

" Tout ce qui existe dans l'univers est le fruit du hasard et de la nécessité. "
Démocrite.

Les Algorithmes Génétiques sont basés sur la théorie de l'évolution de Darwin. Ils consistent à faire évoluer une population de dispositifs à l'aide de différents opérateurs : sélection, croisements, mutations. Ils sont en particulier utilisés pour les problèmes d'optimisation comportant de multiples paramètres et des objectifs multiples.

- [Avant-propos](#)
- [Généralités sur la modélisation](#)
- [Optimisation](#)
 - [Méthodes Monte Carlo](#) (le hasard)
 - [Algorithmes Génétiques](#) (le hasard et la nécessité)
 - [Méthodes utilisées](#)
 - [Convergence](#)
 - [Un exemple concret d'optimisation](#)
 - [Exemple du "voyageur de commerce" \(applet Java\)](#)
- [Conclusion](#)
- [Perspectives](#)
- [Bibliographie](#)
- [Liens](#)



[Retour vers les 10 autres cyber-cours gratuits de l'EUDIL](#)



[Historique du cours](#) : V1.7 - 14 juin 2001.

Auteur : [Vincent MAGNIN](#)

UMR CNRS 8520



IEMN, Equipe Optoélectronique, BP 69, 59 652
Villeneuve d'Ascq Cedex

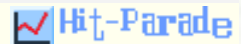


03 20 19 79 67



03 20 19 79 66

Retour vers mon [site professionnel](#).



EUDIL, département Science des Matériaux,
USTL, 59655 Villeneuve d'Ascq Cedex



03 28 76 73 69



03 28 76 73 71

Avant-propos

Ce cyber-cours de niveau Bac+5 s'inscrit dans le cadre de l' [Atelier de Recherches Pédagogiques de l'EUDIL](#) dirigé par M. Bernard Boittiaux.

Ce cours ne prétend pas à l'exhaustivité. Nous voulons simplement montrer comment on peut résoudre certains problèmes d'ingénierie ou de recherche à l'aide de tels algorithmes. Pour aborder les notions exposées, il ne faut pas avoir de "pré-requis" importants, mais beaucoup de curiosité car nous allons croiser sur notre chemin de vastes domaines de recherche en pleine effervescence, au carrefour de l'informatique, de l'ingénierie, des mathématiques, et des sciences naturelles. Une bonne connaissance de la programmation sera nécessaire pour ceux qui voudront écrire leur propre Algorithme Génétique. Pour commencer ils pourront charger des codes tout fait sur le web (voir la page [liens](#)).

En bas de chaque page des icônes vous permettront de naviguer facilement dans ce cours. L'icône "maison" permet de revenir au sommaire. Les icônes "flèche vers le haut" et "flèche vers le bas" permettent respectivement de revenir à la section précédente et de passer à la section suivante.

Je serais très heureux de connaître vos commentaires, critiques, suggestions afin de pouvoir améliorer la qualité pédagogique de ce cours.



Dernière mise à jour : 13 janvier 2001.

Auteur : [Vincent MAGNIN](#)

Modélisation

La *modélisation* est une méthode parmi d'autres pour concevoir ou étudier un *dispositif*. Ce terme large, que nous emploierons tout au long de ce cours, désignera l'objet, matériel ou non, que nous voulons étudier et optimiser : une molécule, un composant électronique, un process industriel, un système complexe, un programme informatique, etc.

La modélisation informatique devient de plus en plus une expérimentation virtuelle. Ce n'est pas un substitut aux expérimentations réelles, mais un outil complémentaire qui permet de diminuer au maximum des expériences de plus en plus coûteuses. Un modèle efficace peut apporter un gain de temps et une économie substantielle dans la conception d'un dispositif.

Il existe des modèles de plusieurs types : des modèles analytiques, des modèles physiques, etc. Les modèles deviennent de plus en plus complexes et plus globaux, ce qui est rendu possible d'une part par les avancées théoriques et algorithmiques, et d'autre part par la montée en puissance exponentielle des ordinateurs. Le développement d'algorithmes d'optimisation complexes tels que les Algorithmes Génétiques permet d'aller encore plus loin dans l'utilisation de la modélisation.



Dernière mise à jour : 13 décembre 1999.
Auteur : Vincent MAGNIN

Optimisation d'un dispositif

Parmi les problèmes rencontrés par le chercheur et l'ingénieur, les problèmes d'optimisation occupent à notre époque une place de choix. Nous n'aborderons pas le problème de l'optimisation d'un point de vue mathématique, mais simplement du point de vue d'un ingénieur souhaitant optimiser un dispositif. Cette partie du cours sera donc loin d'être exhaustive.

La méthode de base de l'optimisation est la méthode d'*essai et d'erreur* : il s'agit de tester un certain nombre de solutions potentielles jusqu'à l'obtention d'une solution adéquate. C'est ce que nous faisons quand nous faisons varier "à la main" un paramètre, pour voir ce que ça donne. Finalement, c'est également ce que nous faisons quand nous faisons varier ce paramètre de façon quasi-continue et que nous traçons la courbe correspondante.

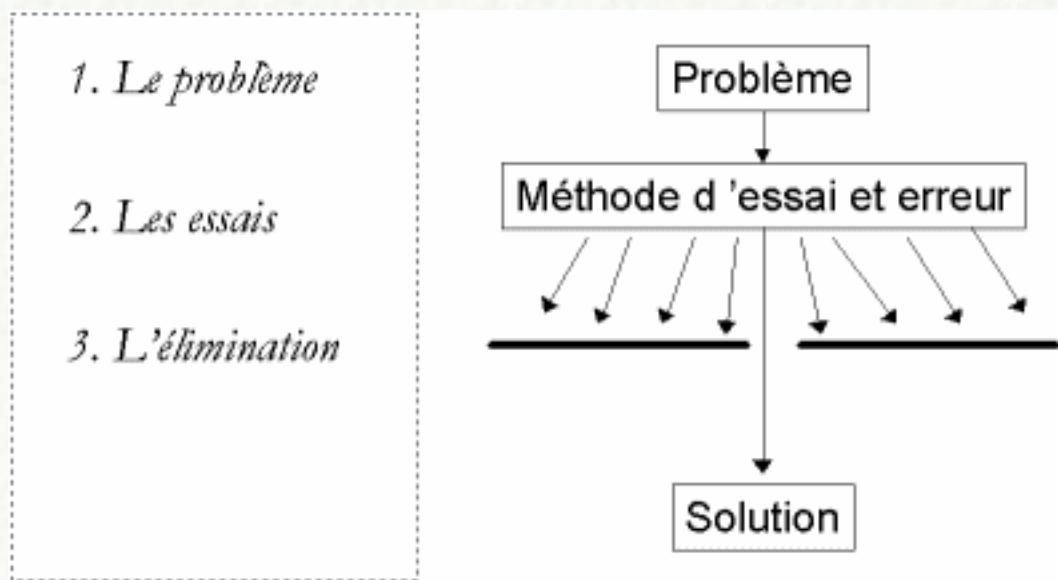


Figure 0 : on peut décrire l'apprentissage par essai et erreur par un schéma à trois niveaux [K. Popper, 1994].

Considérons la figure 0. Soit un problème : une situation nouvelle se présente et nécessite une solution qui n'est pas connue. La *méthode d'essai et d'erreur* est alors employée : un grand nombre de solutions potentielles sont mises à l'essai. Les solutions inadéquates sont éliminées, jusqu'à ce qu'un essai se révèle satisfaisant. Ce schéma est très général. On peut l'appliquer aussi bien au comportement des organismes vivants qu'à l'évolution des espèces (nous y reviendrons plus loin), à l'évolution des sciences ou au processus d'optimisation : "*dans le fond, ce procédé semble être le seul qui soit logiquement possible.*" [Popper, 1994] Les algorithmes informatiques constituent un puissant outil pour l'automatisation de ce processus.

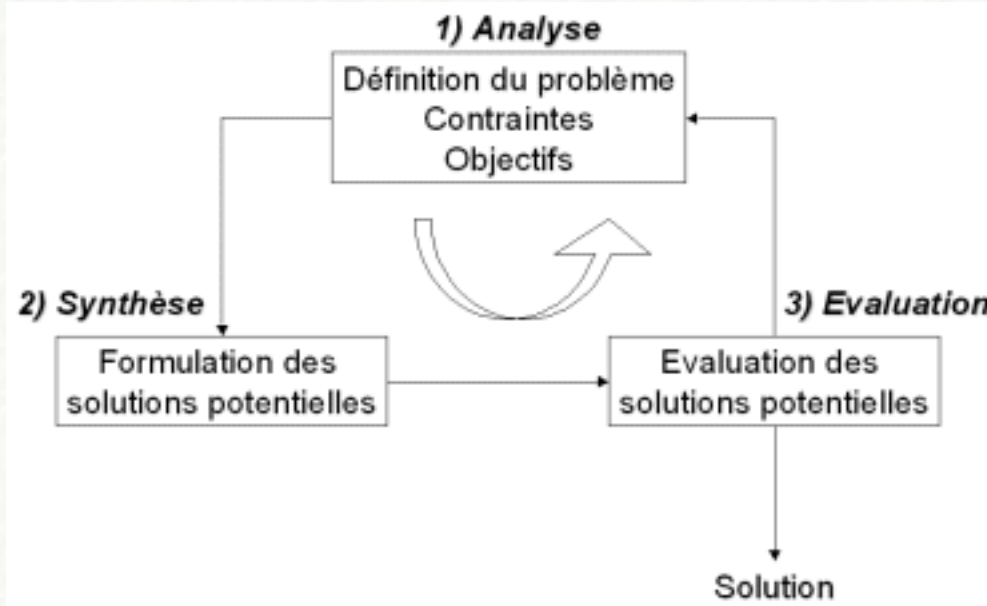


Figure 1 : processus d'optimisation selon Asimow [Balachandran, 1993].

La figure 1 présente les trois étapes du processus d'optimisation : analyse, synthèse et évaluation [Balachandran, 1993]. Tout d'abord, il convient d'analyser le problème et d'opérer un certain nombre de choix préalables :

- **Variables du problème.** Quels sont les paramètres intéressants à faire varier ?
- **Espace de recherche.** Dans quelles limites faire varier ces paramètres ?
- **Fonctions objectif.** Quels sont les objectifs à atteindre ?
- **Méthode d'optimisation.** Quelle méthode choisir ?

Une fois effectués ces différents choix, la méthode choisie synthétise des solutions potentielles qui sont évaluées, puis éliminées jusqu'à obtention d'une solution acceptable. Si nécessaire, le problème peut être redéfini à partir des solutions déjà obtenues.

Variables du problème

Les variables peuvent être de natures diverses. Par exemple, pour un composant électronique il peut s'agir de sa forme et ses dimensions géométriques, des matériaux utilisés, des conditions de polarisation, etc.

C'est à l'utilisateur de définir les variables du problème. Il peut avoir intérêt à faire varier un grand nombre de paramètres afin d'augmenter les degrés de liberté de l'algorithme.

Par la suite nous désignerons par x_1, \dots, x_n les n variables du problème. Celles-ci peuvent être réelles, complexes, entières, booléennes, etc. Ici nous les supposons réelles.

Espace de recherche

Dans certains algorithmes d'optimisation, tels que les *Stratégies d'Evolution*, l'espace de recherche est infini : seule la population initiale est confinée dans un espace fini [Bäck, 1991]. Mais dans le cas des algorithmes de type *Monte Carlo* et *Génétique*, il est généralement nécessaire de définir un espace de recherche fini. Cette limitation de l'espace de recherche n'est généralement pas problématique. En

effet, ne serait-ce que pour des raisons technologiques ou informatiques (taille de la fenêtre de modélisation), les intervalles de définition des variables sont en général limités. De plus, la plupart du temps on a au moins une idée des ordres de grandeur des variables du problème.

Nous désignerons par $x_{i \min}$ et $x_{i \max}$ les bornes de chaque variable x_i :

$$x_{i \min} \leq x_i \leq x_{i \max} \quad \forall i \in [1; n]$$

Fonctions objectif et fonction d'adaptation

Les grandeurs à optimiser peuvent être par exemple une consommation, un rendement, un facteur de transmission, un profit, la faisabilité technologique, un coût, une durée de développement, etc. Un algorithme d'optimisation nécessite généralement la définition d'une fonction rendant compte de la pertinence des solutions potentielles, à partir des grandeurs à optimiser. Il s'agit de la *fonction d'adaptation* f (ou *fitness function* en terminologie anglo-saxonne). Attention, l'algorithme convergera vers un optimum de cette fonction, quelle que soit sa définition. La pertinence de la solution dépendra donc de la pertinence de la " question " posée à l'ordinateur. La fonction f doit donc traduire en langage mathématique le désir de l'utilisateur.

