Chapitre I

Les métaheuristiques pour la résolution des problèmes d’optimisation

Chapitre I : les métaheuristiques pour la résolution des problèmes d'optimisation

1.1 Introduction

Un problème d'optimisation combinatoire est défini par un ensemble d'instances.A chaque instance du problème est associé un ensemble discret de solutions S, un sous-ensemble X de S représentant les solutions admissibles (réalisables) et une fonction de coût f (ou fonction objectif) qui assigne à chaque solution s ∈ X lenombre réel (ou entier) f(s). Résoudre un tel problème (plus précisément une telleinstance du problème) consiste à trouver une solution s* ∈ X optimisant la valeur dela fonction de coût f. Une telle solution s* s'appelle une solution optimale ou un optimum global[PS82].

Nous avons donc la définition suivante :

Définition:Une instance I d'un problème de minimisation est un couple (X, f) où X ⊆ S est un ensemble fini de solutions admissibles, et f une fonction de coût (ou objectif) à minimiser f : X → R. Le problème est de trouver s* ∈ X tel que f(s*) ≤ f(s) pour tout élément s ∈ X[PS82].

Notons que d'une manière similaire, on peut également définir les problèmes de maximisation en remplaçant simplement ≤ par ≥. L'optimisation combinatoire trouve des applications dans des domaines aussi variés que la gestion, l'ingénierie, la conception, la production, les télécommunications, les transports, l'énergie, les sciences sociales et l'informatique elle-même.

1.2 Les métaheuristiques pour la résolution

Il existe deux grandes catégories des méthodes de résolution des problèmes d'optimisation combinatoires : les méthodes exactes et les méthodes approchées. Les méthodes exactes se basent sur l'énumération de l'ensemble des solutions potentielles et permettent d'obtenir la solution optimale on a par exemple l'algorithme de séparation et évaluation (branche and bound) et la programmation dynamique, la taille du problème influe rationnellement sur l'efficacité (temps d'exécution) des méthodes de cette approche, par contre on a les méthodes approchées, encore appelées heuristiques, qui permettent d'obtenir une bonne solution, qui n'est pas toujours la solution optimale, mais avec un temps raisonnable.

Feignebaum et Feldman (1963) définissent une **heuristique** comme une règle d'estimation, une stratégie, une astuce, une simplification, ou tout autre sorte de système qui limite drastiquement la recherche des solutions dans l'espace des solutions possibles. Newell, Shaw et Simon (1957) précisent qu'un processus heuristique peut résoudre un problème donné, mais n'offre pas la garantie de le faire.

Dans la pratique, certaines heuristiques sont connues et ciblées sur un problème particulier.

Il existe d'autres méthodes qui se placent à un niveau plus général, et interviennent dans toutes les situations où l'ingénieur ne connaît pas d'heuristique efficace pour résoudre un problème donné,ces méthodes sont les **métaheuristique**s.

En 1996, I.H. Osman et G. Laporte définissaient la métaheuristique comme « un processus itératif qui subordonne et qui guide une heuristique, en combinant intelligemment plusieurs concepts pour explorer et exploiter tout l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou presque-optimales ».

En 2006, le réseau Metaheuristic (metaheuristics.org) définit les métaheuristiques comme « un ensemble de concepts utilisés pour définir des méthodes heuristiques, pouvant être appliqués à une grande variété de problemes.

Comme on voir la metaheuristique comme une « boîte à outils » algorithmique, utilisable pour résoudre différents problèmes d'optimisation, et ne nécessitant que peu de modifications pour qu'elle puisse s'adapter à un problème particulier ».

Elles ontdonc pour objectif de pouvoir être programmées et testéesrapidement sur un problème [BA06].

1.2.1 Classification

Il existe plusieurs manières pour classifier les métaheuristiques selon :

- La manipulation d'une solution à la fois ou un ensemble des solutions dites population.
- Utilisation de l'historique de la recherche (mémoire).
- Inspiration de la nature ou non.

Il existe trois classes principales :

- Les méthodes constructives : construisent une solution progressivement en prenant à chaque étapeune décision (par exemple, prochaine ville à visiter dans la tournée d'unvoyageur de commerce) qui ne sera jamais remise en cause.
- Méthodes de recherche locale : méthodes d'amélioration itératives qui se déplacent pas à pas dansle « voisinage » de la solution couranteen cherchant des « pistes prometteuses » vers la solution optimale.
- Méthodes évolutives : manipulent un groupe de solutions admissibles à chaque étape derecherche plutôt qu'une seule solution, s'inspirent de phénomènesnaturels (algorithmes génétiques, fourmis…).

1.2.2 Les méthodes de recherche locale

Le principe de la recherche locale est le suivant : l'algorithme débute avec une solution initiale realizable.Sur cette solution initiale, on applique une série de modifications locales (définissant un voisinage de la solution courante), tant que celles-ci améliorent la qualité de la fonction objectif. [HS05]

La figure1illustre bien le fonctionnement de l'algorithme général des méthodes de recherche locale.

Le passage d'unesolution ver une autre se fait grâce à la définition de structure de voisinage qui est un élément très important dans la définition de ce type de méthode. Le voisinage d'une solution est défini en fonction du problème à résoudre.

On définit l'espace de recherche comme l'espace dans lequel la recherche locale s'effectue. Cet espace peut correspondre à l'espace des solutions possibles du problème étudié. Habituellement, chaque solution candidate a plus d'une solution voisine, le choix de celle vers laquelle se déplacer est pris en utilisant seulement l'information sur les solutions voisines de la solution courante, d'où le terme de recherche locale [AK97].

