Chapitre V : Les Algorithmes Génétiques

1. Introduction

Les algorithmes génétiques (**AG**s) sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Leur fonctionnement est extrêmement simple. On part avec une population de solutions potentielles (*chromosomes*) initiales arbitrairement choisies. On évalue leur performance (*fitness*) relative. Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation. On recommence ce cycle jusqu'à ce que l'on trouve une solution satisfaisante. Les AGs ont été initialement développés par John Holland (1975). C'est au livre de Goldberg (1989) que nous devons leur popularisation.

Pour résumer, on distingue 4 principaux points qui font la différence fondamentale entre ces algorithmes et les autres méthodes :

1. Les algorithmes génétiques utilisent un codage des paramètres, et non les paramètres eux mêmes.
2. Les algorithmes génétiques travaillent sur une population de points, au lieu d'un point unique.
3. Les algorithmes génétiques n'utilisent que les valeurs de la fonction étudiée, pas sa dérivée, ou une autre connaissance auxiliaire.
4. Les algorithmes génétiques utilisent des règles de transition probabilistes, et non déterministes.

La simplicité de leurs mécanismes, la facilité de leur mise en application et leur efficacité même pour des problèmes complexes ont conduit à un nombre croissants de travaux

2. Présentation des algorithmes génétiques (AGs)

Selon Lerman et Ngouenet (1995) un algorithme génétique est défini par :

– *Individu/chromosome/séquence* : une solution potentielle du problème ;
– *Population* : un ensemble de chromosomes ou de points de l'espace de recherche ;
– *Environnement* : l'espace de recherche ;
– *Fonction de fitness* : la fonction - positive - que nous cherchons à maximiser.

Avant d'aller plus loin il nous faut définir quelques termes importants généralement définis sous l'hypothèse de codage binaire.

Définition 1 (Séquence/Chromosome/Individu (Codage binaire)).

Nous appelons une séquence (chromosome, individu) A de longueur l(*A*) *une suite* $A = \{a_1, a_2, \ldots, a_l\}$ *avec* $\forall\, l \in [1; l]; a_i \in V = \{0;1\}$.
Un chromosome est donc une suite de bits en codage binaire, appelé aussi chaîne binaire. Dans le cas d'un codage non binaire, tel que le codage réel, la suite *A* ne contient qu'un point, nous avons $A = \{a\}$ avec $a \in \Re$.

Définition 2 (Fitness d'une séquence).

Nous appelons fitness d'une séquence toute valeur positive que nous noterons f (A), où f est typiquement appelée fonction de fitness.
La fitness (efficacité) est donc donnée par une fonction à valeurs positives réelles. Dans le cas d'un codage binaire, nous utiliserons souvent une fonction de décodage *d* qui permettra de passer d'une chaîne binaire à un chiffre à valeur réelle : $d : \{0;1\}^l \rightarrow \Re$ (où l est la longueur de la chaîne). La fonction de fitness est alors choisie telle qu'elle transforme cette valeur en valeur positive, soit $f : d(\{0;1\}^l) \rightarrow \Re_+$. Le but d'un algorithme génétique est alors simplement de trouver la chaîne qui maximise cette fonction *f*. Bien évidemment, chaque problème particulier nécessitera ses propres fonctions *d* et *f* .

Les AGs sont alors basés sur les phases suivantes :

1. **Initialisation**. Une population initiale de *N* chromosomes est tirée aléatoirement.
2. **Évaluation**. Chaque chromosome est décodé, puis évalué.
3. **Sélection**. Création d'une nouvelle population de *N* chromosomes par l'utilisation d'une méthode de sélection appropriée.
4. **Reproduction**. Possibilité de croisement et mutation au sein de la nouvelle population.
5. **Retour** à la phase d'évaluation jusqu'à l'arrêt de l'algorithme.

