Les algorithmes génétiques sont des algorithmes d'exploration s'inspirant des mécanismes de la sélection naturelle et de la génétique. Ils sont basés sur les principes de survie de structures les mieux adaptées et sur les échanges d'informations. À chaque génération, un nouvel ensemble de créatures artificielles (codées sous forme de chaînes de caractères) est construit à partir des meilleurs éléments de la génération précédente. Bien que reposant fortement sur le hasard (et donc sur un générateur de nombres aléatoires) ces algorithmes ne sont pas purement aléatoires.

La littérature étant extrêmement prolyxe, nous renvoyons le lecteur d'une part sur le forum usenet `comp.ai.genetic` et sa FAQ. La meilleure production en français est sans conteste le livre de David Goldberg qui commence à dater [3]. Il existe de nombreux documents introductifs en langue anglaise, entre autres [1, 5, 6].

1 Introduction

L'appellation Algorithmes Evolutionnaires (EAs) est un terme générique pour décrire les approches informatiques reposant sur les principes de l'évolution comme éléments clefs de leurs fonctionnements. Il existe un grand nombre d'algorithmes évolutionnaires : algorithmes génétiques, programmation évolutive, stratégies évolutives, systèmes de classifieurs et programmation génétique. Toutes ces approches ont en commun la simulation de l'évolution de structures individuelles au travers de méthodes de *sélection*, *mutation* et *reproduction*. Ces processus reposent sur la *performance* des structures individuelles dans leur *environnement*, on parle aussi d'*adéquation*. Plus précisément, les EAs maintiennent une population de structures, qui évoluent en accord avec des règles de sélection et d'autres opérateurs, appelés opérateurs de recherche. Chaque individu possède une mesure de son adéquation dans l'environnement : la *fitness*. La reproduction s'appuye sur cette mesure pour favoriser la prolifération des individus les mieux adaptés. La recombinaison et la mutation perturbent ces individus fournissant ainsi des heuristiques pour l'exploration de nouvelles solutions. Bien que relativement simple ces algorithmes sont suffisants pour assurer des mécanismes de recherche robustes et puissants. Le pseudo-code suivant montre le fonctionnement d'un EA :

```
// intitialisation du temps
t = 0
// initialisation de la population
P = initPop
// evaluation de la fitness individuelle
evalFitness(P,t)
while not over: // over critere d'arret
   t = t+1
   Q = selection(P,t)
   recombinaison(Q)
   mutation(Q)
   evalFitness(Q,t)
   P = survivants(P,Q,t)
finWhile
return best + informations utiles
```

- Pour les algorithmes génétiques, les individus sont représentés par leurs chromosomes (sous forme de chaîne de caractères).
- La programmation évolutive, conçue initialement par J. Fogel en 1960, est une stratégie d'optimisation stochastique. À la différence des algorithmes génétiques, elle ne repose pas sur un codage particulier des individus (on peut l'appliquer directement sur les structures que l'on veut faire évoluer - réseaux de neurones par exemple), elle s'intéresse aux relations existant entre parents et enfants plutôt qu'à chercher à émuler des opérateurs génétiques particuliers.
- Un système de classifieurs peut être vu comme un système cognitif qui réagit à son environnement (à mettre en parallèle avec l'approche animat). Pour cela, on dispose d'un environnement, de capteurs permettant de connaître l'environnement (ce qui s'y passe), d'effecteurs (pour agir) et d'un système fonctionnant comme une boîte noire similaire à un système de production (règle `si-alors`). Les informations provenant des capteurs sont interprétées comme des messages, les règles réagissent à ces messages, et font des enchères, la meilleure enchère est sélectionnée, la règle correspondante est choisie (déclenchée) et la partie `alors` renvoie les informations aux

effecteurs. Lorsqu'une règle est choisie, elle paye une contribution aux règles qui ont participé aux enchères mais qui n'ont pas été sélectionnées.

- La programmation génétique est une extension des algorithmes génétiques aux programmes. C'est-à-dire que les individus sont des programmes exprimés sous forme d'arbres syntaxiques, les langages de la famille LISP se prêtent particulièrement bien à cette approche. L'opérateur de croisement agit en échangeant des sous-arbres, le lecteur interessé est renvoyé à la page de Koza `www.genetic-programming.com/johnkoza.html`, ou au site `www.genetic-programming.org`.

Des informations plus précises se trouve dans la faq de `com.ai.genetic`.

