Contrôle d'un algorithme génétique

Michèle Sebag, Marc Schoenauer

Contrôle d'un algorithme génétique

Michèle Sebag* — Marc Schoenauer**

** Laboratoire de Mécanique des Solides*
URA CNRS 317
Ecole Polytechnique
91128 Palaiseau

*** Centre de Mathématiques Appliquées*
URA CNRS 756
Ecole Polytechnique
91128 Palaiseau

RÉSUMÉ. *Après avoir rappelé le principe des algorithmes génétiques (AG), cet article s'intéresse à l'efficacité de ces méthodes d'optimisation. Les liens entre la représentation choisie et les opérateurs d'évolution manipulant cette représentation sont discutés. Deux problèmes artificiels illustrant des limites attendues ou inattendues des AG sont présentés. Enfin, le nombre important de paramètres et de choix structurels intervenant dans un algorithme génétique pose le problème du contrôle d'un AG. Cet article propose une méthode nouvelle pour le contrôle d'un AG, fondée sur l'apprentissage symbolique.*

ABSTRACT. *This paper briefly recalls the principle of genetic algorithms (GA) and discusses their efficiency. The Radcliffe's principles, linking the representation of the problem and the evolution operators, are presented. Last, we address the control of an AG by an induction-based approach. Experimental results obtained on typical GA problems and a combinatoric optimization problem, are discussed.*

MOTS-CLÉS : *algorithme génétique, évolution, contrôle, optimisation, apprentissage inductif, optimisation hybride.*
KEY WORDS : *genetic algorithm, evolution, control, optimization, inductive learning, hybrid optimization.*

1 Introduction

Les algorithmes de Calcul Évolutif (*Evolutionary Computation*, EC) sont des méthodes d'optimisation stochastiques (i.e. fondées sur des tirages aléatoires), opérant dans des espaces de taille extrêmement vaste, et grossièrement inspirés de la métaphore darwinienne *les plus adaptés survivent*. Leur vogue récente ([56, 5, 19, 40, 43, 41]) est due en particulier aux raisons suivantes :

- Les EC sont applicables dans un contexte beaucoup plus vaste que les méthodes d'optimisation classiques : ils permettent d'optimiser des fonctions qui sont non convexes, non dérivables, non continues, et même non explicites mais décrites sous forme de programme. Il devient donc possible de s'attaquer à nombre de problèmes jusqu'à présent hors de portée.

- La montée en puissance des calculateurs rend ces méthodes de plus en plus abordables ; elles sont néanmoins d'un ou deux ordres de grandeur plus coûteuses que les méthodes classiques telles les méthodes de gradient, lorsque ces dernières sont applicables et que l'on peut comparer les performances. Enfin, elles obtiennent des résultats comparables à ceux d'autres méthodes stochastiques plus anciennes telles le recuit simulé [47], tout en pouvant être parallélisées plus facilement.

- Les EC présentent une analogie avec le développement biologique, et la théorie darwinienne de l'évolution : dans une population de solutions qui évoluent et se reproduisent, la loi de la sélection naturelle fait émerger les individus les plus adaptés, i.e. les solutions optimales. Les nombreuses heuristiques mises au point pour accélérer la convergence de l'évolution ou empêcher sa convergence prématurée sont souvent directement interprétables en termes anthropomorphiques ou biologiques (voir par exemple [38, 55]), ce qui peut selon les cas faciliter l'entendement ou agacer le lecteur...

Le Calcul Evolutif regroupe trois grands courants algorithmiques: les algorithmes génétiques (AG) [31, 23], les stratégies d'évolution (*Evolution Strategies* ou ES) [60, 2], et la programmation évolutive (*Evolutionary Programming* ou EP) [18]. Une discussion approfondie des différences entre ces approches peut être trouvée dans [3, 16].

Cet article est consacré essentiellement aux algorithmes génétiques, dont nous présenterons tout d'abord la version canonique, qui travaille dans des espaces binaires [31, 23]. Les principaux opérateurs de l'évolution, sélection, mutation et croisement [65, 15], sont introduits et situés par rapport aux deux tâches antagonistes de l'évolution selon Goldberg [23] : l'exploration de l'espace de recherche et l'exploitation des zones prometteuses découvertes. Enfin, nous discutons brièvement les heuristiques les plus connues [26].

Dans une seconde partie, nous nous intéressons aux raisons de l'efficacité des AG. Ainsi, une représentation binaire est adéquate si l'espace de recherche comprend des sous-espaces performants (*building blocks*), et si les solutions optimales appartiennent à l'intersection de ces sous-espaces. Le théorème des schémas de Holland [31] indique que les AG sont bien adaptés pour explorer un espace de recherche ainsi structuré. Radcliffe [51] s'intéresse aux propriétés et au comportement des AGs dans des espaces de représentation quelconques ; introduisant une généralisation des schémas de Holland, il propose six principes liant la représentation choisie et les opérateurs d'évolution agissant sur cette représentation, "dont on s'attend à ce qu'ils soient vérifiés dans la majorité des

cas de succès".
A titre de contre-exemple, nous présentons deux problèmes artificiels sur lesquels les AG *ne sont pas* efficaces : les difficultés rencontrées sont attendues dans le cas des problèmes trompeurs [23, 67], et inattendues dans le cas du problème de la Voie Royale [45, 46, 20].

La troisième partie s'intéresse au contrôle des AG, qui est d'importance cruciale étant donné la lenteur de ces algorithmes. A côté des méthodes de contrôle fondées sur une approche brutale [57, 27], statistique [30] ou comme sous-produit de l'optimisation génétique elle-même [58, 64, 37], nous proposons une approche symbolique du contrôle des AG, fondée sur l'apprentissage inductif [44, 36]. Le fait d'apprendre en cours d'évolution et d'utiliser les règles apprises pour guider les étapes ultérieures de l'évolution correspond métaphoriquement à une évolution "civilisée" plutôt que darwinienne [63, 54]. Une validation empirique de cette approche est effectuée sur un problème trompeur, le problème de la Voie Royale et un problème d'optimisation combinatoire.

2 Version canonique des AG

Après avoir présenté le principe des AG et décrit les opérateurs d'évolution les plus courants, nous discutons ici quelques difficultés majeures de cette approche et les heuristiques élaborées pour y remédier.

2.1 Principe

La version canonique des AG s'intéresse à des espaces binaires. Plus précisément, étant donné l'espace de recherche $\Omega = \{0,1\}^N$, on cherche à optimiser une fonction performance F (*fitness*) définie sur Ω à valeurs dans $\mathbb{R}^+$.

$$F : \Omega = \{0,1\}^N \rightarrow \mathbb{R}^+$$

Un élément de Ω, appelé individu ou *chromosome*[1] est ici une chaîne de bits. Une *population* Π est un ensemble de P individus. L'opération de base d'un AG consiste à passer de la population courante Π_i , à la population suivante Π_{i+1} ; cette itération de base de l'algorithme est appelée *génération*. Le schéma d'un AG est donc le suivant:

1 **Initialisation.**
Dans le cas le plus fréquent, la population initiale (Π_0 , à la génération 0) est déterminée aléatoirement ; chacun des N bits des P individus de Π_0 est a ors tiré aléatoirement dans {0,1}. Il est également possible d'utiliser les solutions fournies par l'expert ou déterminées lors d'une étape antérieure (cf 2.2).

2 **Evaluation et critère d'arrêt.**
La performance de chaque individu de la population courante Π_i est calculée. Notons que, dans la plupart des problèmes réels, cette étape consomme la majorité du temps calcul. D'autre part, cette étape peut aisément être parallélisée dans la mesure où les calculs sont indépendants.

C'est au vu des performances des individus de la population courante que se détermine l'arrêt ou la poursuite de la recherche (cf 2.3).

[1] Rappelons que les termes *individu* et *chromosome*, s'ils peuvent être utilisés comme synonymes dans la littérature relative aux AGs, recouvrent des notions biologiques différentes.

3 **Sélection**
Une population intermédiaire Π' de même taille que Π_i est créée par copie des individus de Π_i ; chaque individu est recopié un certain nombre de fois, et ce d'autant plus que sa performance est grande (cf 2.4.1).

La population Π_{i+1} est alors construite à partir de la population Π' à l'aide d'opérateurs génétiques, généralement au nombre de deux, le croisement et la mutation :

4 **Croisement**
Des couples d'individus tirés aléatoirement sans remise dans Π' (les *parents*) donnent naissance à des couples d'individus *enfants*, calculé à partir des valeurs des parents. L'ensemble de ces enfants constitue une population Π'' de même taille P. Le *croisement* correspond à une reproduction sexuée de la population (cf 2.4.2).

5 **Mutation**
La *mutation* correspond, elle, à une reproduction asexuée (cf 2.4.3) : chaque individu de Π'' est modifié par altération aléatoire.

6 **Génération suivante**: $\Pi_{i+1} = \Pi''$
Passage à la génération $i+1$: si le critère d'arrêt n'est pas satisfait (cf 2.3), la population Π_{i+1} devient la population courante, et on retourne en 2.

Précisons les modalités et les difficultés des étapes ci-dessus.

2.2 Initialisation

Dans le cas, le plus fréquent, où l'utilisateur ne dispose pas de connaissances particulières relatives à la fonction F, la population Π_0 est initialisée aléatoirement de manière la plus uniforme possible. Cependant, cette étape du processus peut permettre de prendre en compte certains types de connaissances *a priori* de l'utilisateur.

Il est tout d'abord possible d'inclure dans la population initiale Π_0 les solutions données explicitement par l'expert. Ce cas est souvent celui d'un opérateur qui dispose déjà de solutions du problème courant et veut les améliorer (ex : placement de capteurs, positionnement d'antennes,...).

En second lieu, Π_0 peut être elle-même obtenue comme résultat d'une optimisation génétique antérieure, considérant une fonction performance F_{ante}; F_{ante} constitue souvent une version simplifiée de la fonction F considérée. La population finale, "entrainée" sur F_{ante} oriente alors l'exploration et permet de mener à bien l'optimisation de la fonction F considérée. Cette stratégie peut être adoptée lorsque l'optimisation directe de F échoue, ou pour aborder des problèmes d'optimisation sous contraintes[2] [59].

2.3 Critère d'arrêt

Déterminer l'arrêt d'un processus génétique est l'une des difficultés majeures de l'approche AG. En effet, si l'on excepte le cas des problèmes artificiels, on ne

[2]Une première évolution se fixe alors simplement le but de trouver des individus admissibles, i.e. satisfaisant les contraintes. Lorsque une certaine proportion de la population est admissible (de l'ordre de 80%), on passe à l'optimisation de la fonction objectif proprement dite.

sait jamais si l'on a trouvé l'optimum[3]. Dans la pratique, l'utilisateur déclare un nombre de générations maximum, à compter du début de l'évolution, ou de la dernière amélioration trouvée.

La recherche peut également être stoppée lorsque tous les individus d'une même population sont des copies d'un même individu (ou de quelques individus). On dit alors qu'il y a "perte de la diversité génétique". Dans la pratique [49], la population ne se rediversifie jamais lorsqu'elle a atteint cet état[4] ; si donc les solutions optimales cherchées ne figurent pas dans la population, il y a convergence prématurée.
De nombreuses heuristiques visent à empêcher ou retarder la perte de la diversité génétique.

2.4 Opérateurs d'évolution

Les opérateurs de l'évolution (sélection, croisement et mutation) ont une double mission : celle de permettre au processus génétique d'exploiter les zones prometteuses découvertes, i.e. le voisinage des individus constituant les optima (locaux) de la performance découverts à ce jour, et celle d'explorer les zones encore inconnues. Ce dilemme est identifié par Goldberg sous le nom d'EVE (Exploitation *Versus* Exploration) [23]. Dans la mesure où la taille de la population reste constante, ces objectifs sont clairement antagonistes : il faut disposer d'individus pour explorer le voisinage des solutions optimales locales, et il faut également des individus pour explorer la "terra incognita".

Parmi les trois étapes majeures de l'évolution, la sélection est orientée vers l'exploitation seule. Le croisement et la mutation permettent également l'exploitation et l'exploration, avec la différence suivante: les enfants obtenus par croisement sont en général loin des parents, mais appartiennent à une région réduite de l'espace de recherche (qui dépend uniquement des parents). Par opposition, les enfants obtenus par mutation sont en général proches du parent, mais peuvent être situés dans *tout* l'espace de recherche (ce qui est une condition nécessaire pour espérer atteindre le maximum de la fonction F indépendamment de la population initiale).

2.4.1 La sélection

Principe

La sélection élabore une population intermédiaire Π' en recopiant les individus de la population courante Π_i . Ce sont les individus de Π' qui seront croisés et mutés pour fournir la population suivante Π_{i+1} . Un individu donné x de Π_i va figurer zéro, une ou plusieurs fois dans la population Π' ; Π' comprend le même nombre d'individus que Π_i .
L'espérance du nombre de copies de l'individu x est fonction du rapport entre sa performance $F(x)$ et la performance moyenne des individus de la population courante notée $\bar{F}$. Un individu de performance relative élevée disposera en moyenne de nombreuses copies dans la population Π' . En revanche, un individu de performance relative faible a peu de chances d'apparaître dans la population Π' , et donc d'être croisé ou muté.

---

[3]Dans le cas d'un espace continu, les AG déterminent des résultats " voisins" de l'optimum ; il est alors souhaitable de les coupler avec un algorithme local, hill-climbing par exemple, pour achever les derniers pas de l'optimisation.

[4]Certaines études théoriques par contre se fondent sur le fait que *tous* les bassins d'attraction de l'évolution finissent par être visités, sous certaines conditions [8]. Il suffit en effet d'attendre une mutation *ad hoc* ; mais cette mutation *ad hoc* n'est pas observée expérimentalement, du moins en un temps raisonnable.

La sélection concerne en général toute la population courante. Elle peut aussi se limiter à considérer une fraction de la population (le *fossé des générations* ou *generation gap*) [23, 38]. Une version extrême de cette variante consiste à ne sélectionner qu'un unique individu ; on parlera alors d'Algorithme Génétique Stationnaire (*Steady State Genetic Algorithm*) plutôt que d'AG [65]. L'argument invoqué en faveur d'un fossé des générations, est que cette approche augmente la stabilité du processus. En effet, si toute la population évolue, les chances de perdre les individus les meilleurs sont plus grandes que si seule une fraction de la population est modifiée. Néanmoins, aucun argument définitif n'a été avancé à ce jour pour l'une ou l'autre de ces variantes (voir [25] pour une discussion sur le sujet).

A noter que dans le cas d'un fossé des générations, les nouveaux individus créés peuvent soit remplacer leurs parents, soit remplacer des individus tirés au sort dans la population courante par *anti-sélection*, i.e. sélection basée sur l'inverse de la performance. La stratégie dite élitiste consiste à ne remplacer un individu que par un individu meilleur.

Une variante particulière consiste à distinguer en fonction du problème les individus sur lesquels doit agir la sélection à chaque génération (les *tories*), des individus à conserver à plus longue échéance (les *whigs*) [21].

