Présentation des algorithmes génétiques et de leurs applications en économie

Thomas Vallée[♯] et Murat Yıldızoğlu[*]

[♯] LEN-C3E
Université de Nantes, LEA-CIL
Chemin de la Censive du Tertre
F-44312 NANTES
Thomas.Vallee@sc-eco.univ-nantes.fr
[*]IFREDE-E3i
Université Montesquieu Bordeaux IV
Avenue Léon Duguit
F-33608 PESSAC
yildi@montesquieu.u-bordeaux.fr

7 septembre 2001, v. 1.2

Résumé

Nous avons assisté ces dernières années une croissance très rapide des travaux utilisant les algorithmes génétiques (AGs). Nous présentons dans cet article à la fois les mécanismes de bases de ces algorithmes et un panorama de leurs applications en économie, accompagné d'une bibliographie représentative.

1 Introduction

Les algorithmes génétiques (**AG**s) sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Leur fonctionnement est extrêmement simple. On part avec une population de solutions potentielles (*chromosomes*) initiales arbitrairement choisies. On évalue leur performance (*fitness*) relative. Sur la base de ces performance on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation. On recommence ce cycle jusqu'à ce que l'on trouve une solution satisfaisante.

Les AGs ont été initialement développés par John Holland (1975). C'est au livre de Goldberg (1989) que nous devons leur popularisation. Leurs champs d'application sont très vastes. Outre l'économie, ils sont utilisés pour l'optimisation de fonctions (De Jong (1980)), en finance (Pereira (2000)), en théorie du contrôle optimal (Krishnakumar et Goldberg (1992), Michalewicz, Janikow et Krawczyk (1992) et Marco *et al.* (1996)), ou encore en théorie des jeux répétés (Axelrod (1987)) et différentiels (Özyildirim (1996, 1997) et Özyildirim et Alemdar (1998)). La raison de ce grand nombre d'application est claire : simplicité et efficacité. Bien sûr d'autres techniques d'exploration stochastique existent, la plus connue étant le *recuit simulé* (*simulated annealing* – voir Davis (1987) pour une association des deux méthodes).

Pour résumer, Lerman et Ngouenet (1995) distinguent 4 principaux points qui font la différence fondamentale entre ces algorithmes et les autres méthodes :

1. Les algorithmes génétiques utilisent un codage des paramètres, et non les paramètres eux mêmes.
2. Les algorithmes génétiques travaillent sur une population de points, au lieu d'un point unique.
3. Les algorithmes génétiques n'utilisent que les valeurs de la fonction étudiée, pas sa dérivée, ou une autre connaissance auxiliaire.

4. Les algorithmes génétiques utilisent des règles de transition probabilistes, et non déterministes.

La simplicité de leurs mécanismes, la facilité de leur mise en application et leur efficacité même pour des problèmes complexes ont conduit à un nombre croissants de travaux en économie ces dernières années (*cf.* Figure 1 – compilé à partir de Alander (2001)).

Comme il n'existe encore aucune présentation en français de cette approche en économie, il nous a paru utile de réaliser un article qui présente à la fois les mécanismes des AG et leurs principales applications en économie et finance, accompagnées d'une bibliographie relativement complète sans être exhaustive (*cf.* Alander (2001) pour une bibliographie quasi-exhaustive).

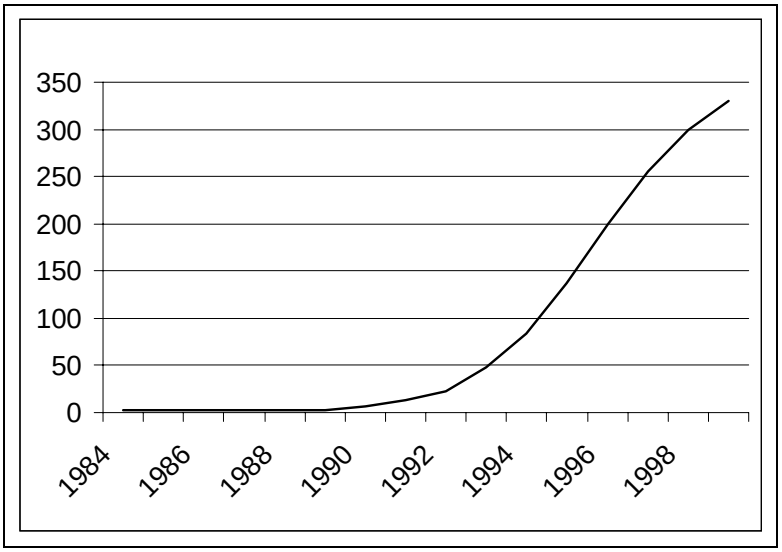

FIG. 1: Évolution cumulée du nombre de publications en économie utilisant les AGs

La plan de l'article découle naturellement de cette constatation. La section 2 est consacrée à la présentation des mécanismes qui constituent l'algorithme. Ces mécanismes s'inspirent directement des mécanismes de l'évolution. Nous présentons la sélection, le croisement et la mutation d'abord dans le cadre originel des AGs où la population est formée de chromosomes binaires (formés de 0 et de 1). Nous présentons ensuite la version des AGs où l'on utilise directement des réels pour représenter les stratégies. Malgré le fait qu'il rend caduc les résultats théoriques que Holland a établis sur les AG, le codage réel apporte la simplicité et la flexibilité (ainsi que la vitesse) dans les AGs.

La section 3 de l'article cherche à donner un tableau aussi complet que possible de toute la diversité des applications des AGs en économie et la finance. Deux types d'utilisation peuvent d'ores et déjà être signalés. Plus en accord avec la conception initiale de Holland, certains travaux utilisent les AGs pour résoudre des problèmes complexes qui ne possèdent pas de solutions analytiques : des estimations économétriques, la résolution des modèles dynamiques, la recherche de l'équilibre de Nash de jeux dynamiques, etc. Une autre lignée de travaux profitent de la capacité adaptative des AGs pour les utiliser en vue de représenter, de manière heuristique, l'apprentissage adaptatifs des agents économiques hétérogènes. Ces modèles cherchent soit à tester la robustesse des résultats obtenus avec des anticipations rationnelles, soit à analyser des questions nouvelles qui bénéficient directement de la possibilité de tenir compte de l'hétérogénéité des agents à rationalité limitée dans les modèles. Les résultats des deux ensembles d'application démontrent bien que les AGs apportent soit des solutions nouvelles, soit un nouvel éclairage dans l'analyse économique.

2 Présentation des algorithmes génétiques (AGs)

Selon Lerman et Ngouenet (1995) un algorithme génétique est défini par :

– *Individu/chromosome/séquence* : une solution potentielle du problème ;

– *Population* : un ensemble de chromosomes ou de points de l'espace de recherche ;

– *Environnement* : l'espace de recherche ;

– *Fonction de fitness* : la fonction - positive - que nous cherchons à maximiser.

Avant d'aller plus loin il nous faut définir quelques termes importants généralement définis sous l'hypothèse de codage binaire.

Définition 1 (Séquence/Chromosome/Individu (Codage binaire)).
Nous appelons une séquence (chromosome, individu) A de longueur $l(A)$ une suite $A = \{a_1, a_2, ..., a_l\}$ avec $\forall i \in [1,l]$, $a_i \in V = \{0,1\}$.

Un chromosome est donc une suite de bits en codage binaire, appelé aussi chaîne binaire. Dans le cas d'un codage non binaire, tel que le codage réel, la suite A ne contient qu'un point, nous avons $A = \{a\}$ avec $a \in \Re$.

Définition 2 (Fitness d'une séquence). *Nous appelons fitness d'une séquence toute valeur positive que nous noterons $f(A)$, où f est typiquement appelée fonction de fitness.*

La fitness (efficacité) est donc donnée par une fonction à valeurs positives réelles. Dans le cas d'un codage binaire, nous utiliserons souvent une fonction de décodage d qui permettra de passer d'une chaîne binaire à un chiffre à valeur réelle : $d : \{0,1\}^l \rightarrow \Re$ (où l est la longueur de la chaîne). La fonction de fitness est alors choisie telle qu'elle transforme cette valeur en valeur positive, soit $f : d(\{0,1\}^l) \rightarrow \Re_+^*$. Le but d'un algorithme génétique est alors simplement de trouver la chaîne qui maximise cette fonction f. Bien évidemment, chaque problème particulier nécessitera ses propres fonctions d et f.

Les AGs sont alors basés sur les phases suivantes :

1. **Initialisation**. Une population initiale de N chromosomes est tirée aléatoirement.
2. **Évaluation**. Chaque chromosome est décodé, puis évalué.
3. **Sélection**. Création d'une nouvelle population de N chromosomes par l'utilisation d'une méthode de sélection appropriée.
4. **Reproduction**. Possibilité de croisement et mutation au sein de la nouvelle population.
5. **Retour** à la phase d'évaluation jusqu'à l'arrêt de l'algorithme.

Voyons maintenant plus en détail les autres phases de l'algorithme génétique. Nous présentons ces opérateurs sous l'hypothèse implicite que le codage est binaire. La section 2.6 abordera le codage réel.

2.1 Codage et population initiale

Il existe trois principaux type de codage : binaire, *gray* ou réel. Nous pouvons facilement passer d'un codage à l'autre (voir Michalewicz (1992)). Certains auteurs n'hésitent pas à faire le parallèle avec la biologie et parlent de génotype en ce qui concerne la représentation binaire d'un individu, et de phénotype pour ce qui est de sa valeur réelle correspondante dans l'espace de recherche.

Rappelons que la transformation la plus simple (fonction de décodage d) d'une chaîne binaire A en nombre entier x s'opère par la règle suivante :

$$x = d(A) = \sum_{i=1}^{l} a_i 2^{l-i-1} \tag{1}$$

Ainsi le chromosome $A = \{1,0,1,1\}$ vaut $1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = 8+2+1 = 11$.

Évidemment, la fonction d sera modifiée selon le problème. Ainsi si nous cherchons à maximiser une fonction $f : [0,1] \rightarrow [0,1]$ une méthode possible serait la suivante (la taille du chromosome dépendant bien évidemment de la précision voulue) :

$$x = d(A) = \sum_{i=1}^{l} a_i 2^{-i-1} \tag{2}$$

Pour une précision au cinquième chiffre après la virgule nous prenons alors $l = 16$ puisque $d(\{\underbrace{1,..,1,..,1}_{16}\}) = 0.999992 \rightarrow 1$. Une autre façon de faire est de choisir d telle que :

$$x = d(A) = \sum_{i=1}^{l} \frac{a_i 2^{l-i-1}}{2^{l+1}-1} \tag{3}$$

Avec $l = 16$ nous avons $2^{17} - 1 = 131071$ et $d\{\underbrace{1,..,1}_{16}\}) = \frac{131071}{131071} = 1$. La précision est vérifiée puisque : $d\{\underbrace{0,..,0}_{14}10\}) = \frac{2}{131071} = 0.0000152589$.