D'une manièreabstraite, une recherche locale peut se résumer de la façon suivante :

1. Démarrer avec une solution.
2. Améliorer la solution.
3. Répéter le processus jusqu'à la satisfaction des critères d'arrêt.

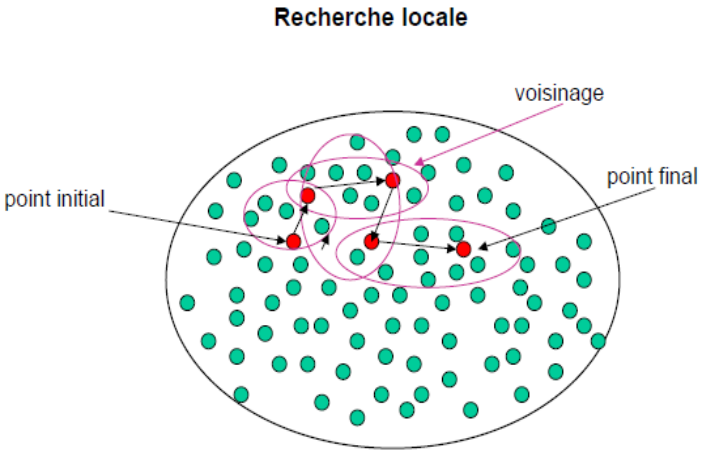

Figure 1 : Représentation de la procédure générale de la recherche locale.

1.2.2.1 La descente récursive

Le principe de la méthode de descente (dite aussi basic local search) consiste à partir d'une solution s on choisit une solution s' qui appartient au voisinage de s, telle que s' améliore la recherche (généralement telle que f(s') < f(s)).

```
Procédure descente_simple (solution initiale s)
Répéter :
        Choisir s' dans N(s)
        Si f(s') < f(s)
                alors s ← s'
Jusqu'à ce que f(s') ≥ f(s),  s'∈S
Fin.
```

Pseudo-coude de la décante récursive.

On peut décider : soit d'examiner toutes les solutions du voisinage et prendre la meilleure de toutes (ou prendre la première trouvée), soit d'examiner un sous-ensemble du voisinage.

La méthode de descente est la méthode de recherche locale la plus élémentaire. On peut la schématiser comme suit :

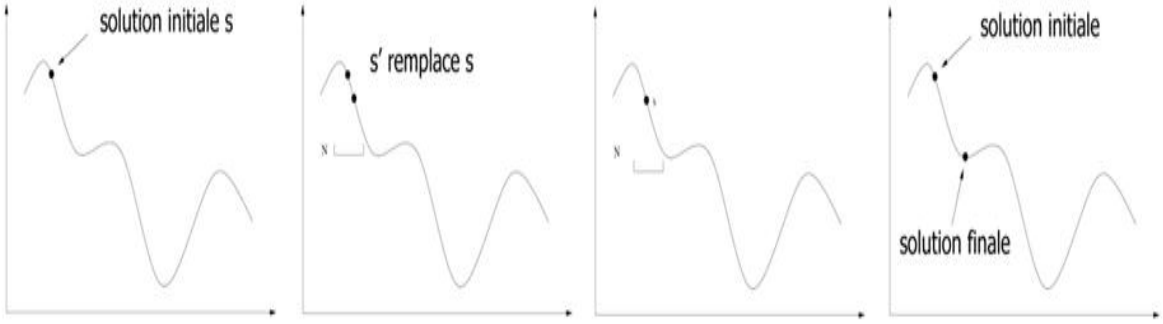

Figure 2 : évolution d'une solution dans la méthode de descente.

En général, l'efficacité des méthodes de descente simples est très peu satisfaisante. D'abord, par définition, la recherche s'arrête au premier minimum local rencontré, c'est là leur principal défaut. Pour améliorer les résultats, on peut lancer plusieurs fois l'algorithme en partant d'un jeu de solutions initiales différentes, mais la performance de cette technique décroît rapidement.

1.2.2.2 Le recuit simulé

La méthode du Recuit Simulé (RS) est une technique de recherche locale inspirée du processus utilisé en métallurgie. Ce processus alterne des cycles de refroidissement lents et de réchauffage ou de recuit qui tendent à minimiser l'énergie du matériel. Le recuit simulé s'appuie sur l'algorithme de Metropolis-Hastings, qui permet de décrire l'évolution d'un système thermodynamique. Par analogie avec le processus physique, la fonction à minimiser deviendra l'énergie E du système. On introduit également un paramètre fictif, la température T du système.

L'algorithme du recuit simulé part d'une solution donnée, et la modifie itérativement jusqu'au refroidissement du Système. Les solutions trouvées peuvent améliorer le critère que l'on cherche à optimiser, on dit alors qu'on a fait baisser l'énergie du système, comme elles peuvent le dégrader. Si on accepte une solution qui améliore le critère, on tend ainsi à chercher l'optimum dans le voisinage de la solution de départ. Contrairement autres méthodes de recherche locale, le recuit simulé peut accepter des solutions dont la qualité est moins bonne en fonction de la dégradation de la solution considérée [AK89].

