République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Batna 2 Chahid Mostefa Benboulaïd
Faculté de Technologie
Département d'Electrotechnique

Dr Bachir ABDELHADI
Professeur

ALGORITHMES GENETIQUES

SUPPORT DE COURS

2020-2021

Sommaire

Bibliographie

[1] J. M. Renders, ''Algorithmes Génétiques et Réseaux de Neurones : *Application à la Commande de Processus*''. Editions Hermes, Paris, 1995

[2] D.E. Goldberg, ''Algorithmes génétiques : *Exploration, Optimisation et Apprentissage Automatique*''.

[3] S. Taïbi, ''Cours : AG. Magistère 2004-2005'', Département d'Electrotechnique, Université de Batna, 2005.

[4] B.Abdelhadi, ''Contribution à la Conception d'un Moteur à Induction Spécial à Rotor Externe pour Système de Propulsion Electrique : *Développement d'un Algorithme Génétique Adaptatif pour Identification Paramétrique*-'', Thèse de Doctorat en Sciences, Université de Batna, Algérie, Juillet 2004''.

[5] www.eudil.fr (chercher dans : ressources en ligne).

1. Introduction :

Trouver la solution optimale d'un problème dans un espace complexe implique un compromis entre deux objectifs : l'exploitation des meilleures solutions et l'exploration robuste de l'espace de recherche. Les méthodes de type grimpeur procèdent itérativement en tentant, à chaque pas, de trouver localement une solution intermédiaire meilleure que la solution courante ; ce genre de méthode est pénalisée par son incapacité à traiter des problèmes représentant des reliefs de solutions multimodales (système possédant plusieurs optimaux locaux).

Un algorithme génétique (GA) est une technique d'optimisation stochastique fondée sur les mécanismes de la sélection naturelle et de la génétique et elle est employée en informatique pour chercher les solutions approximatives à des problèmes d'optimisation. Les algorithmes génétiques sont une classe particulière des algorithmes évolutionnaires qui emploient des techniques inspirées de la biologie évolutionnaire telle que la mutation, la sélection, et la recombinaison (ou le croisement).

Les algorithmes génétiques (AGs) représentent une stratégie de recherche réalisant un compromis équilibré entre l'exploration de l'espace de recherche et l'exploitation des meilleures solutions. Des analyses théoriques ont montré que les algorithmes génétiques gèrent ce compromis de façon optimale.

Le but de ce cours est de présenter le principe de la méthode d'optimisation basé sur l'AG, son fondement ainsi que les différentes étapes nécessaires à son développement, pour en finir avec la recherche de la solution optimale de certaines applications simples qui faciliteront la compréhension de ces algorithmes.

2. Définitions

L'optimisation par algorithme génétique prend son origine dans les mécanismes de la sélection naturelle et la génétique de l'évolution. Cette méthode a été mise en œuvre par J.H. Holland dans les années 70. Comme son nom l'indique, elle est basée sur la traduction mathématique des phénomènes naturels qui sont la reproduction des espèces, la survie et l'adaptation des individus. Cette traduction est exploitée pour la résolution de problèmes nécessitant l'optimisation d'une fonction ou d'un système dépendant de plusieurs paramètres et qui ont besoin d'être calculés pour un critère bien défini (maximisation, minimisation, …).

Cette technique constitue une méthode d'optimisation robuste. L'AG peut résoudre, avec fiabilité, des fonctions représentant des reliefs de solution réputés très difficiles pour les méthodes d'optimisation classiques (Simplex, le plus fort gradient …). Les fonctions réputées difficiles sont des fonctions qui présentent plusieurs optima locaux, des fonctions discontinues ou des fonctions à plusieurs dimensions où les méthodes ordinaires ne peuvent pas prendre en compte l'effet d'interaction entre tous les paramètres.

3. Analogie avec le fonctionnement biologie

Un algorithme génétique (GA) est un processus informatique mimant l'évolution biologique naturelle d'une population d'êtres vivants où chaque individu représente une solution potentielle à un problème donné. Comme l'illustre la figure ci-dessous, ce processus crée une nouvelle population d'individus par une opération de reproduction où chaque individu a hérité d'une partie du patrimoine génétique de ses parents.

Un algorithme génétique recherche les extrema d'une fonction définie sur un espace de données appelé population initiale. Par analogie avec la génétique, chaque individu de cette population est un chromosome et chaque caractéristique de l'individu est un gène.

Dans le cas du codage binaire, un gène sera représenté par un bit (0 ou 1), un chromosome par une chaîne de bits et un individu par un ensemble de chaîne de bits.

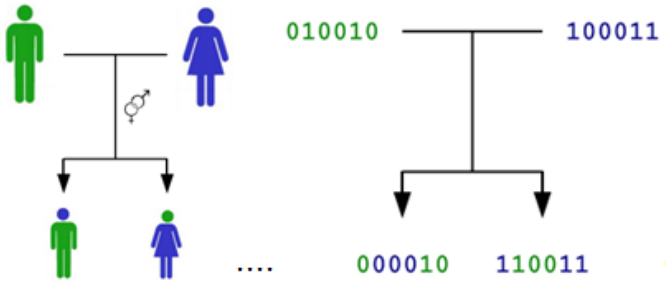

Figure 1: Analogie entre évolution biologique naturelle et algorithme génétique. La reproduction de deux individus permet de créer deux nouveaux individus où chacun a hérité d'une partie du patrimoine génétique des parents

4. Caractéristiques de l'AG

Les principales caractéristiques relatives à cette technique se concentrent autour des trois points :

1) Le parallélisme : l'algorithme génétique travaille en parallèle sur un certain nombre de candidats et non pas sur un candidat unique. La méthode de recherche est globale et couvre tout l'espace de recherche.
2) L'utilisation minimale d'informations : il n'a besoin que de la mesure d'adéquation (la qualité d'une solution), il ne repose sur aucune autre information, par exemple des dérivés ou hypothèses telles que la continuité et la différentiabilité. Il ne requiert qu'une capacité à classer les solutions entre elles.
3) L'utilisation de règles probabilistes plutôt que déterministes dans l'exploration de l'espace de recherche. L'introduction du hasard est très bénéfique pour l'optimisation de fonctions présentant plusieurs optima et aussi en cas de fonction non permanente (déplacement ou changement des optima au cours du temps).