Cette dernière règle peut se généraliser. Ainsi, admettons que nous cherchons à maximiser f en fonction d'un variable réelle x. Soit $D = [x_{min}, x_{max}]$, avec $D \subset \Re$, l'espace de recherche permis avec x_{min} et x_{max} les bornes inférieures et supérieures. Soit *prec* la précision (chiffre après la virgule) avec laquelle nous cherchons x. Soit $ld = x_{max} - x_{min}$ la longueur de l'intervalle D. Nous devons alors diviser cet intervalle en $n_i = ld * 10^{prec}$ sous-intervalles égaux afin de respecter la précision. Par exemple, soit $D = [-1, 2]$, nous avons donc $ld = 3$, si nous voulions une précision $prec = 6$, alors il nous faut diviser cet intervalle en $n_i = 3000000$ sous-intervalles.

Avec s l'entier naturel tel que $2^s > n_i$ (dans notre exemple, $s = 22$ car $2^{21} = 2097152 < 3000000 < 2^{22} = 4194304$), la transformation d'une chaîne binaire $A = \{a_1, ..., a_s\}$ en un nombre réel x peut alors s'exécuter en deux étapes :

1. conversion (base 2 en base 10) : $x' = \sum_{i=1}^{s} 2^{i-1}$;
2. recherche du nombre réel correspondant :

$$x = x_{min} + x' \frac{x_{max} - x_{min}}{2^s - 1}$$

ou ce qui revient au même directement en une seule étape par :

$$x = x_{min} + \sum_{i=1}^{s} \frac{2^{i-1} ld}{2^s - 1}$$

Pour ce qui est de la phase d'initialisation, la procédure est assez simple. Elle consiste en un tirage aléatoire de N individus dans l'espace des individus permis. En codage binaire, selon la taille l de la chaîne, nous effectuons pour un chromosome l tirage dans $\{0, 1\}$ avec équiprobabilité.

2.2 Les opérateurs

Les opérateurs jouent un rôle prépondérant dans la possible réussite d'un AG. Nous en dénombrons trois principaux : l'opérateur de sélection, de croisement et de mutation. Si le principe de chacun de ces opérateurs est facilement compréhensible, il est toutefois difficile d'expliquer l'importance isolée de chacun de ces opérateurs dans la réussite de l'AG. Cela tient pour partie au fait que chacun de ces opérateurs agit selon divers critères qui lui sont propres (valeur sélective des individus, probabilité d'activation de l'opérateur, etc.).

2.2.1 Opérateur de Sélection

Cet opérateur est peut-être le plus important puisqu'il permet aux individus d'une population de survivre, de se reproduire ou de mourir. En règle général, la probabilité de survie d'un individu sera directement reliée à son efficacité relative au sein de la population.

Il existe plusieurs méthodes pour la reproduction. La méthode la plus connue et utilisée est sans nul doute, la roue de loterie biaisée (*roulette wheel*) de Goldberg (1989). Selon cette méthode, chaque chromosome sera dupliquée dans une nouvelle population proportionnellement à sa valeur d'adaptation. On effectue, en quelque sorte, autant de tirages avec remises qu'il y a d'éléments dans la population. Ainsi, dans le cas d'un codage

binaire, la fitness d'un chromosome particulier étant $f(d(c_i))$, la probabilité avec laquelle il sera réintroduit dans la nouvelle population de taille N est :

$$\frac{f(d(c_i))}{\sum_{j=1}^{N} f(d(c_j))}$$

Les individus ayant une grande fitness ont donc plus de chance d'être sélec tionnés. On parle alors de sélection proportionnelle.

L'inconvénient majeur de cette méthode repose sur le fait qu'un individu n'étant pas le meilleur peut tout de même dominer la sélection. Elle peut aussi engendrer une perte de diversité par la domination d'un super individu. Un autre inconvénient est sa faible performance vers la fin quand l'ensemble des individus se ressemblent. Dawid (1999, p.39) résume très bien tous ces inconvénients :

> Pensez à une situation où une chaîne [chromosome pour nous] de la population a comparativement une fitness élevée mais n'est pas optimal ou proche de l'optimum. Disons que la fitness de cette chaîne est dix fois plus grande que la fitness moyenne. [...] il pourrait facilement arriver, après quelques générations, que la population ne soit entièrement constituée que de cette chaîne. Dans un tel cas, l'algorithme génétique n'évoluera plus et l'optimum ne sera pas trouvé. Ce phénomène est appelé "convergence prématurée" et est l'un des problèmes les plus fréquents lors de l'utilisation des algorithmes génétiques. Un autre problème issu de la sélection proportionnelle est celui du "fine tuning" à la fin de la recherche.

Une solution à ce problème ne tient pas dans l'utilisation d'une autre méthode de sélection mais dans l'utilisation d'une fonction de fitness modifiée. Ainsi, nous pouvons utiliser un changement d'échelle (*scaling*) afin de diminuer ou accroître de manière artificielle l'écart relatif entre les fitness des individus.

Brièvement, il existe d'autres méthodes, la plus connue étant celle du tournoi (*tournament selection*) : on tire deux individus aléatoirement dans la population et on reproduit le meilleur des deux dans la nouvelle population. On refait cette procédure jusqu'à ce que la nouvelle population soit complète. Cette méthode donne de bons résultats. Toutefois, aussi important que soit la phase de sélection, elle ne crée pas de nouveaux individus dans la population. Ceci est le rôle des opérateurs de croisement et de mutation.

2.2.2 Opérateur de Croisement

L'opérateur de croisement permet la création de nouveaux individus selon un processus fort simple. Il permet donc l'échange d'information entre les chromosomes (individus). Tout d'abord, deux individus, qui forment alors un couple, sont tirés au sein de la nouvelle population issue de la reproduction. Puis un (potentiellement plusieurs) site de croisement est tiré aléatoirement (chiffre entre 1 et $l-1$). Enfin, selon une probabilité p_c que le croisement s'effectue, les segments finaux (dans le cas d'un seul site de croisement) des deux parents sont alors échangés autour de ce site (voir figure 2).

Cet opérateur permet la création de deux nouveaux individus. Toutefois, un individu sélectionné lors de la reproduction ne subit pas nécessairement l'action d'un croisement. Ce dernier ne s'effectue qu'avec une certaine probabilité. Plus cette probabilité est élevée et plus la population subira de changement.

Quoi qu'il en soit, il se peut que l'action conjointe de la reproduction et du croisement soit insuffisante pour assurer la réussite de l'AG. Ainsi, dans le cas du codage binaire, certaines informations (i.e. caractères de l'alphabet) peuvent disparaître de la population. Ainsi aucun individu de la population initiale ne contient de 1 en dernière position de la chaîne, et que ce 1 fasse partie de la chaîne optimale à trouver, tous les croisements possibles ne permettront pas de faire apparaître ce 1 initialement inconnue. En codage réel, une telle situation peut arriver si utilisant un opérateur simple de croisement, il se trouvait qu'initialement toute la population soit comprise entre 0 et 40 et que la valeur optimale était de 50. Toutes les combinaisons convexes possibles de chiffres appartenant à l'intervalle $[0,40]$ ne permettront jamais d'aboutir à un chiffre de 50. C'est pour remédier entre autre à ce problème que l'opérateur de mutation est utilisé.

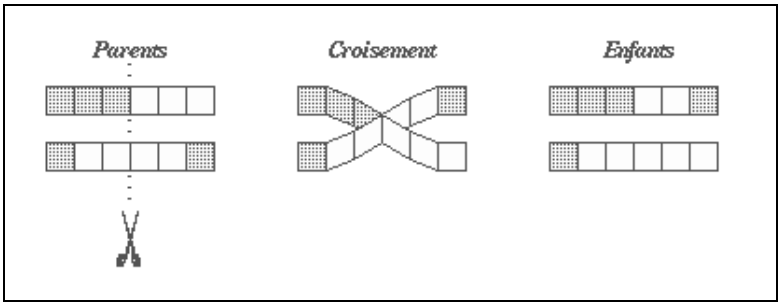

FIG. 2: Le croisement en codage binaire

2.2.3 Opérateur de Mutation

Le rôle de cet opérateur est de modifier aléatoirement, avec une certaine probabilité, la valeur d'un composant de l'individu. Dans le cas du codage binaire, chaque bit $a_i \in \{0,1\}$ est remplacé selon une probabilité p_m par son inverse $a_i^{'} = 1 - a_i$. C'est ce qu'illustre la figure 3. Tout comme plusieurs lieux de croisement peuvent être possibles, nous pouvons très bien admettre qu'une même chaîne puisse subir plusieurs mutations.

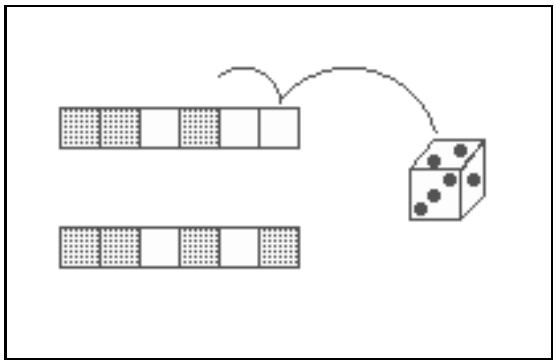

FIG. 3: La mutation en codage binaire

La mutation est traditionnellement considérée comme un opérateur marginal bien qu'elle confère en quelque sorte aux algorithmes génétiques la propriété d'ergodicité (i.e. tous les points de l'espace de recherche peuvent être atteints). Cet opérateur est donc d'une grande importance. Il a de fait un double rôle : celui d'effectuer une recherche locale et/ou de sortir d'une trappe (recherche éloignée).

2.3 Autres paramètres

Les opérateurs de l'algorithme génétique sont guidés par un certain nombre de paramètres fixés à l'avance. La valeur de ces paramètres influence la réussite ou non d'un algorithme génétique. Ces paramètres sont les suivants :

- La taille de la population, N, et la longueur du codage de chaque individu l (dans le cas du codage binaire). Si N est trop grand le temps de calcul de l'algorithme peut s'avérer très important, et si N est trop petit, il peut converger trop rapidement vers un mauvais chromosome. Cette importance de la taille est essentiellement due à la notion de *parallélisme implicite* qui implique que le nombre d'individus traité par l'algorithme est au moins proportionnelle au cube du nombre d'individus.
- La probabilité de croisement p_c. Elle dépend de la forme de la fonction de fitness. Son choix est en général heuristique (tout comme pour p_m). Plus elle est élevée, plus la population subit de changements importants. Les valeurs généralement admises sont comprises entre 0.5 et 0.9.
- La probabilité de mutation p_m. Ce taux est généralement faible puisqu'un taux élevé risque de conduire à une solution sous-optimale.