Méthode

Pratiquement, la sélection est souvent effectuée par tirage dit *à la roulette*. La roulette comprend P cases, la largeur de la case i étant proportionnelle à $F(x_i)/\bar{F}$. La boule est tirée P fois ; lorsqu'elle tombe dans la case i, une copie de l'individu x_i est ajoutée à Π' (Figure 1). La sélection de nombreux exemplaires d'un individu mal adapté reste ainsi possible, même si elle est peu probable.

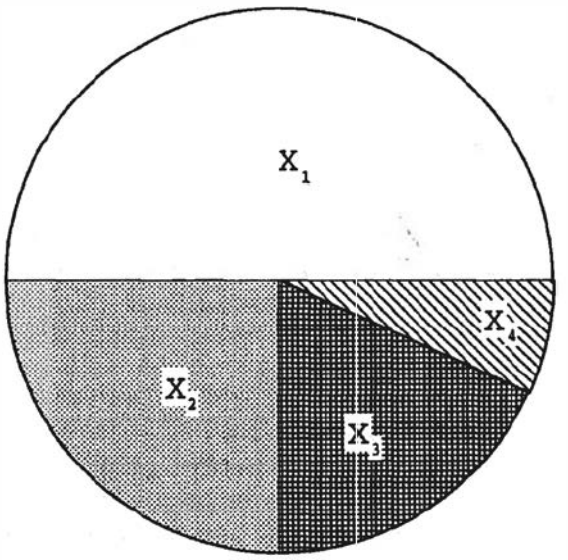


Figure 1 : Sélection par tirage à la roulette

La méthode dite *du reste stochastique*, moins biaisée par les petites tailles de population, consiste à recopier dans la population Π' un nombre de copies de x égal à la partie entière de $F(x)/\bar{F}$. La population Π' est ensuite complétée par tirage à la roulette comme ci-dessus, en associant à la case i la partie décimale de $F(x_i)/\bar{F}$.

La sélection par le rang consiste à effectuer le tirage de roulette sur des valeurs correspondant aux rang des individus (le meilleur ayant une valeur

égale à la taille de la population, le pire une valeur de 0). Elle a l'avantage de ne pas nécessiter de *mise à l'échelle* (cf 2.5.1).

Une autre méthode de plus en plus populaire, car pouvant être facilement implantée sur des machines massivement parallèles, est la sélection par *tournoi* [6, 14] ; un ensemble de K (généralement entre 2 et 10) individus de la population courante Π_i est choisi au hasard (uniformément), et le vainqueur du tournoi (i.e. celui des K individus ayant la meilleure performance) est ajouté à la population Π' .

Difficultés

La sélection remplit plusieurs tâches :

- Elle permet d'éliminer les individus inadaptés. Sous ce rapport, si la sélection est insuffisante, la convergence du processus est retardée ou empêchée.
- Elle permet de dupliquer les individus adaptés, afin d'explorer leur voisinage. Si cette duplication est insuffisante, les individus adaptés peuvent disparaître sous l'effet destructeur des croisements et mutations (voir 4).
- Cependant, la diversité génétique de la population Π' est clairement inférieure à celle de la population Π . Si la duplication des individus prometteurs est excessive (un cas limite consistant à composer la population Π' de P clones de l'individu le meilleur de Π), il y a perte de la diversité génétique, et l'évolution est arrêtée.

Plusieurs heuristiques permettent de contrôler les effets de la sélection ; nous nous limiterons à décrire l'ajustement de la pression sélective [42] (cf 2.5.1) et le partage [26] (cf 2.5.2).

2.4.2 Le croisement

Principe

Le croisement permet à partir de deux individus parents, d'obtenir deux individus enfants. L'idée générale (intuitive) est que les enfants peuvent cumuler les bons côtés de leurs parents, et que le croisement permettra ainsi de progresser rapidement vers les régions optimales de l'espace de recherche. Le croisement est ainsi plus orienté du côté *exploitation*, que du côté *exploration*.
Le croisement constitue un opérateur majeur de l'approche AG [23], et son utilité est fortement contestée par la communauté de la programmation évolutive (EP) [16]. L'approche EP soutient notamment que le croisement peut avantageusement être remplacé par une mutation de probabilité convenable [17] (cf 2.4.3) ; voir également [33].

Dans le cadre des appr ches théoriques, le croisement est fort difficile à prendre en compte et il est le plus souvent modélisé comme un bruit [31, 13], avec une exception [8].

Méthode

Deux individus x et y sélectionnés dans la population Π' sont croisés avec une probabilité p_c (ou recopiés tels quels dans Π' avec une probabilité $1 - p_c$). La probabilité p_c appelée *taux de croisement* est généralement à valeurs dans l'intervalle $[0.2, 0.8]$. Le croisement élabore deux enfants x' et y' à partir des parents x et y[5].

[5]Certains auteurs ne considèrent qu'un enfant par couple ; deux fois plus de croisements sont alors effectués. Les résultats obtenus par cette variante ne se distinguent pas significativement des résultats obtenus par la variante usuelle.

Le croisement le plus simple (*croisement à un point*) consiste à tirer un indice k dans $[1, N]$, puis à construire x' et y' comme suit :

$$\left\{\begin{array}{llllll} x = & x_1 & x_2 & \ldots & x_N \\ y = & y_1 & y_2 & \ldots & y_N \end{array}\right. \rightarrow \left\{\begin{array}{llllll} x' = & x_1 & \ldots & x_{k-1} & y_k & \ldots & y_N \\ y' = & y_1 & \ldots & y_{k-1} & x_k & \ldots & x_N \end{array}\right.$$

Plus généralement, un croisement peut être modélisé par un masque c, avec $c = (c_1, ..c_N) \in \{0,1\}^N$, et :

$$\begin{array}{l} x_1 \ldots x_N \\ y_1 \ldots y_N \end{array} \rightarrow \begin{array}{l} x'_1 \ldots x'_N \\ y'_1 \ldots y'_N \end{array} \quad \text{avec} \left\{\begin{array}{llll} x'_i = x_i & y'_i = y_i & si\ c_i = 1 \\ x'_i = y_i & y'_i = x_i & si\ c_i = 0 \end{array}\right.$$

Parmi les croisements les plus fréquents, outre le croisement à un point qui correspond à un masque du type 1...10...0, citons le *croisement à deux points*, qui correspond à un masque du type 1..10...01...1, et le *croisement uniforme* [65], où chaque bit du masque est tiré uniformément dans $\{0, 1\}$.

Difficultés

Le croisement peut faire apparaître des enfants réunissant les bons traits de leurs parents. Mais il peut aussi se faire qu'aucun enfant n'hérite des bons traits des parents. On parle alors de croisement destructif (*disruptive*): les parents prometteurs disparaissent alors, et il faudra les redécouvrir. Nous proposerons (partie 4) une stratégie permettant de caractériser et de rejeter les croisements destructifs [40].

Le risque de croisement destructif est particulièrement important lorsque la fonction performance à optimiser comprend plusieurs optima. Croiser les individus proches de deux optima distincts conduit à obtenir des individus hybrides qui sont souvent peu performants et ralentissent l'évolution. Cet inconvénient peut être évité en s'abstenant de croiser des individus trop éloignés, selon l'heuristique dite de *croisement restreint* [26] (cf 2.5.3).

Enfin, le croisement défini ci-dessus est adapté au contexte binaire. L'approche des stratégies d'évolution (ES) [60] s'est focalisée sur la prise en compte d'un espace de recherche réel ($\Omega = \mathbb{R}^N$) et a proposé des opérateurs de croisement spécifiquement adaptés à ce contexte (voir aussi [42, 51]), que nous décrirons en 3.3.

De façon générale, il est en effet souhaitable de repenser les opérateurs de croisement et de mutation compte tenu de l'espace de représentation adopté (cf 3.2.3).

2.4.3 La mutation

Principe

La mutation consiste à reproduire un individu de la population Π'' , en l'altérant aléatoirement.

La mutation est l'opérateur d'évolution le plus important du point de vue de l'exploration de l'espace de recherche. En effet, la sélection ne permet pas d'explorer des zones nouvelles, et nous avons vu que les zones accessibles par croisement sont limitées de façon déterministe par la population courante. Seule la mutation peut donc permettre d'accéder à tout l'espace de recherche. Cette propriété d'*ergodicité* est cruciale dans l'intérêt de l'optimisation globale : l'évolution doit pouvoir explorer tout l'espace de recherche, afin d'avoir

une chance de déterminer les optima indépendamment de la population initiale. Enfin, seule la mutation permet de lutter contre le phénomène de *dérive génétique*, i.e. le fait qu'un bit prenne une valeur constante sur toute la population. Il y a alors convergence (souvent prématurée) à cette position, que seule une mutation chanceuse peut combattre.

Méthode
Pratiquement, chaque bit de chaque individu de Π' est remplacé par son complémentaire avec une probabilité p_m appelée *taux de mutation*, prenant dans la littérature des valeurs de l'ordre de 10^{-2} à 10^{-4}. En effet, choisir un taux de mutation trop élevé revient quasiment à remplacer l'évolution génétique par une marche aléatoire.

Difficultés
La mutation doit permettre d'obtenir tout individu de l'espace de recherche; mais il est souhaitable, notamment pour les dernières étapes de la convergence, d'obtenir en moyenne un enfant "proche" du parent. Cette condition ne présente pas de difficulté dans le cas booléen (le nombre de bits mutant étant en moyenne très faible).

Par ailleurs, la communauté EP [16, 17] estime que le croisement apparaît comme nécessaire pour les AGs du fait justement que ceux-ci emploient une mutation de probabilité et d'amplitude faibles. Une mutation permettant de larges sauts (ou macromutation) rendrait le croisement superflu (voir également [33]).

Enfin, la mutation présentée ci-dessus est adaptée au cas binaire. Son extension au cas d'un espace de recherche réel a été étudiée de manière approfondie, dans le cadre ES [60] comme dans le cadre AG [42] (cf 3.3.2).

2.5 Heuristiques

La simplicité fondamentale du mécanisme que nous venons de décrire, et sa lenteur reconnue, ont généré des multitudes de variantes et d'heuristiques, mises au point pour des applications particulières. Toutefois, comme le note Goldberg dans un article intitulé *Zen and the art of GA* [24], la plupart des heuristiques bénéfiques dans un contexte donné se révèlent inopérantes, voire nuisibles, dans la généralité des cas. Nous allons nous limiter à la description des trois heuristiques les plus courantes : les deux premières sont relatives à la sélection, la troisième est relative au croisement.

2.5.1 Ajustement de la pression sélective

Dans le cas d'une sélection proportionnelle (cf 2.4.1), un indicateur des effets de la sélection est la quantité $\frac{F_{Max}}{\bar{F}}$, rapport de la performance maximale et de la performance moyenne observée sur la population courante. Cette quantité, appelée *pression sélective*, donne l'espérance du nombre de copies du *meilleur* individu dans la population courante, lors de l'étape de sélection.

Comment varie la sélection en fonction de la pression sélective ? On voit que lorsque cette pression est voisine de 1, la sélection n'a pratiquement pas d'effet : chaque individu a (en espérance) exactement une copie. En revanche, lorsque la pression sélective est très élevée ($F_{Max} >> \bar{F}$), le nombre de copies du meilleur individu est très grand et il y a risque de dérive génétique.

Pour éviter ces deux extrêmes, l'heuristique dite d'ajustement de la pression sélective, ou mise à l'échelle (*fitness scaling*) consiste à fixer *a priori* l'espérance du nombre de copies que doit avoir le meilleur individu lors de la sélection

[23]. La valeur communément recommandée dans la littérature est la valeur 2. Cette recommandation est fondée sur l'empirisme ; à notre connaissance, aucune justification théorique n'est disponible.

Dans la pratique, le contrôle de la pression sélective peut être effectué de la façon suivante (mise à l'échelle linéaire) : il est théoriquement identique de chercher les optima de la fonction F, ou les optima de la fonction $\alpha F + \beta$. D'autre part, il est possible d'ajuster les coefficients α et β de façon que le ratio

$$\frac{\alpha F_{Max} + \beta}{\alpha \bar{F} + \beta}$$

soit égal à 2 pour la population courante Π_i . La sélection est alors opérée sur la base de la fonction $\alpha F + \beta$; les coefficients α et β sont recalculés à chaque génération. On trouvera dans [42] d'autres possibilités de mise à l'échelle

Dans le cas d'une sélection par le rang ou par tournoi (cf 2.4.1), la pression sélective est indépendante de la valeur des performances des individus; la sévérité de la sélection est alors uniquement déterminée par les valeurs (arbitraires mais constantes) donnée à chaque rang, ou par la taille des tournois. Dans la pratique, la sélection par tournoi est de plus en plus souvent utilisée [6]; signalons également qu'elle se prête à une exécution massivement parallèle des AG (les individus en compétition étant choisis dans un "voisinage" donné par la topographie des processeurs élémentaires [14]).

2.5.2 Partage

L'heuristique de partage (*sharing* [23]) concerne les fonctions performance ayant plusieurs optima globaux. Elle vise la découverte par l'AG de *tous* les optima globaux de F.
En effet, que se passe-t-il au cours d'une évolution génétique classique ? L'un des optima de F est approché avant les autres ; les voisins de cet optimum tendent à se reproduire en grand nombre et à saturer la population, ce qui empêche la découverte des autres optima. En général, l'évolution s'arrête au premier optimum découvert, par perte de la diversité génétique.

Pour éviter ce phénomène de saturation par le premier optimum découvert, l'heuristique de partage prend en compte non seulement la performance d'un individu, mais son isolement dans la population courante. Plus précisément, lors de la sélection, le nombre de copies d'un individu est obtenu en divisant la performance relative d'un individu, par le nombre de ses proches voisins dans la population. Un individu de performance relative moyenne, mais isolé, aura alors autant sinon plus de chances d'être sélectionné qu'un individu de performance relative très élevée, et situé dans une région très représentée dans la population courante.

Pratiquement, une fonction de similarité S est définie sur l'espace Ω :

$$S : \Omega \times \Omega \dashrightarrow \mathbb{R}^+$$

La fonction couramment utilisée est linéaire par morceaux, avec $S(x,x) = 1$ et $S(x,y) = 0$ si la distance de x et y est supérieure à un seuil s donné (au sens de la distance de Hamming dans le cas d'une représentation binaire, et de la distance euclidienne dans le cas de $\mathbb{R}^N$). On fixe alors :

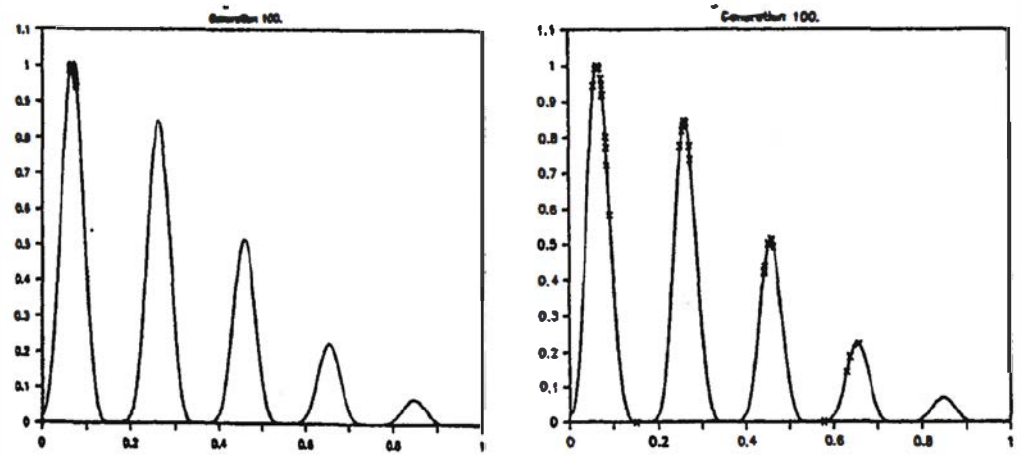
$$G(x) = \frac{F(x)}{\sum_y S(x,y)}$$

et la sélection est effectuée sur la base de la fonction G, i.e. l'espérance du nombre de copies de x est donnée par $G(x)/\bar{G}$.

