COMPARAISON ENTRE PSO ET AUTRES HEURISTIQUES D'OPTIMISATION AVEC OPERATEURS IMPLICITES

L.Gacôgne * **

* LIP6 - Université Paris VI 8 rue du Capitaine Scott 75015 Paris
tel : 01 44 27 88 07 mail : Louis.Gacogne@lip6.fr
** Institut d'Informatique d'Entreprise (CNAM) 18 allée J.Rostand 91025 Evry

Mots clés : optimisation - algorithmes évolutionnaires - opérateurs génétiques - optimisation par essaims de particules

Résumé
Nous présentons une comparaison sur un certain nombre de fonctions tests, entre différentes heuristiques évolutionnaires qui ont en commun de ne pas prendre aléatoirement des opérateurs génétiques et de les appliquer au hasard, mais d'appliquer des règles de transition fixes entre génération.
- Le "space partition algorithm" SPA qui réalise une partition de l'espace en raffinant les régions où la fonction à optimiser est la meilleure et en fusionnant les autres. Les individus de la population sont alors des blocs de tailles changeantes dont l'évaluation est néanmoins moyennée aléatoirement en leur sein.
- Le "macroevolutionary algorithm" MGA inspiré par la dynamique des espèces, heuristique dont le principe est de relier les individus par des poids mesurant leurs influences mutuelles. Il y a au cours des générations, extinction ou bien application d'une sorte de crossover continu entre individus.
- L'"évolution différentielle" DE où les individus donnent naissance à de nouveaux points de l'espace de recherche grâce à une sorte de tétra-crossover.
- Le "Self organizing migration algorithm" SOMA dont l'idée principale est de déterminer des groupes à l'intérieur desquels chaque individu réalise des "sauts" vers le meilleur du groupe et en s'arrêtant sur la meilleure position rencontrée. Les classes sont modifiées à chaque génération.
- Le "particle swarm optimization" dont le principe est que chaque individu soit animé d'une "vitesse" qui va déterminer sa "position" suivante. Les règles de transition prennent en compte le meilleur individu ainsi que le meilleur de chaque groupe d'individus, groupes déterminés par une distance ou bien arbitrairement au départ.

Enfin, nous comparons ces heuristiques avec quelques algorithmes évolutionnaires classiques, notamment celui SSGA dont le renouvellement des générations est régulier.

I INTRODUCTION

Si on désire tenter un classement des méthodes de résolutions de problèmes, on peut former quatre grandes classes. Un problème n'est pas nécessairement un problème d'optimisation, mais s'y ramène très souvent, ainsi dans les jeux ou dans le fameux problème des reines de Gauss, il est toujours possible de trouver une fonction à minimiser. La difficulté résidant le plus souvent dans la formalisation : définition de la fonction, puis définition d'un codage des solutions.

1 - méthodes combinatoires, où un nombre fini d'éventuelles solutions peut être examiné.

2 - méthodes constructives (exploration d' arborescence avec retour en arrière), la méthode de l' élastique pourrait être assimilée à une méthode constructive.

3 - méthodes locales (les méthodes mathématiques classiques de descente de gradient, mais aussi des heuristiques stochastiques comme le recuit simulé)

4 - méthodes évolutionnaires ce sont des méthodes stochastiques et globales : faisant intervenir une population de points en s' inspirant de l' évolution des espèces vivantes. Parmi ces méthodes nous pouvons distinguer celles qui utilisent des opérateurs explicites (mutation, croisement ...) et celles ayant implicitement des règles de transitions.

4.1 Les méthodes évolutionnaires à opérateurs explicites

La "programmation évolutionnaire" [Fogel 66] utilise une population et sa descendance par mutations où une méthode de sélection permet d' établir des tournois, en vue de ne retenir que le meilleur dans une sous-population.

Les "algorithmes génétiques" [De Jong 75], [Holland 75], [Goldberg 89] sont apparus, 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, exercer une mutation 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 et se distinguent aussi des AG 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).

Le "steady-state algorithm" SSGA [Whitley, Kauth 88] Nous présentons plus loin une version consistant à chaque génération, à appliquer tous les opérateurs produisant μ enfants pour μ parents. Un taux, par exemple 25%, est choisi et les 25% meilleurs enfants remplacent les 25% pires parents.

4.2 Les méthodes évolutionnaires à opérateurs implicites

Nous présentons sous ce titre quelques algorithmes dont les détails suivent. Dans toute la suite, la fonction "fitness" à optimiser sera notée f, et on prend le problème de la minimisation.

II ALGORITHMES AVEC OPERATEURS IMPLICITES

Le "space partition algorithm" SPA

Cet algorithme réalise une partition de l' espace en raffinant les régions où il semble qu' il y ait plus de valeurs optimales dans le but d' optimiser une fonction f [Gautard 99]. Pour $[a, b]^{dim}$, nous décidons de diviser l' espace en blocs réguliers (ce peut être par exemple 5 dans [a, b], soit 5^{dim} blocs). Un individu est alors un bloc et chacun d' eux reçoit une évaluation. Une génération consiste alors à diviser les "bons" blocs et à fusionner les "mauvais". Pour cela, une première méthode d' évaluation consiste à prendre la moyenne de p = 10 points aléatoirement choisis dans chaque bloc et de décider d' une combinaison linéaire $am + b\sigma$ où m est la moyenne des valeurs de f pour les p points et σ leur écart-type. Si l' évaluation est meilleure que la moyenne des individus, le bloc est divisé, (raffinement), sinon, si c' est possible, il est fusionné avec un bloc ayant une face commune avec lui. Une première idée était d' utiliser la triangulation de Delaunay. Mais cette option, jointe à la difficulté d' évaluer les blocs rend l' algorithme très complexe.

