5.1Les méthodes Métaheuristiques

Les métaheuristiques constituent une classe de méthodes qui fournissent des solutions de bonne qualité en temps raisonnable à des problèmes combinatoires réputés difficiles pour lesquels on ne connaît pas de méthode classique plus efficace. On appelle métaheuristiques (du grec, meta = qui englobe) des méthodes conçues pour échapper aux minima locaux. Le terme meta s'explique aussi par le fait que ces méthodes sont des structures générales dont il faut instancier les composants en fonction du problème par exemple, le voisinage, les solutions de départ ou les critères d'arrêt. Les métaheuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global, c'est-à-dire l'extremum global d'une fonction en évaluant une certaine fonction objectif. Elles se comportent comme des algorithmes de recherche, tentant d'apprendre les caractéristiques d'un problème afin d'en trouver une approximation de la meilleure solution d'une manière proche des algorithmes d'approximation. L'intérêt croissant apporté aux métaheuristiques est tout à fait justifié par le développement des machines avec des capacités calculatoires énorme, ce qui a permis de concevoir des métaheuristiques de plus en plus complexes qui ont fait preuve d'une certaine efficacité lors de la résolution de plusieurs problèmes à caractère NP-difficile.

Il existe de nombreuses métaheuristiques allant de la simple recherche locale à des algorithmes plus complexes de recherche globale. Ces méthodes utilisent cependant un haut niveau d'abstraction, leur permettant d'être adaptées à un large éventail de problèmes d'optimisation combinatoire. Nous pouvons partager les méthodes heuristiques en deux catégories. Celles qui permettent de déterminer un minimum local, et celles qui s'efforcent de déterminer un optimum global.

On appelle méthode (ou algorithme ou recherche) locale celle qui converge vers un minimum local. Les méthodes de recherche locale, appelées aussi méthodes de recherche par voisinages, partent d'une solution initiale et, par raffinements successives, construisent des suites de solutions de coûts décroissants pour un problème de minimisation. Le processus s'arrête lorsqu'on ne peut plus améliorer la solution courante ou parce que le nombre maximal d'itérations (fixé au départ) est atteint. Quoique, ces méthodes ne soient pas complètes (rien n'assure qu'elles pourront trouver toutes les solutions existantes), ni n'assurent la preuve d'optimalité, de telles méthodes parviennent très souvent à trouver des solutions de bonne qualité dans des temps de calcul raisonnables. En effet, elles sont souvent les premières méthodes testées sur les nouveaux problèmes combinatoires émergeant des applications réelles et académiques. On trouve dans la littérature de nombreuses méthodes locales. Les plus anciennes et les plus utilisées sont : la méthode de la descente, le recuit simulé, la recherche tabou, etc.

Contrairement aux méthodes de recherche locale, les méthodes globales ont pour objectif d'atteindre un ou plusieurs optima globaux. Ces méthodes sont appelées également des méthodes à population. Celles-ci sont d'une grande diversité : parmi elles on retrouve notamment les algorithmes génétiques, les algorithmes à évolution différentielle, la recherche dispersée, etc.

D'autre part, on peut partager les métaheuristiques en deux grandes classes: les métaheuristiques à solution unique (i.e.; évoluant avec une seule solution) et celles à solution multiple ou population de solution (Figure 5.1). Les méthodes d'optimisation à population de solutions améliorent, au fur et à mesure des itérations, une population de solutions. L'intérêt de ces méthodes est d'utiliser la population comme facteur de diversité.

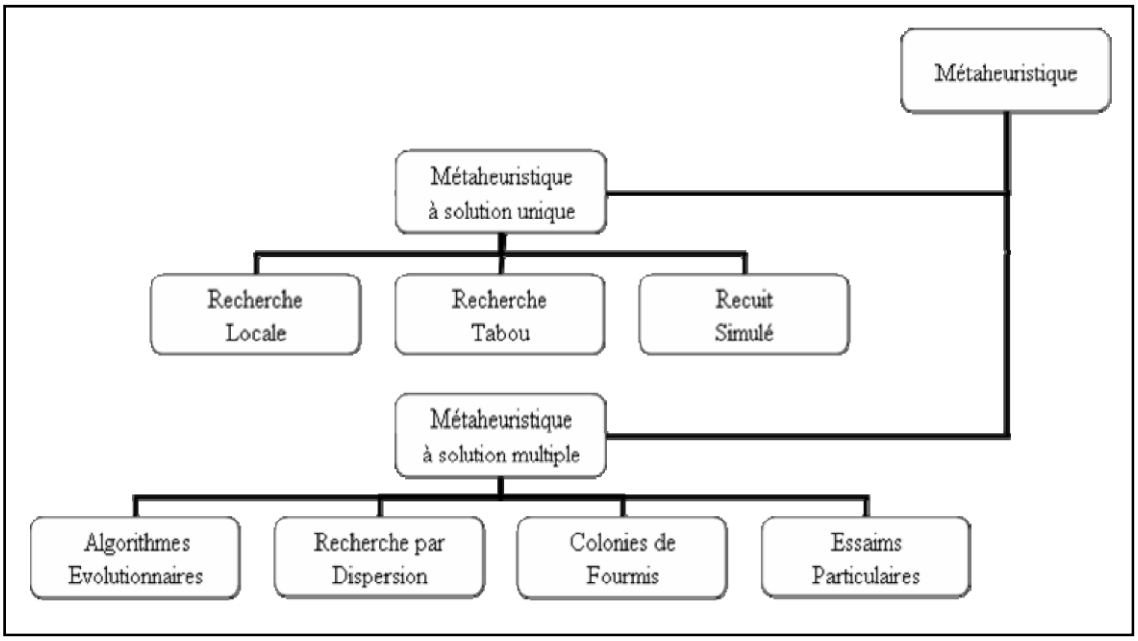


Figure 5.1 – Classification des métaheuristiques.