Plutôt que de réduire p_m, une autre façon d'éviter que les meilleurs individus soient altérés est d'utiliser la reconduite explicite de l'élite dans une certaine proportion. Ainsi, bien souvent, les meilleurs 5%, par exemple, de la population sont directement reproduits à l'identique, l'opérateur de reproduction ne jouant alors que sur les 95% restant. Cela est appelée une stratégie élitiste.

Partant du constat que les valeurs des paramètres des différents opérateurs sont eux-mêmes inconnus et ne peuvent être améliorés au fur et à mesure que de façon expérimental, certains auteurs, tels Novkovic et Sverko (1997), proposent d'utiliser une sorte de méta-AG : l'un pour trouver l'individu optimal et l'autre pour trouver la valeur optimale des paramètres. Ces deux algorithmes tourneraient alors simultanément ou séquentiellement. Toutefois, il est inévitable que le temps de calcul s'alourdisse en conséquence.

2.4 Remarque sur la fonction fitness

Le choix de la fonction de fitness retenue est important et dépend du problème à résoudre et de l'espace de recherche qui en découle. Admettons que l'on cherche simplement à maximiser la fonction : $f(x) = x^2$ avec $x \in [0,10]$. Dans un tel cas, la fonction de fitness coïncide avec celle du problème. Pour des problèmes de minimisation simples, tel $\min(x-3)^2$, nous utiliserons en général soit la propriété que : $\max f(x) = -min f(x)$ ou bien, si la fonction est bornée supérieurement la fonction : $\max C - f(x)$, où C est une constante positive supérieure à cette borne.

Cependant l'espace de recherche, appelons le S, est généralement constitué de deux sous-espaces disjoints : l'espace des solutions admissibles F et l'espace des solutions non admissibles U. De nombreux problèmes de programmation linéaire ou non linéaire n'échappent pas à ce problème. A tout moment, en cherchant un maximum faisable, l'algorithme génétique peut au cours du processus de recherche créée des solutions non admissibles, solutions qui violeraient au moins l'une des contraintes. Il n'est jamais simple de traiter ces problèmes. La solution passe en général par l'utilisation d'une fonction de fitness à pénalité. L'efficacité d'une solution non admissible est automatiquement réduite. Toutefois, le choix de cette fonction de pénalité soulève des questions : comment deux solutions non admissibles doivent-elles être comparées ? Devons-nous considérer que toute solution admissible est préférable à une solution non admissible ? Devons-nous automatiquement supprimer les solutions non admissibles de la population ? Pouvons-nous par une fonction dite de réparation changée une solution non admissible en une admissible ? etc. Toutes ces questions reviennent à poser celle de l'utilisation de solutions non admissibles pour leurs potentiels d'information concernant la recherche de l'optimum : une solution non admissible pouvant être plus proche de la solution optimal que de nombreuses autres solutions admissibles. Toutes ces questions n'ont pas encore de solution unanime (voir Michalewicz (1995) pour un résumé).

2.5 Un exemple simple

Nous reprenons ici l'exemple de Goldberg (1989). Il consiste à trouver le maximum de la fonction $f(x) = x$ sur l'intervalle $[0,31]$ où x est un entier. La première étape consiste à coder la fonction. Par exemple, nous utilisons un codage binaire de x, la séquence (chromosome) contenant au maximum 5 bits. Ainsi, nous avons $x = 2 \rightarrow \{0,0,1,0\}$, de même $x = 31 \rightarrow \{1,1,1,1,1\}$. Nous recherchons donc le maximum d'une fonction de fitness dans un espace de 32 valeurs possibles de x.

2.5.1 Tirage et évaluation de la population initiale

Nous fixons la taille de la population à $N = 4$. Nous tirons donc de façon aléatoire 4 chromosomes sachant qu'un chromosome est composé de 5 bits, et chaque bit dispose d'une probabilité $\frac{1}{2}$ d'avoir une valeur 0 ou 1.

Le maximum, 16, est atteint par la deuxième séquence. Voyons comment l'algorithme va tenter d'améliorer ce résultat.

Numéro	Séquence	Fitness	% du total
1	00101	5	14.3
2	10000	16	45.7
3	00010	2	5.7
4	00110	12	34.3
Total		35	100

2.5.2 Sélection

Une nouvelle population va être créée à partir de l'ancienne par le processus de sélection de la roue de loterie biaisée.

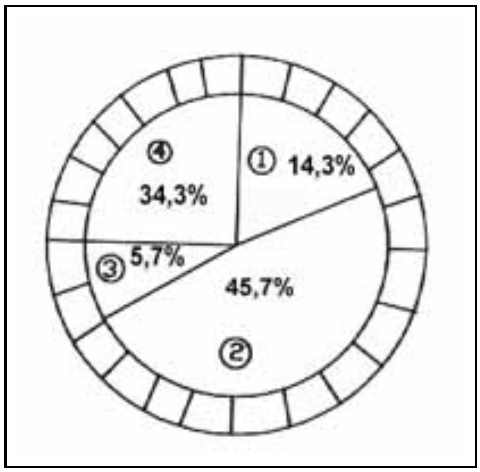

FIG. 4: La roue de loterie biaisée : opération de sélection

Nous tournons cette roue 4 fois et nous obtenons au final la nouvelle population décrite dans le tableau 1.

Numéro	Séquence
1	10000
2	01100
3	00101
4	10000

TAB. 1: Nouvelle Population

2.5.3 Le croisement

Les parents sont sélectionnés au hasard. Nous tirons aléatoirement un lieu de croisement (site ou *locus*) dans la séquence. Le croisement s'opère alors à ce lieu avec une probabilité p_c. Le tableau 2 donne les conséquences de cet opérateur en supposant que les chromosomes 1 et 3, puis 2 et 4 sont appariées et qu'à chaque fois le croisement s'opère (par exemple avec $p_c = 1$).

2.5.4 La mutation

Dans cet exemple à codage binaire, la mutation est la modification aléatoire occasionnelle (de faible probabilité) de la valeur d'un bit (inversion d'un bit). Nous tirons ainsi pour chaque bit un chiffre aléatoire entre 0 et 1 et si ce chiffre est inférieur à p_m alors la mutation s'opère. Le tableau 3, avec $p_m = 0.05$, met en évidence ce processus.

$l=3$	$l=2$
100|00	01|100
001|01	10|000
10001	01000
00100	10100

TAB. 2: Le croisement

Anc. Chr.	Tirage aléat.	Nveau Bit	Nveau Chr.
10001	15 25 36 **04** 12	1	100**1**1
00100	26 89 13 48 59	–	00100
01000	32 45 87 22 65	–	01000
10100	47 **01** 85 62 35	1	1**1**100

TAB. 3: La mutation

Maintenant que la nouvelle population est entièrement créée, nous pouvons de nouveau l'évaluer.

2.5.5 Retour à la phase d'évaluation

Numéro	chaîne	Fitness	% du total
1	10011	19	32.2
2	00100	4	6.8
3	01000	8	13.5
4	11100	28	47.5
Total		59	100

TAB. 4: Nouvelle évaluation

Le maximum est maintenant de 28 (séquence 4). Nous sommes donc passé de 16 à 28 après une seule génération. Bien sûr, nous devons recommencer la procédure à partir de l'étape de sélection jusqu'à ce que le maximum global, 31, soit obtenu, ou bien qu'un critère d'arrêt ait été satisfait.

2.6 Le codage réel

A l'aide du codage binaire, toutes les opérations sont assez simples à mettre en place. Malgré tout, quelques inconvénients existent (voir à ce sujet Michalewicz (1992) pour des exemples concrets) :

– Il peut être difficile d'adapter ce codage à certains problèmes :
  La représentation binaire traditionnelle utilisée pour les algorithmes génétiques crée des problèmes pour les problèmes d'optimisation de grandes dimensions à haute précision numérique. Par exemple, avec 100 variables appartenant au domaine $[-500,500]$ et dont une précision de 6 chiffres après la virgule est requise, la taille du chromosome est 3000. Cela, en retour, génère un espace de recherche d'environ 10^{1000}. Pour de tels problèmes, les algorithmes génétiques basés sur des représentations binaires ont de faibles performances.

– La distance de Hamming[1] entre deux nombres réels proches peut être grand (exemple : 0111 qui vaut 7 et 1000 qui vaut 8, la distance est de 4). Ce qui créé bien souvent une convergence mais non l'obtention de la valeur optimale.

[1] La distance de Hamming entre deux chaînes de bits est le nombre de bits qui diffèrent de l'une à l'autre. Ainsi entre 01100101 et 00101100 la distance de Hamming vaut 3.

– Suivant le problème, la résolution de l'algorithme peut être coûteux en temps.
– Le croisement et la mutation peuvent être inadaptés (création d'individus n'appartenant pas à l'espace de recherche).

Une des améliorations majeures consiste alors à se servir de nombres réels directement. Les résultats donnés par Michalewicz (1992) et Michalewicz, Logan et Swaminathan (1994) montrent que la représentation binaire aboutit souvent à une moins bonne précision et qu'en règle général le gain en termes de temps de calcul (CPU) est positif. La conclusion à laquelle il arrive est naturellement qu'une représentation plus naturelle du problème offre des solutions plus efficaces.

En utilisant le codage réel, notre individu n'est alors plus qu'un chiffre à valeurs réelles dans l'espace des valeurs permises : $A = a,\ a \in D \subset \Re$. L'opérateur de sélection reste identique à celui de la roue de loterie biaisée ou du tournoi. En revanche, il existe d'autres opérateurs de croisement et de mutation. Voyons lesquels (voir aussi Michalewicz et Michalewicz, Logan et Swaminathan (1994) pour une présentation de ces opérateurs à codage réel).

2.6.1 Opérateur de croisement

L'opération de croisement simple tel que décrit dans le cas binaire ne peut s'effectuer ici dans le cas de recherche d'un point unique. Toutefois, pour une recherche de plus grande dimension, nous pouvons utiliser de façon analogique cet opérateur. Ainsi, soient $Y = (y_1, y_2, y_3)$ et $X = (x_1, x_2, x_3)$ deux membres (vecteur de dimension trois) de la population initiale. Nous recherchons donc trois points dans un espace de recherche de dimension trois.

