Chapitre 1

L'algorithme génétique

L'algorithme génétique (AG) est un algorithme de recherche basé sur les mécanismes de la sélection naturelle et de la génétique. Il combine une stratégie de "survie des plus forts" avec un échange d'information aléatoire mais structuré. Pour un problème pour lequel une solution est inconnue, un ensemble de solutions possibles est créé aléatoirement. On appelle cet ensemble la population. Les caractéristiques (ou variables à déterminer) sont alors utilisées dans des séquences de gènes qui seront combinées avec d'autres gènes pour former des chromosomes et par après des individus. Chaque solution est associée à un individu, et cet individu est évalué et classifié selon sa ressemblance avec la meilleure, mais encore inconnue, solution au problème. Il peut être démontré qu'en utilisant un processus de sélection naturelle inspiré de Darwin, cette méthode convergera graduellement à une solution.

Comme dans les systèmes biologiques soumis à des contraintes, les meilleurs individus de la population sont ceux qui ont une meilleure chance de se reproduire et de transmettre une partie de leur héritage génétique à la prochaine génération. Une nouvelle population, ou génération, est alors créée en combinant les gènes des parents. On s'attend à ce que certains individus de la nouvelle génération possèdent les meilleures caractéristiques de leurs deux parents, et donc qu'ils seront meilleurs et seront une meilleure solution au problème. Le nouveau groupe (la nouvelle génération) est alors soumis aux même critères de sélection, et par après génère ses propres rejetons. Ce processus est répété plusieurs fois, jusqu'à ce que tous les individus possèdent le même héritage génétique. Les membres de cette dernière génération, qui sont habituellement très différents de leurs ancêtres, possèdent de l'information génétique qui correspond à la meilleure solution au problème.

L'algorithme génétique de base comporte trois opérations simples qui ne sont pas plus compliquées que des opérations algébriques :

- Sélection
- Reproduction
- Mutation

L'algorithme génétique fut développé par Holland [1].

1.1 Codage d'individus

Avant de passer à l'explication des différents processus génétiques, il faut tout d'abord expliquer le codage des individus. La procédure normale pour coder un algorithme génétique ayant plusieurs paramètres est de coder chaque paramètre comme une séquence de bits. Les séquences sont ensuite tronquées l'une après l'autre pour former une grande séquence, le chromosome, qui représente le vecteur des paramètres. Chaque séquence du vecteur total représente un gène, et la valeur de chaque gène est un allèle.

Example : Soit un vecteur $\overline{x}$ composé de trois paramètres x_1, x_2, et x_3, codés à 4 bits.

$$x_1 = 0011$$
$$x_2 = 1010$$
$$x_3 = 0100$$

La séquence totale serait la suivante :

$$\overline{x} = \{\overset{x_1}{0011}|\overset{x_2}{1010}|\overset{x_3}{0100}\}$$

Un chiffre codé en binaire dans ce cas ne représente pas une valeur spécifique mais plutôt un intervalle. Par exemple, si on code des valeurs de 0 à 1 à l'aide de 5 bits, le chiffre 11111 représente l'intervalle de 31/32 à 32/32, plutôt que la valeur de 31/32 exactement.

1.2 Opérateurs génétiques

Sélection : processus où les individus sont copiés selon la valeur de leur fonction objective f. On peut décrire la fonction f comme une mesure de profit, utilité ou qualité que l'on veut maximiser (minimiser). Si on copie des individus selon leur valeur f, ceci implique que les individus ayant des valeurs plus élevées ont une plus grande probabilité de contribuer des rejetons à la prochaine génération. Ceci correspond à une version artificielle de la "survie des plus forts" (*survival of the fittest*) de Darwin.

L'implantation de la sélection peut se faire de plusieurs façons. La plus facile est peut-être la roue de roulette biaisée, où la probabilité de reproduction d'un individu dépend de sa valeur par rapport au total des valeurs de la population.

Reproduction : processus où de nouveaux individus sont formés à partir de parents. Ces nouveaux individus, les rejetons, sont formés en effectuant un croisement entre deux parents. On choisit une position aléatoire k entre $[1, l-1]$ où l est la longueur de l'individu. Le croisement se fait en échangeant les bits de la position $k+1$ à l.

