Les Algorithmes d'évolution : vers des stratégies avec opérateurs adaptatifs

L.Gacôgne * **

* LIP6 - Université Paris VI 4 place Jussieu 75252 Paris 5°
tel : 01 44 27 88 07 fax : 01 44 27 70 00 mail : Louis.Gacogne@lip6.fr

** Institut d'Informatique d'Entreprise (CNAM) 18 allée J.Rostand 91025 Evry

Résumé
Après un bref survol de l'historique des algorithmes évolutionnaires et des principales idées mises en oeuvre, nous présentons des stratégies diverses dont l'ambition est souvent de s'affranchir du problème du réglage des paramètres initiaux. Nous nous concentrons sur la comparaison de stratégies ayant comme point commun de se baser sur des idées naturelles assez simples. Celles-ci diffèrent principalement sur le choix et le mode d'application des opérateurs génétiques, ainsi que sur la procédure de renouvellement des générations. Tout en sachant qu'il est illlusoire d'obtenir un jour une solution universelle quant à ces questions, nous présentons une technique où les opérateurs sont eux mêmes évalués de façon à être appliqués à la mesure de leur performance. Nous observons sur l'optimisation de fonctions classiques une amélioration grâce à cette méthode par rapport aux stratégies d'évolution, ainsi que grâce à un algorithme très simple de remplacement également testée.

Abstract
After an overview of the evolutionary algorithms field and their main ideas, we present some strategies that often intend to get off the problem of the initial parameters tuning. We focuse our attention about the comparizon of quite simple strategies, specially about the choice of genetic operators, the way to apply them and finally how each generation is built from the previous one. Knowing that it is not possible to reach an universal heuristic able to choose the genetic operators and to manage them, we present a method where the genetic operators themselves are evaluated according to their performance. The amelioration observed on some classical functions, with this method shows a future way to combine it thanks to a very simple steady state algorithm which has also given a good behaviour.

Mots-clés algorithmes évolutionnaires - algorithmes génétiques - stratégies d'évolution

Keywords evolutionary algorithm - genetic algorithm - evolution strategy

INTRODUCTION

Du point de vue le plus général, les méthodes de résolution de problèmes pourraient être classées suivant quatre grandes classes comme :

méthodes combinatoires , dans le cas discret où un nombre fini et suffisamment petit, (au regard du temps de parcours), d'éventuelles solutions peut être examiné, on parle alors d'algorithmes gloutons car dévoreurs de temps de calcul,

méthodes de construction avec exploration d'arborescence (ce sont les méthodes de recherche avec retour en arrière bien connues en intelligence artificielle),

méthodes locales (les méthodes mathématiques classiques comme la descente de gradient, mais aussi l'heuristique du recuit simulé) et enfin,

méthodes évolutives s'inspirant de l'évolution des espèces vivantes. Ce sont de ces méthodes dont il sera question ici.

Naturellement, il existe des méthodes hybrides, et des «méta-heuristique» indiquant, par exemple, d'appliquer une méthode locale sur certains points d'une population et marier ainsi différentes heuristiques. Ce sont d'ailleurs ces méthodes hybrides qui donnent de bons résultats sur des problèmes comme celui du voyageur de commerce [Talbi 98].

Les méthodes d'évolution artificielle, sont avant tout réservées aux problèmes d'optimisation pour lesquels les autres méthodes ont échoué, soit que l'espace de recherche est fini mais trop grand, soit que la fonction à optimiser est mal définie et ne possède aucune «bonne propriété» mathématique, en effet, ces méthodes sont gourmandes en temps de calcul et n'ont pu faire l'objet d'expérimentation qu'avec les progrès en matériel informatique.

Cependant, ces méthodes ne sont pas réservées à l'optimisation où grâce à une «population» variée de «solutions candidates», on peut trouver une solution «acceptable», mais elles sont aussi liées à l'apprentissage en général et peuvent enfin être appliquées à des systèmes multi-agents où le but est, cette fois, de trouver une population «acceptable», dans le but de résoudre collectivement un problème.

La question essentielle des algorithmes évolutionnaires est d'arriver à un compromis entre l'exploration de l'espace de recherche et l'exploitation des régions déjà explorées. C'est pourquoi, beaucoup d'idées implémentées, cherchent à éviter une trop grande homogénéité de la population. La population peut être de taille variable. On peut vouloir modifier artificiellement la fonction à optimiser, les opérateurs peuvent être notés, ils peuvent évoluer en cours (mutation non uniforme), on peut créer des sous-populations mises en compétition et surtout créer un partage de la population en «niches» correspondant à des minima locaux.

Nous examinerons dans cet article les grandes orientations de ce domaine de recherche, en insistant, parmi toutes les idées qui se sont fait connaître, sur celles qui tendent le plus à se passer de paramètres initiaux, c'est à dire s'emploient à suivre des principes généraux d'évolution, qui eux-même pourraient évoluer. Par ailleurs, sur un problème donné, un algorithme ne peut être plus performant en moyenne que n'importe quel autre, tant que des connaissances particulières à ce problème ne lui soit donné [Wolpert, McReady 95]. Il est donc illusoire de vouloir résoudre sans connaissances, mais on peut espérer progresser quelque peu dans le sens de cette ambition par des heuristiques se modifiant grâce à des méta-heuristiques les évaluant et les combinant.

Nous choisissons ensuite certaines heuristiques, en vue de les comparer sur des problèmes numériques classiques et nous conservons enfin les trois meilleures pour les comparer sur des problèmes plus symboliques.

I ALGORITHMES GÉNÉTIQUES ET STRATÉGIES D'ÉVOLUTION

Les algorithmes génétiques [Bagley 67], [Holland 75], [Goldberg 82, 89] sont inspirés par les lois de l'évolution en biologie où par croisements, la population d'une espèce se renouvelle et où des mutations accidentelles de très faibles probabilités peuvent modifier dans un sens ou dans l'autre les aptitudes d'un individu et, par suite, favoriser ou non la possibilité qu'il aura de se reproduire. Un individu muté dans le sens d'une plus grande performance (suivant certains critères, ne serait-ce que la longévité) aura alors plus de chance de se reproduire et donc de transmettre à sa descendance son nouveau code.

C'est la fameuse théorie darwinienne de la sélection naturelle [Monod 70], [Dessalles 96]. Cette théorie a donc inspiré des méthodes stochastiques appliquées à la résolution des problèmes, non seulement mal aisés à résoudre mais à définir. C'est avec un certain succès que des problèmes mal posés se voient offrir des solutions souvent inatendues par une simulation d'évolution :
Pour un problème quelconque d'optimisation, les solutions vont d'abord être codées le plus souvent sous forme de chaînes de caractères appelées «chromosomes», les caractères étant les «gènes». L'algorithme consiste alors à reproduire par étapes une «population» P de chromosomes. Il ne s'agira pour l'instant que d'«haploïdie» c'est à dire que tout individu dans l'espace de recherche est représenté par une suite de gènes appelée le chromosome et non par des paires de plusieurs chromosomes comme c'est le cas pour beaucoup d'espèces naturelles.

Au départ on prendra donc des individus-chromosomes plus ou moins arbitraires, générés aléatoirement ou réalisant déjà un semblant de solution intuitive, ou bien de réelles solutions mais non optimales.
De génération en génération, la population doit s'améliorer suivant une fonction d'évaluation dictée par le problème, on peut alors classer la population suivant cette fonction pour permettre aux meilleurs chromosomes de se reproduire.

L'illustration est assez simple à faire sur le problème de la minimisation d'une fonction numérique définie sur l'intervalle [a, b]. Les algorithmes génétiques et plus généralement tous les algorithmes évolutionnaires partent d'une population de points (10 sur la figure ci-dessous), il est clair que moyennant un temps de calcul multiplié, les chances d'obtenir plusieurs minima et le minimum global en sont augmentés.

Remarque si une fonction de E vers R a n minimums dans E, et si nous partons de k points (avec une distribution aléatoire uniforme), la probabilité d'atteindre un nombre X = p de ces n minimums est donnée par :

$$pr(X = p) = \sum_{i=0}^{p} (-1)^{p+i} C_p^i \left(\frac{i}{n}\right)^k$$

Si nous supposons l'equiprobabilité pour les n bassins d'attraction, (ce qui n'est bien sûr que rarement le cas) alors la probabilité est le rapport entre le nombre d'applications surjectives d'un ensemble de cardinalité k dans un ensemble de cardinal p, et le nombre d'applications d'un ensemble de cardinalité k dans un ensemble de cardinal n pour $p \leq k$ et $p \leq n$. Par exemple si n = p = 3 et k = 9, nous avons déjà une probabilité 0.922 seulement avec p = 3 c'est pourquoi dans le "stratégies d'évolution sans reproduction, il est possible de ne considérer souvent avec des populations de petite cardinalité comme 10 ou 20.

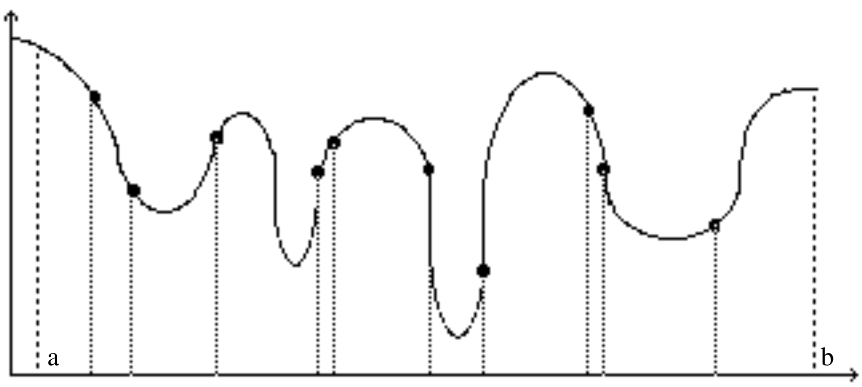

Figure 1 Population aléatoire dans un intervalle [a, b].

Mais ces stratégies ne se contentent pas de poursuivre des itérations séparées dans plusieurs régions de «l'espace de recherche» en recréant des «niches écologiques» si l'on poursuit l'image de l'évolution naturelle des espèces, elles innovent dans l'exploration de cet espace de recherche en se croisant et en brouillant les éventuelles solutions par des «opérateurs génétiques».

Toutes les méthodes de descente du gradient (adaptées aux fonctions différentiables et convexes), ou bien les méthodes stochastiques telle que celle du recuit simulé consistent à itérer l'évaluation de la fonction sur une suite de points «voisins» en partant d'un point initial arbitraire. Si la fonction n'est pas convexe et présente plusieurs minima, on ne peut garantir la convergence vers le minimum global même si une certaine remontée est tolérée suivant une probabilité à définir comme c'est le cas dans le recuit simulé.
Avant d'aborder les méthodes évolutives, rappelons cette heuristique, la plus connue, également inspirée de la nature, dont le but est d'éviter une trop rapide convergence, ainsi que celle du «Hill-climbing».

MÉTHODE COMBINATOIRE DU "HILL-CLIMBING"

Cette méthode locale, partant d'un point choisi aléatoirement, consiste itérativement, à examiner tout le voisinage du point courant afin de se placer au meilleur d'entre eux pour l'étape suivante. Naturellement, en chaque point x on définit un voisinage fini V_x de points pour lesquels on peut calculer f, ce peut être des points voisins au sens d'une distance discrete ou bien, comme on le verra plus tard, des points définis à partir de x par des opérateurs génétiques.

1 Choisir x aléatoire dans le domaine de la fonction f à minimiser
2 $t \leftarrow 0$ $\quad f_{min} \leftarrow +\infty$
3 incr(t)
4 Evaluer f sur chacun des points voisins de x, soit x' tel que $f(x') = \min f/V_x$
5 Si $f(x') < f(x)$ alors $x_{min} \leftarrow x'$
6 $x \leftarrow x'$
7 Si $t < t_{max}$ alors aller en 3

L'inconvénient manifeste de cette méthode est que l'on ne peut pas vraiment espérer sortir d'un minimum local.

L'ALGORITHME DE NISSEN (MUTATIONS SEULEMENT)

Le principe de cet algorithme [Nissen 93, 94], [Sandalidis 98] est de maintenir un ancêtre, celui qui a pu engendrer les autres durant une "ère", un un père "provisoire". L'exploration du voisinage est réalisé par des "mutations", petites variations au sein du codage des éventuelles solutions.

1 Engendrer λ individus (par des mutations, transposition, ... mais pas de croisement).
2 Soit I_0 le meilleur d'entre eux (l'ancêtre).
3 $I_1 \leftarrow I_0$ (I_1 père provisoire) $\quad$ ère $\leftarrow 0$ $\quad t \leftarrow 0$
4 répéter
incr (t) (nouvelle génération)
si ère = $ère_{max}$ alors ère $\leftarrow 0$ (déstabilisation)
Engendrer λ individus à partir du père I_1, soit I_2 le meilleur d'entre eux
$I_1 \leftarrow I_2$ (le meilleur fils prend la place du père à chaque génération)
si I_2 meilleur que I_1 alors ère $\leftarrow 0$; $I_0 \leftarrow I_2$ (nouvel ancêtre)
sinon incrémenter (ère)
jusqu'à $t = t_{max}$

Les ères sont donc constituées par des suites de générations où il n'y a pas de progrès, et où l'ancêtre détient la meilleure solution. La convergence n'est pas monotone puisqu'il suffit d'une amélioration pour modifier l'ancêtre. Cependant si λ est suffisamment grand, au risque de tomber plusieurs fois dans le même bassin d'attraction, on passe d'un minimum local à un autre. En fait, c'est l'éventail des possibilités d'exploration grâce aux λ enfants qui est le trait le plus important. La déstabilisation est là pour éviter la convergence prématurée, elle consiste à changer d'ère chaque fois qu'un fils est meilleur que son père ou que le nombre de génération est jugé suffisant, en ce dernier cas, cela revient à changer de point initial.

MÉTHODE STOCHASTIQUE D'OPTIMISATION, LE RECUIT SIMULÉ

Cette méthode [Kirkpartrick 83], [Aarts Korst 89], [Reeves 93] consiste à obtenir par itérations successives le minimum absolu d'une fonction, elle est inspiré d'une technique de refroidissement consistant à accepter dans certains cas une remontée de la fonction (pour ne pas descendre trop vite vers un minimum local). En thermodynamique la probabilité d'avoir une augmentation d'énergie ΔE (la fonction que l'on veut minimiser suivant la loi de Boltzmann) est $e^{-\Delta E/\theta}$. On accepte un état voisin s_1 augmentant la fonction, dans la mesure où la probabilité ci-dessus est décroissante suivant ΔE. Le paramètre de contrôle (la température) va décroître, ce qui fait que pour un même ΔE, la probabilité d'accepter une remontée diminue suivant le refroidissement.
La difficulté de cette méthode, qui par ailleurs donne de bons résultats, réside dans la détermination des paramètres. On peut montrer la convergence sous certaines conditions reliant n et la loi de décroissance de la température.