Les résultats plus loin sont donnés avec l' option parallélépipédique, où régulièrement, ils sont divisés en 2^{dim} nouveaux blocs. De plus, chaque bloc-individu est évalué par la valeur de f en son centre. Malheureusement, il est tout à fait possible de laisser de "bonnes" zones pour f à l' intérieur de "mauvais" blocs, et comme nous le verrons, cette heuristique n' est pas performante au regard du critère de comparaison utilisé.

L'algorithme macroévolutionnaire MGA (μ, ρ, τ)

Cette heuristique [Marin, Solé 99] est inspirée de la dynamique des espèces et leur diversification. Les solutions candidates dans l' espace de recherche de la fonction f sont appelées espèces et sont reliées dans un réseau de poids.
Les liens entre individus déterminent le nouvel état de la génération suivante : vivant ou éteint pour chacun d' entre eux, w_{ij} mesure l' influence de x_j sur x_i en temps réel, une influence négative en termes de minimisation, signifiera une survivance.

1 Une population de μ individus est aléatoirement choisie.
2 Une matrice de "connectivité" W est mise à jour grâce à $w_{ij} = (f(x_i) - f(x_j)) / \| x_i - x_j \|$
3 A chaque génération t, si l' entrée $j = 1...\mu$ $w_{ij}(t)$ est négative ou nulle, alors $x_i(t+1) = x_i(t)$, (en ce cas, x_i est meilleur que les autres en moyenne pondérée, x_i est dite vivante) sinon, x_i est supprimé et remplacé avec une probabilité τ par un nouvel individu aléatoire, (colonisation). Dans l' autre cas, une sorte de crossover continu est appliqué, en choisissant un individu aléatoire x_k dans la population comme un point attractif, et $x_i(t+1)$ devient $x_i(t+1) = x_k(t) + \rho\lambda(x_k(t) - x_i(t))$ avec λ un nombre aléatoire dans [-1, 1].
4 La "température" τ peut être fixée ou décroissante de façon à réduire l' exploration en faveur de l' exploitation.
Comme pour le recuit simulé, une façon simple est de suivre $\tau(t) = 1 - t/ng_{max}$, ainsi τ abaissé à chaque génération.
5 aller en 2

Notons que les définitions des poids favorisent l' élimination des voisins de solutions et maintiennent les différentes "solutions" à des distances importantes. Dans leur travail [Marin, Solé 99] ne trouvent pas une influence significative pour la constante ρ. Celle-ci peut être fixée à 0.5. Ils observent une amélioration par rapport aux AG, en particulier pour la fonction de Griewank (leurs meilleurs résultats étant obtenus avec un τ linéairement décroissant durant 500 générations).
Malheureusement, cette heuristique est très lente à cause d' une complexité en μ^2 et très peu d' individus sont remplacés à chaque génération, même dans le cas d' une forte probabilité de migration (0.5 à 0.8), le nuage des points qui se dessine autour des "leaders" est trop petit pour être capable par croisements, de poursuivre l' exploration. Cette remarque sera aussi vérifiée pour SOMA, la meilleure solution est souvent très proche de la solution optimale, mais le fait de fixer mutation et croisement avec un voisin, entraîne une stagnation. Cependant, comme pour PSO, différents nuages de points sont maintenus durant toute l' évolution.

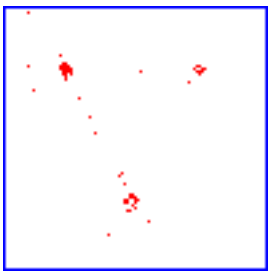

Figure 1 Nuage de points obtenu dans [-100, 100]2 pour minimiser la fonction "tripod" définie plus loin, grâce à MGA avec p = 0.3 La figure montre clairement des points alignés sur des lignesjoignant les trois optimums, lignes dues à un croisement spécial entre bonnes solutions. shows lines due to the "special" crossover between good solutions. Il s' agi de la population à la 35ième génération, on observe une trop grande concentration.

L'évolution différentielle DE (μ, p_c, χ)

Dans cette heuristique [Storn, Pice 95], [Storn 96], chaque individu peut être changé grâce à une sorte de "tetra-crossover" qui apporte une large perturbation. P est aléatoirement initialisée, et à chaque génération, pour un individu qui est un vecteur x, dont les composantes sont x_1, ..., x_{dim}, un numéro de composante k et trois autres individus x, y et z sont aléatoirement choisis tels que x, y, z, t soient distincts, alors, la mise à jour de x est x' , vecteur dont les composantes sont :

$x'_j = t_j + \chi(y_j - z_j)$ pour j = k, mais aussi pour j # k avec une probabilité p_c, sinon $x'_j = x_j$.

La mise à jour des générations est élitiste en remplaçant x par x' chaque fois que ce dernier est meilleur, c' est-à-dire f(x') < f(x). Dans l' autre cas, c' est x qui est conservé. Cet algorithme, initialement prévu pour opérer sur des variables continues a été étendu à des variables discrètes ou continues [Lampinen, Zelinka 99].

Les résultats ci-dessous sont assez bons (de plus, aucun tri de la population n' est nécessaire). Il est possible d' observer que la population se groupe de façon rectiligne autour des optimums (si p_c est bas) et se groupe très rapidement vers l' optimum global.

Des changements de p_c montrent que les meilleurs résultats sont obtenus avec 0.5, sans différence pertinente entre 0.4 et 0.6.