Exemple : Soit $k = 4$ pour deux parents (P_1 et P_2) codés à 5 bits (donc $l = 5$). Les rejetons sont O_1 et O_2.

$$\left.\begin{array}{l} P_1 = 0110|1 \\ P_2 = 1100|0 \end{array}\right\} \begin{array}{l} O_1 = 01100 \\ O_2 = 11001 \end{array}$$

On voit bien l'échange qui s'est produit ici, le bit 5 ($k+1$) a passé d'un individu à l'autre, pour ainsi former deux nouveaux individus(O_1 and O_2).

Ce sont ces deux opérations, la sélection et la reproduction, qui sont à la base des algorithmes génétiques. Ceci peut paraître simple à première vue, puisque aucune opération mathématique complexe n'a été effectuée. Mais on peut comparer le processus précédent à l'innovation humaine : souvent, les découvertes n'arrivent pas par chance. Elles sont le résultat d'un échange d'idées qui crée d'autres idées et finalement mènent à une solution désirée.

Mutation : processus aléatoire où un bit change de valeur. Ce processus joue un rôle secondaire dans l'algorithme génétique, mais il est quand même important. La mutation assure qu'aucun point dans l'espace de recherche a une probabilité nulle d'être atteint.

Exemple : mutation du bit 2.

$$A_1 = 00101 \Rightarrow A_1^{'} = 01101$$

Une explication plus complète de ces phénomènes ainsi qu'une preuve théorique de leur performance sont disponibles dans le livre de Goldberg [2].

1.2.1 Du codage binaire au codage réel

Les premières techniques d'utilisation de l'algorithme génétique utilisaient un codage binaire pour coder les individus. Mais le codage binaire entraîne certains problèmes, et le codage réel est plus précis (Wright, [3]).

Il y a plusieurs raisons pour lesquelles il est préférable de coder l'algorithme génétique en nombres réels. Une raison est que le codage réel permet une plus grande marge de valeurs possibles des paramètres. Par exemple, si on code à 5 bits, il n'est possible d'obtenir que $2^5 = 32$ différentes valeurs. Si on augmente le codage à 10 bits, il n'est possible d'avoir que 1024 différentes valeurs pour les paramètres. Pour des calculs où chaque point décimal est important, où une petite différence dans une valeur varie la performance de façon assez drastique, le codage réel a de grands avantages.

On se réfère ici à un article de M. Alden H. Wright [3], qui fut le premier à proposer une stratégie pour coder l'algorithme génétique en nombres réels. Son travail est la base de l'algorithme utilisé ici. Les sections suivantes résument son travail.

1.2.2 Sélection

Le processus de sélection n'est pas modifié par le codage réel, puisque aucune opération algébrique n'est effectuée sur la séquence elle-même. Les meilleurs individus sont encore ceux qui ont la meilleure performance et sont ceux qui ont la meilleure chance de se reproduire. C'est au niveau du croisement qu'il faut regarder pour observer les effets du codage réel.

1.2.3 Reproduction (croisement)

La technique principale de croisement en codage binaire est la coupure d'une séquence en deux parties, et l'échange de ces deux parties. Un rejeton reçoit la partie droite du

parent 1 et la partie gauche du parent 2 ; le deuxième rejeton reçoit la partie gauche du parent 1 et la partie droite du parent 2. Comme on a vu dans la section précédente, le croisement se fait comme suit : soit deux parents $P_1 = 011|10010$ et $P_2 = 100|11110$, avec $k = 3$, on obtient les deux rejetons $O_1 = 01111110$ et $O_2 = 10010010$.

Pour voir l'effet du croisement binaire (le croisement effectué sur des séquences), on considère en premier le cas où le point de croisement tombe entre les bits de deux paramètres. Dans ce cas, le rejeton reçoit quelques-uns de ses paramètres d'un parent et les autres de l'autre parent.

Exemple : Deux séquences $\overline{p_1} = (x_1, x_2, \ldots, x_m)$ et $\overline{p_2} = (y_1, y_2, \ldots, y_m)$ sont deux parents. Le point de croisement (choisi aléatoirement) se situe entre x_i et x_{i+1}. Les deux rejetons seront :

$$O_1 = (x_1, x_2, \ldots, x_i, y_{i+1}, \ldots, y_m)$$
$$O_2 = (y_1, y_2, \ldots, y_i, x_{i+1}, \ldots, x_m)$$

Dans ce cas, le croisement binaire est le même que le croisement réel, puisqu'on n'a fait qu'échanger des paramètres. Aucune opération n'a été effectuée sur des chiffres.

