



# DarwinIT

---

## La résolution de problème inspirée par la nature.

Charles Darwin.

## Table des matières

Table des figures.....	2
Table des tableaux.....	2
Résumé.....	3
Abstract .....	3
Préface .....	4
Darwin, Bibliographie. ....	4
Problématique .....	5
Public visé.....	5
Hypothèses de travail.....	6
Contexte et bornes .....	6
La démarche / Les outils .....	7
Fil directeur .....	7
Etat de l'art.....	8
Historique .....	8
Terminologie biologique .....	9
Les algorithmes génétiques.....	11
Champs d'applications .....	15
Le dilemme du prisonnier, un exemple de résolution classique.....	17
Introduction de la thèse professionnelle. ....	20
Résolution de problèmes .....	22
Programmation automatique.....	22
Analyse et prédiction.....	26
Ajustement de réseaux neuronaux .....	28
Problème du voyageur de commerce : Un modèle d'efficacité pour les algorithmes génétiques.....	30
Enoncé .....	30
Historique .....	31
L'existant .....	31
Algorithme génétique.....	33
Conclusion .....	35
Bibliographie.....	36

## Table des figures

Figure 1 - Paire de chromosomes.....	9
Figure 2 - Crossover .....	10
Figure 3 - Chromosome en chaine de caractères.....	10
Figure 4 – Crossover à base de bits .....	11
Figure 5 - Mutation aléatoire .....	11
Figure 6 - Population initiale .....	14
Figure 7 - Equation d'adaptation à l'environnement .....	14
Figure 8 - Crossover .....	14
Figure 9 - Mutation aléatoire .....	14
Figure 10 - Arbre d'opérations pour l'expression $\sqrt{(A3)}$ .....	23
Figure 11 - Troisième loi de Kepler.....	23
Figure 12 - Exemple de crossover dans le cadre de la programmation génétique .....	25
Figure 13 - Projection de la structure tridimensionnelle de la myoglobine. (Gray & Haight, 1967, p. 419).....	27
Figure 14 – Représentation schématique d'un neurone. Modifié et réimprimé depuis (Heaton, p. 40) .....	28
Figure 15 - Schéma d'un réseau dit feedforward à backpropagation. Traduit de (Whitley, p. 50) .....	29
Figure 16 - Problème du voyageur de commerce résolu (src <a href="http://morihi.net/">http://morihi.net/</a> ) .....	30
Figure 17 - Problème du voyageur de commerce .....	32

## Table des tableaux

Tableau 1 - Nombre d'étapes par ville .....	33
Tableau 2 - Nombre d'étapes requises par ville / GA.....	34



## Résumé

Les organismes vivants ont démontrés à maintes reprises leurs extraordinaires capacités pour résoudre les problèmes auxquels ils faisaient face. En effet, les organismes vivant ont dû évoluer et muter afin d'être le plus adapté possible aux problèmes qu'offrait leurs environnements et ce, au prix de leurs survie. Cet état de fait a été formalisé par Charles Darwin au sein de la théorie dite de la « Survie du plus fort ». En s'inspirant des mécanismes mis en place par les êtres vivants afin de survivre ; des scientifiques ont imaginés des algorithmes permettant de résoudre des problèmes complexe et impraticables pour des algorithmes traditionnels. Ces méthodes vont de la création de réseaux de neurones artificiels que l'on peut entrainer pour une tâche spéciale ; comme on entraine son cerveau à faire des mathématiques ; à la génération de milliers de chromosomes que l'on laisse évoluer via les mêmes mécanismes que les êtres vivants—mutation et crossover—afin de trouver celui qui est le plus adapté à son environnement ; celui qui est le plus à même de résoudre le problème. Après avoir exposé les fondements de la théorie de Charles Darwin et les algorithmes qui en ont découlé ; nous présentons le problème du voyageur du commerce—connu depuis 180 ans—et toujours résolu de manière non-optimale par des algorithmes vieux de 50 ans. Finalement, nous présenterons un algorithme inspiré par les mécanismes qui ont permis aux êtres vivants de survivre ; qui réduit le nombre d'étape de  $10^{64}$  et le temps d'exécution par  $4^{63}$  comparativement aux approches classiques toujours utilisées aujourd'hui.