Le paramètre χ recommandé est 1. Notons que χ pourrait être plus grand que 1 ce qui pourrait faire sortir les individus de l' espace de recherche et d' autre part, les résultats que nous avons trouvé sont moins bons avec de plus petites valeurs pour ce paramètre.

Nos expériences montrent des résultats similaires en variant p_c, aussi, celui-ci a été fixé à 0.5. De plus, nous obtenons de meilleurs résultats avec une modification appelée "DE homogène", procédure identique mis à part que chaque composante est modifiée grâce à la probabilité p_c, le choix de k devenant inutile.

Le "self organizing migration algorithm" SOMA(μ, r, step, mass, ml, χ)

L' idée principale de cette heuristique est de définir des classes ou groupes où chaque solution appelée "particule" fait des sauts vers le leader de son groupe [Zelinka, Lampinen 2000]. Une marche aléatoire (x_1, ...x_{ml}) est créée depuis chaque individu x_0 vers son "leader" m, de telle sorte que $x_i x_{i+1}$ soit colinéaire à $x_0 m$. Un petit saut est appelé "migration loop" et est équivalent à une mutation d' un algorithme d' évolution classique.

1) P_0 est un ensemble de μ points aléatoires dans le domaine de la fonction f à minimiser (habituellement $[a, b]^{dim}$), le pire cas étant $\mu = 2$ et l' initialisation des groupes est :
2) Le meilleur individu m est choisi comme centre du premier groupe et les adhérents à ce groupe sont simplement les individus proches {x / d(x, m) < r} définis grâce à un rayon r qui détermine un domaine d' attraction du leader. Du reste des individus, est de nouveau choisi le meilleur et un autre groupe est formé en fonction du même rayon r. Ceci est répété de telle sorte que le nombre de groupes est variable mais inférieur à μ. Il est possible d' éviter qu' un individu crée seul son propre groupe par des considérations sur μ, r et le domaine $[a, b]^{dim}$.
3) Chaque particule x commence des sauts vers son leader avec une distance fixe "step" (il y a donc plusieurs positions où la fonction f est à chaque fois évaluée). Le paramètre "mass" signifie que si par exemple mass = 1, le "voyage" ou "migration loop" s' arrête sur la première position derrière le leader, si mass = 2 à la seconde etc. Le nombre de sauts est borné par le paramètre ml et la position suivante qui sera retenue pour la particule est la meilleure de la succession.
4) si non(condition d' arrêt) aller en 2 (les groupes sont donc redéfinis)
5) fin

Au point 3, il est possible de dispenser le leader du groupe de sauts, mais on observe alors une stagnation de ces leaders et il est beaucoup plus intéressant qu' ils en fassent eux-mêmes aléatoirement. Nous avons testé différentes valeurs de "mass" avant de le fixer à 5. Le problème général est celui de fixer empiriquement les paramètres de l' algorithme.
L' expérimentation montre que les groupes ne doivent pas être trop petits, par exemple, une classe avec un unique élément pourrait être stationnaire et une classe avec seulement deux éléments restreindrait l' exploration à un segment trop petit de la droite qu' ils définissent.
Si nous souhaitons avoir une connaissance empirique des paramètres, nous pouvons supposer, comme pour SPA, que μ individus sont répartis uniformément dans l' espace $[a, b]^{dim}$ et considérer qu' en moyenne 5 individus par classe font $\mu / 5$ classes par exemple. Ainsi si ν est l' effectif moyen des classes, et relativement à la distance de Hamming, si nous considérons que les μ/ν classes de diamètre 2r doivent recouvrir l' espace, nous avons une estimation de r à partir de $\mu\nu = (b - a)^{dim}$.
En fait, dans les expériences $\mu = 100$ et une valeur raisonnable ν serait au moins 5, r doit être au moins 1 pour $[-5, 5]^2$, et au moins 7.5 dans $[-10, 10]^{10}$.
Expérimentalement, nous observons la formation de nuages pour le paramètre "step" aux alentours de r / 10. Pour être plus précis, dans une application pratique, si r = 2 dans $[-5, 5]^2$, alors nous choisissons step = 0.1. Mais pour éviter la stagnation "step" peut être aussi décroissant, et nous avons testé avec succès suivant une "température" comme $step(t) = step_0.e^{-0.05t}$, afin de l' abaisser à chaque génération.
Un autre point de discussion est la perturbation apportée par la possibilité de changer "step" ou de choisir quelques nouveaux individus aléatoires avec une probabilité χ durant la "migration loop". L' essai montre qu' une assez forte perturbation grâce à $\chi = 0.5$ donne de meilleurs résultats que sans perturbation.

L'optimisation par essaim de particules PSO (μ, α, β, γ)

A travers la coopération et la compétition parmi les solutions potentielles, l' heuristique "particle swarm optimization" [Eberhart, Kennedy 95] est motivée par la simulation du comportement social. Dans cette technique, pour la génération t = 0, un ensemble P_0 de μ solutions $x_i(0)$ est aléatoirement choisi dans le domaine de la fonction f à minimiser, et chacun (chaque particule) aura une position $x_i(t)$ et une vitesse $v_i(t)$.

A chaque génération t, la "fitness" f de chaque position $x_i(t)$ est calculée, et si g désigne la position du meilleur, pour chaque particule i, on regarde dans son voisinage pour chercher le meilleur n "leader du groupe", de telle sorte que les règles de mise à jour soient :

$v_i(t+1) = \alpha v_i(t) + \beta(n - x_i(t)) + \gamma(g - x_i(t))$ sera la nouvelle vitesse :

$x_i(t+1) = x_i(t) + v_i(t+1)$ sera la nouvelle position.