5. Principe de fonctionnement

L'AG est une des méthodes de recherche itératives. Il permet d'optimiser une fonction définie par l'utilisateur appelée *fonction objective* ou *fonction d'adéquation.*

Pour atteindre cet objectif, l'algorithme travaille en parallèle sur une population de points candidats appelés *individus* ou *chromosomes* (solution particulière) qu'on note **I**, chaque individu est constitué d'un ensemble d'éléments appelés *gènes* notés $\mathbf{x_{nN}}$ figure (1.a).

5.1 Codage

Dans l'algorithme génétique de base, tel qu'il a été fondé par Holland, les gènes sont formés de **1** et **0**. Dans ce cas, chaque valeur réelle $\mathbf{x_n}$ (paramètres à optimiser) est codée par son équivalent en **binaire** et l'individu obtenu est représenté par une chaîne codée de plusieurs gènes représentant une solution particulière pour la fonction objective figure (1.b).

De nouvelles versions d'algorithme génétique sont apparues. Elles ne se basent plus sur le codage binaire mais elles travaillent directement sur les paramètres réels.

Ces versions, appelées algorithmes génétiques **codés réels**, offrent d'une part l'avantage d'**accélérer** la recherche et d'autre part de rendre plus **facile** le **couplage** avec d'autres méthodes d'optimisation. Ce codage est de plus en plus répandu.

A chaque itération, appelée *génération,* est généré une nouvelle population avec, toujours, le même nombre d'individus au total (n=*N*). Cette population est mieux adaptée à l'environnement tel qu'il est représenté par la fonction objective et les critères de l'optimisation (maximisation, minimisation, …). Plus on progresse dans les générations, plus les individus vont devoir tendre vers l'optimum de la fonction objective.

Le passage d'une génération à l'autre s'effectue en trois étapes, **évaluation** puis **sélection** et enfin **reproduction** avec des opérateurs de *croisement* puis de *mutation*.

Ces différentes étapes vont permettre, à la fin de la procédure, de trouver la combinaison optimale de gènes constituent l'individu le mieux adapté.

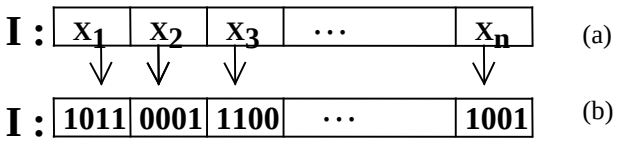

Figure 2 : *Représentation d'un individu*; *codage réel* (1.a), *codage binaire* (1.b)

5.2 Procédé d'implantation des AGs

Pour utiliser l'AG, on doit disposer de six éléments :

- Un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace considéré une structure de données. Elle se place après la phase de modélisation mathématique du problème traité. Les codages binaires ont été les premiers à être utilisés. Actuellement, on se sert de plus en plus de codages réels notamment pour l'optimisation des problèmes à variables réelles.
- Un mécanisme capable de générer une population initiale non homogène qui servira de base pour les générations futures. Ce choix conditionne la rapidité de la convergence vers l'optimum. 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.

- Une fonction à optimiser (fonction d'évaluation ou fitness).
- Un mécanisme de sélection des individus candidats à l'évolution. On utilise généralement la ''roulette'' du casino pour sélectionner les individus au hasard. Chaque individu occupe sur la roulette un secteur proportionnel à sa fonction d'évaluation : cela fait que le hasard est biaisé envers les éléments les plus justes (aptes) qui ont plus de chances d'être sélectionnés que les autres.
- Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. L'opérateur de croisement recompose les gènes d'individus existants. L'opérateur de mutation a pour but de garantir l'exploration de l'espace.
- Des paramètres de dimensionnement: taille de la population, critère d'arrêt, probabilité d'application des opérateurs génétiques.

5.3 Paramètres de l'AG

En réalité, il n'existe pas de paramétrage universel pour la quantification de ce paramètres de dimensionnement. Cependant, certaines valeurs largement utilisées pour résoudre concrètement des problèmes méritent d'être retenues :

- Taille de la population: entre **N** = 30 et 50 individus
- Taux de croisement: entre **Pc** = **pc** = 70% et 95%
- Taux de mutation: **Pm** = **pm** =0,5% à1%.

5.4 Ossature de l'algorithme

```
        p désigne la population
        t désigne la génération
Début
        t =0
        initialise p(t)
        évaluer p(t)
                tantque(condition de terminaison est fausse)
                début
                        t = t + 1
                        sélectionner p(t) de p(t-1)
                        recombiner p(t)
                        évaluer p(t)
                fin
Fin.
```

5.5 Cycle de l'algorithme génétique

Le cycle de l'AG commence à générer aléatoirement une population initiale (comme solutions possibles). Il opère, à une évaluation des individus de cette population selon leur fonction objective et ensuite à un test d'arrêt. Si ce dernier n'est pas satisfait, le processus de reproduction, qui regroupe principalement la sélection, le croisement et la mutation, pour créer une nouvelle population qui sera soumise à l'évaluation et l'opération se répète. Lors du processus de reproduction, les meilleurs individus choisis par la fonction de sélection sont soumis au croisement. Ce dernier (hybridation) ou crossover, en anglais, consiste en l'échange d'un certain nombre de bits (gènes) entre les deux parents. Les meilleurs enfants obtenus seront croisés, à leur tour, pour obtenir encore une meilleure génération. Les individus issus de ce croisement sont soumis à l'opérateur de mutation. Cet opérateur permet de changer la valeur de quelques bits aléatoirement des individus afin de bien imiter le processus naturel.
Ensuite, le processus se répète à partir de la dernière génération jusqu'à ce que soit atteint le critère d'arrêt défini au départ.

