Darwinisme artificiel : une vue d'ensemble.

Evelyne LUTTON

INRIA - Rocquencourt - Equipe COMPLEX - Projet FRACTALES

B.P. 105, 78153 LE CHESNAY Cedex, France

Evelyne.Lutton@inria.fr

Tel : +33 (0)1 39 63 55 23 — Fax : +33 (0)1 39 63 59 95

`http ://fractales.inria.fr/`

3 février 2004

Résumé

Les algorithmes génétiques, la programmation génétique, les stratégies d'évolution, et ce que l'on appelle maintenant en général les algorithmes évolutionnaires, sont des techniques d'optimisation stochastiques inspirées de la théorie de l'évolution selon Darwin. Nous donnons ici une vision globale de ces techniques, en insistant sur l'extrême flexibilité du concept d'évolution artificielle. Cet outil a un champ très vaste d'applications, qui ne se limite pas à l'optimisation pure. Leur mise en oeuvre se fait cependant au prix d'un coût de calculatoire important, d'où la nécessité de bien comprendre ces mécanismes d'évolution pour adapter et régler efficacement les différentes composantes de ces algorithmes. Par ailleurs, on note que les applications-phares de ce domaine sont assez souvent fondées sur une hybridation avec d'autres techniques d'optimisation. Les algorithmes évolutionnaires ne sont donc pas à considérer comme une méthode d'optimisation concurrente des méthodes d'optimisation classiques, mais plutôt comme une approche complémentaire.

1 L'évolution artificielle, la théorie de Darwin en informatique.

L'observation des phénomènes biologiques est une très riche source d'inspiration pour les informaticiens. Le concept d'algorithme génétique notamment, généralisé ces dix dernières années sous le terme d'algorithme évolutionnaire en est un excellent exemple. Les principes de base de ces algorithmes (on dit aussi "AE" pour "Algorithme évolutionnaire") sont une transposition informatique simplifiée de la très célèbre théorie de Darwin. En d'autres termes, on imite au sein d'un programme la capacité d'une population d'organismes vivants à s'adapter à son environnement à l'aide de mécanismes de sélection et d'héritage génétique. Depuis une quarantaine d'années, de nombreuses méthodes de résolution de problèmes, d'optimisation stochastique, ont été développées à partir de ces principes, pour constituer ce que l'on nomme maintenant le "Darwinisme artificiel." Le terme "algorithmes évolutionnaires" couvre en fait un ensemble de techniques, nommées algorithmes génétiques, programmation génétique, stratégies d'évolution, programmation évolutionnaire, suivant la façon dont les principes Darwiniens sont traduits dans le modèle artificiel.

L'ingrédient commun de cet ensemble de techniques est la manipulation de populations (représentant par exemple des points d'un espace de recherche) qui évoluent sous l'action d'opérateurs stochastiques. L'évolution est usuellement organisée en générations et copie de façon très simplifiée la génétique naturelle. Les moteurs de cette évolution sont d'une part *la sélection*, liée à la performance d'un individu, à une mesure de sa qualité vis à vis du problème que l'on cherche à résoudre, et d'autre part *les opérateurs génétiques*, usuellement nommés croisement et mutation, qui génèrent les individus d'une nouvelle génération (action d'exploration).

L'efficacité d'un algorithme évolutionnaire dépend du paramétrage du processus stochastique précédent : les populations successives doivent converger vers ce que l'on souhaite, c'est-à-dire le plus souvent l'optimum global de la fonction de performance. Une grande part des recherches théoriques sur les algorithmes évolutionnaires est consacrée à ce délicat problème de convergence, et à celui de savoir ce qui rend la tâche aisée ou difficile pour un algorithme évolutionnaire (notion d'AE-difficulté). Comme nous le verrons plus loin, un certain nombre de réponses théoriques rassurantes existent (oui, cela converge, si l'on respecte un certain nombre d'hypothèses), mais

d'autres question cruciales d'un point de vue pratique (vitesses de convergence, notamment) restent ouvertes. On peut cependant dire que l'intérêt et l'efficacité des algorithmes évolutionnaires en tant qu'heuristiques de recherche aléatoire a été globalement prouvée d'un point de vue théorique, confortant ainsi des constatations expérimentales vieilles d'une quarantaine d'années.

Par ailleurs, les algorithmes évolutionnaires sont des méthodes d'optimisation stochastique d'ordre 0, en effet aucune propriété de continuité ni de dérivabilité n'est nécessaire au bon déroulement de la méthode, seule la connaissance des valeurs de la fonction à optimiser aux points d'échantillonnages est requise (parfois même une approximation suffit). Ces méthodes sont donc particulièrement adaptées aux fonctions très irrégulières, mal conditionnées ou complexes à calculer. En revanche un algorithme évolutionnaire a un coût calculatoire qui peut devenir important. Cela fonde la recommandation usuelle de leur emploi dans le cas où les méthodes standards plus rapides calculatoirement (par exemple des méthodes de descente de gradient) ne sont plus applicables, du fait qu'elles se trouvent trop rapidement piégées dans des optima locaux : espace de recherche trop vaste, fonctions trop irrégulières, jeu de variables mixtes, par exemple. Nous verrons plus loin que d'autres problèmes (comme les problèmes dynamiques ou les problèmes interactifs) peuvent être traités à l'aide d'une approche évolutionnaire et qu'il est aussi souvent avantageux d'hybrider approche traditionnelle et approche évolutionnaire.

Malgré l'attirante simplicité d'un processus évolutionnaire, fabriquer un algorithme évolutionnaire efficace est une tâche difficile, car les processus évolutionnaires sont très sensibles aux choix algorithmiques et paramétriques, aux choix des représentations notamment. Le design d'un algorithme évolutionnaire efficace est difficile, et l'expérience prouve que les plus belles réussites sont fondées sur une très bonne connaissance du problème a traiter, sur beaucoup de créativité, et sur une compréhension fine des mécanismes évolutionnaires. L'emploi de ces techniques en "boîte noire," comme un "optimiseur universel," est à proscrire d'emblée.

Cela dit, les techniques évolutionnaires ne sont pas très loin de faire partie de notre quotidien, les exemples de "succes-stories" régulièrement cités sur le site du reseau Europeen EvoNet (http ://evonet.dcs.napier.ac.uk/) sont éloquents. Le champ d'application des algorithme évolutionnaire est en fait très large : il va des applications réelles complexes comme le contrôle du flux de pipelines de gaz, le design de profils d'ailes, le routage aérien ou la planification de trajectoires

de robots, à des problèmes plus théoriques et combinatoires, en théorie des jeux, en modélisation économique, en finance, en commande de processus et pour l'apprentissage (voir par exemple [32, 68, 2, 83, 45, 26, 18]).

1.1 Les sources d'inspiration : l'évolutionnisme et la théorie de Darwin

L'idée d'évolution est très ancienne, et déjà dans l'antiquité grecque, plusieurs auteurs proposent des interprétation évolutionnistes du monde, à base de phénomènes d'adaptation au milieu et de lutte pour la vie[1]. Pendant la renaissance, puis au XVIIème, émergent des hypothèses intermédiaires entre le dogme religieux des espèces créées une fois pour toute par Dieu, et une diversification des caractéristiques des espèces sous l'influence du milieu.

La première expression de l'évolutionnisme revient à un géomètre philosophe, Maupertuis, qui montre aux environs de 1750 l'importance des variations héréditaires et de la sélection. Lamack, disciple de Buffon, fut le premier à formuler une théorie de l'évolution selon 2 principes : le besoin crée l'organe nécessaire, et les caractères acquis sous l'action des conditions du milieu se transmettent de génération en génération.

La publication de l'ouvrage "Sur l'origine des espèces" par Charles Darwin en Novembre 1859 a ensuite remis en question la notion d'espèce. L'espèce était jusqu'alors considérée comme une sorte d'essence métaphysique immuable (c'est le "fixisme"), tout au plus susceptible de former des "variétés," qu'il était essentiel de classer systématiquement comme l'avait proposé Linné.

Le célèbre voyage de Charles Darwin sur le "Beagle", effectué de 1831 à 1836 à destination des côtes sud-américaines, des mers du sud, du Pacifique, de l'Australie, et de la Nouvelle-Zélande, lui a permis d'observer systématiquement de nombreuses lignées d'espèces. Les espèces endémiques des îles Galapagos notamment attirèrent son attention, lui rendant évidentes l'extrême variabilité des espèces, accentuée par l'isolement géographique, et l'impitoyable concurrence vitale due au

[1] *Ainsi dès l'antiquité, se manifestent plusieurs conceptions d'importance : hiérarchie des êtres vivants et gradation naturelle (Aristote) ; production de l'harmonie organique par le hasard et la mort (Empédocle, Démocrite, etc ...) ; lutte des vivants pour la vie (Lucrèce).*"

L'évolution des espèces. Histoire des idées transformistes. Chapitre I. L'antiquité. Jean Rostand. Librairie Hachette. 1932.

confinement. Selon lui, l'évolution se fonde sur trois principes [2] :

- partout, toujours, et de mille manières, les faunes et les flores ont varié,
- les lignées observées individuellement par voie d'élevage ou de culture, présentent d'innombrables variations de détail,
- la lutte pour la vie est si féroce et la sélection naturelle si rigoureuse que la plus infime variation utile fait triompher la lignée qui la possède.

Selon Darwin, la base de l'évolution est liée à des modifications aléatoires héréditaires au sein d'organismes d'une même espèce. Les modifications avantageuses sont adoptées tandis que les autres disparaissent par sélection naturelle : un organisme plus faible a moins de chances de trouver de la nourriture et de se reproduire dans un environnement hostile.

De nombreux biologistes et philosophes ont suivi ce courant, opposant la théorie de Darwin à celles de ses prédécesseurs, comme Linné et Lamarck, puis proposant des variations autour des notions de base du Darwinisme. Le mutationnisme, par exemple, conçu par Hugo de Vries (1848-1935), propose une variation discontinue à base de mutations inspirée des lois de Mendel, et inscrites dans le patrimoine héréditaire. Le mutationnisme et d'autre évolutions du modèle Darwinien, ont abouti vers 1930 à ce qu'on appelle la théorie syntétique de l'évolution, où, couplée à d'autre phénomènes complexes la sélection naturelle joue un rôle essentiel.

