A decorative frame resembling a scroll, with a vertical strip on the left and a horizontal strip at the top, both featuring rounded ends and a slight shadow effect.

CHAPITRE II :

LES TECHNIQUES D'OPTIMISATION :

ALGORITHMES GENETIQUES

II.1 Introduction

Les ingénieurs et les décideurs sont confrontés quotidiennement à des problèmes de Complexité grandissante, qui surgit dans des secteurs techniques très divers, comme dans la conception de systèmes mécaniques, le traitement des images, l'électronique ou la recherche opérationnelle [10], Parmi les problèmes rencontrés par le chercheur et l'ingénieur, les problèmes d'optimisation occupent à notre époque une place de choix.

Le problème à résoudre peut souvent s'exprimer comme un problème d'optimisation : on définit une fonction objectif, ou fonction fitness, que l'on cherche à minimiser ou à maximiser par rapport à tous les paramètres concernés. Tous les paramètres des solutions retenues doivent respecter des contraintes, qui peuvent aussi être vues comme définissant l'espace de recherche, faute de quoi ces solutions ne sont pas réalisables.

Les méthodes d'optimisation peuvent être classées de différentes manières : nous les classerons en méthodes déterministes et méthodes non-déterministes. Les méthodes déterministes sont généralement efficaces quand l'évaluation de la fonction est très rapide, ou quand la forme de la fonction est connue a priori. Les cas plus complexes (temps de calcul important, nombreux optima locaux, fonctions non-dérivables, fonctions fractales, fonctions bruitées...) seront souvent traités plus efficacement par des méthodes non-déterministes. Ces dernières font appel à des tirages de nombres aléatoires. Elles permettent d'explorer l'espace de recherche plus efficacement [11].

Donc, dans ce chapitre nous aborderons une nouvelle technique la plus utilisée pour optimiser les problèmes complexes qui sont les algorithmes évolutionnaires, en particulier: Les algorithmes génétiques (Genetic Algorithms).

II.2. Historique

La plupart des écrits qui abordent les algorithmes évolutionnaires se réfèrent à l'évolutionnisme de Darwin, d'où ces algorithmes sont considérés comme étant la version artificielle du Darwinisme. Sans entrer dans la polémique de l'évolutionnisme, les tentatives des néo-darwinistes à prouver ce qu'avancait Darwin dans son livre intitulé, l'origine des espèces (1859), et les preuves scientifiques réfutant totalement cette théorie, nous diront que les techniques évolutionnaires d'optimisation globale ont émergé depuis plus d'une quarantaine d'années. Ces techniques, s'inspirant des phénomènes biologiques, ont été conçues pour une catégorie de problèmes bien spécifiques.

Les algorithmes évolutionnaires ont eu une histoire relativement courte. Cependant, c'est une histoire qui a progressé rapidement en commençant en 1960 par la modélisation de

l'évolution biologique sur un ordinateur [20]. Après plus d'une vingtaine d'années de publications sporadiques, les algorithmes évolutionnaires se sont déployés d'une façon très claire après 1980.

II.3 Les algorithmes évolutionnaires

Les algorithmes évolutionnistes ou algorithmes évolutionnaires (evolutionary computation en anglais), sont une famille d'algorithmes s'inspirant de la théorie de l'évolution pour résoudre des problèmes divers. Leur principe est de simuler l'évolution d'une population d'individus divers auquel on applique différents opérateurs génétique et que l'on soumet à chaque génération à une sélection. Ces algorithmes sont de plus en plus utilisés dans l'industrie car ils sont particulièrement adaptés aux problèmes d'optimisation comportant de nombreux paramètres [12].

II.4 Les Algorithmes génétiques

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle : croisements, mutations, sélection, etc. Les algorithmes génétiques ont déjà une histoire relativement ancienne, puisque les premiers travaux de John Holland sur les systèmes adaptatifs remontent à 1962 [13],[14]. Un algorithme génétique recherche le ou les extrema d'une fonction définie sur un espace de données. Cette technique repose sur l'évolution d'une population de solutions sous l'action de règles précises en optimisant un comportement donné, exprimé sous forme d'une fonction, dite fonction sélective (*fitness function*) ou fonction d'adaptation.