La température initiale peut être déterminée par une probabilité, par exemple 1/2, d'accepter une hausse au cours du premier palier, si m est la valeur moyenne de ces hausses, alors $\theta_0 = m / \ln(2)$. Le nombre d'itérations sur chaque palier peut être de l'ordre de la centaine, la loi de décroissance de la température est généralement prise comme une suite géométrique telle que $\theta_{n+1} = 0{,}9\ \theta_n$

Plus précisément :

1 Choisir un état initial s_0

2 Faire le «palier» consistant à répéter n fois

Chercher un voisin s_1 de s_0, on calcule $\Delta = f(s_1) - f(s_0)$

Si $\Delta < 0$ alors $s_0 \leftarrow s_1$

Si $\Delta > 0$ alors on accepte la même affectation de s_0 avec une probabilité $e^{-\Delta/\theta}$

3 On abaisse la température θ et on réitère ces paliers jusqu'à un test d'arrêt défini par le problème.

MÉTHODE TABOU

On part de la même façon d'une «solution» initiale quelconque x. A chaque point courant x du domaine de définition du problème, on cherche grâce à une discrétisation du problème, le meilleur voisin (hors du point lui-même) non tabou de x (ce sera le point suivant). La liste des «tabous» est simplement constituée d'états antérieurs sur lesquels on ne souhaite pas revenir. Cette liste doit avoir une taille fixe et chaque état n'y reste donc qu'un nombre fini d'itérations. On la gère en file : chaque mise-à-jour y place un nouveau venu et en retire le plus ancien [Glover 86].

Les premières directions de recherche

La "programmation évolutionnaire" Historiquement, est apparue la «programmation évolutionnaire» [Fogel 66] utilisant une population P et sa descendance par mutations P' où une méthode de sélection permet d'établir des tournois, en vue de ne retenir que le meilleur dans une sous-population (de l'ordre du tiers de l'effectif de P) tirée au hasard. Chaque individu est muté et la sélection s'opère sur l'union des populations mère et fille, il n'y a donc que des mutations mais pas de croisement. Le classement se fait, non pas par la "fitness" dont on voudrait l'optimum, mais par un score : un individu gagne un point chaque fois qu'il est meilleur qu'un autre dans un tournoi, P est classée et tronquée suivant ce score.

Les "algorithmes génétiques" [De Jong 75], [Holland 75], [Goldberg 89] sont apparues ensuite, caractérisés par une volonté de brouiller les individus par un codage binaire. Une mutation peut alors donner un individu aussi bien voisin qu'éloigné. La démarche la plus courante consiste à croiser un individu (suivant une probabilité de l'ordre de 0,6), exercer une mutation (avec une probabilité de l'ordre de 0,1) sur les deux enfants et remplacer les parents par les enfants.

Les "stratégies d'évolution" [Rechenberg 73], [Schwefel 90] s'écartent de l'évolution naturelle en modifiant tous les individus à chaque génération (il n'y a pas de recopie d'individus identiques) et se distinguent aussi des "algorithmes génétiques" par leur désir de conserver un codage lisible.

La "programmation génétique" [Koza 92, 94], [Kinnear 96] constitue un intéressant domaine d'expérimentation où chaque chromosome est un arbre (une fonction structurée dans le langage LISP) dans le but de trouver des expressions de fonctions (les croisements sont des échanges de sous-arbres, les mutations portent uniquement sur des constantes).

Une application de la programmation génétique, est par exemple de trouver une courbe séparatrice pour des ensembles de points. On forme un ensemble de gènes à zéro place (des terminaux) qui sont la variable x et les constantes, des fonctions unaires (sin, exp ...) et des gènes à deux places (+, -, *, /), un chromosome étant un arbre bien formé (par exemple (+ (* x x) (* 3 x))). L'évaluation de ce chromosome pour un point (x, y) dira s'il se trouve au dessus ou en dessous de la fonction représentée par le chromosome. L'algorithme génétique consiste alors à partir d'une population aléatoire de tels chromosomes, et d'opérer de générations en générations des mutations : un bruit sur une constante comme ajouter 0.1, une mutation de constante en x ou l'inverse, une mutation plus générale remplaçant un sous-arbre par une expression aléatoire, le cross-over échangeant des sous-expressions entre deux chromosomes ... Dans beaucoup de problème, la performance est calculée à partir de l'effet du programme, ainsi pour la "tondeuse à gazon" sur un quadrillage torique 8*8, les terminaux sont "tondre et avancer", "left" et "progn" et la fitness est une combinaison linéaire du nombre de cases tondues et de la taille de l'arbre. Pour une fonction booléenne à 6 arguments, les terminaux sont x, y, z, u, v ,w, les fonctions or, and, not, nand, nor et la performance peut être définie par l'écart sur les 64 cas entre ce que donne l'individu et la fonction attendue.

II DIFFÉRENTES SPÉCIFICATIONS DES ALGORITHMES ÉVOLUTIONNAIRES

Les différent points de vue et techniques, qui ont suivi la première approche, pourraient se distinguer sur les sept points suivant :

a) Définition du problème : la fonction d'évaluation.

Naturellement ces algorithmes sont utilisés pour l'optimisation de fonctions non triviales (non continues, bruitées et multimodales), sans propriétés de dérivabilité connues et définies sur des domaines bien plus complexes qu'un intervalle. Ils sont utilisés là où les méthodes analytiques ont échoué, mais aussi sur les problèmes symboliques pour lesquels les méthodes de dénombrement sont trop coûteuses. Le premier pas consiste dans la définition du problème, dans toute la suite, il s'agit de minimiser une certaine fonction f («fitness»). En fait pour des problèmes d'optimisation multicritère, nous verrons plus loin qu'il est très difficile de donner une fonction d'évaluation, le seul fait de bien poser le problème, n'est en rien trivial.

b) Codage des solutions

Les algorithmes génétiques ont débuté avec un codage binaire des éventuelles solutions de façon à ce qu'il y ait une grande possibilité d'exploration de l'espace de recherche avec simplement de petites transformations comme la mutation consistant à modifier un 1 en 0 ou le contraire au sein de la chaîne chromosomique. Si les codages les plus adaptés aux stratégies que l'on a en tête ne sont pas ceux qui permettent la plus grande capacité d'exploration, le manque de signification est l'inconvénient du codage binaire. En fait, comme cela a été pratiqué par la suite sous le nom d'algorithmes évolutifs ou évolutionnaires, le codage peut être quelconque, ainsi un nombre peut être codé par sa suite décimale et rester «lisible» et une solution d'un problème symbolique peut rester une suite finie (ou non) de symboles.
Pour poursuivre l'analogie avec le monde vivant, on parle de «génotype» ou «génome» pour désigner la suite des gènes, la question est d'établir le lien avec le «phénotype» ou ensemble des caractéristiques de morphologie et de comportement. On sait maintenant, que dans la nature, une très grande partie du génome est sans utilité apparente, mais constitue en fait une réserve de possibilité. Le problème qui est souvent soulevé à propos de l'homogénéité d'une population, est de définir une distance voire une ressemblance floue entre individus, cela peut se faire aussi bien sur l'ensemble des génotypes c'est à dire des codes que sur l'ensemble des phénotypes c'est à dire des solutions réelles.
Signalons encore que tout problème d'optimisation est défini avec des contraintes, ce qui fait que l'espace de recherche est généralement d'une expression mathématique assez complexe. Une approche consiste alors à produire des individus dans un espace plus large mais plus simple à définir, et à pénaliser fortement les individus qui violent les contraintes par un système artificiel de bonus / malus qui s'ajoute à la fonction à optimiser.

c) Choix de la population initiale

On peut partir d'un ensemble de solutions approchées déjà connues, mais d'une manière générale il est souvent avantageux de partir d'une population aléatoire (sa taille peut aller de quelques dizaines à quelques centaines). Dans l'algorithme «standard» qui est une reproduction assez fidèle de l'évolution, la population est toujours de grande taille, mais avec beaucoup de recopies d'individus identiques. [Cerf 94] montre que pour ce type d'algorithmes, la vitesse de convergence est de l'ordre de la taille de la population.

d) Choix des opérateurs

Les transitions entre générations se font par «opérateurs génétiques» c'est à dire des «fonctions aléatoires» de $P \to P$ ou $P^2 \to P^2$ si P est la population.
L'opérateur le plus simple est la mutation, un gène est remplacé par un autre aléatoirement choisi, par exemple au chromosome AB**C**BBAC → AB**E**BBAC.
Des mutations bien particulières peuvent être envisagées suivant le problème, ainsi la transposition : deux gènes aléatoirement choisis sont permutés AB**C**EB**A**CD → AB**A**EB**C**CD, ou l'inversion : une coupure est choisie et les deux parties sont permutées ABCB**DEA** → **DEA**ABCB, l'inversion partielle : une partie du chromosome est inversée ABCD**EFD**CD → ABCD**DFE**CD etc.

Le «cross-over» ou croisement à deux sites : deux parents engendrent deux enfants en échangeant une partie choisie aléatoirement : AA**BED**ABC, DA**CBC**DAE → AA**CBC**ABC, DA**BED**DAE
C'est l'opérateur le plus puissant car intuitivement cela correspond à deux solutions comportant chacune une bonne partie du code, aussi en regroupant ces parties peut-on espérer améliorer la solution globale.
Coisements à un site : cet opérateur choisi un rang au hasard et échange les parties commençantes et finissantes des deux chromosomes.
Croisement uniforme : chaque gène de l'enfant est aléatoirement pris chez l'un des deux parents.
La reproduction : c'est dans le cas des «algorithmes génétiques standards», un simple dédoublement, une recopie du chromosome.
D'autres opérateurs peuvent être imaginés notamment des opérateurs respectant la cohérence dans la représentation du problème en chromosomes (exemple : respecter un aspect injectif ou bijectif...).

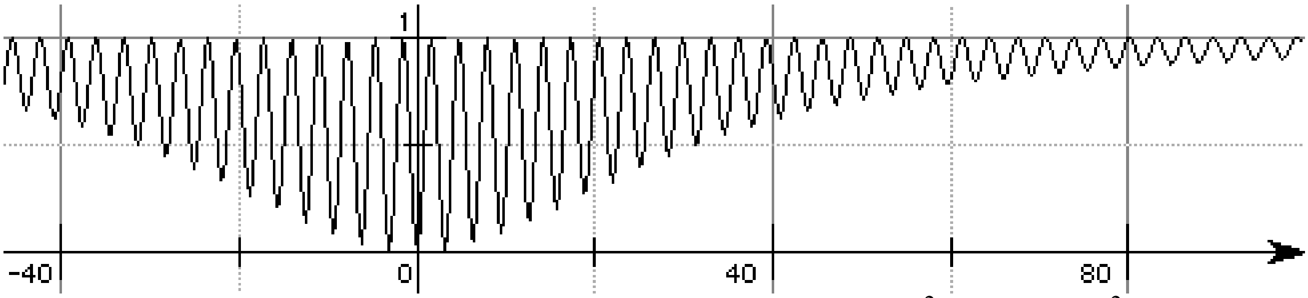

Figure 2 Exemple (dimension 1), de la fonction de Shaffer $f(x, y) = 1 - [1 - \sin^2\rho] / [1 + 0{,}001\rho^2]$ possèdant le minimum 0 en (0, 0), (ρ est la distance à l'origine de (x, y)) il ne faut pas moins de 1500 évaluations avec un algorithme standard où $p_c = 0.6$ et $p_m = 0.001$, pour y accéder, compte tenu du grand nombre d'oscillations sur $[-100, 100]^2$.

CROISEMENT ARITHMÉTIQUE

Le croisement arithmétique (dans le cas d'un codage réel) est une combinaison linéaire des chromosomes parents [Davis 93]. Plus précisément, dans le cas d'un codage réel, deux gènes x et y vont donner le gène $ax + (1 - a)y$ pour a tiré au hasard dans $[\max(\alpha, \beta), \min(\gamma, \delta)]$ si $x > y$, dans $[\max(\gamma, \delta), \min(\alpha, \beta)]$ si $x < y$ et a nul en cas d'égalité, avec [l, u] le domaine de ce gène et :
$\alpha = (l - y) / (x - y)$, $\beta = (u - x) / (y - x)$, $\gamma = (l - x) / (y - x)$, $\delta = (u - y) / (x - y)$. Ce croisement peut se faire sur un gène ou sur tous les gènes. Le croisement uniforme consiste à choisir le gène de l'enfant dans l'intervalle défini par les deux gènes correspondant des parents suivant une répartition uniforme, et le croisement gaussien consiste à le choisir suivant une répartition gaussienne centrée sur l'un des deux parents suivant un autre gène, ce qui revient à la dominance [Munteanu, Lazarescu 99].

MUTATIONS NON UNIFORMES [MICHALEWICZ 92]

Ces mutations consistent à effectuer par exemple des mutations de plus en plus faible, c'est à dire si l'ensemble des gènes est métrisable, à opérer dans des voisinages de plus en plus fins lors du remplacement d'un gène par un autre. Par exemple le gène numérique g est remplacé par $g \pm rg(1 - t/T)^a$ où $r \in [0, 1]$ et t le rang de la génération ramené au nombre maximal que l'on s'est fixé de générations T.

MUTATION DÉPENDANT DE LA DISTANCE

Le taux de mutation n'est pas fixé comme dans l'algorithme génétique standard, chaque fois qu'un croisement est opéré, une mutation s'exerce avec un taux $1 - d(p, m)$ où $d(p, m)$ est la distance relative définie dans [0, 1] entre les deux parents [Sefrioui 98]. De cette façon, plus les parents sont proches, plus on va chercher à explorer loin d'eux.

CROISEMENT RESPECTANT UNE CONTRAINTE

Pour rester dans l'espace admissible, [Michalewicz, Schoenauer 96] proposent par exemple une mutation sphérique pouvant se définir sur un individu x par $x_i \to qx_i$ et $x_j \to x_j[1 + (1 - q^2)(x_i/x_j)^2]^{1/2}$ et un cross-over sphérique entre x et y, par $x_i, y_i \to [\alpha x_i^2 + (1 - \alpha)y_i^2]^{1/2}$ si la recherche se situe sur une hypershère $\sum x_i^2 = 1$.
Pour rester dans l'hyperboloïde $\prod xi = 1$, on peut de même définir une mutation géométrique $x = (\dots x_i \dots y_i \dots) \to (\dots qx_i \dots x_j/q \dots)$ et un croisement géométrique $x, y \to (\dots x_i^q . y_i^{1-q} \dots)$ avec q un nombre aléatoire uniforme dans [0, 1].

