Le recuit simulé (*Simulated annealing*), est une métaheuristique proposée par Kirkpatrick et al. [**?**] en 1983. cette méthode est inspirée du recuit en métallurgie, une technique impliquant le chauffage et le refroidissement contrôlé d'un matériau pour augmenter la taille de ses cristaux et réduire leurs défauts. Les deux sont des attributs du matériau qui dépendent de son énergie thermodynamique. Le chauffage et le refroidissement du matériau affectent à la fois la température et l'énergie libre thermodynamique.

0.1 Pseudocode

Algorithm 1 Recuit simulé

```
Générer une solution initiale S aléatoirement.
Initialiser les paramètres Tinit , T0 et alpha
T=Tinit (T: température courante)
while T<T0 and Time < limite do
  temperature de gel n'est pas atteinte et temps limite n'est pas dépassé
  repeat
    Génerer un voisin S' ∈ V(S)
    if F(S')<= F(S) then
      S=S' //une meilleure solution a été trouvée
    else
      S=S' avec une probabilité P de métropolis
    end if
    if f(S)<Best then
      Best = S //garder dans Best la meilleure solution en terme de nombre de boite utilisées. f(S) étant le nombre de boîtes utilisées par S.
    end if
  until R itérations or R/2 sans amélioration
  T=αT //diminution de la température
end while
```

L'algorithme du recuit simulé pour le problème du bin packing est donné ci-dessus. L'algorithme commence par une température initiale T_{init}, cette température est progressivement diminuée par un facteur α jusqu'à l'atteinte d'un seuil minimal T_0 (Température de Gel). Dans chaque température T, le système (solution actuelle) est perturbé plusieurs fois (R fois). l'algorithme commence par une solution initiale S (affectation des articles aux boîtes),

cette solution est générée aléatoirement, à chaque itération une solution voisine S' est générée aléatoirement en utilisant le processus expliqué dans la partie (*1.1.3 Génération des voisins*) . la valeur de la fonction objective $F(S')$ est calculée. Si le voisin S' a amélioré la valeur de la fonction objective ($F(S') > F(S)$) , la solution S' est acceptée comme la nouvelle solution (S=S') , sinon, elle est acceptée selon la probabilité de métropolis

$$P_{acceptation}(S') = e^{\frac{F(S')-F(S)}{T}}$$

L'algorithme s'arrête quand la température T_0 est atteinte, et retourne la meilleure solution trouvée, dans cette étape on prend la meilleure solution en terme de nombre de boîtes utilisées (gardée dans la variable Best).

0.1.1 Critère d'arrêt

L'algorithme s'arrête quand la température de gel T_0 est atteinte, et dans chaque température T le processus de recherche de voisinage est exécuté R fois.

0.1.2 Fonction objective

La fonction objective est utilisée pour accepter une solution voisine S' = V(S), elle est donnée par la formule suivante [**?**] :

$$\max F(S') = \sum_{i=1}^{m} (\sum_{j=1}^{k_i} w_j)^2 \tag{1}$$

avec :

- m : nombre de boîtes utilisées dans S'
- k_i : nombre d'articles rangés dans la boîte i
- w_j : le volume de l'article j

Cette fonction permet de maximiser le taux de remplissage des boîtes utilisées dans la solution ce qui va nous mener vers une meilleure solution en terme de nombre de boîtes utilisées.

la raison de ne pas avoir utilisé la fonction objective définie dans le problème du bin packing (minimiser le nombre de boîtes utilisées) est que cette dernière donne la même valeur pour plusieurs solutions différentes et ne montre pas les changements au niveau du contenu des boîtes.

0.1.3 Génération des voisins

Pour chercher de nouvelles solutions à partir d'une solution S, l'algorithme utilise l'une des 2 techniques: Swap (0,1) , Swap (1,1).