Finalement une « pseudo-classe » d'algorithmes a émergé ces dernières années pour résoudre des problèmes d'optimisation, qui contient des méthodes hybrides. Le principe consiste à combiner des algorithmes exacts et/ou des algorithmes approchés pour essayer de tirer profit des points forts de chaque approche et améliorer le comportement global de l'algorithme. Néanmoins, ces méthodes appartiennent forcément à l'une des classes de méthodes de résolution vues précédemment. En fait, une méthode hybride est soit exacte (i.e. donne une solution optimale) ou bien approchée (i.e. donne une solution approchée).

5.2 Les Algorithmes Evolutionnaires

Parmi l'ensemble de techniques de recherche et d'optimisation, le développement des algorithmes évolutionnaires (AE) a été très important dans la dernière décennie. Les

algorithmes évolutionnaires font partie du champ de l'Intelligence Artificielle (IA). Il s'agit d'IA de bas niveau, inspirée par " l'intelligence " de la nature. Les algorithmes évolutionnaires sont basés sur le concept de la sélection naturelle élaborée par Charles Darwin [Koza, 1999]. En effet, ces algorithmes adoptent une sorte d'évolution artificielle analogue à l'évolution naturelle. Le vocabulaire utilisé dans les AE est directement inspiré de celui de la théorie de l'évolution et de la génétique. En effet, nous trouvons des mots d'individus (solutions potentielles), de population, de gènes (variables), de chromosomes, de parents, de croisement, de mutations, etc., et nous nous appuierons constamment sur des analogies avec les phénomènes biologiques. Il s'agit de simuler l'évolution d'une population d'individus diverse à laquelle on applique différents opérateurs d'évolution comme les recombinaisons, les mutations, la sélection, etc (figure 5.2). Si la sélection s'opère en se basant sur une fonction d'adaptation, alors la population tend à s'améliorer. Un tel algorithme ne nécessite aucune connaissance du problème. En effet, on peut représenter celui-ci par une boîte noire comportant des entrées (les variables) et des sorties (les fonctions objectifs). L'algorithme ne fait que manipuler les entrées, lire les sorties, manipuler à nouveau les entrées de façon à améliorer les sorties, etc.

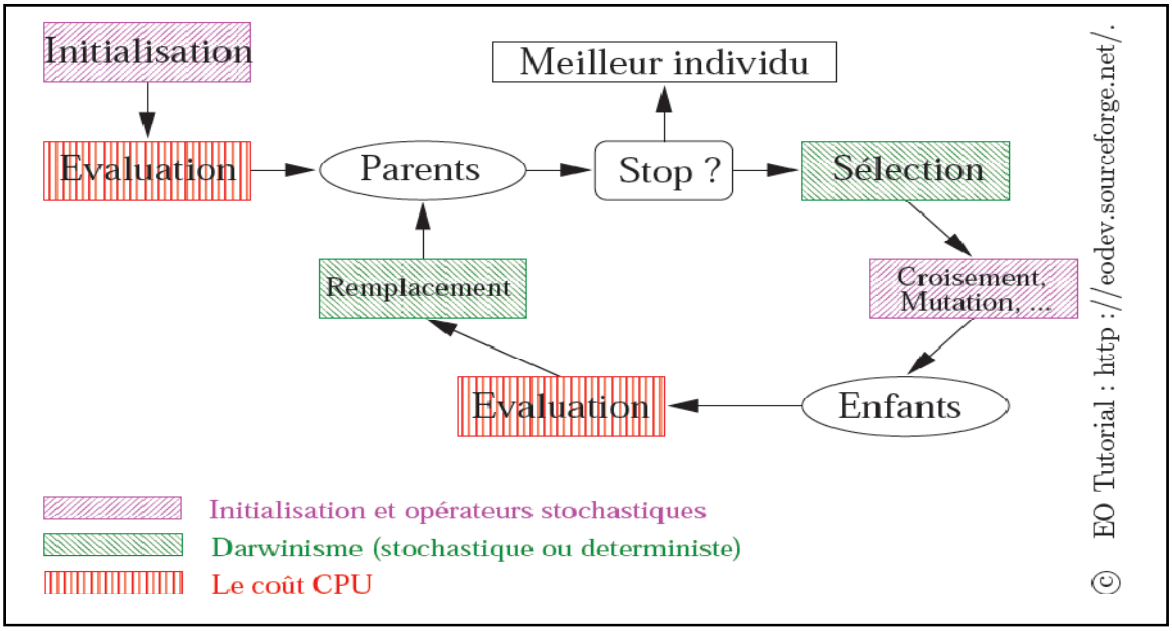


Figure 5.2 – Squelette d'un algorithme évolutionnaire.

On distingue quatre types d'AE (figure 5.3): les Algorithmes Génétiques (AG), les Stratégies d'Évolution (SE), la Programmation Évolutionnaire (PE), et la Programmation Génétique (PG). Dans les années 90, ces quatre champs ont commencé à sortir de leur isolement et ont été regroupés sous le terme anglo-saxon *d'Evolutionnary Computation*.

Le rapport entre la qualité de la solution finale et le temps d'exécution des métaheuristiques se diffère d'un algorithme à un autre, et dépend étroitement du problème à

optimiser. Généralement, la fonction objectif a un grand impact sur la complexité de l'algorithme. En effet, le coût machine d'une optimisation est conditionné par le nombre d'évaluation de la fonction objectif. D'un autre coté, Les applications des métheuristques sont multiples : optimisation de fonctions numériques difficiles (discontinues, multimodales, bruitées…), traitement d'image (alignement de photos satellites, reconnaissance de suspects…), optimisation d'emplois du temps, optimisation de design, contrôle de systèmes industriels, apprentissage des réseaux de neurones, etc.