L'évolutionnisme, s'est aussi développé en sociologie et en ethnologie dans le but, parfois controversé, de trouver des modèles du développement des sociétés (du plus simple au plus complexe, notamment). Nous n'entrerons pas ici dans les querelles de l'évolutionnisme, de l'inné et de l'acquis, il nous suffit ce savoir que les trois principes darwiniens constituent un modèle d'évolution, qu'il est intéressant d'étudier d'un point de vue informatique.

1.2 Le Darwinisme vu par les informaticiens

La transposition des principes d'évolution Darwiniste en technique d'optimisation stochastique globale est née indépendamment des deux côtés de l'océan Atlantique il y a une quarantaine d'années. L'idée maitresse en est l'imitation du phénomène d'apprentissage collectif – ou d'adaptation – d'une population naturelle. Deux courants ont évolué parallèlement jusqu'à il y a une

[2] extrait de La Grande Encyclopédie Larousse, 1973

quinzaine d'années, chacun ayant son champ d'application propre. La récente fusion de ces différents courants sous le terme d'algorithme évolutionnaire est corrélée à leur attrait actuel pour les chercheurs et les industriels, grâce notamment à la vulgarisation des calculateurs parallèles, à l'accroissement des puissances de calculs, et aussi à la diffusion d'utilitaires de programmation évolutionnaire (GA-lib `lancet.mit.edu/ga/`, EO `eodev.sourceforge.net/`, EASEA `www-rocq.inria.fr/EASEA/`).

Le courant américain, initialisé par Holland dans les années soixante, est à l'origine de ce que l'on appelle les *Algorithmes Génétiques* (AG) [37]. Bien qu'ils aient été prévus initialement dans le cadre d'optimisation ou d'adaptations dans le domaine discret, les AG ont été facilement étendus à l'optimisation sur des domaines continus [41].

En Allemagne, sont apparues à peu près en même temps des méthodes appelées *Stratégies d'Evolution* (SE) (Rechenberg [65] puis Schwefel [70], voir [35]). Ces méthodes étaient au contraire prévues pour explorer des domaines continus, et ont été étendues à des applications en optimisation discrète [70].

Le principal reproche que l'on a pu faire aux méthodes évolutionnaires, en comparaison avec d'autres méthodes d'optimisation plus classiques, est le manque de résultats théoriques généraux de convergence. En effet, la modélisation de ces processus stochastiques à base de populations est beaucoup plus ardue, et il a fallu attendre assez longtemps pour voir établir des théorèmes de convergence convaincants (voir section 3).

Les algorithmes génétiques sont encore aujourd'hui les techniques les plus connues de ce domaine, grâce notamment aux ouvrages de David Goldberg. En fait, la publication du livre de John Holland en 1975 (*Adaptation in Natural and Aritficial Systems*) marque la véritable date de naissance des AG, ou du moins la date où ils sont devenus "publics". Ce livre est issu d'un certain nombre d'idées et de concepts sur lesquels il a travaillé à partir des années 60 [36]. Quelques-uns de ses élèves à l'université du Michigan ont pris la suite (Bagley [8], Caviccio [13], Rosenberg [67]). Un historique plus complet sur les AG peut se trouver dans [26], ou [17].

Jusque dans les années 80, l'intérêt pour ces méthodes était plutôt anecdotique, car il n'existait assez peu d'applications réelles. Au cours de cette période, un grand nombre d'études ont porté sur la représentation binaire de longueur fixe dans le cadre de l'optimisation de fonctions (De Jong

[41], Hollstien [38]) ou sur l'analyse des effets de différentes stratégies de croisement et de sélection sur les performances des AG. De Jong s'est intéressé plus particulièrement aux mécanismes adaptatifs, pour cela il a étudié les comportements des AG sur un certain nombre de fonctions, qui sont à la base de jeux de fonctions-test largement répandus actuellement (voir par exemple `www.geatbx.com/docu/fcnindex.html`). Il a ainsi étudié l'influence de la continuité, de la dimensionnalité, de la modalité, de l'aspect aléatoire et de la convexité des fonctions à optimiser sur les performances des AG.

Dans les années 80 sont apparues de nombreuses applications dans des domaines très divers. Cette diversification des applications a occasionné de nouvelles étude concernant la robustesse et la spécialisation des opérateurs génétiques. Les travaux de Goldberg, et notamment ses applications au contrôle de pipelines de gaz en sont un exemple maintenant classique [27].

Parallèlement à des recherches théoriques dont les bases les plus solides (les modèles Markoviens) n'ont été posées que relativement récemment, on s'est intéressé assez vite à d'autres adaptations des paradigmes naturels. Cela a donné naissance à de nouveaux modèles d'algorithmes évolutionnaires : implantation de phénomènes biologiques plus évolués comme le partage des ressources et les *niches écologiques* (sharing : voir section 4.1), la *co-évolution* de différentes espèces (notions de prédateurs) [12, 33, 40, 43, 46, 47, 5, 74], ou la *dominance* et la *diploidie* [30, 77]. Dans ce dernier cas, les représentations chromosomiques (les codes) contiennent l'information en double (représentation diploide). Cette particularité permet semble-t-il de rendre les algorithmes plus robustes, particulièrement en environnement dynamique, grâce à cette redondance d'information qui lui donne une sorte de "mémoire" à long terme.

David Goldberg a été très créatif dans ce domaine, en proposant différents modèles évolutionnaires : par exemple les *AG "désordonnés"* (messy : mGA) qui travaillent sur des codes de longueurs variables, grâce à des opérateurs génétiques plus "souples" [28]. Cette approche est fondée sur la constatation qu'un facteur important de l'adaptation naturelle est lié a la souplesse de la représentation de l'information. D'autres approches plus récentes comme les algorithmes à estimation de distributions, correspondent à une interprétation plus statistique et plus globale des opérateurs génétiques (travaux de Muhlenbein, [60], Pelikan et Goldberg [63], Larranaga [52]).

Enfin, de nouveaux thèmes de recherche se sont ouverts progressivement, en s'éloignant quelque

peu des préoccupations d'optimisation numérique pure : programmation génétique, vie artificielle, optimisation multi-objectif, contrôle et suivi de données dynamiques, classification, apprentissage.

1.3 L'importance des chromosomes

Comme nous venons de le voir, les algorithmes évolutionnaires tirent une grande partie de leurs origines des recherches menées sur les populations d'organismes vivants, tant du point de vue génétique qu'au point de vue comportemental.

Par exemple, si l'on s'intéresse à des phénomènes comportementaux comme l'altruisme, le sacrifice d'un animal pour sauver ses petits est moins directement explicable par la théorie de Darwin. En effet, si la sélection naturelle tendait à lutter contre les facteurs qui risquent de dégrader l'organisme, des comportements comme celui que nous venons de décrire ne pourraient persister au sein d'une population. En posant l'hypothèse que la sélection naturelle fonctionne à un autre niveau, celui des chromosomes, il devient possible de modéliser de tels comportements. Cela signifie qu'un animal qui risque sa vie pour ses petits est motivé par la conservation de ses gènes (les jeunes qui portent les gènes des parents ont plus de chance de se reproduire) plutôt que par la survie de sa propre personne.

En considérant que les gènes conditionnent les motivations des individus, la volonté de vivre n'est plus une réaction de protection de l'organisme, mais un mécanisme qui augmente les chances pour les gènes d'être reproduits. En fait, si le concept d'évolution fondée sur les gènes semble correct, il reste un bon nombre de questions en suspens, voir par exemple [17].

D'un point de vue informatique, le parallèle entre chromosome et codage de l'information s'est aussi révélé puissant. Il est bien évident qu'un codage ne change rien, intrinsèquement, à un problème d'optimisation. Cependant, si l'espace de recherche possède une structure particulière, il semble intéressant de "réordonner" l'espace de recherche de façon à pouvoir exploiter cette structure (certaines corrélations par exemple). La question est donc de trouver quel est le codage ou la représentation qui permet de mieux "capturer" cette structure sous-jacente. L'attention accordée par les chercheurs au problème de représentation aussi bien d'un point de vue expérimental que théorique trouve là encore une justification biologique. La théorie des Schémas par exemple a élaboré un formalisme simple qui permet dans une certaine mesure de conclure sur la difficulté, au

sens “génétique” du terme, des fonctions à optimiser (les fonctions GA-difficiles, aussi appelées fonctions déceptives).

Il faut aussi préciser que les études statistiques sur la génétique des populations et sur l’hérédité, ont largement contribué à l’émergence des algorithmes évolutionnaires. Notamment la théorie de Mendel sur la génétique des populations (1918) [44] a inspiré un bon nombre de chercheurs en génétique de la première moitié du XXème siècle. C’est une lourde “hérédité” que portent les algorithmes évolutionnaires, notamment au niveau du vocabulaire, dont l’aspect résolument “génétique” (et non mathématique) peut parfois choquer ...

Il est clair qu’étant donné la diversité des opérateurs et des stratégies que l’on peut employer au sein d’un tel algorithme, une définition trop restrictive des algorithmes évolutionnaires serait réductrice. A tel point que l’on a parfois du mal à reconnaître ce qu’il y a de “génétique” dans une implantation informatique. Ainsi, il serait plus exact de parler de “philosophie” plutôt que de “définition” en ce qui concerne les AG et les AE en général (“Zen and the Art of Genetic Algorithms” [25]).

Pour résumer, les algorithmes évolutionnaires ont emprunté à la génétique naturelle (et simplifié !) un certain nombre de principes sous-jacents. Ces emprunts méthodologiques à la biologie se sont assez naturellement assortis d’emprunts de vocabulaire. Ainsi dans un algorithme évolutionnaire des *individus* représentent des solutions ou des points de l’espace de recherche. Cet espace de recherche est appelé *environnement*, c’est sur cet environnement que l’on cherche à maximiser une fonction appelée *fonction d’évaluation* ou *fitness*.

Usuellement, on représente les individus par des codes (réels, binaires, de taille fixe ou variable, simples ou composés), ce sont des *chromosomes* ou des *génotypes*, les solutions correspondantes (les vecteurs de l’espace de recherche) étant des *phénotypes*. L’algorithme évolutionnaire fait évoluer sa population de façon à *adapter* les individus à l’environnement, ce qui se traduit par une *maximisation* de la fonction d’évaluation sur les individus de la population.