Algorithme

```
Choisir une solution initiale S
Choisir une température initiale Ti > 0
Choisir une température finale Tf > 0 // Tf <Ti
Choisir un nombre d'itération NB
(à une température Fairennée)
Choisir le coefficient de diminution de latempérature F ∈ [0,1[
    T <- T_{i}
    TQ (Tf < T) faire
        Pour (k = 1 à NB)
            S' <- voisin aléatoire de S
            Δ <- f(S') - f(S)
            Si (Δ≤ 0) alors
                S <- S'
            Si non
            S <- S' avec la probabilité e-Δ/T
            Fin si
        Fin pour
    T <- T *
    Fin TQ
```

Pseudo code de la méthode du recuit simulé.

Explication :

Initialement on donne le système une très haute température puis on le refroidit petit à petit Le refroidissement du système doit se faire très lentement pour avoir l'assurance d'atteindre un état d'équilibre à chaque température T.

Le voisinage N(s) d'une solution s. s'appartient à l'ensemble des états atteignables depuis l'état courant en faisant subir des déplacements élémentaires aux atomes du système physique.

A chaque itération, une seule solution voisine s'est générée et elle est acceptée si elle est meilleure que la solution courante s dans le cas contraire on a les cas :

- si T grande, $\exp^{(-\Delta E/T)}$) est de l'ordre de 1, on garde toujours le mouvement, même il est mauvais.

- si T très petit, $\exp^{(-\Delta E/T)}$) est de l'ordre de 0, donc les mouvements qui augmentent l'énergie (la différence) sont disqualifiés.

Méthode : on tire au hasard dans l'intervalle [0,1[, si le nombre est <exp(-ΔE/T) , on garde sinon on jette.

La meilleure solution trouvée est mémorisée dans la variable s*.

La méthode recuite simulée donne des très bons résultats pour le problème de voyageur de commerce mais pas pour le problème d'affectation ; leur performance est liée fortement au schéma de refroidissement [Wid01].

Les avantages de recuit simulé:

Parmi les avantages de la méthode on cite

- Traite les fonctions de coût avec les degrés tout à fait arbitraires de non linéarité, la discontinuité, et l'imprévisibilité.
- Processus assez arbitraire sur les conditions limites et les contraintes imposées sur les fonctions de coût.
- Simple à implémenter.
- Garantie Statistique de trouver une solution optimale.

Les inconvénients de recuit simulé

La méthode comprend quelques inconvénients parmi lesquelles :

- le non déterminisme.
- Difficulté de choisir les paramètres efficaces ; par exemple le schéma de refroidissement.
- Le compromis entre la vitesse et l'optimisation.

1.2.2.3 La méthode Tabou

La méthode Tabou est une méthode de recherche dont les principes ont été proposés pour la première fois par Fred Glover dans les années 80, et elle est devenue très classique en optimisation combinatoire. Elle se distingue des méthodes de recherche locale simples par le

recours à un historique des solutions visitées, de façon à rendre la recherche un peu moins« aveugle ».pour éviter de retomber périodiquement dans un minimum local, certaines solutions sont bannies, elles sont rendues « taboues ».

A l'inverse du recuit simulé qui génère de manière aléatoire une seule solution voisine $s' \in N(s)$ à chaque itération, Tabou examine un échantillonnage de solutions de *N(s)* et retient la meilleure *s'* même si *f(s')>f(s)*. La recherche Tabou ne s'arrête donc pas au premier optimum trouvé.

Le danger serait alors de revenir à *s* immédiatement, puisque *s* est meilleure que *s'*. Pour éviter de tourner ainsi en rond, on crée une liste *T* qui mémorise les dernières solutions visitées et qui interdit tout déplacement vers une solution de cette liste. Cette liste *T* est appelée liste Tabou[BaP06].

1.2.3 Les méthodes évolutives

Avec les algorithmes évolutionnaires, nous passons à une autre catégorie des métaheuristiques, celles des méthodes dites évolutionnaires, qui manipulent un ensemble des solutions simultanément.

On distingue deux approches utilisées par les méthodes basées sur une population de solutions : la première est l'utilisation des mécanismes d'évolution naturels, cette approche est représentée par les AGs, EV, et autres. La deuxième est l'utilisation de l'intelligence collective, cette approche est représentée par ACO, PSO et autres.

Dans les deux sections suivantes, on va donner un exemple de chacune de ces stratégies qui sont les AGs et l'ACO.

1.2.3.1 Les Algorithmes Génétiques

Un AG est une méthode d'optimisation dans laquelle un ensemble appelé population, de solutions potentielles, appelées individus, est progressivement mis à jour par le biais d'un mécanisme de sélection et par des opérations génétiques : le croisement et la mutation.

Déroulement

Dans les algorithmes génétiques, on essaye de simuler le processus d'évolution d'une population. Au début on génère une population de solutions d'une manière aléatoire, cette population forme la population initiale. Le degré d'adaptation de chaque individu à l'environnement (exprimé par la valeur de la fonction coût dite fonction fitness) est calculé.

Après, des couples des individus P1 et P2 (appelés parents) sont sélectionnés en fonction de leurs adaptations ; ensuite un opérateur de croisement est appliqué sur P1 et P2 avec une probabilité Pc et génère des couples C1 et C2 (dites enfants). D'autres individus P sont sélectionnés en fonction de leur adaptation ; un opérateur de mutation est appliqué sur P avec une probabilité Pm (Pm est généralement très inférieur à Pc) et génère des individus mutés P0.