Objectif unique

Dans le cas d'un *objectif unique*, la définition de la fonction d'adaptation ne pose généralement pas de problème. Par exemple, si l'on se fixe l'objectif de trouver un dispositif dont le rendement est maximum, cette fonction sera égale au rendement. Le calcul de la fonction d'adaptation se fait en deux étapes. On commence par évaluer les caractéristiques des solutions potentielles en utilisant le modèle. Puis on calcule la fonction d'adaptation à partir de ces caractéristiques. Dans le cadre de ce travail, nous utiliserons des fonctions d'adaptation normalisées sur l'intervalle $[0 ; 1]$. La valeur 0 correspond à une solution totalement inadaptée et la valeur 1 à une solution parfaite.

Objectifs multiples

Les problèmes d'optimisation doivent souvent satisfaire des *objectifs multiples*, dont certains sont concurrents. Une méthode classique consiste à définir des *fonctions objectif* f_i , traduisant chaque objectif à atteindre, et de les combiner au sein de la fonction d'adaptation. On établit ainsi un compromis. Le plus simple est de se ramener à une somme pondérée des fonctions objectif [Fonseca, 1993]:

$$f = \sum_i a_i f_i \quad (1)$$

où les poids a_i doivent être tels que la fonction d'adaptation reste bornée dans l'intervalle $[0 ; 1]$. Remarquons que certains a_i peuvent être négatifs afin de tenir compte de certaines contraintes du problème. C'est à l'utilisateur de fixer convenablement les poids a_i . On peut souvent classer les objectifs par importance mais les poids seront généralement adaptés par tâtonnement, jusqu'à l'obtention d'une solution acceptable. Le processus d'optimisation a beau être automatisé, l'utilisateur doit donc quand même optimiser " à la main " la définition de la fonction d'adaptation. Or, si dans la littérature les algorithmes sont analysés sous toutes leurs coutures, ce problème délicat est souvent laissé dans l'ombre, simplement parce que l'on se situe ici au cœur des recherches actuelles en

optimisation.

A la place d'une somme, on peut également utiliser un produit du type :

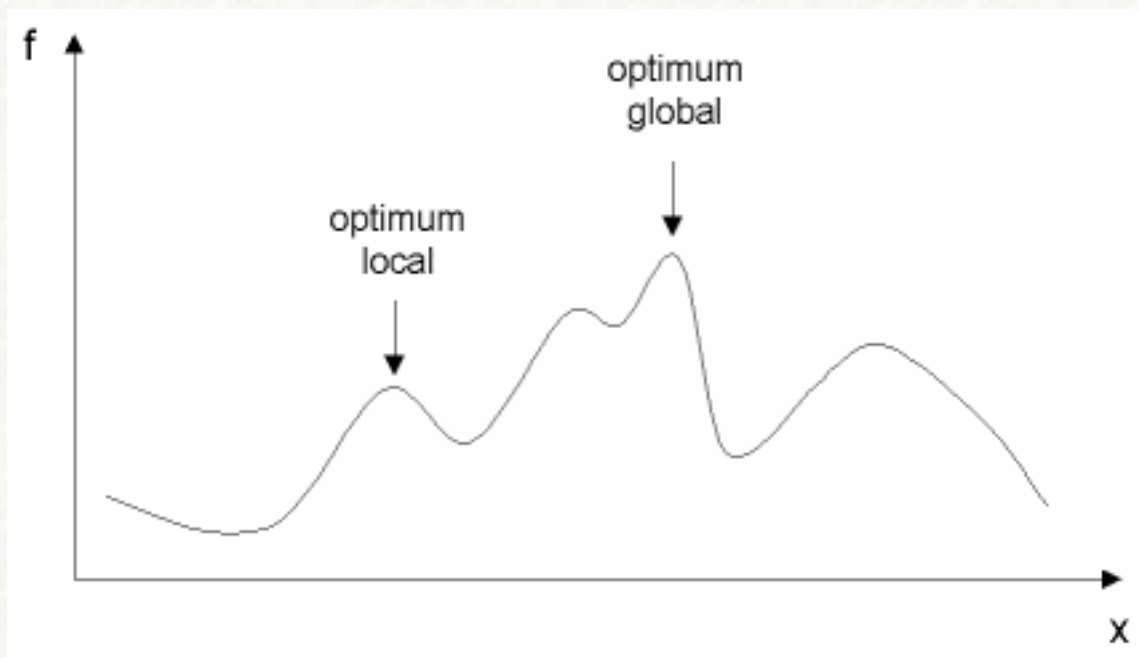
$$f = \prod_i f_i^{\alpha_i} \quad (2)$$

ou des expressions plus complexes.

Il faut néanmoins être conscient des effets d'une telle combinaison des objectifs. En effet, deux solutions potentielles dont les fonctions objectif n'ont pas la même valeur peuvent aboutir à une même valeur de la fonction d'adaptation. De plus, un algorithme utilisant une telle approche ne convergera que vers une seule solution alors qu'il existe peut-être toute une famille de solutions remplissant les objectifs fixés. L'optimisation à objectifs multiples est un domaine de recherche très actif actuellement, de par les enjeux économiques et industriels auxquels il répond. Des concepts tels que les niches écologiques ou l'optimalité de Pareto semblent prometteurs pour la résolution de ce genre de problème [Fonseca, 1995].

Méthodes d'optimisation

Une fois définie la fonction à optimiser, il s'agit de choisir une méthode adaptée au problème posé. 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.



Méthodes déterministes

La recherche des extrema d'une fonction f revient à résoudre un système de n équations à n inconnues, linéaire ou non :

$$\frac{\partial f}{\partial x_i}(x_1, \dots, x_n) = 0 \quad (3)$$

On peut donc utiliser des méthodes classiques telles que la méthode du gradient ou la méthode de Gauss-Seidel [Ciarlet, 1990 et Nougier, 1987]. En général, l'utilisation de ces méthodes nécessite comme étape préliminaire la localisation des extrema. Celle-ci peut être faite, par exemple, sur un graphique ou par une discrétisation fine de l'espace de recherche. La fonction à optimiser est évaluée en chacun des points de discrétisation. La valeur maximale est alors considérée comme une bonne approximation de l'optimum de la fonction. Cette méthode est brutale et le temps de calcul augmentera exponentiellement en fonction du nombre de variables.

En effet, considérons une optimisation sur huit variables. Si on discrétise l'intervalle de définition de chaque variable en seulement 3 points, une exploration systématique nécessite $3^8 = 6561$ exécutions du modèle. Dans le cas d'un modèle physique nécessitant 10 secondes de calcul sur un PC standard, cela représente 18 heures de calcul, pour un résultat inutilisable !

Méthodes non-déterministes

Ces méthodes font appel à des tirages de nombres aléatoires. Elles permettent d'explorer l'espace de recherche plus efficacement. Citons entre autres [Beasley, 1993a] :

- **Les méthodes Monte Carlo** : la fonction est évaluée en un grand nombre de points choisis aléatoirement.
- **Les méthodes hybrides** : on peut par exemple utiliser la méthode des gradients en partant d'un grand nombre de points choisis aléatoirement. On peut ainsi espérer déterminer au fur et à mesure tous les optima locaux de la fonction.
- **Le recuit simulé** : on effectue des déplacements aléatoires à partir d'un point initial. Si un déplacement mène à une valeur plus grande de la fonction f , il est accepté. Sinon, il est accepté avec une probabilité :

$$p = e^{-\frac{|\Delta f|}{kT}} \quad (4)$$

où " Δf " est la variation de la fonction, T est assimilé à une température qui décroît au cours du temps et k est une constante. Cette méthode est basée sur une analogie avec les processus de recuit utilisés en métallurgie et qui visent à atteindre une configuration d'énergie minimale ($-f$ est alors assimilée à une énergie).

- **Les Algorithmes Evolutionnaires**: le principe est de simuler l'évolution d'une population d'individus divers auxquels on applique différents opérateurs génétiques 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.

Compromis exploration et exploitation

Si nous opposons *exploration* et *exploitation* de l'espace de recherche, nous pouvons dire que les méthodes Monte Carlo permettent une bonne exploration puisque tout point a une probabilité identique d'être atteint, mais qu'il n'y a pas d'exploitation des résultats déjà obtenus. Avec la méthode

des gradients, l'exploration est moindre mais l'exploitation des données précédentes par l'intermédiaire des gradients permet une bonne recherche locale. Enfin, les Algorithmes Evolutionnaires offrent un bon compromis entre exploration et exploitation [Beasley, 1993a]. Nous allons nous y intéresser de plus près.



Dernière mise à jour : 13 décembre 1999.

Auteur : Vincent MAGNIN

Optimisation par méthode Monte Carlo

Les *méthodes Monte Carlo* consistent en des simulations expérimentales ou informatiques de problèmes mathématiques ou physiques, basées sur le tirage de nombres aléatoires. Généralement on utilise en fait des séries de nombres [pseudo-aléatoires](#) générées par des algorithmes spécialisés. Les propriétés de ces séries sont très proches de celles d'une véritable suite aléatoire.

On peut par exemple approcher la valeur de Pi par une méthode Monte Carlo : on considère le cercle de rayon 1 inscrit dans un carré de côté 2. On tire un grand nombre de points au hasard dans ce carré. On calcule le rapport entre le nombre de points qui sont tombés dans le cercle et le nombre de points total. En multipliant ce rapport par 4 (aire du carré), on obtient une approximation de l'aire du disque, donc de Pi. Citons également la méthode des [Aiguilles de Buffon](#).

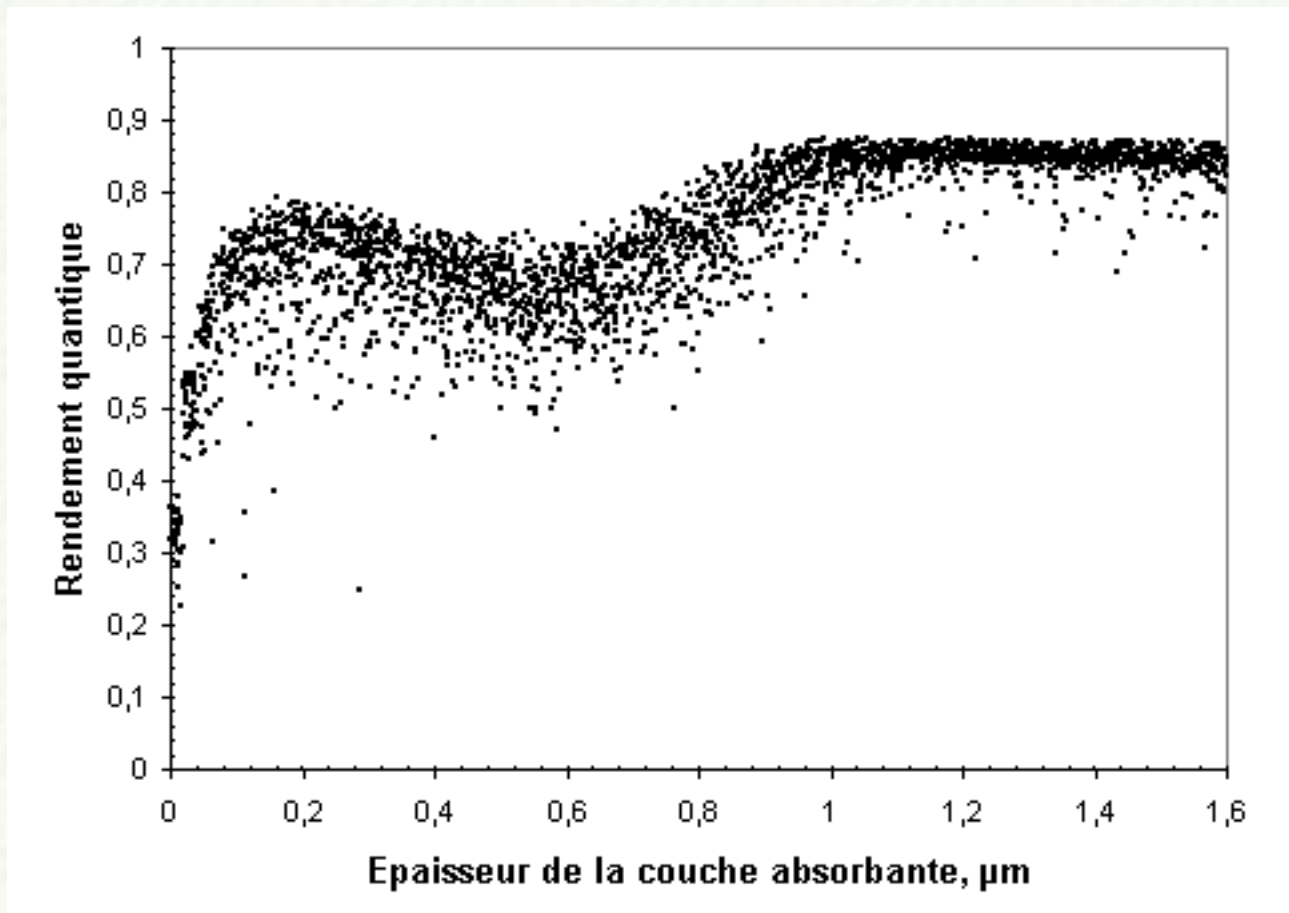


Figure 1 : étude Monte Carlo de 2854 photodiodes. On notera la résonance vers 0,2 μm et le palier après 1 μm .