2 Les outils évolutionnaires de base.

Les éléments d'un algorithme évolutionnaire "canonique" peuvent être décrits simplement, mais ce que nous présentons ci-dessous doit être vu comme une "recette". Les applications efficaces à base d'algorithmes évolutionnaires sont évidemment plus complexes, le problème essentiel étant d'adapter, de créer même, les opérateurs répondant aux spécificités du problème. De même qu'une recette d'un livre de cuisine nécessite d'être adaptée avec finesse au matériel et ingrédients disponibles, aux goûts des convives, pour être vraiment appréciée (!)

2.1 La boucle évolutionnaire.

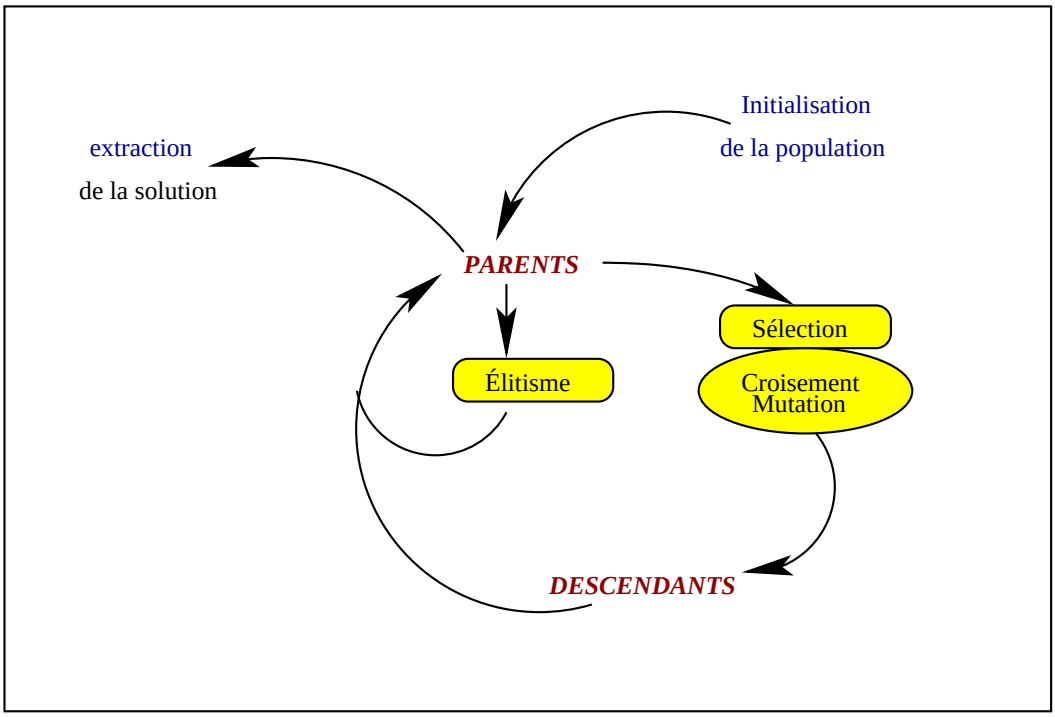

FIG. 1 – Organigramme de l'algorithme évolutionnaire simple

Le premier point est une boucle générationnelle de populations d'individus (représentés sous forme discrète ou continue, à l'aide de chromosomes ou gènes) correspondant chacun à une solution au problème considéré (voir [19] [26], [58]). Cette boucle est représentée dans la figure 1, ses étapes principales sont les suivantes.

2.1.1 L'initialisation

Elle est usuellement aléatoire (diverses stratégies sont d'ailleurs envisageables pour échantillonner correctement un espace de recherche complexe ou de grande dimensionnalité). C'est là que l'on peut injecter la ou les solutions initiales au problème que l'on a pu obtenir par ailleurs (par exemple à l'aide d'autres techniques de résolution). Si l'initialisation a théoriquement peu d'importance (on converge en limite toujours vers l'optimum global), on constate expérimentalement que cette étape influence énormément la variance des résultats (et aussi la rapidité d'obtention de ceux-ci !). Il est ainsi souvent très avantageux d'injecter le maximum de connaissances a priori sur le problème par le biais de l'initialisation : placer dans la population initiale des individus que l'on sait proches par différents aspects de l'optimum recherché ne peut qu'accélérer la convergence de l'algorithme.

2.1.2 La sélection

Cet opérateur a pour rôle de détecter quels individus de la population courante seront autorisés à se reproduire (les "parents"). La sélection est fondée sur la qualité des individus, estimée à l'aide d'une fonction, nommée "fitness," "fonction d'évaluation," ou encore "performance." Dans le schéma canonique de l'AG "à la Goldberg", 2 parents donnent 2 enfants, ainsi on sélectionne un nombre de parents égal au nombre d'enfants désirés, mais évidemment bien d'autres schémas moins conventionnels peuvent être programmés (2 parents pour 1 enfant, n parents pour p enfant, etc ...). Le paramètre principal de cette étape de sélection est ce que l'on appelle la *pression sélective* qui correspond globalement au quotient de la probabilité de sélection du meilleur individu sur la probabilité de sélection de l'individu moyen de la population courante. Ce paramètre, comme nous le verrons plus loin gère la rapidité de concentration de la population autour de son meilleur individu.

2.1.3 La reproduction

Les parents sélectionnés sont utilisés pour générer des descendants. Les deux opérations principales sont le croisement, qui combine les gènes de 2 parents, et la mutation qui consiste en une légère perturbation du génôme. Ces opérations sont appliquées aléatoirement, à l'aide de deux pa-

ramètres, la probabilité de croisement p_c et la probabilité de mutation p_m. Ces probabilités sont des paramètres très importants, qui influent de façon considérable sur la qualité des résultats globaux (convergence et qualité des résultats).

2.1.4 L'évaluation

Cette étape consiste à calculer (ou estimer) la qualité des individus nouvellement créés. C'est là, et uniquement là qu'intervient la fonction à optimiser. Aucune hypothèse n'est faite sur la fonction elle-même, excepté le fait qu'elle doit servir de base au processus de sélection (elle doit pouvoir permettre de définir une probabilité ou au minimum un ordonnancement des solutions).

2.1.5 Le remplacement

Le remplacement gère la constitution de la génération $n+1$. Il a été prouvé (expérimentalement et théoriquement) que la stratégie simpliste qui consiste à remplacer l'ensemble des parents par tous les descendants produits est assez inefficace pour des applications d'optimisation. Le maintien d'un certain taux d'élitisme est nécessaire, tout simplement pour ne pas perdre la mémoire des bons individus visités. Les stratégies usuelles de remplacement consistent à transmettre directement un pourcentage des meilleurs individus de la population courante dans la population suivante (De Jong [41], par exemple, emploie un paramètre qui gère le pourcentage de renouvellement de la population, le "Generation gap"). Cette proportion de remplacement est un paramètre essentiel du comportement de convergence de l'algorithmes évolutionnaires.

Par exemple les stratégie d'évolution (μ, λ) et $(\mu + \lambda)$ [7, 34, 35] signifient que λ descendants sont générés à partir d'une population de μ individus. La stratégie "," consiste à gérer l'élitisme par le biais de la différence $\mu - \lambda$ (on garde les $\mu - \lambda$ meilleurs individus de la population courante et on complète par les λ descendants), tandis que la stratégie "+" est une version plus adaptative dans sa gestion de l'élitisme : à partir d'un ensemble intermédiaire de taille $\mu + \lambda$ constitué des μ individus de la population courante et des λ descendants, on sélectionne les μ meilleurs individus de la génération suivante.

Il est aussi possible (dans le cas des implantations parallèles, notamment) de s'affranchir de la notion de génération, par un *schéma stationnaire*, consistant à produire à chaque itération de

l'algorithme un ou deux nouveaux individus et à les injecter dans la population courante via une opération de remplacement, ou plus exactement une "sélection inverse" (déterministe ou aléatoire) pour remplacer les plus mauvais individus de la population par des nouveaux venus.

2.1.6 L'arrêt

Stopper le processus au bon moment est essentiel du point de vue pratique. Si l'on a peu ou pas d'informations sur la valeur-cible de l'optimum recherché (ce qui autorise un arrêt dès que cette valeur est atteinte par le meilleur individu de la population courante), il est délicat de savoir quand arrêter l'évolution. En l'absence de toute information, une stratégie couramment employée consiste à stopper l'algorithme dès qu'un nombre maximal d'itérations est atteint, ou qu'un stade de "stagnation" est identifié. Il est aussi possible de tester la dispersion de la population. Il est évident qu'une bonne gestion de l'arrêt de l'évolution contribue de façon importante à l'efficacité de la méthode, et intervient au même titre que le réglage des principaux paramètres de l'algorithme (taille de population, probabilités de croisement et de mutation, pression sélective, proportion de remplacement).

2.2 Représentations et opérateurs génétiques

Les opérateurs génétiques dépendent directement du choix de la représentation, c'est en fait cet aspect qui distingue les divers courants d'algorithmes évolutionnaires (algorithmes génétiques, stratégies d'évolution, programmation génétique, grammatical evolution, etc ...). Nous présentons rapidement ci-dessous les représentations, les opérateurs, les stratégies de sélection et de remplacement les plus classiques, mais on pourra trouver dans la littérature bien d'autres déclinaisons de ces composantes dans des espaces de recherche non standard, en fonction des différentes applications (espaces de listes, de graphes, . . .)

2.2.1 Algorihtmes Génétiques et représentation discrète

Les algorithmes génétiques sont initialement fondés sur l'emploi d'une représentation binaire des solutions, assez rapidement étendue ensuite à une représentation discrète[3].

Chaque individu de la population est représenté par une chaîne de longueur fixe, dont les éléments (gènes ou allèles) sont choisis dans un alphabet fini. Cette représentation sied évidemment bien à des problèmes combinatoires discrets mais des problèmes continus peuvent aussi être abordés ainsi par le biais d'un échantillonnage de l'espace de recherche. Dans ce dernier cas la longueur du chromosome est un paramètre important, qui contrôle la précision de l'échantillonnage de l'espace, et par voie de conséquence la précision sur le résultat [84].