Il est possible de réduire le problème du choix des paramètres en prenant :

$v_i(t+1) = \chi[\alpha v_i(t) + \beta\phi_1(n - x_i(t)) + \beta\phi_2(g - x_i(t))]$ où χ (coefficient de constriction dans]0,1], plus grand sera χ, plus lente sera la convergence) et des coefficients positifs ϕ_1, ϕ_2 aléatoirement choisis à chaque fois inférieurs à 4 [Angeline 98].
Plusieurs façons peuvent être implémentées pour déterminer le groupe de chaque particule, l' une des plus simples (géométrique) est de regarder les (par exemple nn = 5) plus proches voisins dans une population de $\mu = 100$ particules. Une autre manière (sociale) est de définir les nn "voisins" comme arbitrairement choisis dès le départ, ce peut être ceux dont l' indice (défini circulairement modulo μ) est proche. Le paramètre de définition des voisinages nn est nécessairement fonction de la taille de la population, si nn est trop grand, la dispersion demeure et s' il est trop petit, il y aura trop d' essaims.

Une troisième solution non testée ici, consiste à définir les voisinages selon un rayon σ comme |b - a]/5 par exemple si f : [a, b]^{dim} -> R est la "fitness".

On peut voir que la règle de mise à jour de chaque particule est une sorte de "tri-crossover". Cette idée est aussi présente dans la stratégie de "coralfish" des AG où le crossover n' est réalisé qu' entre un individu et un des 10% meilleurs d' entre eux.
Ici, on s' attend à obtenir des nuages autour des minimums locaux.
Comme dans toutes les heuristiques inspirées de la nature, le réglage des paramètres est une réelle difficulté. Pour les vitesses initiales par exemple, nous avons choisi de petites vitesses prenant environ 10% de la longueur de l' intervalle [a, b] comme maximum. Afin de prendre en compte cette difficulté concernant les vitesses, nous avons considéré, en dimension 2, une grille toroïdale dans l' intention de rester dans [a, b]^2. Ce paramètre de longueur maximale n' est pas très pertinent car il peut être modifié le long de l' évolution, mais concernant les coefficients α, β, γ, nous voudrions favoriser un comportement de groupe en prenant β plus grand que les autres.
Les conclusions expérimentales pourraient être très grossièrement : plus α, est grand, plus la dispersion serait grande et plus le mouvement se fait dans l' espace de recherche. Mais la convergence vers un minimum local n' est pas atteinte avec un faible γ et la formation de nuages locaux autour des minima n' est pas non plus favorisée par un grand β. L' évolution montre que si les nuages sont petits, ils peuvent rester stationnaires dans des régions de l' espace de recherche qui sont malheureusement éloignées de l' optimum de f. Les résultats sont néanmoins assez bons relativement à DE.

Notons que dans l' exemple de la fonction "tripod" définie plus loin, comme ayant trois bassins d' attractions sur [-100, 100]^2, le premier "run" parvient à l' optimum, alors que le second fait converger vers un minimum local.

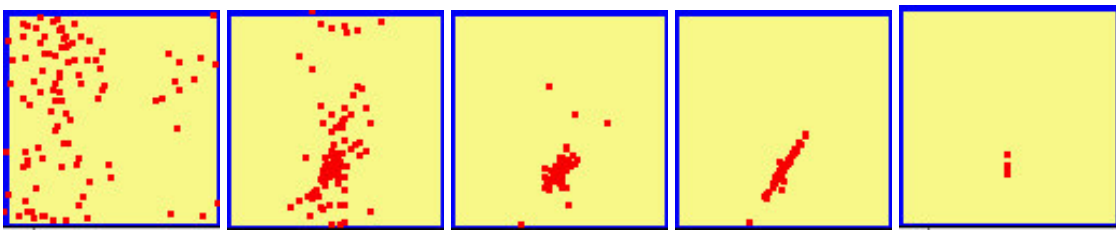

Figure 2 PSO avec nn = 5 voisins arbitrairement définis, pour les générations 3, 6, 11, 16, 35

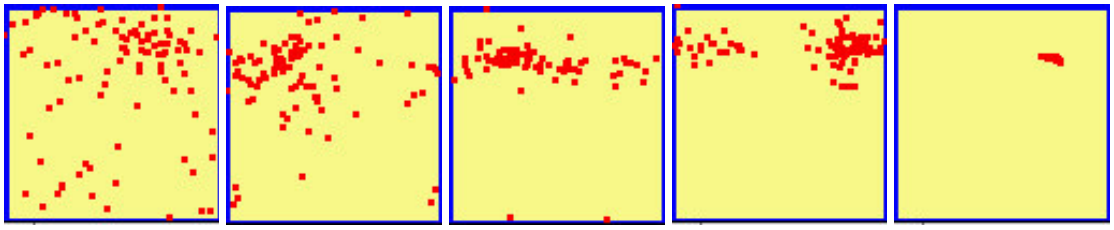

Figure 3 PSO avec nn = 5 voisins géométriquement définis, pour les générations 1, 4, 15, 30, 40.

III ALGORITHMES AVEC UNE FAMILLE D'OPERATEURS

Pour rappel, nous donnons brièvement les principaux algorithmes évolutionnaires classiques.