Ou alors sous une forme algorithmique :

```
Générer une population aléatoire de n chromosomes
Tant que Continue Faire
    Calculer la fonction fitness f (x), pour tout chromosome x
    Tant que NouvellePopulationARemplir Faire
        Sélectionner deux chromosomes parents
        En faire un nouveau chromosome
        Lui imposer des mutations éventuelles
        Ajoute le nouveau chromosome à la nouvelle population
    Fin Tant que
    Remplace l'ancienne population par la nouvelle
Fin Tant que
```

Voyons maintenant plus en détail les autres phases de l'algorithme génétique. Nous présentons ces opérateurs sous l'hypothèse implicite que le codage est binaire.

2.1. Codage de la population initiale

Il existe trois principaux type de codage : binaire, *gray* ou réel. Nous pouvons facilement passer d'un codage à l'autre. Certains auteurs n'hésitent pas à faire le parallèle avec la biologie et parlent de génotype en ce qui concerne la représentation binaire d'un individu, et de phénotype pour ce qui est de sa valeur réelle correspondante dans l'espace de recherche.

Rappelons que la transformation la plus simple (fonction de décodage *d*) d'une chaîne binaire *A* en nombre entier *x* s'opère par la règle suivante :

$$x = d(A) = \sum_{i=1}^{l} a_i 2^{l-i}$$

Ainsi le chromosome $A = \{1;0;1;1\}$ vaut $1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = 8+2+1 = 11$.

Évidemment, la fonction *d* sera modifiée selon le problème. Ainsi si nous cherchons à maximiser une fonction $f : [0;1] \rightarrow [0;1]$ une méthode possible serait la suivante (la taille du chromosome dépendant bien évidemment de la précision voulue) :

$$x = d(A) = \sum_{i=1}^{l} a_i 2^{-i}$$

Pour une précision au cinquième chiffre après la virgule nous prenons alors $l = 16$ puisque :

$$d(\{\underbrace{1,..,1,..,1}_{16}\}) = \quad 0.999992$$

Une autre façon de faire est de choisir d telle que :

$$x = d(A) = \sum_{i=1}^{l} \frac{a_i 2^{l-i}}{2^l - 1}$$

Avec l=16 ous avons 2^l= 65536 et d{1,…….,1}= $\frac{65535}{65535}$ =1, la precision est verifiée.

Cette dernière règle peut se généraliser. Ainsi, admettons que nous cherchons à maximiser f en fonction d'une variable réelle x. Soit $D = [xmin;xmax]$, avec $D \subset \Re$, l'espace de recherche permis avec $xmin$ et $xmax$ les bornes inférieures et supérieures.

Soit $prec$ la précision (chiffre après la virgule) avec laquelle nous cherchons x. Soit $ld = xmax - xmin$ la longueur de l'intervalle D.

Nous devons alors diviser cet intervalle en $ni = ld * 10^{prec}$ sous-intervalles égaux afin de respecter la précision.

Par exemple, soit $D = [-1;2]$, nous avons donc $ld = 3$, si nous voulions une précision $prec$ =6, alors il nous faut diviser cet intervalle en ni =3000000 sous-intervalles. Avec s l'entier naturel tel que $2^s > ni$ (dans notre exemple, $s = 222$ car $221 = 2097152 < 3000000 < 222 = 4194304$), la transformation d'une chaîne binaire $A = \{a_1,\ldots,a_s\}$ en un nombre réel x peut alors s'exécuter en deux étapes :

1- conversion (base 2 en base 10) : $x' = \sum_{i=1}^{s} 2^{i-1}$
2- recherche du nombre réel correspondant :

$$x = x_{min} + x' \frac{x_{max} - x_{min}}{2^s - 1}$$

ou ce qui revient au même directement en une seule étape par :

$$x = x_{min} + \sum_{i=1}^{s} \frac{2^{i-1} ld}{2^s - 1}$$

Pour ce qui est de la phase d'initialisation, la procédure est assez simple. Elle consiste en un tirage aléatoire de N individus dans l'espace des individus permis. En codage binaire, selon la taille l de la chaîne, nous effectuons pour un chromosome l tirage dans $\{0;1\}$ avec équiprobabilité.