Le niveau d'adaptation des enfants (C1, C2) et des individus mutés P0 sont ensuite évalués avant de les insérer dans la nouvelle population ; l'insertion est faite par le choix de N individus parmi la population des parents et la population des enfants. En itérant ce processus jusqu'à un critère d'arrêt est atteint. Un critère d'arrêt peut être un nombre fixe d'itérations ou lorsque la population n'évolue d'une manière plus ou moins suffisamment rapide [Dur04].La figure (3) montre ce déroulement.

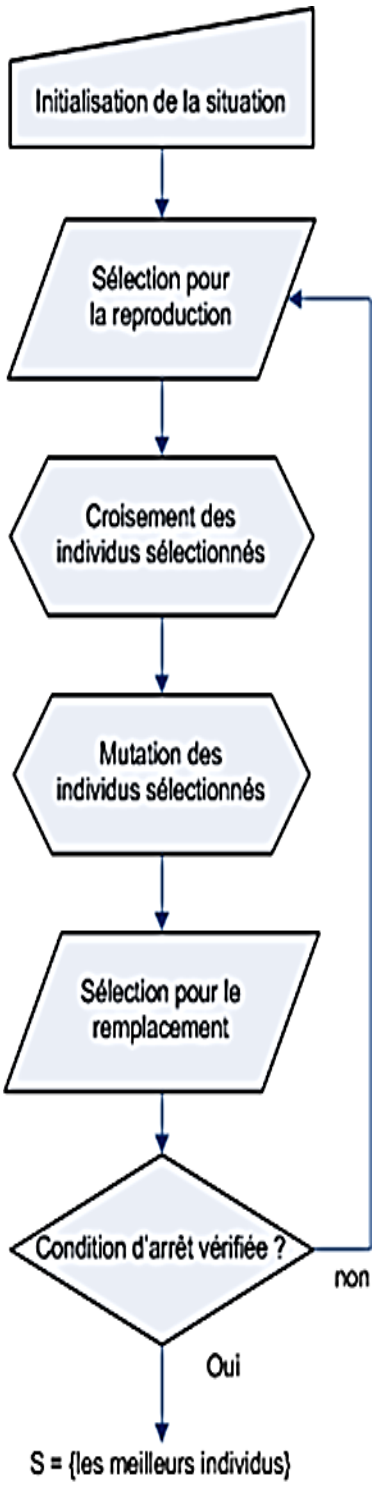

Figure 3 : principe de fonctionnement d'un algorithme génétique.

Principe de l'algorithme génétique

L'algorithme génétique repose sur une boucle qui enchaîne des étapes de *sélections* et des étapes de *croisements*. Dans un premier temps, à partir d'une population de α individus, on désigne ceux autorisés à se reproduire.

On croise ensuite ces derniers, de façon à obtenir une population d'enfants, dont on peut faire muter aléatoirement certains gènes.

La performance des enfants est évaluée, grâce à la fonction *fitness*, et l'on désigne, dans la population totale résultante parents+enfants, les individus autorisés à survivre, de telle manière que l'on puisse repartir d'une nouvelle population de α individus.

La boucle est bouclée, et l'on recommence une phase de sélection pour la reproduction, une phase de mutation, et ainsi de suite.

Comme pour les métaheuristiques vues précédemment, un critère d'arrêt permet de sortir de la boucle, par exemple un certain nombre d'itérations sans amélioration notable de la performance des individus [BA06].

Codage des chromosomes

Les individus de la population doivent être codés selon une représentation spécifique. Le choix d'adopter un tel codage ou un autre est une question qui dépend des caractéristiques du problème posé. Chaque paramètre d'une solution du problème traité est assimilé à un gène.

Parmi les techniques les plus fréquemment utilisées pour coder les individus, on distingue :

Le codage en binaire :

Dans ce type de codage, pour un individu on code ses variables et on les concatène par exemple, la chaîne binaire 1000| 0110| 1101, correspond à un individu défini par 3 variables (8, 6, 13) en codage binaire naturel sur 4 bits chacune.

C'est le codage le plus utilisé pour plusieurs raisons : pour des raisons historiques, ce codage a été utilisé par J. Holland et ses étudiants ; plusieurs résultats théoriques sont basés sur ce codage, et il est facile de mettre en place les opérateurs génétiques avec ce codage.Cependant si la longueur de la chaîne augmente la performance de l'algorithme diminuera.

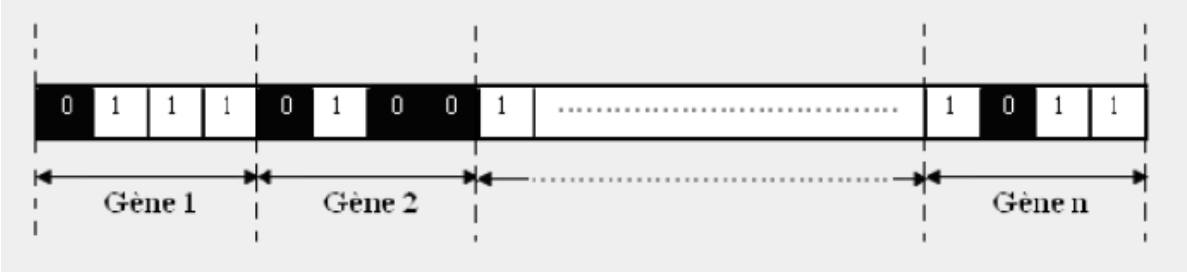

Figure 4 : Structure d'un chromosome en codage binaire.