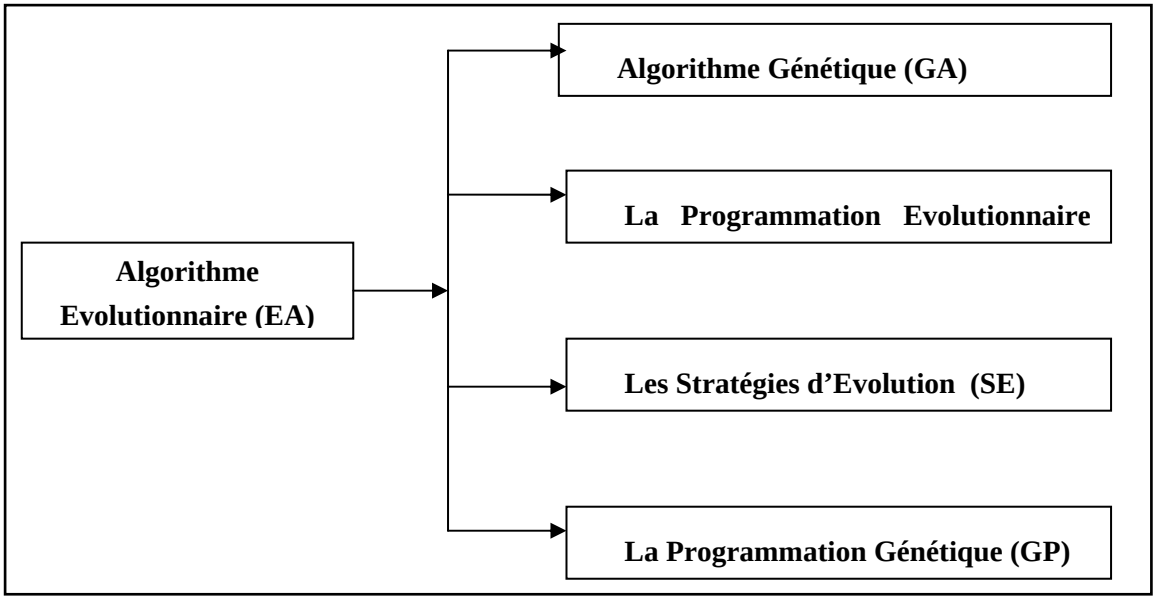


Figure 5.3 – Présentation de types des algorithmes évolutionnaire (EA).

5.3 Les algorithmes génétiques

Les algorithmes génétiques sont à la base des algorithmes d'optimisation stochastiques, mais peuvent également servir pour l'apprentissage automatique, par exemple. Les premiers travaux dans ce domaine ont commencé dans les années cinquante, lorsque plusieurs biologistes américains ont simulé des structures biologiques sur ordinateur. Puis, entre 1960 et 1970, John Holland sur la base des travaux précédents, développe les principes fondamentaux des algorithmes génétiques dans le cadre de l'optimisation mathématique. Malheureusement, les ordinateurs de l'époque n'étaient pas assez puissants pour envisager l'utilisation des algorithmes génétiques sur des problèmes réels de grande taille. La parution en 1989 de l'ouvrage de référence écrit par D.E. Goldberg qui décrit l'utilisation de ces algorithmes dans le cadre de résolution de problèmes concrets a permis de mieux faire connaître ces derniers dans la communauté scientifique et a marqué le début d'un nouvel intérêt pour cette technique d'optimisation, qui reste néanmoins très récente. Parallèlement, des techniques proches ont été élaborées, dont on peut notamment citer la programmation génétique ou les stratégies évolutionnistes. Dans le même temps, la théorie mathématique associée aux algorithmes

génétiques s'est développée mais reste encore bien limitée face à la complexité théorique induite par ces algorithmes.

1.3.1 Les éléments des algorithmes génétiques

Les algorithmes génétiques sont inspirés de la génétique classique et utilise le même vocabulaire. De manière générale, un algorithme génétique est constitué d'une population P de solutions appelées individus, dont l'adaptation à leur entourage est mesurée grâce à une fonction d'aptitude g qui retourne une valeur réelle, appelée fitness. Le principe général d'un AG consiste à simuler l'évolution d'une population d'individus jusqu'à atteindre un critère d'arrêt. Avant d'expliquer en détail le fonctionnement d'un algorithme génétique, nous allons présenter quelques mots de vocabulaire relatifs à la génétique. Ces mots sont souvent utilisés pour décrire un algorithme génétique :

Définition 1.8. (Gène) Un gène est une suite de bases azotées (adénine (A), cytosine (C), guanine (G) et la thymine (T)) 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.

Définition 1.9. (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.

Définition 1.10. (Individu) On appellera individu une des solutions potentielles. Dans la plupart des cas un individu sera représenté par un seul chromosome, dans ce cas, par abus de langage, on utilisera indifféremment individu et chromosome.

Définition 1.11. (Population) On appellera population l'ensemble des solutions potentielles qu'utilise l'AG.

Définition 1.12. (Génération) On appellera génération l'ensemble des opérations qui permettent de passer d'une population Pi à une population Pj. Ces opérations seront généralement : sélection des individus de la population courante, application des opérateurs génétiques, évaluation des individus de la nouvelle population.

Définition 1.13. (La fitness ou fonction d'évaluation) : La fonction de fitness est la pièce maitresse dans le processus d'optimisation. C'est l'élément qui permet aux algorithmes génétiques de prendre en compte un problème donné. Pour que le processus d'optimisation puisse donner de bons résultats, il faut concevoir une fonction de fitness permettant une évaluation pertinente des solutions d'un problème sous forme chiffrée. Cette fonction est déterminée en fonction du problème à résoudre et du codage choisi pour les chromosomes. Pour chaque chromosome, elle attribue une valeur numérique, qui est supposée proportionnelle à la qualité de l'individu en tant que solution. Le résultat renvoyé par la

fonction d'évaluation va permettre de sélectionner ou de refuser un individu selon une stratégie de sélection.

1.3.2 Principes généraux des algorithmes génétiques

Nous allons présenter d'une manière abstraite le principe et le fonctionnement des algorithmes génétiques. Les opérations successives utilisées dans les algorithmes génétiques sont illustrées sur la figure ci-après (figure 5.4).