Croisement entre bits

Si le point de croisement se situe entre les bits d'un paramètre, la partie du code binaire à gauche du point de croisement correspond aux bits les plus significatifs et la partie de droite aux bits moins significatifs. Donc le rejeton reçoit une partie plus importante d'un parent et une partie moins significative de l'autre parent.

On peut donc considérer le rejeton comme une perturbation du premier parent, où l'amplitude de la perturbation est déterminée par la différence entre les bits moins significatifs des parents. Si le point de croisement est entre les bits k et $k+1$, la perturbation équivaut à changer les bits $k+1$ à n d'un des parents (pour une séquence codée à n bits). Si $S_i = b_i - a_i$ est l'intervalle du paramètre, la perturbation maximale est de $S_i 2^{-k}$. Afin de mieux illustrer ce point, on donne l'exemple suivant :

Soit $S_i = 0$ à 1, codé à 5 bits. Deux parents, P_1 et P_2 sont

$$P_1 = 00101 \text{ ou } 5/32$$
$$P_2 = 11011 \text{ ou } 27/32$$

Si le point de croisement est entre les bits 3 et 4, les deux rejetons sont :

$$O_1 = 01111 \text{ ou } 7/32$$
$$O_2 = 11001 \text{ ou } 25/32$$

Les bits moins significatifs des parents sont 01 et 11 respectivement, et la différence entre eux est de $\pm 2/32$. On peut donc écrire que les rejetons sont

$$O_1 = P_1 + 2/32$$
$$O_2 = P_2 - 2/32$$

On remarque que chaque rejeton est égal à son parent principal plus une perturbation de $\pm 2/32$. Pour cet exemple, la perturbation maximale est de

$$\frac{1-0}{2^3} = \frac{1}{8} = \frac{4}{32}$$

ce qui correspond à changer les deux derniers bits de 00 à 11 ou vice versa.

On peut donc conclure que le croisement est un échange de paramètres x_i, x_{i+1}, suivi d'une perturbation du paramètre x_i d'au plus $S_i 2^{-k}$, où k est le point de croisement. Ce croisement est appelé le croisement réel.

Mutation

Dans un codage binaire, lorsqu'il y a mutation, les bits sont changés de 0 à 1 ou de 1 à 0. Lorsqu'un bit subit une mutation, on peut le percevoir comme une perturbation du paramètre réel. L'amplitude de la perturbation dépend du bit qui est modifié. Soit $S_i = b_i - a_i$ l'intervalle du paramètre x_i. En binaire, la mutation du k^e bit correspond à une perturbation de $S_i 2^{-k}$. La direction de la perturbation dépend du bit qui est modifié. Si le bit change de 0 à 1, la perturbation est positive. Si le bit change de 1 à 0, la perturbation est négative.

Exemple : Soit $x_i = 10010$ un paramètre codé à 5 bits. L'intervalle de ce paramètre est de 0 à 32 ($S_i = 32$). Donc x_i représente le chiffre 18. Si le second bit est modifié, $x_i = 11010$, ou 26. La perturbation est positive, puisque le bit a changé de 0 à 1. L'amplitude du changement est $32 \cdot 2^{-2} = 8$, ce qui correspond à un changement de 18 à 26 (une augmentation de 8).

1.3 Le codage réel

Les deux seules opérations qui sont différentes des opérations binaires lorsqu'on code en réel sont la reproduction (croisement) et la mutation.

1.3.1 Croisement entre paramètres

Si on commence avec une population finie, le croisement entre paramètres ne permet d'atteindre qu'un nombre fini de points dans l'espace de recherche, soit ceux où les paramètres sont choisis parmi les paramètres de la population. En fait, si la population initiale est de taille n, on peut atteindre au plus n^m vecteurs de paramètres avec le croisement, où m est le nombre de variables. Donc un des buts de la mutation est de permettre d'atteindre un point arbitraire dans l'espace.

On peut voir le problème du croisement entre paramètres dans la figure 2. Les ellipses de la figure 2 représentent des lignes de contour de la fonction objective. Un minimum se trouve au centre de l'ellipse interne. Les points 1 et 2 sont de bons points, mais n'importe quel point qui sera créé par croisement entre paramètres sera moins bon que les deux parents.

Pour contourner ce problème, Wright propose une autre forme de croisement qu'il appelle le *croisement linéaire* : de deux parents P_1 et P_2, trois rejetons sont générés de la façon suivante : :

$$O_1 = 0.5 \cdot P_1 + 0.5 \cdot P_2$$
$$O_2 = 1.5 \cdot P_1 - 0.5 \cdot P_2$$
$$O_3 = -0.5 \cdot P_1 + 1.5 \cdot P_2$$

Les deux meilleurs rejetons sont choisis, ce qui conserve la taille de la population.