2.2. Les opérateurs

Les opérateurs jouent un rôle prépondérant dans la possible réussite d'un AG. Nous en dénombrons trois principaux : l'opérateur de sélection, de croisement et de mutation. Si le principe de chacun de

ces opérateurs est facilement compréhensible, il est toutefois difficile d'expliquer l'importance isolée de chacun de ces opérateurs dans la réussite de l'AG. Cela tient pour partie au fait que chacun de ces opérateurs agit selon divers critères qui lui sont propres (valeur sélective des individus, probabilité d'activation de l'opérateur, etc.).

Operateur de sélection

Cet opérateur est peut-être le plus important puisqu'il permet aux individus d'une population de survivre, de se reproduire ou de mourir. En règle générale, la probabilité de survie d'un individu sera directement reliée à son efficacité relative au sein de la population.

Il existe plusieurs méthodes pour la reproduction. La méthode la plus connue et utilisée est sans nul doute, la roue de loterie biaisée (*roulette wheel*) de Goldberg (1989). Selon cette méthode, chaque chromosome sera dupliqué dans une nouvelle population proportionnellement à sa valeur d'adaptation. On effectue, en quelque sorte, autant de tirages avec remises qu'il y a d'éléments dans la population. Ainsi, dans le cas d'un codage binaire, la fitness d'un chromosome particulier étant f ($d(ci)$), la probabilité avec laquelle il sera réintroduit dans la nouvelle population de taille N est :

$$\frac{f(d(c_i))}{\sum_{j-1}^{N} f(d(c_j))}$$

Les individus ayant une grande fitness ont donc plus de chance d'être sélectionnés. On parle alors de sélection proportionnelle. L'inconvénient majeur de cette méthode repose sur le fait qu'un individu n'étant pas le meilleur peut tout de même dominer la sélection. Elle peut aussi engendrer une perte de diversité par la domination d'un super individu. Un autre inconvénient est sa faible performance vers la fin quand l'ensemble des individus se ressemblent.

Une solution à ce problème ne tient pas dans l'utilisation d'une autre méthode de sélection mais dans l'utilisation d'une fonction de fitness modifiée. Ainsi, nous pouvons utiliser un changement d'échelle (*scaling*) afin de diminuer ou accroître de manière artificielle l'écart relatif entre les fitness des individus.

Brièvement, il existe d'autres méthodes, la plus connue étant celle du tournoi (*tournament selection*) : on tire deux individus aléatoirement dans la population et on reproduit le meilleur des deux dans la nouvelle population. On refait cette procédure jusqu'à ce que la nouvelle population soit complète. Cette méthode donne de bons résultats. Toutefois, aussi important que soit la phase de sélection, elle ne crée pas de nouveaux individus dans la population. Ceci est le rôle des opérateurs de croisement et de mutation.

Operateur de croisement

L'opérateur de croisement permet la création de nouveaux individus selon un processus fort simple. Il permet donc l'échange d'information entre les chromosomes (individus). Tout d'abord, deux individus, qui forment alors un couple, sont tirés au sein de la nouvelle population issue de la reproduction. Puis un (potentiellement plusieurs) site de croisement est tiré aléatoirement (chiffre entre 1 et $l-1$). Enfin, selon une probabilité p_c que le croisement s'effectue, les segments finaux (dans le cas d'un seul site de croisement) des deux parents sont alors échangés autour de ce site (voir figure 2).
Cet opérateur permet la création de deux nouveaux individus. Toutefois, un individu sélectionné lors de la reproduction ne subit pas nécessairement l'action d'un croisement. Ce dernier ne s'effectue qu'avec une certaine probabilité. Plus cette probabilité est élevée et plus la population subira de changement.
Quoi qu'il en soit, il se peut que l'action conjointe de la reproduction et du croisement soit insuffisante pour assurer la réussite de l'AG. Ainsi, dans le cas du codage binaire, certaines informations (i.e.

caractères de l'alphabet) peuvent disparaître de la population. C'est pour remédier entre autre à ce problème que l'opérateur de mutation est utilisé.