La figure 1 illustre l'optimisation du rendement quantique d'une photodiode par méthode Monte Carlo. Il s'agissait ici d'optimiser l'épaisseur de plusieurs couches épitaxiales. Pour chacune des 2854 structures présentées, on a tiré au sort (à l'aide de nombres pseudo-aléatoires) les épaisseurs des couches dans un espace de recherche donné. Chaque structure a alors été évaluée à l'aide d'un modèle physique de propagation de la lumière qui nous a fourni le rendement quantique du composant.

Pour obtenir des résultats statistiquement fiables, il est nécessaire d'évaluer des centaines de structures, typiquement un millier. Ce nombre est bien sûr d'autant plus grand que le nombre de variables est élevé. Plus le nombre de structures évaluées est grand, plus on trouvera des structures de

rendement quantique élevé. A la rigueur, un temps de calcul infini nous permettrait de trouver la structure optimale de notre espace de recherche. Le temps de calcul constitue une des faiblesses de ces méthodes dans le cadre du problème de l'optimisation.

Le grand avantage de cette méthode est sa simplicité. Elle permet entre autres de visualiser l'effet de différents paramètres et de donner ainsi des orientations, d'étudier des structures intéressantes qui auraient été *a priori* écartées et de trouver facilement des structures que l'on n'aurait pas aussi bien optimisées " à la main ".



Dernière mise à jour : 13 décembre 1999.

Auteur : Vincent MAGNIN

Algorithmes Evolutionnaires et Algorithmes Génétiques

" En vérité, aux tout premiers temps, naquit Chaos "

Hésiode

" To create a little flower is the labour of ages. "

William Blake

Les *Algorithmes Evolutionnaires* (AE) sont inspirés du concept de *sélection naturelle* élaboré par [Charles Darwin](#). Le vocabulaire employé est directement calqué sur celui de la théorie de l'évolution et de la génétique. Nous parlerons donc d'*individus* (solutions potentielles), de *population*, de *gènes* (variables), de *chromosomes*, de *parents*, de descendants, de reproduction, de *croisement*, de *mutations*, etc. Et nous nous appuierons constamment sur des analogies avec les phénomènes biologiques.

- [La sélection naturelle](#)
- [Algorithmes Evolutionnaires et Génétiques](#)
 - [Généralités](#)
 - [Historique](#)
 - [Applications](#)

La sélection naturelle

S'il n'existe pas de preuve générale de l'efficacité des AE, il est par contre aisé de constater l'efficacité de la sélection naturelle dans le monde vivant. Si l'on adhère à ce paradigme, il est clair que l'évolution a permis l'émergence d'organismes étonnamment adaptés à leur environnement. Les AE sont conçus par analogie avec ce processus d'évolution biologique et tirent leur puissance des mêmes mécanismes, que nous allons rappeler sommairement.

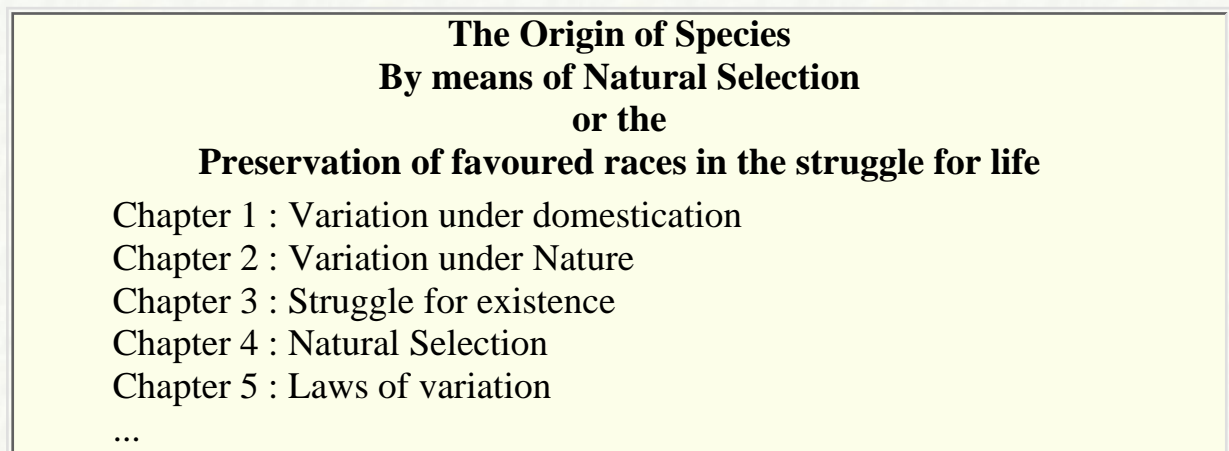


Figure 1 : titre complet et premiers chapitres de *The Origin of Species* (1859), dans la langue de l'auteur.

Dans [The Origin of Species](#) (1859), Darwin montre que l'apparition d'espèces distinctes se fait par le

biais de la sélection naturelle de *variations individuelles* (voir figure 2). Cette sélection naturelle est fondée sur la lutte pour la vie, due à une population tendant naturellement à s'étendre mais disposant d'un espace et de ressources finis. Il en résulte que les individus les *plus adaptés* (*the fittest* en anglais) tendent à survivre plus longtemps et à se reproduire plus aisément. Le terme " adapté " se réfère à l'environnement, que l'on peut définir comme étant l'ensemble des conditions externes à un individu, ce qui inclut les autres individus. Les lois de variation (croisements et mutations) furent expliquées plus tard par [Mendel](#), puis par la génétique moderne.

Le point clef est l'apparition, par hasard, de variations individuelles. C'est ce hasard qui permet d'expliquer les phénomènes d'évolution et d'adaptation sans avoir recours ni à une création, ni à une modification directe de l'hérédité par le milieu, ni même à une finalité. Dans ce cadre, les espèces évoluent et s'adaptent à leur environnement mais sans tendre vers aucun but prédéterminé, et même sans forcément tendre vers plus de complexité (la simplicité est parfois préférable).

Notons que les mêmes principes sont à l'origine de l'efficacité du système immunitaire : celui-ci produit d'énormes quantités d'anticorps aléatoires. Ceux qui se révèlent par hasard efficaces sont alors, par un mécanisme de rétroaction, produits en plus grande quantité [Monod 1970 et Wills, 1991]. Enfin, les principes de l'évolution sont maintenant utilisés par les biologistes [pour produire de nouvelles enzymes](#) : de très nombreuses séquences d'ADN synthétisées aléatoirement sont soumises à une sélection et des mutations (évolution en éprouvette) jusqu'à obtention des propriétés désirées.

Dernière remarque : la sélection naturelle n'est certainement pas le seul mécanisme en oeuvre dans l'évolution des espèces. Si la compétition a certaines vertus, et même des vertus certaines, chacun sait bien que "l'union fait la force", la Nature la première. Pour preuve, je citerai certaines étapes de l'évolution telles que l'intégration des mitochondries aux cellules, le passage des organismes monocellulaires aux organismes pluricellulaires, les phénomènes de symbiose et bien sûr la vie en société. L'algorithme très simple que nous présentons ne prend pas en compte ces mécanismes, mais rien n'interdit de le faire !

Algorithmes Evolutionnaires et Algorithmes Génétiques

Généralités

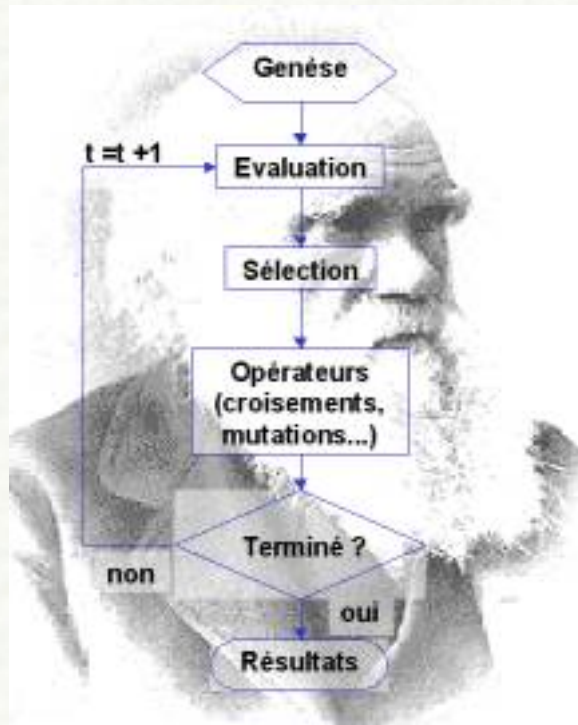


Figure 2 : organigramme d'un Algorithme Evolutionnaire.

La Figure IV-2.2 présente l'organigramme d'un AE. Il s'agit de simuler l'évolution d'une population d'individus divers (généralement tirée aléatoirement au départ) à laquelle on applique différents opérateurs (recombinaisons, mutations...) et que l'on soumet à une sélection, à chaque génération. Si la sélection s'opère à partir de la fonction d'adaptation, alors la population tend à s'améliorer [Bäck, 1996 et Bäck, 1997]. Un tel algorithme ne nécessite aucune connaissance du problème : on peut représenter celui-ci par une boîte noire comportant des entrées (les variables) et des sorties (les fonctions objectif). L'algorithme ne fait que manipuler les entrées, lire les sorties, manipuler à nouveau les entrées de façon à améliorer les sorties, etc. [Whitley, 1993] C'est ainsi qu'ont procédé les éleveurs pendant des millénaires : ils ont réussi à modifier selon leurs désirs de nombreuses espèces animales sans connaissance en génétique ou biologie moléculaire.

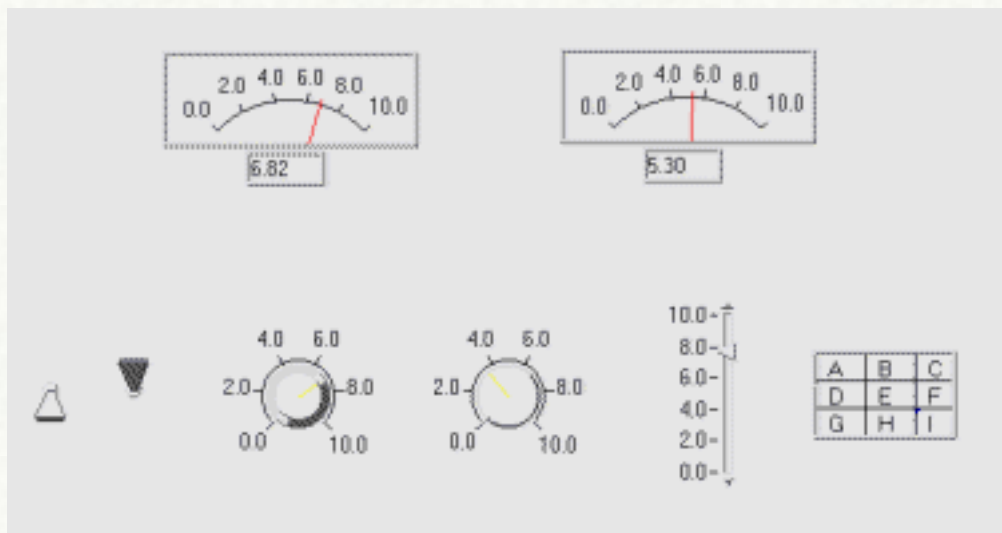


Figure 6 : on peut symboliser un problème d'optimisation par une *boîte noire*. Les sorties (en haut) peuvent être optimisées en manipulant les entrées (en bas) jusqu'à obtenir le résultat voulu, sans aucune connaissance de l'intérieur de la boîte. Les AE sont une automatisation de cette méthode d'essai et erreur.

Les AE constituent une approche originale : il ne s'agit pas de trouver une solution analytique exacte,

ou une bonne approximation numérique, mais de trouver des solutions satisfaisant au mieux différents critères, souvent contradictoires. S'ils ne permettent pas de trouver à coup sûr la solution optimale de l'espace de recherche, du moins peut-on constater que les solutions fournies sont généralement meilleures que celles obtenues par des méthodes plus classiques, pour un même temps de calcul.