Cette technique n'est pas compatible avec le Théorème des Schématas (la preuve de la performance des algorithmes génétiques ; Goldberg, [2]) mais elle donne de bons résultats, des résultats qui sont meilleurs que l'algorithme codé en binaire.

1.3.2 Mutation codée réelle

Pour concevoir un opérateur de mutation, il faut premièrement se demander si le point dans R^m (le vecteur des paramètres) doit subir un changement ou si chaque paramètre doit subir une mutation.

Il est plus difficile de faire correspondre une mutation en R^m au Théorème des Schématas, bien que celle-ci semble donner de meilleurs résultats (Schwefel, 1981 et Matyas, 1965). La mutation effectuée par Wright est faite sur chaque paramètre.

Il y a alors deux problèmes :

1. Il faut choisir l'amplitude des mutations.
2. Le point muté doit demeurer dans l'espace de recherche.

Deux nouvelles valeurs sont alors définies :

1. Une probabilité de mutation p_m.
2. Une amplitude maximale de changement.

Si un paramètre est choisi pour subir une mutation, la direction (positive ou négative) de la mutation est choisie aléatoirement avec une probabilité de ½ (50%). Il faut ensuite déterminer l'amplitude du changement. Wright utilise une mutation uniforme.

Soit M, l'amplitude maximale du changement pour un paramètre x_i défini dans l'interval $a < x_i < b$.

– Si la mutation est positive, x' = nombre aléatoire dans $[x_i, \min(M, b)]$
– Si la mutation est négative, x' = nombre aléatoire dans $[\max(M, a), x_i]$

Le chiffre x' est choisi aléatoirement avec une distribution uniforme.

Cette mutation ne correspond pas trop à la mutation binaire, mais les résultats obtenus par Wright sont quand même bons. La mutation binaire favorise de plus petits changements, surtout si les paramètres sont codés avec plusieurs bits. Par contre, la mutation de Wright est utilisée par presque tous ceux qui codent l'algorithme génétique en réel.

1.4 Implantation de l'algorithme génétique

L'algorithme génétique de base fut expliqué dans les sections précédentes. Bien qu'il puisse être implanté tel qu'il fut présenté, l'algorithme utilisé dans ce travail diffère un peu et comporte certaines modifications.

En premier lieu, comme dans tous les problèmes d'optimisation, il faut bien définir le problème à optimiser. Il faut bien définir la quantité à optimiser, et aussi savoir quels paramètres on veut déterminer.

La première chose à faire est de définir le nombre de paramètres à déterminer et les bornes de ces paramètres.

1.4.1 Population initiale

Lorsqu'on crée une population initiale, les valeurs des paramètres (les allèles des gènes) sont créées aléatoirement avec une distribution uniforme sur l'intervalle spécifié. Comme exemple, on considère une population composée de n individus. On veut optimiser m variables $(x_1, x_2, \ldots, x_m)$, donc chaque chromosome aura m gènes. Donc la population est :

$$Pop = \begin{cases} I_1 = (x_1^1, x_2^1, \ldots, x_m^1) \\ I_2 = (x_1^2, x_2^2, \ldots, x_m^2) \\ I_3 = (x_1^3, x_2^3, \ldots, x_m^3) \\ \vdots \\ I_n = (x_1^n, x_2^n, \ldots, x_m^n) \end{cases}$$

Chaque individu I_i est composé d'un chromosome à m gènes dont les valeurs sont choisies aléatoirement. Normalement, chaque paramètre d'un individu a sa valeur propre (puisqu'on crée les individus de façon aléatoire), c'est-à-dire,

$$x_i^1 \neq x_i^2 \neq \ldots \neq x_i^n \text{ où } i = [1, 2, \ldots, m]$$

Une fois la population initiale créée, il faut ensuite procéder à la première étape du calcul, la sélection.

1.4.2 Sélection

On évalue la performance de chaque individu à l'aide de la fonction objective. La performance est une mesure de l'exactitude de la solution obtenue avec les paramètres d'un individu. Cette fonction objective varie d'un problème à l'autre. Elle varie aussi si on essaie de maximiser ou de minimiser un problème. Elle est calculée à partir des valeurs des paramètres de chaque individu. On obtient donc une valeur de fonction objective pour chaque individu, donc n valeurs au total pour l'ensemble de la population.