Operateur de mutation

Le rôle de cet opérateur est de modifier aléatoirement, avec une certaine probabilité, la valeur d'un composant de l'individu. Dans le cas du codage binaire, chaque bit $a_i \in \{0;1\}$ est remplacé selon une probabilité p_m par son inverse $a'_i = 1 - a_i$. C'est ce qu'illustre la figure 5.1. Tout comme plusieurs lieux de croisement peuvent être possibles, nous pouvons très bien admettre qu'une même chaîne puisse subir plusieurs mutations.

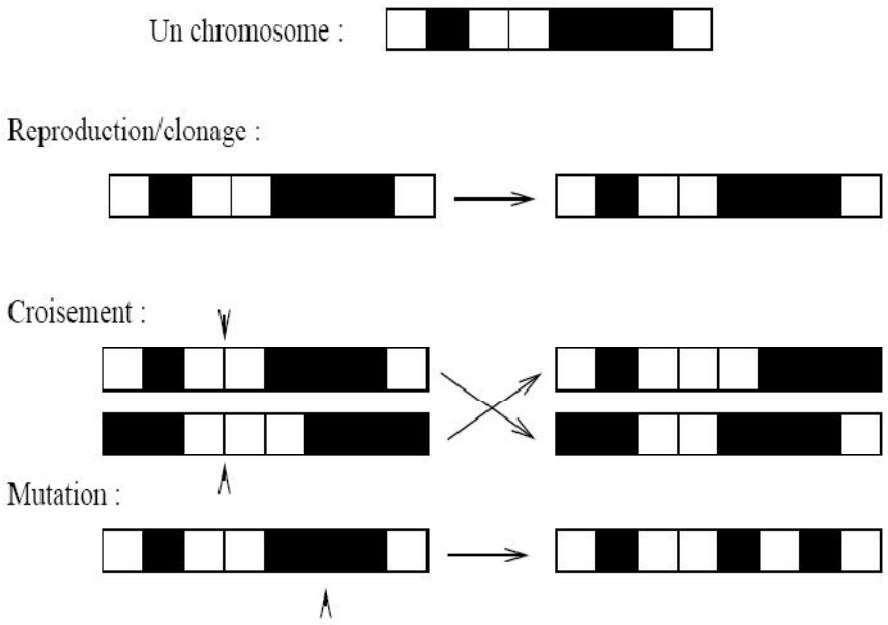


Fig 5.1. Reproduction clonage et mutation

La mutation est traditionnellement considérée comme un opérateur marginal bien qu'elle confère en quelque sorte aux algorithmes génétiques la propriété d'ergodicité (i.e. tous les points de l'espace de recherche peuvent être atteints). Cet opérateur est donc d'une grande importance. Il a de fait un double rôle : celui d'effectuer une recherche locale et/ou de sortir d'une trappe (recherche éloignée).

2.3. Autres paramètres

Les opérateurs de l'algorithme génétique sont guidés par un certain nombre de paramètres fixés à l'avance.
La valeur de ces paramètres influence la réussite ou non d'un algorithme génétique. Ces paramètres sont les suivants :

- La taille de la population, *N*, et la longueur du codage de chaque individu *l* (dans le cas du codage binaire). Si *N* est trop grand le temps de calcul de l'algorithme peut s'avérer très important, et si *N* est trop petit, il peut converger trop rapidement vers un mauvais chromosome. Cette importance de la taille est essentiellement due à la notion de *parallélisme implicite* qui implique que le nombre d'individus traité par l'algorithme est au moins proportionnel au cube du nombre d'individus.

- La probabilité de croisement p_c. Elle dépend de la forme de la fonction de fitness. Son choix est en général heuristique (tout comme pour p_m). Plus elle est élevée, plus la population subit de changements importants. Les valeurs généralement admises sont comprises entre 0*:*5 et 0*:*9.
- La probabilité de mutation p_m. Ce taux est généralement faible puisqu'un taux élevé risque de conduire à une solution sous-optimale.

Plutôt que de réduire p_m, une autre façon d'éviter que les meilleurs individus soient altérés est d'utiliser la reconduite explicite de l'élite dans une certaine proportion. Ainsi, bien souvent, les meilleurs 5%, par exemple, de la population sont directement reproduits à l'identique, l'opérateur de reproduction ne jouant alors que sur les 95% restant. Cela est appelée une stratégie élitiste.