Historique

Les AE font partie du champ de l'Intelligence Artificielle (IA). Il s'agit d'IA de bas niveau, inspirée par " l'intelligence " de la Nature. Intelligence que l'on peut définir de la façon suivante : " *the capability of a system to adapt its behaviour to meet its goals in a range of environments* " [Fogel, 1995].

Trois types d'AE ont été développés isolément et à peu près simultanément, dans les années 60, par différents scientifiques : les *Algorithmes Génétiques*, les *Stratégies d'Evolution*, et la *Programmation Evolutionnaire*. Présentant des différences marquées à l'origine, ils tendent de plus en plus à se confondre suite à leurs emprunts respectifs [Bäck, 1997]. Dans les années 90, ces trois champs ont commencé à sortir de leur isolement et ont été regroupés sous le terme anglo-saxon d'*Evolutionary Computation*. Ainsi en avril 1997, un [*IEEE Transactions on Evolutionary Computation*](#) a vu le jour [Fogel, 1997].

Notons que les AE incluent également la *Programmation Génétique* qui consiste à faire évoluer le code d'un logiciel afin qu'il remplisse au mieux certaines tâches. Citons enfin le domaine de la [Vie Artificielle](#) où l'on tente de reproduire les mécanismes de la vie dans la mémoire d'un ordinateur afin de mieux comprendre l'organisation et l'évolution du vivant.

Parmi les AE que nous venons de citer, nous avons choisi de traiter des Algorithmes Génétiques (AG). En effet, ils nous paraissent concilier au mieux puissance, généralité et facilité de programmation. Leur particularité est qu'ils sont fondés sur le Néo-Darwinisme, c'est-à-dire l'union de la théorie de l'évolution et de la génétique moderne. Ainsi, les variables sont généralement codées en binaire (par analogie avec les quatre lettres de l'alphabet génétique) sous forme de gènes dans un chromosome. Des opérateurs génétiques (croisement, mutation) sont appliqués à ces chaînes binaires que sont les chromosomes [Bäck, 1996 et Goldberg, 1994].

Applications

Les applications des AG sont multiples : optimisation de fonctions numériques difficiles (discontinues, multimodales, bruitées...), traitement d'image (alignement de photos satellites, reconnaissance de suspects...), optimisation d'emplois du temps, optimisation de design, contrôle de systèmes industriels [Beasley, 1993a], apprentissage des réseaux de neurones [Renders, 1995], etc. Les AG peuvent être utilisés pour contrôler un système évoluant dans le temps (chaîne de production, centrale nucléaire...) car la population peut s'adapter à des conditions changeantes. En particulier, ils supportent bien l'existence de bruit dans la fonction à optimiser. Ils peuvent aussi servir à déterminer la configuration d'énergie minimale d'une molécule ou à modéliser le comportement animal.

Les AG sont également utilisés pour optimiser des réseaux (câbles, fibres optiques, mais aussi eau, gaz...), des circuits VLSI [Beasley, 1993a], des antennes [Reineix, 1997]... Ils peuvent être utilisés pour trouver les paramètres d'un modèle petit-signal à partir des mesures expérimentales [Menozzi, 1997]. Des commutateurs optiques adiabatiques ont été optimisés à l'aide des Stratégies d'Evolution (autres AE) chez SIEMENS AG [Moosburger, 1997]. On envisage l'intégration d'AG dans certaines puces électroniques afin qu'elles soient capables de se reconfigurer automatiquement en fonction de

leur environnement (*Evolving Hardware* en anglais).

Nous allons présenter les techniques de base que nous avons testées.



Dernière mise à jour : 14 juin 2001.
Auteur : Vincent MAGNIN

Algorithme Génétique

Codage des variables

La première étape est de définir et de coder convenablement le problème. A chaque variable d'optimisation x_i (à chaque paramètre du dispositif), nous faisons correspondre un *gène*. Nous appelons *chromosome* un ensemble de gènes. Chaque dispositif est représenté par un *individu* doté d'un génotype constitué d'un ou plusieurs chromosomes. Nous appelons *population* un ensemble de N individus que nous allons faire évoluer.

D'un point de vue informatique, nous utilisons dans notre algorithme un codage binaire. C'est-à-dire qu'un gène est un entier long (32 bits). Un chromosome est un tableau de gènes (figure 4 et figure 5). Un individu est un tableau de chromosomes. La population est un tableau d'individus. Notons qu'on pourrait aussi utiliser d'autres formes de codage (réel, codage de Gray...) [Davis, 1991].

On aboutit à une structure présentant cinq niveaux d'organisation (figure 3), d'où résulte le comportement complexe des AG :

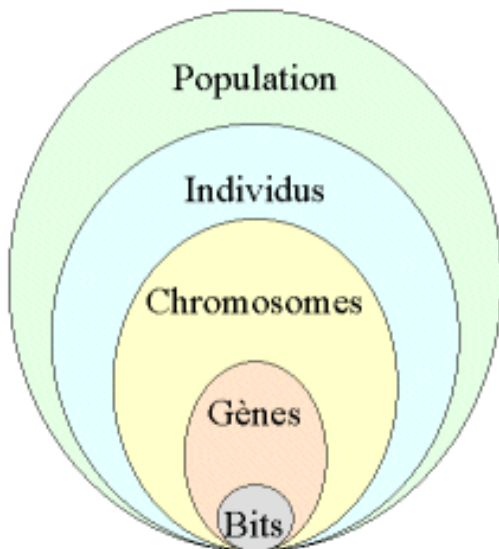


Figure 3 : les cinq niveaux d'organisation de notre Algorithme Génétique.

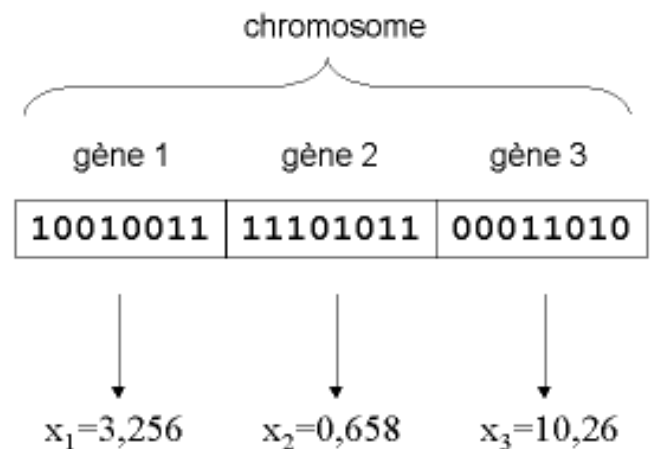


Figure 4 : illustration schématisque du codage des variables d'optimisation x_i .

Un des avantages du codage binaire est que l'on peut ainsi facilement coder toutes sortes d'objets : des réels, des entiers, des valeurs booléennes, des chaînes de caractères... Cela nécessite simplement l'usage de fonctions de codage et décodage pour passer d'une représentation à l'autre.

Rappelons que dans cette étude les n variables sont supposées réelles. Nous considérons un espace de recherche fini :

$$x_{i\min} \leq x_i \leq x_{i\max} \quad \forall i \in [1, n]$$

Afin de coder nos variables réelles en binaire, nous discrétisons l'espace de recherche. Ainsi un codage sur 32 bits implique une discrétisation des intervalles en $g_{\max}=2^{32}-1=4\,294\,967\,295$ valeurs discrètes. Notons au passage que si cette discrétisation est plus fine que celle du modèle physique utilisé, la fonction est assimilable à une fonction escalier, si on la considère à une échelle suffisamment petite.

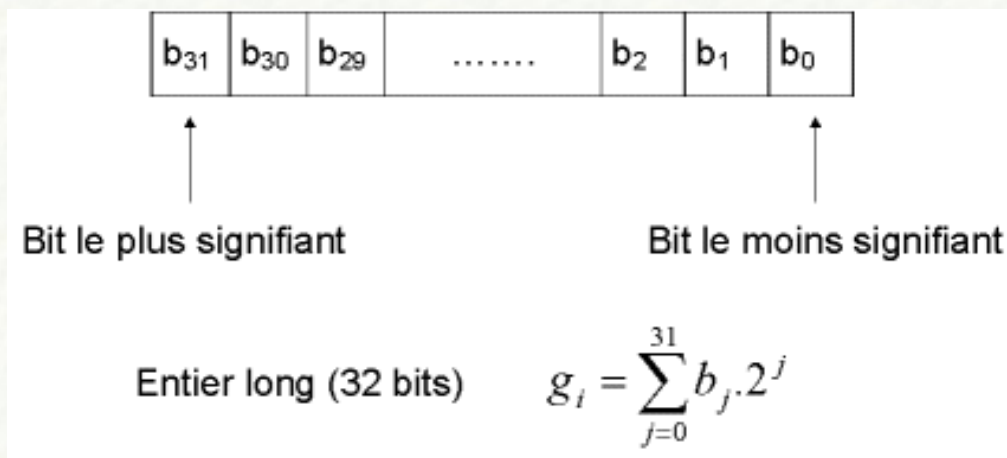


Figure 5 : chaque gène (chaque paramètre du dispositif) est codé par un entier long (32 bits).

A chaque variable réelle x_i on associe donc un entier long g_i :

$$0 \leq g_i \leq g_{\max} \quad \forall i \in [1, n]$$

Les formules de codage et décodage sont alors les suivantes :

$$g_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \cdot g_{\max}$$

$$x_i = x_{\min} + (x_{\max} - x_{\min}) \cdot \frac{g_i}{g_{\max}} \quad (5)$$

Genèse de la population

La première étape de l'algorithme est la genèse de la population, c'est-à-dire le choix des dispositifs de départ que nous allons faire évoluer. On pourrait prendre des individus régulièrement répartis dans l'espace. Néanmoins, une initialisation aléatoire est plus simple à réaliser : les valeurs g_i des gènes sont tirées au hasard selon une distribution uniforme. Notons qu'on peut, si nécessaire, introduire des individus déjà calculés.

Nous discuterons plus loin de la taille N de cette population, mais nous pouvons déjà dire qu'elle résultera d'un compromis entre temps de calcul et qualité de la solution.

Evaluation

L'évaluation de chaque dispositif est réalisée par le modèle utilisé. Si le dispositif possède plusieurs états de fonctionnement, le modèle peut être lancé plusieurs fois. Les résultats obtenus sont alors utilisés pour calculer les fonctions objectif et la fonction d'adaptation.

Notons que dans le cas d'un modèle physique, la majeure partie du temps de calcul sera probablement due à l'exécution de ce modèle. En effet, le reste de l'AG est essentiellement composé de manipulation d'entiers et de bits, donc très rapide.

Sélection – élimination

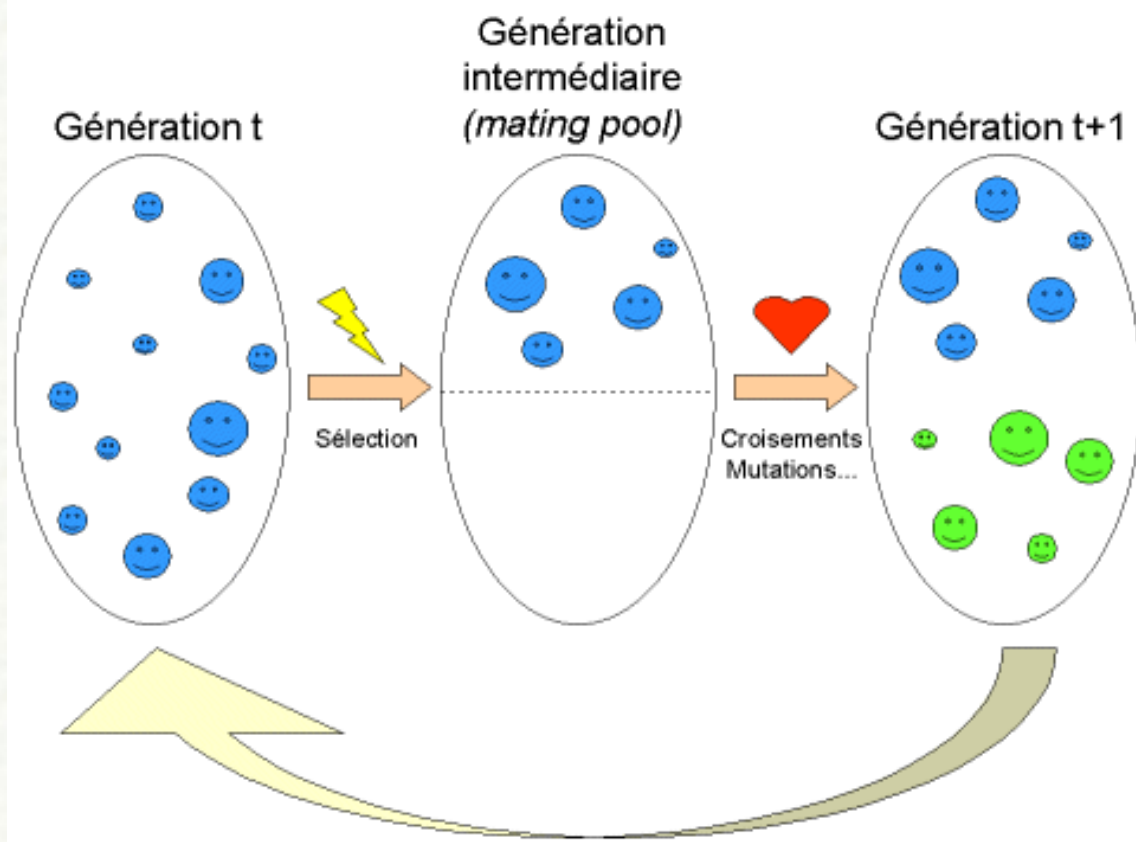


Figure 6 : représentation schématique du fonctionnement de notre algorithme.