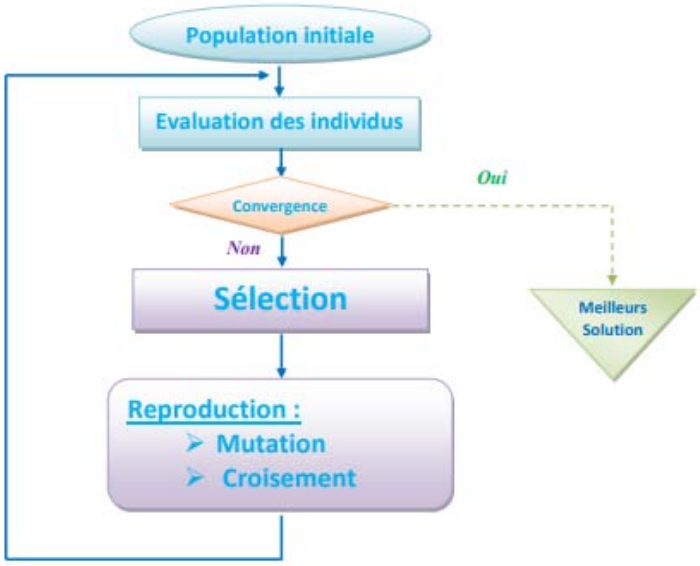

Figure 3 : Cycle de l'algorithme génétique

6. Evaluation

L'algorithme génétique évalue une population d'individus qui forme la génération courante qu'on appelle (G_t). Les individus les plus forts, au sens des critères de la fonction objective, auront théoriquement plus de descendants, que les autres individus, dans la génération qui suit, donc (G_{t+1}). Si l'on transpose dans un sens mathématique, l'algorithme génétique évalue la fonction d'adéquation que nous notons (F) pour chaque individu (I) de la population courante $P(G_t)$. Suivant les critères de l'optimisation, un classement entre individus sera effectué et les meilleurs, qui répondent le mieux aux critères de la fonction objective auront une probabilité de reproduction (clonage) plus importante que les autres. C'est cette information qui guidera l'algorithme génétique vers les meilleurs individus.

7. Sélection

Suivant les probabilités de reproduction (p_s) données aux individus, au moment de l'évaluation, on fera la sélection stochastique des individus pour construire une population intermédiaire, notée $P'(G_t)$, toujours constituée de *n* individus.

Elle permet d'identifier les meilleurs individus d'une population, qui répondent au mieux à la fonction d'adéquation, et d'éliminer les mauvais. On trouve dans la littérature un nombre important de principe de sélection plus ou moins adaptés aux problèmes qu'ils traitent. A titre illustratif, on ne s'intéressera qu'à la **sélection à l'aide de la roue de loterie biaisé**.

Pour sélectionner à l'aide de la roue de loterie biaisé, on fait tourner la roulette N fois (taille de la population) de la façon suivante : à chaque fois, on génère aléatoirement un nombre r dans l'intervalle [0, 1]. Ensuite, on compare ces nombres aux probabilités q_j. Si $r_1 < q_1$, v_1 est sélectionné, sinon v_j est sélectionné avec $2 \leq j \leq N$ tel que $q_{j-1} < r_1 < q_j$. On procède de la même manière pour le reste des r_i (i = 2, N).

La probabilité cumulative, q, pour chaque chromosome est donnée par :

$$q_j = p_1 + p_2 + \ldots + p_j$$

La probabilité de sélection, p_s, de chaque chromosome est exprimée par :

$$p_{sj} = \text{feval}(x_j)/F \text{ et } F = \sum_{j=1}^{N} \text{feval}(x_j)$$

8. Reproduction avec les opérateurs de croisement et de mutation

Une fois l'étape de sélection achevée, l'algorithme génétique poursuit sa recherche par l'application des opérateurs de croisement et de mutation. L'opérateur de croisement joue le rôle de recombinaison et d'échange entre certains individus. Quant à l'opérateur de mutation, il modifie « localement » un individu en changeant sa composition.

8.1. Croisement

Un pourcentage de la population intermédiaire sélectionnée $P'(G_t)$ sera soumis au croisement. Ainsi, l'opérateur de *croisement* choisit au hasard, et avec une probabilité fixée (p_c), deux individus (deux parents) parmi cette population intermédiaire. Il construit alors deux enfants en faisant l'échange de certains gènes choisis aléatoirement d'un parent avec ceux de l'autre. Les deux enfants issus de ce croisement sont injectés dans la population que l'on note $P''(G_t)$. Cette dernière sera alors constituée d'un pourcentage d'enfants issus du croisement. Le reste est issu directement de la population $P'(G_t)$ sans aucune modification. Le nombre total d'individus dans $P''(G_t)$ est toujours égal à *n* individus.

Il existe plusieurs type d'opérateurs de croisement :

8.1.1. Croisement un point ou discret

Pour ce type de croisement, on choisira au hasard "un site de coupe" entre les deux parents, figure 2. Cela nous amène à prendre un nombre *k* (indice de site) entre 1 et (*L*-1) où *L* représente le nombre de gènes d'un individu. On obtient alors deux enfants en prenant les *k* premiers gènes du premier parent et les (*L*-*k*) derniers gènes du second parent. L'autre enfant est obtenu avec les *k* premiers gènes du second parent et les (*L*-*k*) derniers gènes du premier parent. Ces deux enfants sont, comme indiqué ci dessus, injectés dans la population $P''(G_t)$.