Le codage réel

Il sagaie de concaténation des variables ***xi*** d'un individu ***x***. par exemple un individu ***x*** (25, 31, 8) est codé 25| 31| 8 ; ce codage est plus précis que le codage binaire, l'espace de

recherche est le même que l'espace du problème, l'évaluation de la fonction coût est plus rapide ; mais son alphabet est infini et il a besoin d'opérateurs appropriés.

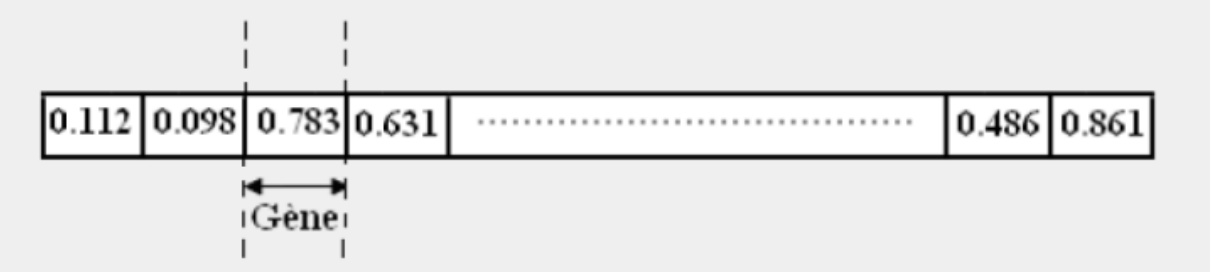

Figure 5 : Structure d'un chromosome en codage en nombres réels**.**

Les opérateurs génétiques

Cette opération symbolise dans l'algorithme génétique la reproduction.

La sélection :

Une fois réalisée l'évaluation de la génération, on opère une sélection à partir des valeurs d'adaptation. Seul les individus passant l'épreuve de sélection peuvent se reproduire. Notons que les étapes de sélection et de remplacement sont indépendantes de l'espace de recherche. On distingue deux types de sélection :

Sélection déterministe

Seuls les meilleurs individus seront toujours sélectionnés, les autres sont totalement écartés.

Sélection stochastique

Les meilleurs individus sont toujours favorisés mais de manière stochastique ce qui laisse une chance aux individus moins adaptés pour participer à former la prochaine génération.

Le croisement :

On distingue les opérateurs de croisement, qui génèrent de nouveaux individus à partir de plusieurs (un couple, le plus souvent), et les opérateurs de mutation, qui transforment un seul individu.

Il n'y a pas de règle absolue pour construire des opérateurs de croisement. Il est possible d'apparier au hasard, mais en général, on préférera définir une distance dans l'espace de recherche, et combiner les individus proches selon cette distance. Par exemple, dans un algorithme génétique appliqué à un problème de régulation de transport, on associera ensemble des individus générant peu de conflits, ou le moins de retards.

Pour pouvoir croiser, il faut déjà représenter les solutions.

les solutions du problème sont comparables à des chromosomes, et sont constituées d'une série de gènes, chaque gène étant associée à une séquence de bits, à une chaîne binaire codée en fonction d'une règle de conversion particulière. Croiser des individus consiste alors à « croiser » les séquences binaires de deux parents, pour former les « génomes » des enfants [BA06].

Il existe plusieurs façons pour réaliser un croisement entre deux chromosomes :

Croisement en point

On choisit au hasard un point pour couper les deux chromosomes de chaque couple puis on échange les fragments situés après le point de coupure permettant la création de deux nouveaux génotypes. Il est à noter ici que le croisement peut également ne pas s'effectuer au niveau des gènes ce qui rend possible à un chromosome d'être coupé au milieu d'un gène.

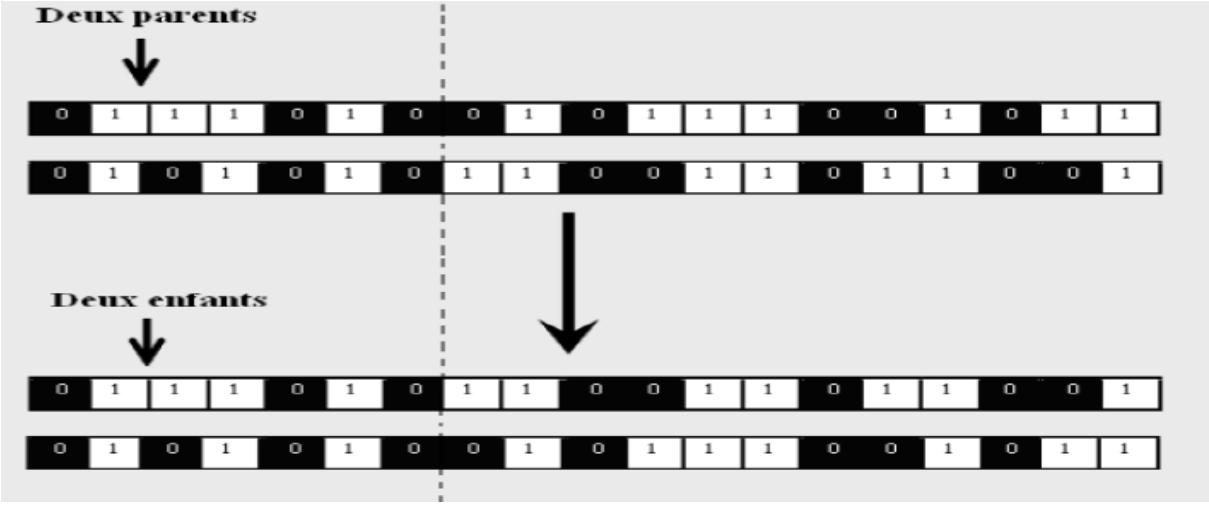

Figure 6 : Croisement 1-point.