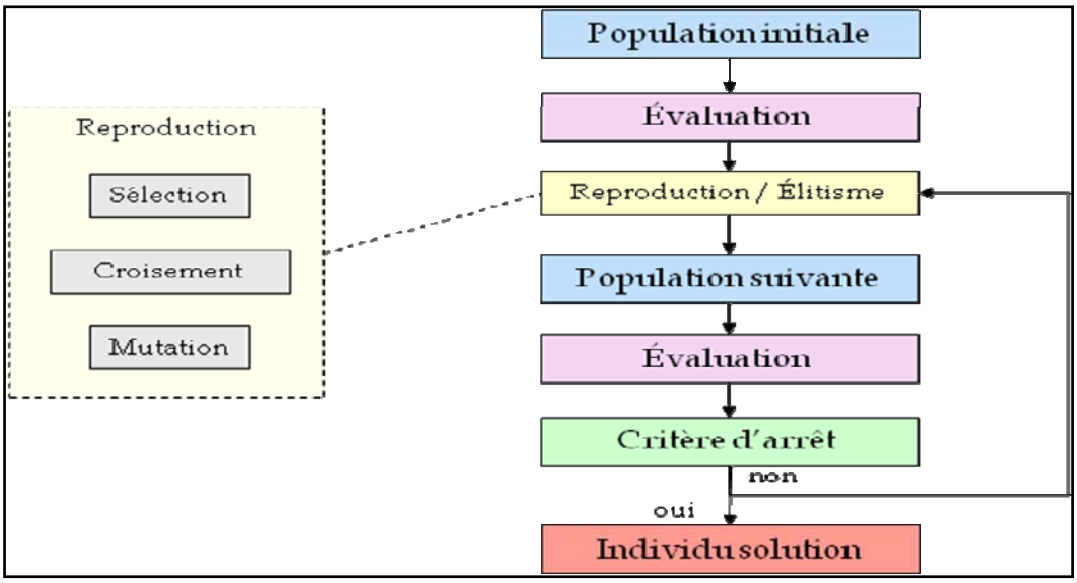


Figure 5.4– Les opérations successives utilisées dans les algorithmes génétiques.

L'algorithme génétique est constitué des étapes suivantes :

- Tout d'abord une population d'individus est générée de façon aléatoire.
- On procède à l'évaluation de l'ensemble des individus de la population initiale.
- On sélectionne un certain nombre d'individus dans la population, afin de produire une population intermédiaire, appelée aussi « mating pool ».
- On sélectionne deux chromosomes parents P1 et P2 en fonction de leurs adaptations. On applique aléatoirement l'opérateur de croisement avec une probabilité Pc pour générer deux chromosomes enfants C1 et C2. L'opération de croisement est suivie par l'opération de mutation avec une probabilité Pm, ce qui produit deux nouveaux individus C'1 et C'2 pour lesquels on évalue leurs fitness avant de les insérer dans la nouvelle population. Contrairement à la reproduction et au croisement qui favorisent l'intensification, cet opérateur favorise la diversification des individus. On réitère les opérations de sélection, de croisement et de mutation afin de compléter la nouvelle population, ceci termine le processus d'élaboration d'une génération.

- On réitère les opérations précédentes à partir de la seconde étape jusqu'à la satisfaction du critère d'arrêt.

De manière plus formelle, voici un algorithme génétique de base :

```
Algorithme 5.1 : algorithme génétique de base

Début
1: Générer une population aléatoire de n chromosomes.
2: Evaluer la fitness des chromosomes avec la fonction f (x)
3: Répéter
4: Calculer la fonction fitness f (x), pour tout chromosome x
5: Appliquer l'opération de sélection
6: Appliquer l'opération de croisement avec une probabilité PC
7: Appliquer l'opération de  mutation avec une probabilité PM
8: Ajouter les nouveaux chromosomes à la nouvelle population
9: Calculer la fonction fitness f (x), pour tout chromosome x
10: Appliquer l'opération de remplacement
11 : Jusqu'à la satisfaction des conditions de terminaison
Fin
```

1.3.3 Codage

C'est une modélisation d'une solution d'un problème quelconque en un chromosome. Le choix du codage est important est souvent délicat. En effet, le choix du type de codage ne peut pas être effectué de manière évidente. La méthode actuelle à appliquer dans le choix du codage consiste à choisir celui qui semble le plus naturel en fonction du problème à traiter. L'objectif est bien sûr d'abord de pouvoir coder n'importe quelle solution. Mais il est souhaitable, au-delà de cette exigence, d'imaginer soit un codage tel que toute chaîne de caractères représente bien une solution réalisable du problème, soit un codage qui facilite ensuite la conception du croisement de telle sorte que les « enfants » obtenus à partir de la recombinaison de leurs « parents » puissent être associés à des solutions réalisables, au moins pour un grand nombre d'entre eux. Parmi les codages les plus utilisés on peut citer.

1.7.3.1 Codage binaire

Ce type de codage est certainement le plus utilisé car il présente plusieurs avantages. Son principe est de coder la solution selon une chaîne de bits (les valeurs 0 ou 1). Le codage

binaire a permis certes de résoudre beaucoup de problèmes, mais il s'est avéré obsolète pour certains problèmes d'optimisation numérique. Il est plus pratique d'utiliser un codage réel des chromosomes.

1	**0**	**0**	**1**	**1**	**0**	**1**

Figure 5.5 – codage binaire d'un chromosome (problème Max Sat).

1.7.3.2 Codage réel

Une autre approche semblable est de coder les solutions en tant que des suites de nombres entiers ou de nombres réels, avec chaque position représentant encore un certain aspect particulier de la solution. Cette approche tient compte d'une plus grande précision et de complexité que le codage basé sur les nombres binaires seulement. Ce type de codage est intuitivement plus proche à l'environnement des problèmes à résoudre.

3	**5**	**2**	**1**	**4**	**6**

Figure 5.6 – Codage binaire d'un chromosome
(Problème TSP).