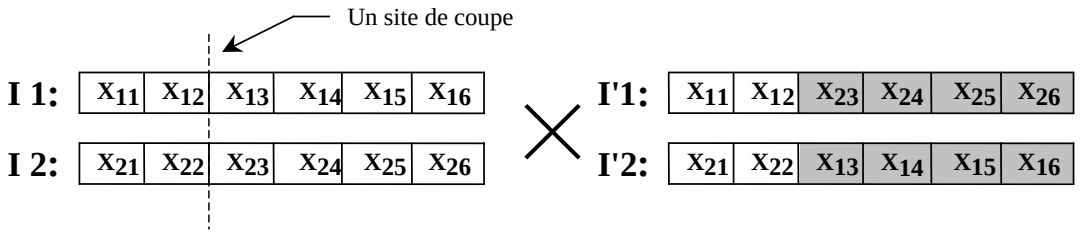

Figure 4: *Croisement un point (exemple d'un individu à six gènes)*

8.1.2. Croisement à deux points

Le croisement à deux points est basé sur le même principe que le croisement un point. La différence réside dans le fait que deux sites de coupe sont introduits au hasard entre les deux parents. Comme le montre la figure 3, les gènes se trouvant entre les deux sites de coupe sont échangés respectivement entre les deux individus (parents) pour former les deux enfants qui rejoindront la population $P''(G_t)$.

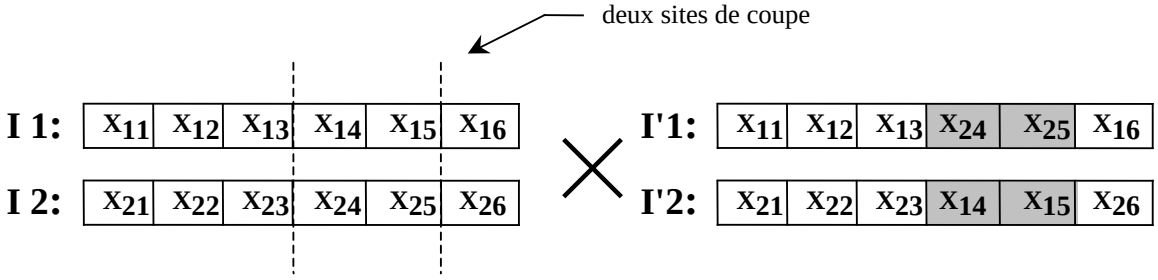

Figure 5: *Croisement deux points*

8.1.3. Croisement continu (ou uniforme)

Ce croisement effectue une opération de type moyenne sur certains gènes des parents. Chaque gène a une chance sur deux d'être moyenné avec son homologue chez l'autre parent -voir figure 4.

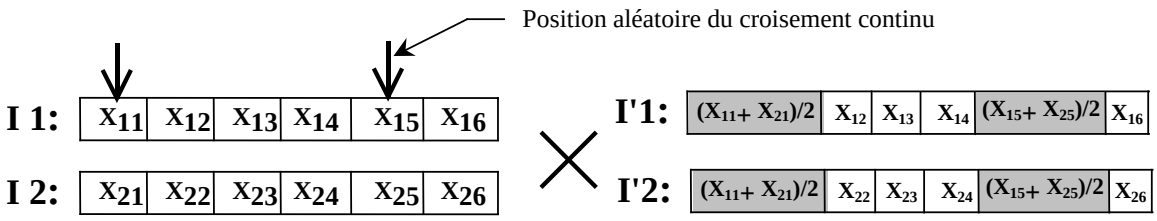

Figure 6: *Croisement continu (uniforme)*

8.1.4. Croisement arithmétique

Ce type de croisement est proche du croisement continu. Comme précédemment, on choisit aléatoirement deux positions d'échange puis on effectue une moyenne arithmétique pondérée par un coefficient *a*. soit, pour la figure 5 les deux nouveaux gènes :

$Y_{13} = a\,X_{13} + (1-a)\,X_{23}$ **et** $Y_{23} = (1-a)\,X_{13} + a\,X_{23}$

qui remplacent X_{13} et X_{23}. Ils forment les enfants qui intègrent la population intermédiaire $P''(G_t)$. La valeur de *a* est générée aléatoirement dans l'intervalle $[0\,,1]$.

On note que si $a = 0.5$, on retrouve le croisement continu.

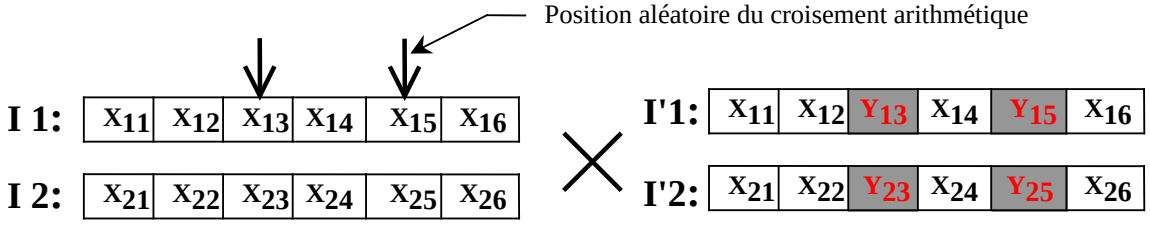

Figure 7: Croisement arithmétique

On peut aussi, appliquer une autre loi de croisement. Les gènes qui se trouvent par exemple dans les sites de croisement X_{11} et X_{21}, sont remplacés respectivement par :

$$X_{11} + a(X_{21} - X_{11}) \qquad \text{et} \qquad X_{21} + a(X_{11} - X_{21})$$

Une fois l'étape de croisement est achevée, on applique la procédure de mutation.