UN ALGORITHME PARTICULIER D'APPRENTISSAGE INCRÉMENTAL BASÉ SUR UNE POPULATION [BALUJA 95]

Pour résoudre un problème d'optimisation défini sur $\{0, 1\}^n$, on utilise un vecteur $(p_1, p_2, \dots p_n)$ de probabilités, toutes initialement fixée à 0,5. Une population de μ (de l'ordre de 100) individus situés dans l'espace de recherche est recréée à chaque itération de la façon suivante : chaque composante x_i de chaque individu x est aléatoirement choisie 0 ou 1

suivant la probabilité p_i. Cette technique évolutionnaire utilise donc un opérateur génétique unique, la mutation (binaire) d'un gène.
Ces individus sont ensuite classés suivant la fonction à optimiser, du meilleur M au pire m. Le vecteur de probabilité est alors mis à jour par : $p_i \leftarrow (1 - \eta)p_i + \eta M_i$. grâce à un taux d'apprentissage η fixé à 0,1, puis en s'éloignant de la plus mauvaise solution, si $M \neq m$, par une mise à jour : $p_i \leftarrow (1 - \xi)p_i + \xi M_i$. où le "taux négatif d'apprentissage" est $\xi = 0,075$.
Une variante consiste à réaliser l'apprentissage de la même façon mais avec $\xi = 0$ et une modification aléatoire des probabilités : $p_i \leftarrow (1 - \omega)p_i + \omega\partial$ utilisant un taux de mutation ω fixé à 0,02 et $\partial = 0$ ou 1 avec équiprobabilité.
Des tests tels que les problèmes du voyageur de commerce ou du sac-à-dos, pour 2000 générations et $\mu = 100$ donnent un certain avantage à cet algorithme.

REMARQUES

Si $G = \{g_1, ..., g_{ng}\}$ est l'ensemble des gènes sur lequel on définit une distance $d(g_i, g_j) = \partial_{ij}$ ou par $|i - j| / (ng - 1)$ au cas où on dispose d'un ordre canonique sur G. Si G^n est l'ensemble des chromosomes, donc de topologie discrète, on peut cependant définir une distance dans [0, 1] par : $d(C, C') = (\sum_{1 \leq i \leq n} d(g_i, g'_i))) / n$, si $C = g_1 ... g_n$ et $C' = g_1 ... g'_n$, on aura par exemple :
Pour une mutation (ou une recopie d'un gène à droite) $d(C, mut(C)) \leq 1 / n$
Pour une transposition (un échange de deux gènes) $d(C, transpo(C)) \leq 2 / n$
Pour un cross-over où k gènes sont échangés, si C'_1 and C'_2 sont les fils de C_1 et C_2 alors : $d(C_1, C'_1) \leq k / n$ mais surtout : $\max [d(C_1, C'_1), d(C_1, C'_2), d(C_2, C'_1), d(C_2, C'_2)] \leq d(C_1, C_2)$ se qui s'exprime par le fait que la distance entre les parents détermine deux voisinages dont l'intersection contient les enfants.

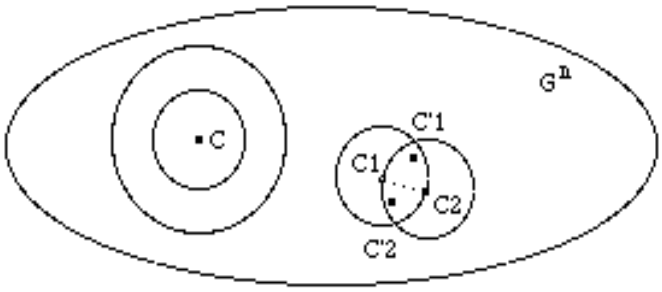

Figure 3 Dans l'espace métrique des chromosomes, visualisation de l'exploration par mutation, transposition et cross-over.

P ne doit pas être trop petit, ni la fonction d'évaluation trop restrictive sinon on risque de voir les solutions converger vers un optimum local. La fréquence d'application de chacun des opérateurs doit être surveillée afin d'utiliser les «bonnes» transitions entre générations. Il a été proposé de mesurer la diversité de la population, si m = card(P), par l'indice $\partial(P) = (2 / m(m - 1)) \sum\sum_{i < j} d(C_i, C_j)$ (le cross-over ne la change pas) afin de trouver des opérateurs qui ne baissent pas trop cette diversité.

EXEMPLE, dans le problème classique du voyageur de commerce, une solution est une permutation de toutes les villes, la liste (A B C D E F) par exemple s'il y en a 6. La fonction d'évaluation à minimiser est alors la somme des distances, mais les transitions devront être adaptées de façon à ce que les «solutions» restent toujours bijectives sur l'ensemble des villes. Les algorithmes génétiques sont appliqués à toutes sortes d'autres problèmes d'optimisation où les contraintes sont assez fortes mais difficiles à exprimer, depuis la recherche du dessin d'un graphe minimisant le nombre d'arêtes se coupant jusqu'au problème de l'ordonnancement d'atelier où il faut minimiser la charge totale (temps utilisé par chaque machine où différentes pièces occupent différentes machines pour des durées différentes et suivant des contraintes de préséances) [Ghedjati 94].

e) Mode d'application des opérateurs

PRINCIPE DE ROUE DE LOTERIE BIAISÉE

Dans l'algorithme génétique désormais appelé «standard» les opérateurs sont appliqués suivant une certaine probabilité choisie au départ et suivant le principe de «roue de loterie biaisée», on donne l'avantage aux meilleurs individus, si les valeurs des individus sont $v_1, v_2, ..., v_n$ et que $p_i = 1 - v_i / (\sum v_k)$, en tirant au hasard dans la population un numéro entre 1 et n avec cette distribution $(p_1, p_2, ..., p_n)$ dans le cas de la minimisation d'une fonction positive. Le chromosome choisi est tout simplement recopié à la génération suivante dans le cas de la reproduction.
Cette probabilité peut aussi être choisie comme inversement proportionnelle au rang de x dans le classement de P du meilleur au pire (stratégie «ranking»).

Si le cardinal de la descendance d'un individu est a (en moyenne) pour le meilleur, et b pour le moins bon, la différence entre les deux $a - b \in [1, 2]$ est appelée pression de sélection, il a été observé une uniformisation de la population d'autant plus rapide que cette pression est grande.

RESTRICTION "POISSON DE CORAIL"

Parmi toutes les idées, on peut citer celle du «poissson de corail» [Koza 94] consistant à déclarer «étalons» les 10% individus les plus performants et à réserver le croisement entre un étalon et un autre individu afin de favoriser le brassage et l'exploration de la population.

LE "MATING"

Cette idée consiste également à apporter une restriction dans le choix des parents [Fonseca, Fleming 95], le croisement ne s'effectue que si les parents sont distants de moins de σ. Cette borne peut être prise identique à celle du "sharing", (voir plus loin le partage en niches) les deux méthodes pouvant d'ailleurs être combinées.

"FITNESS DISTANCE CORRELATION" [JONES 95] [KALLEL, SCHOENAUER 97]

Toujours dans le cas métrisable, soit ^d la moyenne de la distance des individus à l'optimum, pour une population de taille n et dans le cas de la maximisation, le coefficient $\tau = (1/n\sigma_d\sigma_f)\Sigma_{1\leq i\leq n}(f - \hat{f})(d_i - \hat{d})$ permet de classer les problèmes :

$\tau < -0{,}7$	Problème trompeur, la performance se dégrade quand on se rapproche de l'optimum.
$-0{,}7 < \tau < 0{,}7$	Problème difficile, pas de direction pour "tirer" vers l'optimum.
$0{,}7 < \tau$	Problème facile, si f croît, les individus sont amenés à l'optimum global.

f) Renouvellement des générations

A chaque étape ces opérateurs sont tirés et appliqués au hasard, on peut alors prendre les chromosomes et leurs images et éliminer les plus mauvais (ce qui fait par exemple deux suppressions pour quatre individus dans le cas du cross-over) ou alors trier toute la population et sa descendance et ne conserver que les meilleurs en assurant à P un cardinal constant.

La stratégie standard consiste à appliquer (suivant de très faibles probabilités) des mutations (plutôt, d'après ce qui précède, aux meilleurs individus issus du cross-over) et à garder une taille fixe pour la population, en supprimant les plus mauvais.

Les stratégies dites élitistes (évolution lamarckienne) consistent à trier immédiatement entre parents et enfants à chaque application d'un opérateur génétique, pour ne garder que les meilleurs. Mais on trouve également les stratégies étudiées depuis 1964 en Allemagne, suivant laquelle les opérateurs sont appliqués à chaque génération à chacun des μ individus et produisent λ enfants (le rapport λ / μ étant de l'ordre de 7).

Le choix est alors la statégie $(\mu + \lambda)$ où μ parents produisent λ enfants, et le tout est trié à l'issue de la génération ou bien la stratégie (μ, λ) où les μ meilleurs uniquement pris dans la descendance sont retenus, en ce cas, la meilleure performance n'est pas obligatoirement monotone comme pour les algorithmes standard [Bäck, Kursawe 94]. Récement, [Bäck 97] ces stratégies ont été redéfinies dans un cadre plus général où un paramètre κ désigne le nombre de générations maximal où un individu est présent (κ est donc 1 pour les stratégies (μ, λ) et infini pour les $(\mu + \lambda)$).

LE "STEADY-STATE ALGORITHM" SSGA [WHITLEY, KAUTH 88]

A chaque génération deux parents sont aléatoirement choisis suivant le principe de la roulette favorisant les meilleurs, le crossover puis une mutation sur chacun des enfants étant appliqué, ceux-ci sont évalués. Les deux enfants ne remplacent pas leurs parents comme dans l'algorithme classique, mais les deux pires individus de la population.
Une autre version testée plus loin consiste à chaque génération, à appliquer tous les opérateurs produisant n enfants pour n parents. Un taux est choisi (par exemple 25%) et les 25% meilleurs enfants remplacent les 25% pires parents.

LA SEGMENTATION DE L'ESPACE DE RECHERCHE

Une idée analogue à celle des niches, mais plus "géométrique" consiste à segmenter l'espace de recherche de manière régulière au départ, en prenant un individu dans chaque pavé. A chaque génération, ce sont ces pavés qui sont classés, et, suivant la "fitness" de leur représentant, il sont coupés pour les meilleurs, ou fusionnés pour les plus mauvais.

g) Arrêt de l'évolution

L'arrêt peut être décidé si la solution connue est obtenue à un seuil de précision fixé, ou bien au bout d'un nombre de générations limité ou encore après un nombre donné d'évaluations de la fonction.
Signalons pour finir que l'intérêt de ces méthodes évolutionnaires est son applicabilité aux architectures parallèles.

LE THÉORÈME FONDAMENTAL DE L'ALGORITHME STANDARD

Pour un alphabet de k symboles (plus le symbole * utilisé pour signifier un gène quelconque) et des chromosomes de longueur l, il y a $(k + 1)^l$ schémas. Un schéma est une famille de chromosomes décrite par un mot de l'alphabet avec *, ainsi pour le mot 0*1*, il y a une unification possible avec les chromosomes 0010, 0011, 0110, 0111 sur l'alphabet {0, 1}. Chaque chromosome choisi appartient à 2^l schémas.
Pour un schéma H, on note o(H) le nombre de gènes définis, ainsi o(*10*1**) = 3, et d(H) la longueur entre les deux gènes définis extrèmes ainsi d(*10***1*) = 5 et d(**1*) = 0.
Soit f la fonction d'évaluation, (ici à maximiser) et f_m sa moyenne sur la population entière à chaque génération. Pour un schéma H, on notera f(H) la moyenne de f sur la famille représentée par H dans la population.
Si à la génération t on note m(H, t) le nombre de chromosome ayant le schéma H le théorème est alors :
$\mathbf{m(H, t + 1) \geq m(H, t)f(H) [1 - p_c d(H) / (l - 1) - o(H)p_m] / f_m}$ où p_m est la probabilité d'une mutation, p_c celle du croisement. Il signifie que les schémas de courte longueur, d'ordre bas et de bonne valeur (les «building blocks») vont se reproduire exponentiellement.
On a pour une reproduction $m(H, t + 1) = m(H, t)\ f(H) / f_m$, en effet si C est un chromosome présentant le schéma H, la probabilité qu'il se reproduise est le quotient $f(C) / \sum f_i$ et $f(H) / f_m = n\sum\{f_i / C \in H\} / card(H).\sum f_i$ d'où la relation :

$$m(H, t)\ f(H) / f_m = n \sum\{f_i / C\in H\} / \sum fi = n\ proba(H)$$

En notant $f(H) = (1 + c)\ f_m$, la relation s'écrit $m(H, t + 1) = (1 + c)m(H, t)$ soit plus précisément $m(H, t) = (1 + c)^t$ ce qui veut dire que si $c > 0$ le bon schéma croît exponentiellement et que si $c < 0$ le mauvais schéma décroît exponentiellement.
Pour l'action d'un croisement simple (échange des parties droite et gauche d'une coupure), la probabilité p_s de survivance du schéma H de longueur d(H) est que la coupure survienne hors de la zone définie :

$$p_s = 1 - d(H) / (l(H) - 1) \geq 1 - p_c d(H) / (l(H) - 1)$$

Pour l'action d'une mutation, la probabilité de survivance de chaque gène est $1 - p_m$ et donc la probabilité de survivance du schéma H est $(1 - p_m)^{o(H)}$ approchée par la valeur $1 - o(H)p_m$ dans la mesure où $p_m << 1$, d'où l'inégalité annoncée.

III LE PROBLÈME DE L'HOMOGÉNÉITÉ

Un des problèmes majeurs de tous les algorithmes évolutionnaires est d'éviter une convergence prématurée : un individu nettement meilleur que les autres va entraîner toute la population vers un optimum local, la population formée par des individus trop voisins ne peut plus explorer correctement l'espace. L'expérience montre effectivement souvent une trop rapide et trop grande homogénéité de la population [Schwefel 95].

La première idée est celle de suivre fidèlement la nature qui utilise le plus souvent la diploïdie [Smith, Goldberg 92]. Chaque individu est codé par deux chaînes binaires dont chacune est suffisante pour décrire complètement le phénotype c'est à dire la solution. Une relation de dominance (par exemple 1 domine 0 aux allèles de rang pair, le contraire aux rangs impairs...) permet de construire un individu (les allèles récessifs préservent en fait la diversité).

LA PRESSION SÉLECTIVE

Si f est la fonction à minimiser, et f_μ sa moyenne à une génération donnée, on modifie f en f' de façon à avoir $f'_{max} = cf_\mu$ avec $c \in]1, 2]$ appelée pression de sélection, le plus simple étant d'opérer une contraction de f autour de la moyenne, c'est à dire $f' = f_\mu[(c - 1)f + f(f_{max} - cf_{min})] / (f_{max} - f_\mu)$. L'effet est de rétrécir l'intervalle des valeurs de f, et de conserver la même moyenne, cette modification étant reprise à chaque génération.

LE TOURNOI RESTREINT

Cette méthode [Harik 95] cherche à éviter une convergence trop rapide. Quand le paramètre k augmente, la probabilité pour qu'une solution bien placée dans une niche soit détrônée, diminue. De bons résultats ont été obtenus pour une valeur k = 4*(nb d'optima à trouver) et $p_{cross} = 0{,}4$.