- Swap(0,1): Consiste à déplacer un article choisi aléatoirement d'une boite à une autre en maximisant la fonction objective (ie augmenter le remplissage des boîtes). Il a été montré que cette technique est efficace dans les hautes températures ($T >= T_{1/2}$) avec $T_{1/2} = \frac{T_{init}+T_0}{2}$

- Swap(1,1): Consiste à permuter entre 2 articles choisis aléatoirement et qui sont dans 2 boîtes différentes, en maximisant la fonction objective (ie augmenter le remplissage des boîtes). Il a été montré que cette technique est efficace dans les basses températures ($T < T_{1/2}$)

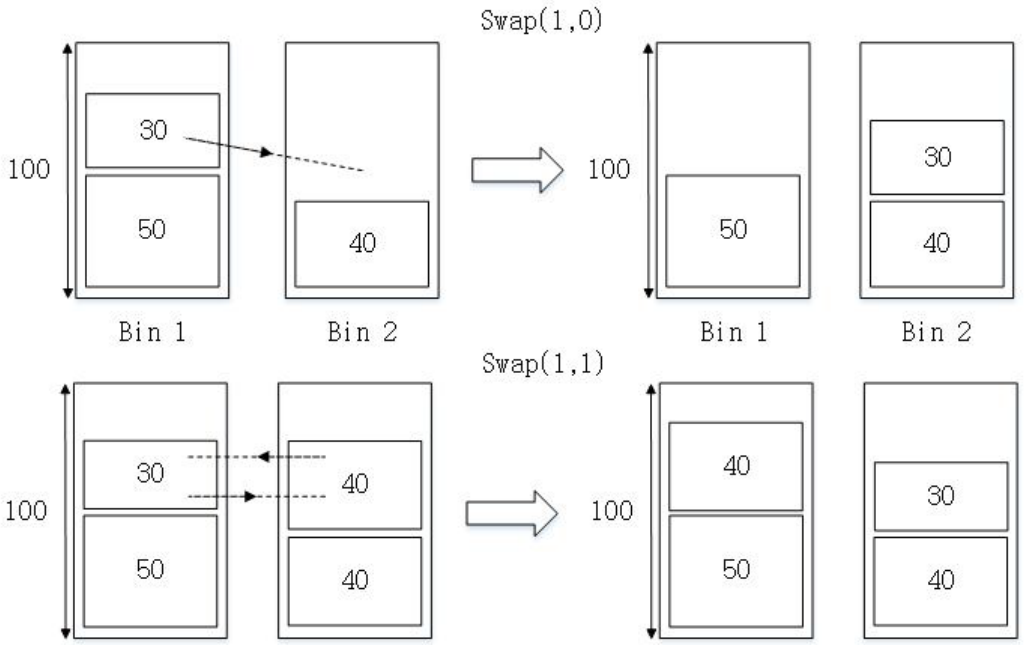

Figure 1: fonctions swap(1,0) et swap (1,1) [**?**]

0.2 Les paramètres du recuit simulé

0.2.1 Le nombre d'itérations R

Signifie le nombre de fois à faire une recherche locale avant de diminuer la température. Dans notre cas cette valeur est fixée par calibrage de paramètres à $R = 1000$. De plus la boucle peut s'arrêter si après $R/2$ itérations on n'obtient aucune amélioration.

0.2.2 La température T

La température T décroît au cours des itérations et influe d'une façon directe sur la probabilité d'acceptation des solutions non améliorantes:

- Premières itérations : T elevée $\Longrightarrow$ acceptation fréquente des solutions non améliorantes (Diversification)
- Dernières itérations : T faible $\Longrightarrow$ acceptation rare des solutions non améliorantes (Intensification)

0.2.3 Valeur de Tinit

La valeur initiale Tinit doit permettre d'accepter initialement la plupart des solutions voisines (Diversification), généralement elle est fixée pour avoir une probabilité d'acceptation p0 de 0.8. La valeur de la température initiale Tinit dépends de la fonction objective et de l'instance du problème. Dans notre cas elle est estimée en effectuant une recherche initiale en acceptant toute les solutions générées, et en calculant la moyenne des différences dans la fonction objective, elle est donnée donc par la formule suivante :

$$T_{init} = \frac{|\Delta F|}{\ln p_0} \tag{2}$$

0.2.4 Valeur de T0

Cette valeur définie le nombre d'itération à effectuer, elle doit être suffisamment petite pour atteindre l'état de gèle, dans notre cas, elle est fixée à 0.1, mais l'algorithme peut s'arrêter si les conditions suivantes sont vérifiées:

- Pas d'amélioration trouvée durant les R itérations d'une température T et la probabilité d'acceptation est assez petite (< 0.01).

- Le temps d'exécution limite est atteint.

0.2.5 Le facteur de diminution de la températures

Ce facteur définit le schéma de refroidissement de notre système, c'est à dire la vitesse de convergence vers la solution finale. sa valeur est généralement entre 0.8 et 1.