Les implantations sur code binaire, c'est-à-dire où l'aphabet est $0, 1$, sont les plus courantes encore actuellement. Holland [37] a argumenté l'optimalité, selon la théorie des Schémas, de l'aphabet binaire (alphabet le plus petit possible), ce qui a poussé de nombreux programmeurs à adopter ce codage quasi systématiquement. Cette stratégie a depuis été de nombreuses fois remise en question. Goldberg [26], par exemple, introduit la notion de *complexité* d'un code, et explique que d'un point de vue pratique, le choix d'un code est un compromis entre taille de l'alphabet et complexité du codage. Cette notion de complexité pour un code correspond au fait que de petites variations du code peuvent ou non induire de grandes modifications sur les solutions associées. En d'autres termes, cela traduit la corrélation des tailles des voisinages au niveau du génotype et au niveau du phénotype. Une autre contrainte importante au niveau du codage qui intervient aussi dans ce compromis et que l'on oublie parfois, est la complexité algorithmique du processus de codage/décodage. En effet ces opérations sont effectuées un grand nombre de fois au cours de l'évolution de l'algorithme.

Intuitivement, l'opérateur de croisement permet une concentration de la population autour des "bons" individus. La probabilité de croisement (qui décide si les gènes de deux individus sont mélangés ou transmis sans modifications aux descendants) règle un dosage subtil des composantes d'exploration et d'exploitation de l'algorithme. La plupart du temps le choix de cette probabilité est fait empiriquement.

[3]Même s'il existe actuellement des algorithmes génétiques à codage réel, le codage discret est la caractéristique historique du courant "algorithme génétique"

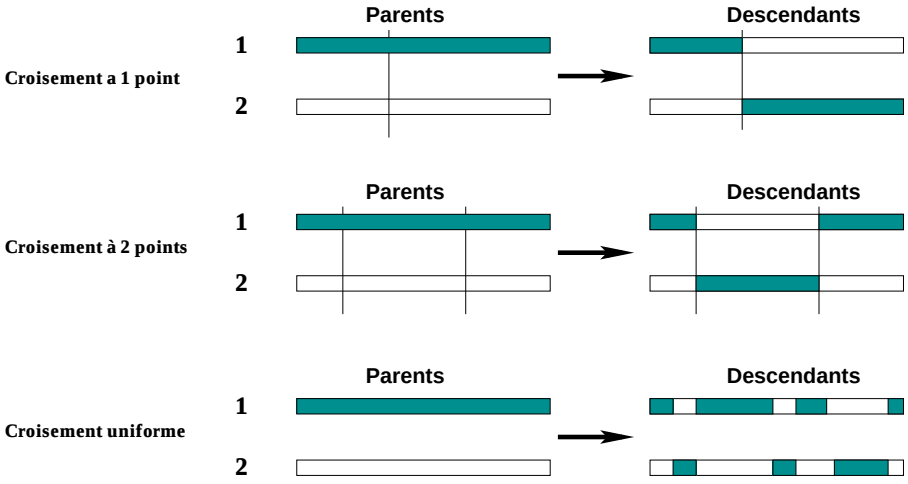

FIG. 2 – Croisements binaires

Les opérateurs de croisement les plus classiques dans le cadre de l'optimisation stochastique sont décrits dans la figure 2. *Le croisement à un site* consiste à choisir aléatoirement un point sur le chromosome, et à échanger les chaînes de code autour de ce point. *Le croisement à deux sites* de même effectue le même échange de portions de codes, mais autour de deux points de croisement. Enfin, *le croisement uniforme* est une généralisation multi-site des deux précédents croisements : chaque gène d'un descendant est choisi aléatoirement parmi les gènes des parents ayant la même position dans le chromosome (un second descendant est construit en prenant les choix complémentaires du premier).

D'autres croisement spécialisés existent, comme dans le cas du problème du voyageur de commerce ou des problèmes d'ordonnancement, qui tiennent compte de la structure particulière du codage employé.

De façon schématique, la mutation effectue une perturbation mineure du chromosome de l'individu, par exemple dans le cas d'un codage binaire, un site de mutation est choisi aléatoirement, et le bit correspondant est inversé, voir figure 3.

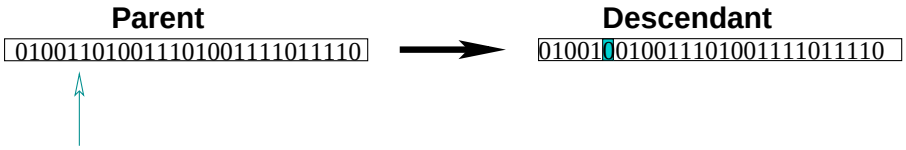

FIG. 3 – Mutation binaire

L'effet de cet opérateur est de "contrarier" l'attraction exercée par les meilleurs individus de façon à laisser la population explorer d'autres zones de l'espace de recherche. Il a été prouvé que cet opérateur limite la "dérive génétique" liée à la sélection élitiste (voir par exemple [30]).

La probabilité de mutation p_m est habituellement très faible et fixée tout au long de l'évolution de l'AG. Des schémas génétiques fondés sur une probabilité de mutation variable, qui décroît au fur et à mesure de l'évolution de l'AG existent cependant. D'un point de vue théorique, il a été prouvé qu'un AG converge vers l'optimum global de son espace de recherche pour une taille de population finie, et pour une probabilité de croisement fixée, si sa probabilité de mutation $p_m(k)$ décroît à chaque génération en respectant la borne minimale[20] : $p_m(k) \geq \frac{1}{2} * k^{-\frac{1}{M*L}}$, où M est la taille de la population et L est la longueur des chromosomes[4]. En pratique, une décroissance (plus rapide que le modèle théorique) de p_m permet d'améliorer l'efficacité de l'AG en permettant dans les premières générations une exploration large de l'espace de recherche, puis une convergence plus rapide lors des dernières générations de l'AG (ce qui correspond à une variation graduelle des capacités d'exploration et d'exploitation de l'algorithme).

2.2.2 Stratégies d'évolution et représentation continue

La représentation continue, ou représentation réelle, est historiquement liée aux approches de type stratégies d'évolution, elle a été ensuite étendue aux algorithmes génétiques. Dans cette approche, la recherche s'effectue dans $\mathbb{R}^n$ ou une partie de celui-ci. Les opérateurs génétiques associés à cette repésentation sont soit des extensions continues des opérateurs discrets, soit directement des opérateurs continus.

Les *croisements discrets* consistent à mélanger les gènes réels d'un chromosome, sans toucher à leur contenu, on peut ainsi adapter directement les opérateurs de croisement binaires précédents, par exemple les croisements à un point, plusieurs points ou les croisements uniformes.

L'avantage de la représentation continue est certainement mieux exploitée avec des opérateurs spécialisés, dits *croisements continus* qui mélangent plus intimement les composantes des vecteurs

[4]Bien sûr une telle formule de décroissance est peu intéressante en pratique ; un calcul de la formule précédente pour une population de 100 individus ayant des chromosomes de longueur 32 nous donne une valeur de probabilité de mutation qui décroit de 0.5 à 0.499 en 1000 generations !

parents pour fabriquer de nouveaux individus : *croisement barycentrique*, encore appelé *croisement intermédiaire* ou *arithmétique*, qui produit un descendant x' à partir d'un couple de vecteurs (x, y) de $\mathbf{R}^n$ grâce au tirage aléatoire d'une constante α, usuellement choisie de façon unifome dans $[0, 1]$ ou $[-\epsilon, 1 + \epsilon]$ pour une version étendue du croiement (BLX-ϵ), tel que :

$$\forall i \in 1, .., n, \quad x'_i = \alpha x_i + (1 - \alpha) y_i$$

La constante α peut être tirée une fois pour toute pour l'ensemble des coordonnées de x', ou tirée indépendamment pour chacune de ses coordonnées (variante dite "uniforme" de ce croisement).

La généralisation de ce croisement à un croisement à plus de 2 parents, voire même à l'ensemble de la population (croisement "global") est assez directe [71].

Beaucoup d'opérateurs de mutation ont été proposés et expérimentés pour la représentation réelle, nous donnons ci-dessous les plus classiques.

- La *mutation Gaussienne* consiste à ajouter un bruit Gaussien aux composantes du vecteur-individu concerné, ce qui implique l'ajustement d'un paramètre supplémentaire, σ, la déviation standard de ce bruit :

  $$\forall i \in 1, ..., n, \quad x'_i = x_i + N(0, \sigma)$$

  L'ajustement de σ est relativement complexe (trop petit, il ralentit l'évolution, trop grand, il perturbe la convergence de l'AE), de nombreuses stratégies ont été proposées, consistant à rendre ce paramètre variable au cours de l'évolution, soit en fonction du temps, de la valeur de fitness, dépendant des axes de l'espace de recherche (mutations non isotropes), ou encore auto-adaptatif, comme ci-après. Des études ont aussi été conduites sur l'emploi de bruits non Gaussiens.
- La *mutation Log-normale auto-adaptative* est une des grandes innovations des stratégies évolutionnaires, et consiste à faire gérer le paramètres σ directement par l'AE, en l'intégrant au code génétique. Les individus sont donc des vecteurs (x, σ), et la mutation se traduit simplement comme suit :

$$\forall i \in 1, ..., n, \quad \sigma_i' = \sigma_i exp(N(0,\tau) \quad et \quad x_i' = x_i + N(0,\sigma_i')$$

De nombreux travaux théoriques et expérimentaux ont montré l'intérêt et la puissance de ces méthodes auto-adaptatives.

– La *mutation uniforme*, inspirée des approches AG consiste à tirer uniformément la nouvelle valeur de la composante x_i de l'individu x dans un intervalle Min_i, Max_i, c'est un opérateur plus "brutal," mais qui peut être efficace malgré tout lorsqu'il convient de maintenir une une bonne diversité (en complément, avec d'autres opérateurs de mutation, par exemple) .

– Il existe de nombreuses autres *mutations spécialisées*, par exemple pour le traitement des contraintes, particulièrement utiles lorsque les solutions recherchées se trouvent près des limites de l'espace contraint. Une excellente revue des divers opérateurs génétiques se trouve dans [10].

2.2.3 Programmation génétique, et représentation par arbre

La programmation génétique (PG) correspond à une représentation de structures de longueurs variables sous forme d'arbres. La GP a été impaginée à l'origine pour manipuler des programmes codés en LISP, [45] dans le but créer des programmes qui puissent résoudre des problèmes pour lesquels ils n'ont pas été explicitement programmés. La richesse de la représentation arborescente de taille variable est l'une des clés du succès de ce courant dans le milieu évolutionnaire, en effet beaucoup de problèmes d'optimisation, de commande ou de contrôle peuvent se formuler comme un problème d'induction de programme.