L'opération de croisement simple est identique dans le principe à celle décrite auparavant. Pour ce faire, nous générons un nombre aléatoire r à partir d'une distribution uniforme sur l'ensemble $\{1,2,3\}$, et deux nouveaux individus, $\tilde{X}$ et $\tilde{Y}$, sont créés selon la règle suivante :

$$\tilde{x}_i = \begin{cases} x_i, & \text{si } i < r \\ y_i, & \text{sinon} \end{cases} \tag{4a}$$

$$\tilde{y}_i = \begin{cases} y_i, & \text{si } i < r \\ x_i, & \text{sinon} \end{cases} \tag{4b}$$

Un autre opérateur est le croisement arithmétique (valable même pour une recherche de dimension un). Ce croisement effectue une simple combinaison linéaire entre les parents. Soit, après avoir généré un chiffre aléatoire, $\alpha = U(0,1)$, les nouveaux parents sont :

$$\tilde{X} = \alpha X + (1-\alpha)Y \tag{5a}$$

$$\tilde{Y} = (1-\alpha)X + \alpha Y \tag{5b}$$

Enfin, il existe aussi le croisement heuristique. Cet opérateur effectue une extrapolation linéaire des deux individus. Un nouvel individu, $\tilde{X}$, est créé selon le processus suivant (sous l'hypothèse que $X > Y$ en terme de fitness, sinon nous inversons X et Y dans les équations) :

$$\tilde{X} = X + r(X - Y),$$
$$\tilde{Y} = X.$$

et où :

$$\text{Faisabilité} = \begin{cases} 1, & \text{if } b_1^i < \tilde{x}_i < b_2^i,\ \forall i \\ 0, & \text{sinon.} \end{cases}$$

où b_1^i et b_2^i sont les bornes autorisées pour x_i, et avec r un nombre aléatoire tirée dans $U(0,1)$. Nous devons donc avoir tout le temps $x_i \in [b_1^i, b_2^i]$. Si $\tilde{X}$ n'est pas faisable (i.e. faisabilité nulle) alors un nombre r est retiré et la procédure est recommencée jusqu'à ce que la solution soit faisable où qu'un certain nombre d'essais ait été effectué. Dans le cas où $f(X) = f(Y)$ (même fitness) on reproduit simplement X et Y. Cet opérateur est le seul des croisements à utiliser *directement* une information reliée à la fitness. Comme nous le disent Michalewicz, Logan et Swaminathan (1994) :

> Cet opérateur est un croisement unique pour les raisons suivantes : (1) il utilise les valeurs de la fonction objectif afin de déterminer une direction de recherche, (2) il produit seulement un enfant et (3) il peut ne produire aucun enfant. [...] Il semble que le croisement heuristique contribue à trouver une solution plus précise ; ses principales responsabilités [dans la recherche de la solution] sont (1) un fine tuning local et (2) une recherche dans une direction prometteuse.

2.6.2 Opérateur de mutation

La mutation uniforme est identique à celle du codage binaire. Ainsi, chaque variable $x_i \in X$ est changée selon une certaine probabilité en un nombre aléatoire tiré dans une distribution uniforme sur l'intervalle $[b_1^i, b_2^i]$, avec b_1^i et b_2^i les bornes inférieures et supérieures pour x_i.

La mutation non uniforme revient à changer la variable x_i en un nombre tiré dans une distribution non uniforme. Cette nouvelle variable $\tilde{x}_i$ est telle que :

$$\tilde{x}_i = \begin{cases} x_i + (b_2^i - x_i) f(G), & \text{si } \alpha < 0.5, \\ x_i - (x_i + b_1^i) f(G), & \text{si } \alpha \geq 0.5. \end{cases} \tag{6}$$

avec

$$\begin{aligned} f(G) &= (\tilde{\alpha}(1 - \frac{G}{G_{max}}))^b, \\ \alpha, \tilde{\alpha} &= \text{nombres aléatoires} \in (0,1), \\ G &= \text{la génération courante}, \\ G_{max} &= \text{la nombre maximum de génération (i.e. de création de nouvelle population)}, \\ b &= \text{un paramètre déterminant le degré de non uniformité}. \end{aligned}$$

Un dernier opérateur de mutation existe : la mutation dans les bornes. Avec cet opérateur, chaque variable $x_i \in X$ choisie pour muter prend pour valeur l'une des deux bornes b_1^i ou b_2^i avec équiprobabilité. A l'évidence, cet opérateur n'a d'intérêt et d'efficacité que si la solution est proche des bornes de l'espace de recherche.

Notons qu'il est possible de combiner plusieurs opérateurs en même temps.

3 Applications des Algorithmes Génétiques

Cette section présente des applications des AGs notamment en économie et la finance. Sans prétendre à l'exhaustivité, on peut néanmoins classer ces applications autour de deux grands thèmes : l'utilisation des AGs comme outil d'optimisation et de prévision (section 3.1), et l'utilisation des AGs comme représentation de l'apprentissage (section 3.2).

3.1 Optimisation et prévision

Dans cette sous-section, nous allons nous intéresser principalement à l'application des AGs dans trois domaines : résolution numérique, l'économétrie et la finance. Le point commun est l'utilisation des AGs comme un simple "algorithme de calcul".

3.1.1 Résolution numérique

Les AGs, à l'inverse des méthodes traditionnelles de résolutions numériques de type gradient ne sont pas basés sur une approche analytique mais sur une approche itérative et heuristique. En cela, peu d'information est nécessaire pour leur utilisation : l'espace de recherche possible et un critère d'efficacité. Les ingénieurs et les spécialistes en recherche opérationnelle ont très vite vu le potentiel de l'utilisation des AGs comme outil de résolution numérique. Il s'en est suivit, depuis le début des années 90, une liste importante de papiers sur le

sujet comparant les AGs à d'autres méthodes de résolution numérique, quelles soient analytiques ou aléatoires (cf. entre autre Zhu et al. (1997), Baluja (1995) et surtout Michalewicz et al. (1992)).

La complexité des problèmes en économie ne cessant de se développer, c'est tout naturellement que les économistes se sont mis eux aussi à utiliser ces algorithmes. Ainsi Dorsey et Mayer (1995) ont montré la potentialité des AGs pour résoudre numériquement certains problèmes d'optimisation difficiles présentant des non-différentiabilités, des multi-modalités ou encore des discontinuités. Cette étude fut reprise par Östermark (1999) qui propose d'utiliser une version dite "hybride" des AGs afin d'améliorer, entre autre, les résultats de Dorsey et Mayer (1995). Cet AG hybride se distingue par : a) l'ajout d'une méthode classique de type Newton si le problème est différentiable, ce qui permet d'améliorer l'efficacité des opérateurs de croisement et de mutation ; b) une évolution dynamique de l'espace de recherche, ce qui doit permettre d'accélérer la convergence vers la solution optimale ; et c) une meilleure prise en compte des relations de contrainte entre les variables, ce qui doit permettre de réduire la dimension de l'espace de recherche. L'auteur conclue sur l'efficacité de cet AG hybride en soulignant cependant que les problèmes testés ne sont pas d'une "taille" (i.e. dimension vectorielle) très importante.

Dans une même logique, Beaumont et Bradshaw (1995) proposent d'utiliser les algorithmes génétiques afin de résoudre des problèmes non-linéaires tels que les modèles de croissance optimale. Afin d'éviter les problèmes de convergence prématurée vers un minimum local, les auteurs développent et utilisent une version dite "distribuée parallèlement" de l'algorithme génétique[2]. Par le biais de la résolution d'un modèle de croissance optimale déterministe à horizon infini et fini, les auteurs comparent l'efficacité de leur AG distribué parallèlement avec une méthode plus traditionnelle – la méthode de projection appelée méthode de Galerkin – afin de retrouver les fonctions de récurrence qui résolvent les équations d'Euler correspondantes du modèle. Tout en acceptant la conclusion que la vitesse de résolution de l'AG est bien plus lente et que les résultats ne sont pas forcément plus précis, les auteurs estiment néanmoins que la résolution par cet AG est bien plus facile à mettre en place. En outre, il ne fait aucun doute pour eux que dans le cadre de modèles plus complexes, cet AG surpassera la méthode de Galerkin.

Toujours dans le cadre de la résolution de modèle de croissance optimale, mais cette fois stochastique, Duffy et McNellis (2001) arrivent eux aussi à la conclusion de l'efficacité des AGs. En effet, cherchant à résoudre directement les équations d'Euler d'un modèle de croissance stochastique par le biais de la méthode PEA[3], Duffy et McNellis comparent deux méthodes d'approximation de la fonction à paramétrer : la première utilise des réseaux de neurones combinés avec un AG hybride[4] tandis que la seconde, plus traditionnelle, utilise une expansion polynômiale optimisée par une méthode de gradient. Les résultats obtenus sont suffisamment précis pour que les auteurs concluent sur la potentialité de cette méthode alternative.

En théorie des jeux différentiels, les applications principales concernent la recherche numérique de l'équilibre de Nash ou de Stackelberg. Ainsi Özyildirim (1997) et Alemdar et Özyildirim (1998) s'intéressent à la résolution d'un jeu différentiel de commerce international Nord-Sud avec pollution. Se basant sur la propriété que dans un jeu différentiel à n joueurs un équilibre de Nash en boucle ouverte peut être obtenu comme une solution jointe de n problèmes de contrôle optimal (cf Başar et Olsder 1982), ils utilisent n algorithmes génétiques parallèles pour résoudre le problème. Le codage est binaire et la procédure élitiste. L'algorithme utilisé semble obtenir de bons résultats. Dans une même logique de recherche d'une stratégie optimale, Vallée et Başar (1999b) ont montré que l'utilisation des AGs pouvaient permettre de trouver des stratégies de Stackelberg en boucle fermée dans des jeux différentiels.

Au final, l'ensemble de ces travaux souligne donc l'efficacité des AGs comme outil de résolution numérique. Tous cependant mettent en avant une limite : si les AGs convergent vers une solution optimale rien ne permet de dire, quand cette solution est inconnue, que le résultat soit la solution optimale parfaite. En outre, les AGs peuvent rester longtemps proche de la solution optimale sans l'atteindre. C'est la raison pour laquelle de nombreuses méthodes dites hybrides, combinant AG et méthode traditionnelle de gradient, sont de plus en

[2]Cette approche divise virtuellement la population initiale de l'AG en différentes sous-populations travaillant en parallèle. Tout se passe comme si plusieurs AGs cherchaient la solution optimale pendant un certain temps puis s'échangeaient leurs résultats.

[3]Parameterized expectations algorithm : algorithme de résolution numérique développé par Marcet (1988) et Den Haan et Marcet (1990) afin de résoudre les modèles de croissance stochastique.

[4]Un algorithme de gradient (quasi-Newton) est effectué sur le meilleur chromosome à la fin de l'AG afin de faire un "fine tuning".

plus utilisées. Enfin, la durée de calcul (temps CPU) peut être longue.

3.1.2 Économétrie et séries temporelles

Une voie de recherche actuelle et future à grand potentiel porte sur l'application des algorithmes génétiques pour des problèmes économétriques. Dans cette optique, l'utilisation des AGs est très diverse : recherche d'une forme fonctionnelle, des valeurs des coefficients de régression, etc.

Ainsi, Pan et al. (1995) utilisent les AGs afin de trouver les paramètres optimaux de régression non-linéaires. L'efficacité des AGs est comparée avec d'autres méthodes de résolution numérique à l'aide de plusieurs modèles non-linéaires. Les auteurs montrent que les AGs sont tout autant performant tout en demandant cependant nettement moins d'information analytique.