Le coût de cette heuristique est en $\frac{P.(P-1)}{2}$ (où P est la taille de la population) ; il peut donc ralentir sensiblement le processus si le calcul des distances est du même ordre que celui de la fonction F.

Cette heuristique est efficace : elle permet non seulement de découvrir la plupart des optima globaux de la fonction performance F, mais aussi les optima locaux : seul un optimum global de F est découvert dans le cas d'une évolution sans partage (Figure 2.(a)). Le partage permet de découvrir la plupart des pics de F (Figure 2.(b)).

Théoriquement, le partage permet de déterminer les principaux pics de la fonction F, le nombre de représentants d'un pic étant inversement proportionnel à la hauteur relative du pic.



(a). Sans partage (b). Avec partage

Figure 2 : Découverte d'optima locaux multiples avec ou sans partage

Enfin, il existe d'autres techniques pour découvrir tous les optima d'une fonction par AG, au niveau tant théorique [8] que pratique [4, 39].

2.5.3 Croisement restreint

L'heuristique de croisement restreint (*restricted mating*) concerne elle aussi les fonctions performance ayant plusieurs optima (locaux ou globaux) et vient généralement en complément de la technique de partage présentée ci-dessus. Elle est fondée sur la remarque suivante : lorsque la population comprend des individus voisins de divers optima, le croisement entre individus proches d'optima distincts n'est pas judicieux : il débouche en général sur des individus moins performants (situés entre les pics), dont l'irruption ralentit l'évolution de la population.

L'heuristique dite de croisement restreint mise au point pour pallier cet inconvénient, consiste à interdire le croisement de deux individus dont la similarité est inférieure à un seuil t fixé par l'utilisateur. Plus précisément, le croisement de deux individus x et y est effectué avec une probabilité $S(x,y)$, où S est une fonction de similarité définie sur Ω comme celle qui intervient dans l'heuristique de partage (2.5.2) [4].

2.5.4 Remarques

Une discussion plus approfondie de l'optimisation de fonctions avec plusieurs optima globaux et locaux peut être trouvée dans [26, 39]. L'analogie biologique

de ce type d'optimisation est le phénomène de niche écologique : l'analogue d'un pic de la fonction f est une espèce biologique, le but de l'évolution consistant à découvrir simultanément plusieurs espèces qui doivent se partager un même espace, et telles que le croisement entre espèces différentes n'est pas viable.

De façon générale, on doit souligner que le contrôle d'un AG devrait être adaptatif, c'est à dire qu'il devrait dépendre de l'état de la population courante. Ainsi, l'heuristique de croisement restreint est adaptée en fin d'évolution, lorsque des optima distincts ont émergé (i.e. lorsque la population comprend des individus voisins des optima en question). Mais elle représente une contrainte inutile en début d'évolution.

Les heuristiques utilisées sont paramétrées compte tenu des connaissances *a priori* de l'utilisateur sur le problème et les solutions cherchées. Ainsi, le seuil intervenant dans les heuristiques de partage ou de croisement restreint dépend de la distance entre deux optima de la fonction performance ; ce seuil ne peut être déterminé *a priori*, et il est estimé par l'utilisateur en fonction de ses attentes.

La question du contrôle d'un algorithme génétique sera abordée de manière plus détaillée dans la quatrième partie.

3 Convergence, Codage et Problèmes Difficiles

Nous nous intéressons ici à l'efficacité des AG. Cette efficacité dépend essentiellement de la représentation du problème choisie et des opérateurs d'évolution agissant sur cette représentation.
En ce qui concerne la représentation, on introduit classiquement la différence entre le phénotype d'un individu (une solution dans l'espace intrinsèque du problème) et son génotype (le codage de cette solution dans l'espace de représentation que manipule l'AG). L'AG séminal défini par J.Holland [31] travaille toujours dans l'espace génotypique $\{0,1\}^n$ et ne considère pas le passage des phénotypes aux génotypes. Mais dans le contexte du voyageur de commerce par exemple [28, 52], le phénotype est un circuit (de N villes) alors que le génotype est en général une liste (d'où N génotypes possibles pour un phénotype).
Au niveau des opérateurs d'évolution agissant sur une représentation, nous allons indiquer les justifications classiques de l'approche binaire présentée ci-dessus[6]. Nous évoquerons ensuite la généralisation de la notion de schéma à un espace génotypique quelconque introduite par Radcliffe [51], ainsi que les principes heuristiques à respecter pour continuer à jouir des mêmes propriétés. Enfin, nous présenterons deux problèmes artificiels qui mettent en lumière quelques limites des AG [22, 45].

3.1 Schémas

Un schéma est un sous ensemble de l'espace de recherche $\Omega = \{0,1\}^n$ qui peut être décrit sous la forme d'une chaine de l'alphabet $\{0, 1, \star\}$, où le caractère '$\star$' peut prendre indifféremment la valeur 0 ou 1. Ainsi, la chaine 0*11* décrit un schéma de $\{0,1\}^5$, qui comprend les individus { 00110, 00111, 01110, 01111 }.

[6] La convergence de l'évolution est également abordée par [8], sous l'angle de l'étude d'un processus dynamique visitant des bassins d'attraction, et subissant des perturbations. Citons également [13], dont les résultats utilisent aussi une modélisation des AGs comme chaines de Markov et sont fondés sur la parenté entre AG et recuit simulé. Les résultats obtenus sont toutefois des résultats de convergence à la limite, peu utilisables dans le cadre algorithmique.

Par définition, la performance du schéma H est la moyenne des performances des individus appartenant à ce schéma :

$$F(H) = \frac{\sum_{x \in H} F(x)}{\#H}$$

où $\#H$ dénote le nombre d'individus de H. Par abus de langage, nous dirons qu'un schéma est performant, ou pertinent, si sa performance est supérieure à la performance moyenne $F(\Omega)$.
Il est important de noter que l'on n'a accès qu'à la performance d'un schéma *estimée d'après la population courante* :

$$\hat{F}_i(H) = \frac{\sum_{x \in H \bigcap \Pi_i} F(x)}{\#H \bigcap \Pi_i}$$

Le *parallélisme implicite* des AG consiste à considérer qu'en évaluant un individu x, on acquiert de fait une estimation de la performance des 2^N schémas auquel appartient x. Cette estimation peut naturellement être trompeuse, comme nous le verrons par la suite (cf 3.4.1).

3.1.1 Théorème des Schémas

Le théorème des schémas de J. Holland [31] exprime le fait que si la performance estimée relative d'un schéma est élevée, a ors l'espérance du nombre d'individus appartenant à ce schéma dans la population courante croît géométriquement au fur et à mesure des générations. On note $N_i(H)$ le nombre d'individus appartenant à H dans la population courante Π_i . Alors

$$N_{i+1}(H) \geq N_i(H) \frac{\hat{F}_i(H)}{\bar{F}_i} [1 - \epsilon] \qquad (1)$$

Le coefficient $\hat{F}_i(H)/\bar{F}_i$ correspond à l'effet de la sélection : la sélection duplique les individus d'un schéma en proportion de leur performance relative. Le terme ϵ exprime la perte de transmission du schéma H entre une génération et la génération suivante, due aux effets destructeurs des opérateurs d'évolution appliqués, et peut être calculé dans les cas particuliers suivants :

- Cas de la mutation.
  Définissons *l'ordre* d'un schéma, noté $o(H)$, comme le nombre de bits définis (i.e. de valeurs différentes de $\star$) de ce schéma. Alors, la probabilité que la mutation d'un individu de H n'appartienne pas à H, est majorée par $p_m.o(H)$.

- Cas du croisement à un point.
  Si un individu de H est croisé avec un individu n'appartenant pas à H, la probabilité qu'aucun de leurs enfants n'appartienne à H est majorée par la probabilité que le point de croisement intervienne entre les bits définis de H. Si donc $l(H)$ désigne la longueur du schéma H (différence entre la position du bit défini le plus à droite et celle du bit défini le plus à gauche), alors les enfants d'un individu de H n'appartiennent pas à H avec une probab lité inférieure à $p_c \cdot \frac{l(h)}{N-1}$.

Ainsi pour fixer les idées et sachant que la probabilité de mutation est négligeable devant la probabilité de croisement, le facteur ϵ de pe te d'un schéma de longueur 10 pour un problème comportant 100 bits est de .05 avec une probabilité de croisement $p_c = .5$.

3.1.2 Combinaison des Schémas

L'inégalité (1) ci-dessus exprime que lorsqu'un schéma est estimé performant, sur la base de la population courante, alors son exploitation est garantie. Ce qui est encore plus intéressant est que le croisement favorise l'émergence d'individus appartenant à plusieurs schémas si ces schémas sont compatibles, i.e ont une intersection non vide.

Pour que cette stratégie soit pleinement efficace, il est souhaitable que le principe des *Building Blocks*, énoncé par Goldberg [23], soit vérifié : *The user should select a [representation] so that short, low-order schemata are relevant to the underlying problem and relatively unrelated to schemata over other [defining] positions.*

L'idée en effet est qu'un schéma pertinent est d'autant plus facilement découvert que la performance de ses représentants ne dépend pas du contexte. Et d'autre part, un schéma est d'autant moins fragile sous l'effet des opérateurs de l'évolution qu'il est court et comporte peu de positions définies.
La recherche comporte ainsi un 'biais' en faveur des schémas courts ; or, la longueur d'un schéma dépend de la représentation du problème choisie, et donc de l'état des connaissances ou des conjectures de l'expert. En l'absence de telles connaissances ou *a priori*, ce biais peut se révèler inutile ou nuisible; il est alors préférable d'utiliser un croisement uniforme (cf 2.4.2) qui privilégie toujours les schémas d'ordre petit, mais ne comporte plus de biais positionnels [65].

3.2 Codage

Que deviennent les considérations ci-dessus lorsque la représentation choisie par l'expert n'est plus une représentation binaire ?

N. Radcliffe [51] a proposé une extension du théorème des schémas dans le contexte d'un espace de recherche qui est le produit cartésien d'espaces finis $\Omega = \Omega_1 \times \Omega_2 \times .. \times \Omega_N$. L'extension de la notion de schéma est appelée *forme*[7], une forme H étant donnée comme une chaine $\eta_1\eta_2..\eta_N$, où η_i est soit une valeur définie appartenant à Ω_i, soit la valeur indéfinie $\star$.

3.2.1 Alphabet Minimum

Dans ce cadre élargi, l'évaluation d'un individu x donne toujours une indication sur la performance des formes auquel il appartient. Il est clair par ailleurs, que l'information rela ive apportée est d'autant plus grande que le cardinal des espaces Ω_i est petit. D'où le principe de l'alphabet minimal énoncé par Goldberg [23] : *The user should select the smallest alphabet that permits a natural expression of the problem.*

3.2.2 Redondance et Performance

L'idéal consiste à définir une représentation non redondante, i.e. où un phénotype est associé à un génotype unique. Au cas où cette propriété ne peut être vérifiée, alors il est souhaitable (principe 1, redondance minimale) que les divers génotypes associés à un même phénotype soient "voisins", i.e. fassent partie d'une même forme de cardinal petit. Au cas contraire, l'AG devra traiter ces représentations d'un même phénotype comme des individus différents, ce qui augmente inutilement la taille du problème.

[7] Une extension plus générale que nous ne présentons pas ici s'intéresse aux formes définies comme les classes d'équivalence induites par une relation d'équivalence arbitraire.

En second lieu, pour être capable de découvrir des formes utiles, il faut que ces formes regroupent des individus de performance comparable. Il faut ainsi (principe 2, corrélation formes-performance) qu'il existe des formes de cardinal petit, regroupant des individus de performance relativement comparables[8].

3.2.3 Représentation et Opérateurs

Pour respecter la stratégie de combinaison des schémas, il faut tout d'abord que l'intersection de deux formes compatibles (i.e. d'intersection non vide) soit une forme (principe 3, clôture des formes).

En second lieu, le croisement doit respecter les formes, i.e. le croisement de deux individus d'une même forme H doit donner des individus appartenant à H (principe 4, fermeture du croisement).

Enfin, si deux formes H_1 et H_2 sont compatibles, le croisement entre individus appartenant respectivement à H_1 et H_2 doit pouvoir donner un individu appartenant à l'intersection des deux formes $H_1 \bigcap H_2$ (principe 5, ergodicité du croisement).

Le dernier principe est relatif à l'action de la mutation (principe 6, ergodicité de la mutation) : il doit être possible à partir de toute population Π, d'atteindre un point quelconque de l'espace de recherche par l'application d'un nombre fini de mutations.

Le respect de ces principes permet de démontrer le *théorème des formes*, analogue au théorème des schémas (1).

3.3 AG et espace des réels

L'adaptation des AG au cas d'un espace de recherche de nombres réel a été particulièrement étudiée. Nous nous limiterons à discuter quelques codages proposés et les opérateurs adaptés. Signalons que, parallèlement aux AG, les stratégies d'évolution [60] ont été historiquement mises au point pour des problèmes à variables réelles.

3.3.1 Le croisement des réels

Dans le cas d'un codage binaire des réels, les opérateurs usuels de croisement s'appliquent, mais violent les principes énoncés en 3.2.2 (les formes naturelles sont alors les produits d'intervalles).

Un opérateur de croisement directement conçu pour les réels consiste à utiliser des sommes barycentriques [42]. Ainsi, si l'espace de recherche Ω est $\mathbb{R}^N$, on définit :

$$\begin{array}{cccc} x_1 & x_2 & \dots & x_P \\ y_1 & y_2 & \dots & y_P \end{array} \rightarrow \begin{array}{cccc} x'_1 & x'_2 & \dots & x'_P \\ y'_1 & y'_2 & \dots & y'_P \end{array} \text{ avec } \begin{cases} x'_i = \alpha_i x_i + (1-\alpha_i) y_i \\ y'_i = \alpha_i y_i + (1-\alpha_i) x_i \end{cases}$$

Le croisement est ici représenté par le vecteur des α_i, où α_i est généralement compris entre 0 et 1. Notons qu'un tel croisement est compatible au sens de Radcliffe (cf 3.2.2) avec les formes réelles.