Si on essaie de minimiser une fonction, la fonction objective prend souvent la forme d'une fonction d'erreur telle que : :

$$f_{obj} = (\Delta_m - \Delta_c)^2 \tag{1.1}$$

où Δ_m est la valeur mesurée et Δ_c est la valeur calculée. La fonction objective est mise au carré pour éviter des valeurs négatives de la fonction objective. La valeur de la fonction objective devient donc la "performance" de l'individu.

Une fois la performance de chacun des individus évaluée, il faut ensuite classifier les individus et leur associer une probabilité de reproduction. La procédure utilisée ici est celle de Davis [4]. L'individu avec la meilleure performance reçoit un poids relatif (RW) de n^a (où a est généralement entre 1.0 et 1.5). Le deuxième meilleur individu reçoit un RW de $(n-1)^a$, le troisième meilleur individu reçoit un RW de $(n-2)^a$ jusqu'au dernier individu, le moins bon, qui reçoit un RW = $(n-(n-1))^a = 1$.

On calcule ensuite la probabilité de reproduction (PR) de chaque individu de la façon suivante :

$$PR_i = \frac{RW_i}{\frac{1}{n}\sum RW_i} \tag{1.2}$$

On voit encore que le meilleur individu aura une plus grande chance de se reproduire que le moins bon.

On utilise une procédure de sélection stochastique (*stochastic remainder selection procedure*) de Goldberg [2] pour déterminer la fréquence de sélection de chaque individu. On utilise la partie entière de la PR de chaque individu pour déterminer sa fréquence de sélection.

Exemple : Si PR = 2.1, l'individu est choisi deux fois pour reproduction. Si PR = 1.9, l'individu est choisi une fois pour reproduction.

Le reste des individus de l'ensemble de reproduction est choisi avec une liste triée des parties fractionnelles.

On a maintenant un ensemble d'individus qui ont survécu à la sélection et qui représentent les meilleures solutions obtenues.

1.4.3 Reproduction

Il faut ensuite choisir les paires d'individus qui vont se croiser pour former des rejetons. Chaque paire d'individu est choisie aléatoirement parmi l'ensemble de sélection. La probabilité qu'un individu soit choisi est proportionnelle à sa PR.

On prend une paire d'individus (P_1 et P_2) et on vérifie s'il y a croisement, selon la probabilité de croisement p_c (qui est typiquement entre 0.85 et 1). S'il y a croisement, on utilise la technique de Wright pour créer des rejetons, soit :

$$O_1 = 0.5 \cdot P_1 + 0.5 \cdot P_2 \tag{1.3}$$

$$O_2 = 1.5 \cdot P_1 - 0.5 \cdot P_2 \tag{1.4}$$

$$O_3 = -0.5 \cdot P_1 + 1.5 \cdot P_2 \tag{1.5}$$

Les deux meilleurs individus sont conservés afin de garder la taille de la population constante.

Le croisement est appliqué à chaque paramètre. Soit deux parents P_1 et P_2 constitués

de m variables indépendantes,

$$P_1 = \begin{pmatrix} x_1^1 \\ x_2^1 \\ \vdots \\ x_m^1 \end{pmatrix}, \qquad P_2 = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \end{pmatrix} \tag{1.6}$$

Puisque $O_1 = 0.5P_1 + 0.5P_2$,

$$O_1 = 0.5 \cdot \begin{pmatrix} x_1^1 \\ x_2^1 \\ \vdots \\ x_m^1 \end{pmatrix} + 0.5 \cdot \begin{pmatrix} x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \end{pmatrix} \tag{1.7}$$

Les rejetons sont :

$$O_1 = \begin{pmatrix} 0.5 \cdot x_1^1 + 0.5 \cdot x_2^1 \\ 0.5 \cdot x_1^2 + 0.5 \cdot x_2^2 \\ \vdots \\ 0.5 \cdot x_m^1 + 0.5 \cdot x_m^2 \end{pmatrix} \tag{1.8}$$

Le processus est le même pour créer les deux autres individus.

Une fois les rejetons créés, on obtient à nouveau une population de n individus qui, en théorie, ont une meilleure performance en moyenne que les individus de la génération précédente.

1.4.4 Mutation

La prochaine étape est la mutation. Dans le travail effectué ici, on utilise une mutation non-uniforme, de Michalewicz [5]. On vérifie d'abord s'il y a mutation, selon la probabilité de mutation p_m (qui se situe habituellement entre 0.00001 et 0.1).