2 Objectif

Les algorithmes génétiques sont des algorithmes d'exploration fondés sur la sélection naturelle et la génétique. Ils utilisent les principes de la survie des structures les mieux adaptées, les échanges d'informations pseudo-aléatoires. Ils reposent de manière intensive sur le hasard mais ne sont pas purement aléatoires. Développés à l'origine par J. Holland et son équipe au sein de l'université du Michigan [4]. La recherche des algorithmes génétiques a pour but : l'amélioration de la robustesse et l'équilibre entre la performance et le coût nécessaire à la survie dans un environnement. Ils sont utilisés comme alternative à l'optimisation de fonction lorsque les méthodes "classiques", telles que les méthodes indirectes qui recherchent à atteindre les extrema locaux en résolvant des sytèmes d'équations et les méthodes directes par suivi de gradient ou par énumération, ne sont pas applicables ou ont échoué. Ce qui différencie les AG des autres approches peut s'exprimer en quatre points :

1. Les AG utilisent un codage des paramètres et non les paramètres eux-mêmes.
2. Les AG travaillent sur une population de points et non sur un point particulier.
3. Les AG n'utilisent que les fonctions étudiées, pas leurs propriétés (telle que la dérivabilité ou autre).
4. Les AG utilisent des règles de transitions probabilistes et non déterministes.

3 AG

Pour les AG nous allons modifier légèrement l'algorithme initial, en :

- manipulant des chaînes de caractères sur un alphabet prédéfini ;
- remplaçant l'opérateur de recombinaison par celui de croisement ou "cross-over" ;
- supprimant la notion de survivants

Nous allons revenir maintenant sur les différents constituants des algorithmes génétiques. Avant de pouvoir utiliser un AG, il nous faut définir un codage adapté au problème (les *chromosomes*), ainsi qu'une fonction (*fitness*) qui va caractériser l'adéquation de la solution (représentée par son chromosome) au problème.

3.1 Chromosomes, gènes, ...

On suppose dans cette approche qu'une solution du problème peut être représentée par un ensemble de paramètres, ces paramètres sont regroupés sous la forme d'une chaîne de caractères. J. Holland a le premier montré pourquoi une représentation sous forme binaire était efficace, bien qu'il existe d'autres approches, nous nous restreindrons dans ce qui suit à un alphabet binaire.
En génétique, un *gène* est un composant du chromosome, la valeur d'un gène est appelé *allèle*. L'ensemble du matériel génétique est appelé le *génotype*, le *phénotype* est quant à lui une instance particulière d'un génotype. La fonction d'évaluation dépend du phénotype. On peut construire un tableau de correspondance entre les termes issus des systèmes biologiques et artificiels (table 1):

naturel	artificiel
chromosome	chaîne de caractères
gène	trait, caractéristique, paramètre
allèle	valeur du trait
génotype	structure
phénotype	solution

Table 1: Naturel Artificiel

3.2 Fitness

Cette fonction est déterminée en fonction du problème à résoudre et du codage choisi pour les chromosomes. À un chromosome particulier, elle attribue une valeur numérique, qui est supposée proportionnelle à l'intérêt de l'individu en tant que solution du problème. Dans le cas d'une optimisation de fonction, la fitness est en première approximation la fonction que l'on cherche à optimiser, mais cela n'est pas toujours le cas. Par exemple si nous nous intéressons à la survie d'un organisme cherchant à accomplir une tâche (recherche d'objets et évitement d'obstacles) ; dans ce cas la fonction d'évaluation sera une composition de l'âge atteint par la bestiole (expression de son aptitude à survivre dans l'environnement), du nombre d'objets trouvés, pénalisée en fonction du nombre d'obstacles rencontrés.

3.3 Sélection

Pour cette phase, nous supposons connu : les individus de la population au temps t, la fonction d'évaluation. Il va falloir trouver une manière de choisir les individus répondant le mieux à notre problème. Il s'agira donc de privilégier les individus ayant un score au-dessus de la moyenne, et de pénaliser ceux en-deça de cette moyenne. Plusieurs approches sont possibles :

1. La roue de la fortune "fortune wheel" On associe à chaque individu A_i son évaluation f_i, on constitue une roue dont la valeur maximale est donnée par $\sum_i f_i$. On applique alors l'algorithme suivant :

```
function Wheel(int p){
  sigma := ∑i  fi
  S = ∅
  for(j = 0 ; j < p ; j++){
   k = 0 ;
   x = random(sigma) ;
   y = sigma ;
   for(i = 0; i<n && y > x ; i++){
     y = y - fi ;
     k = i ;
   }
   S = S ∪ { Ak } ;
  }
}
```

2. Méthode des fractions : On effectue le calcul en deux étapes, dans un premier temps, chaque individu est sélectionné autant de fois que la valeur de la partie entière du rapport entre son score f_i et du score moyen de la population $\overline{f} = \frac{\sum_i f_i}{n}$. Dans un second temps on constitue une roulette à partir des parties décimales de chaque $\frac{f_i}{\overline{f}}$.