2.4. Remarque sur la fonction de fitness

Le choix de la fonction de fitness retenue est important et dépend du problème à résoudre et de l'espace de recherche qui en découle. Admettons que l'on cherche simplement à maximiser la fonction: $f(x) = x2$ avec $x\ 2\ [0;10]$.

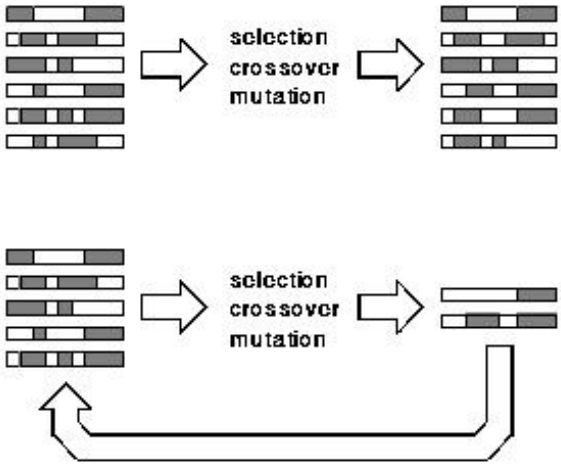
Dans un tel cas, la fonction de fitness coïncide avec celle du problème. Pour des problèmes de minimisation simples, tel $\min(x-3)^2$, nous utiliserons en général soit la propriété que : $\max f(x) = -min\ f(x)$ ou bien, si la fonction est bornée supérieurement la fonction : $\max C - f(x)$, où C est une constante positive supérieure à cette borne.

Cependant l'espace de recherche, appelons le S, est généralement constitué de deux sous-espaces disjoints : l'espace des solutions admissibles F et l'espace des solutions non admissibles U. De nombreux problèmes de programmation linéaire ou non linéaire n'échappent pas à ce problème. A tout moment, en cherchant un maximum faisable, l'algorithme génétique peut au cours du processus de recherche créée des solutions non admissibles, solutions qui violeraient au moins l'une des contraintes. Il n'est jamais simple de traiter ces problèmes. La solution passe en général par l'utilisation d'une fonction de fitness à pénalité. L'efficacité d'une solution non admissible est automatiquement réduite. Toutefois, le choix de cette fonction de pénalité soulève des questions : comment deux solutions non admissibles doivent-elles être comparées ? Devons-nous considérer que toute solution admissible est préférable à une solution non admissible ? Devons-nous automatiquement supprimer les solutions non admissibles de la population ? Pouvons-nous par une fonction dite de réparation changée une solution non admissible en une admissible ? etc. Toutes ces questions reviennent à poser celle de l'utilisation de solutions non admissibles pour leurs potentiels d'information concernant la recherche de l'optimum : une solution non admissible pouvant être plus proche de la solution optimal que de nombreuses autres solutions admissibles. Toutes ces questions n'ont pas encore de solution unanime (voir Michalewicz (1995) pour un résumé)

2.5. Régimes de remplacement

Il existe deux régimes de remplacement de la population initiale, bien sure après croisement et mutation, à savoir Régime simple et Régime permanent.

- En régime simple toute la population précédente est remplacée par la population enfants.
- En régime permanent, une partie de la population parent est remplacé par une partie de la population enfants. On tend a garder les parents les plus fort (ceux avec une fonction fitness élevé) mais pas toujours var il est bon de garder quelques individus faibles qui pourront donner, si ils sont sélectionnés des enfants performants.



3. Exemple

Nous reprenons ici l'exemple de Goldberg (1989). Il consiste à trouver le maximum de la fonction $f(x)=x$ sur l'intervalle [0;31] où x est un entier. La première étape consiste à coder la fonction. Par exemple, nous utilisons un codage binaire de x, la séquence (chromosome) contenant au maximum 5 bits. Ainsi, nous avons $x = 2 \rightarrow \{0;0;0;1;0\}$, de même $x = 31 \rightarrow \{1;1;1;1;1\}$. Nous recherchons donc le maximum d'une fonction de fitness dans un espace de 32 valeurs possibles de x.