Dans un autre ordre d'idée, Boné et al. (1998) utilisent les AGs pour trouver la forme économétrique théorique la mieux adaptée à la modélisation de séries temporelles. Chez eux, la population de chromosomes définit le type de modélisation (*AR*, *MA* ou *ARMA*) et la valeur des coefficients associés. Si les AGs fournissent une estimation correcte des séries temporelles, les auteurs pensent que l'utilisation de ces algorithmes peut surtout être bénéfique afin d'initialiser correctement des procédures plus classiques. Dans un même registre de prévision de séries temporelles, Slimane et al. (1998) utilise un modèle d'AG hybride, appelé GHOSP, qui incorpore une recherche de type gradient au sein de la boucle principale de l'AG. Ici, la prévision est effectuée à l'aide de modèles stochastiques de type *Hidden Markov*. L'algorithme, quant à lui, doit rechercher les coefficients et l'architecture optimale de ce modèle de prévision stochastique. L'algorithme est utilisé pour la prévision de modèle simple (de type sinus) et de modèles plus complexes (comme l'évolution d'obligations). Les résultats des auteurs sont encourageants et montrent qu'une application en temps réel est certainement envisageable.

Weiss (1999) et Weiss et Hirsh (1998,1999,2000) utilisent, dans une optique proche du *data mining*, les AGs afin de prévoir des événements rares. La prévision de telles événements est d'une importance élevée pour certaines activités : prévision d'utilisation frauduleuse de cartes de crédit à partir d'un historique d'achats, prévision de comportements inhabituels sur un marché financier, prévision de casses ou d'échecs de fonctionnement d'équipements téléphoniques ou électroniques à partir d'une série d'alarmes, etc. Pour réaliser de telle prévision, les auteurs ont mis au point un système d'apprentissage basé sur les algorithmes génétiques appelé *Timeweaver*. A partir d'une série historique de données, l'algorithme recherche et construit des règles de prévision permettant de savoir si un événement rare est susceptible d'apparaître dans un futur proche. Les auteurs montrent dans leurs différents articles tout le potentiel de cet algorithme, notamment en le comparant (Weiss (1999), Weiss et Hirsh (2000)) avec d'autres méthodes probabilistes. Plus précisément, *Timeweaver* est utilisé pour la prévision d'échecs d'équipements téléphoniques construite à partir d'une série de données d'alarmes, ou encore pour la prévision de la prochaine commande Unix à partir d'une série historique de commandes Unix utilisées. Dans les deux exemples, les résultats sont prometteurs et de futures recherches par les auteurs sont attendus.

Sans nul doute, une voie de recherche en cours promis à un grand avenir est celle basée sur l'utilisation d'algorithmes génétiques non restreints appelés programmation génétique[5], pour rechercher de formes fonctionnelles reproduisant le mieux une série de données. Ainsi, plutôt que de chercher les coefficients optimales d'un modèle donné (souvent linéaire) pour reproduire une série empirique, Koza (1991) recherche la forme fonctionnelle (potentiellement non linéaire) et ses coefficients optimaux. A partir de données sur la monnaie, M, la production, Q, la vitesse de circulation de la monnaie V et les pris aux USA, P, il retrouve la relation fonctionnelle de la théorie quantitative, à savoir une relation non-linéaire : $P = \frac{MV}{Q}$. Koza aboutit à des résultats très significatifs. Dans une même logique, Szpiro (1997a, 1997b) et Beenstock et Szpiro (1999) utilise aussi les AGs afin de rechercher les formes fonctionnelles non-linéaires reproduisant au mieux une série de données. Szpiro (1997b) montre cependant que si les AGs sont capables de trouver des formes fonctionnelles

[5]La programmation génétique est similaire à l'algorithme génétique dans son fonctionnement tout en étant moins restrictive. La différence principale repose sur le fait que la taille du chromosome n'est pas fixe. Pour une application de la programmation génétique à la finance, le lecteur peut aller voir la page web :
http ://alphard.ethz.ch/hafner/ggp/ggpthings.htm

"solutions" donnant de bon résultats, toutes ne sont pas d'une haute pertinence économique. Il restera donc toujours à l'utilisateur de guider la recherche à un moment ou un autre.

3.1.3 Finance

Les applications en finance se sont fortement développées ces dernières années (voir Pereira (2000) pour un tour d'horizon récent) et commencent à intégrer les livres de finance (Bauer (1994), Ruggiero (1997)). La raison d'un tel développement, comme le souligne Pereira (2000), est évidente :

> Genetic algorithms are a valid approach to many practical problems in finance which can be complex and thus require the use of an efficient and robust optimisation technique. Some applications of genetic algorithms to complex problems in financial markets include : forecasting returns, portfolio optimisation, trading rule discovery, and optimisation of trading rules.

Eddelbüttel (1992, 1996) utilise dans le cadre du management passif les algorithmes génétiques afin de trouver des portefeuilles qui reproduisent celui de l'indice DAX (définit comme la solution optimale à atteindre). Ainsi, sachant qu'un portefeuille est composé des 30 actions de l'indice DAX, l'AG utilisé doit gérer les poids relatifs de ces actions dans l'indice (chaque chromosome est un vecteur de poids possible). Autre possibilité, l'AG doit reconstituer l'évolution de l'indice DAX à l'aide uniquement d'un sous-ensemble d'actions le constituant. Ici, chaque chromosome définit un sous-ensemble d'actions retenu. Au final, Eddelbüttel (1992,1996) montre toute l'efficacité computationnelle des AGs pour ce type d'approche.

Dans une même logique, Loraschi et al. (1995, 1996) utilisent les AGs afin de choisir un portefeuille optimal. L'AG doit trouver les poids des actions dans un portefeuille qui minimisent un certain niveau de risque étant donné un niveau attendu de rendement espéré. Les auteurs concluent sur l'efficacité de la méthode notamment au regard de la prise en compte de l'existence possible d'équilibres multiples.

Mahfoud et Mani (1996) étudient l'utilisation des AGs comme outil de prévision des performances futures d'actions individuelles. Pour ce faire, l'AG utilisé doit chercher des règles optimales de prévision[6] de l'évolution future du cours d'une action en fonction de son évolution passée. Pour l'expérience principale, les auteurs utilisent une base de données de plus de 1600 actions. A chaque fin de semaine, une prévision de l'évolution du cours (haut/bas) ainsi que de la magnitude de cette évolution est effectuée pour les trois semaines à venir. A la fin de 12 semaines, les prévisions sont comparées avec la réalité. L'efficacité de leur AG est comparé avec un modèle de prévision utilisant les réseaux de neurones. Si l'AG donne de meilleurs résultats, les auteurs montrent néanmoins qu'un modèle de prévision mixte améliore nettement l'ensemble des résultats (gain relatif de 20% par rapport à l'AG seul et de 50% par rapport aux réseaux de neurones seuls).

Une autre utilisation des AGs assez proche de la précédente concerne la découverte de règles optimales d'échanges (*trading rule*[7]) que ce soit sur le marché des actions (Allen et Karjalainen (1999) pour l'indice *S&P 500*) ou le marché des changes (Neely et al. (1997)). Les résultats sont assez mitigés. Si Allen et Karjalainen (1999) concluent sur la faible évidence de gains dus aux règles trouvées, Neely et al. (1997) montrent que les règles découvertes apportent une opportunité de gains.

Enfin, dans une autre optique, Varreto (1998) utilise lui les AGs pour prévoir les risques de banqueroute en Italie. L'auteur compare les AGs à une méthode statistique plus traditionnelle concernant la classification et la prédiction des banqueroutes : la LDA (*linear discriminant analysis*). Cette étude fait suite à une autre étude par le même auteur comparant la LDA avec une méthode basée sur l'utilisation des réseaux de neurones. Ici, les AGs sont utilisés dans deux finalités distinctes : la création de fonctions linéaires optimales et la création de règles optimales. Bien que concluant sur une relative supériorité de la LDA, l'auteur rappelle néanmoins que les résultats obtenus par les AGs le sont plus rapidement et pour moins de contraintes.

En plus de leur puissance technique, dans une approche plus heuristique, les AGs peuvent être utilisés pour représenter l'apprentissage adaptatifs des agents avec une rationalité limitée.

---

[6] Comme : **SI** [*prix <15 ET Volatilité>7% Et ...*] **ALORS** *Prévision=Haut*.

[7] Une règle d'échange est une règle explicite qui transforme un signal quelconque (évolution du cours d'une action ou du change, évolution de certaines variables financières, macroéconomiques, etc..) en une recommandation sur le marché financier (achat, vente, position courte, position longue, etc.).

3.2 Représentation de l'apprentissage

Les AGs correspondent en définitive à un algorithme d'exploration d'un espace de stratégies. Or, l'exploration devient une dimension importante de la dynamique économique dès que l'on quitte le cadre des anticipations rationnelles. La question de l'adaptation des anticipations et des choix des agents à l'évolution de leur environnement se pose alors. Du fait des mécanismes exploratoires simples et faciles à interpréter qu'ils proposent, les AGs peuvent être utilisés pour représenter ce processus adaptatif. Ils constituent alors une solution très intéressante pour un problème fondamental des modèles économiques avec des agents à rationalité limitée : la représentation de l'apprentissage des agents. Cette représentation tient de plus compte d'une dimension importante de l'activité économique : l'hétérogénéité des processus d'apprentissage et des anticipations des agents. Un ensemble très divers de des travaux font recours aux AGs sur ce point et cela, aussi bien en macroéconomie qu'en micro-économie.

Dans les paragraphes qui suivent nous allons donner des exemples représentatifs de ce que l'utilisation des AGs peut apporter à la compréhension de la dynamique économique. Nous commençons par des modèles macroéconomiques où les AGs ont principalement été utilisés pour représenter l'apprentissage social. La même utilisation en est aussi faite dans les modèles de théorie des jeux et de dynamique d'ajustement des marchés. Les modèles de la finance et de l'économie de l'innovation souligne l'intérêt d'utiliser les AGs pour représenter les processus d'apprentissage individuel.

3.2.1 Dynamique macroéconomique

En cherchant à dépasser la particularités des résultats liés aux anticipations adaptatives peu sophistiquées, la révolution des *Nouveaux Classiques* a mis les anticipations rationnelles au coeur de la modélisation macroéconomique. Tout en permettant d'appréhender les propriétés à long terme des économies, cette approche a néanmoins éliminé une dimension empirique importante de la dynamique économique : l'ajustement des anticipations et des comportements des agents. La nature auto-référencielle des modèles aux anticipations rationnelles a aussi fait apparaître un problème supplémentaire : la multiplicité des équilibres.

Il devient alors important de tester dans quelle mesure les équilibres à anticipations rationnelles sont robustes face aux tâtonnements des agents et de comprendre le rôle que ce tâtonnement peut jouer dans la sélection d'équilibres quand ils sont multiples. Les AGs ont alors été utilisés par plusieurs travaux pour représenter l'apprentissage des agents qui est à la source de ce type de tâtonnements (Sargent(1993)).