Standard Genetic Algorithm GA(μ, p_c, p_m)

Codage binaire, grande population, Sélection, Croisement, Mutation

1) P_0 ensemble de μ chromosomes au hasard

2) Evaluation de f sur chaque x de P_t et P_t trié

3) Sélection de chromosomes suivant leur fitness
Crossover avec probabilité p_c
Mutation sur les enfants avec probabilité p_m
Remplacement des parents par les enfants

4) Si non (condition d' arrêt) alors incrémenter t, aller en 3

5) Fin

Evolution Strategy ES(μ + λ) et ES(μ, λ)

Codage réel, petite population, pas de sélection

1) P_0 ensemble de μ chromosomes au hasard

2) Evaluation de f sur chaque x de P_t et P_t trié

3) Chaque individu x produit λ/μ enfants par des opérateurs génétiques

P_{t+1} formé des μ meilleurs parmi les λ enfants pour ES(μ, λ)

P_{t+1} formé des μ meilleurs parmi la réunion parents + enfants pour ES(μ + λ)

Steady State Genetic Algorithm SSGA(μ, τ)

Codage réel, petite population, pas de sélection

1) P_0 ensemble de μ chromosomes au hasard

2) Evaluation de f sur chaque x de P_t et P_t trié

3) Renouvellement τ meilleurs enfants μ-τ meilleurs parents

Pour ce dernier algorithme, un taux d' élimination peut être fixé de manière à remplacer tous les voisins (relativement à ce taux) au profit du meilleur. Cette technique donne d' excellents résultats, ainsi sur la même fonction ayant trois minimums locaux :

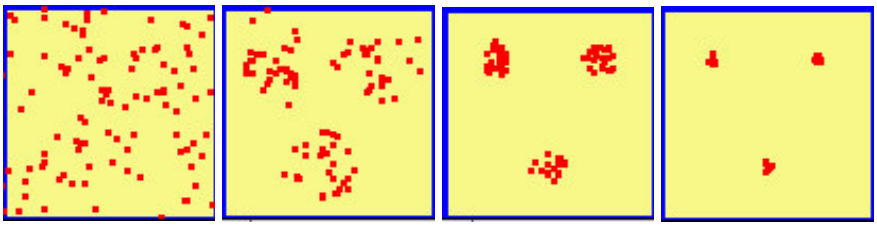

Figure 4 SSGA pour la même fonction "tripod" μ = 100 τ = 33% sans élimination pour les générations 0, 1, 4, 30

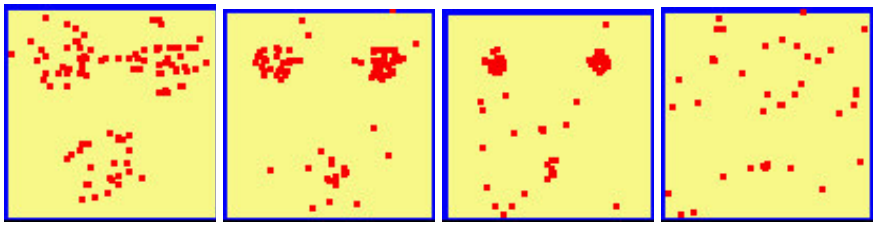

Figure 5 SSGA μ = 100 τ = 33% avec élimination pour les générations 1, 4, 9, 30. On remarque dans la dernière population qu' un petit groupe autour de l' optimum n' empêche pas une poursuite de l' exploration favorisée par l' élimination.

IV FONCTIONS TESTS UTILISEES

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' évaluations, un maximum «max» de ce nombre est fixé à 20000 (50000 en dimension 10 et 100000 pour Griewank) constituant une borne.
Pour certaines stratégies non élitistes comme ES(μ, λ), les résultats étant très dispersés, la moyenne est assez peu significative, c' est pourquoi elle a pu être portée sur 50 ou 100 expériences.

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

Paraboloïde Si $x = (x_1, x_2, ..., x_n) \in [-a, a]^n$, $f(x) = \bullet(x_i - 1)^2$ (atteindre 10^{-6})
Fonction de De Jong $F_J(x) = \bullet |round(x_i)|$ pour $x \in [-a, a]^n$
Fonctions de Griewank et de Rastrigin $F_R(x) = \bullet_{1 \bullet i \bullet n} x_i^2 + 10 - 10 \cos(2\pi x_i)$

Figure 6 Fonction de Griewank $F_G(x) = x^2/d - \cos\bullet x + 1$ en dimension 1 pour d = 400