Nous appelons *génération* la population à un instant t donné. Une fois réalisée l'évaluation de la génération, on opère une sélection à partir de la fonction d'adaptation. Seuls les individus passant l'épreuve de sélection (rappelons qu'il ne s'agit ici que de dispositifs !) peuvent accéder à la *génération intermédiaire* (*mating pool* en terminologie anglo-saxonne) et s'y reproduire. En fait, cette génération intermédiaire est deux fois plus petite ($N/2$ dispositifs) que la génération dont elle est issue (figure 6). Notre algorithme étant conçu de façon à ce que chaque couple d'individus parents donne naissance à deux enfants, nous aboutissons à nouveau à une génération entière à l'instant $t+1$.

Nous avons essayé deux techniques de sélection [Michalewicz, 1994] :

- " **$N/2$ -élitisme**" : les individus sont triés selon leur fonction d'adaptation. Seule la moitié supérieure de la population, correspondant aux meilleurs composants, est sélectionnée. Nous avons pu constater que cette méthode induisait une convergence prématurée de l'algorithme : la pression de sélection est trop forte. Il est en effet nécessaire de maintenir une diversité génétique suffisante dans la population, celle-ci constituant un réservoir de gènes pouvant être utiles par la suite. En effet, tout individu peut transmettre à sa descendance des gènes (paramètres de composant) qui, une fois combinés avec d'autres, peuvent se révéler intéressants. Nous avons donc essayé la méthode suivante.
- "**sélection par tournoi**" : deux individus sont choisis au hasard et combattent (on compare leurs fonctions d'adaptation) pour accéder à la génération intermédiaire. Le plus adapté l'emporte avec une probabilité $0,5 < p \leq 1$, que nous avons généralement prise égale à 1 (une valeur inférieure permet de réduire la pression de sélection si nécessaire). Cette étape est répétée jusqu'à ce que la génération intermédiaire soit remplie ($N/2$ composants). 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.

Les résultats de ce chapitre ont été obtenus avec cette dernière méthode, à laquelle nous avons associé une procédure d'élitisme : si par hasard l'individu le plus adapté (le meilleur dispositif) n'a pas été sélectionné, il est copié d'office dans la génération intermédiaire à la place d'un individu choisi aléatoirement.

Opérateur croisement

Une fois la génération intermédiaire à moitié remplie, les individus sont aléatoirement répartis en couples hermaphrodites. Les chromosomes (ensembles de paramètres) des parents sont alors copiés et recombinaison de façon à former deux descendants possédant des caractéristiques issues des deux parents. On forme ainsi la génération $t+1$ (Figure 6).

L'opérateur croisement favorise l'exploration de l'espace de recherche. Considérons deux gènes A et B pouvant être améliorés par mutation. Il est peu probable que les deux gènes améliorés A' et B' apparaissent par mutation dans un même individu. Mais l'opérateur de croisement permettra de combiner rapidement A' et B' dans la descendance de deux parents portant chacun un des gènes mutants. Il est alors possible que la présence simultanée des deux gènes produise un individu encore plus adapté [Dessales, 1996]. L'opérateur de croisement assure donc le brassage du matériel génétique et l'accumulation des mutations favorables. En termes plus concrets, cet opérateur permet de créer de nouvelles combinaisons des paramètres des composants.

Le phénomène de croisement est une propriété naturelle de l'ADN. C'est par analogie qu'ont été conçus les opérateurs de croisement dans les AG. Nous avons testé deux méthodes de croisement classiques :

- **" croisement en un point "** : on choisit au hasard un point de croisement, pour chaque couple (Figure 7). Notons que le croisement s'effectue directement au niveau binaire, et non pas au niveau des gènes. Un chromosome peut donc être coupé au milieu d'un gène.

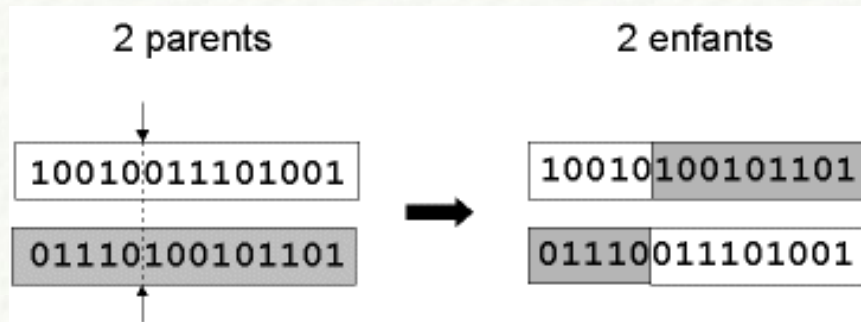


Figure 7 : représentation schématique du croisement en 1 point. Les chromosomes sont bien sûr généralement beaucoup plus longs.

- **" croisement un deux points "** : on choisit au hasard deux points de croisement (Figure 8). Par la suite, nous avons utilisé cet opérateur car il est généralement considéré comme plus efficace que le précédent [Beasley, 1993b]. Néanmoins nous n'avons pas constaté de différence notable dans la convergence de l'algorithme.

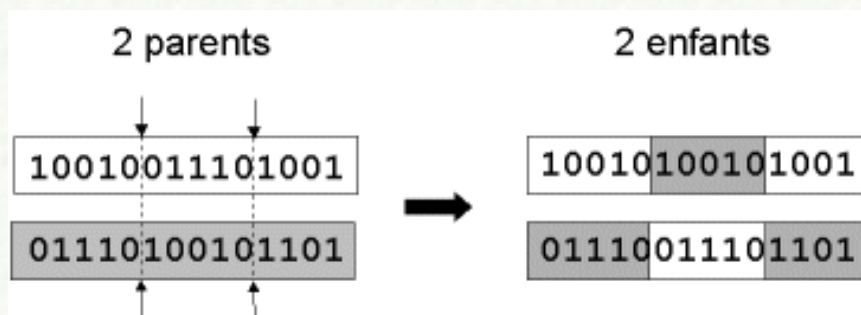


Figure 8 : représentation schématique du croisement en 2 points.

Notons que d'autres formes de croisement existent, du croisement en k points jusqu'au cas limite du croisement uniforme...

Opérateur mutation

Nous définissons une *mutation* comme étant l'inversion d'un bit dans un chromosome (Figure 9). Cela revient à modifier aléatoirement la valeur d'un paramètre du dispositif. Les mutations jouent le rôle de bruit et empêchent l'évolution de se figer. Elles permettent d'assurer une recherche aussi bien globale que locale, selon le poids et le nombre des bits mutés. De plus, elles garantissent mathématiquement que l'optimum global peut être atteint.

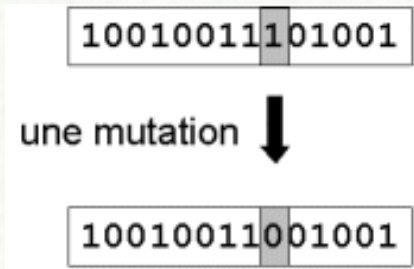


Figure 9 : représentation schématique d'une mutation dans un chromosome.

D'autre part, une population trop petite peut s'homogénéiser à cause des erreurs stochastiques : les gènes favorisés par le hasard peuvent se répandre au détriment des autres. Cet autre mécanisme de l'évolution, qui existe même en l'absence de sélection, est connu sous le nom de *dérive génétique*. Du point de vue du dispositif, cela signifie que l'on risque alors d'aboutir à des dispositifs qui ne seront pas forcément optimaux. Les mutations permettent de contrebalancer cet effet en introduisant constamment de nouveaux gènes dans la population [Beasley, 1993b].

Comment réaliser notre opérateur mutation ? De nombreuses méthodes existent. Souvent la probabilité de mutation p_m par bit et par génération est fixée entre 0,001 et 0,01. On peut prendre également $p_m=1/l$ où l est le nombre de bits composant un chromosome. Il est possible d'associer une probabilité différente à chaque gène. Et ces probabilités peuvent être fixes ou évoluer dans le temps.

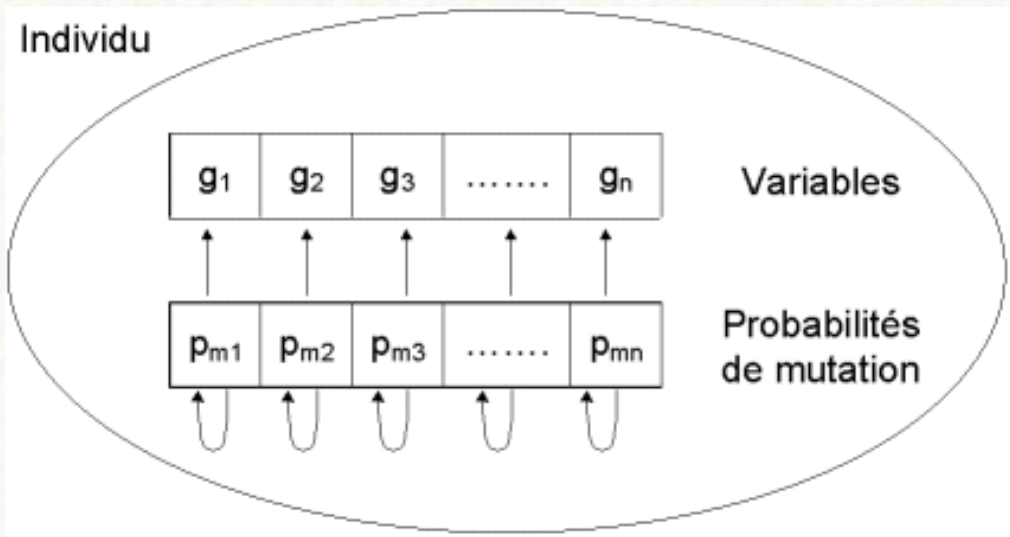


Figure 10 : principe de l'auto-adaptation. A chaque variable est associée sa propre probabilité de mutation, qui est elle-même soumise au processus d'évolution. L'individu possède donc un second chromosome codant ces probabilités.

Après divers essais, nous avons abouti à la méthode d'*auto-adaptation* des probabilités de mutation [Bäck, 1992]. Si dans un environnement stable il est préférable d'avoir un taux de mutation faible, la survie d'une espèce dans un environnement subissant une évolution rapide nécessite un taux de mutation élevé permettant une adaptation rapide. Les taux de mutation d'une espèce dépendent donc de leur environnement [Wills, 1991].

Pour prendre en compte cette formulation biologique et l'adapter à notre cas, nous avons introduit dans chaque individu (dispositif) un second chromosome (ensemble de paramètres) dont les gènes (paramètres) représentent les probabilités de mutation de chaque gène du premier chromosome (Figure 10). Ce second chromosome est géré de façon identique au premier, c'est-à-dire qu'il est lui-même soumis aux opérateurs génétiques (croisement et mutation). Cela revient à fixer les probabilités assurant la modification des valeurs des paramètres du composant en

fonction des valeurs d'un ensemble d'autres paramètres (les probabilités de mutation).

Lors de la genèse, les probabilités de mutation sont posées égales à 0,1 (valeur qui nous a paru la meilleure après plusieurs essais). Au cours du déroulement de l'algorithme, les gènes et les individus ayant des probabilités de mutation trop élevées ont tendance à disparaître. De même, les gènes ayant des probabilités de mutation trop faibles ne peuvent pas évoluer favorablement et tendent à être supplantés. Les probabilités de mutation dépendent donc du gène considéré et de la taille de la population. De plus, elles évoluent au cours du temps. Il y a donc auto-adaptation des probabilités de mutation.



Dernière mise à jour : 13 décembre 1999.