Arifovic(1995) utilise les AGs dans le cadre d'un modèle à générations imbriquées avec des agents qui vivent deux périodes et avec deux types de politiques monétaires : offre de monnaie constante ou financement constant du déficit avec le seugneriage. Dans le modèle initial avec prévisions parfaites, la première politique conduit à un équilibre stationnaire unique où la monnaie a de la valeur et à un continuum d'équilibres. Dans le premier cas la monnaie a de la valeur mais cet équilibre n'est pas stable. Le continuum converge vers un équilibre où la monnaie n'a pas de valeur. La seconde politique conduit à deux équilibres stationnaires : un avec une inflation faible et l'autre avec une inflation élevée. Le premier est Pareto-supérieur mais le second est stable. Dans la version avec les AGs, Arifovic introduit deux populations de chromosomes : une pour la jeune génération et une pour la vieille. Les populations sont actualisées de manière alternée, après que chaque membre a vécu sa vie de deux périodes. Les chromosomes représentent les consommations de la première période pour chaque génération. La population pour la génération $t+1$ est générée à partir de celle de $t-1$ en utilisant les opérateurs génétiques habituelles.

Les simulations avec l'offre monétaire constante convergent vers l'équilibre stationnaire où la monnaie a de la valeur. Cela reproduit donc des résultats expérimentaux de Lim et all.(1994). Les simulations avec l'économie à déficit positif convergent vers l'équilibre stationnaire à faible inflation. De plus, cette convergence a lieu même pour des conditions initiales pour lesquelles l'apprentissage avec des moindres carrées de Marcet & Sargent (1989) divergent. Dans une approche similaire, Bullard & Dufy (1998) implémente les AGs pour représenter les anticipations d'inflation des agents dans le modèle avec déficit positif. Cette approche converge aussi vers l'équilibre à faible inflation. Ces résultats correspondent donc mieux aux résultats expérimentaux (Marimon and Sunder (1993), Arifovic(1995)).

Vallée (2000) utilise les AGs pour étudier les ajustements dans un modèle de jeu répété consacré à la crédibilité de la politique monétaire du Gouvernement. Il s'agit de la répétition d'un jeu inflation–chômage à la Barro et Gordon où la stratégie du gouvernement (le meneur) correspond aux taux d'inflation annoncé et réalisé, tandis que celle des agents privés (les suiveurs) est le taux d'inflation anticipé. La population des agents privés est de taille N et un AG avec N chromosomes est utilisé pour représenter leur taux d'inflation anticipé (le chromosome i représente l'anticipation de l'agent i). Après l'annonce du gouvernement, l'évolution de l'AG correspond à l'apprentissage, par les agents, de la fonction de réaction du gouvernement en fonction de l'évolution de l'inflation réalisée. Cet article teste différentes structures possibles pour un AG avec codage réel et cherche à mesurer l'information générée par chaque structure et son impact sur l'évolution du jeu. Cet article, qui se place aussi dans un cadre où l'AG représente l'apprentissage social, souligne bien l'intérêt que nous devons porter à la structure de l'AG retenu et la nature de l'information générée et traité par le AG.

3.2.2 Théorie des jeux

Plus directement dans le domaine de la théorie des jeux, l'exemple le plus connu de l'utilisation des AGs pour résoudre un problème standard est le travail d'Axelrod sur l'émergence de la coopération dans le dilemme du prisonnier répété. Axelrod (1987) utilise l'AG pour faire évoluer une population de stratégies qui jouent, contre toutes les autres stratégies dans la population, au dilemme du prisonnier répété à deux joueurs. Dans cette approche, chaque chromosome représente l'histoire récente des choix et des observations de chaque joueur. La performance (ou le fitness) de chaque stratégie est alors évaluée dans ce jeu. L'environnement de chaque stratégie est formé par la population des autres stratégies dans la population. Comme cette population évolue dans le temps, l'environnement de chaque stratégie évolue aussi. Axelrod montre alors que cette évolution produit des stratégies dont le fitness moyen est au moins aussi bien que le gain moyen de la stratégie de Tit-for-Tat qui avait dominé les tournois précédents organisés par lui. Certains autres cadres ont même conduit à des stratégies plus performantes que le Tit-for-Tat.

Plus récemment, Yao et Darwen (2000) ont étendu cette approche au cas du dilemme du prisonnier répété à N joueurs. Ils montrent que l'émergence de la coopération devient plus difficile quand N augmente. Cela permet l'étude d'un jeu somme toute assez compliqué et améliore notre compréhension concernant la coopération entre les agents économiques.

Dans une approche différente, Dawid (1999, ch. 5) étudie l'apprentissage dans les jeux évolutionnaires. Il complète alors les mécanismes couramment utilisés (jeux fictifs, apprentissage bayésien, apprentissage basé sur la meilleure réponse) par l'apprentissage avec un AG. Chaque chromosome correspond à une stratégie mixte et la population des stratégies évolue grâce au AG. Il montre qu'en général l'apprentissage génétique atteint l'équilibre de Nash dans les jeux évolutionnaires. Il existe des problèmes de convergence dans des jeux avec une structure de gains particulière (si par exemple les stratégies qui ont un fitness élevé au début du jeu ne font partie d'aucune trajectoire d'équilibre) mais il s'agit des cas très particuliers qui seraient aussi source de problèmes dans l'activité économique. Ces résultats soulignent la robustesse de l'équilibre de Nash même si l'on suppose des agents avec une information réduite ou une rationalité limitée.

Enfin, Vallée et Deissenberg (1998) s'intéressent à l'utilisation des AGs comme outil d'apprentissage d'une taxe optimale de type pollueur-payeur dans le cadre d'un jeu différentiel entre un gouvernement et une firme monopolistique. Traditionnellement, la stratégie envisagée est une stratégie de Stackelberg. Cette dernière nécessite cependant que le gouvernement connaisse la fonction de réaction du monopole. Faisant l'hypothèse d'incertitude totale, Vallée et Deissenberg (1998) montrent qu'un apprentissage basé sur un AG, où chaque chromosome définit une séquence de taxation pour plusieurs périodes futures, converge vers un équilibre de Stackelberg optimal. Néanmoins, les auteurs montrent que la convergence est longue au regard d'un temps économique réaliste et qu'une application réelle reste encore difficilement envisageable.

3.2.3 Dynamique des marchés

Comme en macroéconomie, les algorithmes génétiques peuvent être utilisés pour étudier la dynamique des marchés avec des agents qui utilisent des stratégies adaptatives *intelligentes*. Dans une approche plutôt

standard, cela permet la vérification de l'émergence et la robustesse des équilibres. Il est même possible de s'intéresser à la dynamique hors-équilibre des marchés.

Arifovic (1994) étudie la dynamique d'un modèle de cobweb où il y a n firmes preneurs de prix qui produisent un bien homogène périssable. L'existence de délais de production implique que les quantités produites dépendent des anticipations de prix. Le modèle possède un équilibre à anticipations rationnelles (EAR) unique. Arifovic confrontent deux types d'AGs aux données obtenues dans les expériences avec ce modèle et aux autres hypothèses pour l'apprentissage (anticipations de cobweb, moyennes simples des prix observés et moindres carrés). Dans une première approche, un seul AG est utilisé pour représenter les stratégies des firmes. Dans ce cas nous sommes dans un cadre d'apprentissage social où chaque chromosome correspond aux quantités produites par une firme. Dans une seconde approche, chaque firme possède son propre AG et donc nous sommes dans un cadre où l'évolution du AG représente l'apprentissage de chaque firme et il existe une co-évolution entre ces processus. Les résultats démontrent que dans les deux cas l'utilisation des AGs donne une dynamique plus conforme avec les expériences que les trois autres hypothèses, surtout quand l'AG est élitiste : la convergence vers l'EAR même dans le cas instable, fluctuations autour de l'équilibre et des une variance plus grande des prix dans le cas instable. L'élitisme apparaît comme une hypothèse nécessaire pour la convergence dans le cas multi-population. L'apprentissage individuel est donc plus exigeant en ce qui concerne la *sophistication* des agents.

Vriend (2000) s'intéresse plus directement aux différences qui existent entre l'utilisation des AGs pour représenter un apprentissage social ou individuel dans le cadre d'un oligopole de Cournot homogène où les firmes doivent apprendre les quantités *optimales* à produire. Deux utilisations des AGs sont confrontées dans ce cadre. Dans le premier cas, un AG unique représente l'apprentissage au niveau de la population des firmes (*apprentissage social*), dans le second, chaque firme possède son propre AG : chaque firme possède plusieurs règles de décisions dans son AG même si une seule de ces règles est effectivement utilisée à chaque période par chaque firme (*apprentissage individuel*). Les résultats des simulations font apparaître une différence fondamentale entre ces deux cas : Dans le cas de l'apprentissage individuel le modèle converge bien vers l'équilibre de Cournot tandis qu'avec l'apprentissage social la convergence se fait sur l'équilibre concurrentiel (la tarification au coût marginal). Il est donc nécessaire de bien comprendre l'interaction entre la dynamique de l'apprentissage et la dynamique des forces économiques sous-jacentes. Vriend en déduit que pour tout algorithme d'apprentissage, un mécanisme de sélection qui est monotone par rapport aux gains va nécessairement faire apparaître une différence fondamentale entre son application au niveau individuel ou social. Il ne s'agit donc pas d'une hypothèse anodine qui peut être guidée par un critère de parcimonie dans la modélisation.

3.2.4 Économie de l'innovation

Yildizoglu (2001*a*) étudie la pertinence de AGs pour la modélisation de l'apprentissage individuel dans les stratégies de R&D des firmes. Une version épurée du modèle de Nelson & Winter (1982) est utilisée pour mettre en concurrence deux types de firmes : les NWFirms qui utilisent une règle fixe pour arbitrer entre l'investissement en R&D et celui en capital physique, et les GenFirms qui utilisent un AG individuel pour ajuster cet arbitrage à l'évolution de leur industrie. Les principaux résultats de ce modèle montrent que l'existence des GenFirms est une source d'efficacité au niveau de l'industrie en ce qui concerne le bien-être social et le progrès technique. De plus, cet apprentissage individuel est aussi source d'avantage concurrentiel pour les GenFirms qui finissent par dominer l'industrie.

Yildizoglu (2001*c*) montre que ces résultats sont encore plus forts si l'on utilise un système classificateur (SC) pour représenter l'apprentissage individuel. Cela correspond à une utilisation plus pertinente des AGs et le troisième type de firme (XCSFirms) qui sont mis en concurrence avec les NWFirms et les GenFirms conduisent à une efficacité encore plus grande pour le bien-être social et le progrès technique.