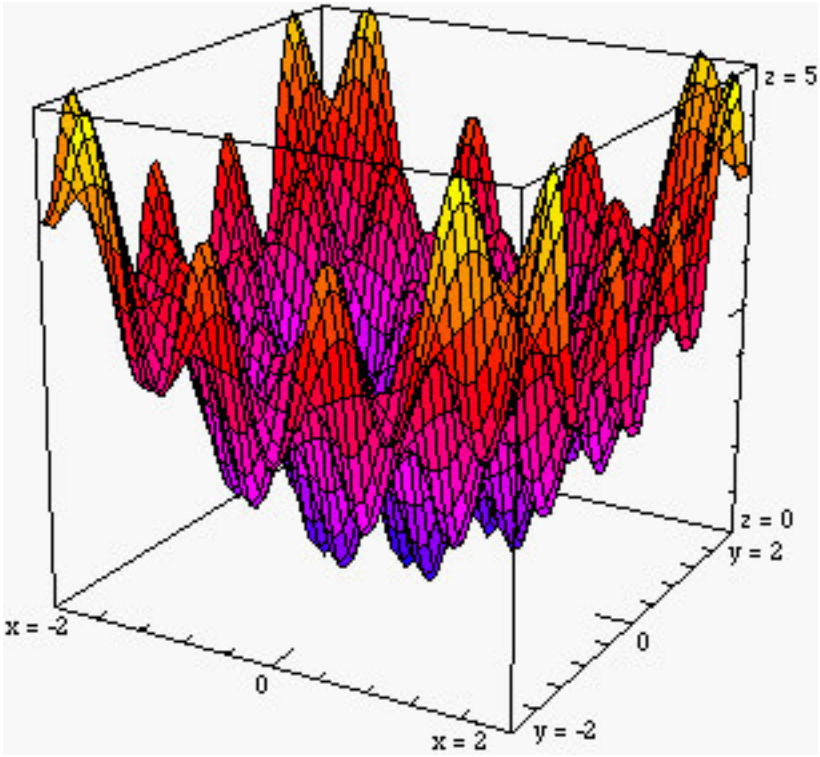

Figure 7 Fonction de Griewank $F_G(x) = (x^2 + y^2) / d - \cos 2\pi x.\cos 2\pi y + 1$ en dimension 2 pour d = 2. D' apparence analogue à la fonction de Rastrigin, la fonction de Griewank est en fait bien plus difficile à optimiser.

Une fonction à trois bassins d'attraction

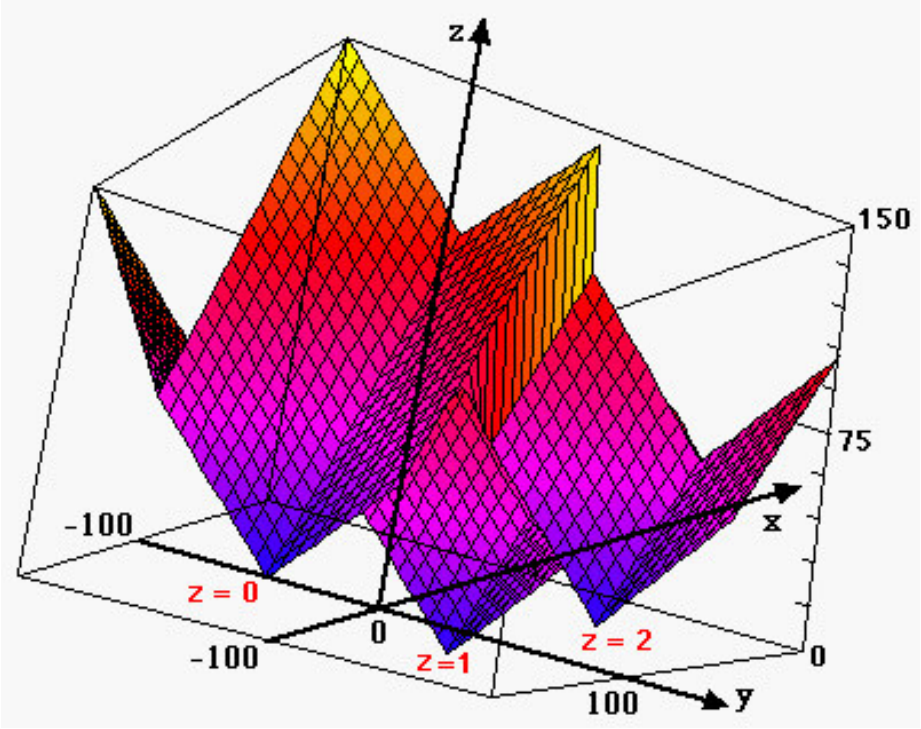

Figure 8 Fonction Tripod avec 3 minimums attractifs pour x, y ∈ [-100, 100]

f(x, y) = if y < 0 then |x| + |y + 50|
else if x < 0 then 1 + |x + 50| + |y - 50|
else 2 + |x - 50| + |y - 50|

Fonction de Rosenbrock

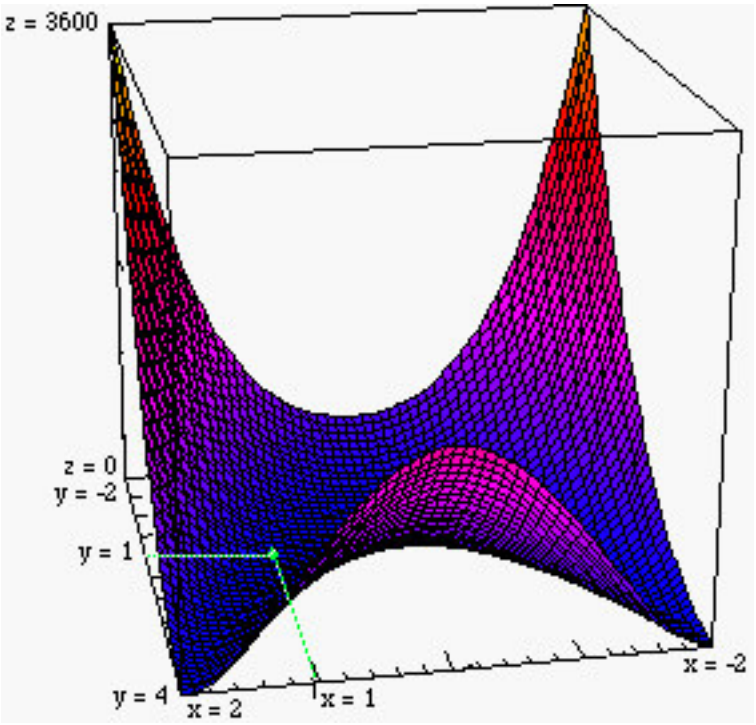

Figure 9 Fonction de Rosenbrock $F_R(x, y) = 100(x^2 - y)^2 + (1 - x)^2$ minimum 0 en (1, 1), la vallée dont les bords sont très abrupts dès que |x| > 2, est courbe et le minimum se trouve en un endroit extrêmement plat. L' orientation choisie ici, est destinée à montrer les deux branches non symétriques 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.