1) Deux individus a et b sont choisis aléatoirement
2) Le croisement et deux mutations permettent d'obtenir a' et b'
3) Un nombre k d'individus est choisi aléatoirement, soit a'' est le plus proche (au sens du génotype) de a', la même procédure produisant b'' le plus proche de b'.
4) a est remplacé par celui de a' ou a'' ayant la meilleure fitness, de même pour b.
5) Si non (critère d'arrêt) retour en 1

LES ÈRES

Des idées assez simples peuvent être envisagées pour opérer plusieurs évolutions séparées, ainsi, séparer la population en k sous-populations endogames à l'intérieur desquelles l'évolution se fait pendant un nombre donné de générations (une ère), puis (à chaque ère) la plus mauvaise de ces sous-populations est remplacée par le résultat de l'évolution sur la fusion des autres sous-populations.

LE "STUD-GA" DE [KHATIB, FLEMING 98] (CROISEMENTS UNIQUEMENT)

L'idée est de conserver le meilleur individu, pour produire une descendance grâce au reste de la population. Le codage étant binaire.

1 $t \leftarrow 0$ Initialiser une population aléatoire
2 Tant que $t < t_{max}$ Prendre le meilleur individu I de la population
3 Exécuter un croisement entre I et chacun des autres, pour remplacer la population
4 Incrémenter (t) Aller en 2

Cet algorithme peut être agrémenté dans le cas où il existe une distance d'un test à chaque génération du type si "diversité" est supérieure à un seuil donné alors faire un croisement sinon muter le conjoint avant.

L'ALGORITHME GÉNÉTIQUE LIBRE DE PARAMÈTRE [SAWAI, KIZU 98]

Cet algorithme inspiré par l'observation de la disparité, est plutôt dans la lignée de l'algorithme génétique standard, il prend en compte une population P, une sous-population P' et une famille P'' :

1 Choisir un individu aléatoirement dans la population courante P (initialement aléatoire) et l'ajouter à P'
2 Choisir un individu aléatoirement dans P et l'ajouter à P'
3 Choisir 2 individus distincts de P' et les croiser
4 Prendre l'un des enfants et le muter, soit P'' l'ensemble de ces 2 parents et 2 enfants, évaluer P''.
5 Parmi P'', sélectionner 1 à 3 individus suivant la règle :
    si des 4, les 2 meilleurs sont les parents, on en prend 1,
    si ce sont les enfants, on prend les 3 meilleurs de P'',
    si c'est un enfant et un parent, on prend ces deux.
    Ces individus sélectionnés sont ajoutés à P', il y aura donc répétitions.
6 si card P' > 1 aller en 3 sinon aller en 2

P' possède un cardinal variable, il doit donc être borné.

MOSES (MUTATION OR SELECTION EVOLUTIONARY STRATEGY) [FRANÇOIS 98, 99]

Si f est définie sur un ensemble fini E, on note d(a, b) le nombre minimal de mutations pour aller de a vers b (définir un chemin dans le graphe d'exploration), soit d* la plus grande distance dans E entre le minimum global et un autre minimum local et n* la distance maximale entre l'optimum global et un autre point de E. L'algorithme consiste à diminuer le nombre de mutations de générations en générations (pas de cross-over), les meilleurs individus ayant la permisssion de faire de longue marches aléatoires dans E:

1 $t \leftarrow 0$, $p_t \leftarrow 1$
(p_t est une probabilité, éventuellement plus petite que 1 au début, et qui va décroître au cours des générations t)
2 P est une population de taille n d'individus aléatoirement produits dans l'espace de recherche E.
3 Incrémenter t, décroître p_t et choisir un nombre aléatoire N suivant la loi binomiale $B(n, p_t)$
4 P n'étant pas triée, sélectionner le meilleur individu I de la population.
5 Muter les N premiers individus et les évaluer (il y a donc très peu de chance de tout muter)
6 Remplacer les n - N derniers individus par I
7 Tant que non (critère d'arrêt) aller en 3

A chaque génération t, le nombre d'évaluation de la fonction f à optimiser est donnée par l'espérance $E(N) = np_t$ qui joue le rôle du paramètre de température dans le recuit simulé.

Si $n > n^*$ et la suite p_t est décroissante et converge vers 0 et si la série $\sum p_t^{\ d^*}$ diverge, alors il y a convergence en probabilité du meilleur individu vers l'optimum global, c'est à dire $\lim_{t \to \infty} pr(I_t = x_{opt}) = 1$.
C'est donc le cas de lois de décroissnce telles que $p_t = 1 / t^{1/D}$ où D est le diamètre de E et la taille $n > D$ (donc $n > n^*$).

Le rayon de mutation α peut également être modifié à chaque génération.
Pour les test réalisés sur des fonctions telles que Rastrigin, la taille n'est pas un facteur significatif, pour n = 100 par exemple le nombre d'évaluation est de l'ordre de 1000 à 10000 si D = 60 pour 10^7 points visitables.

Une autre idée conforme à la nature peut être exploitée, les «niches» ou sous-populations à l'intérieur desquelles l'évolution est entraînée par des fonctions d'évaluations différentes, voire encore les «ères» ou durées d'évolution à définir au terme desquelles un traitement spécial est apporté... Pour garder le plus longtemps possible, plusieurs minima locaux des fonctions à optimiser, ces algorithmes établissent une partition de la population, en migrant entre ces niches périodiquement, idée déjà avancée par [Franz 72] et [Tanese 87].

LES NICHES

Afin de garder aussi longtemps que possible tous les minimums d'une fonction et d'éviter une trop grande homogenéité, il est possible de faire une partition de la population en «niches» [Horn 93] et d'effectuer périodiquement des «migrations» entre les sous-population [Franz 72]. Un autre point de vue est de changer la fonction d'évaluation [Goldberg 87]. La distance peut être définie sur les génotypes ou bien sur les phénotypes c'est à dire être définie sur l'ensemble des codes des chromosomes ou bien sur les effets, sur la fitness.

Le principe, est le suivant : soit σ un rayon et $n_i = \text{card}\{j \in P \,/\, d(i, j) < \sigma\}$, effectif mesurant l'importance du voisinage de i. En cherchant à minimiser f, le fait de remplacer f(i) par f'(i) = nif(i), on avantage les individus isolés dans le classement (et donc la troncature) de P.
Ainsi si $n_1 < n_2$ et $f_1 = f_2$ alors $f'_1 < f'_2$ et si de plus $f_1 < f_2$ alors l'écart s'accroit. Inversement si $f_1 < f_2$ mais $n_1 > n_2$, on peut risquer une inversion dans le classement, par exemple $f_1 = 2$, $f_2 = 3$ $n_1 = 3$ $n_2 = 1$ alors $f'_1 > f'_2$.

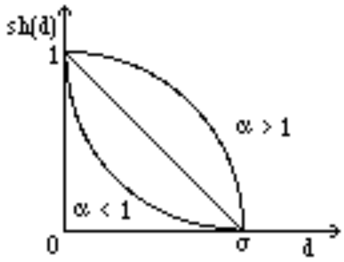

Figure 4 Fonction d'appartenance au voisinage de rayon σ

Cette idée a été étendue au moyen d'une notion floue de voisinnage. Si sh est une fonction de partage (sharing function) sur [0, 1] alors on peut voir comme une «accentuation» de la négation avec $sh(x) = 1 - x^{\alpha}$ où $\alpha < 1$ pour avoir $sh(x) < x$ ou bien $\alpha > 1$ pour le contraire $sh(x) > x$. Soit enfin $sh(x) = 0$ si $x > 1$.

Si n est la taille de la population et σ un paramètre déterminant la taille des niches, au cas où il est possible de définir une distance d dans l'espace de tous les chromosomes, alors si f est l'évaluation à la génération courante où les individus sont $c_1, c_2, \ldots, c_n$ alors la nouvelle évaluation f' pour la génération suivante est :

$f'(c_i) = f(c_i) . \Sigma_{1 \le k \le n}\, sh\,[d(c_i, c_k) / \sigma]$. Dans le but de minimiser cette évaluation, (il faut faire une division si f est à maximiser) on peut voir que plus un chromosome possède d'autres chromosomes voisins, meilleure est son évaluation, en cherchant à pénaliser les zones denses, l'exploration est ainsi favorisée. [Goldberg, Richardson 87] ont montré qu'une stabilisation est observée quand les valeurs des maximums f(i) sont en proportion avec la taille des ensembles flous $n_i = \Sigma_{1 \le k \le n}\, sh\,[d(c_i, c_k) / \sigma]$ formés par les niches.

IV OPTIMISATION MULTICRITÈRE

Si le problème du codage est délicat, si celui du choix de l'algorithme l'est aussi, le problème de la définition de la performance est encore plus difficile. En effet, dans bien des questions pratiques, l'adéquation d'un individu à un objectif n'est pas clairement formalisable, le plus souvent, il s'agit d'une agrégation de plusieurs critères hétérogènes (performances physiques, prix, complexité du codage ...) dont les importances relatives sont sujettes à discussion.
L'idée d'adapter des stratégies évolutionnaires à une optimisation multicritère pour trouver une population de «bonnes» solutions au lieu d'une solution unique est apparu avec [Schaffer 85] qui considère une partition de la population en relation avec chacun des critères, puis avec [Kursawe 91] et enfin [Horn 93].

Ordre partiel de Pareto Pour plusieurs critères $f_1, f_2, f_3, \ldots f_p$, à minimiser, on dit que deux individus i et j de l'espace de recherche E sur lequel s'appliquent les fonctions $f_1, \ldots f_p$, sont comparables, si pour tous les critères, ils sont rangés dans le même ordre. Le meilleur est dit dominant l'autre. On dit qu'il sont incomparables dans le cas contraire. Par exemple, pour deux critères, (0, 1) et (1, 2) sont comparables, (0, 1) et (1, 0) ne le sont pas. Un individu est premier (ou atomique) s'il est minimal pour cet ordre partiel. L'ensemble de Pareto est défini comme l'ensemble des éléments premiers de E (et donc incomparables deux à deux).

En ce qui concerne l'optimisation multicritère, un certain nombre de méthodes ont été utilisées pour se ramener à un seul critère :

MOYENNE PONDÉRÉE

La méthode jusque là la plus employée consiste à définir une moyenne pondérée de ces différents critères pour se ramener à une seule fonction. Bien entendu cette méthode ne peut être qu'empirique car les critères sont hétérogènes (puissance, prix, consommation ..) et de petites variations sur les poids accordés à ces critères peuvent entrainer de fortes différences pour les optima trouvés.
Dans tout problème d'optimisation, on peut tolérer une sortie de l'espace de recherche moyennant certaine pénalité, par exemple, pour minimiser f dans l'espace E caractérisé par $g < 0$ on pourra tenter de minimiser $f + p.\max(0, g)$. Le problème critique de la fixation de la pénalité p est analogue à celui des coefficients de moyennes [Michalewicz, Schoenauer 96].

L'APPROCHE "MINIMAX"

Il s'agit de maximiser le nombre d'optima atteint, mais lorsque des buts sont fixés. On peut aussi, lorsque les buts sont connus, tenter de minimiser la distance entre le vecteur à atteindre et le vecteur (f_1, ... f_p)

L'APPROCHE LEXICOGRAPHIQUE

Plus réaliste, cette méthode consiste d'abord à attribuer un ordre de priorité envers les différents critères. L'objectif le plus prioritaire est utilisé en premier pour ordonner la population et ainsi de suite par exemple pour trois critères : (0, 1, 5) < (0, 2, 1) < (1, 2, 1) < (1, 3, 0) < (2, 0, 2) < (2, 1, 0) < (2, 1, 1). A propos d'un problème multicritère en génie chimique, on trouve chez [Viennet Fonteix, Marc 95] une application des algorithmes génétiques, s'exerçant sur chaque critère successivement.

L'APPROCHE VEGA (VECTOR EVALUATED GENETIC ALGORITHM)

Due à [Schaffer 85], des sous-populations P_k sont sélectionées pour chaque objectif, l'algorithme standard est appliqué dans chacune des sous-populations (spéciation).
Puis, pour l'ensemble, le critère ordonant la population est $\Phi(x) = \Sigma_{1 \leq k \leq p}$ card(P_k) $f_k(x) / (\Sigma_{y \in P} f_k(y))$. Les tailles des populations (cardinaux) peuvent être les mêmes.

LE PARETO RANKING (MOGA : MULTIPLE OBJECTIVE GENETIC ALGORITHM)

Pour se ramener à un ordre total, on se base sur une importante population sans éliminer les individus dominés, il suffit alors pour rang(x) de compter le nombre d'individus meilleurs que x pour le pré-ordre de Pareto, les individus provisoirement "premiers" ont le rang 0. L'idée de priorité et de classement en groupes de priorités est de plus intégrée. [Fonseca, Fleming 95, 97]. Ce processus est représenté avec les rangs, ci-dessous dans le cas de deux critères f_1 et f_2 à minimiser, les individus sont classés en fronts successifs.

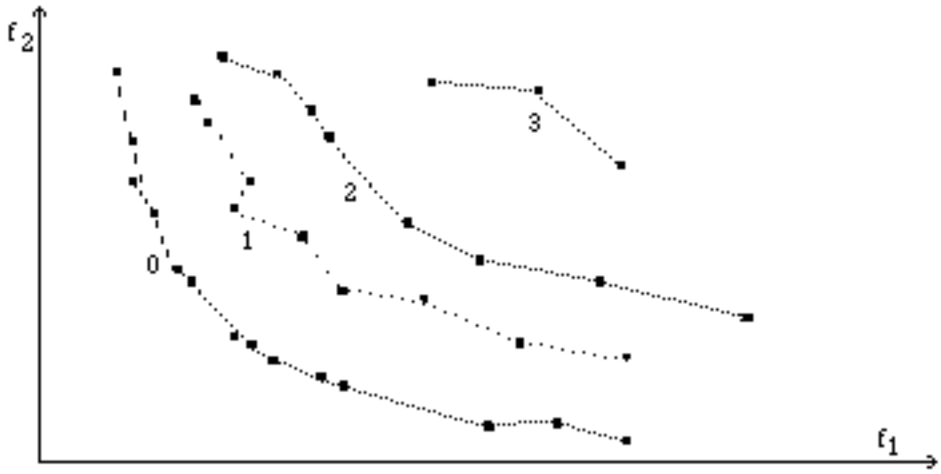

Figure 5 Classement par non-dominance pour deux critères, les éléments de rang 0 forment le front de Pareto.

LE NSGA : NICHED SORTING GENETIC ALGORITHM

Il s'agit simplement du pareto-ranking, qui détermine les fronts, où on modifie les critères à optimiser par le nichage. Seules les solutions voisines sont désavantagées. [Srivinas, Deb 94]

LE NPGA : NICHED PARETO GENETIC ALGORITHM

Dans cet algorithme qui combine l'idée de tournoi avec la dominance de Pareto, il n'y a pas de tri mais une épuration progressive des individus. [Horn, Nafploitis, Goldberg 94]
La population P contient un sous-ensemble T de taille réduite t.