Par ailleurs, le choix de α_i dans l'intervalle $[-.5\ ,\ 1.5]$ peut aussi être judicieux, et permettre d'explorer les alentours de l'intervalle (x_i, y_i).

[8] La variance de la fitness, par rapport à l'ordre des schémas ou formes, donne ainsi une bonne indication quant à l'adéquation de la représentation choisie: plus cette variance est élevée et plus l'exploration de l'espace de recherche est susceptible d'être leurrée [14, 53].

3.3.2 La mutation des réels

D'après le principe de l'alphabet minimum (cf 3.2.1), les nombres réels devraient également être codés de façon binaire. Cependant, dans le codage binaire usuel, la mutation d'un bit de poids fort donne un enfant très éloigné du parent. Or, nous avons dit que dans l'intérêt de la convergence, il est souhaitable que la mutation conduise en probabilité à un enfant "voisin" du parent (cf 2.4.3). Pour résoudre ce problème, plusieurs solutions sont envisageables.
La première consiste à définir une probabilité de mutation qui dépende du bit considéré, la probabilité de mutation d'un bit de poids fort étant bien inférieure à celle d'un bit de poids faible [1].
La seconde possibilité, proposée par Caruana et al [7], consiste à adopter le codage Gray, où deux nombres entiers successifs diffèrent d'un unique bit ; pour ce codage, la mutation est en moyenne moins violente que pour le codage binaire standard.

L'alternative consiste à travailler directement sur les nombres réels. La mutation peut alors être effectuée par l'ajoût d'un bruit gaussien, de moyenne nulle et de variance à déterminer, comme pour les stratégies d'évolution [60], ou de fonction de répartition triangulaire [42]. Une solution élégante, toujours issue des stratégies d'évolution, consiste à ajuster cette variance, c'est à dire l'amplitude de la mutation, au moyen de l'évolution génétique elle-même : la représentation d'un individu comprend non seulement les valeurs des variables du problème, mais aussi l'amplitude de la mutation à appliquer à chaque variable. L'optimisation porte alors simultanément sur la valeur et le champ dans lequel on fait varier cette valeur.

La mutation sur nombres réels peut aussi chercher à réintroduire les valeurs extrêmes des domaines de variations autorisées, dans la population: le but est de garantir que l'évolution a toujours accès à *tout* l'espace de recherche). La mutation d'un réel donne alors l'une des valeurs extrêmes de son domaine de variation autorisé [51].

3.4 Problèmes difficiles

Après avoir décrit de façon abstraite comment faciliter la tâche d'un AG, nous allons illustrer les pièges et les trappes de ce type de recherche sur des problèmes dits "AG-difficiles".
Nous présentons tout d'abord le principe des problèmes trompeurs, reposant sur un couplage particulier entre la représentation et la fonction performance. Le problème de la *Voie Royale* est ensuite décrit : ce problème conçu *a priori* comme facile (ouvrant une "Voie Royale"), pose des difficultés inattendues.

3.4.1 Problèmes trompeurs

La notion de problème trompeur pour les AG (*GA-deceptive*) a été définie par Goldberg [22, 23] dans le contexte binaire. Cette notion a été étendue au cas d'une représentation quelconque par Radcliffe [51].

Intuitivement, supposons qu'un sous-espace ou forme H de l'espace de recherche Ω soit d'une performance élevée d'après les observations faites sur la population courante. Alors, cette forme H en viendrait à être de mieux en mieux représentée dans la population courante. Si par ailleurs, H ne contient pas les solutions optimales, alors l'exploitation de H risque d'empêcher à jamais la découverte des optima.

Formellement, considérons les schémas d'ordre k, i.e. comportant k positions définies. Une fonction F est dite *partiellement trompeuse* s'il existe des

schémas d'ordre k de performance supérieure à celle des schémas de même ordre contenant les optima globaux. F est *globalement trompeuse* si tout schéma d'ordre k est de performance supérieure à celle des schémas de même ordre contenant les optima globaux.

Prenons un exemple [23, 67]. Soit l'espace de recherche $\Omega = \{0,1\}^3$ et la fonction F définie par :

$$F(x) = \begin{cases} 3 & \text{si } x = 1\,1\,1 \\ 2 & \text{si } x \in 0\,*\,* \\ 0 & \text{sinon} \end{cases}$$

Cette très simple fonction se révèle trompeuse dans la mesure où la performance moyenne du schéma $H_1 = 1\,*\,*$, qui contient l'optimum, est inférieure à celle du schéma $H_2 = 0\,*\,*$ ($\bar{F}(H_1) = 3/4 < \bar{F}(H_2) = 2$); l'exploration risque ainsi de se trouver cantonnée dans la région sous-optimale H_2; les chances de découvrir l'optimum global de la fonction sont alors faibles...

La notion de problème trompeur demande toutefois à être maniée avec précautions. En effet, des heuristiques très simples (pour tout x, considérer aussi son complémentaire $\neg x$) permettent de résoudre trivialement des problèmes totalement trompeurs [66]. Par ailleurs, il semble clair que tous les problèmes intéressants sont au moins partiellement trompeurs [10]. Enfin, Grefenstette [29] souligne que la dynamique de l'évolution peut permettre d'échapper aux pièges des problèmes trompeurs (la "déceptivité"). En effet, l'évolution est trompée dans la mesure où elle exclut de la population des formes qui contiendraient les optima, au profit d'autres formes de performance moyenne supérieure. Mais c'est sur la performance moyenne *observée* que se fonde l'évolution : les populations successives peuvent ainsi se situer dans des sous-espaces qui amplifient ou au contraire combattent la déceptivité.

3.4.2 La Voie Royale

Le problème de la Voie Royale (*Royal Road*) a été conçu [45] dans le but d'étudier l'interaction des caractéristiques adaptées aux AG.
D'une part, ce problème est délibérément conçu en termes de schémas de base (*building blocks*) : la performance d'un individu est la somme des scores associés aux schémas pertinents auquels cet individu appartient. Et d'autre part, la structure des schémas pertinents est de type hiérarchique, un schéma pertinent d'ordre élevé étant défini comme l'intersection de deux schémas pertinents d'ordre inférieur (Table 1).
Pratiquement, le schéma $H_{i,j}$ étant défini comme une suite de i bits consécutifs à 1, commençant au rang j, la première version du problème comprend les schémas $H_{8,1}, H_{8,9}, H_{8,1+8*k}$, $H_{16,1}, H_{16,17}, H_{16,1+16*k}$ $H_{32,1}, H_{32,33}$ et $H_{64,1}$, définis dans l'espace de recherche $\Omega = \{0,1\}^{64}$. Un schéma pertinent d'ordre i (i.e. comprenant i bits définis) est de score i.

									score
$H_{8,1}$	11111111	....	....	....	....	....	....	....	8
$H_{8,9}$	....	11111111	....	....	....	....	....	....	8
$H_{8,17}$	....	....	11111111	....	....	....	....	....	8
$H_{8,25}$	....	....	....	11111111	....	....	....	....	8
$H_{8,33}$	....	....	....	....	11111111	....	....	....	8
$H_{8,41}$	....	....	....	....	....	11111111	....	....	8
$H_{8,49}$	....	....	....	....	....	....	11111111	....	8
$H_{8,57}$	....	....	....	....	....	....	....	11111111	8
$H_{16,1}$	11111111	11111111	....	....	....	....	....	....	16
$H_{16,17}$	....	....	11111111	11111111	....	....	....	....	16
$H_{16,33}$	....	....	....	....	11111111	11111111	....	....	16
$H_{16,49}$	....	....	....	....	....	....	11111111	11111111	16
$H_{32,1}$	11111111	11111111	11111111	11111111	....	....	....	....	32
$H_{32,33}$	....	....	....	....	11111111	11111111	11111111	11111111	32
$H_{64,1}$	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	64

Table 1 : Les schémas pertinents pour le problème de la Voie Royale

$$F(x) = \sum_{x \in H_{i,j}} score(H_{i,j})$$

Contrairement aux attentes, ce problème n'est pas bien résolu par AG. L'analyse des auteurs [46] est que cet échec relatif est dû à un phénomène d'"auto-stop" (*hitchhiking*). En effet, le premier individu membre d'un schéma très performant, par exemple $H_{32,1}$, tend à envahir la population génétique peu après sa découverte. Or, il véhicule (emmène en stop, d'où le terme de "hitchhiking") sur ses 32 derniers bits une information qui n'a aucune raison d'être pertinente et qui peut chasser des schémas tels que $H_{8,33}$ ou $H_{8,41}$, s'ils avaient déjà apparu dans la population. La découverte du schéma $H_{32,1}$ ramène ainsi l'évolution à zéro, pour ce qui est de la découverte du schéma complémentaire $H_{32,33}$.

4 Contrôle d'un AG

Nous avons insisté précédemment sur le fait que les algorithmes génétiques doivent s'accompagner d'une représentation du problème et d'opérateurs soigneusement mis au point[9].

Nous allons maintenant aborder la tâche cruciale du contrôle d'un AG : de la valeur des nombreux paramètres et choix à ajuster dépend en effet qu'un AG soit simplement lent ou pratiquement inutilisable [12].

Après un bref aperçu de l'état de l'art, nous présentons et discutons une méthode originale de contrôle d'AG, fondée non plus sur la force brute, ni sur un modèle statistique, ni sur un méta-AG, mais sur l'apprentissage symbolique [44, 36].

Cette approche sera expérimentée sur les deux problèmes présentés en 3.4, ainsi que sur un problème d'optimisation combinatoire (problème du sac à dos multiple).

4.1 Etat de l'art

Nous distinguerons trois niveaux de contrôle :

- Un premier niveau concerne les paramètres globaux de l'algorithme, tels que taille de population, nombre maximum de générations, probabilité de mutation et de croisement.
- Un second niveau concerne la *structure* des opérateurs à appliquer à un individu, par exemple le choix d'un croisement uniforme ou à deux points.
- Un niveau de contrôle plus précis encore s'intéresse au *masque* des opérateurs à appliquer sur les individus : par exemple, où placer les points de croisements.

Notons que ces niveaux de contrôle sont interdépendants : ainsi, à supposer que l'on sache "parfaitement" déterminer les masques de mutation et de croisement adaptés aux individus, la question de déterminer les probabilités de croisement

[9] Selon [34], les deux réactions négatives aux AG, les plus fréquentes, sont typiquement : 1- *Je l'ai essayé en presse-bouton et ça n'a pas marché* et 2- *Je l'ai essayé sur un problème que je savais déjà résoudre et c'était ridiculement plus lent.*

et de mutation devient moins pressante.
La littérature offre trois approches du contrôle d'un AG : la force brute (à la main, ou par "méta-AG"), l'approche statistique, et l'approche darwinienne elle-même, les paramètres à contrôler évoluant avec les individus.

La force brute consiste à faire de multiples essais, et à retenir le meilleur jeu de paramètres pour procéder aux expérimentations complètes. Cette voie, historiquement la première [57], reste la plus employée. Certains essais d'optimisation par AG ... des paramètres de AG [27] n'ont pas été concluant, malgré leur coût exorbitant.

Le contrôle peut également s'appuyer sur l'estimation et l'utilisation d'un modèle statistique. Ainsi, Grefenstette [30] construit un estimateur des effets des opérateurs d'évolution, pour ajuster *a priori* les probabilités de croisement et de mutation. Il peut être souligné toutefois, qu'une telle estimation statistique, faite au préalable et une fois pour toute, n'est pas nécessairement fiable au fur et à mesure que la population se restreint dans les régions de performance élevée.
L'approche plus adaptative de Davis [11] consiste à ajuster les diverses probabilités des opérateurs, en utilisant un algorithme de type "bucket brigade" pour récompenser les opérateurs qui ont conduit à des individus fructueux.

La troisième possibilité consiste enfin à s'en remettre à l'évolution elle-même. L'évolution peut en effet agir sur les paramètres de contrôle eux-mêmes, s'ils sont compris dans la représentation du problème !

Ainsi, Lee et Takagi utilisent le vecteur de la logique floue [37] : des règles extrêmement générales sont données au système, telles que : *Si la taille de la population est faible et la diversité (nombre d'individus distincts) petite, il faut augmenter le taux de mutation.* Le système ajuste au cours de l'évolution les fonctions d'appartenance définissant les notions de *population faible* et *dispersion petite.*

Spears [64] augmente la représentation des individus d'un bit, qui code le type de croisement applicable à l'individu, 2-point ou uniforme.

Schaffer et Morishima [58] doublent la representation des individus, pour inclure le masque de croisement le plus adapté à l'individu.

Une technique très similaire est utilisée dans le cadre des stratégies d'évolution, pour ajuster la mutation à appliquer sur des individus vecteurs de réels : la représentation s'augmente du vecteur des variances de mutation à appliquer sur les différentes composantes [60].

Dans la majorité des cas, l'optimisation porte à la fois sur l'individu, et sur le "type" d'évolution qui lui est le plus profitable : l'idée étant que les survivants soient munis des bonnes valeurs en ce qui regarde à la fois la performance, et les capacités d'évolution et de transmission lors des générations suivantes.

Pour voir l'intérêt d'une approche adaptative du contrôle, revenons sur les raisons de la lenteur d'un AG.

4.2 Pourquoi si lent ?

L'évolution génétique permet de découvrir des individus performants ; mais, de par sa nature stochastique, elle les perd assez souvent. L'importance de ce phénomène de "découverte volatile" (*fleetingly discovered*) est attestée par la différence entre le temps où un schéma intéressant apparait pour la première fois dans la population, et le temps où il s'installe (dispose de représentants depuis 10 générations au moins) [46].

Métaphoriquement, l'enjeu consiste à définir de quelle mémoire doit disposer l'évolution : si cette mémoire est trop fidèle, il n'y a pas de risque de perte des acquis, mais ceci peut rigidifier l'évolution et la cantonner dans des optima locaux. Inversement, si cette mémoire est volatile, l'évolution passe une partie du temps à redécouvrir des solutions connues — ce qui rallonge évidemment la recherche. Pratiquement, la mémoire de l'évolution est contrôlée au niveau de la sélection des individus : la perte des individus intéressants peut être diminuée ou évitée par (a) une reproduction partielle ou stationnaire (cf 2.4.1); (b) l'élitisme, qui consiste à ne remplacer un individu que par un individu meilleur; ou (c) la génération d'un grand nombre d'enfants et leur sélection en bas age (i.e. avant la phase de reproduction suivante) [60].

Mais le degré de contrôle et de mémoire souhaitable varie au cours de l'évolution : intuitivement, des stratégies différentes doivent être adoptées selon que l'on se trouve en situation de début, de milieu ou de fin de partie... d'où la nécessité d'un contrôle adaptatif. En contraste avec les précédentes approches du contrôle adaptatif fondées sur un usage "méta" de l'évolution [11, 64 37, 58], nous allons présenter et discuter une autre approche du contrôle, fondée sur un paradigme externe à l'évolution : l'apprentissage à partir d'exemples ou apprentissage inductif [44, 36].