Dans cette approche, les AGs permettent l'inclusion dans les modèles de ce que Nelson et Winter appellent les *meta-routines* et qui correspondent aux règles utilisées par les firmes pour ajuster leur règles de décisions courantes à l'évolution de leur environnement. En cela cette approche correspond bien à une modélisation plus riche de la rationalité limitée des firmes.

3.2.5 Finance

Le même type d'approche est retenu par Arthur, Holland, LeBaron, Palmer & Tayler (1997) pour modéliser l'évolution d'un marché financier *artificiel*. Le marché contient des agents hétérogènes dont les anticipations s'adaptent continûment au marché que ces anticipations créent de manière agrégée. La constations de la récursivité des anticipations (les agents doivent anticiper ce que les autres anticipent – le concours de beauté de Keynes) conduit les auteurs à retenir une approche inductive des anticipations. Ils supposent que chaque agent possède à chaque moment du temps une multiplicité de modèles linéaires de prévision qui correspondent à des différentes *hypothèses* quant à la direction du marché et en utilise ceux qui apparaissent les mieux adaptés à la situation courante. Les agents apprennent alors quelles sont les hypothèses qui se sont avérées les meilleurs et inventent de nouvelles hypothèses de temps en temps grâce à un AG. L'ensemble des modèles de chaque agent est donc assez proche d'un système classificateur à la Holland. Les simulations font apparaître les deux types de régimes qui sont souvent opposés dans la vision qu'on a des marchés financiers : quand les agents actualisent rarement leurs anticipations, le marché converge vers l'équilibre aux anticipations rationnelles caractérisant l'hypothèse de marché efficient. Quand les agents explorent intensivement des hypothèses alternatives, le marché s'auto-organise dans un régime plus complexe où des cracks et des bulles spéculatives apparaissent de manière assez similaire aux données provenant des marchés réels. Ce qui, en définitive, souligne l'intérêt d'une approche inductive quant à la rationalité des agents. Dans une approche différente, Schulenburg & Ross (2000) étudie la performance des agents artificiels face aux données provenant d'un marché financier réel. Les agents sont d'abord *entraînés* sur des données des neuf années et ensuite ils utilisent les règles qu'ils ont développées pendant cette période pour gérer leurs transactions pendant la dixième année. Les simulations montrent que les agents artificiels développent une large diversité de stratégies innovantes qui ont une performance supérieure à celle des stratégies de base de type buy and-hold.

4 Conclusion

L'objectif principal de cet article était de combler l'absence d'une présentation générale des algorithmes génétiques mettant l'accent sur leur pertinence en économie. La seconde partie de cet article fait clairement apparaître la richesse des applications des AGs en économie. Concernant leur puissance et leur pertinence, nous nous joignons aux conclusions d'une des personnes qui a le plus contribué à leur diffusion :

> The main conclusion is that GAs have already passed beyond only their technical applications to artificial systems, they are ready to comme into the real world of living systems and even have made the first steps in this direction. (Kosorukoff et Goldberg (2001))

Malgré leur simplicité, les AGs représentent bien l'exploration stochastique mais orientée d'un espace de stratégies par les agents. Que cela soit pour résoudre un problème d'optimisation particulièrement difficile ou pour représenter les stratégies de R&D adaptatives des firmes. Cela ne signifie nullement que ces algorithmes soient suffisantes pour résoudre tous les problèmes en économie. En effet, quand ils s'agit d'explorer un espace de stratégies particulièrement complexes, ils peuvent s'avérer extrêmement coûteux en temps de calcul et cela peut limiter considérablement, par exemple, leur application en temps réel, en tant qu'outil de trading en bourse. Quand il s'agit de représenter l'apprentissage adaptatif des agents, ils peuvent s'avérer à la fois trop simples et trop compliqués : trop simples car ils ne tiennent pas compte de la capacité des agents à former des anticipations qui guident l'exploration de l'espace de stratégies ; trop compliqués car ils correspondent à une exploration trop systématique et donc trop coûteuse de l'espace de stratégies.

Dans les deux cas, leur efficacité peut être considérablement augmentée si l'on intègre dans leur utilisation un mécanisme inductif qui correspond à une certaine représentation de leur environnement de manière à orienter leur exploration. Cela est à la base de plusieurs algorithmes d'optimisation hybrides que nous avons considérés. Mais cela déjà était à la base des systèmes classificateurs formalisés par John Holland. En effet, les systèmes classificateurs adjoignent aux chromosomes (donc aux stratégies) des conditions sous lesquelles ils seront utilisés : l'utilisation d'une règle est considérée uniquement dans le cas où l'état de l'environnement (où du problème considéré) correspond aux conditions d'application spécifiques de cette règle. Cela évite

automatiquement une utilisation trop systématique de tous les chromosomes et cela, même dans des contextes qui ne sont pas adaptés. Dans le cas de la représentation de l'apprentissage, Yildizoglu (2001*c*) montre que cela augmente considérablement l'efficacité de l'apprentissage et des performances industrielles qui en découlent. Un autre mécanisme inductif qui peut être utilisé pour compléter les AGs est un réseau de neurones artificiels (RNA). Dans ce cas, le RNA fournit à l'AG une représentation du problème qu'on cherche à résoudre ou de l'environnement de l'agent dont il représente l'apprentissage. En ce qui concerne le cadre de l'apprentissage, Yildizoglu (2001*b*) montre que l'utilisation du RNA améliore considérablement l'apprentissage de l'AG.

Les perspectives pour l'utilisation des algorithmes génétiques en économie restent très riches. Les connaissances de base de l'économiste peut en plus mise en oeuvre pour obtenir de nouvelles applications (algorithmes hybrides) qui dépassent la puissance technique des AGs, pour les utiliser comme la base d'une approche heuristique très riche des problèmes et des comportements économiques.

Références

Alander, J. T. (2001), An indexed bibliography of genetic algorithms in economics, Technical Report Report Series No. 94-1-ECO, Department of Information Technology and Production Economics, University of Vaasa, http ://www.uwasa.fi/∼jal/.

Alemdar, N. & Özyildirim, S. (1998), 'A genetic game of trade, growth and externalities', *Journal of Economic Dynamics and Control* **22**(6), 811–832.

Allen, F. & Karjalainen, R. (1999), 'Using genetic algorithms to find technical trading rules', *Journal of financial Economics* **51**(2), 245–271.

Alliot, J. & Schiex, T. (1994), *Intelligence artificielle et informatique théorique*, Cépaduès-Éditions.

Arifovic, J. (1994), 'Genetic algorithm learning and the cobweb model', *Journal of Economic Dynamic and Control* **18**, 3–28.

Arifovic, J. (1995), 'Genetic algorithms and inflationary economies', *Journal of Monetary Economics* **36**, 219–243.

Arifovic, J. (1996), 'The behavior of the exchange rate in the genetic algorithm and experimental economies', *Journal of Political Economy* **104**(3), 510–541.

Arifovic, J. (1998), 'Stability of equilibria under genetic-algorithm adaptation : an analysis', *Macroeconomic Dynamics* **2**, 1–22.

Arifovic, J. (2001), 'Evolutionary dynamics of currency substitution', *Journal of Economic Dynamics and Controls* **25**(3-4), 395–417.

Arifovic, J., Bullard, J. & Duffy, J. (1997), 'The transition from stagnation to growth : An adaptive learning approach', *Journal of Economic Growth* **2**, 185–209.

Arthur, W. B., Holland, J. H., LeBaron, B., Palmer, R. & Tayler, P. (1997), Asset pricing under endogenous expectations in an artificial stock market, *in* W. Arthur, S. N. Durlauf & D. Lane, eds, 'The Economy as an Evolving Complex System II', Vol. Proceedings Volume XXVII of *Santa Fe Institute Studies in the Science of Complexity*, Addison-Wesley, Reading :MA, pp. 15–44.

Aslock, D., Smucker, D., Stanley, A. & Tesfatsion, L. (1996), 'Preferential partner selection in an evolutionary study of the prisoner's dilemma', *BioSystems* **37**(1-2), 99–125.

Axelrod, R. (1987), The evolution of strategies in the iterated prisoner's dilemma, *in* L. D. Davis, ed., 'Genetic algorithms and simulated annealing', Morgan Kaufmann.

Başar, T. & Olsder, G. (1995), *Dynamic Noncooperative Game Theory*, 2nd edn, Academic Press, New York.

Baluja, S. (1995), An empirical comparison of seven iterative and evolutionary function optimization heuristics. Working Paper.

Bauer, R. (1994), *Genetic algorithm and investment strategies*, John Wiley & Sons.

Beaumont, P. & Bradshaw, P. (1995), 'A distributed parallel genetic algorithm for solving optimal growth models', *Computational Economics* **8**(3), 159–180.

Beenstock, M. & Szpiro, G. (1999), Specification serach in non-linear time series models using the genetic algorithm. Working Paper.

Boné, R., Thillier, R., Yvon, F. & Asselin, J. (1998), Optimisation by genetic algorithm of stochastic linear models of time series, *in* J.-M. Aurifeille & C. Deissenberg, eds, 'Bio-Mimetic approaches in Management Science', Kluwer, pp. 153–162.

Bullard, J. & Duffy, J. (1997), 'A model of learning and emulation with artifical adaptive agents', *Journal of Economic Dynamics and Control* **22**, 179–207.

Davis, L. (1987), *Genetic algorithms and Simulated annealing*, Morgan Kaufmann, Los Altos, CA.

Dawid, H. (1997), On the convergence of genetic learning in a double auction market. Working Paper, University of Vienna.

Dawid, H. (1999), *Adaptive learning by Genetic Algorithm. Analytical results and applications to economic models*, Springer, Berlin.

Den Haan, W. & Marcet, A. (1990), 'Solving the stochastic growth model by parameterizing expectations', *Journal of Business and Economic Statistics* **8**, 31–34.

Dorsey, R. & Mayer, W. (1995), 'Genetic algorithms for estimation problems with multiple optima, nondifferentiability, and other irregular features', *Journal of Business and Economic Statistics* **13**(1), 53–66.

Duffy, J. & Nelis, P. M. (2001), 'Approximating and simulating the stochastic growth model : Parameterized expectations, neural networks, and the genetic algorithm', *Journal of Economic Dynamics and Control* **25**, 1273–1303.

Eddelbüttel, D. (1996), A hybrid genetic algorithm for passive management. Working Paper presented at the Second conference on computing in economics and Finance, Society of computational economics, Geneves, Suisse.

Goldberg, D. (1981), Robust learning and decision algorithms for pipeline operations. Unpublished dissertation proposal, University of Michigan, Ann Arbor.

Goldberg, D. (1989), *Genetic Algorithm In Search, Optimization And Machine Learning*, Addison-Wesley.

Holland, J. H. (1975), *Adaptation In Natural And Artificial Systems*, University of Michigan Press.

Holland, J. H. (1995), *Hidden Order*, Addison-Wesley.