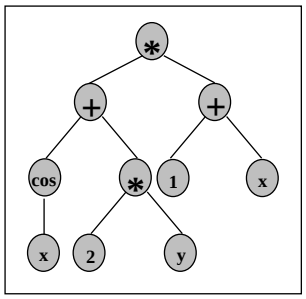

FIG. 4 – Exemple d'une représentation arborescente de la fonction $((\cos(x) + 2 * y) * (1 + x))$.

Un algorithme de PG explore un espace de programmes récursivement composés à partir d'éléments d'ensembles de fonctions, de variables et de terminaisons (données, constantes)[5]. Les individus de la population sont des programmes qui, lorsqu'ils sont exécutés, produisent les solutions au problème posé.

Les croisements sont le plus souvent des échange de sous-arbres. Les mutations sont plus complexes, on en distingue plusieurs‘ suivant leur action sur la structure du génôme.

- suppression/ajout de noeud,
- modification de la fonction d'un noeud en conservant son arité,
- mutation des constantes (valeurs continues), par exemple par ajout d'un bruit Gaussien,
- mutation des variables discrètes, par permutation ou tirage aléatoire uniforme dans un ensemble de valeurs.

Les applications de la programmation génétique sont très nombreuses, par exemple [45] :

- *en contrôle optimal*, exemple du contrôle par un programme de la force appliquée sur un chariot, de façon à lui faire atteindre un point donné en un minimum de temps,
- *en planification de trajectoires et d'actions en robotique :* exemple de la fourmi artificielle qui doit trouver de la nourriture le long d'un chemin irrégulier,
- *pour la régression symbolique* : faire évoluer une expression mathématique (c'est-à-dire un programme) de façon à modéliser au mieux un ensemble fini de données,

2.2.4 Évolution grammaticale et représentation par grammaire

L'expression des gènes ou plus exactement des protéines codées par des séquences de gènes semble dans la nature indépendante de leur position dans le chromosome. Cette propriété d'indépendance vis-à-vis de la position est une propriété qui peut être avantageuse dans bien des problèmes combinatoires. L'évolution grammaticale (GAUGE pour "Genetic Algorithm using Grammatical Evolution", ou en plus court GE) propose une représentation discrète de longueur variable, qui grâce à une étape de traduction fondée sur l'emploi d'une grammaire (BNF : Backus Naur

[5]Un problème couramment rencontré dans ce domaine est celui du "bloat," c'est-à-dire de la saturation de l'espace mémoire liée à une croissance démesurée des tailles d'arbres au cours de l'évolution, si l'on ne prend pas la précaution de limiter la taille des génômes par un moyen ou un autre [51, 50, 78].

Form) produit des individus complexes répondant aux critères de validité de la grammaire de traduction. Le génome dicte en fait quelles sont les règles de la grammaire qui doivent être appliquées à chaque étape de la dérivation à partir d'un symbole prédéfini.

Du fait de la structure des gène manipulés, tous les opérateurs classiques de la représentation discrète peuvent être employés, la difficulté ainsi que l'intérêt de cette approche réside essentiellement dans le choix de la grammaire de traduction des gène qui permet l'intégration de contraintes complexes. En même temps il faut s'assurer que les dérivations ne bouclent pas indéfiniment, et ne deviennent pas excessives en termes de temps de calcul. Une description détaillée de l'évolution grammaticale peut se trouver dans [69, 62].

2.3 Le dosage de la pression de sélection

L'opération de sélection est le seul point de l'algorithme où intervient la fonction de "fitness" qui est optimisée au cours de l'évolution : cette fonction est employée pour faire un parallèle avec l'adaptation d'un individu à son environnement. L'unique contrainte qui pèse sur elle est que l'on doit pouvoir guider l'opérateur de sélection à partir des valeurs qu'elle renvoie. Usuellement, elle doit pouvoir être traduite en probabilité (par une transformation quelconque). *En aucun cas elle a besoin d'être continue ou dérivable.*

L'idée fondamentale est de sélectionner aléatoirement des individus de la population courante devant être reproduits, en favorisant les meilleurs individus. La *sélection proportionnelle*, par exemple, se fait par tirage aléatoire biaisé où la probabilité de sélection d'un individu est directement proportionnelle à sa valeur de fitness vis-à-vis des individus de sa population : $P(i) = fitness(i)/(\sum_{k=1}^{TaillePop} fitness(k))$. L'influence de la *pression sélective* sur la diversité génétique est énorme, et le bon déroulement de l'algorithme en dépend crucialement. Un excès de pression sélective amènera par exemple un appauvrissement génétique des population par sur-sélection d'un "super-individu" pour les opérations de reproduction si sa probabilité de sélection est trop dominante.

Les méthodes de sélection les plus efficaces permettent un contrôle de la pression sélective, nous donnons ci-dessous les méthodes usuelles.

– *Le scaling* se fait par transformation linéaire sur la fonction de fitness pour que le maximum

de la fitness réétalonnée soit C fois sa moyenne sur la population courante. C représente une mesure de la *pression sélective* (voir [26]). En règle générale, C est fixé entre 1.2 et 2.

La probabilité de sélection d'un individu de la population courant est donc calculée par :

$$P(x) = \frac{F'(x)}{\sum_{x \in Population} F'(x)} \quad \text{avec} \quad F'(x) = a * F(x) + b$$

$$a = \frac{(C-1) * Fmoy}{Fmax - Fmoy} \quad \text{et} \quad b = \frac{Fmoy * (Fmax - C * Fmoy)}{Fmax - Fmoy}$$

$Fmax$ and $Fmoy$ sont les valeurs maximales et moyennes de la population courante (voir figure 5). Les facteurs a et b sont recalculés à chaque génération.

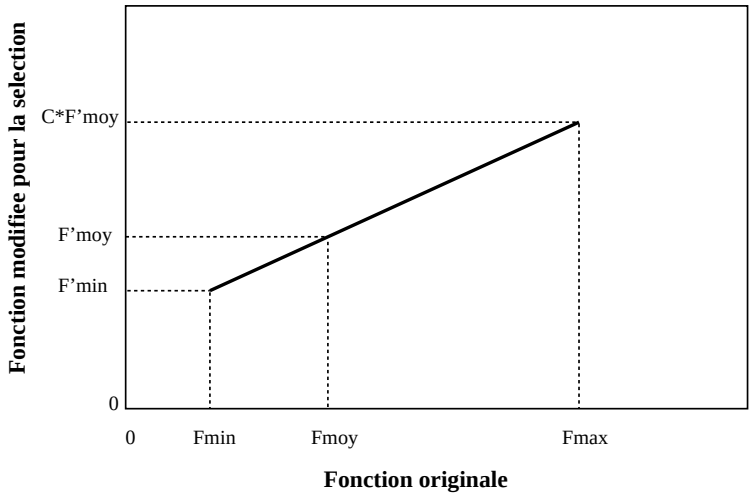

FIG. 5 – Réétalonage de la fonction d'évaluation

– *la sélection par le rang* consiste à affecter une probabilité de sélection proportionnelle au rang de chaque individu dans la population classée par ordre de fitness décroissant.
– *la sélection par tournoi* consiste à chaque fois qu'il faut sélectionner un individu, à tirer aléatoirement T individus dans la population (indépendamment de leurs valeurs de fitness), et à choisir le meilleur d'entre eux. La pression sélective est directement reliée à la taille du tournoi : plus T est grand, plus elle est importante.

3 Aspects théoriques

Il est assez clair que les AE sont des algorithmes d'optimisation assez complexes à mettre en oeuvre (on l'a dit, l'approche en "boîte noire" n'est pas raisonnable) et relativement coûteux en temps de calcul. Leur efficacité dépend d'un dosage subtil entre capacité d'exploration et capacité d'exploitation. Intuitivement, on peut voir un AE comme un algorithme de "tâtonnement aléatoire," où la composante aléatoire doit être dosée intelligemment, en fonction de ce que l'on peut savoir a priori sur le problème à résoudre : trop d'exploration aléatoire fait perdre du temps, trop peu risque de laisser l'algorithme bloqué dans un optimum local.

Les approches téoriques tentent de répondre à un certain nombre de grandes quetions concernant ces algorithmes de recherche stochastique, et notamment à celle de savoir quand et comment utiliser efficacement des AE. La réponse à cette question n'est pas simple. Cependant, d'assez nombreuses analyses théoriques et expérimentales fournissent quelques éléments de réponse, pour comprendre quand l'emploi d'un AG se justifie ou peut apporter quelque chose en comparaison ou en collaboration avec des techniques classiques.

La théorie des schémas est chronologiquement la première "théorie" de convergence globale des algorithmes génétiques, c'est en fait une modélisation extrêmement simplifiée du comportement d'un AE[37, 26], résidant sur des hypothèses valides uniquement dans les premiers pas de l'algorithme et dans le cas de populations de taille infinie. Cette approche a été relativement décriée, car elle repose sur des approximations assez peu réalistes, elle reste cependant une excellente base intuitive pour comprendre les mécanismes de convergence, notamment concernant le rôle de l'opérateur de croisement.

Des résultats de convergence locale pour les stratégies d'évolution [6, 23, 24] ont été aussi développés, puis enfin des modèles globaux rigoureux, comme les modélisations Markoviennes, à base de systèmes dynamiques [85, 3, 39, 20, 61], ou selon des modèles de physique statistique [1, 72, 73]. Ces approches beaucoup plus complexes nous donnent des résultats précis sur la dynamique de populations de taille finie, mais encore sur des modèles simplifiés.

La modélisation des AE est un sujet théorique très complexe et très actif, qui peu à peu attire les mathématiciens. Les enjeux sont importants et de nombreux séminaires, conférences et groupes de

travail sont spécialisés sur ce sujet. Les thèmes favoris en principalement le problème de la convergence, la facilité ou difficulté des paysages de fitness pour un AE (étude des fonctions “déceptives”, NK-landscapes), le choix des divers paramètres et le choix d’une bonne représentation.