4.3 Contrôle par Induction

Intuitivement, le contrôle par induction d'un AG part des remarques suivantes : l'évolution se compose de bons et de mauvais évènements (croisements, mutations, ...). L'apprentissage inductif permet, à partir d'exemples de tels évènements, de bâtir des règles caractérisant les classes d'évènements bons ou mauvais. Ces connaissances sont alors alors disponibles à toutes fins utiles (les fins souhaitables sont discutées ci-après) pour guider l'évolution au cours des générations suivantes.

Mais rappelons tout d'abord briévement les prérequis et la démarche de l'apprentissage inductif. Nous discuterons ensuite ce que peut être un exemple dans le contexte de l'évolution, ce que l'on peut apprendre à partir de tels exemples, et enfin comment utiliser les règles apprises de façon à contrôler l'évolution.

4.3.1 Apprentissage inductif

Nous nous restreindrons ici au cadre de l'apprentissage supervisé à partir d'exemples [44, 36] (voir aussi le chapitre de G. Venturini dans ce volume). Dans ce cadre, une fonction *classe* à valeurs discrètes est définie sur l'espace du problème Ω ; on dispose d'un ensemble de points de Ω, appelés *exemples d'apprentissage*, dont la classe est connue. Le but de l'apprentissage est de déterminer un estimateur de la fonction classe à partir des exemples disponibles. Dans le cadre de l'apprentissage inductif, l'estimateur est cherché sous forme d'une base de règles. Une règle se compose d'un schéma de l'espace du problème (ou hypothèse), et d'une classe donnée (appelée conclusion de la règle).

Ainsi, la Table 2 montre quelques exemples de $\{0,1\}^6$ appartenant aux classes + et −, ainsi qu'une règle.

E_1	1	1	1	0	0	1	+
E_2	0	0	0	1	1	1	+
E_3	1	1	0	0	1	1	−
E_4	1	0	0	0	1	1	−
E_5	0	0	0	0	1	1	+
R	⋆	⋆	⋆	0	1	⋆	−

Table 2 : Exemples d'apprentissage et une règle

L'induction cherche à déterminer une base de règles, optimales au sens d'une fonction des critères suivants : (a) généralité ou spécificité des règles (ordre du schéma hypothèse) ; (b) complétude (nombre d'exemples d'apprentissage appartenant au schéma hypothèse de la règle; e.g. R_1 couvre E_3, E_4 et E_5) ; et (c) correction (fraction des exemples couverts par la règle qui vérifient la conclusion de la règle; e.g. E_5 ne vérifie pas la conclusion de R_1).

L'induction procède en examinant les exemples de façon descendante ou ascendante. Les approches descendantes, illustrées par la famille des algorithmes ID3 [50], construisent des arbres de décision en sélectionnant successivement les attributs les plus discriminants, i.e. les composantes dont les valeurs donnent une information maximale concernant la classe des exemples. L'approche ascendante, illustrée par la famille des algorithmes AQ [44], considère successivement les exemples et cherche à déterminer pour tout exemple les règles les plus générales couvrant cet exemple, et ne couvrant pas (ou peu) d'exemples appartenant à d'autres classes. Les exemples couverts par ces règles sont ôtés de la base d'apprentissage, et la procédure est itérée tant que la base d'apprentissage n'est pas vide. Dans ce qui suit, nous utiliserons un algorithme d'apprentissage ascendant de type AQ, appelé *DiVS* [62] qui détermine avec une complexité polynomiale toutes les règles les plus générales de correction supérieure à un seuil donné.

4.3.2 Exemples relatifs à l'évolution

Pour appliquer l'induction au contrôle de l'évolution, il faut disposer d'exemples (de préférence faciles à décrire et à rassembler) représentatifs de l'évolution. Dans ce qui suit, les exemples considérés sont les évènements élémentaires de l'évolution, correspondant à la naissance par croisement ou mutation d'un nouvel individu. Cette approche, initialement conçue pour l'observation et le contrôle des croisements [63], s'étend en effet naturellement au cas de la mutation [54].

Un tel évènement est complètement défini par le masque de l'opérateur de croisement ou de mutation utilisé (cf 2.4.2), ainsi que par le ou les parents concernés. Dans ce qui suit, la description d'un exemple est donnée par le masque de l'opérateur concerné, et éventuellement la donnée du parent concerné (le plus performant des deux parents dans le cas du croisement).

Dans le cadre de l'apprentissage supervisé, il faut également que les exemples soient classés, sachant que la caractérisation des classes ainsi construites doit permettre ultérieurement de guider l'évolution. De façon naturelle, un évènement élémentaire est bon ou mauvais pour l'évolution, selon qu'il donne naissance à des individus plus ou moins performants que les parents. Plus précisément, un exemple est classé comme :

- *bon*, s'il conduit à un enfant plus performant que le parent (que le plus performant des deux parents, dans le cas du croisement);
- *mauvais*, si l'enfant obtenu (le meilleur dans le cas du croisement) est moins performant que le parent (ou le meilleur des parents); l'opérateur est alors dit destructif ;
- *inactif*, sinon.

4.3.3 Acquisition et Induction

Les exemples sont acquis par expérimentation sur la population courante (après la phase de sélection) :

- Un masque d'opérateur est généré aléatoirement en fonction des paramètres de l'AG (taux de mutation, type de croisement);
- Un ou deux individus (selon qu'il s'agit d'un opérateur de mutation ou de croisement) sont tirés dans la population courante;
- L'opération est effectuée conformément au masque et aux parents choisis. La description de l'exemple est donnée par le masque de l'opérateur, éventuellement complété par la description du meilleur des parents. La classe de l'exemple est déterminée selon la performance du ou des enfants, comparée à celle du ou des parents[10].

A partir de ces exemples, l'apprentissage bâtit des règles. Prenons un exemple (Table 3) :

	Chromosome						*Masque*						*Classe*
E_1	1	1	1	0	0	0	1	1	1	0	0	1	*bon*
E_2	1	1	1	0	0	0	0	0	0	1	1	1	*bon*
E_3	1	1	1	0	0	0	1	0	1	1	1	1	*mauvais*
E_4	1	1	1	0	0	0	0	0	1	1	1	1	*mauvais*
E_5	0	1	0	0	0	0	1	0	1	1	1	1	*bon*
R	1	1	1	$\star$	$\star$	$\star$	$\star$	0	1	$\star$	$\star$	$\star$	*mauvais*

Table 3 : Induction à partir d'évènements génétiques

La Table 3 montre une règle R induite à partir d'exemples de croisements à un ou deux points. Cette règle peut être interprétée de la façon suivante : si l'un des deux parents appartient au schéma 1 1 1 $\star$ $\star$ $\star$, et que le masque de croisement appartient au schéma $\star$ 0 1 $\star$ $\star$ $\star$, alors les enfants obtenus seront moins bons que les parents. En d'autres termes, cette règle déconseille d'utiliser un point de croisement entre le deuxième et le troisième bit, pour les parents du schéma considéré.

Cet exemple montre combien les règles apprises dépendent de la population courante : ainsi R ne peut être apprise avant que le schéma 1 1 1 $\star$ $\star$ $\star$ soit découvert (ait des représentants dans la population). Mais R ne sera pas apprise non plus si ce schéma a envahi la population : en effet, si les trois premiers bits ont des valeurs constantes, l'effet d'un point de croisement dans cette région sera nul.... La conclusion de R serait donc *inactif* et non plus *mauvais*.

Pratiquement, seules les règles significatives (couvrant plus d'un certain nombre d'exemples) sont retenues. L'induction se restreint ainsi à la détection des grandes tendances, en matière d'opérations destructives ou inactives.

[10] Notons que les exemples peuvent se révèler incohérents : le croisement de x_1 avec x_2 selon un masque c peut conduire à des enfants plus performants que les parents, alors que le croisement du même x_1 avec un autre parent selon le même masque de croisement c peut conduire à des enfants moins performants; comme un exemple comprend la description d'un seul parent, il est alors possible que deux exemples de *même description* appartiennent à des classes différentes, i.e. soient incohérents. Le risque d'incohérence augmente si les exemples sont décrits par le seul masque de l'opérateur concerné.
Ces incohérences, qui restent marginales dans la pratique, ne pénalisent cependant pas la démarche proposée dans la mesure où de nombreux algorithmes d'apprentissage (dont DiVS) permettent de gérer les incohérences [50, 9].

4.3.4 Contrôle par règles

Les règles apprises permettent d'estimer *a priori* si un nouvel évènement (croisement ou mutation de parents donnés) sera bon, mauvais ou inactif ; elles peuvent donc être utilisées pour filtrer les évènements souhaitables pour l'évolution. Trois types de contrôle sont envisageables :

- La première possibilité consiste à n'effectuer que des opérations bonnes (estimées bonnes d'après les règles). Cette solution est écartée dans la mesure où elle conduirait à privilégier considérablement la tâche d'*exploitation* aux dépens de la tâche d'*exploration* dévolue à l'AG (cf 2.4).

- La seconde possibilité consiste à rejeter (ne pas effectuer) les opérations estimées mauvaises. Cette option est celle d'un contrôle que nous dirons *classique* : après avoir appris qu'un certain type d'opérations donne de mauvais résultats, on ne l'emploie plus.

- La troisième possibilité consiste à rejeter les opérations estimées inactives. Cette option sera dite *moderne* : il s'agit de rejeter les opérations qui n'ont pas d'effets visibles sur la performance de la population.

Notons que ni le contrôle classique ni le contrôle moderne ne rompent l'équilibre entre exploration et exploitation : mais les règles mémorisent le fait que certains types d'exploration et/ou d'exploitation ont été effectués avec des résultats nuls ou négatifs; et cette mémoire permet de ne pas répéter de telles opérations inutilement.

La mise au point d'un contrôle inductif implique deux types de coûts :

- Le coût d'acquisition des exemples, qui revient à $2 \times K$ calculs de performance si la base d'apprentissage comprend K exemples (K est fixé à P, taille de la population dans les expérimentations qui suivent). Le facteur 2 est dû au fait que les exemples relatifs aux croisements demandent l'évaluation des deux enfants.
  Notons qu'au lieu de procéder par expérimentation, il serait envisageable d'acquérir les exemples simplement en observant l'évolution : le coût d'acquisition des exemples serait alors nul. La raison pour laquelle nous avons préféré procéder par expérimentation est la suivante : si les exemples fondant les règles de contrôle étaient issus d'une évolution elle-même contrôlée par règles, on pourrait craindre que les biais des règles de contrôle n'aillent en s'amplifiant (métaphoriquement, il y aurait "validation des préjugés"). Le fait de procéder par expérimentation permet de remettre à zéro la mémoire du contrôle.

- Le coût de l'induction, en $K^2 \times N$ pour l'algorithme DiVS que nous avons utilisé [62].

4.3.5 Limitations

Examinons les échecs possibles d'un contrôle inductif. Un premier cas de figure est celui où l'induction ne permet pas de construire de règles significatives ; ceci se produit typiquement lorsqu'aucun schéma intéressant n'a encore émergé. Le coût du contrôle est ici simplement non productif.
Un cas bien pire est celui où l'induction élabore des règles significatives, et où le contrôle fondé sur ces règles égare l'évolution.... Le contrôle est alors contre-productif. Ceci peut en particulier être le cas lorsque les règles ont été apprises sur une population trop différente de la population courante.

Examinons plus précisément les effets trompeurs éventuels d'un contrôle inductif, et considérons tout d'abord le cas du contrôle de type classique.

Supposons ainsi qu'un schéma prometteur $H = 1\ 1\ 1\ \star\ \star\ \star$ soit apparu dans la population, et que les règles indiquent qu'il ne faut pas détruire ce schéma, soit en mutant l'un des bits définis de H, soit en plaçant un point de croisement entre les bits 1 et 2 ou 2 et 3.

Supposons tout d'abord pour simplifier que les règles ne prennent en compte que les masques des opérateurs (et non le parent concerné). L'effet de bord des règles visant à préserver le schéma H sera d'interdire toute modification des bits 1, 2 et 3, pour tout individu ... La conséquence indésirable est la suivante : si un autre schéma désirable, disjoint de H, n'a pas été déjà découvert (par exemple $H' = 1\ 0\ 1\ \star\ \star\ \star$), il ne peut plus être découvert. Cet effet de bord est très pénalisant dans le cadre d'un problème trompeur (cf 3.4.1) où, par construction, il existe des schémas intéressants incompatibles.

Cependant, cet inconvénient disparaît si les règles prennent en compte la description des parents : il est alors possible de préciser que les bits 1, 2 et 3 ne doivent pas être modifiés uniquement dans le cas de parents appartenant au schéma H.

Considérons à présent les effets indésirables d'un contrôle de type moderne. Il est clair que ce type de contrôle pénalise l'exploitation des individus situés sur un plateau de performance (puisque la plupart des opérations appliquées à ces individus amènent des résultats nuls). Si de plus, les règles ne prennent en compte que le masque des opérateurs (et non les parents concernés), la fonction d'exploitation de *tous* les individus s'en trouve pénalisée.

Ce dernier inconvénient est diminué en réservant le contrôle de type moderne à la phase finale de l'évolution. Dans ce contexte en effet, la population est en voie de converger : les opérations inactives sont légion, la fonction d'exploration n'est plus convenablement assurée et la recherche s'en trouve fort ralentie. Il devient alors souhaitable de rediversifier la population et de rétablir l'exploration[11] [15]. La rediversification de la population peut être assuré en préférant toute opération, même destructive, aux opérations inactives.

La question devient ainsi de savoir quand passer d'un contrôle classique à un contrôle moderne. L'heuristique proposée se fonde sur un indicateur simple : la proportion des exemples d'apprentissage dans la classe des *inactifs*. Quand cette proportion dépasse un certain seuil, c'est le signe que la population converge, et que l'évolution aborde une phase de "fin de partie". Un contrôle moderne est alors choisi. Au cas contraire, on choisira un contrôle classique.

Indiquons que le passage d'un type de contrôle à l'autre peut dépendre du phénomène de niche écologique. Si l'évolution converge vers la découverte d'une seule espèce, la plupart des croisements sont inactifs bien avant d'avoir une population uniforme. Si plusieurs espèces sont à découvrir, et si les croisements entre ces espèces sont contre-indiqués, alors l'interdiction des croisements mauvais peut avoir le même effet que l'heuristique de croisement restreint (cf 2.5.3).

[11] Avec la précision suivante : certes, lorsque l'évolution a véritablement convergé, seule la tâche d'exploitation garde un sens (il n'est plus temps de renvoyer la population explorer les confins de l'espace). Mais dans la pratique, lorsque l'évolution converge, la question est toujours la même : ne s'agit-il pas d'une convergence prématurée ? Le fait de pouvoir rediversifier la population est donc toujours souhaitable.