1 Deux individus de P sont tirés au hasard,
  si l'un est dominé par au moins un élément de T et l'autre non, on garde le second.
  si les deux sont dominés ou non dominés par T, on ne garde que celui qui a la plus petite niche.

3 Celui qui a été conservé est sélectionné de nouveau.

LE SPGA : STRENGTH PARETO GENETIC ALGORITHM

Une population "externe" comprend les non-dominés, on leur donne comme fitness, le nombre d'individus plus mauvais qu'eux dans la population interne. P_e est une approche de la zone de Pareto. La population "interne" P_i est classée suivant la somme des fitness des individus externes qui le domine. On a ainsi deux ordres totaux [Zitzler, Thiele 98]. Il semble que SPGA rendent de meilleurs résultats que NSGA, lui-même supérieur à VEGA > NPGA > MOGA.

La recherche approchée de l'ensemble de Pareto

L'idée maîtresse de la recherche de l'ensemble (ou zone) de Pareto, consiste, au lieu de faire évoluer la population vers une bonne solution, à obtenir toute une population de solutions, les meilleures possibles suivant tous les critères mais incomparables deux à deux, en laissant le choix à l'utilisateur parmi un petit nombre de ces solutions. Une vision concrète de ce problème a été étudié par [Korhonen, Laakso 85]. Pour des solutions codées suivant deux dimensions, une représentation graphique et bien entendu approchée de l'ensemble de Pareto est proposée à l'utilisateur [Korhonen, Wallenius 88, 90]. Celui-ci peut changer les poids relatifs à un certain nombre de buts afin de modifier la direction de recherche et sa rapidité. En vue d'optimiser plusieurs critères, il est possible de ne conserver au sein de la population que des individus incomparables entre eux au sens de Pareto, c'est à dire qu'il existe au moins deux critères pour lesquels deux à deux les individus sont classées de façon inverse [Fonseca, Fleming 93], [Gacôgne 97].

1) L'option est de simuler en parallèle l'évolution de plusieurs sous-populations. Chacune d'entre elles étant triée suivant l'ordre lexicographique correspondant simplement à un ordre de priorité donné aux critères. Plus précisément, si f_1, f_2, f_3, sont les trois critères qui nous intéressent et qu'il est clair, par exemple pour trois critères, que l'optimisation de f_3 est moins importante que celle des deux premiers mais que nous hésitons sur l'ordre d'importance de ceux-ci, alors nous trirons une population P_1 suivant (f_1, f_2, f_3) et une seconde population P_2 suivant (f_2, f_1, f_3).
Il est en effet patent pour tout problème multicritère, qu'il est difficile de concevoir une combinaison linéaire satisfaisante des critères avec ou sans bonus-malus. Par contre, on peut souvent classer ces critères par ordre d'importance (prix, performances, sécurité...).

2) La seconde caractéristique de cet algorithme, est que chaque nouvel individu crée par un opérateur génétique est comparé aux autres de sa sous-population, et donc, est soit éliminé, soit conservé en provoquant ou non des éliminations. La population reste ainsi toujours formée d'éléments incomparables deux à deux afin de converger vers une partie de l'ensemble de Pareto. La conséquence de ce choix est que la taille de chaque groupe formant la population peut augmenter, ce qui nous oblige à fixer une taille maximale N_{max} .

3) Les deux populations suivant cette stratégie évoluent vers les meilleurs au sens respectifs de f_1 et de f_2 et ne sont pas croisés lors d'«ères» mais en permanence. Ce n'est donc pas l'endogamie qui les départage, mais le classement.

Exemple : Nous donnons ci-dessous quelques exemples de l'ensemble de Pareto pour deux problèmes multicritères, en partant d'une faible population aléatoire dans l'espace $[-1, 1]^2$, nous conservons toujours les meilleurs individus incomparables au sens de Pareto, en poursuivant sans cesse l'espace de recherche. Nous utilisons pour cela une population de taille variable, à l'intérieur de certaines limites, N_{min} et N_{max} (plusieurs centaines) ainsi qu'une population d'opérateurs génétiques définis en relation avec le problème considéré et évalués suivant le cumul de leur performance à améliorer les individus.

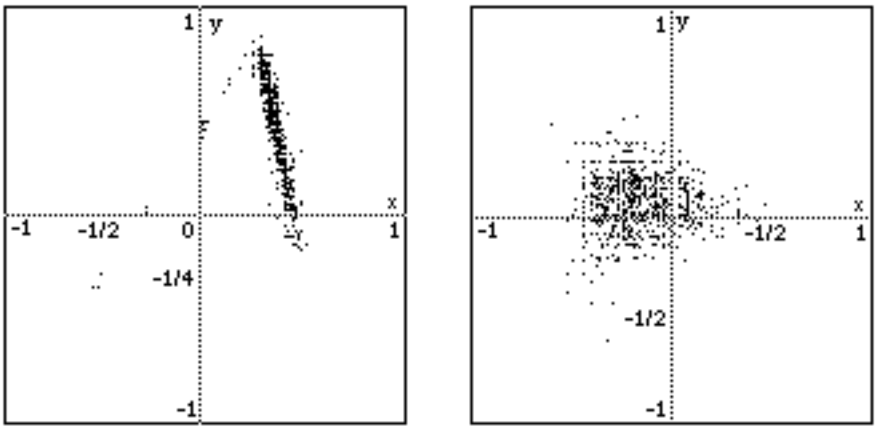

Figure 6 A gauche, exemple d'ensemble de Pareto non convexe (un boomerang), une population de 200 points obtenus en 50 générations pour deux critères :
$f_1(x, y) = 200(1 + 3x - 2y)^2 + 4(1 + 4x - 4y)^2$ et $f_2(x, y) = 200(1 - 2x)^2 + 7(1 + 4y)^2$ dont les minima sont respectivement (-1/2, -1/4) et (1/2, -1/4).
A droite la 50^e génération d'une population de taille 300 pour les 3 critères sans aucun ordre :
$f_1(x, y) = 50(x - 0.5)^2 + 80y^2$, $f_2(x, y) = 100(x + 0.5)^2 + 50(y + 0.5)^2$ et $f_3(x, y) = 50(x + 0.5)^2 + 100(y - 0.5)^2$,
Leurs minima 0 sont respectivement en (0.5, 0), (-0.5, -0.5) et (-0.5, 0.5).

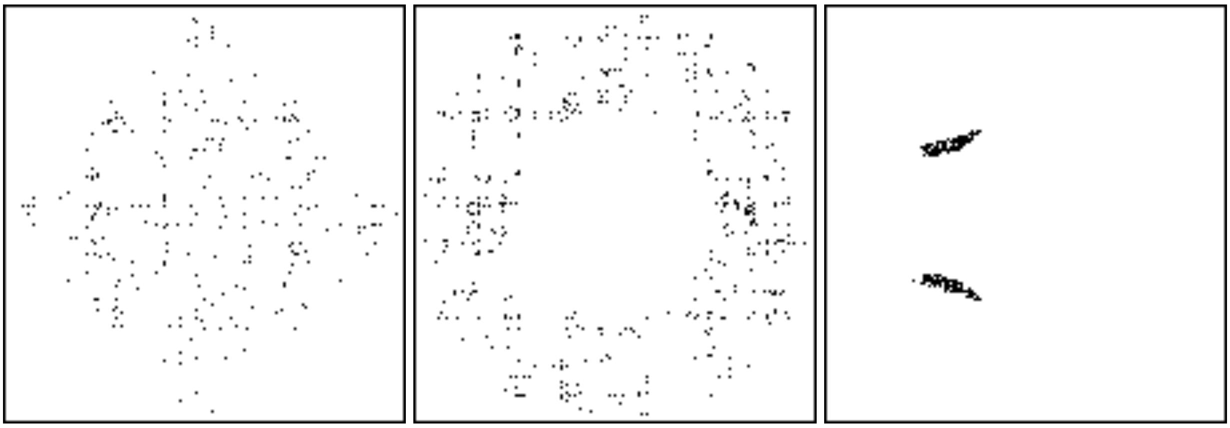

Figure 7 Population de 250 chromosomes obtenue en 50 générations sans aucun ordre pour les 4 fonctions à minimiser $|x + y - 1|$, $|x - y - 1|$, $|x + y + 1|$, $|x - y + 1|$.
Au milieu, la 50^e population de 500 chromosomes sans ordre, pour $100|x^2 + y^2 - 1|$ et $100|x^2 + y^2 - 1/4|$
A droite, la 50^e génération d'une population de 250 points qui converge vers les deux points de Pareto $(-1/2, \pm 1/2)$, pour les deux critères $f_1(x, y) = (2x + 1)^2$ et $f_2(x, y) = 1 + |\cos \pi y(1 - x)|$ (exemple d'ensemble de Pareto non connexe).

V CO-EVOLUTION

La co-évolution a d'abord été explorée de manière différentielle, le modèle le plus connu en dynamique des systèmes est celui de Lokta-Volterra datant des années 20. Si nous cherchons maintenant à simuler les populations réelles, l'évaluation d'un individu (ou d'une espèce) serait définie comme somme de ses performances face à tous les individus de la population courante, cette définition est tout à fait originale par rapport à ce qui est fait habituellement dans les problèmes d'optimisation résolus par algorithmes évolutionnaires, car de générations en générations cette fonction est modifiée et est donc à chaque étape une fonction de la population.

Mais cette définition de la performance présente un très gros inconvénient, à savoir que l'évaluation ne dépendant que de la population actuelle, il est impossible de faire des comparaisons. D'autre part, et c'est d'autant plus le cas si la population est réduite ou assez homogène, l'évaluation d'un individu n'est en fait que son évaluation relativement à son «voisinage».

C'est pourquoi la plupart du temps, on confronte les individus à une population de référence (voire à la population initiale). Le choix d'une population de référence suffisamment large et hétérogène reste bien sûr sujet à discussion, en fait un individu, quel que soit l'objectif fixé, doit être jugé sur ses aptitudes en toutes circonstances, c'est à dire ici, si le résultat moyen est bon lorsqu'il est confronté à un «grand nombre» d'autres individus très diversifiés. La solution est toujours un compromis. Voir le problème itéré des prisonniers. [Axelrod 87, 92] [Adachi, Matsuo 90], [Delahaye 92], [Gacôgne 94].

Un problème simple permet de comprendre la difficulté qu'il peut y avoir à définir la fitness. Dans le problème d'une poursuite le meilleur poursuivant serait celui qui obtient un bon score en toutes circonstances, c'est à dire confronté à n'importe quel pousuivi. On voit que le temps de calcul de cette évaluation oblige à faire un compromis sur le choix d'un échantillon de pousuivis représentatifs. Mais l'évaluation de ceux-ci oblige également à établir une moyenne sur un échantillon de poursuivants fixé ou bien eux-même en cours d'évolution ?

Quant au problème de l'évaluation d'une colonie évolutive d'individus, il s'agit d'un autre domaine. La colonie est-elle évaluée dans son ensemble à la fin d'une session pour déterminer la génération suivante, ou bien les individus peuvent-ils être pris séparément afin de se reproduire ?

VI L'ALGORITHME ÉVOLUTIONNAIRE ESAO

On introduit une stratégie génétique qui se distingue fortement de l'approche classique, son but est de laisser agir les opérateurs au gré de l'évolution, suivant leur capacité à faire évoluer la population.
Avant d'expérimenter de telles idées, nous exposons l'algorithme déjà utilisé [Gacôgne 93, 94].

L'algorithme

1) Un chromosome est codé en fonction du problème, ce peut être une chaîne de caractère, une liste, une structure quelconque mêlant champs numériques et symboliques. La population initiale P_0 est formée d'individus aléatoire et possède une taille fixe ou bien une taille entre deux bornes N_{min} (fixé à quelques unités) et N_{max} dans le cas de la recherche de l'ensemble de Pareto.

2) Une population OP_0 aléatoire d'opérateurs génétiques (mutation, transpositions de deux gènes, bruit gaussiens pour des gènes numeriques, création ou suppression d'un champ au sein du chromosome, croisement à un site, à deux sites, ... en particulier remplacement d'un sous-arbre par un autre) est constituée parmi des opérateurs qui peuvent être choisis en grand nombre en fonction du problème. Chaque opérateur est noté (initialement zéro) puis par le cumul sur plusieurs générations de sa capacité algébrique à faire décroître les individus.

3) A chaque génération, les opérateurs de OP_t sont appliqués dans l'ordre à la population P_t elle-même triée. Quand une transition op est appliquée à un élément i, nous trions immédiatement le père et le fils, ou les 4 individus s'il s'agit d'un croisement, afin de ne conserver que le meilleur. Le score de op est alors augmenté de f(op(i)) - f(i), ainsi, en triant les opérateurs dans l'ordre croissant, le meilleur sera appliqué au meilleur individu et ainsi de suite. D'autre part, avec cet «élitisme immédiat», nous ne conservons que peu d'individus lors de la descente vers un minimum local.

4) Si au moins un progrès a été enrégistré à la génération t, nous trions les opérateurs suivant leur performance, et nous créons un nouvel opérateur de l'espèce du meilleur d'entre eux. De cette façon, nous avons une adaptation de la population OP_{t+1} des opérateurs.

5) Dans le cas opposé où aucune amélioration n'est observée, tous les opérateurs sont réinitialisés et aléatoirement brouillés afin d'éviter la convergence de cette population vers un unique opérateur, qui après avoir remplit sa mission (une mutation par exemple) ne pourrait pas céder la place à un autre (le cross-over par exemple).

6) Si le nombre d'évaluations de la «fitness» dépasse le seuil fixé (par exemple 500000), l'algorithme est stoppé, sinon retour en 3.

Remarques : une procédure de déstabilisation peut être ajoutée, consistant à réinitialiser la population P_{t+1}, (son meilleur élément excepté) elle-même au cas (5) où aucun progrès n'est observé.
Une procédure d'élimination peut également être introduite, consistant à chaque génération, à ne conserver que le meilleur de chaque couple d'individus de génotypes proches à un seuil donné. Les individus éliminés étant remplacés par des individus aléatoirement créés, ce qui nécessite un balayage de la population du pire au meilleur, suivi d'un nouveau tri.

VII TESTS NUMERIQUES ET SYMBOLIQUES

Fonctions tests

Il est important de tester des fonctions numériques en dimensions variées 3, 5 ou 10, par contre les bornes fixées pour ces fonctions ne sont pas très pertinentes dans la mesure où elles sont «fortement multimodales» exceptées les deux premières fonctions qui sont convexes.

Paraboloïde

Si $x = (x_1, x_2, ..., x_n) \in [-5, 5]^n$ pour n = 5 ou 10, $f(x) = \sum(x_i - 1)^2$ (atteindre 10^{-6})

Fonction de De Jong

$F_J(x) = \sum |\text{round}(x_i)|$ pour $x \in [-50, 50]^n$ pour n = 3, 5 ou 10.