1.7.3.3 Codage à l'aide de suite alphabétique

Une troisième approche est de représenter des individus dans AG comme une suite de caractères, où chaque caractère représente encore un aspect spécifique de la solution. Ce type de codage est utilisé dans de nombreux cas poussés d'algorithmes génétiques comme en bioinformatique.

A	**-**	**G**	**T**	**C**

Figure 5.7 – Codage alphabétique d'un chromosome
(Problème bioinformatique).

1.7.3.4 Codage sous forme d'arbre

Ce codage utilise une structure arborescente avec une racine de laquelle peuvent être issus un ou plusieurs fils. Un de leurs avantages est qu'ils peuvent être utilisés dans le cas de problèmes où les solutions n'ont pas une taille finie.

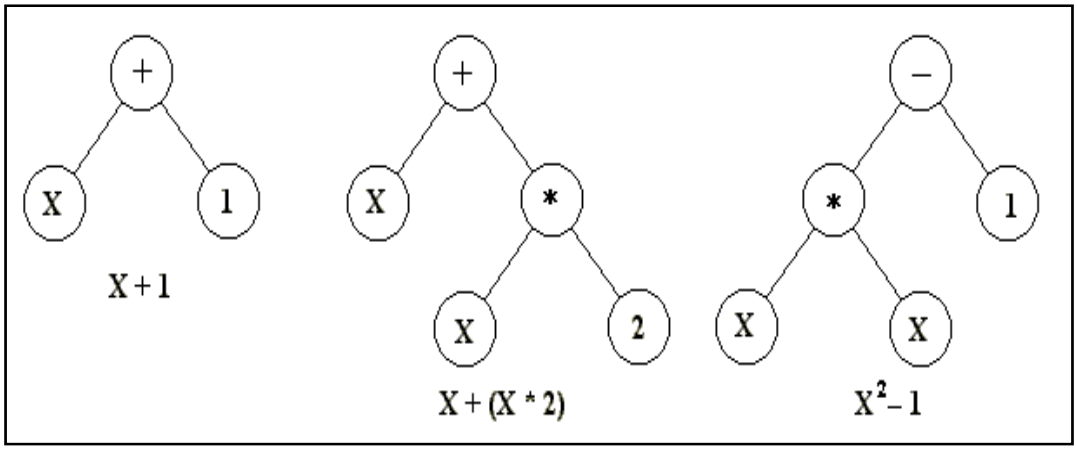


Figure 5.8 – Codage en arbre Trois arbres simples de programme de la sorte normalement utilisée dans la programmation génétique.

1.3.4 Les opérateurs d'un algorithme génétique

Quatre opérateurs caractérisent les algorithmes génétiques et rappellent l'origine de ces méthodes. Ils vont permettre à la population d'évoluer, par la création d'individus nouveaux construits à l'aide des individus anciens. Ces opérations sont : la sélection et le croisement, la mutation et le remplacement.

1.7.4.1 La sélection

La sélection permet d'identifier statistiquement les meilleurs individus de la population courante qui seront autorisés à se reproduire. Cette opération est fondée sur la performance des individus, estimée à l'aide de la fonction d'adaptation. Il existe différents principes de sélection :

- *Sélection par roulette (Wheel)*

Elle consiste à associer à chaque individu un segment dont la longueur est proportionnelle à sa fitness. Ces segments sont ensuite concaténés sur un axe gradué que l'on normalise entre 0 et 1 (figure 5.9). On tire alors un nombre aléatoire de distribution uniforme entre 0 et 1, puis on regarde quel est le segment sélectionné, et on reproduit l'individu correspondant. Avec cette technique, les bons individus seront plus souvent sélectionnés que les mauvais, et un même individu pourra avec cette méthode être sélectionné plusieurs fois. Néanmoins, sur des populations de petite taille, il est difficile d'obtenir exactement l'espérance mathématique de sélection à cause du faible nombre de tirages. Le cas idéal d'application de cette méthode est bien évidemment celui où la population est de taille infinie. On aura donc un biais de sélection plus ou moins fort suivant la dimension de la population. Un autre problème rencontré lors de l'utilisation de la sélection par roulette est lorsque la valeur d'adaptation des chromosomes varie énormément. Si la meilleure fonction d'évaluation d'un chromosome représente 90% de

la roulette alors les autres chromosomes auront très peu de chance d'être sélectionnés et on arriverait à une stagnation de l'évolution.

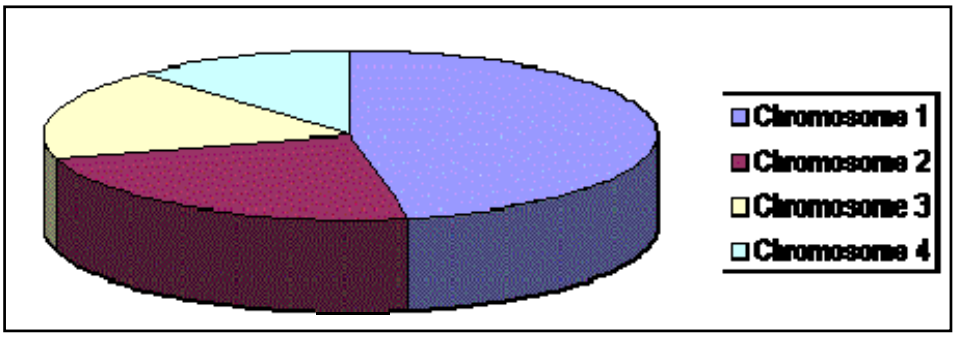


Figure 5.9 – Sélection par roulette.