3. Méthode élitiste : on force la population sélectionnée à contenir les k meilleurs individus de la population, puis on complète par l'une des méthodes précédentes.

4. Recalage de la fitness : en fait cette méthode est un préalable aux autres méthodes, elle consiste à recalibrer les scores effectifs soit par compression (pour éviter que les éléments les meilleurs soient trop privilégiés), soit par étirement (pour favoriser les éléments les meilleurs). Soit par une méthode de "recentrage", typiquement il s'agit d'ordonner les individus en fonction de leur score puis de sélectionner les différents éléments soit par

une stratégie linéaire, soit une stratégie exponentielle. D'une manière générale ces méthodes sont utilisées après étude mathématique des valeurs de fitness et en fonction d'un objectif qui est soit la convergence accélérée (que l'on priviligiera principalement en fin d'évolution, soit au contraire en vue d'éviter une convergence prématurée et donc principalement lors des premières générations).

3.4 Cross-Over

Le cross-over est une méthode permettant de construire de nouveaux individus à partir de leurs parents. Il existe deux grandes approches, le cross-over simple qui consiste à choisir un point de croisement compris entre 1 et $(l-1)$ où l est la longueur de la chaîne et à recombiner les enfants comme étant le début d'un chromosome et la fin de l'autre. Par exemple, si l'on considère les deux chromosomes suivants `00110011` et `10101010` et que l'on choisisse comme point de croisement la valeur 3, on obtiendra les deux nouveaux individus suivants : `001 01010` et `101 10011`. La seconde approche est le cross-over double, qui consiste à sélectionner deux points de croisements et à échanger les parties intermédiaires. Ainsi en reprenant l'exemple précédent et en considérant les points 3 et 6 on obtient les deux nouveaux individus : `001 010 11` et `101 100 10`. À noter que le cross-over simple peut être considéré comme un cross-over double si l'on interprète les chromosomes comme des chaînes circulaires, et le deuxième point de croisement comme étant en position l.

3.5 Mutation

La mutation est appliquée individuellement à tous les nouveaux individus obtenus après croisement. Elle a pour but d'altérer aléatoirement une information avec une faible probabilité (de l'ordre de 1 pour mille). L'intérêt de la mutation et de pouvoir explorer aléatoirement une autre portion de l'espace de recherche, garantissant ainsi que tous les points de l'espace ont une probabilité non nulle d'être testés.

3.6 Convergence

Si l'AG a été correctement implémenté, la population évolue petit à petit au cours des générations successives et l'on constate que le score moyen de la population évolue globalement vers l'optimum de la fonction à maximiser. La *convergence* est la progression vers cet optimalité. Un gène aura convergé si $n\%$ de la population possède cette caractéristique. La population a convergé lorsque chaque gène a convergé.

4 Quelques applications

Les algorithmes génétiques ont été appliqués à de très nombreux domaines[1]. Dans la recherche de solutions de jeu de stratégie, l'optimisation de réseaux de neurones [7, 8], la recherche de solutions de problèmes NP [2], la simulation de cellules biologiques, la reconnaissance de formes. L'étude d'animat au moyen de classifieurs.

References

[1] D. Beasley, D. Bull, and R. Martin. An overview of genetic algorithmes: Part 1, fundamentals. *University Computing*, 15(2):58–69, 1993.

[2] K. De Jong and W. Spears. Using genetic algorithms to solve NP-complete problems. In *International Conference on Genetic Algorithms*, pages 124–132, 1989.

[3] David E. Goldberg. *Algorithmes Génétiques*. Vie artificielle. Addison-Wesley, June 1994. ISBN 2-87908-054-1.

[4] John H. Holland. *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control and AI*. Ann Arbor, The University of Michigan Press, 1975.

[5] M. C. South, G. B. Wetherill, and M. T. Tham. Hitch-hiker's guide to genetic algorithmes. *Journal of Applied Statistics*, 20(1):153–175, 1993.

[1] `http://www.cs.gmu.edu/~eclab/publications.html`

[6] Darrell Whitley. A genetic algorithm tutorial. Technical Report CS-93-103, Department of Computer Science, Colorado State University, march, 10 1993. URL www.cs.colostate.edu/~whitley/. voir aussi Q.10.5 de la faq de comp.ai.genetic.

[7] Xin Yao. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 8: 539–567, 1993. URL www.cs.bham.ac.uk/~xin/.

[8] Xin Yao. Evolving artificial neural networks. *PIEEE: Proceedings of the IEEE*, 87(9):1423–1447, 1999. URL www.cs.bham.ac.uk/~xin/.