- Si α est très grand, la convergence est trop rapide, dans ce cas on aura une convergence prématurée (on reste dans un optimum local).
- Si α est très petit , la convergence est trop lente, dans ce cas on aura une exploration trop importante (temps d'exécution très élevé). Après calibrâge de paramètres, cette valeur est fixée à 0.925.

0.3 Tests et résultats

Dans cette partie on présentera les résultats d'exécution de notre méthode du recuit simulé avec les paramètres défini précédemment $R = 1000, T_0 = 0.1, \alpha = 0925$. Le tableau dans ***figure 2*** montre le temps d'exécution moyen de chaque type d'instance (la moyenne d'exécution de 5 instances de même type) par la métaheuristique recuit simulé.

		Temps d'execution moyen
Classe	N	
Classe 01	50	2.4369
	100	10.5735
	200	42.7212
	500	356.9223
Classe 02	50	0.7101
	100	3.0479
	200	12.2010
	500	154.0767
Classe 03 HARD	200	28.3268

Figure 2: tableau des temps d'exécution moyens du recuit simulé

L'histogramme ***figure*** **3** présente les temps d'exécutions du recuit simulé en fonction des instances.

Figure 3: histogramme des temps d'execution du recuit simulé

La ***figure*** **4** compare le nombre de boîtes obtenu par le recuit simulé avec la solution optimale