Auteur : Vincent MAGNIN

Convergence de l'Algorithme Génétique

- [Convergence](#)
- [Réglage des paramètres de l'AG](#)

Convergence et temps de calcul

On peut constater figure 11 que l'amélioration de la population est très rapide au début (*recherche globale*) et devient de plus en plus lente à mesure que le temps passe (*recherche locale*). Le bruit dans la moyenne est essentiellement dû aux mutations.

On voit que la valeur moyenne de la fonction d'adaptation a tendance à se rapprocher de celle de l'individu le plus adapté. Cela correspond à une uniformisation croissante de la population. Nous avons donc introduit dans notre logiciel une fenêtre graphique (Figure 12) permettant de visualiser la totalité de la population [Dessales, 1996]. Chaque ligne représente le génotype d'un individu, autrement dit les bits qui conduisent aux valeurs des paramètres d'un composant. Chaque pixel représente la valeur d'un bit dans son chromosome principal (blanc pour 0, et noir pour 1). Chaque groupe de 32 bits, entre deux graduations de la Figure 12, correspond à un gène. Les individus sont triés selon leur fonction d'adaptation : les plus adaptés correspondent aux lignes du haut, les moins adaptés aux dernières lignes. Dans cet exemple, nous avons représenté une population aux générations 1, 3, 10, 20, 50 et 130 (de gauche à droite, et de haut en bas). La taille de la population est de 200 individus (dispositifs) et le chromosome principal comprend 8 gènes (paramètres) de 32 bits.

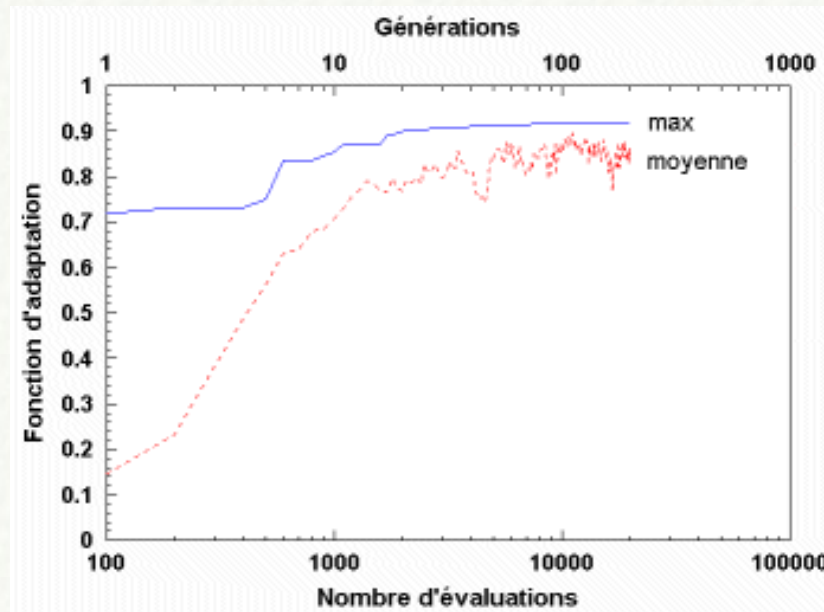


Figure 11 : exemple de convergence de l'AG. On a reporté la valeur de la fonction d'adaptation de l'individu le plus adapté de chaque génération (trait), et la moyenne des fonctions d'adaptation (pointillés), pour une population de 200 individus.

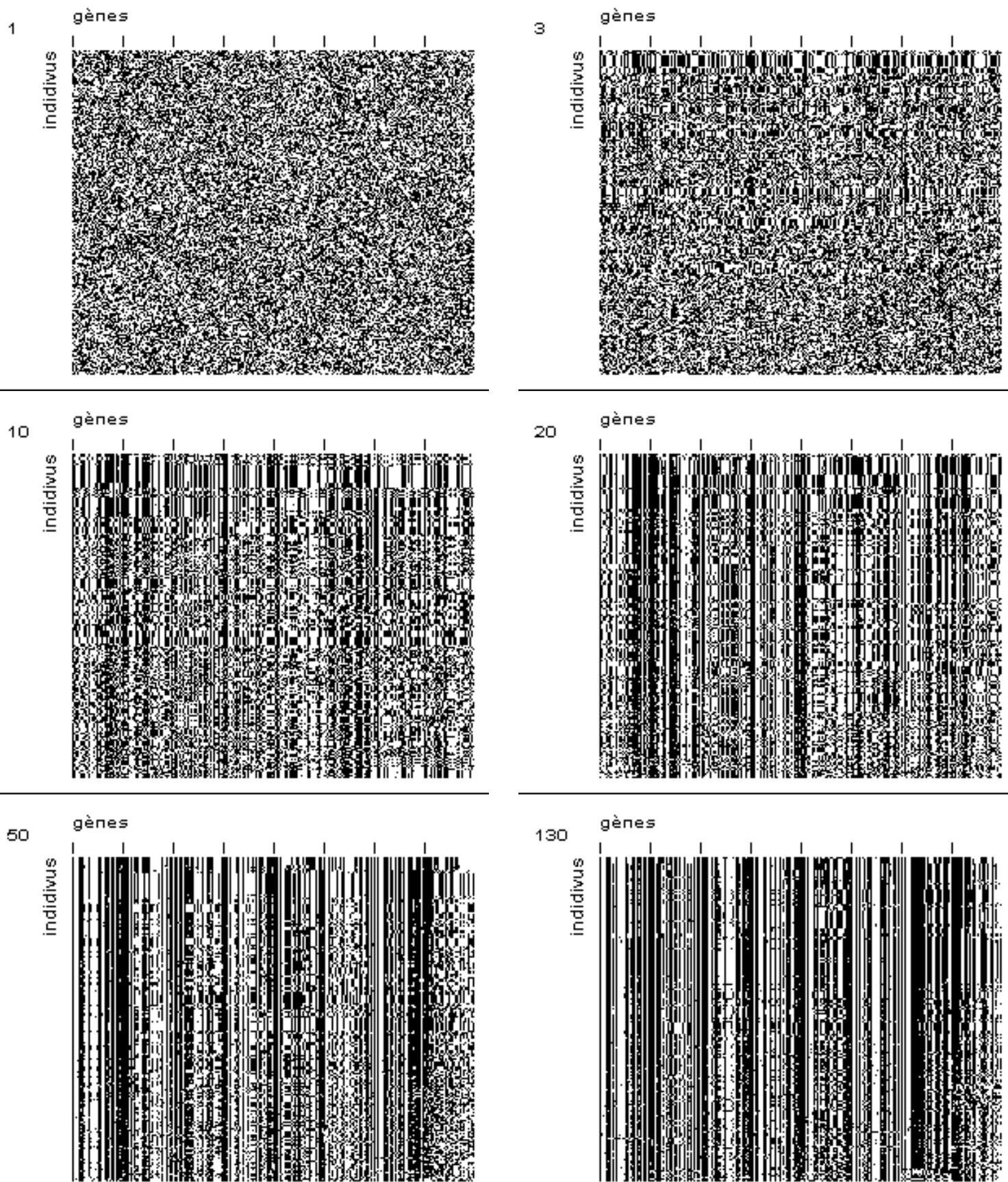


Figure 12 : générations 1, 3, 10, 20, 50 et 130. Chaque pixel représente un bit du chromosome principal d'un individu. Optimisation comportant 8 variables et 200 individus.

Ce genre de représentation permet de mieux comprendre le fonctionnement de l'algorithme. Tout d'abord la première image représente la population initiale tirée aléatoirement et ne présente donc aucun motif apparent. Les images suivantes (génération 3, 10 et 20) montrent l'apparition progressive de schémas, ce qui signifie que certaines valeurs des paramètres du composant se propagent dans la population parce qu'elles procurent un avantage aux individus qui en sont porteurs. Sur les deux dernières images (génération 50 et 130), on voit que la

population s'est à peu près uniformisée. L'utilisateur peut alors stopper l'algorithme.

En chargeant le fichier suivant, vous pourrez voir le film entier : [charger le film au format AVI \(1,3 Mo !\)](#)

Un des intérêts des AE et AG est que le temps de calcul ne croît pas exponentiellement avec le nombre n de variables, mais plutôt en $n \ln n$. D'autre part, ce temps de calcul est proportionnel au temps de calcul de la fonction d'adaptation, donc du modèle utilisé, et à la taille de la population.

Réglage des paramètres de l'AG

Quelle doit être la taille de cette population ? Une population trop petite évoluera probablement vers un optimum local peu intéressant. Une population trop grande sera inutile car le temps de convergence sera excessif. La taille de la population doit être choisie de façon à réaliser un bon compromis entre temps de calcul et qualité du résultat.

Pour le problème qui nous intéressait et avec les moyens de calcul dont nous disposions, nous avons pu constater qu'une population de 100 individus constituait un bon compromis. Nous pouvions ainsi calculer 100 à 200 générations en une nuit sur un Pentium Pro 200 MHz.

Mais il faut être conscient que cette taille de population dépend de la puissance de calcul dont on dispose, des méthodes utilisées (sélection, opérateurs génétiques...), du nombre de variables considérées et de la fonction d'adaptation. Si la fonction à optimiser comporte peu d'optima locaux et un optimum global net, la population nécessaire sera plus petite que dans le cas d'une fonction beaucoup plus compliquée comportant de nombreux optima locaux.

Nous touchons là au délicat problème du réglage des paramètres de l'algorithme. Celui-ci doit être optimisé pour chaque type de problème traité, ce qui constitue une part importante du travail de l'utilisateur. Les caractéristiques de l'algorithme doivent être adaptées à la topologie du paysage dessiné par la fonction (*fitness landscape*). L'ensemble problème-méthodes-paramètres constitue donc un tout. En témoigne certaines études où les paramètres d'un Algorithme Génétique sont réglés et optimisés par un autre Algorithme Génétique [Goldberg, 1994]. Dans la pratique, les méthodes et paramètres des AG sont tout d'abord réglés approximativement par tâtonnement avec des fonctions de n variables couramment utilisées pour tester les algorithmes d'optimisation. Le temps de calcul de ces fonctions étant minime, on peut ainsi régler rapidement les paramètres. Nous avons ainsi utilisé une fonction sphérique, une fonction de Fletcher-Powell et une fonction fractale [Bäck, 1996]. Nous avons ensuite réglé plus finement les paramètres en fonction des problèmes traités.



Exemple d'optimisation par AG d'un commutateur optique

Nous présentons l'optimisation de *commutateurs à réflexion interne totale* (ou *commutateurs TIR*), étude effectuée dans le cadre d'une collaboration entre l'[IEMN](#) et Dassault Electronique. Un *commutateur* directionnel permet d'aiguiller la lumière dans un guide ou un autre en faisant varier l'indice de réfraction au niveau de l'embranchement. Ces composants sont destinés à la réalisation de *matrices de commutation* pour la génération de retards temporels, afin de commander des antennes actives. Ils sont constitués de deux guides sécants et d'une électrode surplombant l'intersection (figure 1). Cette électrode permet d'injecter des porteurs dans la structure (figure 2). Une injection forte de courant peut entraîner une diminution d'indice de réfraction allant jusqu'à plusieurs pour-cent, suffisamment importante pour induire une réflexion totale. Pour plus de détails sur ces composants, le lecteur se reportera à [Cayrefourcq, thèse 1998].



Figure 1 : structure et principe de fonctionnement d'un commutateur TIR, dans l'état passant (à gauche) et l'état commutant (à droite).

Les commutateurs optiques sont caractérisés par le courant I_s nécessaire pour commuter, les pertes qui sont la différence en dB des puissances d'entrée et de sortie, et la *diaphonie* qui est la différence en dB des puissances des deux sorties. Les pertes et la diaphonie sont bien sûr données pour l'état passant // et l'état commutant X.