## Abstract

Living organisms have consistently demonstrated their amazing capacities to solve problems. Indeed, living organisms have to evolve—by successive mutation—to survive in their environments. This statement has been formalized by Charles Darwin and his famous “Survival of the fittest” theory. Being inspired by the mechanisms shown by living organisms; scientist creates algorithms able to solve complex and impractical problems for classical algorithms. These methods ranging from the creation of artificial neural networks that can be trained to perform a specific task; as same as we train our brains for mathematics; to the generation of thousands of chromosomes that we let evolve in the same ways as living organisms—mutation and crossover—to find the most fit to its environment; the one the more able to solve the problem. After presenting the Charles Darwin's theory and the resulting algorithms; we present the traveling salesman problem—known for 180 years—and still solved by 50-year-old and non-optimum algorithms. Finally, we introduce an algorithm inspired by living organisms that reduces the number of steps by  $10^{64}$  and the computational time by almost  $4^{63}$  comparing to classical—and still used nowadays—approaches.

## Préface

### Darwin, Bibliographie.

Naturaliste anglais (12 Février 1809 – 19 Avril 1882), né dans une famille de médecins. Après avoir abandonné ses études de médecine, Charles Darwin entre à Cambridge dans le but d'obtenir une charge de prêtre anglican. Mais il entreprend, en 1831, un long voyage de cinq ans autour du monde, en Amérique du Sud et dans les îles du Pacifique, comme naturaliste sur le navire de recherche Beagle. Durant cette période, il recueille une énorme quantité d'observations biologiques et géologiques. Les phénomènes naturels qu'il constate (distribution des fossiles) le persuadent qu'ils ne peuvent être expliqués par la seule création et que les espèces animales et végétales ne sont pas immuables.

Installé à Londres, Charles Darwin publie le récit de son périple, "Voyage d'un naturaliste autour du monde" (1839), et commence à exploiter la masse de données qu'il en a ramenées. En 1843, il s'établit définitivement à Downe, dans le calme de la campagne londonienne, pour poursuivre ses recherches. Il s'intéresse à Malthus<sup>1</sup> et aux pratiques de la sélection par les éleveurs.

Par analogie avec la sélection artificielle, il découvre le mécanisme de la sélection naturelle. Les individus d'une espèce les mieux adaptés à leur environnement subsistent, se reproduisent, les autres disparaissent. Seules les variations utiles à l'espèce sont transmises d'un individu à ses descendants. Ces caractéristiques nouvelles deviennent ainsi progressivement dominantes. Les espèces ne sont donc pas figées comme on le croyait jusqu'alors, en cohérence avec la Bible. Le cœur de la théorie de Darwin est exposé dans "De l'origine des espèces par voie de sélection naturelle" (1859) où sont décrits l'évolution générale et les principes de la sélection, et dans "La descendance de l'homme et la sélection sexuelle" (1871). Il convient de préciser que seule la génétique moderne qui se développa à partir des travaux de Mendel<sup>2</sup> a pu valider la théorie de Darwin et expliquer les modes de transmission des caractères héréditaires.

Karl Marx (1818-1883) a été l'un des premiers philosophes à se rendre compte de l'importance de ces travaux. Selon le terme de Sigmund Freud (1856-1939), Charles Darwin a infligé une profonde "blessure narcissique" à l'homme en montrant qu'il n'était ni une créature de Dieu, ni l'espèce élue de la nature, mais le fruit d'une longue évolution du règne animal. La théorie de l'évolution a détruit l'argument du dessein, de l'intention (divine), selon lequel la beauté du monde et la perfection des organismes vivants démontrent l'existence d'un Créateur.