Principe des algorithmes génétiques

Les algorithmes génétiques sont basés sur cinq éléments principaux :

1. Un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace de recherche une structure de données. Elle se place généralement après une phase de modélisation mathématique du problème traité. Le choix du codage des données conditionne le succès des algorithmes génétiques. Les codages binaires ont été très employés à l'origine [13]. Les codages réels sont désormais largement utilisés, notamment dans les domaines applicatifs, pour l'optimisation de problèmes à variables continues [14].

2. Un mécanisme de génération de la population initiale. Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la

convergence vers l'optimum global. Dans le cas où l'on ne connaît rien du problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.

3. Une fonction à optimiser. Celle-ci prend ses valeurs dans \mathbb{R}^+ et est appelée *fitness* ou fonction d'évaluation de l'individu. Elle est utilisée pour sélectionner et reproduire les meilleurs individus de la population.

4. Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace de recherche. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace de recherche.

5. Des paramètres de dimensionnement : taille de la population, nombre total de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation.

Le principe général du fonctionnement d'un algorithme génétique est représenté dans la figure II.1.

Population d'individus

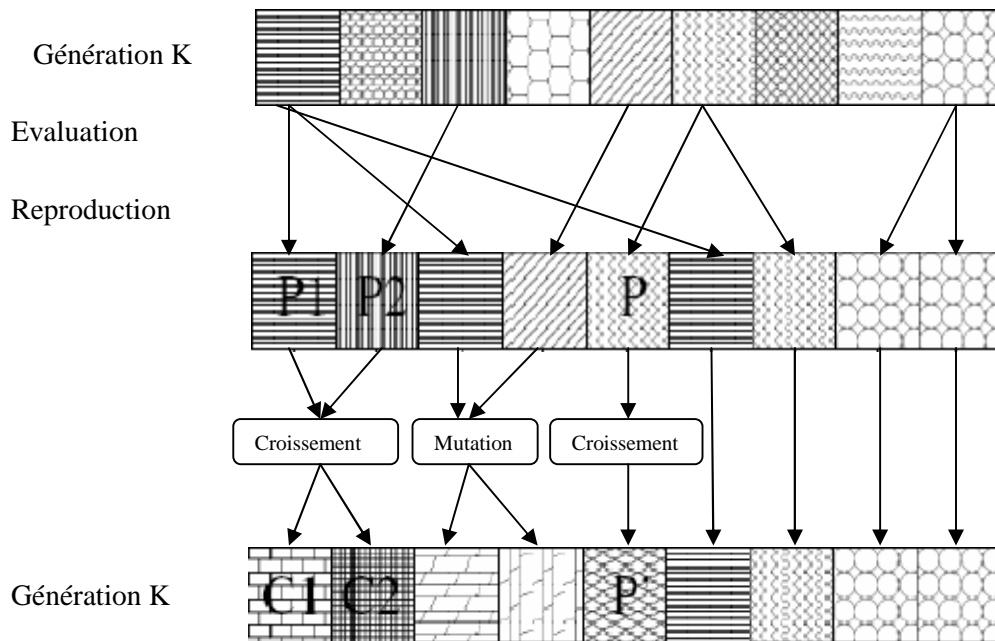


Figure. II.1 Principe général des algorithmes génétiques.

On commence par engendrer une population d'individus de façon aléatoire. Pour passer d'une génération k à la génération $k+1$, les trois opérations suivantes sont répétées pour tous les éléments de la population k . Des couples de parents $P1$ et $P2$ sont sélectionnés en fonction de leurs adaptations. L'opérateur de croisement leur est appliqué avec une probabilité P_c (généralement autour de 0.6) et engendre des couples d'enfants $C1$ et $C2$. D'autres éléments P sont sélectionnés en fonction de leur adaptation. L'opérateur de mutation leur est appliqué avec la probabilité P_m (P_m est généralement très inférieur à P_c) et engendre des individus mutés P' .

Les enfants (C1,C2) et les individus mutés P' sont ensuite évalués avant insertion dans la nouvelle population (la figure 2 présente le cas où les enfants et les individus mutés remplacent les parents).