4 Au delà de l’optimisation

4.1 Paysages multimodaux et simulation de niches écologiques

Cette technique est apparue pour pallier certaines insuffisances des AE classiques, notamment lorsque les fonctions à optimiser présentent plusieurs optima globaux. En effet, dans ce cas l’AE classique converge aléatoirement vers un seul des optima. Or dans de nombreuses applications, il est parfois important de pouvoir détecter tous ces optima, c’est-à-dire faire plus qu’une simple recherche de *la* solution optimale. Des extensions de l’AE classique, copiées sur le phénomène naturel de création de niches écologiques, sont fondées sur la gestion implicite ou explicite de sous-populations (ou de niches).

Un premier jeu de méthodes consiste à maintenir la diversité des populations [26, 29]. L’idée est de favoriser l’émergence d’espèces distinctes en modifiant le mécanisme de remplacement de façon à éviter l’accumulation d’individus dans les zones déjà explorées. Une notion de similarité des génômes intervient pour décider quel descendant remplace quel parent (“crowding scheme” [26, 29, 41], ou opérateur d’unicité fondé sur une distance de Hamming [26, 29, 57]).

Les méthodes les plus répandues sont actuellement les méthodes de *partage des ressources* (sharing, [29, 26])qui exploitent la notion de distance au niveau de la sélection : si les individus d’une même sous-population ont à partager les ressources, la croissance de cette sous-population est limitée, et lorsqu’il y a surpopulation, les individus tendent à rechercher de nouvelles régions à coloniser. Cette technique est fondée sur une mesure de distance inter-chromosomes et une notion de voisinage : la fonction d’évaluation d’un individu est réduite proportionnellement au nombre de ses voisins.

Il existe enfin des techniques qui créent explicitement des sous-populations [15], parfois sur des processeurs séparés, et échangent des individus entre ces sous-populations à intervalles fixes, lorsqu’on estime que ces sous populations sont stabilisées.

4.2 Co-évolution, apprentissage collectif et évolution Parisienne

Il est parfois possible de formuler un problème comme une tâche d'apprentissage collectif, la solution recherchée étant construite à partir de l'ensemble d'une population évoluée et non plus à partir du seul meilleur individu de la population finale.

Les thèmes les plus marquants de cette tendance sont les systèmes de classeur, l'approche Parisienne, les techniques de co-évolution (proie/prédateurs), et les techniques à base de colonies d'insectes sociaux, dont les plus connus sont les ACO, algorithmes à colonies de fourmis (ant colony algorithms) [21, 22].

Un système de classeurs (classifier system), est un système qui apprend un ensemble de règles syntactiques simples (appelées *classeurs*), manipulées sous la forme "SI $<$ *condition* $>$ ALORS $<$ *action* $>$."

Les informations venant de l'environnement sont transmises via des *détecteurs* sous forme de un ou plusieurs *messages* de tailles finies (sous forme d'une chaîne de bits, le plus souvent). Ces message provenant de l'environnement sont stockés dans une *liste de messages* et peuvent alors activer les règles ou *classeurs*. Lorsqu'il est activé[6], un classeur renvoie un message (la partie *action*) à la liste de messages. Ces messages peuvent alors à leur tour activer d'autres classeurs, ou bien déclencher une action transmise à l'environnement par l'intermédiaire *déclencheurs*. De cette façon, le système combine des interactions avec l'extérieur et des actions de "réflexion" internes, pour déterminer quelle sera la prochaine action.

La population de classeurs évolue à l'aide d'un algorithme génétique standard. On voit ici que le système dans son ensemble est évolué. Le point délicat de cette approche étant l'évaluation individuelle des classeurs. Celle-ci est faite à l'aide d'un système d'attribution des crédits relativement complexe, appelé *bucket-brigade*. Schématiquement, il distribue la récompense à une action renvoyée par l'environnement aux classeurs qui ont été activés pour générer cette action [26].

L'idée de co-évolution peut s'exploiter par ailleurs dans le cadre de l'optimisation pure. Ce qu'on appelle actuellement *l'approche Parisienne* est une exploitation relativement récente de ce

[6] c'est-à-dire lorsque sa partie *condition* correspond à un message de la liste : dans le cas de l'aphabet binaire, la condition est décrite par une chaîne de caractères sur l'aphabet 0,1,#, le caractère # pouvant correspondre soit à un 0 soit à un 1 dans le message

concept [16], elle est fondée sur la capacité des EA à drainer la population dans des zones quasi-optimales de l'espace de recherche. L'idée est donc de fabriquer un paysage de fitness et un processus d'évolution où l'ensemble de la population (ou du moins une grande part) représente la solution recherchée. Les individus ne correspondent plus individuellement mais communautairement à une solution potentielle au problème posé. La population est une société qui construit en commun la solution recherchée : c'est une co-évolution. Cependant, contrairement aux approches de co-évolution plus classiques, les espèces ne sont pas clairement identifiées en termes de proie et de prédateur par exemple.

Bien entendu, la spécification et le calibrage de tels algorithmes est encore plus complexe que pour les approches évolutionnaires classiques, et la diversité des population devient un facteur crucial. En outre, le fractionnement d'un problème en sous-problèmes interconnectés qui se prête à une approche Parisienne n'est pas toujours possible.

De façon schématique, les AE Parisiens ont toutes les composantes des AE classiques plus :

- *deux* fonctions de fitness : *une fonction globale* qui est calculée sur l'ensemble de la population, ou sur une proportion de celle-ci (par exemple à la suite d'un processus de clusterisation, ou d'élimination des individus trop mauvais), et *une fonction locale* calculée sur chaque individu, qui mesure la proportion avec laquelle cet individu contribue à la solution globale.
- un processus de re-distribution qui répartit à chaque génération le fitness global sur les individus ayant contribué à la solution,
- un mécanisme de maintien de la diversité, afin d'éviter les modes dégénérés où tous les individus sont concentrés sur la même zone de l'espace de recherche.

Cette approche a permis de nombreuses applications, et notamment des applications "en ligne" : text-retrieval [48, 49], applications artistiques [14]) et temps-réel (vision stéréo par algorithme des mouches [54]), ce qui est tout-à-fait notable pour des algorithmes ayant une réputation d'extrême gourmandise en temps de calcul !

4.3 L'approche multi-objectif.

Dans de nombreuses applications, l'identification précise des quantités à optimiser est parfois difficile, particulièrement dans des cas où il existe plusieurs critères de jugement, quelquefois

incompatibles (maximiser la résistance d'une pièce mécanique, tout en minimisant son poids et son coût de fabrication, par exemple). Ces optimisations sont d'autant plus complexes à traiter que l'on ne sait pas forcément estimer les poids relatifs des divers critères et contraintes en jeu. On parle alors non plus d'optimisation simple mais d'optimisation multi-critère, sans donner de priorités aux divers critères $f_1(x), f_2(x), ..., f_n(x)$.

La notion de *domination* est essentielle : une solution x_1 domine une solution x_2 (on écrit $x_1 \succ x_2$) si et seulement si :

$$\forall j \in 1, ..., n \qquad f_j(x_1) \geq f_j(x_2) \quad \text{et} \quad \exists i \in 1, ..., n \qquad f_i(x_1) > f_i(x_2)$$

La solution à un problème d'optimisation multi-critère est alors un ensemble de solutions, formant ce que l'on nomme le *front de Pareto*, et qui représente l'ensemble des solutions non-dominées de l'espace de recherche. L'idée d'utiliser des AE pour trouver le front de Pareto d'un problème multi-critère vient alors assez naturellement, au prix d'une légère modification du schéma évolutionnaire classique. La procédure de sélection doit notamment être adaptée de façon à pousser la population vers le front de Pareto, tout en maintenant une bonne diversité pour assurer un bon échantillonnage du front de Pareto (éviter les solutions dégénérées où toutes les solutions sont concentrées en un seul point du front de Pareto). Là encore, la maîtrise de la diversité de la population est cruciale. Une étude comparative des diverses méthodes évolutionnaires pour l'optimisation multi-critère se trouve dans [86].

4.4 L'évolution interactive.

Dans le cas où l'on ne sait pas précisément définir ce que l'on souhaite optimiser, il est nécessaire de développer des stratégies spécifiques. C'est le cas, on l'a vu, pour les techniques d'optimisation multi-critères, où celles-ci s'imposent lorsque l'on ne sait pas donner de poids relatifs aux différents critères en jeu (l'approche d'optimisation classique fondée sur une combinaison pondérée des critères n'est plus possible). Le problème se complique encore lorsqu'on se trouve dans des cas où ce que l'on souhaite optimiser n'est pas mesurable à l'aide d'une fonction mathématique (par exemple la simple notion de "satisfaction"), il s'agit alors de faire intervenir un utilisateur humain dans la boucle évolutionnaire. Cela correspond au courant des *algorithmes évolutionnaires*

interactifs.

Les premiers travaux concernant l'évolution interactive [76, 75, 81, 4] concernaient la création artistique et la syntèse d'images numériques. De nombreuses études touchent maintenant divers domaines d'application, où les quantités à optimiser sont liées à des jugements subjectifs (visuels ou auditifs). Des travaux tout-à-fait caractéristiques sur le sujet sont par exemple [80] pour l'adaptation de prothèse auditives, [42] pour le développement de lois de contrôle de bras de robot qui correspondent à des mouvements souples, proches de l'humain et psychologiquement bien percus par l'utilisateur, ou [59] pour le design de pages HTML. On pourra trouver une revue de ce vaste sujet dans [79].

Définir les algorithmes évolutionnaires interactifs (AE-I) comme des AE dont la fonction de fitness est donnée par une interaction humaine ne suffit pas à représenter la variété des applications actuelles. Il faut en fait définir l'interaction de façon plus large. La fonction de fitness peut n'être établie que partiellement par interaction, des interventions de l'utilisateur directement au niveau des génômes, des opérateurs génétiques, des paramètres et des diverses stratégies peuvent être envisagées. Ainsi, il sembe actuellement mieux adapté de considérer les AE-I comme des processus évolutionnaire contraint par une interaction avec un utilisateur humain.

En outre, l'interaction pose le problème de la lassitude de l'utilisateur [64]. Il faut en effet trouver des moyens d'éviter des interactions répétitives, ennuyeuses ou mal percues de l'utilisateur. Le travail de l'artiste Steven Rooke [66] montre bien par exemple la quantité extraordinaire d'interactions nécessaires à l'apparition d'images esthétiques lorsqu'on part d'une "soupe primordiale" de composantes primitives. Les techniques usuelles [64, 79, 9] sont

- de réduire la taille des populations et du nombre de générations,
- de choisir des modèles spécifiques pour contraindre la recherche dans des zone a priori interessantes de l'espace de recherche,
- de faire de l'apprentissage automatique (fondé sur un nombre limité de quantités caractéristiques) afin de proposer une pré-notation automatique des individus et de ne présenter à l'interaction que les individus les plus intéressants de la population, en considérant les votes antérieurs de l'utilisateur.