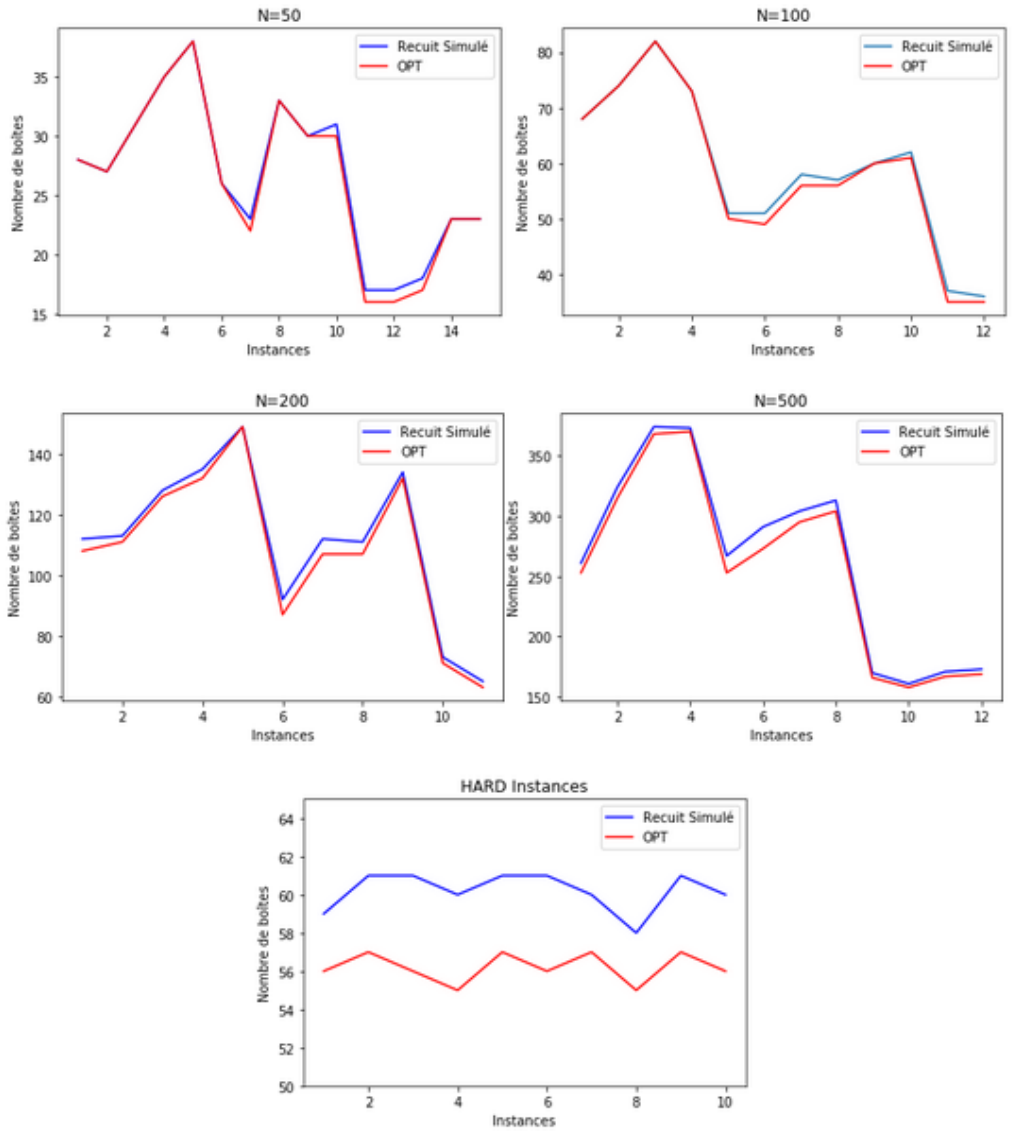

Figure 4: le nombre de boites obtenues par le recuit simulé par rapport à la solution optimale

Dans le tableau et le graphe ***figure*** 5on donne la qualité de solution obtenue par le recuit simulé en utilisant la métrique du worse case ratio.

Classe	N	Ratio
		RS
Classe 01	50	1.0625
	100	1.0571
	200	1.0574
	500	1.0553
Classe 02	50	1.0
	100	1.0714
	200	1.034
	500	1.0
Classe 03 (HARD)	200	01.09

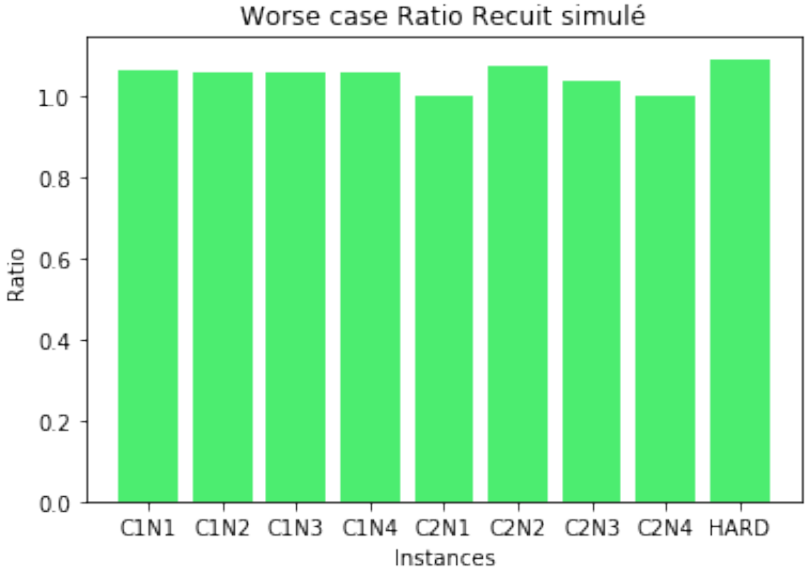

Figure 5: worse case ratio du recuit simulé

0.3.1 Analyse et interprétation des résultats

- Le temps d'exécution du recuit simulé est très élevé, même avec les plus petites instances.
- Ce temps d'exécution augmente avec la taille du problème et varie de *0.7s* pour N=50 (classe 2) , jusqu'à *357s* pour N=500 (classe1),[***figure***

2 , ***figure*** **3**].

- le temps d'exécution correspond à l'exécution 10 fois du Recuit simulé, ceci était nécessaire à cause de la nature stochastique de l'algorithme, la solution initiale générée aléatoirement et la variable aléatoire u utilisée pour accepter la solution non améliorante, influent sur les résultats obtenus.

- En terme de qualité de solution, le recuit simulé est capable de délivrer une bonne qualité de résultats pour le Scholl Benchmark, il arrive souvent à trouver la solution optimale pour les 2 premières classes du benchmark, et un peu moins pour la 3ème classe. [***figure*** **4**]

- Le Ratio obtenu par le recuit simulé est proche de 1 dans la plupart des cas, sauf dans le cas N=4 de la 1ere classe (C1N4), où on remarque une petite augmentation du ratio.

- Cela signifie qu'on a pas une grande déviation par rapport à la solution optimale, donc d'une façon générale, le recuit simulé fournit une bonne qualité de résultats même dans le pire des cas (les instances les plus difficiles), avec une qualité un peu moins bonne dans les instances C1N4.On justifie cette dégradation par la taille de ces instances (N4=500 objets), le temps limité accordé au recuit simulé n'est pas suffisant pour lui permettre de bien explorer l'espace de recherche.