Différents critères d'arrêt de l'algorithme peuvent être choisis :

- Le nombre de générations que l'on souhaite exécuter peut être fixé a priori. C'est ce que l'on est tenté de faire lorsque l'on doit trouver une solution dans un temps limité.
- L'algorithme peut être arrêté lorsque la population n'évolue plus ou plus suffisamment rapidement.

On peut détailler le fonctionnement des algorithmes génétiques par la figure suivante (figure: II.2)

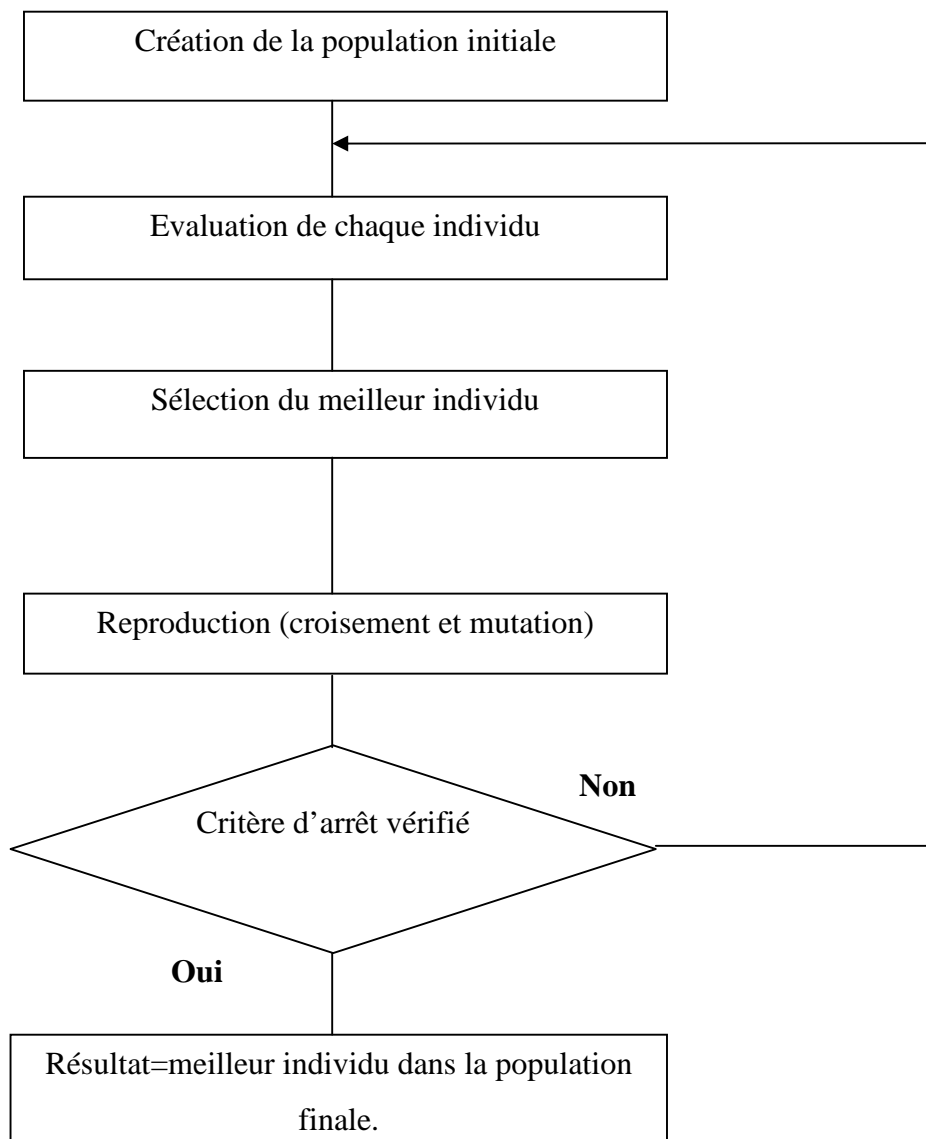


Figure II.2. L'Algorithme Génétique de base.

II.5 Caractéristiques des algorithmes génétiques

II.5.1 Codage des données

Historiquement, le codage utilisé par les algorithmes génétiques était représenté sous forme de chaînes de bits contenant toute l'information nécessaire à la description d'un point dans l'espace de recherche. Ce type de codage a pour intérêt de permettre de créer des opérateurs de croisement et de mutation simples. C'est également en utilisant ce type de codage que les premiers résultats de convergence théorique ont été obtenus.