Croisement en deux points

On choisit au hasard deux points de croisement puis on échange les fragments situés entre ces deux points. Cette définition peut également se généraliser pour effectuer un croisement à n points.

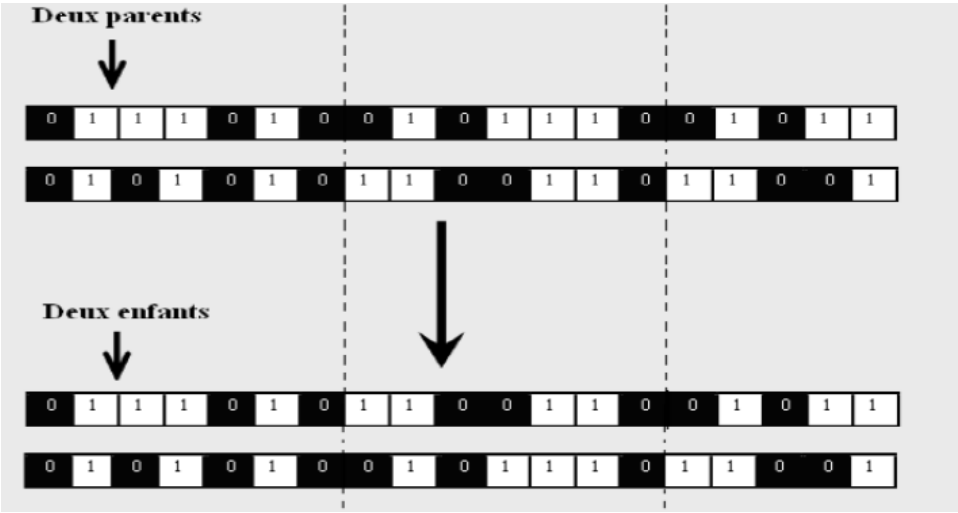

Figure 7 : Croisement 2-points.

La mutation :

La mutation est un mécanisme qui protège les algorithmes génétiques contre les pertes prématurées d'informations pertinentes ces informations qui ont pu être perdues lors de l'opération de croisement ; la mutation participe aussi au maintien de la diversité(pour l'exploration des nouvelles régions). Il y a une autre méthode pour faire la mutation consiste à changer la valeur d'un bit de 1 vers 0 par exemple.

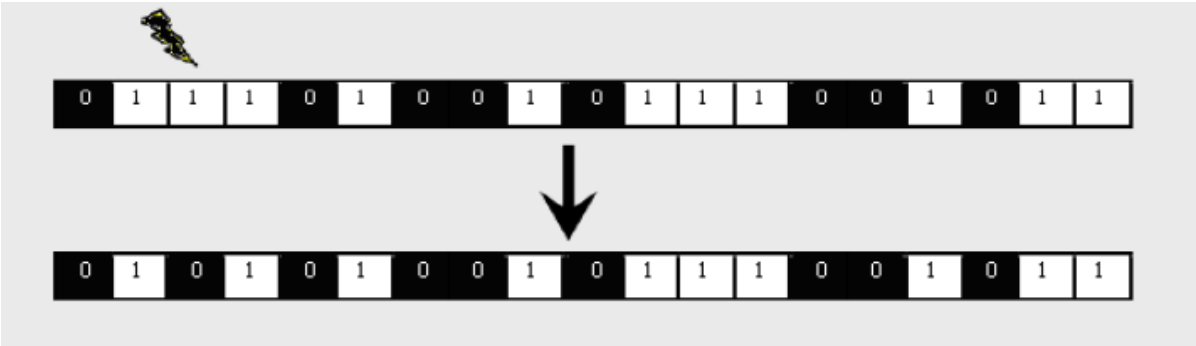

Figure 8 : Mutation.

Les avantages des algorithmes génétiques

Les algorithmes génétiques ont plusieurs avantages citons:

- Ils sont adaptables aux plusieurs types des problèmes.
- Robustes.
- Facile à implémenter.

- Facile à hybrider.
- Facile à paralléliser.

Les inconvénients des algorithmes génétiques:

Parmi les inconvénients des algorithmes génétiques on trouve:

- Pas de garantie de convergence.
- Temps de calcule important (si la taille de population est grande).

1.2.3.2 L'optimisation par colonies de fourmis

L'optimisation par colonies de fourmis ou Ant Colony Optimization (ACO) est une méthode d'optimisation basée sur la manipulation d'un ensemble de solutions, cette méthode représente toute une classe des métaheuristiques qui reposent sur la notion de l'intelligence collective. La solution finale est plus complexe que celle d'un composant simple. Plusieurs mots apparaissent dans ce domaine : l'auto-organisation, émergence et autres.