Comparaison entre SPA, SSGA avec nichage et SSGA

Adaptation du nichage à SSGA

Afin de garder le plus longtemps possible tous les minima trouvé de f, et en vue d' éviter une trop grande homogénéité, la population est divisée en "niches" [Horn 93]

L' idée du nichage réside dans une modification de la fonction f à minimiser, selon la densité de la population autour de chaque individu [Goldberg 87]. La distance peut être définie sur les génotypes ou bien sur les phénotypes, nous prenons la première dans l' espace $[a, b]^{dim}$.

Soit σ un rayon et $n_i = card\{j \in P / d(i, j) < \sigma\}$, une mesure de la densité autour de i. Pour minimiser f, on substitue f(i) par $f'(i) = n_i f(i)$, ce qui est un avantage pour les individus isolés quand la population est triée et tronquée.

Ainsi, si $n_1 < n_2$ et $f_1 = f_2$ alors $f'_1 < f'_2$ de plus, si $f_1 < f_2$ alors la différence décroît. D' autre part, si $f_1 < f_2$ avec $n_1 > n_2$, une inversion dans le classement peut arriver, par exemple $f_1 = 2$, $f_2 = 3$ $n_1 = 3$ $n_2 = 1$ alors $f'_1 > f'_2$.

Ceci est étendu à des voisinages flous, si sh est une fonction "de partage" sur [0, 1], vue comme une «accentuation» de la négation avec $sh(x) = 1 - x^{\alpha}$ où $\alpha < 1$ pour obtenir $sh(x) < x$ ou $\alpha > 1$ pour le cas opposé $sh(x) > x$, et $sh(x) = 0$ pour $x > 1$.

Si μ est la taille de la population et que σ déterminine les niches dans l' espace, alors f est remplacée par f' pour la génération suivante avec :

$f'(c_i) = f(c_i) . \Sigma_{1 \le k \le n} sh[d(c_i, c_k) / \sigma]$ où les individus sont c_1, c_2, ... , c_μ. En vue de minimiser f, (division en cas de maximisation) plus la population est faible autour d' un individu, meilleure sera sa "fitness" f, et une pénalité dans la configuration contraire favorise l' exploration.

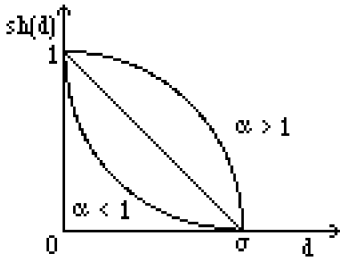

Figure 10 Fonction d' appartenance de support défini par le rayon σ.

[Goldberg, Richardson 87] ont montré qu' une stabilisation a lieu quand les valeurs des optima f(i) sont en proportion avec les effectifs flous $n_i = \Sigma_{1 \le k \le n} sh[d(c_i, c_k) / \sigma]$ des niches.

Nous avons adapté ceci au SSGA au lieu d' une habituelle élimination "clearing" mais les résultats sont pires.

Résultats avec une population d' effectif constant 100.

	SPA p = 10	SPA p = 50	SPA p =100	SSGA σ =0.5 α =0.1	SSGA σ =0.5 α =1	SSGA σ =0.5 α =10	SSGA σ =1.5 α =0.1	SSGA σ =1.5 α =1	SSGA σ =1.5 α =10	SSGA
Parabola dim 1	730	760	731	1340	1420	1302	1277	1183	1067	812
... ... dim 3	61074	61745	61880	2702	2289	2240	2718	1879	1931	1953
Rastrigin dim 1	2356	2663	2823	4266	4081	2948	3012	2760	2612	2150
... ... dim 3	52656	52309	51885	6867	12662	14269	4685	12050	1929	1172

Figure 11 Nombre d' évaluations de la fitness (moyenne sur 20 runs) pour atteindre le minimum 0 sous un seuil donné. Il s' agit de quelques résultats pour SPA avec p points pris au hasard dans chacun des 6 sous-ensembles initiaux en dimension 1 et $6^3 = 216$ sous-ensembles en dimension 3. Résultats pour un taux de 33% avec la méthode SSGA avec nichage et comparaison avec SSGA.

Comme il a été dit ci-dessus, SPA n' est pas performant et il fut impossible d' atteindre des résultats avec des fonctions plus élaborées. La dernière colonne montre qu' il n' y a aucun avantage à adapter le nichage à l' algorithme SSGA. La dernière colonne montre que nous n' avons pas avantage à mixer

nichage et SSGA pour aucune taille de blocs (σ plus grand ne donne pas d' amélioration) ni pour d' autre forme.

Tests entre algorithmes avec opérateurs implicites et SSGA

Nous montrons à présent quelques résultats sur une population de $\mu = 100$ individus. PSO fonctionne avec $\alpha = 0.3$; $\beta = 0.6$; $\chi = 0.6$ et nn = 5 dans la définition géométrique (première colonne), définition sociale dans la seconde colonne. DE fonctionne avec $p_c = 0.5$ et $\chi = 1$, la première colonne est donnée avec une procédure homogène pour toutes les composantes et la seconde avec l' algorithme décrit (une composante est toujours perturbée) ce qui semble adapté aux fonctions "difficiles".

