N4W4B1R6	500	1000	58	56	56
N4W4B1R9	500	1000	57	56	55
N4W4B2R2	500	1000	57	57	57
N4W4B2R5	500	1000	56	55	55
N4W4B2R8	500	1000	56	55	55
N4W4B3R1	500	1000	54	54	54
N4W4B3R4	500	1000	55	55	55
N4W4B3R7	500	1000	57	57	57

Tableau 3.10 : *Les résultats obtenus pour la deuxième classe de test.*

4.3.3. Les résultats pour la classe difficile de test

Le tableau 3.11 expose les résultats obtenus pour la classe de tests qualifiés difficiles, et pour ces résultats, ceux obtenus avec notre approche sont meilleurs que ceux obtenus avec l'heuristique BFD dans tout les cas.

En outre, la différence entre nos résultats et les meilleurs résultats connus est ne dépasse pas qu'un seul bin (dans les cas HARD2, HARD3).

Instance	*N*	*Cap*	*BFD*	*CRO*	*Best Known*
HARD0	200	100000	59	56	56
HARD1	200	100000	60	57	57
HARD2	200	100000	60	57	56
HARD3	200	100000	59	56	55
HARD4	200	100000	60	57	57
HARD5	200	100000	59	56	56
HARD6	200	100000	60	57	57
HARD7	200	100000	59	55	55
HARD8	200	100000	60	57	57
HARD9	200	100000	60	56	56

Tableau 3.11 : *Les résultats obtenus pour la troisième classe de test.*

4.4. Etude comparative

Selon l'étude des paramètres précédente, nous remarquons que le nombre de bin diminue lorsque :

- La valeur de FELimit augmente.
- La valeur de PopSize diminue.
- La valeur de KELossRate augmente.
- La valeur de MoleColl augmente.

Pour toutes les instances testées, on note que la différence entre les solutions trouvées par notre approche et celles trouvées dans la littérature, est entre 0 et 2 (tableau 3.12).

	N4W3B3R7	N4W1B1R6	N4W3B3R4	N4W1B2R5	HARD2	HARD3
Best Known	74	167	73	161	56	55
CRO	74	169	73	163	57	56
Différence	0	2	0	2	1	1

Table 3.12: *Comparaison des solutions CRO avec les solutions optimales connues.*

4.5.Evaluations statistiques des résultats

On a utilisé le test de Friedman pour comparer statistiquement les résultats trouvés par les algorithmes BFD, CRO, et les meilleures solutions connues, et les résultats sont décrits par les figures suivantes :

La figure (3.11) montre que pour la première classe des problèmes de type easy, les solutions obtenues par notre algorithme CRO et les résultats de BFD sont totalement identiques aux solutions optimales connues dans la littérature.

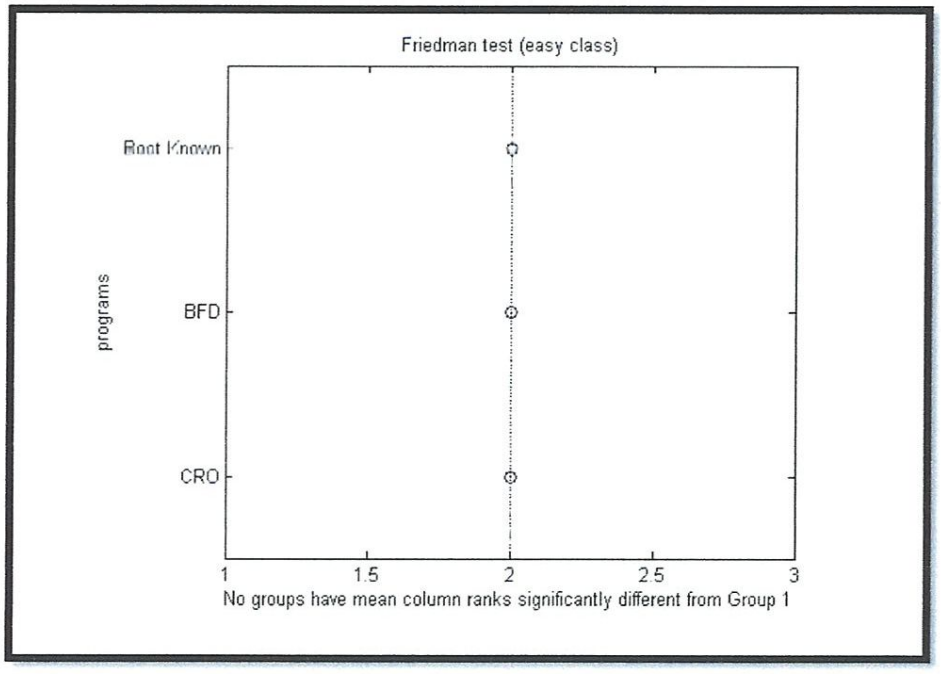

Figure 3.11 : *Test de Friedman pour la série d'instances de type easy.*

La figure (3.12) montre une grande différence entre les solutions obtenues par l'heuristique simple BFD et celles obtenues par CRO pour des instances moyennes, par contre on aperçoit que les résultats de CRO sont très proches aux meilleurs résultats par rapport aux BFD.