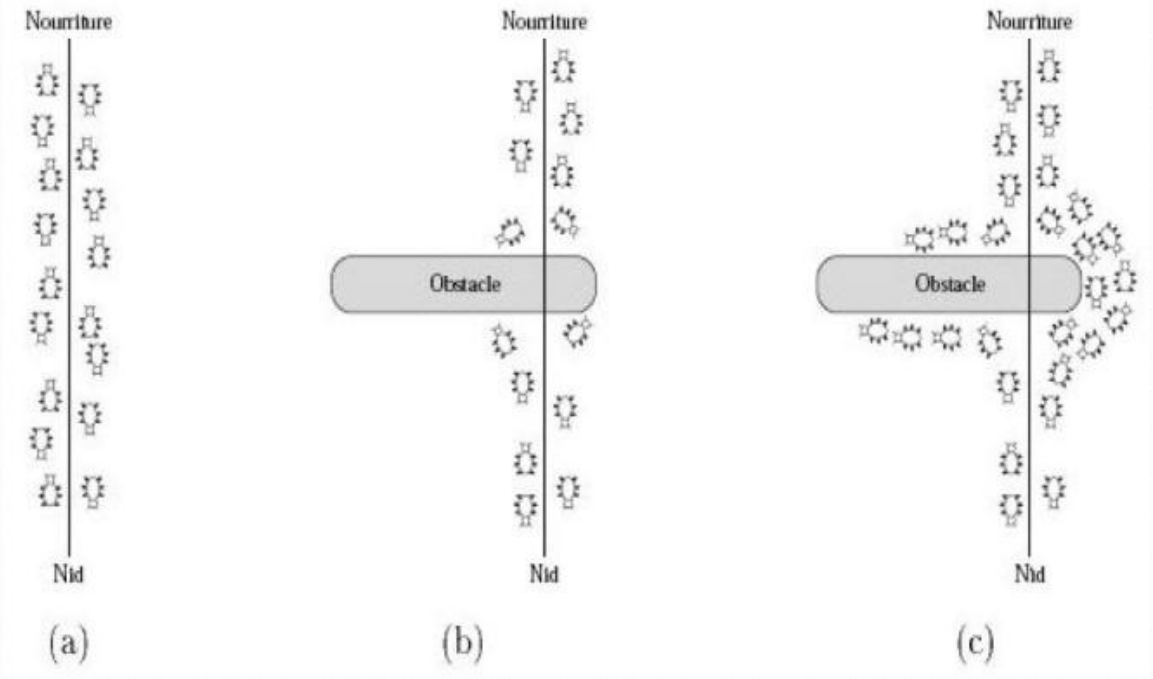

Figure 9: l'influence de l'expérience sur le choix des fourmis.

L'origine de la méthode

Les fourmis ont la particularité d'employer pour communiquer des substances volatiles appelées phéromones. Elles sont très sensibles à ces substances, qu'elles perçoivent

grâce à des récepteurs situés dans leurs antennes. Ces substances sont nombreuses et varient selon les espèces. Les fourmis peuvent déposer des phéromones au sol, grâce à une glande située dans leur abdomen, et former ainsi des pistes odorantes, qui pourront être suivies par leurs congénères [SDPT03].

Les fourmis utilisent les pistes de phéromones pour marquer leur trajet, par exemple entre le nid et une source de nourriture. Une colonie est ainsi capable de choisir (sous certaines conditions) le plus court chemin vers une source à exploiter, sans que les individus aient une vision globale du trajet [SDPT03].

Description et algorithme

Pour bien comprendre comment fonctionne l'optimisation par colonie de fourmis, on va retourner vers le premier algorithme proposé, c'est l'algorithme « Ant System » proposé par Colorni et autres en 1992 ; au début chaque fourmi est mise aléatoirement sur une ville et elle a une mémoire qui stocke la solution partielle qu'elle a construit jusqu'ici (au commencement la mémoire contient seulement la ville du début). À partir de sa ville de début, une fourmi se déplace itérativement d'une ville vers une autre mais avec une règle de probabilité. Étant donné à une ville i une fourmi k choisit d'aller à la ville j avec une probabilité donnée près :

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k} [\tau_{il}(t)]^{\alpha} * [\eta_{il}]^{\beta}} \qquad \text{if } j \in N_i^k$$

Où $\eta_{ij} = 1/d_{ij}$ est l'information heuristique a priori disponible, et N_i^k est l'ensemble de villes que la fourmi k n'a pas encore visité, cet ensemble forme le voisinage faisable de la fourmi k. α et β sont deux paramètres qui déterminent le degré d'influence de la densité de phéromone et de l'information heuristique sur le choix de la prochaine ville.

Si α = 0, les probabilités de choix sont proportionnelles à [le β d'ij de] et les villes les plus proches seront plus probablement choisies: dans ce cas AS correspond à un algorithme glouton stochastique classique (avec multi-départ puisque les fourmis au commencement sont aléatoirement distribuées sur les villes).

Si le β = 0, seulement l'amplification de phéromone est au travail: ceci mènera à l'apparition rapide d'une convergence prématurée vers un optimum local [DS00].