Fonction de Rosenbrock

$F_R(x, y) = 100(x^2 - y)^2 + (1 - x)^2$ dont le minimum 0 est atteint en (1, 1), (vallée en forme de banane) si $(x, y) \in [-2, 2]^2$ on teste avec le coefficient 100 et avec 10.

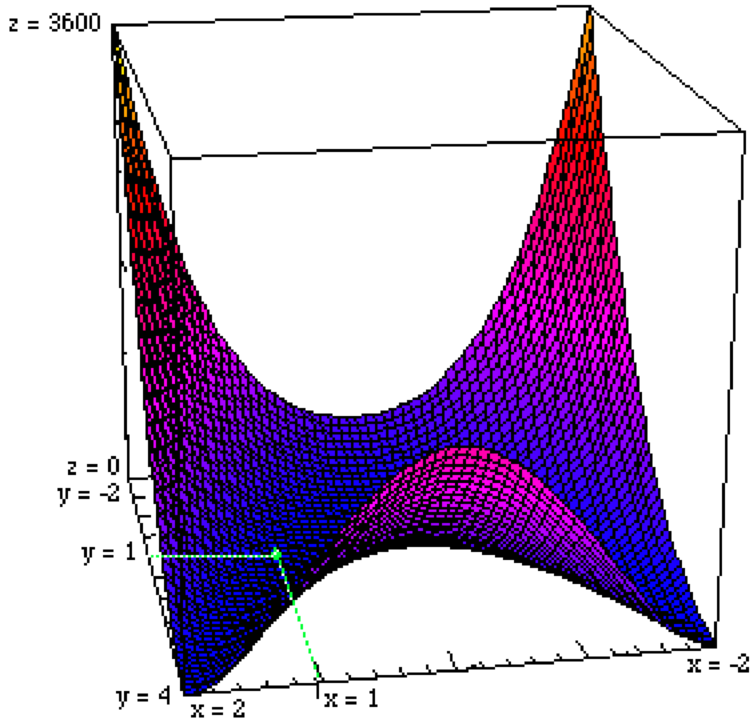

Figure 8 Fonction de Rosenbrock F_R (x, y) = $100(x^2 - y)^2 + (1 - x)^2$ minimum 0 en (1, 1), la vallée dont les bords sont très abrupts dès que |x| > 2, est courbe et le minimum se trouve en un endroit extrêmement plat. L'orientation choisie ici, est destinée à montrer les deux branches non symétriques de cette vallée, ces deux branches montent doucement pour y > 2, au point que cela n'est pas encore perceptible sur la figure pour y = 4.

Fonction de Rastrigin

$$F_R(x) = \sum_{1 \le i \le n} x_i^2 + 10 - 10 \cos(2\pi x_i) \text{ pour } |x_i| < 512$$

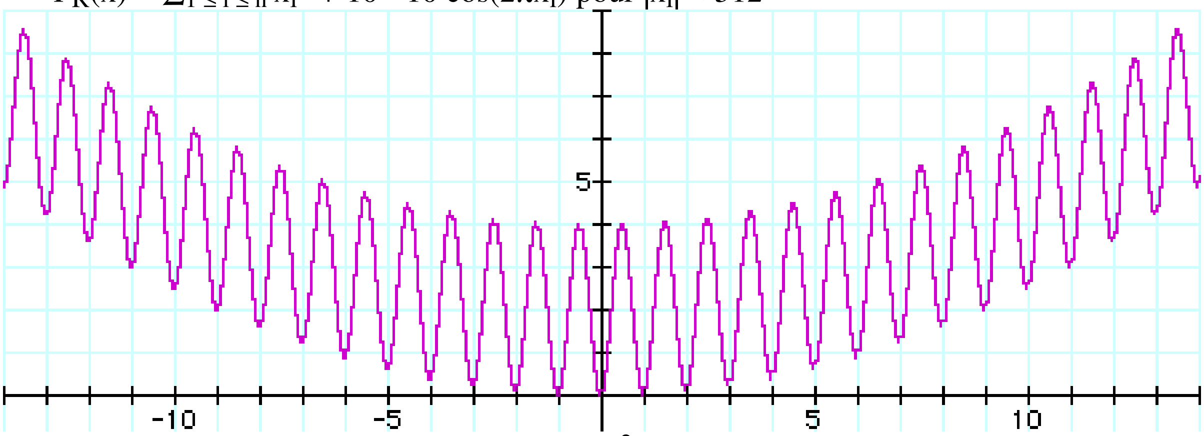

Figure 9 Fonction de Rastrigin $F_R(x) = x^2 / 40 + 2 - 2 \cos(2\pi x)$ en dimension 1

Fonction de Griewank n = 5, 10 d = 4000 dans $[-600, 600]^n$, atteindre 10^{-3}

$$F_G(x) = (1/d)\sum(x_i - 100)^2 - \prod\cos((x_i - 100)/\sqrt{i}) + 1$$

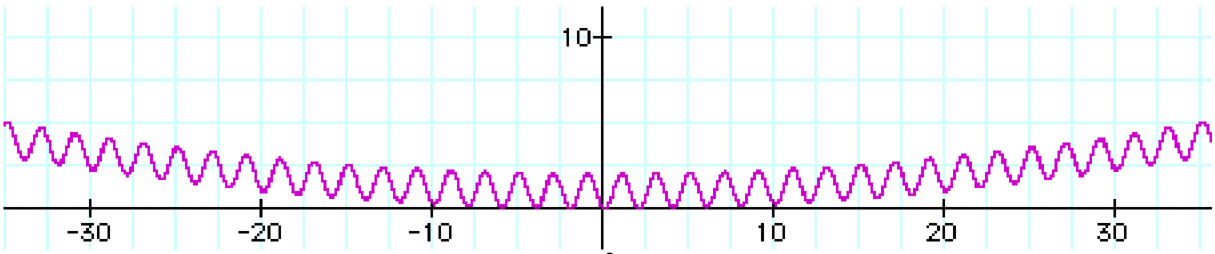

Figure 10 Fonction de Griewank $F_G(x) = x^2/d - \cos\pi x + 1$ en dimension 1 pour d = 400

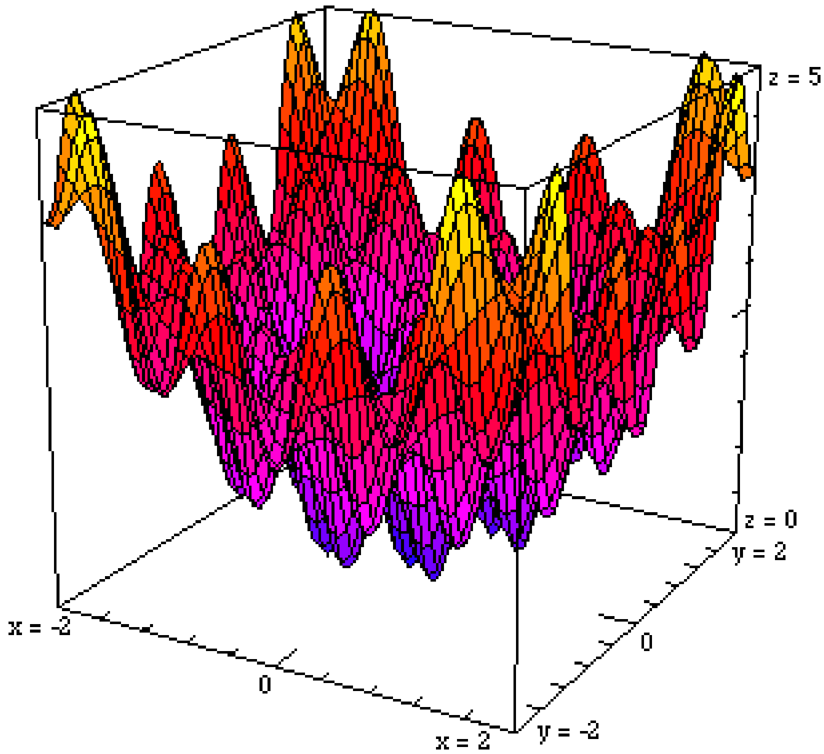

Figure 11 Fonction de Griewank $F_G(x) = (x^2 + y^2) / d - \cos 2\pi x . \cos 2\pi y + 1$ en dimension 2 pour d = 2. D'apparence analogue à la fonction de Rastrigin, la fonction de Griewank est en fait plus difficile à optimiser.

Fonction de Schwefel $F_W(x) = 4189.8291 - \sum_{1 \leq i \leq 10} x_i . \sin(\sqrt{|x_i|})$ pour $|x_i| < 512$
Toutes ces fonctions atteignent leur minimum 0 dans l'espace considéré.

Signalons encore les fonctions de Michalewicz : $F_M(x) = -\sum \sin(x_i) \sin^{2m}(i x_i^2 / \pi)$,
de Shekel (les terriers de renard) $F_S(x) = - \sum_{1 \leq i \leq m} 1 / (\sum_{1 \leq j \leq n} c_i + (x_j - a_{ij})^2)$ si $x \in [0, 10]^n$
et celle de Langerman $F_L(x) = - \sum_{1 \leq i \leq m} c_i (\exp(-(1/\pi) \sum_{1 \leq j \leq n} (x_j - a_{ij})^2) . \cos\pi(\sum_{1 \leq j \leq n} (x_j - a_{ij})^2))$
Les coefficients étant donnés, ces fonctions sont plus longues à évaluer.

Critère de comparaison

Une discussion peut être faite sur le critère de comparaison à définir à propos de diverses stratégies évolutionnaires. En effet, si dans le cas présent, le minimum est connu, il est possible de rapporter le nombre d'évaluations au nombre de convergences à un seuil donné, il est également possible de compter simplement le nombre d'évaluations ou encore de fixer celui-ci et de discuter sur la meilleure valeur obtenue pour la fonction. Cette dernière façon de voir est bien sûr mieux adaptée aux problèmes réels où l'optimum n'est pas connu a priori. En définitive, nous avons choisi, le nombre d'évaluations moyen sur 20 essais, pour atteindre le minimum au seuil fixé avec une restriction : pour comparer le nombre d'évaluation, un maximum «max» de ce nombre est fixé à 20000 (50000 en dimension 10 et 100000 pour Griewank) constitue une borne.
Pour les stratégies non élitiste ES(μ, λ), les résultats étant très dispersés, la moyenne est assez peu significative, c'est pourquoi elle a pu être portée sur 50 expériences.

Tests algorithmes standards et stratégies d'évolutions

Un grand nombre de tests expérimentaux ont déjà été effectués amenant à une synthèse générale, qui pourrait se résumer par deux faits. Toutes les stratégies proches de l'algorithme standard montrent une convergence lente avec une forte hétérogénéité de la population, l'amplitude de la fitness restant importante et les stratégies d'évolution présentent une convergence plus rapide avec une forte homogénéité [Bäck 93, 97]. Nous montrerons que la stratégie ESAO réalise un bon compromis entre ces deux positions.

	GA binaire avec croisement à un site, uniforme, mutation et mutation locale	GA binaire sans mutation, mais croisement à un site et uniforme	ES(50 + 500) binaire croisement et mutations
Paraboloïde dim = 3 (max = 50000)	24491	12600	7425
Michalewicz dim = 3 (max = 50000)	13756	8650	5400
Griewank dim=3 (max = 10000000)	902499	28487	139483

Figure 12 Comparaison des nombres moyens (sur 20 essais) d'évaluations des fonctions pour parvenir à un seuil fixé.

Tests sur des algorithmes divers

De nombreux tests ont également déjà été réalisés entre algorithmes génétiques et «hill climbing» [Davis 91], [Mitchell, Holland, Forrest 92] et «recuit simulé» [Ingbert, Rosen 92]. Ceux que nous réalisons ici sont quelque peu différents car ils le sont avec un codage réel identique à toutes les heuristiques et avec les mêmes opérateurs. Ainsi pour l'exploration du «voisinage» d'un point dans ces deux heuristiques, nous tirons au hasard un des opérateurs de la liste ci-dessous à l'exception des croisements. Pour les tests qui suivent, la population est de taille 50, à l'exception bien sûr des stratégies locales où elle est 1 et des algorithmes standard et «moses» où elle est de 500. Les solutions sont toutes représentées dans le même codage réel. Naturellement certains choix pris dans la perspective d'un banc d'essais resteront toujours discutables, ainsi pour les tests numériques suivants dans des espaces du type $[a, b]^n$, nous avons représenté les individus par des vecteurs de dimension n, de listes de chiffres (les décimales). Par exemple, pour n = 1, sur l'intervalle [7, 10], la liste d'entiers [3;3;3;3;3;3;3;3;3;3;3;3] signifie 1/3, qui ramené à cet intervalle, représente 8.
Les opérateurs utilisés pour la recherche d'un voisin étant choisis au hasard parmi la liste :

migration : individu entièrement nouveau, cet opérateur est utilisé pour initialiser les populations, mais aussi pour remplacer les doublons, et éventuellement suivant la stratégie choisie, au même titre que les autres.
migration-1 : tout est perturbé sauf les premiers éléments de toutes les composantes
migration-2 : la moitié, tirée au sort, des composantes est modifiée
migration-3 : une composante est modifiée
mutation-1 : mutation d'un chiffre choisi aléatoirement dans le développement décimal d'une des composante
mutation-2 : mutation identique d'un chiffre mais limitée à ce chiffre ±1.
ajout : ajout d'une décimale à la fin d'une des composantes
transposition-1 : deux chiffres sont échangées au sein d'une même composante
transposition-2 : deux composantes sont échangées pour une dimension supérieure à 2 seulement
crossover-0 : croisement uniforme avec un individu choisi aléatoirement dans le reste de la population
crossover-1 : croisement à 1 site
crossover-2 : croisement à 2 sites

Par ailleurs des individus trop voisins sont éliminés au profit du meilleur d'entre eux : entre deux vecteurs de R^n ou plutôt de leur codage en listes d'entiers, on définit un taux de ressemblance comme le minimum des taux observés sur chaque composante ceux-ci étant calculés comme le nombre d'items égaux, dans l'ordre d'énumération, rapporté à la plus petite longueur. Ce n'est pas du tout une extension floue de l'égalité mais une relation adaptée à la représentation décimale et destinée à éliminer de manière assez drastique les «ressemblances».
Ainsi on aura proche ([1;2;3], [1;2;3;4;5;6]) = 100% et proches ([1;2;3;4;5;6;7;8;9;0], [1;2;3;4;8;5;6;7;8;9;0]) = 40%. Le taux au dessus duquel l'élimination est pratiqué peut être fixé entre 20% et 50%.