Cependant, ce type de codage n'est pas toujours bon :

- Pour des problèmes d'optimisation dans des espaces de grande dimension, le codage binaire peut rapidement devenir mauvais. Généralement, chaque variable est représentée par une partie de la chaîne de bits et la structure du problème n'est pas bien reflétée, l'ordre des variables ayant une importance dans la structure du chromosome, alors qu'il n'a pas forcément dans la structure du problème.

Les algorithmes génétiques utilisant des vecteurs réels [18], [19] évitent ce problème en conservant les variables du problème dans le codage de l'élément de population, sans passer par le codage binaire intermédiaire. Certains les appellent RCGA (*Real Coded Genetic Algorithms*, d'autres parlent d'algorithmes évolutionnaires. La structure du problème est conservée dans le codage. Dans notre travail, nous avons utilisé le RCGA pour éviter les problèmes de codage binaire et aussi minimiser le temps de calcul (le temps de codage et décodage des paramètres).

II.5.2 Génération aléatoire de la population initiale

Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Si la position de l'optimum dans l'espace de recherche est totalement inconnue, il est naturel d'engendrer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace d'état, en veillant à ce que les individus produits respectent les contraintes [17]. Si par contre, des informations a priori sur le problème sont disponibles, il paraît bien évidemment naturel d'engendrer les individus dans un sous-domaine particulier afin d'accélérer la convergence. Dans l'hypothèse où la gestion des contraintes ne peut se faire directement, les contraintes sont généralement incluses dans le critère à optimiser sous forme de pénalités.

II.5.3 Gestion des contraintes

Un élément de population qui viole une contrainte se verra attribuer une mauvaise *fitness* et aura une probabilité forte d'être éliminé par le processus de sélection. Il peut cependant être

intéressant de conserver, tout en les pénalisant, les éléments non admissibles car ils peuvent permettre de générer des éléments admissibles de bonne qualité. Pour de nombreux problèmes, l'optimum est atteint lorsque l'une au moins des contraintes de séparation est saturée, c'est-à-dire sur la frontière de l'espace admissible.

Gérer les contraintes en pénalisant la fonction *fitness* est difficile, un “dosage” s'impose pour ne pas favoriser la recherche de solutions admissibles au détriment de la recherche de l'optimum ou inversement.

Disposant d'une population d'individus non homogène, la diversité de la population doit être entretenue au cours des générations, afin de parcourir le plus largement possible l'espace de recherche. C'est le rôle des opérateurs de croisement et de mutation.

II.5.4 Opérateur de croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants. Initialement, le croisement associé au codage par chaînes de bits est le croisement à découpage de chromosomes. Pour effectuer ce type de croisement sur des chromosomes constitués de M gènes, on tire aléatoirement une position dans chacun des parents. On échange ensuite les deux sous-chaînes terminales de chacun des deux chromosomes, ce qui produit deux enfants C1 et C2 (voir figure II.3).