4.3.6 Mécanisme d'interaction

Nous avons appelé "évolution civilisée" l'algorithme hybride obtenu par contrôle d'un AG par apprentissage inductif — en restant bien sûr conscients de la simplicité du modèle au regard de l'analogie ... Voici le mécanisme d'interaction proposé (Figure 3) :

1. Une première période dite d'évolution "darwinienne" comprend N générations classiques. Cette période darwinienne permet aux premiers schémas intéressants d'émerger, de façon à prévenir le premier cas d'échec discuté en 4.3.5 (aucune règle n'est détectée).

2. L'étape suivante comprend une phase d'acquisition des exemples et une phase d'induction des règles, comme décrit en 4.3.3. Nous reviendrons ci-après sur les échecs possibles de cette étape.

3. Suit une période dite d'évolution "civilisée", qui comprend M générations soumises au contrôle des règles : la seule différence d'avec l'évolution génétique classique est que les masques des opérateurs sont sélectionnés conformément à la base de règles, les masques prédits comme mauvais ou inactifs (selon que le contrôle courant est classique ou moderne) étant rejetés. (Le choix du type de contrôle est détaillé ci-après). Le nombre M est appelé *longueur de la civilisation* ; nous avons pris M = N dans tout ce qui suit.

4. Après ce nombre de générations, la population a évolué ; les connaissances stockées sous forme des règles ne sont plus nécessairement adaptées à l'état courant de la population. Une nouvelle phase d'apprentissage est donc nécessaire et on retourne à l'étape 2.

Nous appelons *civilisation* l'ensemble de la phase d'acquisition des règles, et des M générations qui suivent, obéissant à ces règles (Figure 3).

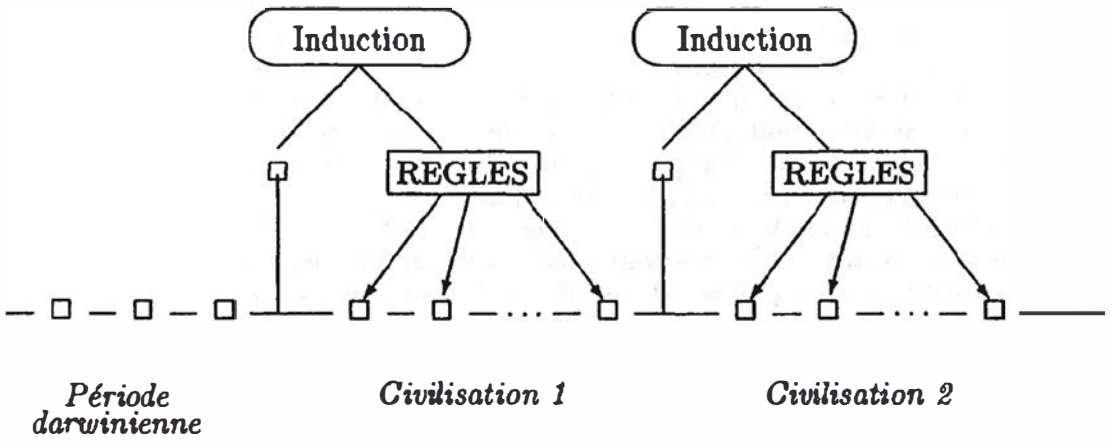


Figure 3 : Evolution civilisée : AG contrôlé par apprentissage inductif

Donnons quelques précisions sur les étapes cruciales et les possibles goulets d'étranglement du mécanisme ci-dessus.

Acquisition des Exemples.
Soit P la taille de la population génétique. P exemples sont construits comme en 4.3.3. Dans les expérimentations décrites, nous nous sommes restreints aux bases d'apprentissage composées uniquement ou bien d'exemples de mutations, ou bien d'exemples de croisement.
La constitution d'une base d'apprentissage échoue si tous les exemples appartiennent à la même classe. Cet échec est prévenu de la façon suivante : lorsqu'après un nombre maximal d'essais (fixé à $3 \times P$ dans les expérimentations qui suivent) on n'est pas par enu à constituer une base d'apprentissage "équilibrée" (où aucune classe ne regroupe plus d'une fraction F donnée des exemples), alors la période qui suit ne subit aucun contrôle et suit une évolution darwinienne classique.
Le choix du type de contrôle à effectuer dépend de l'importance des opérations inactives, conformément à la discussion qui précède (4.3.5) : si la base d'apprentissage comprend une fraction d'exemples inactifs inférieure à un seuil I donné, la phase courante est classique ; elle est moderne au cas contraire.

Elaboration des règles.
Les règles sont apprises à partir des exemples, à l'aide de l'algorithme DiVS [61, 62]. Notons que tout algorithme d'apprentissage pourrait sans doute être utilisé à cette fin pourvu qu'il soit capable de prendre en compte des données incohérentes.
Seules les L règles les plus significatives (recouvrant le plus d'exemples d'apprentissage) sont retenues. Le poids d'une règle est fixé au nombre d'exemples recou erts par cette règle.

Utilisation des règles.
Lors des périodes civilisées, les masques d'opérateurs sont (a) générés conformément aux paramètres de l'AG et (b) filtrés selon les règles : la classe d'une opération est déterminée d'après les règles qui cou rent le masque (en cas de conflit, par un vote à la majorité pondérée). Selon que la période courante est classique ou moderne, les masques classés destructifs (respectivement inactifs) sont rejetés.
Le fait que les règles apprises à partir d'exemples de croisement puissent servir à contrôler les masques de mutation, et vice versa, se justifie comme suit [54]: La mutation d'un individu x selon un masque m peut être vue comme le croisement de x avec son complémentaire $\neg x$ selon le masque de croisement $c = m$. Les règles induites des exemples de mutation fondent ainsi un contrôle des croisements qui pèche par sévérité : un individu x est croisé avec un y en général moins différent de x que ne l'est $\neg x$, d'une part, et le croisement donne lieu à deux enfants au lieu d'un, d'autre part. Pour les raisons inverses, le contrôle des mutations issu des règles induites d'après des exemples de croisement pèche par laxisme. Néanmoins, le contrôle des mutations et des croisements à partir des règles induites des seuls exemples de croisement ou de mutation reste possible et raisonnable.
Le contrôle peut également mener à un échec si tous les masques d'opérations générés sont rejetés. Pour prévenir cet échec, la base de règles est graduellement relaxée : après un nombre donné d'essais rejetés (fixé à P dans les expérimentations qui suivent), le poids des règles est décrémenté et les règles de poids nul sont éliminées. Seules les règles les plus significatives en viennent alors à être considérées.

4.3.7 Paramètres du contrôle

Récapitulons les paramètres nécessaires à un contrôle inductif d'AG:

- Le paramètre M contrôle la longueur d'une civilisation, c'est à dire le nombre de générations successives soumises aux mêmes règles. Empiriquement, nous avons fait varier ce paramètre de 1 à 10; la valeur 3 s'est révèlée convenir pour les problèmes considérés jusqu'à présent. Une valeur plus élevée peut avoir des effets nuisibles par suite d'un effet de "décadence" : les règles ont été apprises sur une population trop antérieure et différente des populations courantes, et le contrôle fondé sur ces règles égare l'évolution. Une valeur plus faible augmente le coût du contrôle sans résultats appréciables.

- La taille K d'une base d'exemples. Cette taille a été fixée au nombre P d'individus dans la population. Cette taille doit en effet être suffisante pour garantir la représentativité des règles apprises ; mais le coût du contrôle augmente avec K, linéairement comme le coût du calcul de performance ou quadratiquement comme le coût de l'induction, suivant les coûts relatifs des deux processus.

- La fraction maximale F de la base d'apprentissage appartenant à une même classe. S'il est impossible de constituer (en un nombre raisonnable d'essais) une base d'apprentissage où la représentativité de toute classe soit inférieure à F, l'apprentissage échoue et la période courante subit une évolution sans contrôle, ou darwinienne.

- Le seuil I sur la fraction d'exemples inactifs dans la base d'apprentissage; selon que la proportion des exemples inactifs est inférieure ou supérieure à I, la période civilisée courante subit un contrôle classique (rejet des opérations destructives) ou moderne (rejet des opérations inactives).

- Enfin, le nombre L de règles à retenir. Le coût du contrôle des masques croit linéairement avec L ; expérimentalement, nous avons pris L égal à 100.

4.4 Expérimentations

L'évolution civilisée présentée ci-dessus a été expérimentée sur trois problèmes : la Voie Royale (3.4.2), un problème trompeur [67] (3.4.1) et un problème d'optimisation combinatoire, le problème du sac à dos multiple [48].

Ces expérimentations ne peuvent avoir pour but de déterminer exhaustivement l'influence indépendante et conjointe de tous les paramètres du couplage AG-induction ; l'étude porte ainsi plus spécifiquement sur l'effet des paramètres F et I. F commande en effet les interruptions de l'évolution civilisée par des périodes darwiniennes (plus F décroît et plus la condition de représentativité de la base d'apprentissage est difficile à remplir) ; et I commande le passage d'une période civilisée classique à une période moderne (plus I décroît et plus il y a de périodes modernes).

Mais nous chercherons également, à travers l'efficacité de leur contrôle, à étudier le rôle des opérateurs de croisement et de mutation.

4.4.1 Protocole

L'algorithme génétique utilisé est un logiciel réalisé en interne, fondé sur les standards [23, 42] :

- les individus sont codés par des chaines de bits ; la taille de la population est fixée à 25 individus.

- la sélection est effectuée par tirage à la roulette, fondée sur la performance des individus ; la pression sélective *ps* (espérance du nombre de copies de l'individu le plus performant d'une population) est choisie comme une des variables d'expérimentation, et prend les valeurs 1.2 et 2.0.

- le taux de croisement est de 0.6, avec remplacement des parents par les enfants dans tous les cas; les masques de croisement util sés sont à deux points.

- le taux de mutation est de 0.005,

- l'évolution stoppe au bout de 1000 générations.

Sur chaque problème, plusieurs types d'optimisation génétique ont été comparés : un AG classique tout d'abord, i.e. sans contrôle (légende **AG**) ; puis deux AG avec contrôle des croisements par méta-AG, qui implémentent les contrôles proposés par Spears [64] d'une part, avec optimisation du type de croisement, uniforme ou à deux-points, à appliquer à un individu, (légende **Sp**) et par Schaffer et Morishima [58] d'autre part, avec optimisation du masque de croisement à appliquer à un individu, (légende **S-M**); et enfin, quatre AGs incluant divers types de contrôle inductif :

- Les règles induites des exemples de mutations sont utilisées pour contrôler les mutations (légende **M-M**);

- Les règles induites des exemples de mutations sont utilisées pour contrôler les mutations et les croisements (légende **M-MX**);

- Les règles induites des exemples de croisements sont ut lisées pour contrôler les croisements (légende **X-X**);

- Les règles induites des exemples de croisements sont utilisées pour contrôler les croisements et les mutations (légende **X-MX**).

Seuls les masques des opérateurs sont pris en compte pour apprendre les règles ; la description du parent n'intervient pas.

Les divers types d'évolutions sont comparés sous le rapport de leur capacité à atteindre le maximum de la fonction performance (qui est connu pour tous les problèmes considérés). Il est aussi intéressant d'étudier la dynamique de l'évolution (rapidité à atteindre de bonnes performances, qualité relative des meilleures performances atteintes) ; celle-ci est illustrée en traçant la moyenne des performances maximales atteintes pour un nombre donné d'évaluations de la fonction performance (y compris, dans les cas de contrôle inductif, les évaluations nécessaires à l'acquisition des exemples d'apprentissage).

Tous les résultats présentés sont des moyennes issues de 15 exécutions indépendantes : comme le souligne Kinnear [34], "Il ne faut jamais tirer de conclusions d'un essai unique".

4.4.2 La Voie Royale

Le nombre de bits d'un chromosome (individu) est 64. Les schémas élémentaires sont pris de longueur 2 (au lieu de 8 dans 3.4.2). La fonction performance ut lisée est modifiée comme dans [46] : si $N(i)$ est le nombre de schémas d'ordre i dont x est une instance, la performance de x est la somme, pour tous les ordres i, de 0 si $N(i) = 0$ et $i + N(i) \times 0.2$ sinon (au lieu de $N(i) \times i$).

La table 4 présente les résultats comparatifs obtenus par les divers types d'évolution sur le problème de la Voie Royale ainsi modifié. Les résultats des divers AG avec contrôle inductif ont été moyennés pour plusieurs couples de valeurs de I et F, au total sur 45 exécutions (les résultats détaillés figurent en table 5). Les paramètres I et F influencent le type de contrôle d'une période (classique ou moderne en fonction de I, et avec ou sans contrôle en fonction de F). Ils prennent leur valeurs dans {67%, 95%}, avec $I < F$.

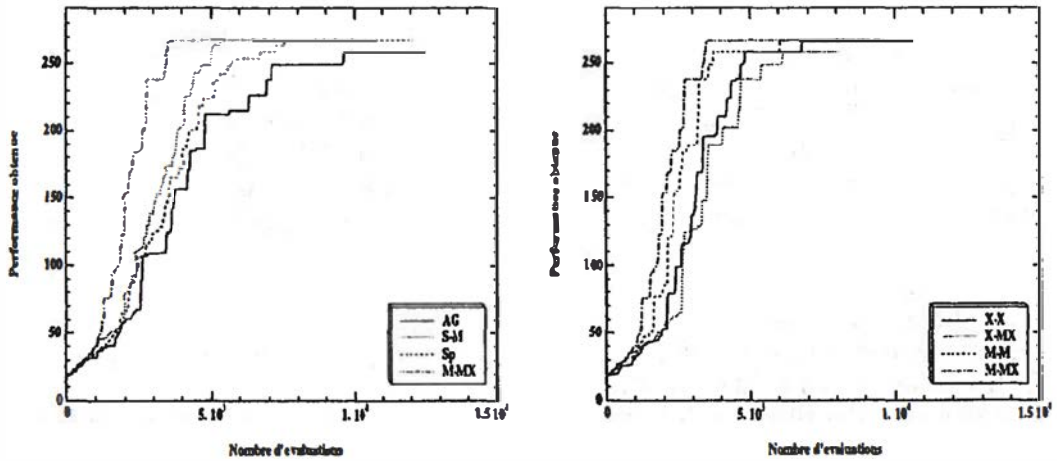
pression sélective	AG	AG avec Contrôle					
		méta-AG		Inductif			
		Sp	*S-M*	*M-M*	*M-MX*	*X-X*	*X-MX*
1.2	80	83	73	93	87	44	60
2	93	100	100	100	98	96	89

Table 4 : La Voie Royale, pourcentage de découverte de l'optimum avec et sans contrôle

La remarque la plus évidente est que lorsque l'AG "naturel" obtient de bons résultats (93 % pour une pression sélective 2.), alors le contrôle produit une amélioration faible, voire dégrade les performances (le contrôle inductif $X - MX$, qui contrôle les mutations et les croisements en fonction de règles induites d'exemples de croisement, obtient 89%).

Une seconde remarque est que le contrôle induit à partir d'exemples de croisement ($X - X$ et $X - MX$) donne de mauvais résultats : dans le seul cas où il améliore sur l'AG non contrôlé (93% à 96%) il est moins performant que tout autre contrôle du croisement.