Holland, J. H., Holyoak, K. J. & Thagard, P. R. (1989), *Induction. Processes of Inference, Learning, and Discovery*, MIT Press, Cambridge :MA.

Jong, K. D. (1980), 'Adaptive systeme design : A genetic approach', *IEEE Transactions on Systems, Man, and Cybernetics* **10**(3), 556–574.

Jong, K. D. (1985), Genetic algorithms : A 10 year perspective, *in* IEEE, ed., 'Proceedings of the First International Conference on Genetic Algorithms', pp. 169–177.

Kosorukoff, A. & Goldberg, D. E. (2001), Genetic algorithms for social innovation and creativity. IlliGAL Report No. 2001005.

Koza, J. (1991), A genetic approach to econometric modeling, *in* P. Bourgine & B. Walliser, eds, 'Economics and Cognitive Science', Pergamon Press, pp. 57–75.

Koza, J. (1992), *Genetic Programming*, MIT Press.

Krishnakumar, K. & Goldberg, D. (1992), 'Control system optimization using genetic algorithm', *Journal of Guidance, Control, and Dynamics* **15**(3), 735–740.

LeBaron, B. (1998), Agent based computational finance : Suggested readings and early research, Technical report, Graduate School of International Economics and Finance, Brandeis University.

Lerman, I. & Ngouenet, F. (1995), Algorithmes génétiques séquentiels et parallèles pour une représentation affine des proximités, Rapport de Recherche de l'INRIA Rennes - Projet REPCO 2570, INRIA.

Lim, S., Prescott, E. & Sunder, S. (1994), 'Stationary solution to the overlapping generations model of fiat money : Experimental evidence', *Empirical Economics* **19**, 255–277.

Loraschi, A. & Tettamanzi, A. (1996), An evolutionary algorithm for portfolio selection within de downside risk framework, *in* C. Dunis, ed., 'Forecasting Financial MArkets', John Wiley and Sons, pp. 275–285.

Loraschi, A., Tettamanzi, A., Tomassini, M. & Verda, P. (1995), Distributed genetic algorithms with an application to portfolio selection. Working Paper.

Mahfoud, S. & Mani, G. (1996), 'Financial forecasting using genetic algorithms', *Applied Artificial Intelligence* **10**(6), 543–566.

Marcet, A. (1999), Solving nonlinear stochastic growth models by parameterizing expectations. Working Paper.

Marcet, A. & Sargent, T. (1989), Least squares learning and the dynamics of hyperinflation, *in* W. Barnett, J. Geweke & K. Shell, eds, 'Economic Complexity : Chaos, Sunspots, Bubbles and Non-linearity', Cambridge University Press.

Marco, N., Godart, C., Désidéri, J.-A., Mantel, B. & Périaux, J. (1996), A genetic algorithm compared with a gradient-based method for the solution of an active-control model problem, Technical report, INRIA. Rapport de Recherche de l'INRIA - Projet SINUS, n0. 2948.

Messa, K. & Lybanon, M. (1991), Curve fitting using genetic algorithms, Technical report, Naval Oceanographic and Atmospheric Research Laboratory (NOARL). Report n^{o}18.

Messa, K. & Lybanon, M. (1992), 'Improved interpretation of satellite altimeter data using genetic algorithms', *Telematics and Informatics* **9**(3-4), 349–356.

Mühlenbei, H. (1991), Evolution in time and space - the parallel genetic algorithm, *in* G. Rawlins, ed., 'Foundations of Genetic Algorithms', Morgan Kaufman, San Mateo, pp. 316–337.

Mühlenbei, H. (1993), Evolutionary algorithms : Theory and applications, *in* E. Aarts & J. Lenstra, eds, 'Local search in Combinatorial Optimization', Wiley.

Michalewicz, Z. (1992), *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag.

Michalewicz, Z. (1994), 'Evolutionary computation techniques for nonlinear programming problems', *International Transactions in Operational Research* **1**(2), 223–240.

Michalewicz, Z. (1995), A survey of constraint handing techniques in evolutionary computation methods. *Proceedings of the 4th Annual Conference on Evolutionary Programming*, MIT Press.

Michalewicz, Z. & Attia, N. (1994), Evolutionary optimization of constrained problems, *in* A. Sebald & L. Fogel, eds, 'Proceedings of the 3rd Annual Conference on Evolutionaray Programming', World Scientific Publishing, pp. 98–108.

Michalewicz, Z., Janikow, C. & Krawczyk, J. (1992), 'A modified genetic algorithm for optimal control problems', *Computers and Mathematics with Applications* **23**(12), 83–94.

Michalewicz, Z., Logan, T. & Swaminathan, S. (1994), Evolutionary operators for continuous convex parameters spaces, *in* A. Sebald & L. Fogel, eds, 'Proceedings of the 3rd Annual Conference on Evolutionaray Programming', World Scientific Publishing, pp. 84–107.

Miller, J. (1986), A genetic model of adaptive economic behavior. Working Paper. Univeristy of Michiga.

Miller, J. (1996), 'The coevolution of automata in the reapeated prisoner's dilemna', *Journal of Economic Behavior and Organization* **29**(1), 87–112.

Miller, J., Butts, C. & Rode, D. (1998), Communication and cooperation. Working Paper, Santa Fe Institute.

Mitchell, M., Crutchfield, J. & Salmon, M. (1996), Evolving cellular automata with genetic algorithms : A review of recent work. in *Proceedings of the First International Conference on Evolutionary Computation and its Application*, Moscou, Russie.

Neely, C., Weller, P. & Dittmar, R. (1997), 'Is technical analysis in the foreign exchange market profitable ? a genetic programming method', *Journal of Financial and Quantitative Analysis* **32**(4), 405–426.

Nelson, R. R. & Winter, S. (1982), *An Evolutionary Theory of Economic Change*, The Belknap Press of Harvard University, London.

Novkovic, S. & Sverko, D. (1997), Genetic waste and the role of diversity in genetic algorithm simulations. Working Paper. Saint Mary's University. Canada.

Östermark, R. (1999), 'Solving irregular econometric and mathematical optimization problems with a genetic hybrid algorithm', *Computational Economics* **13**, 103–115.

Özyildirim, S. (1996), 'Three country trade relations : A discrete dynamic game approach', *Computers and Mathematics with Applications* **32**, 43–56.

Özyildirim, S. (1997), 'Computing open-loop noncooperative solution in discrete dynamic games', *Evolutionary Economics* **7**(1), 23–40.

Özyildirim, S. & Alemdar, N. (1998), Learning the optimum as nash equilibrium. Working Paper.

Pan, Z., Chen, Y., Khang, L. & Zhang, Y. (1995), Parameter estimation by genetic algorithms for nonlinear regression, *in* 'Proceeding of International Conference on Optimization Techniques and Applications', World Scientific, pp. 946–953.

Pereira, R. (2000), Genetic algorithm optimisation for finance and investment, Technical report, La Trobe University.

Ruggiero, M. (1997), *Cybernetic Trading Strategies*, John Wiley & Sons.

Sargent, T. (1993), *Bounded rationality in Macroeconomics*, Oxford Univesity Press.

Schulenburg, S. & Ross, P. (2000), An adaptive agent basel economic model, *in* P. L. Lanzi, W. Stoltzmann & S. W. Wilson, eds, 'Learning classifier systems. From foundations to applications', Vol. 1813 of *Lecture notes in artificial intelligence*, Springer, Berlin, pp. 263–282.

Stanley, A., Ashlock, D. & Tesfatsion, L. (1994), Iterated prisoner's dilemma with choice and refusal partners, *in* C. Langton, ed., 'Artificial Life III, Proceedings Volume 17, Santa Fe Institute Studies in the Sciences of Complexity', Addison Wesley, pp. 131–175.

Szpiro, G. (1997*a*), 'Forecasting chaotic time series with genetic algorithmss', *Physical Review E* **55**, 2557–2568.

Szpiro, G. (1997*b*), 'A search for hidden relationships : Data mining with genetic algorithms', *Computational Economics* **10**, 267–277.

Vallée, T. (2000), Heterogeneous inflation learning : communication versus experiments. Working Paper presenté à la conférence : *Complex behavior in economics : modeling, computing, and mastering complexity*, Aix en Provence, Marseille, France, 4-6 Mai.

Vallée, T. & Başar, T. (1998), Incentive stackelberg solutions and the genetic algorithm. *Proceedings of the* 8^{th} *International Symposium on Dynamic Games and Applications*, Château Vaalsbroek, Maastricht, The Netherlands, p.633-639, July 5-8.

Vallée, T. & Başar, T. (1999a), 'Off-line computation of Stackelberg solutions with the Genetic Algorithm', *Computational Economics* **13**(3), 201–209.

Vallée, T. & Başar, T. (1999b), Computation of the closed-loop stackelberg solution using the genetic algorithm, *in* 'Preprints of the IFAC conférence en Automatic Control, Beijing, Chine, 5-9 juillet 1999, volume M : "Management, Global and Educational Issues"', IFAC, pp. 69–74.

Vallée, T. & Deissenberg, C. (1998), Learning how to regulate a polluter with unknown characteristics : An application of genetic algorithms to a game of dynamic pollution control, *in* J.-M. Aurifeille & C. Deissenberg, eds, 'Bio-Mimetic approaches in Management Science', Kluwer, pp. 197–208.

Varetto, F. (1998), 'Genetic algorithms applications in the analysis of insolvency risk', *Journal of Banking and Finance* **22**, 1421–1439.

Vriend, N. (2000), 'An illustration of the essential difference between individual and social learning, and its consequences for computational analysis', *Journal of Economic Dynamics and Control* **24**, 1–19.

Weiss, G. (1999), Timeweaver : a genetic algorithm for identifying predictive patterns in sequences of events, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)', Morgan Kaufmann, pp. 718–725.

Weiss, G. & Hirsch, H. (1998), Learning to predict rare events in event sequences, *in* 'Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)', AAAI Press, pp. 359–363.

Yao, X. & Darwen, P. (2000), Genetic algorithms and evolutionary games, *in* 'Commerce, Complexity and Evolution', Cambridge University Press, pp. 325–347.

Yildizoglu, M. (2001*a*), 'Competing R&D strategies in an evolutionary industry model', *forthcoming in Computational Economics. Available at http ://yildizoglu.montesquieu.u-bordeaux.fr/* .

Yildizoglu, M. (2001*b*), Connecting adaptive behaviour and expectations in models of innovation : The potential role of artificial neural networks, Technical report, IFREDE-E3i, Universite Bordeaux IV Montesquieu.

Yildizoglu, M. (2001*c*), Modelling adaptive learning : R&D strategies in the model of Nelson & Winter(1982). Working Papers IFREDE-E3i (http ://www.ifrede.org).

Zhu, X., Huang, Y. & Doyle, J. (1997), Genetic algorithms and simulated annealing for robustness analysis. in Proceedings of the American Control Conference, Albuquerque, New Mexico, 3756-3760.