Outre sa théorie sur l'évolution et la sélection, Charles Darwin a produit une œuvre scientifique de la plus haute importance. Chrétien dans sa jeunesse, Charles Darwin a progressivement évolué vers le déisme, puis vers l'agnosticisme à la fin de sa vie.

---

<sup>1</sup> (13 Février 1766 –29 Décembre 1834) connu pour ses travaux sur les rapports entre les dynamiques de croissance de la population et la production, analysés dans une perspective « pessimiste »

<sup>2</sup> (20 juillet 1822 - 6 janvier 1884) est communément reconnu comme le père fondateur de la génétique. Il est à l'origine de ce qui est aujourd'hui appelé les lois de Mendel, qui définissent la manière dont les gènes se transmettent de génération en génération.

## Problématique

Une des composantes clefs de l'évolution décrite par Darwin en son temps est la sélection naturelle, en effet les organismes qui sont le moins adaptés à leurs environnements auront plus de chance de mourir tandis que ceux qui disposent de meilleures adaptations auront tendances à procréer et donc à survivre. En procréant, les organismes les mieux adaptés créeront des descendance qui auront des aptitudes et des comportements améliorés par rapport à ceux de leurs parents. Ainsi les représentants de ces générations futures seront encore mieux adaptés à leurs environnements et donc posséderont des chances encore meilleures de survivre et d'engendrer de nouvelles générations. Cette explication bien connu de l'évolution naturelle peut être résumée par une phrase simple chère à Darwin « The survival of the fittest theory » (i.e « La théorie de la survie du plus fort »). Cette théorie stipule que ce processus continu d'évolution naturelle par la survie du plus fort, du plus adapté, sans discontinuité dans le temps, augmente les chances de survie dans un environnement relativement stable.

Pourrait-on répliquer ce genre de processus éprouvés par la nature depuis des millénaires pour construire de meilleurs algorithmes qui seraient capable d'évoluer, de s'améliorer pour être plus performant dans leurs environnements ?

## Public visé

L'audience attendu pour cette thèse est principalement constituée de professionnels de la résolution de problèmes qui souhaitent explorer des voies parallèles à celles communément enseignées. Nous présenterons notamment des alternatives plus que viables quant à la résolution problèmes de parcours de point (e.g : Algorithmes du chemin le plus court, Dijkstra<sup>3</sup> et consort<sup>4</sup>). Dans une moindre mesure cette thèse peut aussi s'adresser à des biologistes voulant modéliser des systèmes naturels complexes (ce qui n'est habituellement pas dans leur pratique qui consiste généralement à de l'observation d'espèces à reproduction rapide comme les drosophiles). Les algorithmes inspirés par la génétique moderne servent aussi en intelligence artificielle notamment dans le raffinement des connexions synaptiques dans un réseau de neurones artificiels, à ce titre là des ingénieurs du domaine pourraient aussi être intéressés par ce travail.

---

<sup>3</sup>(1930–2002) est un mathématicien et informaticien néerlandais du xx<sup>e</sup> siècle

## Hypothèses de travail

Les principales hypothèses liées à cette thèse sont :

- 1- Il n'y a pas eu d'avancée significative pour les algorithmes de systématiques (deep search), qui recherchent toutes les possibilités avant de retourner la meilleure.
- 2- Les avancées sur les algorithmes systématiques sont minimales et elles se basent souvent sur une augmentation de performance au détriment de la qualité des résultats ou inversement.
- 3- Les algorithmes génétiques sont les plus performants dans les espaces de recherches non linéaires.
- 4- Rapprocher les sciences informatiques de la biologie a toujours été l'un des rêves des grands noms de l'informatique.

## Contexte et bornes

Les avancées technologiques récentes de l'électronique et de l'informatique font sans aucun doute partie de la révolution la plus impressionnante de l'histoire des sciences humaines. Cette révolution, a permis d'augmenter la capacité des humains à comprendre et à contrôler leur environnement d'une manière complètement inimaginable il y a 50 ans. Cependant, l'objectif ultime de toutes ces années de recherche, créer une entité intelligente ou même une nouvelle forme de vie n'a toujours pas été atteint.