	Hill-climbing	Nissen	Recuit simulé	GA 0.5-0.1	Stud-GA	Moses |P|=500	ES 7	SSGA 10%	SSGA 25%	SSGA 33%	SSGA 50%
De Jong dim 3	667	2881	5321	13256	17400	10096	2286	1460	1132	1254	1307
Paraboloïde dim 3	1602	6275	28251	max	48808	22798	5396	6155	4134	4056	5158
... ... dim=10	7433	17129	31721	max	max	45854	47391	16100	13842	12560	13827
Rosenbrock dim 2	29010	44090	47816	max	46884	33816	31952	39431	33359	32587	40210
Rastrigin dim 10	13449	max	23626	max	48937	46943	max	15422	13651	13950	15564
Griewank dim 10	max	max	max	max	max	max	max	73298	70819	43781	45450

Figure 13 Moyennes des nombres d'évaluations pour obtenir une contrainte (meilleure valeur < eps et nombre d'évaluations < max) sur 20 essais aléatoires. Pour le recuit simulé, le paramètre initial de température n'est pas aisé à fixer, il est fonction des valeurs que peut prendre la fonction sur le domaine retenu. En grisé les deux meilleurs résultats par ligne.

Comme on pouvait s'y attendre, le «hill-climbing» (page 4) apporte de meilleurs résultats sur une fonction convexe, avec cependant une forte dispersion au cours des 20 essais sur lesquels sont calculés les moyennes du tableau, mais cette heuristique se laisse enfermer dans des bassins d'attraction locaux pour les fonctions plus complexes. Il faut toutefois noter une très bonne performance pour la fonction de Rastrigin.

L'algorithme de Nissen exposé plus haut (page 4) peut s'apparenter à un «hill-climbing» qui serait renouvellé en cas de trop grande stationarité. Il présente l'inconvénient d'avoir à juger empiriquement d'un nombre maximal de générations : pour 10, la perturbation est beaucoup trop forte, pour 50, les paliers qui se produisent naturellement au cours de la recherche sont en grande partie conservés, cependant on obtient des résultats moins bons qu'avec le «hill-climbing» y compris pour les fonctions convexes.

En ce qui concerne le recuit simulé (page 4), la «température» initiale doit être ajustée suivant les valeurs que prend la fonction dans le domaine qu'on lui assigne, elle a été prise de 10 à 100 000, en fait une assez forte «température» doit être prise sous peine d'être trop vite réduite et donc d'enfermer la recherche. L'observation des expériences montre bien que, malgré de fortes températures initiales, la méthode du recuit simulé finit toujours par aboutir dans un puits qui n'est pas nécessairement celui du minimum global.

La piètre performance de l'algorithme standard (sur 500 individus) n'est pas modifiée en augmentant les probabilités, (en ce cas, la décroissance globale de la fitness est plus chaotique avec beaucoup de remontées), ni en diminuant les probabilités, ce qui a pour effet de ralentir encore la convergence.

Le stud-GA (page 11) converge lentement et par paliers, ceci malgré une procédure de déstabilisation de la population consistant à éliminer les éléments trop proches. Le problème de la précision du codage réel étant pourtant résolu grâce à une initialisation des nombres réels comprenant 6 chiffres en moyenne attribués suivant une loi de Poisson. En effet, fixer le nombre de décimales et opérer uniquement des mutations pourrait interdire définitivement l'attribution de décimales supplémentaires, ici, l'opérateur «ajout» répond à cette question. L'observation du déroulement de l'évolution montre que l'algorithme est incapable d'explorer hors du puits local où il parvient.

L'algorithme Moses (page 11) nécéssite une population plus étendue (500 individus) et une décroissance de la probabilité pas trop forte, (p initialement 1 est multipliée par 0,99 à chaque génération) sinon le nombre de nouveaux individus évalués tombant trop rapidement à zéro, la recherche ne peut, elle aussi, que stopper dans un puits local. En outre, si la descente est assez régulière elle est peu rapide comparée au SSGA.

ES 7 (pages 5 et 9) désigne la stratégie d'évolution pour laquelle chacun des 50 pères produit 7 enfants grâce à deux croisements et cinq mutations, les 50 meilleurs individus de l'ensemble parents + enfants sont retenus pour la génération suivante. Le paragraphe suivant montre qu'il est possible d'améliorer la procédure en variant les opérateurs.

Le SSGA (page 9), du moins la version testée, consiste à appliquer à chaque individu un des opérateurs de la liste aléatoirement tiré. Les plus mauvais parents sont remplacés par les meilleurs enfants à chaque génération. Bien que nécessitant davantage de tris que les autres stratégies, cette méthode a fourni d'assez bons résultats, et surtout la descente vers le minimum global se fait régulièrement sans grands paliers. Le taux de remplacement à appliquer ne semble pas très pertinent, mais il apparaît grossièrement optimal autour de 1/3.

Tests comparatifs entre stratégies d'évolutions ES(μ+λ), ES(μ, λ) et ESAO

Les tests suivants sont toujours réalisés avec une représentation des chromosomes comme vecteurs de listes de chiffres.

Le tableau 14 donne le nombre d'évaluations de la fonction pour satisfaire les mêmes contraintes (0 pour De Jong, 10^{-3} pour Griewank et 10^{-6} pour les autres). Les résultats pour des effectifs 10 et 50 restent du même ordre de grandeur. Dans les tests comparatifs présentés, la stratégie ES(μ+λ) est légèrement modifiée dans la mesure où les différents opérateurs appliqués au même individu ne sont pas pris comme mutations ou croisement, mais sont ceux énumérés plus haut.

Pour ce qui est du rapport λ/μ, on remarque une décroissance généralement plus rapide au dessus de 5 et des paliers plus ou moins longs en dessous. Pour de faibles populations, la création de nouvel individu, causée par des doublons, est relativement fréquente.
Il faut bien souligner qu'initialement dans les ES, les enfants produits par chaque parent ne le sont qu'avec mutations et croisements. Ces expériences, ainsi que d'antérieures, ont montré qu'il y a déjà une amélioration générale en variant les opérateurs ce qui peut s'expliquer par un plus large évantail de méthodes d'exploration de l'espace. Ici, ces derniers tests montrent donc qu'en multipliant les différents types d'opérateurs, l'optimum expérimental se situe encore aux alentours de λ/μ = 7, mais que cette stratégie est dépassée par ESAO, c'est à dire en établissant un score aux opérateurs afin de modifier leurs fréquences d'application.

Rapport λ/μ	ESAO	1	2	3	4	5	6	7	8	9
De Jong dim = 3 élitiste	579	747	745	721	863	910	836	689	811	803
... ... dim = 5	897	2311	1832	3869	1823	2140	1844	1145	1276	1460
... ... dim = 10	1996	10072	11327	6450	4698	5517	5580	2599	2896	3005
Rosenbrock dim = 2 μ = 10 élitiste	12114	18471	15698	15746	16036	15645	14746	13985	15977	14059
... ... non élitiste	18628	16407	19105	17864	16450	16210	15667	14857	13610	15917
Paraboloïde dim = 3 μ = 10 élitiste	2114	16628	15795	14081	max	19050	max	6856	7610	6341
... ... non élitiste	max	max	19634	12025	1932	2220	2305	3667	5942	5846
... ... dim= 5 μ = 10 élitiste	3083	19063	19395	18988	max	max	max	10282	11430	13595
... ... dim= 5 μ = 50	6857	16935	16185	18822	19170	19337	max	11057	12210	14427
... ... dim=10 μ = 50	25302	max	max	48507	48190	49225	max	23990	21310	28670
Griewank dim = 5 μ = 50 élitiste	49821	max	max	max	max	max	max	71927	max	max
... ... dim = 10	72027	max	max	max	max	max	max	80579	max	max

Figure 14 Moyennes des nombres d'évaluations pour obtenir une contrainte (meilleure fitness < eps et nombre d'évaluations < max) sur 20 essais aléatoires. En grisant le ou les deux meilleurs résultats par ligne, on vérifie assez nettement une confirmation du rapport optimal 7 pour λ/μ et une amélioration supplémentaire sensible avec ESAO.

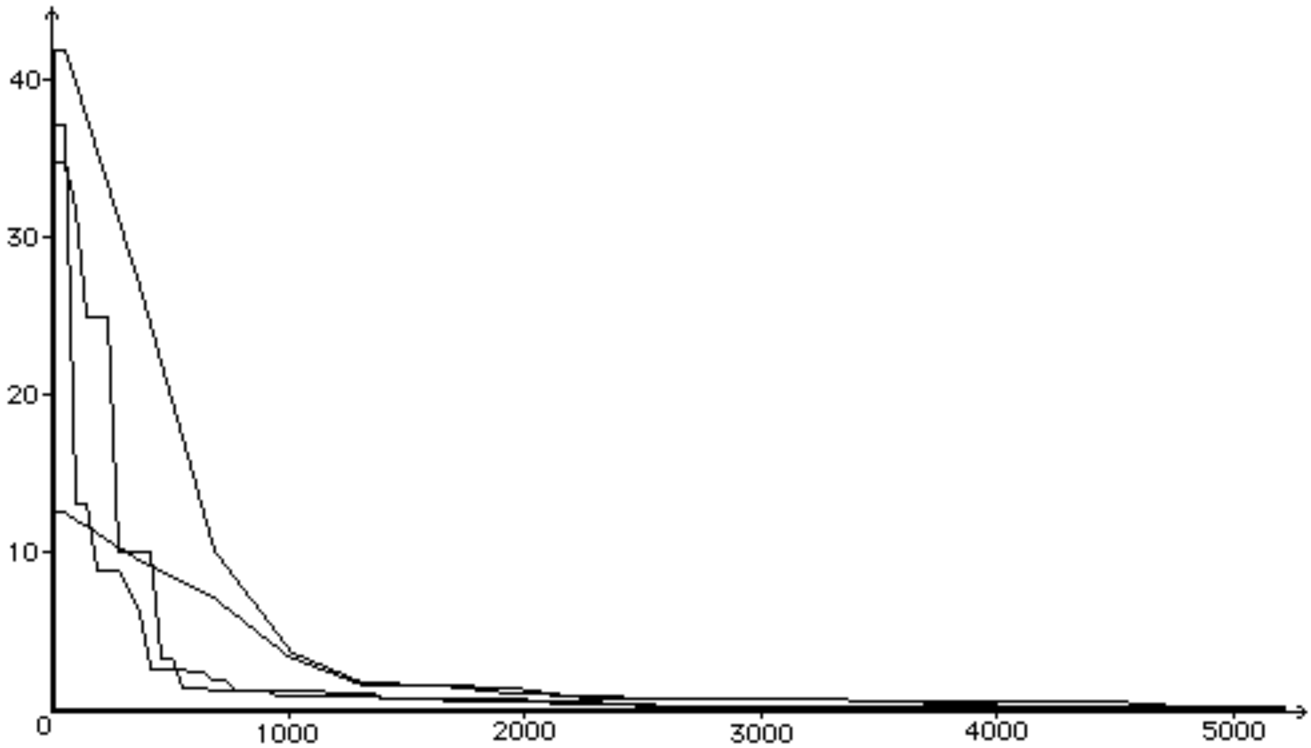

Figure 15 Fonction de Griewank du meilleur individu obtenu au cours d'une évolution, en fonction du nombre d'évaluation de ctte fonction. Les deux coubes à paliers sont obtenues pour des évolutions suivant ESAO, les deux autres pour ES(μ+7μ).

Sur l'exemple de la fonction de Griewank dont le minimum global est difficile à atteindre, on compare deux recherches avec ESAO (les deux courbes à paliers décroissant rapidement) avec deux recherches par ES(μ + 7.μ) dont les courbes sont certes plus régulières mais à décroissance moins rapide.

Résultats sur le classement des opérateurs Lors des expériences précédentes avec ESAO, nous observons maintenant le cumul du nombre de fois où un opérateur génétique d'un certain type est arrivé en tête, et donc a eu sa place renforcée dans la population des opérateurs.

crossover0 : 55727< crossover1 : 52252 < crossover2 : 34358 < migtout : 14347 < mig2 : 11441 < ajout : 10812 < mut1 : 8358 < sym : 8025 < mig3 : 657 < mut2 : 5766 < mig1 : 704

Contrairement à d'autres classes de problème (recherche de systèmes de règles floues par exemple) où les opérateurs las plus perturbants étaient les mieux notés, [Gacôgne 94, 97], ici ce sont les croisements qui arrivent en tête. Cependant les migrations, et notamment celles consistant à créer un individu entièrement nouveau, sont bien représentées.

Tests sur des problèmes de nature symbolique

Des problèmes tels que la reconnaissance d'un dessin constitué de pixels noirs ou blancs, la découverte de carrés magiques ou celui de la suite des transformations du Rubik-cube ont été étudiés dans [Herdy 90] qui aboutit à la conclusion (maintenant commune) sur la puissance d'opérateurs dédiés au codage. Ainsi en est-il des mutations locales (une valeur est remplacée par une valeur proche) ou de croisements significatifs (deux matrices, ou cubes, c'est à dire matrices à trois dimensions, échangeront respectivement deux sous-rectangles ou sous-parallélépipèdes)

Reconnaissance de figures planes 7*7 Il s'agit simplement de reconnaître sur des tableaux binaires 7*7, une figure représentant le dessin d'une lettre. La représentation d'une solution est également un vecteur de liste de chiffres, lesquelles listes sont, cette fois, fixée à sept chiffres binaires. La «fitness» dont on cherche le minimum zéro est simplement la distance de Hamming d'avec la solution. Les quatre dessins testés ne montrent pas de différence significative entre eux, par contre les stratégies employées, grossièrement classées ci-dessous se distinguent nettement.

Figure	H	E	X	A
GA μ = 500, pc = 0.5, pm = 0.1	28706	31273	50066	44475
SSGA (33%) avec élimination	18312	30864	22917	12336
ES (μ + 7μ) μ = 50 avec élimination	13635	10641	20593	9220
ESAO (μ = 50) avec élimination	9716	10917	24766	11439
ES (μ + 7μ) μ = 50 sans élimination	4040	4180	4600	4320
SSGA (33%) sans élimination	2487	2907	2700	3010
ESAO (μ = 50) sans élimination	1990	1820	1980	2170

Figure 16 Comparaison du nombre d'évaluation pour la reconnaissance d'un dessin binaire 7*7 par quatre stratégies évolutionnaires (moyennes sur 20 essais).

Remarquons que pour toutes les stratégies étudiées, l'élimination des individus voisins ralentit l'exercice, ce qui permet de dire que l'évolution a besoin, ici, de solutions voisines (et présentant de bonnes colonnes) pour accéder plus vite à la vraie solution.

Les dames de Gauss Ce problème classique de placer n dames non mutuellement en prise sur un échiquier n*n se résoud habituellement par une stratégie de construction de la solution avec retour en arrière au dernier choix laissé en attente. Pour établir une comparaison avec une stratégie d'évolution qui évalue des dispositions entièrement construites, il est maintenant nécessaire de définir une «fitness» pour ces dispositions et de mesurer les temps d'obtention pour la première solution trouvée. Afin de conserver la même représentation d'un chromosome en vecteur de listes de chiffres, une solution sera représentée par un vecteur de dimension n. La composante i (pour $0 \le i < n$) est simplement le numéro de la colonne j ($0 \le j < n$) où se trouve la dame de la ligne i. Nous adopterons comme «fitness» le nombre de paires de dames en prise, fonction dont on sait qu'elle est minimale en 0 pour presque tout n, avec en général beaucoup de solutions non isométriques.