Autoriser des interactions directes au niveau phénotypique représente un pas de plus vers l'utili-

sation efficace d'un EA-I par exemple dans le cadre artistique, à partir du moment où il est possible pour l'utilisateur d'avoir des intuitions sur des composantes partielles de solution, ou de pouvoir faire interactivement de l'optimisation "locale."

5 Conclusion et perspectives

Le Darwinisme artificiel se conjugue actuellement sous des aspects très variés, et la multiplicité des applications réussies ou actuellement à l'étude témoigne de l'attraction de ce domaine pour les chercheurs et les industriels [7]

Ces méthodes sont maintenant très utilisées en optimisation numérique, lorsque les fonctions à optimiser sont complexes, de forte dimensionnalité, irrégulières, mal connues, ou en optimisation combinatoire, pour des problèmes de théorie des graphes (voyageur de commerce, coloration de graphes), de séquencement de tâches, répartition de ressources, d'emploi du temps, du sac à dos, dont la résolution efficace est fondée le plus souvent sur des AE hybridés avec des techniques "classiques." C'est dans ce cadre que de nombreuses études sur la représentation des problèmes et le traitement des contraintes ont été menées, de façon à produire des algorithmes adaptés aux problèmes réels et de grande taille.

La physique et l'ingénierie ont notamment été source de nombreux problèmes réels complexes, par exemple pour l'optimisations de structures mécaniques [82, 53, 31], de profils d'ailes d'avion, de tuyères de réacteurs, ou de processus chimiques industriels. Des domaines comme l'économie et la finance (simulation d'économies artificielles, optimisation de portefeuilles bancaires), le traitement d'images et de signaux (détection de formes caractéristiques, vision stéréo [54], ou débruitage d'images [56] par exemple), la biologie et la médecine (séquencement du génôme, analyse du repliement de protéines, analyse de données et imagerie médicale) ont aussi largement puisé dans le panel des méthodes évolutionnaires.

Le Darwinisme artificiel est sans nul doute extrêmement attrayant, il faut prendre garde cependant à ne pas utiliser les concepts d'évolution artificielle pour produire des algorithmes ridicule-

[7] voir [26] pages 126 à 129, pour des applications développées jusqu'à 1989, et le site du reseau Europeen EvoNet http ://evonet.dcs.napier.ac.uk/ pour un panel d'applications récentes

ment couteux en temps de calcul en comparaison avec des méthodes plus classiques. D'un point de vue expérimental, il ne faut donc jamais hésiter à comparer et à hybrider un AE avec des techniques classiques du domaine d'application visé. On peut même aller jusqu'à comparer la méthode évolutionnaire avec une recherche aléatoire pure de façon à évaluer le niveau d'intelligence et le coût calculatoire induit par l'évolution, cela peut être riche d'enseignements [55]. L'expérience prouve aussi que si les composantes (autrement dit la représentation et la topologie de recherche induite par les opérateurs génétiques associés) ainsi que les paramètres de l'évolution sont soigneusement réglés, il possible d'obtenir des algorithmes extrêmement efficaces et rapides [11]. Mais cette étape de réglage peut se révéler très délicate et parfois rébarbative.

Faciliter le réglage des paramètres est un des enjeux des recherches théoriques actuelles sur les AE : les travaux récents, essentiellement en ce qui concerne l'étude de la convergence de ces algorithmes (modélisations Markoviennes), ont permis de poser des bases plus solides pour ces techniques, souvent critiquées à cause de leur aspect "empirique." Ces approches fournissent un cadre théorique riche, qui permettra non seulement de proposer des techniques d'ajustement de paramètres, mais aussi de comprendre plus finement quand et pourquoi un AE est efficace.

Le design d'algorithmes auto-adaptatifs répond aussi à ce besoin de recettes de réglage des paramètres : laisser l'évolution prendre en charge elle-même cette tâche est certainement une excellente solution, explorée par de nombreux chercheurs du courant "stratégies d'évolution," si l'on prend bien garde de ne pas alourdir trop l'algorithme. Comme toujours, ce choix est soumis à un compromis : les capacités adaptatives sont coûteuses, il faut les exploiter à bon escient.

L'idée d'exploiter le Darwinisme artificiel dans un cadre élargi, en ne se limitant plus à des tâches l'optimisation "simples," est par ailleurs un courant qui prend actuellement beaucoup d'ampleur, tout comme les travaux sur l'évolution interactive. Le dynamisme de ce domaine de recherche se concrétise par le nombre de conférences internationales consacrées exclusivement à ce thème, ainsi que par le nombre de publications de livres, de revues et de logiciels.

Références

[1] chapter Modelling GA Dynamics., pages 59–86. Proceedings Theoretical Aspects of Evolutionary Computing, 2001.

[2] J. Albert, F. Ferri, J. Domingo, and M. Vincens. An Approach to Natural Scene Segmentation by Means of Genetic Algoritms with Fuzzy Data . In *Pattern Recognition and Image Analysis*, pages 97–113, 1992. Selected papers of the 4th Spanish Symposium (Sept 90), Perez de

la Blanca Ed.

[3] Lee Altenberg. Evolutionary computation models from population genetics. part 2 : An historical toolbox. In *Congress on Evolutionary Computation*, 2000. Tutorial.

[4] P. J. Angeline. Evolving fractal movies. In *Genetic Programming 1996 : Proceedings of the First Annual Conference, John R. Koza and David E. Goldberg and David B. Fogel and Rick L. Riolo (Eds)*, pages 503–511, 1996.

[5] Peter J. Angeline and Jordan B. Pollack. Competitive environments evolve better solutions for complex tasks. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, California, 1993. Morgan Kaufmann.

[6] T. Baeck, F. Hoffmeister, and H. P. Schwefel. A survey of evolution strategies. In *International Conference on Genetic Algorithms*, pages 2–10, 1991. 13-16 July.

[7] T. Baeck and H. P. Schwefel. An overview of evolutionnary algorithms for parameter optimisation. Technical report, University of Dortmund, 1992.

[8] J. D. Bagley. *The Behaviour of adaptative systems which employ genetic and correlation algorithms*. PhD thesis, University of Michigan, 1967. 5106B.

[9] Wolfgang Banzhaf. *Handbook of Evolutionary Computation*, chapter Interactive Evolution. Oxford University Press, 1997.

[10] Sana Ben Hamida. *Algorithmes évolutionnaires : prise en compte des contraintes et applications réelles*. PhD thesis, Université Pris-Sud XI, spécialité informatique, 2001. 29 Mars 2001.

[11] A. Boumaza and J. Louchet. Dynamic flies : Using real-time evolution in robotics. In *EVO-IASP 2001 Workshop, Artificial Evolution in Image Analysis and Signal Processing*, 2001.

[12] Lawrence Bull and Terence C. Fogarty. Co-evolving communicating classifier systems for tracking. pages 522–527, Innsbruck, Austria, 13. -16. April 1993. Springer-Verlag, Wien.

[13] Caviccio. *Adaptative search using simulated evolution*. PhD thesis, University of Michigan, 1970.

[14] J. Chapuis and E. Lutton. Artie-fract : Interactive evolution of fractals. In *4th International Conference on Generative Art*, Milano, Italy, December 12-14 2001.

[15] J. P. Cohoon, W. N. Martin, and D. S. Richards. Genetic algorithms and punctuated equilibria in VLSI. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature - Proceedings of 1st Workshop, PPSN 1*, volume 496 of *Lecture Notes in Computer Science*, pages 134–144, Dortmund, Germany, 1-3 Oct 1991. Springer-Verlag, Berlin, Germany.

[16] Pierre Collet, Evelyne Lutton, Frédéric Raynal, and Marc Schoenauer. Polar ifs + parisian genetic programming = efficient ifs inverse problem solving. *Genetic Programming and Evolvable Machines Journal*, 1(4) :339–361, 2000. October.

[17] Y. Davidor. Robot programming with a genetic algorithm. In *Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering*, number Cat. No. 90CH2867-0, pages 186–191, Tel-Aviv, Israel, May, 8-10 1990. IEEE Computer Society Press, Los Alamitos, CA.

[18] Y. Davidor. *Genetic Algorithms and Robotics. A heuristic Strategy for Optimization*. World Scientific Series in Robotics and Automated Systems - vol 1. World Scientific, Teaneck, NJ, 1991.

[19] L. Davis. *Genetic Algorithms and Simulated Annealing*. Pittman, London, 1987.

[20] Thomas E. Davis and Jose C. Principe. A Simulated Annealing Like Convergence Theory for the Simple Genetic Algorithm . In *Proceedings of the Fourth International Conference on Genetic Algorithm*, pages 174–182, 1991. 13-16 July.

[21] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. *New Ideas in Optimization*, pages 11–32, 1999. D. Corne, M. Dorigo and F. Glover, editors, McGraw-Hill.

[22] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2) :137–172, 1999.

[23] D.B. Fogel. An analysis of evolutionary programming. In D.B. Fogel and W. Atmar, editors, *Proceedings of the First Annual Conference on Evolutionary Programming*, pages 43–51, La Jolla, California, 1992.

[24] D.B. Fogel. Asymptotic convergence properties of genetic algorithms and evolutionary programming : Analysis and experiments. In *Journal of Mathematical Biology*, 1992.

[25] D. E. Goldberg. Zen and the art of genetic algorithms. In *Third International Conference on Genetic Algorithms*, 1989.

[26] David A. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, inc., Reading, MA, January 1989.

[27] David E. Goldberg. Genetic algorithms and rule learning in dynamic system control. In *ICGA85*, pages 8–15, 1985.

[28] David E. Goldberg, Bradley Korb, and Kalyanmoy Deb. Messy genetic algorithms : Motivation, analysis, and first results. *Complex Systems*, 3(5) :493–530, October 1989.

[29] David E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications*, pages 41–49, Hillsdale, New Jersey, 1987. Lawrence Erlbaum Associates.