Variables d'optimisation

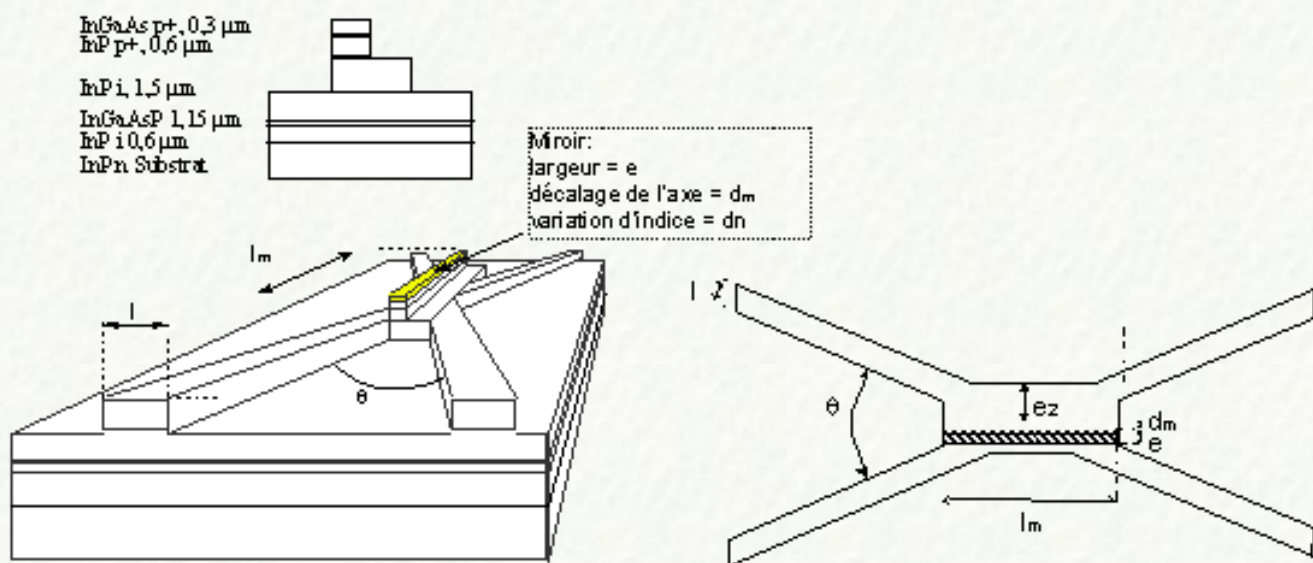


Figure 2 : épitaxie, schémas du commutateur TIR et principales variables d'optimisation.

Les sept variables que nous avons prises en compte dans notre optimisation sont (figure 2) : l'angle entre les deux guides θ , la largeur des guides l , la largeur du miroir e , la longueur du miroir l_m , le décalage de l'axe du miroir d_m , la largeur de la zone intermédiaire e_2 , et la consommation du miroir I_s (fonction de la variation d'indice dn). Ces variables sont soit intrinsèques (paramètres géométriques), soit extrinsèques (consommation). Pour indication, nous donnons les caractéristiques et performances mesurées des composants réalisés avant la mise au

point de notre algorithme :

θ (°)	l (μm)	e (μm)	l_m (μm)	d_m (μm)	e_2 (μm)	I_s (mA)	Pertes (dB)		Diaphonie (dB)	
							//	X	//	X
4	4	2	150	0	2	175	0,3	0,4	-20	-15

Tableau 1 : caractéristiques et performances mesurées du commutateur TIR de départ.

Les caractéristiques de ce composant avaient été optimisées " à la main ", c'est-à-dire par la méthode d'essai et erreur, en utilisant une BPM-2D (modèle physique de la propagation de la lumière dans les guides d'onde) [Cayrefourcq, 1998a]. Ce travail avait été relativement laborieux et limité à un petit nombre d'essais. Néanmoins, il nous laissait déjà présager l'existence de fortes interactions entre certaines variables. Ainsi, l'angle optimum entre les guides semble fortement dépendant de la variation d'indice dans le miroir, ce qui influe sur les paramètres géométriques de ce dernier.

Espace de recherche

Une fois choisies les variables d'optimisation, il faut définir, c'est-à-dire limiter, l'espace de recherche correspondant à l'aide de considérations physiques, technologiques et numériques. Nous savons par exemple que l'angle θ ne doit pas être trop petit pour éviter tout couplage entre guides adjacents. Il ne doit pas non plus être trop grand car la variation d'indice nécessaire et donc la consommation serait trop importante. De plus, le modèle physique utilisée suppose que la lumière se propage dans une direction proche de l'axe de simulation (approximation paraxiale). Quant aux dimensions du miroir, elles doivent répondre à un compromis entre consommation, efficacité et facilité de fabrication. A l'aide de telles considérations, nous avons défini l'espace de recherche suivant :

	θ (°)	l (μm)	e (μm)	l_m (μm)	d_m (μm)	e_2 (μm)
Limite inférieure	1	1	0,5	50	0	0
Limite supérieure	12	6	4	500	3	6

Tableau 2 : espace de recherche pour l'optimisation du commutateur TIR.

Fonctions objectif

Nos objectifs sont des pertes optiques faibles, une diaphonie faible et une consommation faible. Nous définissons donc les fonctions objectif f_i suivantes, calculées pour chaque composant à partir de deux simulations (états // et X) :

$$f_{pertes//} = \frac{E_{sortie//}}{E_{entre}}$$
$$f_{diaphonie//} = 1 - \frac{E_{sortieX}}{E_{entre}}$$
$$f_{pertesX} = \frac{E_{sortieX}}{E_{entre}}$$
$$f_{diaphonieX} = 1 - \frac{E_{sortie//}}{E_{entre}}$$

(1)

où E_x est l'énergie dans le guide x .

Ces fonctions objectif augmentent linéairement de 0 jusque 1 quand la diaphonie et les pertes décroissent. Pour tenir compte de la consommation, nous avons étudié la structure du commutateur à l'aide d'un modèle physique

électrique (Modèle Energie). Ce qui nous a permis d'aboutir à une formule nous donnant le courant I_s consommé par un commutateur en fonction des paramètres d'optimisation. La fonction objectif associée à la consommation du commutateur s'écrit alors simplement :

$$f_{conso} = 1 - \frac{I_s}{I_{s\max}} \tag{2}$$

où $I_{s\max}$ correspond au cas le plus défavorable possible dans l'espace de recherche étudié. Cette fonction augmente linéairement de 0 vers 1 quand la consommation diminue.

Fonction de fitness

Après avoir défini nos fonctions objectif, nous avons définie la fonction d'adaptation par une simple somme pondérée des fonctions objectif :

$$f = \sum_i \alpha_i f_i \tag{3}$$

Les coefficients α_i permettent de privilégier tel ou tel objectif suivant les applications visées. On peut ainsi orienter l'évolution vers un composant qui consomme peu mais dont les pertes optiques sont importantes ou vers un composant dont les pertes seront très faibles mais qui nécessitera une alimentation conséquente.

Résultats

Le temps de calcul du modèle physique (BPM-2D) pour notre composant est de quelques secondes sur un Pentium III actuel. Une population de 100 individus nous a alors paru un bon compromis entre temps de calcul et convergence, nous permettant ainsi de calculer entre 100 et 200 générations en une nuit.

Optimisation sans tenir compte de la consommation

Dans un premier temps, nous avons optimisé le composant uniquement en tenant compte des pertes et de la diaphonie, c'est-à-dire que nous n'avons pas pris en compte la consommation : nous avons ainsi fixé la variation d'indice dn à 0,02 correspondant à une densité de courant d'environ 2.10^7 mA.cm⁻². Etant donné que nous accordons autant d'importance aux pertes qu'à la diaphonie et ce dans les deux états du commutateur, chaque fonction objectif a le même poids. La fonction d'adaptation est donc :

$$f = \frac{f_{pertesII} + f_{diaphonieI} + f_{pertesX} + f_{diaphonieX}}{4} \tag{4}$$

Après convergence, le meilleur composant obtenu était le suivant :

θ (°)	l (µm)	e (µm)	l _m (µm)	d _m (µm)	e ₂ (µm)	I _s (mA)	Pertes (dB)		Diaphonie (dB)	
							//	X	//	X
5,3	5,53	3,98	281	0,12	1,94	290	0,02	0,1	-19,2	-31

Tableau 3 : composant optimisé sans tenir compte de la consommation.

Par rapport au composant de départ (tableau 1), on observe une grande amélioration : les pertes deviennent négligeables et la diaphonie s'améliore fortement dans l'état commutant. Cependant, la consommation est excessive car la surface de l'électrode a été quasiment multipliée par quatre.

Prise en compte de la consommation

Nous avons donc ensuite introduit la consommation dans notre fonction d'adaptation :

$$f = \frac{f_{pertesII} + f_{diaphonieI} + f_{pertesX} + f_{diaphonieX} + \alpha \cdot f_{conso}}{4 + \alpha} \tag{5}$$

où le coefficient α nous permet de pondérer l'importance de la consommation.

L'espace de recherche passe de 6 à 7 dimensions, puisqu'il faut introduire la variation d'indice de réfraction, liée à la consommation :

	θ (°)	l (μm)	e (μm)	l_m (μm)	d_m (μm)	e_2 (μm)	dn
Limite inférieure	1	1	0,5	50	0	0	0,001
Limite supérieure	12	6	4	500	3	6	0,02

Tableau 4 : espace de recherche avec prise en compte de la consommation du commutateur TIR.

De nombreux lancements de l'Algorithme Génétique avec différents coefficients α nous ont amené à penser que la réduction de la consommation ne passe pas nécessairement par la réduction de la surface du miroir. Quand la surface du miroir augmente, il semble que la variation d'indice nécessaire à obtenir la commutation diminue. Ceci s'explique par le fait que si la consommation est proportionnelle à la surface du miroir, elle varie suivant une loi exponentielle en fonction du changement d'indice. Nous présentons ci-dessous les meilleurs composants obtenus pour différentes pondérations α :

α	θ (°)	l (μm)	e (μm)	l_m (μm)	d_m (μm)	e_2 (μm)	dn (μm)	I_s (mA)	Pertes (dB)		Diaphonie (dB)	
									//	X	//	X
0,2	3,6	5,57	3,06	221	0,14	2,93	0,015	81,15	0,24	0,08	-15	-20,1
0,5	3,9	5,59	2,88	253	0,15	4,2	0,015	65,6	0,20	0,17	-17,3	-18,5
1	3,2	5,58	3,47	271	0,15	0,005	0,010	47,1	0,23	0,19	-16,3	-16,85

Tableau 5 : meilleurs résultats obtenus par l'AG, selon l'importance accordée à la consommation.

Comme on pouvait s'y attendre, les propriétés du composant sont globalement d'autant meilleures que la consommation est plus importante. Le choix du composant à fabriquer se fait en fonction de l'application, selon que l'utilisateur préfère un composant de qualité moyenne et peu gourmand en énergie ou un composant ayant de très bonnes propriétés mais ayant une consommation plus importante.

Bibliographie relative aux commutateurs optiques

- Cayrefourcq I., *Conception et fabrication de matrices de commutation optiques en vue de la réalisation de modules de synthèse de retards temporels*, Thèse de l'Université des Sciences et Technologies de Lille, 1998.
- Cayrefourcq I., Schaller M., Fourdin C., Vilcot J.P., Harari J. et Decoster D., " Optical switch design for true time delay array antenna ", *IEEE Proceedings on Optoelectronics part J*, vol. 145, n°1, p. 77-82, 1998a.
- Cayrefourcq I., Schaller M., Vilcot J.P., Harari J., Gouy J.P. et Decoster D., " Low Power Consumption 1x4 Cascade Switch for Microwave Applications ", *Microwave and Optical Technology Letters*, juillet

1998b.

- Hunsperger R.G., *Integrated Optics : theory and technology*. New-York : Springer-Verlag, 1995.
 - Moosburger R., Kostrzewa C., Fischbeck G. et Petermann K., " Shaping the Digital Optical Switch Using Evolution Strategies and BPM ", *IEEE Photonics Technology Letters*, vol. 9, n°11, p. 1484-1486, novembre 1997.
-



Dernière mise à jour : 28 mars 2000.
Auteur : Vincent MAGNIN

AG et voyageur de commerce (applet Java)

Voici un problème d'optimisation classique : un voyageur de commerce doit passer dans chaque ville d'une contrée, et ce une seule fois. Comment faire pour que le trajet soit le plus court possible ? Une méthode d'énumération exhaustive est exclue : s'il y a N villes, pour la seconde étape de son périple notre voyageur a $N-1$ possibilités, pour la troisième $N-2$, etc. Le nombre de combinaisons est donc $(N-1)!$. Pour seulement 40 villes, cela fait à peu près $2e^{46}$ solutions à tester. En supposant que l'on dispose d'un ordinateur permettant de tester un milliard de solutions par seconde, cela nous prendrait plus de $1e^{19}$ fois l'âge de l'univers pour tester toutes les solutions ! Alors qu'un Algorithme Génétique trouvera une très bonne solution en testant seulement quelques milliers de solutions. Ce simple exemple illustre bien la puissance du concept d'évolution naturelle.

L'applet Java ci-dessous illustrant ce problème a été réalisée par deux étudiants de l'EUDIL, alors en 2^{ème} année IMA : Raphaël RAIMBAULT et Laurent DUVAL. A vous de jouer avec ! Grâce à un internaute généreux (J.-P. Neuser), cette applet fonctionne désormais avec Netscape et Internet Explorer.