Dimension	5	10	15	20	30	40	50
Essais avec retours en arrière	0 s	0 s	6 s	85 min	-	-	-
GA μ = 500, pc = 0.5, pm = 0.1	5 s	2 min	9 min	-	-	-	-
Nombre d'évaluations	691	8165	33524	max	-	-	-
ES (μ + 7μ) μ = 50 temps :	2 s	50 s	3 min	5 min	20 min	28 min	-
Nombre d'évaluations	404	6270	15019	31205	62322	83475	max
ESAO (μ = 50) temps :	0.5 s	30 s	50 s	3 min	17 min	23 min	29 min
Nombre d'évaluations	150	4268	5501	26231	50499	68758	94870
SSGA (33%) temps :	2 s	57 s	2 min	3,5 min	9 min	18 min	24 min
Nombre d'évaluations	351	7150	24550	29530	32088	39856	50023

Figure 17 Comparaison des temps d'exécution pour parvenir à la première solution trouvée suivant la stratégie constructive de parcours d'arborescence et quatre stratégies évolutionnaires (moyennes sur 20 essais). Les meilleurs résultats (en gris) sont obtenus avec ESAO et SSGA.

Si la première solution est obtenue instantanément avec le backtracking, pour les petites valeurs de la dimension, au delà de la dimension 15 la croissance du temps d'exécution devient très grande. En fait ces temps sont tous exponentiels mais avec une croissance beaucoup plus modérée pour les algorithmes évolutionnaires. En regardant plus attentivement les valeurs obtenues pour les entiers de 5 à 15, il semble que 10 représente un point singulier. ESAO confirme un légère amélioration par rapport à ES mais se trouve, pour ce problème, dépassé par le SSGA à partir de la dimension 20.

CONCLUSION GÉNÉRALE

Comme dans beaucoup d'autres domaines, le fait de s'inspirer de phénomènes naturels conduit d'abord à essayer de les simuler fidèlement, quoiqu'en les simplifiant, puis à s'en écarter. Il est admis depuis les débuts de l'évolution artificielle que les différentes heuristiques expérimentées sont plus ou moins bien adaptées à différents type de problème. Cependant l'idée se répand petit à petit que face à un problème quelconque, une question au moins aussi importante que celle du choix de la stratégie, est celle du codage des solutions, et que c'est ce codage qui détermine à son tour le choix des opérateurs employés.

Que cette exploration soit optimale avec un nombre de voisins aux alentours de sept et un point de vue élitiste, est tout à fait empirique. Cependant, cette exploration semble encore améliorée à la fois en cherchant des opérateurs dits de migration, très perturbants, c'est à dire favorisant l'exploration et en cherchant systématiquement à favoriser l'hétérogénéité en remplaçant par exemple les proches voisins par des individus migrants. On peut supposer que la variété des opérateurs génétiques imaginés correspond à de plus grandes possibilités d'exploration du voisinage d'un point, pour expliquer les meilleurs résultats des différentes stratégies d'évolutions expérimentées lorsqu'on multiplie les types d'opérateurs. De plus, en ne prenant pas ces opérateurs au hasard, mais suivant leurs propres performances passées, on peut laisser libre cours à l'évolution de traverser elle-même diverses phases.

La seconde conclusion de cette étude est la grande performance d'une adaptation triviale avec ou sans élimination de l'algorithme SSGA qui consiste, rappelons-le, à ce que chaque parent produise un enfant grâce à un opérateur tiré au hasard, le tiers formé par les meilleurs enfants prenant la place du tiers formé par les pires parents. Au regard du critère du nombre moyen d'évaluation de la fonction avant d'obtenir son minimum, on peut tenter une explication sur deux points. Les stratégies où la valeur du meilleur individu n'est pas décroissante au fil des générations, telles l'algorithme standard et ceux qui s'en inspire, ont des performances médiocres. Celles qui conservent toujours au moins le meilleur individu, comme ES($\mu + \lambda$) ou ESAO ont de meilleurs résultats car évitent les remontées, cependant, leur élitisme rend la valeur moyenne de la fitness sur la population, elle aussi décroissante, ce qui signifie une amélioration globale de la population alors que tous les travaux expérimentaux recommandent une très grande hétérogénéité de la population. Avec SSGA, la meilleure valeur de fitness décroit en effet, mais par contre, la valeur moyenne sur la population n'est pas nécessairement monotone, il semble donc que deux principes se dégagent : garder toujours au moins le ou quelques meilleur(s), et tout faire dans le même temps pour bouleverser la population (multiplier les opérateurs et ne pas chercher une fitness moyenne décroissante).

La perspective pour un travail futur, est donc maintenant de chercher à coupler l'idée simple du SSGA concernant le renouvellement des générations, avec, soit une simple élimination, soit l'idée du nichage, et en second lieu avec celle de la notation et du classement des opérateurs.

Références

Adachi N. Matsuo K. *Ecological dynamics under different selection rules in distributive and iterated prisoner's dilemma game,* PPSN I Lec. notes in computing science 496 p388-394 Springer Verlag, 1990

Axelrod R. *The genetic algorithm for the prisoner dilemma problem,* Genetic algorithms and simulated annealing, Morgan Kaufmann Pub. p32-41, 1987

Axelrod R. *Donnant-donnant, théorie du comportement coopératif,* Ed. Odile Jacob, 1992

Bäck T. *Evolutionnary Algorithms in theory and practice,* Oxford University Press, 1995

Bäck T. Fogel D.B. Schwefel H.P. *Handbook of evolutionnary computation,* Oxford University Press, 1997

Bäck T. Rudolph G. Schwefel H.P. *Evolutionary programming and evolution strategies : similarity and differences,* Proc. of the second conf. on evol. prog. p11-22 Evolutionary programming society, San Diego 1993

Baluja S. *An empirical comparison of seven iterative and evolutionary function optimization heuristics,* (http://www.cs.cmu.edu/~baluja) 1995

Cerf R. *Une théorie asymptotique des algorithmes génétiques,* Thèse Univ. Montpellier, 1994

Cerf R. *An asymptotic theory for genetic algorithms,* Artificial evolution n°1063, Lectures notes in computer science p37-53, Sringer Verlag, 1996

Davis L.D. *Bit climbing representatioal bias and test suite design,* Proc. 4th Conf. on GA p18-23, 1991

Davis T.E. Principe J.C. *A simulated annealing like convergence theory for the simple genetic algorithm,* Proceedings of the 4th International Conference on G.A. p174-182, 1991

Davis L. *Adaptating operator probabilities in genetic algorithm,* Proc. of the 5th Int. Conf. on GA p61-69, Morgan Kaufmann 1993

De Jong K.A. *Analysis of behaviour of a class of genetic adaptative systems*, Michigan University thesis, 1975

Delahaye J.P. *L'altruisme récompensé,* Pour la science, novembre 1992

Delahaye J.P. Mathieu P. *Expériences sur le dilemme itéré des prisonniers.* Rapport 233 du Laboratoire d'Informatique fondamentale de Lille, 1992

Delahaye J.P. Mathieu P. *L'altruisme perfectionné.* Rapport 249 du LIFL 1993

Dessales J.L. *L'ordinateur génétique,* Hermès 1996

Fogel L.J. Owens A.J. Walsh M.J. *Artificial intelligence through simulated evolution*, Willey 1966

Fogel D.B. *Evolutionary computation toward a new philosophy of machine intelligence,* IEEE Press, 1992

Fonseca C.M. Fleming P.J. *Genetic algorithms for multiobjective optimization : formulation, discussion and generalization,* Proc. of the 5th conf. on the GA and their applications 1993

Fonseca C.M. Fleming P.J. *An overview of evolutionnary algorithms in multiobjective optimization,* Evolutionary computation 3, 1993

Fonseca C.M. Fleming P.J. *Multiobjective genetic algorithms made easy : selection, sharing and mating restriction,* G.A. in engineering systems : innovations and applications, IEEE 1995

Fonseca C.M. Fleming P.J. *Multiobjective optimization,* Handbook of evolutionary computation, Oxford University Press 1997

François O. *An evolutionary strategy for global minimization and its Markov chain analysis,* IEEE Trans. on evol. computation, vol 2 n°3 p77-90, 1998

François O. Lavergne C. *Plans d'expériences pour l'évaluation d'algorithmes évolutionnaires et la constitution de classes de référence,* Rapport n°3601 INRIA, 1999

Franz *Non-linearities in genetic adaptative search,* Doctoral dissertation, Abstracts International 33 (11) 5240B-5241B University of Michigan, 1972

Gacôgne L. *Une extension floue du problème itéré des prisonniers*. Rapport Laforia 94/15, 1994

Gacôgne L. *Optimisation multicritère de contrôleurs flous par une stratégie d'évolution approchant la zone de Pareto,* Rencontres francophones sur la logique floue et ses applications, Nancy, 1996

Gacôgne L. *Eléments de logique floue,* Hermès, 1997

Gacôgne L. *Research of Pareto set by genetic algorithm, application to multicriteria optimization of fuzzy controler*, EUFIT (European Congress on Intelligent Techniques and Soft Computing) p.837-845, Aix la Chapelle, 1997

Gedjati Guessoum F. *Résolution par des heuristiques dynamiques et des algorithmes génétiques du problème d'ordonnancement de type job-shop généralisé.* Thèse Paris VI, 1994

Goldberg D.E. Richardson J. *Genetic algorithms with sharing for multimodal function optimization*, GA and their applications p.41-49 Hillsdale New Jersey, Lawrence Erlbaum ass., 1987

Goldberg D.E. *Genetic algorithms in search, optimization and machine learning,* Addison Wesley, 1989

Goldberg D.E. Richardson J. *Genetic algorithms with sharing for multimodal function optimization,* GA and their applications p.41-49 Hillsdale New Jersey Lawrence Erlbaum ass., 1987

Harik G.R. *Finding multimodal solutions using restricted tournaments selection,* ICGA n°6, 1995

Herdy M. *Application of the evolution strategies to discrete optimization problems*, Proc. of PPSN 1, 1990

Holland J.H *Adaptation in natural and artificial system.* Ann Arbor University of Michigan Press, 1975

Horn J. *Finite Markov chain analysis of genetic algorithms with niching,* Proceedings of the 5th Int. Conf. on G.A. p110-117, 1993

Horn J. Nafploitis N. Goldberg D.E. *Niched Pareto GA for multiobjective optimization,* Proceedings of the Conf. on Evolutionary computation IEEE vol 1 p82-87, 1994

Ingbert L. Rosen B. *G.A. and very fast simulated reannealing : a comparison,* Mathematical computer modelling n°16 p87-100, 1992

Jones T. Forrest S. *Fitness distance correlation as a measure of problem difficulty for genetic algorithms,* Proc. of the sixth Int. Conf. on GA p.184-192, Morgan-Kaufmann, 1995
Kallel L. Schoenauer M. *An alternative random initialization in GA,* Proc. of the 7th int. conf. on genetic algorithms p268-275, 1997
Khatib W. Fleming P. *The studGA a mini revolution ?* PPSN V Lecture notes in computing science n° 1498, p683-691 Springer Verlag, 1998
Kinnear *Advances in genetic programming,* MIT Press, 1994
Koza *Genetic programming II*, MIT Press, 1994
Krishnakumar K. *Micro-genetic algorithms for stationary and non-stationary function optimization,* Proc. SPIE International Cont. Adaptative systems Philadelphia, 1989
Korhonen Laakso *A visual interactive method for solving multiple criteria problem,* European journal of Operational Research Society, vol.32 n°7 p.577-585, 1985
Korhonen P. Wallenius J. *A multiple objective linear programming decision support system,* Decision Support System, North Holland vol 6 p.243-251, 1990
Kursawe F. *A variant of evolution strategies for vector optimization,* Proc. PPSN I p.193-197 Sringer Verlag, 1991
Michalewicz Z. *Genetic algorithms + data structures= evolution programs*, Springer Verlag 1992
Michalewicz Z. Schoenauer M. *Evolutionary algorithms for constrained parameter optimization problems*, Evolutionary computation n°4, 1996
Mitchell M. Forrest S. Holland J.H. *The royal road for genetic algorithm : fitness landscapes and GA performances,* Varela F.J. Bourgine P. Ed. 1992
Monod J. *Le hasard et la nécessité,* Seuil 1970
Munteanu C. Lazarescu V. *Evolutioary contrast stretching and detail enhancement of satellite images*, Proc. of Mendel p94-99, 1999
Nissen V.A. *A new efficient evolutionary algorithm for quadratic assignment problem,* Operations research proc. of the 21th annual meeting of DGOR p259-269, 1993
Nissen V.A. *Solving the quadratic assignment problem with clues from nature,* IEEE trans. Neural Networks, vol 5 p66-72, 1994
Rechenberg I. *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog Verlag, 1973
Talbi E.G. *A taxonomy of hybrid meta-heuristics,* Publication interne n°183 du Laboratoire d'Informatique Fondamentale de Lille, 1998 (http://www.lifl.fr)
Sawai H. Kizu S. *Parameter-free genetic algorithm inpired by "disparity theory of evolution",* PPSN V Lectures notes in computing science n°1498 p702-711 Springer Verlag 1998
Sefrioui M. *Algorithmes évolutionnaires pour le calcul scientifique, application à l'électromagnétisme et à la mécanique des fluides numériques.* Thèse Université Paris VI, 1998
Schaffer J.D. *Multiple objective optimization with vector evaluated genetic algorithms,* Proc. of the first Int. conf. on GA, p93-100, 1985
Schwefel H.P. *Systems analysis, systems design and evolutionary strategies,* System analysis, Modeling and Simulation vol.7 p.853-864, 1990
Schwefel H.P. *Evolution and optimum seeking*. Sixth generation computer technology series. Wiley, 1995
Smith R. Goldberg D. *Diploidy and dominance in artificial genetic search,* Complex systems n°6, p251-285, 1992
Srivinas N. Deb K. Multiobjective optimization using non dominated sorting in GA, Evolutionary Computation vol 2 n°3, p221-248, 1995
Steuer R.E. *Multiple criteria optimization, theory, computation and applications,* Wiley, 1986
Viennet R. Fonteix C. Marc L. *Optimisation multicritère à l'aide d'un algorithme génétique diploïde,* Evolution Artificielle, Brest, 1995
Whitley D. Kauth J. *Genitor : a different genetic algorithm,* Proc. of Rocky Mountain Conf. on A.I. p118, Denver 1988
Wolpert D. McReady W. *No free lunch theorem for search,* Research report, Santa Fe Institute (32 pages http: //www.santafe.edu), 1995
Zitzler Thiele *Multiobjective Optimization using evolutionary algorithms, a comparative study,* Lectures notes in Computing science n°1498, p292-301, Springer Verlag, 1998