[30] David E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In J. J. Grefenstette, editor, *Proceedings of the second international conference on genetic algorithms and their applications*, pages 59–68, Hillsdale, New Jersey, 1987. Lawrence Erlbaum Associates.

[31] H. Hamda, F. Jouve, E. Lutton, M. Schoenauer, and M. Sebag. Compact unstructured representations in evolutionary topological optimum design. *Applied Intelligence*, 16, 2002.

[32] A. Hill and C. J. Taylor. Model-Based Image Interpretation using Genetic Algorithms . *Image and Vision Computing*, 10(5) :295–301, June 1992.

[33] W. Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. pages 228–234, Cambridge, MA, 1990. MIT Press / North-Holland. (also as Physica D, Vol. 42).

[34] F. Hoffmeister and T. Bäck. Genetic algorithms and evolution strategies : similarities and differences. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature - Proceedings of 1st Workshop, PPSN 1*, volume 496 of *Lecture Notes in Computer Science*, pages 455–469, Dortmund, Germany, 1-3 Oct 1991. Springer-Verlag, Berlin, Germany.

[35] F. Hoffmeister and T. Baeck. Genetic algorithms and evolution strategies : Similarities and differences. Technical Report SYS-1/92, University of Dortmund, February 1992.

[36] J. H. Holland. Outline for a logical theory of adaptative systems. *Journal of the association for the Computing Machinery*, (3) :297–314, 1962.

[37] J. H. Holland. *Adaptation in Natural and Artificial System* . Ann Arbor, University of Michigan Press, 1975.

[38] R. B. Hollstien. *Artificial genetic adaptation in computer control systems*. PhD thesis, University of Michigan, 1971. 1510B.

[39] Jeffrey Horn. Finite Markov Chain Analysis of Genetic Algorithms with Niching . IlliGAL Report 93002, University of Illinois at Urbana Champaign, Department of General Engineering, 117 Transportation Building, 104 South Mathews Avenue, Urbana, IL 61801-2996, February 1993.

[40] Philip Husbands and Frank Mill. Simulated co-evolution as the mechanism for emergent planning and scheduling. pages 264–270, San Diego, 13.-16. July 1991. Morgan Kaufmann Publishers.

[41] K. A. De Jong. *Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975. 5140B.

[42] S. Kamohara, Hideyuki Takagi, and T. Takeda. Control rule acquisition for an arm wrestling robot. In *IEEE Int. Conf. on System, Man and Cybernetics (SMC'97)*, volume 5, Orlando, FL, USA, 1997.

[43] S. A. Kauffman and S. Johnsen. Co-evolution to the edge of chaos : Coupled fitness landscapes, poised states, and co-evolutionary avalanches. In *ALifeII conference*, 1992.

[44] Ken-Ichi Kojima, editor. *Mathematical Topics in Population Genetics* , volume 1. Springer-Verlag, Berlin, 1970. Biomathematics.

[45] J. R. Koza. *Genetic Programming* . MIT Press, 1992.

[46] John R. Koza. Evolution and coevolution of computer programs to control independently acting agents. In *Proceedings SAB90*, pages 366–375, 1990.

[47] John R. Koza. Genetic evolution and co-evolution of computer programs. In *ALifeII conference*, pages 603–629, 1991.

[48] Yann Landrin-Schweitzer, Pierre Collet, and Evelyne Lutton. Interactive gp for data retrieval in medical databases. In *EUROGP'03*. LNCS, Springer Verlag, 2003.

[49] Yann Landrin-Schweitzer, Pierre Collet, Evelyne Lutton, and Thierry Prost. Introducing lateral thinking in search engines with interactivee-volutionary algorithms. In *Annual ACM Symposium on Applied Computing (SAC 2003), Special Track on"Computer Applications in Health Care" (COMPAHEC 2003)*, 2003. March 9 to 12, Melbourne, Florida, U.S.A.

[50] W. B. Langdon. Quadratic bloat in genetic programming. In Darrell Whitley, David Goldberg, Erick Cantu-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 451–458, Las Vegas, Nevada, USA, 10-12 2000. Morgan Kaufmann.

[51] W. B. Langdon and W. Banzhaf. Genetic programming bloat without semantics. In M. Schoenauer, K. Deb, G. Rudolf, X. Yao, E. Lutton, Merelo. J.J., and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VI 6th International Conference*, Paris, France, September 16-20 2000. Springer Verlag. LNCS 1917.

[52] P. Larranaga and J. A. Lozano. *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*. Kluwer, Boston, MA, 2002.

[53] R. Leriche and R.T. Haftka. Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm. *AIAA Journal*, 31(5) :951–969, May 1993.

[54] J. Louchet, M. Guyon, M.-J. Lesot, and A. Boumaza. Dynamic flies : a new pattern recognition tool applied to stereo sequence processing. *Pattern Recognition Letters*, 2002. No. 23 pp. 335-345, Elsevier Science B.V.

[55] E. Lutton, P. Collet, and J. Louchet. Easea comparisons on test functions : Galib versus eo. In *EA01 Conference on Artificial Evolution*, Le Creusot, octobre 2001.

[56] J. Lévy Véhel and E. Lutton. Evolutionary signal enhancement based on hölder regularity analysis. In *EVO-IASP 2001 Workshop, Artificial Evolution in Image Analysis and Signal Processing*, 2001.

[57] M. L. Mauldin. Maintening diversity in genetic search. In *Proceedings of the National Conference on Artificial Intelligence*, 1984.

[58] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Artificial Intelligence. Springer Verlag, New York, 1992.

[59] Nicolas Monmarche, G Nocent, Gilles Venturini, and P. Santini. *Artificial Evolution, European Conference, AE 99, Dunkerque, France, November 1999, Selected papers,*, volume Lecture Notes in Computer Science 1829, chapter On Generating HTML Style Sheets with an Interactive Genetic Algorithm Based on Gene Frequencies. Springer Verlag, 1999.

[60] H Muhlenbein and G. Paass. From recombination of genes to the estimation of distributions. i. binary parameters. In *Parallel Problem Solving from Nature*, 1996.

[61] A.E. Nix and M.D. Vose. Modeling genetic algorithms with markov chains. *Annals of Mathematics and Artificial Intelligence*, 5(1) :79–88, April 1992.

[62] M. O'Neill and C Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4), August 2001.

[63] M Pelikan, D. G. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, (21) :5–20, 2002. also as IlliGAL Report No 99018.

[64] Riccardo Poli and Stefano Cagnoni. Genetic programming with user-driven selection : Experiments on the evolution of algorithms for image enhancement. In *2nd Annual Conf. on Genetic Programming*, 1997.

[65] I. Rechenberg. *Evolutionsstrategie : Optimierung Technicher System nach Prinzipien der Biologischen Evolution*. Fromman Holzboog, Stuttgart, 1973.

[66] S. Rooke. The evolutionary art of steven rooke. http ://www.azstarnet.com/∼srooke/.

[67] R. S. Rosenberg. *Simulation of genetic populations with biochemical properties*. PhD thesis, University of Michigan, 1967. 2732B.

[68] G. Roth and M. D. Levine. Geometric Primitive Extraction Using a Genetic Algorithm . In *IEEE Computer Society Conference on CV and PR*, pages 640–644, 1992.

[69] C. Ryan, M. Nicolau, and M. O'Neill. Genetic algorithms using grammatical evolution. In Springer Verlag, editor, *Proceedings of EuroGP 2002*, 2002.

[70] H P Schwefel. *Evolutionsstrategie und numerische Optimierung*. PhD thesis, Technische Universitat, Berlin, may 1975.

[71] H-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & sons, New-York, 1981, 1995 2nd edition.

[72] J. L. Shapiro and A. Prügel-Bennett. Maximum entropy analysis of genetic algorithm operators. *Lecture Notes in Computer Science*, 993 :14–24, 1995.

[73] J.L. Shapiro, M. Rattray, and A. Prügel-Bennett. The statistical mechanics theory of genetic algorithm dynamics. In *First International Conference on Evolutionary Computation and Its Applications, Moscow*, 1996. Plenary Lecture.

[74] E. Siegel. Competively evolving decision trees against fixed training cases for natural language processing. In *Advances in Genetic Programming*, 1994. ed. Kim Kinnear. Cambridge, MA : MIT Press.

[75] K. Sims. Interactive evolution of dynamical systems. In *First European Conference on Artificial Life*, pages 171–178, 1991. Paris, December.

[76] Karl Sims. Artificial evolution for computer graphics. *Computer Graphics*, 25(4) :319–328, July 1991.

[77] Robert E. Smith and David E. Goldberg. Diploidy and dominance in artificial genetic search. *Complex Systems*, 6 :251–285, 1992.

[78] Terence Soule, James A. Foster, and John Dickinson. Code growth in genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996 : Proceedings of the First Annual Conference*, pages 215–223, Stanford University, CA, USA, 28–31 1996. MIT Press.

[79] Hideyuki Takagi. Interactive evolutionary computation : System optimisation based on human subjective evaluation. In *IEEE Int. Conf. on Intelligent Engineering Systems (INES'98)*, Vienna, Austria, Sept 17-19 1998.

[80] Hideyuki Takagi and Miho Ohsaki. Iec-based hearing aids fitting. In *IEEE Int. Conf. on System, Man and Cybernetics (SMC'99)*, volume 3, Tokyo, Japan, Oct. 12-15 1999.

[81] S.J.P. Todd and W. Latham. *Evolutionary Art and Computers*. Academic Press, 1992.

[82] P. Trompette, J. L. Marcelin, and C. Schmelding. Optimal damping of viscoelastic constrained beams or plates by use of a genetic algotithm. In *IUTAM*, 1993. Zakopane Pologne.

[83] S. Truvé. Using a Genetic Algorithm to solve Constraint Satisfation Problems Generated by an Image Interpreter. In *Theory and Applications of Image Analysis : 7th Scandinavian Conference on Image Analysis*, pages 378–386, August 1991. Aalborg (DK).

[84] J. Lévy Véhel and E. Lutton. Optimization of fractal functions using genetic algorithms. In *Fractal 93*, 1993. London.

[85] Michael Vose. Modeling simple genetic algorithms. In *FOGA-92, Foundations of Genetic Algorithms*, Vail, Colorado, 24–29 July 1992. Email : vose@cs.utk.edu.

[86] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms : Empirical results. *Evolutionary Computation*, 8(2) :173–195, 2000.