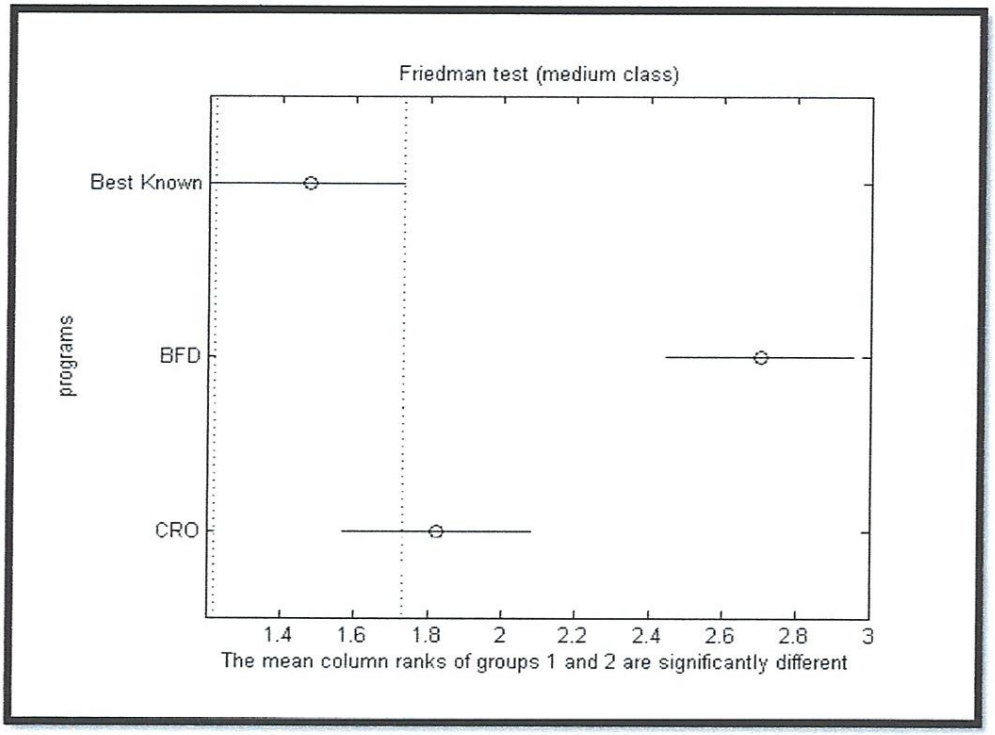

Figure 3.12 : *Test de Friedman pour la série d'instances de type medium.*

La figure (3.13) montre que d'un part, les résultats de CRO pour des instances *hard* semblent très proches aux solutions optimales. Par contre, les résultats trouvés par l'algorithme BFD est très éloignés.