Pour un paramètre x_i, s'il y a mutation, le changement s'effectue de la façon suivante, après le lancement d'une pièce de monnaie non-biaisée :

$$x_i' = \begin{cases} x_i + \delta[\max(x_i) - x_i] & \text{(pile)} \\ x_i - \delta[x_i - \min(x_i)] & \text{(face)} \end{cases} \tag{1.9}$$

où la fonction δ est définie selon :

$$\delta(y) = y \cdot r \cdot \left(1 - \frac{t}{T}\right)^b \tag{1.10}$$

où

- r = nombre aléatoire entre 0 et 1 (distribution uniforme)
- t = génération actuelle
- T = génération maximale
- b = degré d'extinction d'amplitude

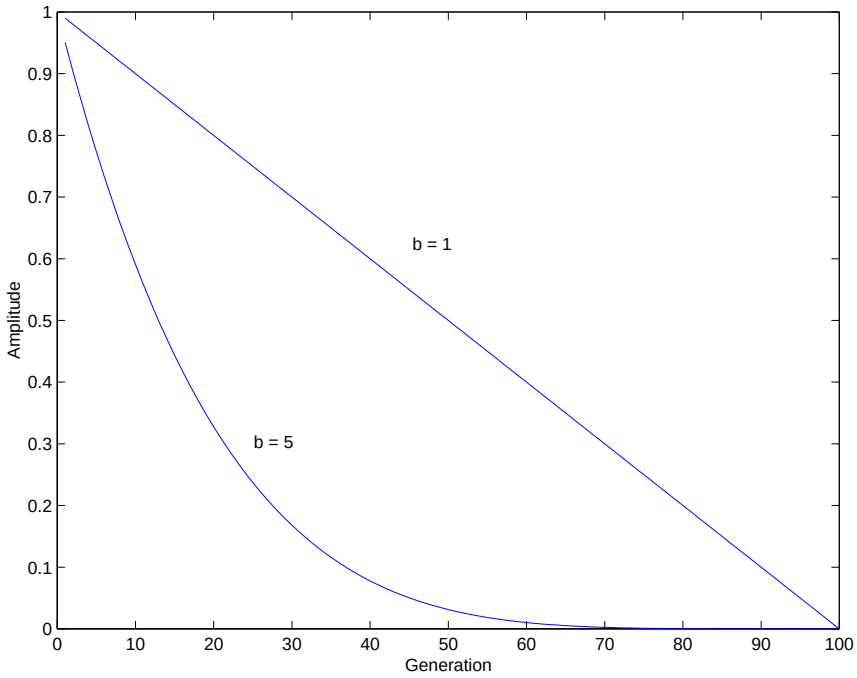

FIGURE 1.1 – Comportement générale de la fonction δ. T = 100.

La fonction est donc de la forme montrée à la figure 1.1, pour deux valeurs différentes de b. On voit bien que l'amplitude du changement du paramètre diminue au fur et à mesure que la génération augmente, d'où le nom de "non-uniforme".

La probabilité de mutation est vérifiée une fois par individu. S'il y a mutation, un seul paramètre est choisi (aléatoirement) pour subir une mutation.

Le processus de sélection, reproduction et mutation est alors répété. Chaque fois qu'on effectue ces trois opérations, cela constitue une génération.

On continue jusqu'à ce que le nombre maximal de générations soit atteint ou jusqu'à ce que la performance du meilleur individu soit plus petite qu'une erreur minimale spécifiée auparavant.

1.4.5 Élitism

Une technique très souvent utilisée avec l'algorithme génétique est l'élitisme. L'élitisme est la conservation du meilleur individu dans la génération suivante.

Après le croisement, on compare le meilleur individu de la nouvelle génération avec le meilleur individu de la génération précédente. Le meilleur individu est conservé et l'autre est détruit. Cette technique permet de s'assurer que la fonction objective augmente (diminue) toujours si on maximise (minimise) une fonction.

Bibliographie

[1] Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press : Ann Arbor, 1975.

[2] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley : Reading, MA, 1989.

[3] A. Wright, *Genetic Algorithms for Real Parameter Optimization*, pp. 205–218. Morgan Kaufmann : San Mateo, CA, 1991.

[4] L. Davis, *Handbook of Genetic Algorithms*. Van Nostrand Reinhold : New York, 1991.

[5] Z. Michalewicz, *Genetic Algorithms*. Springer-Verlag : New York, 1992.