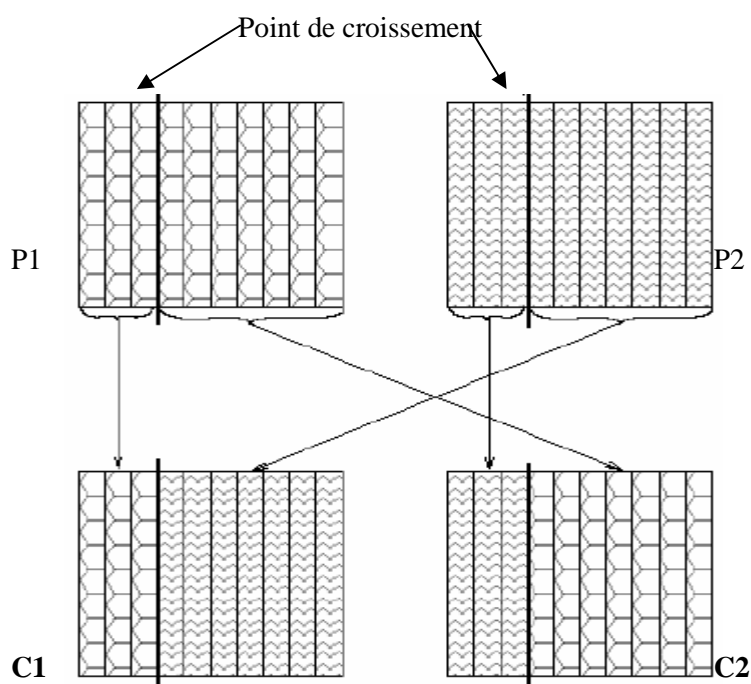


Figure II.3. Croisement à 1 point

On peut étendre ce principe en découpant le chromosome non pas en 2 sous chaînes mais en 3, etc. [13]. (Figure II.4).

Point de croisement

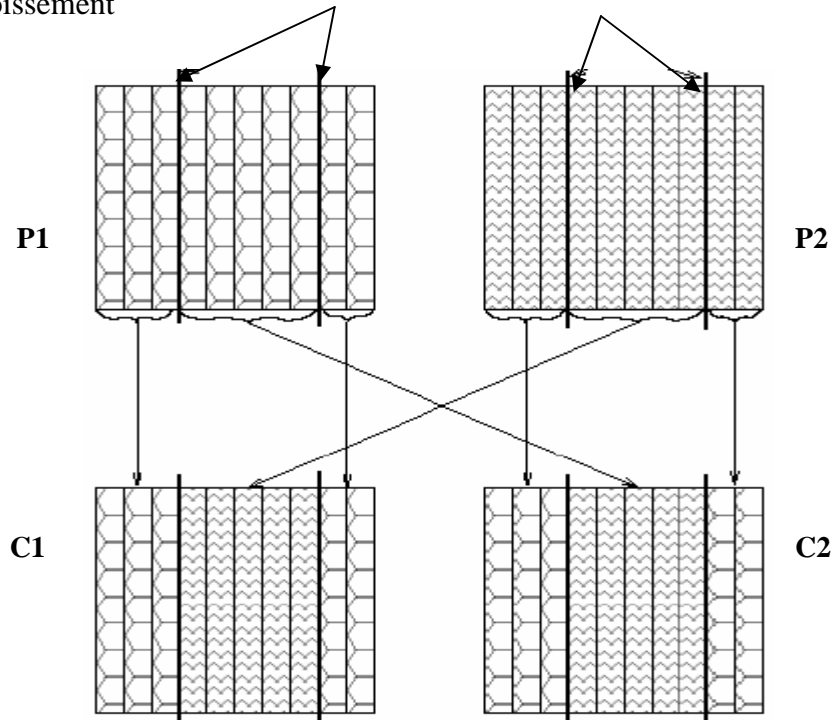


Figure II.4. Croisement à 2 points.

Ce type de croisement à découpage de chromosomes est très efficace pour les problèmes discrets. Pour les problèmes continus, un croisement “barycentrique” est souvent utilisé : deux gènes $P1(k)$ et $P2(k)$ sont sélectionnés dans chacun des parents à la même position i . Ils définissent deux nouveaux gènes $C1(k)$ et $C2(k)$:

par combinaison linéaire :

$$\begin{cases} C1(k) = aP1(k) + (1 - a)P2(k) \\ C2(k) = (1 - a)P1(k) + aP2(k) \end{cases} \quad (\text{II. 1})$$

Où a est un coefficient de pondération aléatoire adapté au domaine d'extension des gènes (il n'est pas nécessairement compris entre 0 et 1, il peut par exemple prendre des valeurs dans l'intervalle $[-0.5, 1.5]$, ce qui permet d'engendrer des points entre, ou à l'extérieur des deux gènes considérés).

II.5.5 Opérateur de mutation

L'opérateur de mutation apporte aux algorithmes génétiques la propriété de parcourir l'espace de recherche. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace de recherche, sans pour autant les parcourir tous dans le processus de résolution. Ainsi en toute rigueur, l'algorithme génétique peut converger sans croisement, et certaines implémentations fonctionnent de cette manière [16]. Les propriétés de convergence des algorithmes génétiques sont donc fortement dépendantes de cet opérateur sur le plan théorique.