- *Sélection par rang*

La sélection par rang est basée sur le trie de la population selon la fitness de chaque individu. Elle consiste à attribuer un entier de 1 à N pour une population de N chromosomes, du mauvais au meilleur. La sélection par rang d'un chromosome est similaire à la sélection par roulette, mais les proportions sont en relation avec le rang plutôt qu'avec la valeur de l'évaluation. Avec cette méthode de sélection, tous les chromosomes ont une chance d'être sélectionnés. Cependant, son grand inconvénient est la convergence lente vers la bonne solution. Ceci est dû au fait que les meilleurs chromosomes ne diffèrent pas énormément des plus mauvais. Vous pouvez voir dans la figure ci-après, comment la situation change après avoir transformé la fitness en rang.

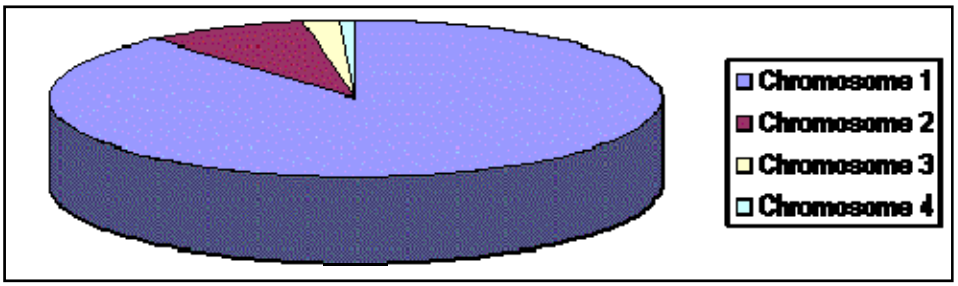
- *Sélection par tournoi*

Cette méthode est celle avec laquelle on obtient les résultats les plus satisfaisants. Elle consiste à choisir aléatoirement k individus (le nombre de participants à un tournoi) et à les confronter entre eux par le biais de la fonction d'adaptation, et de sélectionner ensuite le meilleur parmi eux. On répète ce processus autant de fois de manière à obtenir les n individus de la population qui serviront de parents. La variance de cette méthode est élevée et le fait d'augmenter ou de diminuer la valeur de *k* permet respectivement de diminuer ou d'augmenter la pression de la sélection.

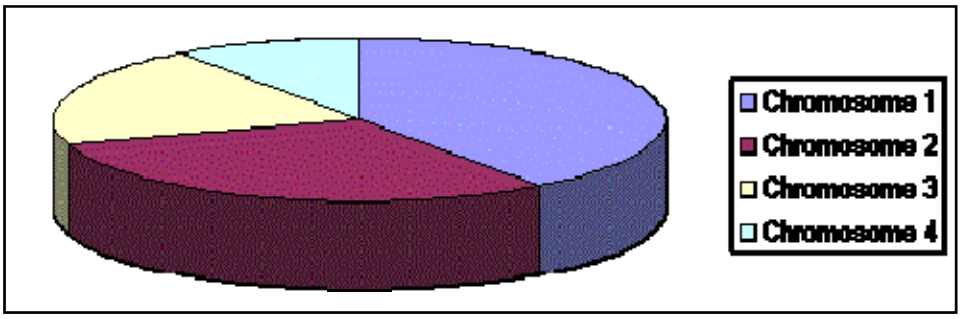
- *Sélection Steady-State*

Ce n'est pas une méthode particulière de sélection des chromosomes parents. L'idée principale est qu'une grande partie de la population puisse survivre à la prochaine génération. L'algorithme génétique marche alors de la manière suivante. A chaque génération quelques chromosomes sont sélectionnés parmi ceux qui ont le meilleur coût, afin de créer des

chromosomes fils. Ensuite les chromosomes les plus mauvais sont retirés et remplacés par les nouveaux. Le reste de la population survie à la nouvelle génération.



a. Situation avant le tri (graphe de fitnesses)



b. Situation après le tri (graph d'ordre)

Figure 5.10 – Sélection par rang.

- *Elitisme*

L'élitisme consiste à conserver à chaque génération un certain nombre des meilleurs chromosomes de la population qui pourraient disparaître par les opérations de mutation, croisement ou sélection. Elle consiste à copier un ou plusieurs des meilleurs chromosomes dans la nouvelle génération. Ensuite, on génère le reste de la population selon l'algorithme de reproduction usuel. Cette méthode améliore considérablement les algorithmes génétiques, car elle permet de ne pas perdre les meilleures solutions.

1.7.4.2 Le Croisement

Le croisement a pour but de produire une nouvelle génération d'individus en recombinant les individus sélectionnés par l'opérateur de sélection. C'est l'opérateur de l'algorithme génétique qui permet le plus souvent de se rapprocher de l'optimum d'une fonction en combinant les gènes. Ainsi, dans un premier temps, les individus sélectionnés sont répartis aléatoirement en couples de parents. Puis, chaque couple de parents subit une opération de recombinaison afin de générer un ou deux enfants. Bien qu'il soit aléatoire, cet échange d'informations offre aux algorithmes génétiques une part de leur puissance : quelque fois, de

"bons" gènes d'un parent viennent remplacer les "mauvais" gènes d'un autre et créent des fils mieux adaptés aux parents.

Le type de croisement le plus ancien est le croisement à découpages de chromosomes, ou *croisement 1-point* (figure 5.11 à gauche). Pour effectuer ce type de croisement sur des chromosomes constitués de L gènes, on tire aléatoirement une position inter-gènes dans chacun des parents. On échange ensuite les deux sous-chaînes de chacun des chromosomes ce qui produit deux enfants C1 et C2. Ce mécanisme présente l'inconvénient de privilégier les extrémités des individus. Et selon le codage choisi, il peut générer des fils plus ou moins proches de leurs parents. Pour éviter ce problème, on peut étendre ce principe en découpant le chromosome non pas en 2 sous-chaînes mais en 3, 4, etc. On parle alors de croisement *k-point ou multi-point* (figures 1.11 à droite).

L'autre type de croisement est le *croisement uniforme*. Le croisement uniforme consiste à générer un enfant en échangeant chaque gène des deux individus parents avec une probabilité uniforme égale à 0.5 (voir Figure 1.12). Cet opérateur peut être vu comme le cas extrême du croisement multi-point dans lequel le nombre de points de coupure est déterminé aléatoirement au cours de l'opération.