Cet objectif qui peut sembler complètement extravagant a néanmoins été approché tout au long de l'histoire de l'informatique. En effet, les pionniers de l'informatique que sont Alan Turing<sup>5</sup>, John von Neumann<sup>6</sup> et d'autres ont toujours voulu apporter plus d'intelligence aux programmes informatiques. Leurs visions de l'informatique étaient d'apporter ultimement la possibilité de se reproduire eux même et de leur apporter la capacité d'apprendre de leur environnement pour y être mieux adaptés. Les scientifiques qui ont été à la genèse de l'informatique étaient autant intéressés par l'électronique et l'informatique que par la biologie et la psychologie. Ce n'est donc pas une surprise de constater que les premiers projets informatiques de grande ampleurs n'ont pas été cantonnés à la calculation de trajectoire pour les missiles ou encore au décodage de communication militaire, en effet il y a aussi eu des travaux visant à modéliser le cerveau humain, simuler l'apprentissage humain ou encore modéliser l'évolution biologique.

Malheureusement ces grandes ambitions se sont évanouies au cours des années à cause de la difficulté de la tâche et du manque de moyen, mais depuis les années 80 certains chercheurs ont pris à bras le corps ces problématiques et se sont regroupés pour atteindre les objectifs des créateurs de l'informatique. Les premiers champs de recherche directement appliqués furent le réseau de neurones et l'apprentissage machine.

---

<sup>5</sup> (23 juin 1912 - 7 juin 1954) est un mathématicien britannique, auteur de l'article fondateur de la science informatique

<sup>6</sup> (1903-1957) est un mathématicien et physicien américano-hongrois. Il a apporté d'importantes contributions tant en mécanique quantique, qu'en analyse fonctionnelle, en théorie des ensembles, en informatique,

Le troisième champ de recherche qui nous concerne lors de ce travail est appelé le calcul évolutif (« Evolutionary computation ») à l'intérieur duquel les algorithmes génétiques que nous étudierons lors de ce travail est l'un des sous domaines les plus prometteur.

Ce travail n'étudiera pas les utilisations indirectes des algorithmes génétiques comme par exemple leur introduction pour réévaluer le poids des connexions synaptiques dans un réseau de neurones en complément ou en remplacement de la propagation arrière (« backpropagation »). Ce travail ne s'attardera pas non plus sur les domaines de recherches qui découlent de celui-ci, comme notamment ceux qui s'inspirent de l'ingénierie humaine plutôt que de la génétique comme l'algorithme dit recuit simulé (« simulated annealing ») qui s'inspire de la thermodynamique et qui est utilisé pour trouver les extrema d'une fonction.

## La démarche / Les outils

Afin de mener à bien ce travail je vais tout d'abord me familiariser avec la théorie de l'évolution de Darwin puis avec le vocabulaire génétique relié. Une fois ces connaissances acquises je passerai à une phase plus technique où je tenterai d'implémenter les algorithmes génétiques les plus prometteurs se basant sur les crossover et les mutations génétiques dans le but de montrer que les algorithmes génétiques (AG) sont plus performants que les algorithmes connus dans les espaces de recherche non linéaires notamment ceux du plus court chemin.

Les outils qui vont me permettent d'acquérir ces briques théoriques et techniques sont tout d'abord la littérature disponible comme les livres situés dans la partie bibliographie qui sont déjà en ma possession ainsi que les exemples disponibles sur Internet. Je profiterai aussi de mon échange universitaire à l'UQAM pour interviewer des professeurs Nkambou, Beaudry, Tremblay et Valtchev qui enseignent respectivement l'informatique cognitive, l'intelligence artificielle, le parallélisme et la fouille de données. J'interviewerai aussi Francis Palma un étudiant en doctorat qui a travaillé sur la Priorisation interactive des exigences en utilisant des techniques d'optimisation et des solveurs de contraintes basé sur des algorithmes génétique lors de son master. J'assisterai aussi aux cours de système tutoriaux intelligent et parallélisme haute performance durant la session Hiver 2013 dans le cadre de l'échange eXia / UQAM.