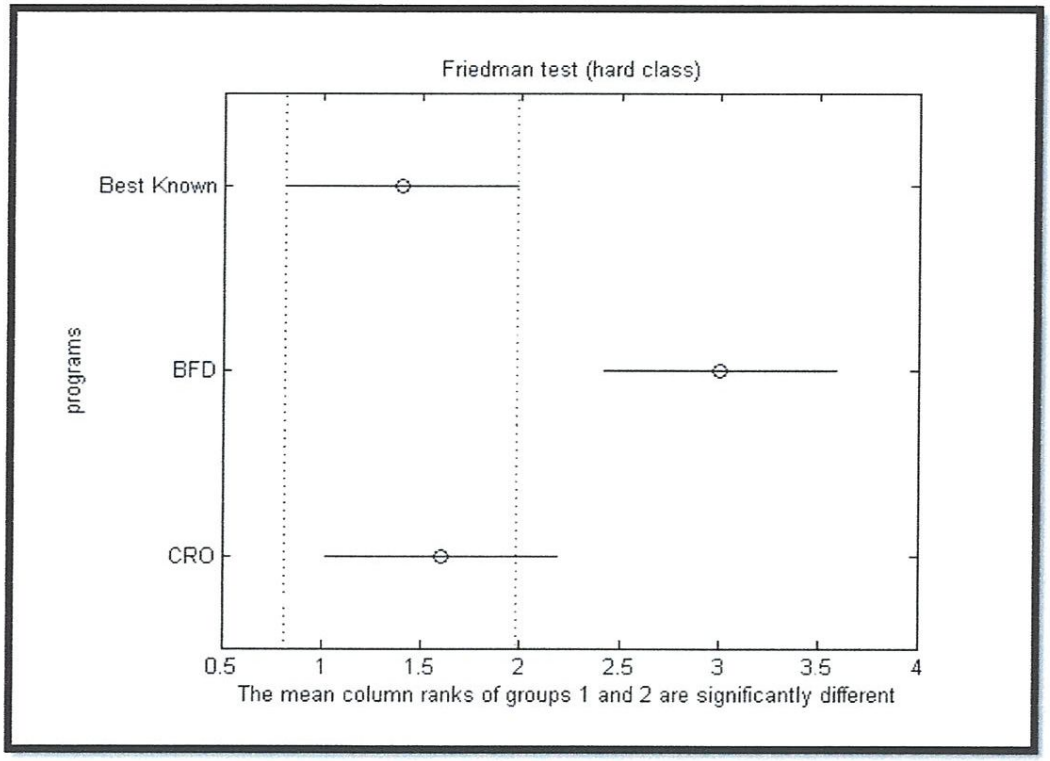

Figure 3.13 : *Test de Friedman pour la série d'instances de type hard.*

La figure 3.14 montre que les résultats de CRO semblent très proches aux solutions optimales pour toutes les classes, et montre l'efficacité de notre approche.

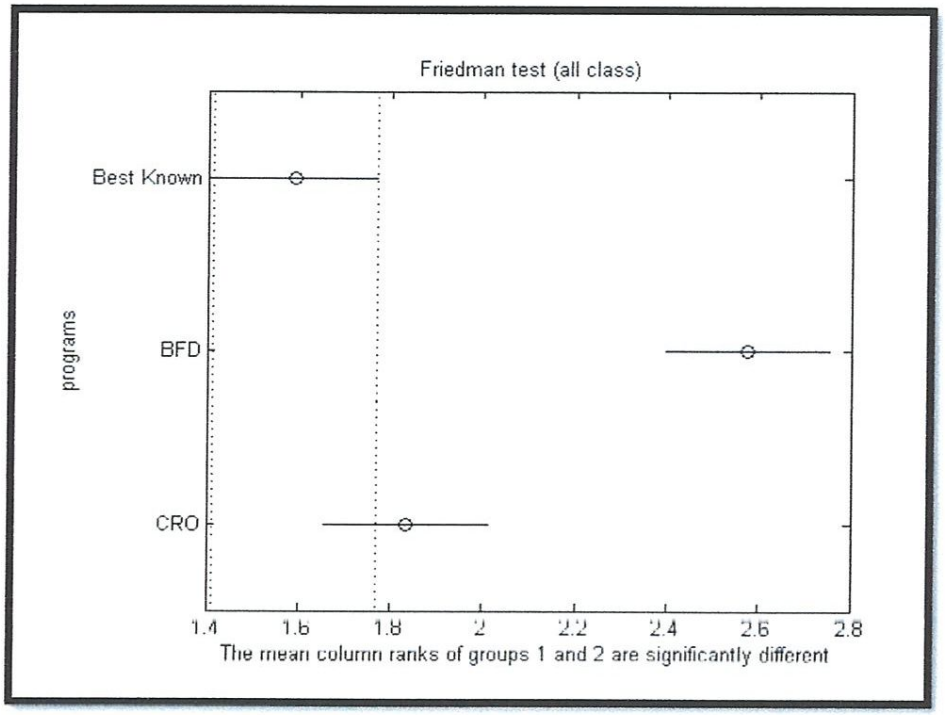

Figure 3.14 : *Test de Friedman pour toutes les classes.*

Conclusion

Dans ce chapitre, nous avons détaillé les différentes réactions élémentaires de la métaheuristique CRO pour la résolution de BPP avec des exemples illustratifs.

Pour l'implémentation, nous avons adopté le langage Java sous l'environnement de développement Eclipse. Pour les expérimentations, nous avons effectué des jeux de tests sur les problèmes Benchmark proposés dans la littérature. Nous avons procédé à une étude comparative par rapport aux Best-Known. Cette étude montre clairement l'efficacité de notre approche.

Conclusion générale

Nous avons traité dans ce mémoire le problème de bin-packing en une dimension. L'importance de ce problème réside dans le domaine vaste de ses applications concrètes en transport, chargement et le monde industriel. Il appartient à la classe des problèmes NP-difficiles. Notre apport personnel et notre contribution sont montrés dans la réalisation d'une méthode métaheuristique optimisation par la réaction chimique « CRO », pour résoudre ce problème.

Dans l'algorithme de CRO, les solutions sont générées itération par itération en passant par les 3 principales étapes de l'algorithme CRO : Initialisation des paramètres, Itérations et le stade final en vérifiant le critère d'arrêt. L'étape d'itération comporte les 4 réactions élémentaires : Collision inefficace sur le mur, collision inefficace inter-moléculaire, collision de décomposition et la collision de synthèse.