Lorsque chaque fourmi termine la construction d'un chemin complet (de longueur n) ; on passera à la phase de mise à jour de la phéromone. Cette dernière se décompose en deux parties, la première consiste à baisser la densité de phéromone (c'est l'évaporation de la phéromone), la deuxième permet à chaque fourmi de déposer la phéromone sur les arcs qui appartiennent à son excursion:

$$\tau_{ij}(t+1) = (1-p) * \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}(t) \ \forall(i,j) \qquad (2)$$

Où 0 < ρ = 1 est le taux d'évaporation de la phéromone et le m est le nombre de fourmis. Le paramètre ρ est employé pour éviter l'accumulation illimitée de phéromone, et permet à l'algorithme d'"oublier" les mauvaises décisions faites précédemment. Avec ce mécanisme la force de phéromone associée aux arcs qui ne sont pas choisis par les fourmis,sera diminuée exponentiellement avec le nombre d'itérations. l'ij du τ k (t) est la quantité de dépôts de la fourmi k de phéromone sur les arcs; elle est définie par :

$$\Delta\tau_{ij}(t) = \begin{cases} 1/L^k(t) & \text{If arc (i,j) is used by ant k(3)} \\ 0 & \text{otherwise} \end{cases}$$

Où $L\,k$ (t) est la longueur de la k^{eme} excursion de la fourmi. On

Remarque que plus que l'excursion de la fourmi est courte, plus la phéromone reçue par les arcs de l'excursion n'est importante. En général, les arcs qui sont employés par beaucoup de fourmis et qui sont contenus dans des excursions plus courtes recevront plus de phéromone et auront plus de chance d'être choisis dans les futures itérations de l'algorithme [DS00].

Conditions d'arrêt

- Si le temps CPU tmax1 a été atteint.
- Si la meilleure solution n'a pas été améliorée depuis un certain nombre d'itérations.
- Ou bien après un nombre NC-max d'itération.

Algorithme

Tantque (la condition d'arrêt n'est pas atteinte)
Tantque (L'état final n'est pas atteint)
Pour (chaque fourmi)
Choisir l'état suivant en fonction de la mémoire et de l'environnement local
Mettre à jour les phéromones sur l'arc choisi.
FinPour
FinTantQue
Pour (chaque fourmi)
Evaluer la solution obtenue
FinPour
Mise à jour globale des phéromones
FinTantque
Afficher la meilleure solution trouvée

Pseudo code de la méthode colonies de fourmis.

Avantages & inconvénients

Les avantages

- Rapidité de la méthode.
- Nouvelle méthode à trouver des solutions acceptables tout en évitant des convergences prématurées.
- Robuste et basée sur une population d'individus.

Les inconvénients

- Complexe à mettre en place et son paramétrage est subtile.
- Coût relativement élevé de la génération des solutions.

Domaines d'application

L'optimisation par colonie de fourmis est appliquée sur plusieurs problèmes de différents types on note[VaD].

- Applications au problème symétrique et asymétrique de voyageur de commerce.
- Applications aux problèmes d'affectation quadratiques.
- Applications aux problèmes de tournée des véhicules.
- Applications aux problèmes d'établissement d'horaires.
- Applications au problème d'ordonnancement séquentiel.

Hybridation des méthodes

L'hybridation est une tendance observée dans de nombreux travaux réalisés sur les métaheuristiques ces dix dernières années. Elle exploite la puissance de plusieurs algorithmes, et les combine en un seul méta-algorithme [Sbi03]. Il existe plusieurs manières pour faire cette hybridation.

Une des techniques les plus populaires d'hybridation concerne l'utilisation de métaheuristiques à base d'une solution unique avec des métaheuristiques à population. La plupart des applications réussies des métaheuristiques à population sont complétées par une phase de recherche locale [BAF08]. afin d'augmenter leurs capacités d'intensification. Les individus d4une population tentent d'explorer l'espace de solutions afin de trouver les zones prometteuses, lesquelles sont ensuite explorées plus en détail par des méthodes de recherche locale comme la recherche tabou ou le hill-climbing, ou par un algorithme de type 2-opt. Il est à noter également que les deux types de recherche, globale et locale, peuvent être lancés de façon asynchrone, sur des processeurs différents afin d'augmenter la vitesse d'exécution de l'algorithme hybride.

Une deuxième manière d'hybrider consiste à lancer en parallèle l'exécution de la même métaheuristique, voire même plusieurs fois la même métaheuristique mais avec des paramètres différents. Ces processus parallèles communiquent entre eux régulièrement pour échanger de l'information sur leurs résultats partiels. Presque toutes les métaheuristiques classiques ont donné naissance à des versions parallèles plus ou moins fidèles à leur origine.

Enfin une troisième forme d'hybridation combine les métaheuristiques avec des méthodes exactes [HL03]. Une méthode exacte peut être utilisée pour la génération du meilleur voisin d'une solution. On peut par exemple utiliser la méthode branch-and-bound sur une partie du problème de petite taille pour générer des solutions de bonnes qualités. Par ailleurs, une métaheuristique peut être utilisée pour fournir des bornes à une méthode exacte de type branch-and-bound [PBN07].

Les algorithmes hybrides sont sans aucun doute parmi les méthodes les plus puissantes. Malheureusement, les temps de calcul nécessaires peuvent devenir prohibitifs à cause du nombre d'individus manipulés dans la population. Une voie pour résoudre ce problème ibués. En effet, le coût de calcul des méthodes hybrides peut être supporté avec le recours aux grilles de calcul [Mel05].

1.3 Conclusion

Dans ce chapitre on a vu les méthodes principales des métaheuritiques chaque méthode est basée sur une idée différente, mais elles partagent le même handicap si l'espace de recherche est très grand la solution trouvée peut être loin de la solution optimale, donc il est intéressent de proposer une approche coopérative et parallèle utilise les points fortes d'ensemble des méthodes avec un parallélisme pour améliorer les résultats données.

Chapitre II

Parallélismes des métaheuristiques