8.2. Mutation

Chaque gène des individus de la population $P''(G_t)$ peut subir une *mutation* avec une probabilité fixée notée p_m. L'opérateur de mutation agit donc en modifiant aléatoirement un ou plusieurs gènes d'un individu. La valeur du gène muté est remplacée par une autre appartenant au même domaine de variation.

On considère les individus choisis aléatoirement, parmi les individus de la population $P''(G_t)$, qui sont soumis à l'opérateur de mutation. Pour chacun de ces individus, on choisit également aléatoirement des gènes qui vont subir une modification.

Il est à noter que les gènes concernés par la mutation vont subir une modification importante durant les premières générations. Par la suite, le niveau des modifications ira en diminuant au fur et à mesure que la recherche évoluera vers la solution.

Il existe plusieurs types d'opérateurs de mutation qui permettent d'assurer cette modification, nous citons ici

8.2.1. Mutation uniforme

Pour chaque gène qui mute, on prend deux nombre *s* et *r*. Le premier « s » peut prendre les valeur ± 1. Dans le cas où *s* est égal à 1, le changement est positif, dans l'autre cas le changement est négatif. Le second « r » détermine l'amplitude du changement. C'est un nombre généré aléatoirement dans l'intervalle [0 1]. Dans ces conditions, le gène X'_j qui remplace le gène qui mute X_j est calculé à partir de l'une des deux relations suivantes :

$$X'_j = X_j + \left(X_{\max} - X_j\right)\left(1 - r^{(1-\frac{G_t}{G_F})^5}\right) \quad \text{si} \quad s = 1$$

$$X'_j = X_j - \left(X_j - X_{\min}\right)\left(1 - r^{(1-\frac{G_t}{G_F})^5}\right) \quad \text{si} \quad s = -1$$

Où $X_{\min}$ et $X_{\max}$ désignent respectivement les limites inférieure et supérieure de la valeur du paramètre X_j et $G_F \le G_T$ représente la génération pour laquelle l'amplitude de la mutation s'annule. A partir de cette génération les individus ne subissent plus de mutation, et l'exploration du reste de l'espace de recherche est assurée uniquement par l'action de croisement.

Exemple 1 : X_{min}=1; X_{max}=10; S=1; G_F=10 et X_j=4;

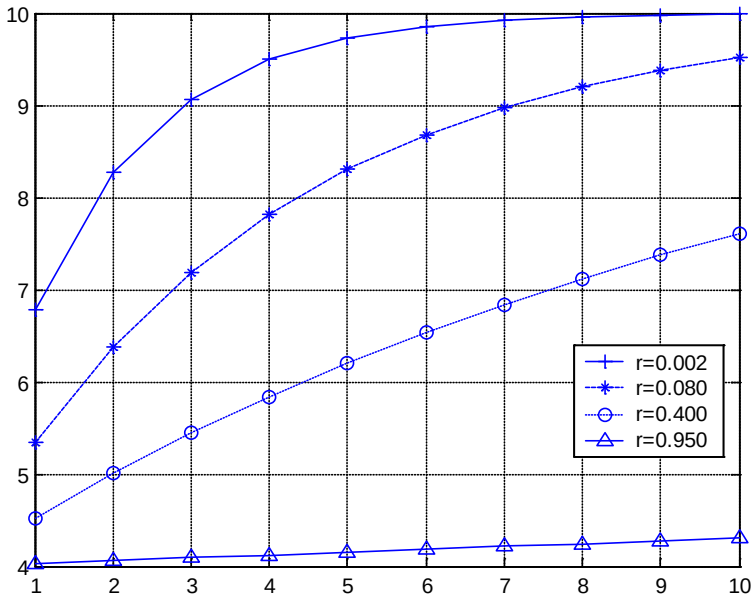

Figure 8 : Evolution du gène muté X'_j en fonction du gène X_j

8.2.2. Mutation non uniforme

On remplace directement la valeur du gène qui mute, par exemple x_j, par une autre valeur prise aléatoirement dans l'intervalle $[X_j^{\min} \quad X_j^{\max}]$.

Il existe un autre type de mutation non uniforme. Dans ce cas, la valeur mutée X'_j du gène X_j est donnée par la formule :

$$X'_j = \begin{cases} X_j + \Delta(t,y) \\ ou \\ X_j - \Delta(t,y) \end{cases} \qquad \text{avec} \qquad \Delta(t,y) = y\, r \left(1 - \frac{G_t}{G_F}\right)^b$$

Dans cette expression, y peut prendre les valeurs $(X_j^{\max} - X_j)$ ou $(X_j - X_j^{\min})$ à condition que le résultat de l'opération ne sorte pas de l'intervalle $[X_j^{\min} \quad X_j^{\max}]$. Le paramètre *r* représente un nombre aléatoire avec $0 < r < 1$. b est un paramètre qui définit le degré de non uniformité. La fonction $\Delta(t,y)$ renvoie, d'une façon aléatoire, un nombre dans l'intervalle $[0 \quad y]$. Elle permet de réaliser une recherche uniforme dans les premières générations, et plus pointue au fur et à mesure que l'on avance.

D'autres lois peuvent être utilisées pour calculer l'évolution des gènes en mutation. On peut remplacer le gène qui mute X_j par X'_j telle que X'_j est calculé par :

$$X'_j = X_j \pm \Delta X_j$$

Avec : $\Delta X_j = (X_{max} - X_{min}).2^{-kr}$

La quantité k est une constante souvent prise égale à 16.

Après la mutation, les individus constitueront la nouvelle population $P(G_{t+1})$ de *n* individus qui donne naissance à la nouvelle génération. Le cycle continue : évaluation, sélection, reproduction (croisement et mutation), évaluation, etc. jusqu'à la dernière génération fixée.

Il y a donc quatre paramètres de base qui doivent être fixés pour assurer le fonctionnement d'un AG : le nombre d'individus dans la population *n*, la génération maximale G_T, les taux de croisement p_c et de mutation p_m.