Pour les problèmes discrets, l'opérateur de mutation consiste généralement à tirer aléatoirement un gène dans le chromosome et à le remplacer par une valeur aléatoire (voir figure II.7). Si la notion de distance existe, cette valeur peut être choisie dans le voisinage de la valeur initiale.

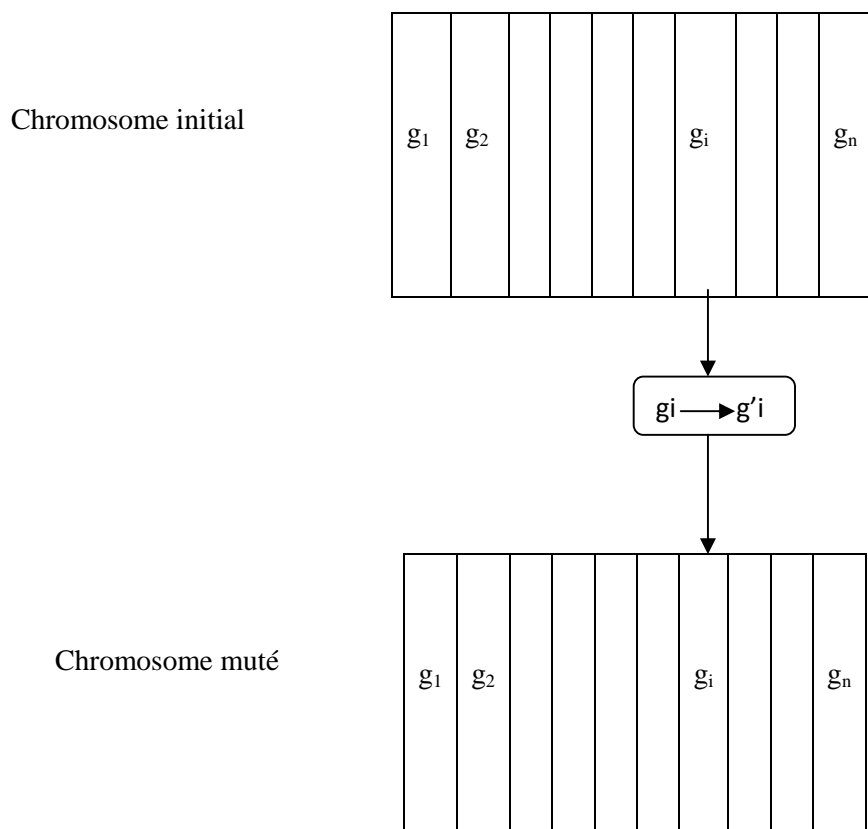


Figure II.7. Principe de l'opérateur de mutation.

Dans les problèmes continus, on procède un peu de la même manière en tirant aléatoirement un gène dans le chromosome, auquel on ajoute un bruit généralement gaussien. L'écart-type de ce bruit est difficile à choisir a priori.

II.5.6 La diversité génétique

Si les individus d'une population se ressemblent trop, alors la population devient trop homogène, et son évolution devient l'évolution d'un seul individu, ce qui permet la découverte du plus proche optimum local et l'enlèvement de la recherche. En pratique, une population qui a convergé ne se re-diversifie pas. Quand la diversité est épuisée et l'algorithme n'a pas obtenu des solutions jugées suffisamment bonnes, on parle de convergence prématurée, ce qui est peut être due à une forte pression sélective (biais plus ou moins important accordé au meilleur).

II.5.7 L'élitisme

L'élitisme est un opérateur optionnel particulier, il permet de garder l'individu le mieux adapté d'une génération à la suivante. En effet, l'opérateur de sélection peut ne pas le sélectionner, le croisement avec un autre individu peut donner des individus moins adaptés si les gènes ne sont pas bien recombinaison ou encore sa mutation peut également le rendre moins adapté [15].