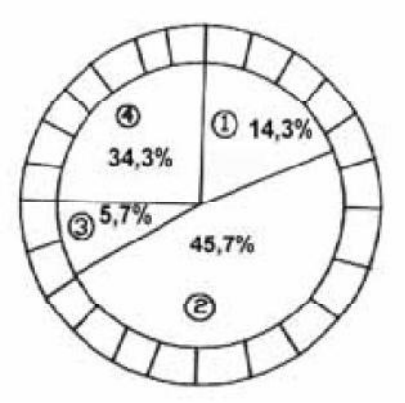
3.1. Tirage et évaluation de la population initiale

Nous fixons la taille de la population à $N = 4$. Nous tirons donc de façon aléatoire 4 chromosomes sachant qu'un chromosome est composé de 5 bits, et chaque bit dispose d'une probabilité 1/2 d'avoir une valeur 0 ou 1.
Le maximum, 16, est atteint par la deuxième séquence. Voyons comment l'algorithme va tenter d'améliorer ce résultat.

Numéro	Séquence	Fitness	% du total
1	00101	5	14.3
2	10000	16	45.7
3	00010	2	5.7
4	01100	12	34.3
Total		35	100

3.2. Sélection

Une nouvelle population va être créée à partir de l'ancienne par le processus de sélection de la roue de loterie biaisée.



Numéro	Séquence
1	10000
2	01100
3	00101
4	10000

TAB. 1: Nouvelle Population

3.3. Croisement

Les parents sont sélectionnés au hasard. Nous tirons aléatoirement un lieu de croisement (site ou *locus*) dans la séquence. Le croisement s'opère alors à ce lieu avec une probabilité p_c. Le tableau 2 donne les conséquences de cet opérateur en supposant que les chromosomes 1 et 3, puis 2 et 4 sont appariés et qu'à chaque fois le croisement s'opère (par exemple avec $p_c = 1$).

$l = 3$	$l = 2$
100|00	01|100
001|01	10|000
10001	01000
00100	10100

TAB. 2: Le croisement

3.4. Mutation

Dans cet exemple à codage binaire, la mutation est la modification aléatoire occasionnelle (de faible probabilité) de la valeur d'un bit (inversion d'un bit). Nous tirons ainsi pour chaque bit un chiffre aléatoire entre 0 et 1 et si ce chiffre est inférieur à p_m alors la mutation s'opère. Le tableau 3, avec p_m = 0:05, met en évidence ce processus.

Anc. Chr.	Tirage aléat.	Nveau Bit	Nveau Chr.
10001	15 25 36 **04** 12	1	10**1**1
00100	26 89 13 48 59	–	00100
01000	32 45 87 22 65	–	01000
10100	47 **01** 85 62 35	1	**1**1100

TAB. 3: La mutation

Maintenant que la nouvelle population est entièrement créée, nous pouvons de nouveau l'évaluer.

3.5. Evaluation

Le maximum est maintenant de 28 (séquence 4). Nous sommes donc passés de 16 à 28 après une seule génération. Bien sûr, nous devons recommencer la procédure à partir de l'étape de sélection jusqu'à ce que le maximum global, 31, soit obtenu, ou bien qu'un critère d'arrêt ait été satisfait.

Numéro	chaîne	Fitness	% du total
1	10011	19	32.2
2	00100	4	6.8
3	01000	8	13.5
4	11100	28	47.5
Total		59	100

TAB. 4: Nouvelle évaluation

Références

Goldberg, D. (1989), *Genetic Algorithm In Search, Optimization And Machine Learning*, Addison-Wesley.

Holland, J. H. (1975), *Adaptation In Natural And Artificial Systems*, University of Michigan Press.

Lerman, I. & Ngouenet, F. (1995), Algorithmes génétiques séquentiels et parallèles pour une représentation affine des proximités, Rapport de Recherche de l'INRIA Rennes - Projet REPCO 2570, INRIA.

Michalewicz, Z. (1995), A survey of constraint handing techniques in evolutionary computation methods. *Proceedings of the 4th Annual Conference on Evolutionary Programming*, MIT Press.