Trouver de bonnes valeurs à ces paramètres est un problème parfois délicat. La valeur de *n* qui désigne le nombre d'individus de la population, dépend fortement du problème à optimiser (en particulier le nombre de gènes de chaque individu). En pratique, on arrive à une solution acceptable en prenant pour les valeurs de p_c et p_m respectivement 0.7 et 0.01.

9. Organigramme de la procédure AG

Nous résumerons, par l'organigramme de la figure ci-dessous, l'ensemble des étapes permettant d'effectuer une optimisation avec l'algorithme génétique.

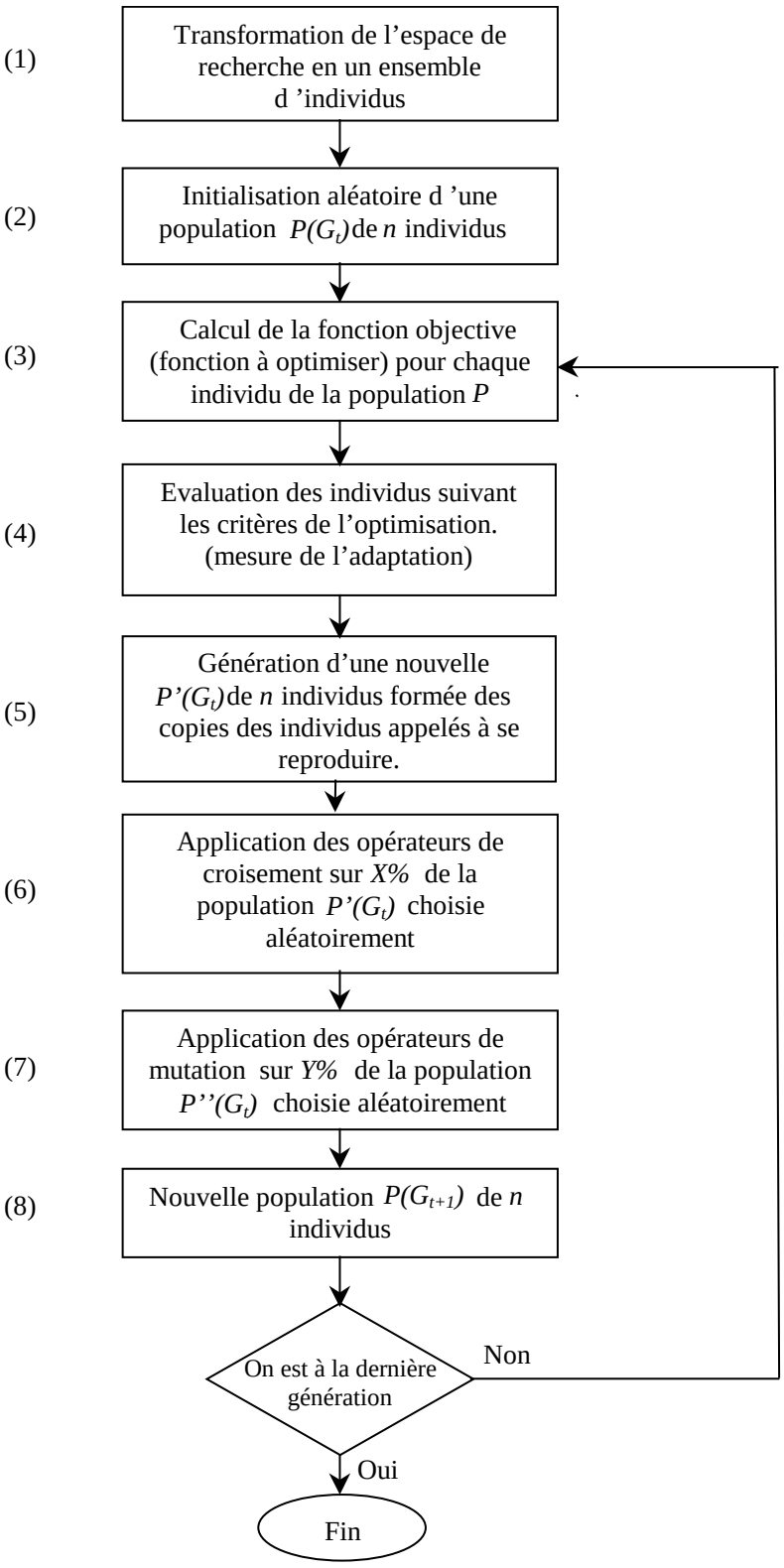

Figure 9 : Les différentes étapes de l'algorithme génétique

10. Application d'optimisation par AGs

Dans le but de tester l'efficacité de l'algorithme génétique, développé sur la base des outils décrits précédemment, nous proposerons une procédure de maximisation par AGs et considérerons quelques exemples d'application.

10.1 Procédure de maximisation par algorithme génétique

Pour chercher le maximum d'une fonction simple f(x) dans l'intervalle [a, b] avec une précision de n chiffres significatifs, on procédera de la manière suivantes:

- L'intervalle [a, b] est subdivisé en (b - a) 10^n petits intervalles qui représenteront chacun un chromosome.
- Chaque chromosome est codé en binaire à l'aide de k bits, avec k vérifiant les inéquations suivantes :

  $$2^{(k-1)} < (b - a)\ 10^n \leq 2^k$$

- La valeur décimale, x', correspondant au code binaire de chaque chromosome binaire $v_j=(a_{k-1} \ldots a_1\ a_0)_j$, est calculé par :

  $$x'_j = \sum_{i=0}^{i=k-1} a_i\ 2^i$$

- Le nombre réel, x, correspondant à la valeur binaire est déterminé par :

  $$x_j = a + x'_j\ ((b-a)/(2^k - 1))$$

- Pour chaque génération les calculs suivants sont effectués :
  - Calcul de la fonction d'évaluation *feval(x_j)* pour chaque chromosome v_j,
  - Calcul de l'évaluation totale, F, de la population constitué de N individus :

    $$F = \sum_{j=1}^{N} \text{feval}(x_j)$$

  - Calcul de la probabilité de sélection, p_s, de chaque chromosome :

    $$p_{sj} = \text{feval}(x_j)/F$$

  - Calcul de la probabilité cumulative, q, pour chaque chromosome :

    $$q_j = p_1 + p_2 + \ldots + p_j$$

  - Pour sélectionner à l'aide de la roue de loterie biaisé, on fait tourner la roulette N fois (taille de la population) de la façon suivante : à chaque fois, on génère aléatoirement un nombre r dans l'intervalle [0, 1]. Ensuite, on compare ces nombres aux probabilités q_j. Si $r_1 < q_1$, v_1 est sélectionné, sinon v_j est sélectionné avec $2 \leq j \leq N$ tel que $q_{j-1} < r_1 < q_j$. On procède de la même manière pour le reste des r_i (i = 2, N).
  - Pour chaque chromosome de la nouvelle génération, on génère, au hasard, N nombres r dans [0, 1] et on les compare à la probabilité de croisement P_c. Si $r_i<P_c$, le $i^{ème}$ chromosome est sélectionné pour le croisement, sinon il n'est pas.
  - Croisement des chromosomes ainsi sélectionnés deux à deux. Si le nombre de ces chromosomes est impair, on peut élaguer un, ou bien reprendre un autre.
  - On mute un bit de l'ensemble des gènes des différents chromosomes si le nombre généré arbitrairement r ≤ probabilité de mutation P_m.
- Dans notre cas, après chaque génération, le nombre d'individus est incrémenté progressivement afin d'introduire une assez grande diversification dans la population.

10.2 Exemples d'application

Les exemples suivants sont choisis très simples pour permettre à un débutant d'y mettre pied. Evidemment dans le cas des fonctions mathématiques usuelles, les méthodes analytiques sont, de loin, plus élégantes et plus précises. Mais, aussitôt que l'on s'approche d'une fonction complexe dont on ne connaît rien sur sa dérivée ou celle-ci est difficile à résoudre, l'approche génétique deviendra incontournable.

10.2.1 Exemple didactique : Recherche du maximum d'une fonction réelle à une variable

Cet exemple, concerne une fonction mathématique à une variable dont on cherche le maximum. Il est traité d'une manière pédagogique afin ce de faciliter la compréhension de l'implémentation de l'approche génétique.

Cherchons le maximum de f(X) = -(X^2) + 4X dans l'intervalle [1, 3] avec une précision de 1/10. Analytiquement, on voit rapidement que f'(X) = -2X + 4, que f''(X) = -2 < 0 et que le maximum correspond

à **X = 2** et **f(X) = 4**. Cherchons la longueur du chromosome (nombre de bits de la chaîne). La longueur de l'intervalle est 3 – 1 = 2.
Chaque unité doit être subdivisée en 10^n (précision souhaitée, n=1). Donc, l'intervalle est subdivisé en 2 * 10 = 20 petits intervalles. Le nombre de bits requis pour représenter tous les réels considérés dans l'intervalle est k tel que $2^{(k-1)} =< 20 =< 2^k$. k = 5.
Pour modéliser le problème, convenons de ce qui suit, on considère une population de 4 individus (chromosomes), chaque individu codé sur 5 bits (gènes). $P_c = 0,75$ et $P_m = 0,01$.
Construisons aléatoirement la génération initiale.

V1	01100	X'1=12	X1=1.77=a+X'1*(b-a)/(2^k -1)= 1+12*(2)/(31)	eval(V1)= f(x1) =3.949
V2	00011	X'1=3	X2=1.19	eval(V2)= f(x2)= 3.349
V3	11011	X'1=27	X3=2.74	eval(V3)= f(x3)= 3.449
V4	10100	X'1=20	X4=2.29	eval(V4)= f(x4)= 3.915

La somme des évaluations est 14.664; la plus grande évaluation 3,949 et la valeur moyenne 3,683.
Formons la première génération.

Sélection:

En calculant les probabilités P et Q, on obtient:
P1 = 0.2693 Q1 = 0.2693
P2 = 0.2284 Q2 = 0.4977
P3 = 0.2352 Q3 = 0.7330
P4 = 0.2670 Q4 = 1.0000
On fait tourner 4 fois la roulette pour générer des nombres r dans [0 , 1], on obtient:
0.512 0.710 0.216 0.773
L'indice de l'individu qui sera sélectionné est déterminé par l'indice de la première probabilité cumulative qui est strictement supérieure à r correspondant.

$r_{s1} = 0,512$	q_3=0.7330> r_{s1} =0,512	V_3	est sélectionné
$r_{s2} = 0,710$	q_3=0.7330> r_{s2} = 0,710	V_3	est sélectionné
$r_{s3} = 0,216$	q_1=0.2693> r_{s3} = 0,216	V_1	est sélectionné
$r_{s4} = 0,773$	q_4=1.0000> r_{s4} = 0,773	V_4	est sélectionné

La première génération devient:

V1' =V3	11011
V2' = V3	11011
V3' = V1	01100
V4' = V4	10100

Croisement:

Assumons qu'aléatoirement, on procède au croisement à partir de la deuxième position. On fait tourner la roulette pour générer des nombres r dans [0, 1]. Si r<P_c=0,75, le chromosome est sélectionné pour le croisement. On obtient: 0,82 - 0,52 - 0,17 - 0,35. Alors V'2, V'3 et V'4 sont sélectionnés. Comme le nombre est impair, on laisse tomber le dernier. Cela donne pour le croisement:

V2'	11 |	011	11100	V2''
V3'	01 |	100	01011	V3''

Après croisement on obtient:

V1'' =V'1	11011
V2'' =	11100
V3'' =	01011
V4'' = V'4	10100

Mutation:

Il y a 4x5 = 20 bits. On tourne la roulette 20 fois pour générer r dans [0, 1]. Si r<P_m=0,01, on mute le bit de ce rang.
r_m = [0.121, 0.92, 0.27, 0.85, 0.02, 0.33, 0.71, 0.42, 0.61, 0.57, 0.107, 0.215, 0.03, 0.42, 0.87, 0.09, 0.82, 0.008, 0.87, 0.15]

Seulement, au 18ème tour, on obtient r=0.008, on mute, alors le 18ème bit qui correspond au 3ème bit du 4ème vecteur. Finalement, la première génération devient:
Finalement, la première génération devient :

V1 11011
V2 11100
V3 01011
V4 10000

En évaluant la première génération, on obtient:

V_1 11011 x'_1=27 x_1=2.74=a+x'_1(b-a)/(2^k-1)=1+27*2/31, eval(V_1)=f(x_1) =3.4495
V_2 11100 x'_2= 28 x_2=2.80 eval(V_2)= f(x_2)= 3.3496
V_3 01011 x'_3=11 x_3=1.71 eval(V_3)= f(x_3)= 3.9157
V_4 10000 x'_4=16 x_4=2.03 eval(V_4)= f(x_4)= 3.9990

Évaluation totale **F_2=14.7138**, la plus grande valeur **F_{max2}=3.9990** valeur moyenne **F_{av1}=3.6826.**
On vient de terminer une itération de la boucle ‘’Tant que’, et la solution délivrée par cette génération est **X4= 2.03** qui correspond à **f(x4)= 3.9990** et elle est très proche du résultat théorique (**x=2** et **F_{max}=4**)..

Si ce résultat n’est pas satisfaisant, on forme une autre génération à partir de celle-ci et on refait la boucle ‘’tant que’’ (on applique les opérateurs de sélection, de croisement et de mutation). Ensuite, on forme autant de générations qu’il faut jusqu’à la satisfaction du critère d’arrêt.
A la fin, on détermine la solution globale qui est le maximum de tous les tous les maximum de chaque itération.

10.2.2 Recherche d’optimum des fonctions multi-variables

1) Une variable: X_0=2, f_{max}=4.
F(x)=-x.^2+4*x; L’espace de recherche : a=[0]; b=[3];

2) 2 variables : $X(1)_0$=1, $X(2)_0$=20, f_{max}=2
f(x(1),x(2))=sin(pi*x(1)/2) + sin(pi/20 * x(2)/2); L’espace de recherche : a=[0 10]; b=[2 30];

3) 3 variables : $X(1)_0$=1, $X(2)_0$=20, $X(3)_0$=10, f_{max}=3
f(x(1),x(2), x(3))=sin(pi.*x(1)./2)+sin(pi/20 * x(2)./2)+sin(pi/10 * x(3)./2);
L’espace de recherche : a=[0 10 5]; b=[2 30 15];

4) 6 variables: $X(1)_0$=10, $X(2)_0$=20, $X(3)_0$=30, $X(4)_0$=40, $X(5)_0$=50, $X(6)_0$=60, f_{max}=6.
f(x(1),x(2), …,x(6))=sin(pi/10*x(1)/2)+sin(pi/20*x(2)/2)+ sin(pi/30*x(3)/2)
+ sin(pi/40*x(4)/2)+ sin(pi/50*x(5)/2)+ sin(pi1/60*x(6)/2);
L’espace de recherche : a=[5 15 25 35 45 55]; b=[15 25 35 45 55 65];

5) 8 variables : $X(1)_0$=10, $X(2)_0$=20, $X(3)_0$=30, $X(4)_0$=40,
$X(5)_0$=50, $X(6)_0$=60, $X(7)_0$=70, $X(8)_0$=80, f_{max}=8.
f=sin(pi/10*x(1)/2)+sin(pi/20*x(2)/2) +sin(pi/30*x(3)/2)+sin(pi/40*x(4)/2)
+sin(pi/50*x(5)/2)+sin(pi/60*x(6)/2) +sin(pi/70*x(7)/2)+sin(pi/80*x(8)/2)';
L’espace de recherche : a=[5 15 25 35 45 55 65 75]; b=[15 25 35 45 55 65 75 85];

10.2.3 Interpolation des fonctions par identification paramétrique

1) Première fonction
Vecteurs de données : XDATA= 0 : 0 .1 :4; YDATA=10.*sin(XDATA.*pi/2);

La fonction f=X.*sin(XEDATA*Y) ; La solution: X_0=10, Y_0=(pi/2)=1.57 ;
L'espace de recherche : ax=1; bx=20; ay=0; by=pi = 3.1416.

2) Deuxième fonction

Vecteurs de données : XDATA=1 : 0.2 : 10;
YDATA=200+10*XDATA - 400./(XDATA.^(1/2)+1);
La fonction f=200+X.*XDATA - Y./(XDATA.^(1/2)+1) ; La solution: X_0=10, Y_0=400
L'espace de recherche ax=5; bx=15; ay=300; by=600;

11- Conclusion

Ce cours constitue une introduction aux algorithmes génétiques et fournit les éléments nécessaires à leur programmation. Les exemples sont choisis très simples pour permettre à un débutant d'y mettre pied. Évidemment dans le cas des fonctions mathématiques usuelles, les méthodes analytiques sont, de loin, plus élégantes et plus précises. Mais, aussitôt que l'on s'approche d'une fonction H( ) complexe dont on ne connaît rien sur sa dérivée ou dont l'équation H'( ) = 0 est difficile à résoudre, l'approche génétique deviendra incontournable.