II.5.8 Exploitation et exploration

Dans les problèmes d'optimisation, l'espace de recherche investit la région qui sera explorée durant le processus d'optimisation. L'espace de recherche inclut toutes les valeurs possibles que la décision peut supposer. Mais cet espace contient des solutions qui ne sont pas possibles. L'opérateur de sélection et de croisement, permettent la recherche locale dans le voisinage, ce qui favorise la concentration de la population dans des régions de l'espace de recherche où la fonction fitness a des valeurs élevées, de ce fait, ces deux opérateurs ont un rôle d'exploitation des meilleures solutions. Ceci peut mener à une perte de la diversité génétique et ainsi arriver à une convergence prématurée. Pour contourner ce problème l'opérateur de mutation permet d'éviter cette stagnation dans une zone particulière, ce qui agit comme une dispersion dans la population et permet d'explorer d'autres zones pour une recherche globale. Ce qui constitue un dosage délicat mais qui influence efficacement la convergence de l'AE vers l'optimum global.

II.5.9 Principes de sélection

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. On trouve dans la littérature un nombre important de principes de sélection plus ou moins adaptés aux problèmes qu'ils traitent. Les trois principes de sélection suivants sont les plus couramment utilisés [15]:

- Selection par Roulette (Roulette wheel selection);
- Stochastic remainder without replacement selection;
- Sélection par Tournoi

Le principe de sélection par *Roulette* consiste à associer à chaque individu un segment dont la longueur est proportionnelle à sa *fitness*. On reproduit ici le principe de tirage aléatoire utilisé dans les roulettes de casinos avec une structure linéaire. Ces segments sont ensuite concaténés sur un axe que l'on normalise entre 0 et 1. On tire alors un nombre aléatoire de distribution uniforme entre 0 et 1, puis on "regarde" quel est le segment sélectionné. Avec ce système, les grands segments, c'est-à-dire les bons individus, seront plus souvent choisis que les petits. Lorsque la dimension de la population est réduite, il est difficile d'obtenir en pratique l'espérance mathématique de sélection en raison du peu de tirages effectués.

Un biais de sélection plus ou moins fort existe suivant la dimension de la population.

La *Stochastic remainder without replacement selection* évite ce genre de problème. Décrivons ce principe de sélection :

- Pour chaque élément i , on calcule le rapport r_i de sa *fitness* sur la moyenne des *fitness*.
- Soit $e(r_i)$ la partie entière de r_i , chaque élément est reproduit exactement $e(r_i)$ fois.
- La *roulette wheel selection* précédemment décrite est appliquée sur les individus affectés des *fitness* $r_i - e(r_i)$.

Lorsque des populations de faible taille sont utilisées, ce principe de sélection s'avère généralement efficace dans les applications pratiques.

Enfin le principe de sélection par Tournoi est comme suit :

On sélectionne n individus aptes à la reproduction. Ces individus sont sélectionnés de la manière suivante :

K individus sont tirés au sort dans la population des n individus (K est un paramètre appelé taille du tournoi). Il existe différentes sélections par tournoi : déterministe ou probabiliste. Dans le cas du tournoi déterministe, le meilleur des K individus gagne le tournoi. Dans le cas probabiliste, chaque individu peut être choisi comme vainqueur avec une probabilité proportionnelle à sa fonction d'évaluation.

Il est tout à fait possible que certains individus participent à plusieurs tournois : s'ils gagnent plusieurs fois, ils auront donc droit d'être copiés plusieurs fois dans la génération intermédiaire, ce qui favorisera la pérennité de leurs gènes. La figure II.9 montre un exemple de sélection par tournoi entre deux individus (tournoi binaire).

Exemple de sélection par Tournoi.

Tournoi

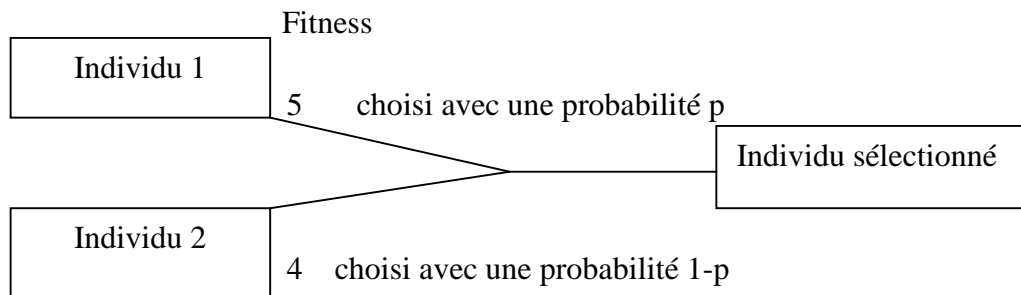


Figure II.9. Exemple de sélection par Tournoi.