$\mu = 100$	MGA	PSO geom	PSO social	SOMA r=(b-a)/5	SOMA (b-a)/10	DE hom	DE comp	SSGA 33%	SSGA 33% elim
De Jong dim 3	4880	3030	2060	2854	3251	2775	2450	765	853
Parabola dim 3	5261	6545	8790	6081	5304	7630	6775	1310	2183
... ... dim 10	max	max	25741	max	max	46265	49265	12880	11204
Tripod dim 2	max	14210	12685	28405	max	16120	11525	9800	11697
Rosenbrock dim 2	22658	3855	4575	32558	46059	20240	10185	20645	15939
Rastrigin dim 2	23550	12705	13225	16552	15245	10815	8395	1525	2546
... ... dim 10	max	max	max	max	max	max	max	3320	2465
Griewank dim 2	47856	95295	85725	96522	18470	9710	7110	5475	3070
... ... dim 10	max	max	max	max	max	max	max	13826	11914

Figure 12 Nombre d' évaluations de la fitness (moyenne sur 20 runs) pour atteindre un seuil donné 10^{-4}.

Conclusion

En examinant quelques heuristiques inspirées de la nature et en les comparant sur un même codage, on doit noter, en point commun, la grande difficulté à régler les paramètres de ces heuristiques. En dehors des paramètres numériques, il y a le choix des règles de mise à jour de la population lors du passage des générations. Il est plus facile lorsque les opérateurs sont explicites, car ils peuvent être mis dans une famille d' opérateurs non figée (mutations diverses, croisements divers ...) et se pose alors le problème de leur mode d' application (en suivant leur ordre, au hasard, en tentant de noter leur score...). Le renouvellement des générations est aussi une question (élitisme ou non, renouvellement régulier...). En fait, l' observation pas à pas sur des fonctions diverses, mais aussi sur des problèmes classiques (route royale, reines de Gauss, reconnaissance d' une figure...) montre très souvent que l' homogénéité de la population doit être à tout prix évitée, et que peu d' individus peuvent suffire alors que plusieurs opérateurs, vus comme des moyens d' exploration de l' espace de recherche, sont une bonne chose. Des tests antérieurs sur ces heuristiques ont montré l' efficacité de l' algorithme SSGA, particulièrement sur de faibles populations. C' est pourquoi nous l' avons retenu comme référence pour les autres méthodes étudiées. Il s' avère que beaucoup de ces méthodes aboutissent trop tôt à une convergence, c' est-à-dire un nuage de points non nécessairement autour de l' optimum global. Néanmoins, nous avons observé des résultats inégaux suivant la difficulté des fonctions et DE, et plus encore PSO réussissent bien sur les fonctions unimodales.

Références

Angeline P. *Evolutionary optimization versus particle swarm optimization : philosophy and performance differences,* Proc. of the 7-th annual conf. on Evolutionary Programming, p601-611, 1998

Angeline P. *Using selection to improve particle swarm optimization,* Proc. of the IEEE Int. conf. on Evolutionary Computation, 1998 (see also http://ics.yediteps.edu.tr/~eozcam/pso/)

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

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

De Jong K. Sarma J., *Generation Gaps Revisited*, in Foundations of G.A. Morgan Kaufmann, p5-17, 1993

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

Eberhart R.C. Kennedy J. *A new optimizer using particles swarm theory,* Proc. 6-th symposium on micro-machine and human science, IEEE 1995 (http://ics.yeditepe.edu.tr/•eozcan/pso/Fogel L.J. Owens A.J. Walsh M.J. *Artificial intelligence through simulated evolution*, Wiley 1966

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

Gacôgne L. *Benefit of a steady state genetic algorithm with adaptative operators,* Mendel 2000 p236-242, Brno 2000

Gautard-Yzquierdo V. *Optimisation automatique de formes en aérodynamique, application à la conception d'aéronefs,* Thèse Université Paris nord, Onera 1999

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

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

Herrera F. Lozano M. Gradual Distributed Real coded genetic Algorithms, IEEE transactions on evol. computation vol 4 n°1 p43-63 2000

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

Koza *Genetic programming II*, MIT Press, 1994

Lampinen J. Differential evolution - New naturally parallel approach for engineering design optimization, in Developments in computational mechanics with high performance computing. Civil Computing Press Edimburgh p217-228, 1999 (www.lut.fi/~lampinen/debiblio.hrm)

Lampinen J. Zelinka I. *Mixed integer-continuous optimization by differential evolution,* Mendel proc. p77-81, 1999

Marìn J. Solé R.V. *Macroevolutionary algorithms : a new optimization method on fitness landscapes,* IEEE Transactions on Evolutionary Computation, vol 3, n°4, p272-286, 1999

Michalewicz Z. *Genetic algorithms + data structures= evolution programs*, Springer Verlag 1992

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

Price K. Storn R. Differential evolution, Dr Dobb' s Journal 1997

Rechenberg I. *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*,

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

SolisF. Wetts J. Minimization by random search techniques, Math. of Operational Research vol 6, 1981

Storn P. Price K. *Differential evolution a simple and efficient adaptative scheme for global optimization over continuous spaces,* Global Optimization vol 11 p341-359, 1997 (www.icsi.berkeley.edu/~storn)

Storn R. *On the usage of differential evolution for function optimization,* NAFIPS p519-523, 1996

Syswerda G. A study of reproduction in generational and steady state genetic algorithm, Rawlins eds Foundations of GA p94-101 Morgan Kaufmann 1991Whitley D. Kauth J. *Genitor : a different genetic algorithm,* Proc. of Rocky Mountain Conf. on A.I. p118, 1988

Wolpert D. McReady W. *No free lunch theorem for search,* Research report, Santa Fe Institute (32 pages http: //www.santafe.edu), 1995

Zelinka I. Lampinen J. *SOMA : self organizing migration algorithm,* Mendel Proc. p177-187, 2000