Mode d'emploi succinct :

Paramètres de l'algorithme génétique :

- Le nombre *d'individus* est le nombre de le voyageur qui "recherchent" le plus court chemin.
- Le taux de reproduction est le nombre de nouveaux individus pour 100 dans l'ancienne génération créés par croisement des meilleurs individus.
- La probabilité de mutation d'un individu : nombre compris entre 0 et 1. Il reflète la probabilité d'un individu au sein de la population de pouvoir muter (0 = pas de mutation d'individu).
- La probabilité de mutation d'un gène : nombre compris entre 0 et 1. Il reflète la probabilité d'un gène au sein d'un individu de pouvoir muter (0 = pas de mutation de gène).

Valeurs liées au résultat affiché :

- Valeur d'adaptation maximum, est la valeur d'adaptation du meilleur individu de la population. La Valeur d'adaptation n'est .ici, pas très représentative
- Distance correspondante, est la distance parcourue par le meilleur voyageur de la population. C'est surtout cette valeur que l'on regarde évoluer.
- Numéro de génération, est le nombre de générations déjà calculées. Lorsque la méthode déterministe est sélectionnée, ce nombre est un équivalent par rapport au nombre d'individus testés.
- Nombre d'individus testés, est le nombre d'individus qui ont été évalués depuis le début du calcul.
- La courbe d'évolution n'est utilisée que par l'algorithme génétique. Elle présente l'évolution sur les générations de la valeur d'adaptation du meilleur individu (en rouge) et de la moyenne des valeurs d'adaptation de la population (en bleu). Ceci permet de se faire une idée sur la rapidité

de la convergence de l'algorithme, de la diversité de la population

- Choix de la méthode de calcul, cela permet de résoudre un problème donné par algorithme génétique ou par calcul de toutes les solutions possibles.
- Le meilleur parcours courant est une représentation du placement géographique des villes et du parcours effectué par le meilleur voyageur.



Dernière mise à jour : 14 juin 2001.

Auteurs : Vincent MAGNIN, Raphaël RAIMBAULT et Laurent DUVAL

Conclusion

La modélisation est un outil puissant qui permet en particulier d'optimiser un dispositif avant de le fabriquer. Les algorithmes d'optimisation permettent de s'affranchir en grande partie de la méthode d'essai et erreur. Parmi ceux-ci nous avons abordé d'abord les méthodes Monte Carlo avant de passer aux Algorithmes Evolutionnaires et aux Algorithmes Génétiques (AG).

Les méthodes Monte Carlo

Dans le cadre de l'optimisation à paramètres multiples, les méthodes Monte Carlo sont intéressantes si le nombre de paramètres reste inférieur à trois ou quatre. Elles ont alors l'avantage de la simplicité. De plus, elles permettent d'avoir un bon aperçu de l'espace de recherche : les nuages de points que l'on peut en tirer sont souvent riches en information.

Si le nombre de paramètres est plus important, les méthodes Monte Carlo peuvent être utilisées dans un espace de recherche restreint, par exemple pour évaluer la sensibilité d'un dispositif aux incertitudes technologiques.

Les Algorithmes Génétiques

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. Parmi ces algorithmes, on distingue les Algorithmes Génétiques qui utilisent un codage des paramètres sous forme de chaîne binaire, par analogie avec l'ADN.

Nous avons présenté quelques techniques de base pour réaliser un AG simple mais efficace, très robuste et général. Mais n'oublions pas qu'il existe un compromis généralité / efficacité. Pour obtenir des performances optimales, il peut être souhaitable d'adapter l'algorithme au problème traité, en introduisant des méthodes spécifiques, et si possible des connaissances sur le problème [Davis, 1991]. On peut également réaliser des algorithmes hybrides mêlant différentes méthodes (par exemple AG et méthode des gradients).

Par rapport aux méthodes habituelles que l'on peut qualifier d'*analytiques*, les algorithmes évolutionnaires peuvent être qualifiés de *synthétiques*. Ils peuvent parfois synthétiser des solutions nouvelles et originales à des problèmes connus car ils expérimentent sans idées préconçues, si ce n'est les paramètres et l'espace de recherche qu'on leur impose. Ils permettent donc d'insuffler un peu de créativité dans l'ordinateur, cette machine qui en est si cruellement dépourvue !

Un AG peut être couplé avec toutes sortes de simulations, pour peu que le processus puissent être totalement automatisé et que le temps de calcul ne soit pas prohibitif. Ces algorithmes sont donc appelés à se répandre de plus en plus dans l'industrie et la recherche grâce à la montée en puissance exponentielle des PC. De plus, les AG étant par nature massivement parallèles, ils bénéficieront du développement des systèmes à architectures parallèles. Les nouvelles versions du FORTRAN (HPF) permettront l'implémentation sur ces systèmes.

Les AG peuvent être utilisés à tous les stades : recherche, développement, production. En effet, il est

tout aussi nécessaire de concevoir un dispositif ayant des caractéristiques optimales, qu'un dispositif ayant une faisabilité optimale, un coût optimal... Ils constituent un domaine de recherche très actif, d'une part de par leur intérêt propre, d'autre part parce qu'ils rejoignent les enjeux industriel et économique actuels.



Dernière mise à jour : 13 décembre 1999.

Auteur : Vincent MAGNIN

Perspectives

Que nous réserve l'avenir ? Pour le savoir, je vous conseille de jeter un coup d'oeil au site web de l'[Université Brandeis](#). Vous y découvrirez comment les AG sont utilisés pour faire évoluer des créatures artificielles capables de se mouvoir de façon parfois étrange... Vous pouvez également vous procurer le "Pour la Science" de juin 2001 qui comporte un article sur d'autres applications à la robotique.

Que nous réserve l'avenir ? N'est-ce pas là une des merveilles de l'évolution que l'avenir soit toujours incertain ?



Dernière mise à jour : 14 juin 2001.
Auteur : Vincent MAGNIN

Bibliographie

Les références sont classées alphabétiquement selon le nom du premier auteur.

Pour les livres disponibles à la [Bibliothèque Universitaire de Lille 1](#), la côte est indiquée en marron.

- Bäck T. et Hoffmeister F., " Global optimization by means of evolutionary algorithms ", in A.N. Antamoshkin, editor, *Random Search as a Method for Adaptation and Optimization of Complex Systems*, p. 17-21, Divnogorsk, ex-URSS, mars 1991.
- Bäck T., " Self-Adaptation in Genetic Algorithms ", *Proceedings of the 1st European Conference on Artificial Life*, p. 263-271. Cambridge, MA : MIT Press, 1992.
- Bäck T., *Evolutionary Algorithms in Theory and Practice*. Oxford: Oxford University Press, 1996. **BU : 006.3 BAC**
- Bäck T., Hammel U. et Schwefel H.P., " Evolutionary Computation: Comments on the History and Current State ", *IEEE Transactions on Evolutionary Computation*, vol. 1, n°1, p. 3-17, avril 1997.
- Balachandran M., *Knowledge-Based Optimum Design, Topics in Engineering Vol. 10*. Southampton: Computational Mechanics Publications, 1993. **BU : 006.3 BAL**
- Beasley D., Bull D.R. et Martin R.R., " An Overview of Genetic Algorithms : Part 1, Fundamentals ", *University Computing*, vol. 15, n°2, p. 58-59, 1993a.
- Beasley D., Bull D.R. et Martin R.R., " An Overview of Genetic Algorithms : Part 2, Research Topics ", *University Computing*, vol. 15, n°4, p. 170-181, 1993b.
- Ciarlet P.G., *Introduction à l'analyse numérique matricielle et à l'optimisation*. Paris : Masson, 1990.
- Darwin C., *The Origin of Species*, 1859.
- Davis L., Ed., *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991. **BU : 511.6 HAN**
- Dessales J.-L., *L'ordinateur génétique*. Paris: Hermès, 1996.
- Fogel D.B., " Evolutionary Computation: A New Transactions ", *IEEE Transactions on Evolutionary Computation*, vol. 1, n°1, p. 1, avril 1997.
- Fonseca C.M. et Fleming P.J., " Genetic Algorithms for Multiobjective Optimization : Formulation, Discussion and Generalization ", in *Genetic Algorithms : Proceedings of the Fifth International Conference*, S. Forrest, editeur, San Mateo, CA : Morgan Kaufmann, juillet 1993.
- Fonseca C.M. et Fleming P.J., " An Overview of Evolutionary Algorithms in Multiobjective Optimization ", *Evolutionary Computation*, vol. 3, n°1, p. 1-16, 1995.
- Goldberg D.E., *Algorithmes génétiques, exploration, optimisation et apprentissage automatique*. Paris: Addison-Wesley, 1994. **BU : 006.3 GOL**
- Heudin J-C., *L'évolution au bord du chaos*. Paris: Hermès, 1998. **BU : 539.1 HEU**
- Menozzi R. et Piazzzi A., " HEMT and HBT Small-Signal Model Optimization Using a Genetic Algorithm ", *EDMO'97*, Londres, p.13-18, 24-25 novembre 1997.
- Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin: Springer-Verlag, seconde édition, 1994. **BU : 005.1 MIC**
- Monod J., *Le hasard et la nécessité*. Editions du Seuil, 1970.
- Moosburger R., Kostrzewa C., Fischbeck G. et Petermann K., " Shaping the Digital Optical

Switch Using Evolution Strategies and BPM ", *IEEE Photonics Technology Letters*, vol. 9, n°11, p. 1484-1486, novembre 1997.

- Nougier J.P., *Méthodes de calcul numérique*. Paris: Masson, 1987. **BU : 519.4 NOU**
- Popper K., *Toute vie est résolution de problèmes*. Actes Sud, 1997.
- Reineix A., Eclercy D. et Jecko B., " FDTD/genetic algorithm coupling for antennas optimization ", *Annales de Télécommunications*, vol. 52, n°9-10, 1997.
- Renders J.M., *Algorithmes génétiques et réseaux de neurones*. Paris: Hermès, 1995. **BU : 006.3 REN**
- Whitley D., " A Genetic Algorithm Tutorial ", *Technical Report CS-93-103*, Colorado State University, Department of Computer Science, 1993.
- Wills C., *La sagesse des gènes, nouvelles perspectives sur l'évolution*. Flammarion, 1991.
- "L'évolution en éprouvette", *Science & Vie*, n°1006, p. 124-127, juillet 2001.
- Meyer J.-A. et Guillot A., "La robotique évolutionniste", *Pour la Science*, n°284, p. 70-77, juin 2001.



Dernière mise à jour : 18 novembre, 2001
Auteur : Vincent MAGNIN

Références Internet

Vous cherchez un code tout fait (C, Fortran, Java, Lisp...) ? Allez faire un tour sur le site suivant : <http://www.aic.nrl.navy.mil/galist/src/>

De nombreux articles nous ayant servi pour la réalisation de notre algorithme génétique sont disponibles sur Internet en format PostScript, en particulier sur les sites suivants :

- <http://www.aic.nrl.navy.mil:80/galist>: archives de la Navy sur les AG.
- <http://GAL4.GE.UIUC.EDU/illegal.home.html>: Illinois Genetic Algorithms Laboratory, dirigé par D.E. Goldberg, un des grands spécialistes des AG.
- <http://www.natural-selection.com/index.html>: Natural Selection Inc., où l'on trouve les publications de D.B. Fogel.
- <http://garage.cps.msu.edu/>: Genetic Algorithms Research and Applications Group (GARAGe) .
- <http://www.shef.ac.uk/~gaipp/mogas.html>: Evolutionary Computation Research, Multiobjective Genetic Algorithms (Carlos Fonseca).

Les sites suivants nous ont été particulièrement utiles pour l'acquisition des connaissances de base sur les AG et comme points de départ vers d'autres sites plus spécifiques :

- <http://surf.de.uu.net/encore/> : " The Hitch-Hiker's Guide to Evolutionary Computation: A list of Frequently Asked Questions (FAQ) " On trouvera dans ce FAQ les réponses à beaucoup des questions que l'on peut se poser sur les AG.
- [news://news.univ-lille1.fr/comp.ai.genetic](http://news.univ-lille1.fr/comp.ai.genetic) (groupe de nouvelles) : l'internaute y participera avec profit. Il trouvera quantité d'informations et pourra poser des questions aux spécialistes des AG.

Pour les développements récents et à venir, citons les sites suivants :

- <http://www.evonet.polytechnique.fr/> : the network of excellence in evolutionary computing.
- <http://frontier.ii.uib.no/Frontier.html> : informations sur le projet européen FRONTIER visant à développer des outils d'optimisation multi-objectifs permettant d'assurer la compétitivité des industries européennes dans le futur.
- <http://w3.ualg.pt/lists/emo-list/> : archives de la *mailing-list* EMO (Evolutionary Multiobjective Optimization List), consacrée aux discussions sur l'optimisation multi-objectifs par algorithme évolutionnaire.



Dernière mise à jour : 25 décembre 2000.
Auteur : Vincent MAGNIN