II.5.10 Les paramètres d'un AG

La convergence d'un AG dépend fortement de certains paramètres qu'on doit fixer à l'avance. Dans ce qui suit nous présentons chacun de ces paramètres et leurs rôles dans le fonctionnement d'un algorithme génétique [16].

II.5.10.1 La taille de la population

Les conditions de convergence changent avec la taille de la population ; lorsque celle-ci est grande, sa diversité augmente ce qui diminue la convergence vers un optimum local. Mais le temps d'exécution de chaque génération augmente, et la recherche risque de s'effectuer d'une façon redondante et l'efficacité de l'algorithme est globalement affectée. Cette taille selon le cas se situe entre 25 et 100 individus. Par contre, si la taille de la population est petite, alors la probabilité de s'attarder sur des minima locaux est grande.

II.5.10.2 Le taux de croisement

Le *taux de croisement* détermine la proportion des individus qui sont croisés parmi ceux qui remplaceront l'ancienne génération. L'opérateur de croisement est appliqué avec une probabilité P_c , et plus cette valeur est grande plus de nouvelles structures (individus) sont introduites dans la nouvelle génération. En effet, si ce taux a été fixé à 1, tout descendant est obtenu par croisement. Mais quand ce taux est élevé, les structures performantes sont trop fréquemment détruites. Par contre, si ce taux est trop bas, la population n'évolue pas assez vite. En général P_c varie entre 0.7 et 1.0.

II.5.10.3 Le taux de mutation

L'opérateur de mutation est appliqué avec une probabilité P_m ; si ce taux est grand alors la recherche devient purement aléatoire et la population est diversifiée, et l'algorithme génétique perd de son efficacité. Si au contraire ce taux est faible, la population est moins diversifiée et en plus il y a risque de stagnation. Des études empiriques conseillent pour l'obtention de bons résultats une fréquence qui se situe autour d'une mutation tous les 1000 bits.

II.6 Avantages et inconvénients des algorithmes génétiques

II.6.1 Avantages

- Facile à implémenter,
- Possibilité d'incorporer d'autres méthodes (méthodes hybrides),
- Fournissent plusieurs alternatives (solutions),
- Bon pour des problèmes avec beaucoup d'optimum locaux (évite les optima locaux),
- L'espace de recherche peut être complètement exploré,
- Conceptuellement simple et flexible.

II.6.2 Inconvénients

- Parfois difficile de trouver :
 - Un bon codage,
 - Une bonne fonction d'adaptation,
 - De bons opérateurs de croisement et de mutation,
- Méthode lente, mais au moins, donne une solution en tout temps,
- Ne garantissent pas de trouver la solution optimale dans un temps fini (Aucune garantie sur la valeur des résultats),
- Moins efficace que d'autres méthodes (hill climbing, newton raphson, recuit simulé, la recherche tabou, branch and bound,...),
- Les AGs ne peuvent pas être utilisés directement comme régulateur en temps réel parce qu'ils sont trop lents, c à d qu'ils nécessitent beaucoup de temps pour fonctionner.

II.7 Conclusion

Ce chapitre constitue une introduction aux algorithmes génétiques et fournit les éléments nécessaires à leur programmation. Les Algorithmes Evolutionnaires sont basées sur la théorie de l'évolution de Darwin. Par analogie avec le monde biologique, l'algorithme fait évoluer une population de dispositifs à l'aide de divers opérateurs : sélection, croisements, mutations. Les algorithmes génétiques possèdent des caractéristiques d'évolution qui leurs permettent d'être très utiles à une grande classe de problèmes d'optimisation. Nous avons présenté le principe de cette méthode : initialisation avec une population de solutions aléatoires, caractériser par une fonction d'évaluation (d'adaptation), qui donnera pour chaque solution possible une valeur reflétant sa qualité pour résoudre le problème posé et faire la mise à jour des générations pour avoir l'optimum. Parmi les divers champs d'applications des AGs, on cite l'optimisation des porteurs de charges dans les semi-conducteurs qui sera discuté dans le chapitre prochain.