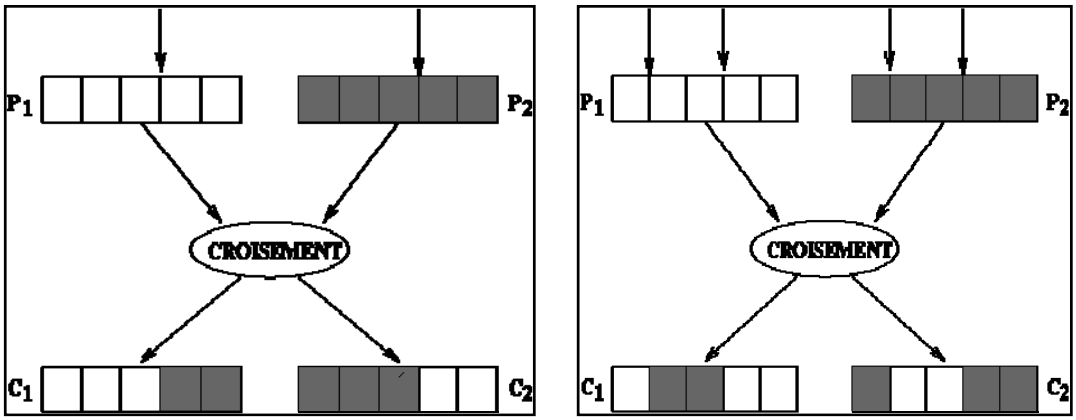


Figure 5.11 – L'opération de croisement k-point. À gauche Croisement 1-point.

Figure 5.12 – L'opération de croisement uniforme.

1.7.4.3 La mutation

Contrairement au croisement qui est un operateur d'exploitation, la mutation est principalement un opérateur d'exploration de l'espace de recherche. Le rôle de cet opérateur consiste à modifier aléatoirement, avec une certaine probabilité, la valeur d'un gène d'un chromosome (figure 5.13). Dans le cas du codage binaire, chaque bit $a_i \in \{0,1\}$, est remplacé selon une probabilité p_m par son complémentaire : $\tilde{a}_i = 1 - a_i$. Malgré, l'aspect marginal de la mutation dans la mesure où sa probabilité est en général fixée assez faible (de l'ordre de 1%), elle confère aux algorithmes génétiques une grande capacité d'exploration de tous les points de l'espace de recherche. Cet opérateur est donc d'une grande importance et il est loin d'être marginal. Comme on le voit dans la figure 5.14, sans mutation, la solution pourrait converger vers une solution locale optimale (les points jaunes). Une mutation réussie peut créer des solutions complètement aléatoires, conduisant à des solutions "inexplorées" qui ne peuvent être atteint via l'opération de croisement une fois que la population a convergée. L'opération de mutation a de fait un double rôle : celui d'effectuer une recherche locale et/ou éviter une stagnation autour d'optima locaux (figure 5.14). On distingue dans la littérature plusieurs types de mutations selon la nature de codage utilisé : mutation 1-bit, mutation réelle, etc.

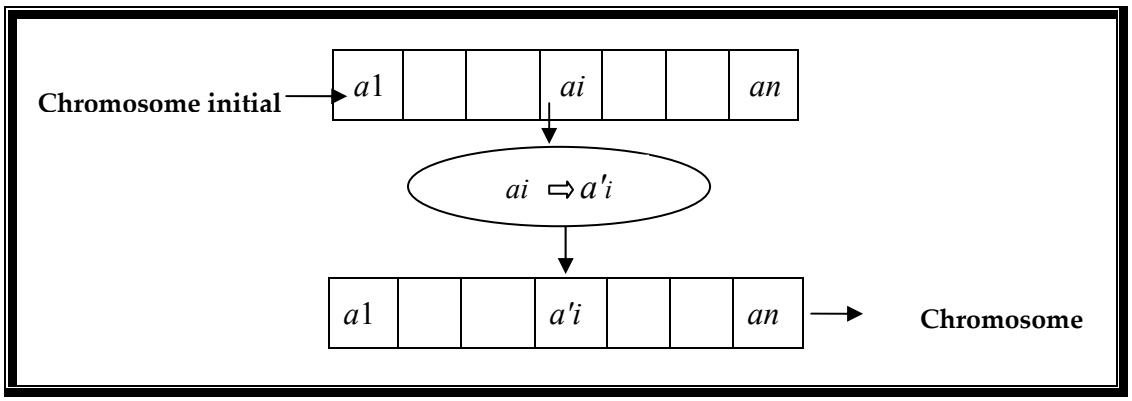


Figure 5.13 – Principe de l'opérateur de mutation.

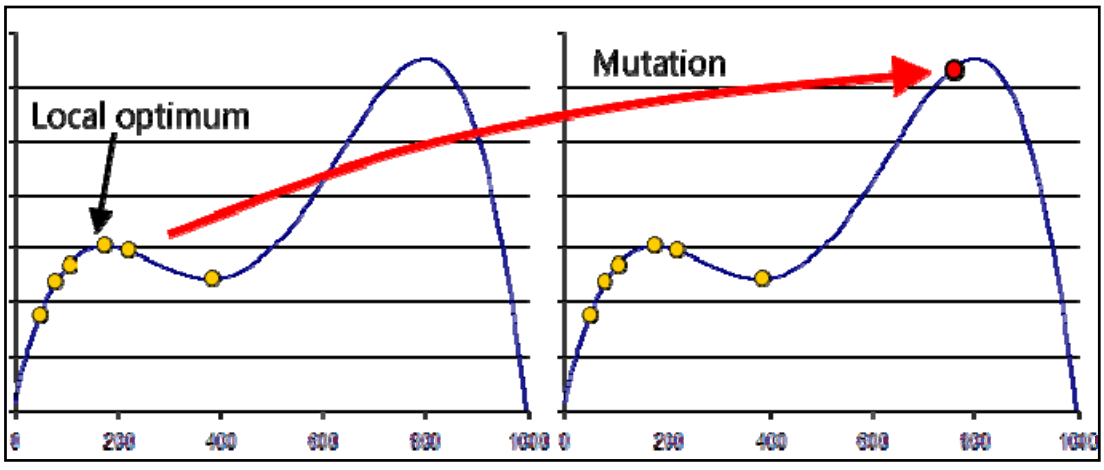


Figure 5.14– La diversification de l'opérateur de mutation.