Par contre, le contrôle induit à partir d'exemples de mutations donne comparativement de bons résultats pour une pression sélective de 2. et de très bons résultats pour une pression sélective de 1.2. Par ailleurs, une étude dynamique de l'évolution (Figure 4) montre qu'un AG avec contrôle induit à partir d'exemples de mutation converge plus rapidement vers l'optimum que les autres AG contrôlés et qu'un AG sans contrôle.



(a) Avec et sans contrôle (b) Divers contrôles inductifs

Figure 4 : Problème de la Voie Royale. Dynamique de l'évolution pour une pression sélective de 2.0, avec $I = F = 95\%$

Regardons plus en détail la façon dont l'efficacité du contrôle inductif varie en fonction des paramètres I et F. La table 5 donne les résultats obtenus pour F

à valeurs dans {50%, 67%, 95%} et I à valeurs dans {50%,67%}, (avec $I < F$). Ces résultats correspondent à une pression sélective de 1.2 (les résultats pour une pression sélective de 2. étant globalement meilleurs, les variations sont moins lisibles).

		I							
		50				67			
		M-M	*M-MX*	*X-X*	*X-MX*	*M-M*	*M-MX*	*X-X*	*X-MX*
	50	73	47	33	40				
F	67	60	80	47	73	93	73	27	80
	95	93	100	60	33	100	87	53	33

Table 5 : Pourcentage de réussite du contrôle inductif en fonction de I et F. Problème de la Voie Royale, pression sélective de 1.2

Une valeur élevée de F est préférable pour les contrôles induits des exemples de mutations ($M - M$ et $M - MX$) ; les périodes contrôlées ont alors peu de chances d'être entrecoupées de périodes darwiniennes. Le contrôle induit des croisements se confirme toujours moins performant que le contrôle induit des mutations, et devient même contre-productif si seuls les croisements sont contrôlés ; par ailleurs, une certaine proportion de périodes darwiniennes ($F = 67\%$) s'avère souhaitable.

4.4.3 Expérimentation sur un Problème Trompeur

Le problème considéré ici est la concaténation de 10 exemplaires du problème trompeur élémentaire décrit en 3.4.1 ; le nombre de bits d'un chromosome est donc 30.

Les résultats obtenus sont récapitulés dans la Table 6. Comme précédemment, les résultats du contrôle inductif moyennent 45 exécutions faites pour F à valeurs dans {50%,67%,95%} et I à valeurs dans {50%,67%}, toujours avec $I < F$.

pression sélective	AG	AG avec Contrôle					
		méta-AG		Inductif			
		Sp	*S-M*	*M-M*	*M-MX*	*X-X*	*X-MX*
1.2	80	83	87	91	89	46	49
2	93	90	87	100	100	60	49

Table 6 : Un problème trompeur, pourcentage de découverte de l'optimum avec et sans contrôle

Ces résultats confirment les observations précédentes : le contrôle induit des exemples de mutations se révèle ici très efficace, le contrôle induit des exemples de croisement est au contraire très mauvais.

De manière détaillée en fonction de I et de F, voici à présent les résultats obtenus par le contrôle inductif pour une pression sélective de 1.2. (moyennes sur 15 exécutions) ; nous reviendrons plus en détail sur ces tendances en 4.4.5.

		I							
		50				67			
		M-M	*M-MX*	*X-X*	*X-MX*	*M-M*	*M-MX*	*X-X*	*X-MX*
	50	87	93	40	27				
F	67	93	53	7	93	80	40	47	80
	95	100	93	73	27	80	87	57	13

Table 7 : Pourcentage de réussite du contrôle inductif en fonction de I et F. Problème trompeur, pression sélective de 1.2

4.4.4 Le problème du sac à dos

Nous traitons maintenant un problème d'optimisation combinatoire, afin d'évaluer le comportement du couplage proposé hors du champ des problèmes artificiels typiques des AGs. Les expérimentations ont été effectuées sur le problème dit du sac à dos multiple [35], et sur les données de référence proposées par Petersen [48].

Le problème du sac à dos multiple peut être modélisé comme suit:

– Etant donné un ensemble de P sacs à dos, de capacité respectives $c_1..c_P$,

– Etant donné un ensemble $\mathcal{O}$ de N objets, de profits respectifs $p_1..,p_N$,

– Sachant que l'encombrement de l'objet i relativement au sac à dos j est donné par $w_{i,j}$;

– Déterminer un sous ensemble d'objets, noté $X = x_1,..x_N$, avec x_i valant 0 ou 1, qui soit admissible, i.e., qui satisfasse les contraintes de capacités de tous les sacs à dos, et qui maximise le profit total :

$$\text{Max } Profit(X) = \sum_{i=1}^{N} p_i.x_i, \text{ avec } \sum_{i=1}^{N} w_{i,j}x_i < c_j, \forall j = 1..P.$$

Il s'agit d'un problème NP-complet où la représentation de l'espace de recherche est "naturellement" binaire puisqu'il s'agit de l'ensemble des parties de $\mathcal{O}$.

Pour appliquer l'approche AG dans le cadre d'un problème d'optimisation sous contraintes [42, 59], l'une des solutions possibles consiste à utiliser une fonction de performance avec pénalisation des solutions non admissibles. Nous avons utilisé la fonction performance proposée dans [35], avec toutefois une pénalisation moins draconienne :

$$F(X) = \begin{cases} \sum_{i=1}^{N} p_i x_i & \text{si } X \text{ est admissible} \\ \frac{r}{2} \sum_{i=1}^{N} p_i x_i & \text{sinon, où } r \text{ est la proportion de contraintes satisfaites} \end{cases}$$

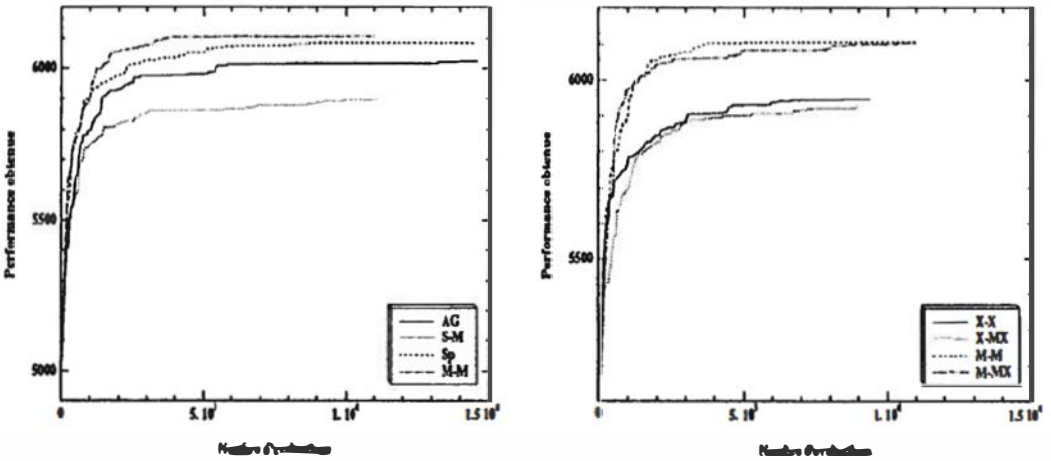
Les résultats qui suivent se rapportent au quatrième problème donné par Petersen [48], avec 20 objets et 10 sacs à dos, et sont représentatifs des résultats obtenus sur les trois jeux de données disponibles.

pression sélective	AG	AG avec Contrôle					
		méta-AG		Inductif			
		Sp	*S-M*	*M-M*	*M-MX*	*X-X*	*X-MX*
1.2	13	20	0	29	28	9	17
2	0	0	0	19	11	0	4

Table 8 : Le problème du sac à dos multiple, pourcentage de découverte de l'optimum avec et sans contrôle

Comme l'optimum est connu, nous pouvons comme précédemment indiquer la proportion des exécutions qui aboutissent à la solution (Table 8) :

La dynamique de l'évolution (Figure 5) montre bien que, même si l'optimum n'est pas toujours atteint, le contrôle inductif permet de converger plus vite vers des solutions meilleures.



(a) Ave et sans contrôle (b) Divers contrôles inductifs

Figure 5 : Problème du sac à dos multiple. Dynamique de l'évolution pour une pression sélective de 1.2, avec $I = F = 95\%$

Ce problème est atypique comparé aux problèmes précédents, à plusieurs égards. Tout d'abord une pression sélective faible (1.2) s'avère ici préférable. Les résultats détaillés du contrôle inductif en fonction des valeurs de I et F (Table 9) montrent également d'autres disparités : en particulier l'introduction de périodes darwiniennes ($F = 50$) est pour la première fois favorable au contrôle induit à l'aide des mutations.

		I							
		50				67			
		M-M	*M-MX*	*X-X*	*X-MX*	*M-M*	*M-MX*	*X-X*	*X-MX*
	50	27	47	7	13				
F	67	20	13	13	33	20	33	7	0
	95	40	20	7	13	40	27	13	27

Table 9 : Pourcentage de réussite du contrôle inductif en fonction de I et F. Problème du sac à dos multiple, pression sélective de 1.2

4.4.5 Modes d'évolution

Le contrôle inductif décrit permet à l'évolution de fonctionner selon trois modes : le mode darwinien, c'est à dire sans contrôle (favorisé lorsque F est faible et lorsque les opéra ions sont en grande majorité ou bien destructives, ou bien inactives) ; le mode classique, avec contrôle et rejet des opérations destructives (favorisé si F et I sont élevés et s'il y a peu d'opérations inactives) ; et enfin le mode moderne avec contrôle et rejet des opérations inactives (favorisé si F est élevé et I faible, et s'il y a beaucoup d'opérations inactives).

Le passage de l'un à l'autre de ces modes est commandé par la fraction f des exemples d'apprentissage inactifs à l'étape courante : s'il y a peu d'opérations inactives ($f < I$) l'évolution fonctionne en mode classique ; s'il y en a beaucoup ($I < f < F$), l'évolution fonctionne en mode moderne ; enfin s'il y en a trop ($F < f$), l'évolution fonctionne en mode darwinien. S'il n'y a pas d'opérations inactives, alors ou bien la fraction d des opérations destructives reste limitée ($d < F$) et l'évolution fonctionne en mode classique ; ou bien d est supérieur au seuil F, et l'évolution fonctionne en mode darwinien.

Pour les deux premiers problèmes considérés, la meilleure stratégie est celle d'un mode classique permanent (I et F élevés), rejetant les mutations destructives. Dans le troisième problème par contre, il se révèle préférable d'alterner de telles périodes classiques avec des périodes darwiniennes (F faible), i.e. de ne rejeter que périodiquement les mutations destructives ; en d'autres termes, l'homogénéisation de la population doit être ralentie (ce que confirme le fait qu'une pression sélective faible est préférable).

Expérimentalement, l'évolution peut tirer parti de telles stratégies de contrôle pour améliorer ses performances. La question posée devient alors de choisir en fonction du problème courant, la stratégie de contrôle adaptée.

Deux perspectives de recherche sont actuellement envisagées. La première consiste à assouplir le contrôle, en ne filtrant qu'un pourcentage donné des opérations ; les taux de contrôle souhaitables sont à ajuster au vu de la répartition des exemples d'apprentissage. Une seconde possibilité, à la Spears [64] consisterait à inclure dans la description d'un individu le type de contrôle qui doit lui être appliqué. L'évolution gèrerait ainsi le degré de contrôle inductif, applicable aux individus.

4.5 Discussion

Le travail présenté comporte deux aspects originaux par rapport aux travaux antérieurs visant à contrôler les AG : le fait qu'il s'agisse d'une approche inductive d'une part, et le fait que cette approche permette de contrôler également la mutation.

L'aspect majeur est sans doute celui du contrôle de la mutation. En effet, nombre de travaux dédiés au contrôle et à l'étude des effets destructifs du croisement [38, 64, 58, 15, 63] partent du principe que le contrôle du croisement est une tâche prioritaire — la probabilité de croisement étant en général d'un ou plusieurs ordres de grandeur supérieure à celle de la mutation.

Or, nos résultats montrent que le contrôle de la mutation peut avoir des effets bénéfiques comparables ou supérieurs à ceux du contrôle du croisement. Tout se passe en outre comme s'il suffisait de contrôler l'effet destructif des mutations ; le contrôle additionnel des croisements n'améliore en général pas les performances.

L'interprétation qui a été avancée [54] est la suivante. Les effets destructifs du croisement *peuvent être masqués compte tenu de la population courante* et de fait ils le sont couramment en situation de fin de partie, lorsque la population est homogénéisée. En revanche, *rien ne vient jamais masquer* les effets destructifs de la mutation ; ceci peut expliquer notamment les difficultés des AG standards pour des problèmes tels que la Voie Royale, où la mutation est en très grande majorité destructive dans de grandes régions autour de l'op imum.

Notons que le seul moyen jusqu'à présent disponible, pour lutter contre les effets destructifs des opérateurs et préserver les acquis de l'évolution, était le recours à la sélection et à l'élitisme. Or le contrôle de la mutation apparaît ici comme un moyen également puissant de préserver les acquis en fin d'évolution. Certes, les limitations d'une trop bonne mémoire sont toujours les mêmes : à

ne pas vouloir oublier, on peut se priver d'apprendre, ce qui se traduit par une convergence prématurée.
Cependant, le contrôle des mutations dispose de deux atouts contre une homogénéisation intempestive de la population, dont la sélection ne dispose pas. Tout d'abord, plus la population est homogène, et plus toute mutation tend à augmenter la diversité. En second lieu, le contrôle des mutations a pour effet de bord d'augmenter le nombre des bits touchés par la mutation d'un individu[12].

Plusieurs perspectives d'extension, en termes de stratégie de contrôle de la mutation, ont été évoquées en 4.4.5. Nous allons ainsi nous concentrer sur les autres aspects du contrôle inductif ; notons que ce travail constitue à notre connaissance la première tentative d'utiliser l'apprentissage inductif pour les fins de l'optimisation génétique[13]. La démarche présentée comporte trois articulations :

Tout d'abord, nous avons défini le contrôle en termes de *ce qui doit être évité*, plutôt que, comme chez [64, 58, 60], en termes de ce qui doit être préféré. Un tel raisonnement négatif diminue tout d'abord les risques d'un contrôle trompeur : en effet, le nombre de mauvaises opérations possibles est d'un ordre de grandeur supérieur au nombre de bonnes opérations, surtout en fin d'évolution ; il est donc moins hasardeux de déconseiller une opération, que de la préconiser.
Mais à côté de cet effet mécanique dû à la taille des ensembles considérés, nous soutenons que l'ensemble des opérations destructives est mieux connu que l'ensemble des opérations souhaitables. Pour un problème non trivial en effet, on ne peut déterminer *a priori* les opérations qui auront un effet bénéfique. En revanche, si une opération s'est révèlée une erreur, elle reste une erreur[14]. Nous connaissons donc une partie des erreurs à éviter : celles qui ont déjà été commises. Le problème pratique est de stocker l'ensemble des erreurs passées, ou de caractériser cet ensemble avec un coût raisonnable en temps calcul et en place mémoire.