Cette approche (CRO) a été testée expérimentalement sur des benchmarks de la littérature rapportés de site (http://www.wiwi.uni-jena.de/Entscheidung/binpp/).Les résultats obtenus sont très encourageants et montrent clairement l'efficacité de notre approche.

Comme perspectives, nous visons à poursuivre les tests sur les autres instances des problèmes benchmark afin d'évaluer davantage de la métaheuristique CRO.

Bibliographie

[1] : Nathalie Klement, Michel Gourgand, Nathalie Grangeon/Problème du bin packing avec incompatibilités: résolution par un couplage hiérarchique métaheuristique heuristique. Université Blaise Pascal,France.

[2] : Joseph El Hayek, Le problème de bin-packing en deux-dimensions, le cas non-orienté : résolution approchée et bornes inférieures, 2007.

[3] : Ben Mohamed Ahmed, résolution approchée du problème de Bin-Packing, 2009.

[4]: P. Gilmore et R. Gomory. A linear programming approach to the cutting stock problem. Ops. Res.,9:849–859, 1961.

[5] : Ali khanafer, thèse : Algorithme pour des problèmes de bin packing mono- et multi – objectif. Université des sciences et technologies de Lille.

[6] : Serial Rayene Boussalia, Mémoire de master: Algorithme Quantique Inspiré de la Recherche Coucou pour le Problème du Chargement de Containers (Bin Packing) Mono et Multi Objectif, Université Mentouri, 25000 Constantine ,2012.

[7] : Sara Chenche, mémoire de master : Une nouvelle métaheuristique basée sur GRASP pour le problème de Bin Packing, Université Mentouri Constantine, 2011.

[8] :Amira Gherboudj, thèse doctorat : Méthodes de résolution de problèmes difficiles Académiques.Université de constantine2,2013.

[9] : Sidi Mohamed Douiri, Souad Elbernoussi, Halima Lakhbab, Cours des Méthodes de Résolution Exactes Heuristiques et Métaheuristiques, Université Mohammed V, Faculté des Sciences de Rabat

[10] : Johanne Cohen, Le problème du Bin Packing (remplissage de sacs).

[11] : Tom Davis, Bin Packing, November 29, 2006.

[12]: N Mohamadi,Application of genetic Algorithm for the bin packing problem with a new representation schema,2010.

[13]: Souquet Amédée ,Radet Francois-Gérard, Algorithme génétique ,2004.

[14] :N.Mohamadi Application de l'algorithme génétique pour l'empaquetage, vol 4, N°. 3(2010) 253-266.

[15] : G.M Weiss.T : a genetic algorithm for predictive patterns in sequences of events. Floria, USA, 1999.

[16]: C.Dhaenens Flipo : Optimisation combinatoire multi-objectif , Apport des méthodes coopératives et contribution à l''extraction de connaissances, Thèse Obtention de grade de 2005

[17] : E.G. Talbi : A Taxonomy of Hybrid Metaheuristics. Journal of Heuristics, Vol. 8, pp. 541-564, 2002.

[18]: Albert Y. S. Lam · Victor O. K. Li, Chemical Reaction Optimization: a tutorial,2012.

[19]: Albert Y.S. Lam, *Student Member, IEEE,* and Victor O.K. Li, *Fellow, IEEE,* Chemical-Reaction-Inspired Metaheuristic for Optimization, JUNE 2010.

[20]: Pan B., Albert Y. S. Lam and Victor O. K. Li. *Network Coding Optimization Based on Chemical Reaction Optimization*. IEEE Communications Society subject matter experts for publication in the IEEE Globecom proceedings, 2011

[21]: Jin Xu, Albert Y. S. Lam and Victor O. K. Li. *Chemical Reaction Optimization for Task Scheduling in Grid Computing*. IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 22, NO. 10, 2011

[22] : James J.Q. Yu, Albert Y.S. Lam and Victor O.K. Li. *Evolutionary Artificial Neural Network Based on Chemical Reaction Optimization*. Browse Conference Publications, 2011

[23]: T.A. FEO, M.G.C. RESENDE, A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters 8: 67-71, 1989.

[24]: Jin Xu, Albert Y. S. Lam and Victor O. K. Li. *Chemical Reaction Optimization for the Grid Scheduling Problem*. IEEE Communications Society subject matter experts for publication in the proceedings, 2010.

[25]: T.A. FEO, M.G.C. RESENDE, A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters 8: 67-71, 1989.

[26]: Martello.S, Monaci.M, Vigo.D :An Exact Approch to the Strip-Packing Problem, Informs Journal on Computing, vol 15, n3, pp 310-319, 2003.