1.7.4.3 Operateur de remplacement

Cet operateur est basé, comme l'operateur de sélection, sur la fitness des individus. Son travail consiste à déterminer les chromosomes parmi la population courante qui constituant la population de la génération suivante. Cette opération est appliquée après l'application successive des opérateurs de sélection, de croisement et de mutation. On trouve essentiellement 2 méthodes de remplacement différentes :

• *Le remplacement stationnaire :* dans ce cas, les enfants remplacent automatiquement les parents sans tenir compte de leurs performances respectives. Le nombre d'individus de la population est constant tout au long du cycle d'évolution, ce qui implique donc d'initialiser la population initiale avec un nombre suffisant d'individus.

• *Le remplacement élitiste :* dans ce cas, on garde au moins les individus possédant les meilleures performances d'une génération à la suivante. En général, on peut partir du principe qu'un nouvel individu (enfant) prend place au sein de la population que s'il remplit le critère d'être plus performant que le moins performant des individus de la population précédente. Donc les enfants d'une génération ne remplaceront pas nécessairement leurs parents comme dans le remplacement stationnaire et par conséquent la taille de la population n'est pas figée au cours du temps. Ce type de stratégie améliore les performances des algorithmes évolutionnaires dans certains cas. Mais présente aussi un désavantage en augmentant le taux de convergence prématuré.

1.3.5 Critères d'arrêt

Le critère d'arrêt indique que la solution est suffisamment approchée de l'optimum. Plusieurs critères d'arrêt de l'algorithme génétique sont possibles. On peut arrêter l'algorithme après un nombre de générations suffisant pour que l'espace de recherche soit convenablement exploré. Ce critère peut s'avérer coûteux en temps de calcul si le nombre d'individus à traiter dans chaque population est important. L'algorithme peut aussi être arrêté lorsque la population n'évolue plus suffisamment rapidement. On peut aussi envisager d'arrêter l'algorithme lorsque la fonction d'adaptation d'un individu dépasse un seuil fixé au départ. Nous pouvons également faire des combinaisons des critères d'arrêt précédents.

1.3.6 Les avantages et les limites des algorithmes génétiques

Un des grands avantages des algorithmes génétiques est qu'ils autorisent la prise en compte de plusieurs critères simultanément, et qu'ils parviennent à trouver de bonnes solutions sur des problèmes très complexes. Le principal avantage des AG par rapport aux autres techniques d'optimisation combinatoire consiste en une combinaison de :

- l'exploration de l'espace de recherche, basée sur des paramètres aléatoires, grâce à une recherche parallèle,

- l'exploitation des meilleures solutions disponibles à un moment donné.

Ils doivent simplement déterminer entre deux solutions quelle est la meilleure, afin d'opérer leurs sélections. Leur utilisation se développe dans des domaines aussi divers que l'économie, la bioinformatique ou la programmation des robots, etc.

L'inconvénient majeur des algorithmes génétiques est le coût d'exécution important par rapport à d'autres métaheuristiques. Les AG nécessitent de nombreux calculs, en particulier au niveau de la fonction d'évaluation. Mais avec les capacités calculatoire des ordinateurs récents, ce problème n'est pas grand. D'un autre coté, l'ajustement d'un algorithme génétique est délicat : des paramètres comme la taille de la population ou le taux de mutation sont parfois difficiles à déterminer. Or le succès de l'évolution en dépend et plusieurs essais sont donc nécessaires, ce qui limite encore l'efficacité de l'algorithme. Un autre problème important est celui des optima locaux. En effet, lorsqu'une population évolue, il se peut que certains individus qui à un instant occupent une place importante au sein de cette population deviennent majoritaires. À ce moment, il se peut que la population converge vers cet individu et s'écarte ainsi d'individus plus intéressants mais trop éloignés de l'individu vers lequel on converge. Il faut mentionner également le caractère indéterministe des AGs. Comme les opérateurs génétiques utilisent des facteurs aléatoires, un AG peut se comporter différemment pour des paramètres et population identiques. Afin d'évaluer correctement l'algorithme, il faut l'exécuter plusieurs fois et analyser statistiquement les résultats.

Bibliographie

[Baeck et al., 1997] Baeck T., Fogel D.B., and Michalewicz Z. Handbook of Evolutionary Computation. *Institute of Physics Publishing and Oxford University Press*, 1997.

[Baeck et al., 2000] Baeck T., Fogel D.B., and Michalewicz Z., editors. Evolutionary Computation: Advanced Algorithms and Operators. *Institute of Physics Publishing*, Bristol, 2000.

[Goldberg, 1989] Goldberg, D. E. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, 1989.

[Goldberg, 1994] Goldberg G.E., Algorithmes génétiques, Éditions Addison-Wesley, Paris, 1994.

[Moscato et al, 2005] Moscato, P., and Cotta, C. A Gentle Introduction to Memetic Algorithms. Operations Research & Management Science 57(2), 105–144, 2005.

[Moscato,1989] Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Caltech Concurrent Computation Program (report 826),1989.

[Palpant, 2005] Palpant M. : Recherche exacte et approchée en optimisation combinatoire : schémas d'intégration et applications. Thèse de Doctorat, Université d'Avignon, 2005.

[Heudin, 1994] Heudin J-C, La Vie artificielle, *éditions Hermès Science*, Paris, 1994.

[Layeb, 2009] Layeb,A. working Thesis, 2009.