En second lieu, le formalisme de contrôle adopté est celui des règles ; ce formalisme permet effectivement de stocker les erreurs passées, et plus généralement, il permet au contrôle d'agir au niveau des schémas d'opérateurs. Par opposition, les approches précédentes se sont intéressées à des contrôles de portée très générale (probabilité ou types d'opérateurs [11, 30, 64]) ou très locale (masque d'opérateur adapté à un individu [58]). Or, le fait d'accéder aux schémas d'opérateurs permet un contrôle à la fois général par rapport aux individus, et local par rapport à l'espace de recherche : muter certains bits et non certains autres, muter simultanément un ensemble de bits,... Ce formalisme permet donc d'exprimer de façon puissante le contenu du contrôle.

En troisième lieu, ce travail propose un moyen de déterminer le *contenu* des règles de contrôle, par induction à partir d'exemples. Notons que l'intérêt mani-

[12]En effet, les exemples de mutations comprennent initialement peu de bits actifs ; les règles apprises de ces exemples interdisent alors les schémas contenant ces exemples. Le tirage des mutations contrôlées se trouve ainsi progressivement biaisé vers des mutations comprenant de plus en plus de bits actifs.

[13]À l'inverse, de nombreux travaux en apprentissage appliquent les AG à la recherche de connaissances optimales ; voir le chapitre de G. Venturini dans ce volume.

[14]Sous l'hypothèse que le paysage de performance ne change pas et que la performance d'un individu ne dépend pas de la population courante.

feste d'un contrôle par règles ne suffit pas en soi à justifier l'usage de l'induction. Il serait en effet envisageable (et nous envisageons) d'utiliser l'évolution elle-même pour fixer le contenu des règles de contrôle. Selon les inconditionels de Dame Nature, l'évolution serait la mieux placée pour déterminer et ajuster ce qui est bon pour elle [24].
Selon une autre thèse, l'usage de l'évolution n'est justifié qu'en dernier recours, lorsqu'aucune autre méthode efficace n'est disponible. Il serait alors préférable d'utiliser des outils spécifiques de construction de règles, autrement dit des algorithmes d'apprentissage, plutôt que de vouloir faire jouer un rôle universel à l'évolution. Nous comptons sur de futures expérimentations pour contribuer au débat.

5 Conclusion

Les algorithmes génétiques sont une méthode d'optimisation dont la plus grande qualité est d'être applicable à une vaste gamme de problèmes. L'efficacité des AG dépend largement de la représentation choisie et peut être grandement améliorée par la conception d'opérateurs d'évolution *ad hoc*, utilisant les connaissances *a priori* de l'utilisateur et les propriétés de l'espace de représentation défini. Pour citer Kinnear [34],

> *A newcomer in the field should pretty much assume that significant adjustment of the fitness function, representation and other parameters will be required to solve any interesting and thereby non trivial problem.*

Ces remarques sont très voisines de celles qui concernaient il y a quelques décennies les résolveurs de problèmes en IA [32] : une phase d'émerveillement devant les capacités d'un outil universel est suivie d'une seconde phase, plus avertie, où l'on redécouvre l'intérêt de tenir compte des particularités des problèmes traités.

Le plus grand défaut des AG est sans conteste leur lenteur. Une cause identifiée de cette lenteur est l'absence de mémoire explicite de l'évolution génétique, qui passe une partie du temps à répéter des expériences passées, à oublier et redécouvrir les régions performantes de l'espace de recherche. Dans ce contexte, nous avons proposé une stratégie de mémorisation des erreurs commises afin de ne pas les répéter. Formellement, cette mémoire est stockée sous forme des schémas d'opérateurs destructifs ; concrètement, elle est générée par induction à partir d'exemples.
L'expérimentation sur des problèmes AG typiques, comme sur un problème d'optimisation combinatoire, conduit à un résultat très inattendu : c'est le contrôle de la mutation, bien plus que le contrôle du croisement, qui apparaît d'efficacité déterminante. En particulier, une mutation contrôlée assure que les acquis de l'évolution seront préservés, sans compromettre la diversité de la population (ou de manière moindre que la sélection).
Nous sommes loin d'affirmer que l'induction reste la meilleure manière de doter l'évolution d'une mémoire explicite ; nos perspectives de recherche concernent notamment l'utilisation de l'évolution elle-même pour déterminer le contenu de la mémoire souhaitable. En revanche, nous sommes convaincus que l'accès à une telle mémoire explicite est un pas crucial vers l'accélération de l'évolution : en filant la métaphore de l'évolution des systèmes biologiques, une mémoire explicite correspond à l'invention de l'écriture....

Remerciements

Les développements les plus récents de ce travail ont bénéficié des apports théoriques et expérimentaux de Caroline Ravisé, de l'Ecole Polytechnique, que nous remercions ici chaleureusement.
Cette recherche se situe dans le cadre du projet Inter-PRC "Approches symboliques-numériques pour la discrimination".

References

[1] J. Antonisse. A new interpretation of schema notation that overturns the binary encoding constraint. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 86–91. Morgan Kaufmann, June 1989.

[2] T. Bäck. *Evolutionary Algorithms in theory and practice.* New-York:Oxford University Press, 1995.

[3] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[4] D. Beasley, R. D. Bull, and R. R. Martin. A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 1(2):101–126, 1993.

[5] R. K. Belew and L. B. Booker, editors. *Proceedings of the 4th International Conference on Genetic Algorithms.* Morgan Kaufmann, 1991.

[6] T. Blickle and L. Thiele. A mathematical analysis of tournament selection. In L. J. Eshelman, editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 9–16. Morgan Kaufmann, 1995.

[7] R. A. Caruna and J. D. Schaffer. Representation and hidden bias : Gray vs binary coding for genetic algorithms. In *Proceedings of ICML-88, International Conference on Machine Learning.* Morgan Kaufmann, 1988.

[8] R. Cerf. *Une théorie assymptotique des algorithmes génétiques.* PhD thesis, Université de Montpellier II, March 1994.

[9] P. Clark and T. Niblett. Induction in noisy domains. In I. Bratko and N. Lavrac, editors, *Proc. of European WorkShop on Learning*, pages 11–30. Sigma Press, 1987.

[10] R. Das and D. Whitley. The only challenging problems are deceptive: Global search by solving order-1 hyperplanes. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 166–173, 1993.

[11] L. Davis. Adapting operator probabilities in genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 61–69, 1989.

[12] L. Davis. *Handbook of Genetic Algorithms.* Van Nostram Reinhold, New York, 1991.

[13] T. E. Davis and J. C. Principe. A markov chain framework for the simple genetic algorithm. *Evolutionary Computation*, 1(3):269-292, 1993.

[14] K. A. DeJong and J. Sarma. On decentralizing selection algorithms. In L. J. Eshelman, editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 17–23. Morgan Kaufmann, 1995.

[15] K.A. DeJong and W.M. Spears. A formal analysis of the role of multi-point crossover in genetic algorithms. *Artificial Intelligence*, 5:1–26, 1992.

[16] D. B. Fogel. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*. IEEE Press, 1995.

[17] D.B. Fogel and L.C. Stayton. On the effectiveness of crossover in simulated evolutionary optimization. *BioSystems*, 32:171–182, 1994.

[18] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. New York: John Wiley, 1966.

[19] S. Forrest, editor. *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*. Morgan Kaufmann, 1993.

[20] S. Forrest and M. Mitchell. What makes a problem hard for a genetic algorithms : Some anomalous results and their explanation. *Machine Learning*, pages 285–319, 1993.

[21] A. Giordana, L. Saitta, and F. Zini. Learning disjunctive concepts by means of genetic algorithms. In Cohen W. and Hirsh H., editors, *Proceedings of ICML-94, International Conference on Machine Learning*, pages 96–104. Morgan Kaufmann, 1994.

[22] D. E. Goldberg. Simple genetic algorithms and the minimal deceptive problem. In Davis L., editor, *Genetic Algorithms and simulated annealing*. Morgan Kaufmann, 1987.

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

[24] D. E. Goldberg. Zen and the art of genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the* 3^{rd} *International Conference on Genetic Algorithms*, pages 80-85, 1989.

[25] D. E. Goldberg and K. Deb. A comparative study of selection schemes used in genetic algorithms. In J. G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69-93. Morgan Kaufmann, 1991.

[26] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Proceedings of the* 2^{nd} *International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum Associates, 1987.

[27] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-16, 1986.

[28] J. J. Grefenstette. Incorporating problem specific knowledge in genetic algorithms. In Davis L., editor, *Genetic Algorithms and Simulated Annealing*, pages 42–60. Morgan Kaufmann, 1987.

[29] J. J. Grefenstette. Deception considered harmful. In D. Whitley, editor, *Foundations of Genetic Algorithms 2*. Morgan Kaufmann, 1993.

[30] J. J. Grefenstette. Virtual genetic algorithms: First results. Technical Report AIC-95-013, Navy Center for Applied Research in Artificial Intelligence, February 1995.

[31] J. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, 1975.

[32] C. Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 13:189–228, 1993.

[33] T. Jones. Crossover, macromutation and population-based search. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 73–80. Morgan Kaufmann, 1995.

[34] K. E. Kinnear Jr. A perspective on gp. In Jr K. E. Kinnear, editor, *Advances in Genetic Programming*, pages 3–19. MIT Press, Cambridge, MA, 1994.

[35] S. Khuri, T. Bäck, and J. Heitkötter. The 0/1 multiple knapsack problem and genetic algorithms. In *Proceedings of the ACM Symposium of Applied Computation*, 1994. (http:\\www-cgi.cs.cmu.edu \afs\cs.cmu.edu\projet\ai-repository).

[36] Y. Kodratoff. *Introduction to Machine Learning.* Pitman Publishing, London, 1988.

[37] M.A. Lee and H. Takagi. Dynamic control of genetic algorithms using fuzzy logic techniques. In S. Forrest, editor, *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*, pages 76–83, 1993.

[38] J. R. Levenick. Inserting introns improves genetic algorithm success rate : Taking a cue from biology. In R. K. Belew and L. B. Booker, editors, *Proceedings of the* 4^{th} *International Conference on Genetic Algorithms*, pages 123–127, 1991.

[39] S. W. Mahfoud. A comparison of parallel and sequential niching techniques. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 136–143. Morgan Kaufmann, 1995.

[40] R. Manner and B. Manderick, editors. *Proceedings of the* 2^{nd} *Conference on Parallel Problems Solving from Nature.* North Holland Publishers, 1992.

[41] J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors. *Proceedings of the* 4^{th} *Annual Conference on Evolutionary Programming.* MIT Press, March 1995.

[42] Z. Michalewicz. *Genetic Algorithms+Data Structures=Evolution Programs.* Springer Verlag, 1992.

[43] Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano, editors. *First IEEE International Conference on Evolutionary Computation.* IEEE Press, June 1994.

[44] R.S. Michalski. A theory and methodology of inductive learning. In R.S Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning : an artificial intelligence approach*, volume 1. Morgan Kaufmann, 1983.

[45] M. Mitchell, S. Forrest, and J.H. Holland. The Royal Road for genetic algorithms : Fitness landscapes and GA performance. In F. J. Valera and P. Bourgine, editors, *Proceedings of the First European Conference on Artificial Life-93*, pages 245–254. MIT Press/Bradford Books, 1993.

[46] M. Mitchell and J.H. Holland. When will a genetic algorithm outperform hill-climbing ? In S. Forrest, editor, *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*, page 647, 1993.

[47] K. Park. A comparative study of genetic search. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 512–519. Morgan Kaufmann, 1995.

[48] C.C. Petersen. Computational experience with variants of the balas algorithm applied to the selection of r & d projects. *Management Science*, 13:736–750, 1967.

[49] P. Preux. Etude de l'uniformisation de la population des algorithmes génétiques. In J.-M. Alliot, E. Lutton, E. Ronald, and M. Schoenauer, editors, *Actes de la Conférence Evolution Artificielle*. Cepadues, September 1995.

[50] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[51] N. J. Radcliffe. Equivalence class analysis of genetic algorithms. *Complex Systems*, 5:183–20, 1991.

[52] N. J. Radcliffe. Set recombination and its application to neural network topology optimisation. Technical Report EPCC-TR-91-21, Edinburgh Parallel Computing Center, 1991.

[53] N. J. Radcliffe and P. D. Surry. Fitness variance of formae and performance prediction. In D. Whitley and M. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 51–72. Morgan Kaufmann, 1994.

[54] C. Ravisé, M. Sebag, and M. Schoenauer. An induction-based control for genetic algorithms. In J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution*. Springer-Verlag, 1996. à paraître.

[55] E. Ronald. When selection meets seduction. In L. J. Eshelman, editor, *Proceedings of the 6^{th} International Conference on Genetic Algorithms*, pages 167–173. Morgan Kaufmann, 1995.

[56] J. D. Schaffer, editor. *Proceedings of the 3^{rd} International Conference on Genetic Algorithms*. Morgan Kaufmann, 1989.

[57] J. D. Schaffer, R. A. Caruana, L. Eshelman, and R. Das. A study of control parameters affecting on-line performance of genetic algorithms for function optimization. In J. D. Schaffer, editor, *Proceedings of the 3^{rd} International Conference on Genetic Algorithms*, pages 51–60. Morgan Kaufmann, 1989.

[58] J.D. Schaffer and A. Morishima. An adaptive crossover distribution mechanism for genetic algorithms. In J. J. Grefenstette, editor, *Proceedings of the 2^{nd} International Conference on Genetic Algorithms*, pages 36–40. Morgan Kaufmann, 1987.

[59] M. Schoenauer and S. Xanthakis. Constrained GA optimization. In S. Forrest, editor, *Proceedings of the 5^{th} International Conference on Genetic Algorithms*, pages 573–580. Morgan Kaufmann, 1993.

[60] H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, New-York, 1981 – 1995 (2nd ed.).

[61] M. Sebag. Using constraints to building version spaces. In L. De Raedt and F. Bergadano, editors, *Proceedings of ECML-94, European Conference on Machine Learning*. Springer Verlag, April 1994.

[62] M. Sebag. Delaying the choice of bias: A disjunctive version space approach. In *Proceedings of ICML-96, International Conference on Machine Learning*, 1996. To appear.

[63] M. Sebag and M. Schoenauer. Controlling crossover through inductive learning. In Y. Davidor, H.-P. Schwefel, and R. Manner, editors, *Proceedings of the 3^{rd}*

Conference on Parallel Problems Solving from Nature. Springer-Verlag, LNCS 866, 1994.

[64] W. M. Spears. Adapting crossover in a genetic algorithm. In R. K. Belew and L. B. Booker, editors, *Proceedings of the 4th International Conference on Genetic Algorithms*. Morgan Kaufmann, 1991.

[65] G. Syswerda. Uniform crossover in genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1989.

[66] G. Venturini. Artificial evolution. In J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution*, chapter Deception and genetic algorithms. Springer-Verlag, 1996. à paraître.

[67] D. Whitley. Fundamental principles of deception in genetic search. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991.