## Fil directeur

Le fil directeur de cette thèse professionnelles sera la découverte de ce que les chercheurs ont pu produire en s'inspirant de mécanismes biologiques et de montrer la pertinence de l'utilisation de ces algorithmes en entreprise pour la résolution de problèmes technologiques complexes réels reposant sur des espaces de recherches non linéaires. Enfin lors des ultimes pages de cette thèse professionnels je présenterai des exemples d'applications réels, créés pour l'occasion, qui démontreront les capacités des algorithmes génétiques à des cas réels.



## Etat de l'art

### Historique

Dans les années 1950 et 1960 de nombreux scientifiques ont étudiés de manière séparée les systèmes basés sur l'évolution dans le but de créer des outils qui optimise des problèmes technologiques complexes. L'idée principale derrière tout ça était de faire évoluer des populations de solutions candidates via des mécanismes issus de l'évolution comme la mutation et la sélection naturelle.

Rechenberg (1965, 1973) a créé le domaine de recherche dénommé « stratégie évolutionniste » qui était une manière d'optimiser des valeurs en temps réel. Ce champ de recherche perdura un moment, soutenu notamment par les algorithmes génétiques. Un autre domaine connexe fût introduit par Fogel, Owens et Walsh en 1966, il s'agit de la programmation évolutionniste qui consista à faire évoluer des solutions qui ont muté de manière aléatoire et qui disposait d'une fonction d'évaluation pour évaluer l'adaptation de la solution à son environnement.

Les deux domaines présentés précédemment ainsi que celui des algorithmes génétiques exposé ci-après forment la clef de voute d'un domaine plus grand nommé le calcul évolutif. (« evolutive computation »)

Les algorithmes génétiques (AG) furent une création de John Holland (1914- 2009) qu'il a dévoilés à la communauté scientifique dans les années 1960. Il fût aidé par certains de ses étudiants à l'université du Michigan. Au contraire des deux autres domaines, stratégie évolutionniste et la programmation évolutionniste, les algorithmes génétiques n'avaient pas pour but, à leurs création, de résoudre des problèmes spécifiques, en effet Holland les a conçus pour étudier les mécanismes d'adaptation au milieu dans lequel l'algorithme évolue selon les principes de l'évolution naturelle. Son ouvrage 1975 [7] « Adaptation in Natural and Artificial Systems » présenta les AG en tant qu'une abstraction de l'évolution naturelle et donna un socle de travail théorique pour les AG.

Un des autres contributeurs prolifiques à la recherche liée aux algorithmes génétiques est David Goldberg (1953). Il fût l'un des étudiants de Holland et a publié de nombreux ouvrages sur le sujet comme « Genetic Algorithms in Search, Optimization and Machine Learning (1989) ou encore « The Design of Innovation (2002) ».

Les algorithmes génétiques consistent à partir d'une population de chromosomes, par exemple une chaîne de caractères de 0 et de 1, pour aller vers une nouvelle population composée des descendants de la première génération en utilisant les mécanismes propres à la sélection naturelle tel que les crossover (enjambement), les mutations ou les inversions.

## Terminologie biologique

Pour comprendre les algorithmes génétiques avec la profondeur exposée lors de cette thèse il est nécessaire d'introduire et d'expliquer le vocabulaire génétique qui sera présent dans les prochains chapitres.

Chaque organisme vivant est composé de cellules et ces cellules sont elles-mêmes composées de chromosomes. Ces chromosomes sont constitués de chaînes d'ADN et constituent une carte d'identité unique pour chaque organisme vivant. Il est possible de diviser un chromosome en gènes qui occupent une position précise sur son chromosome, cette position est appelée locus. Les gènes sont responsables de marques identifiables comme la pilosité ou encore la taille. Les différentes solutions possibles pour une marque sont nommées allèles.

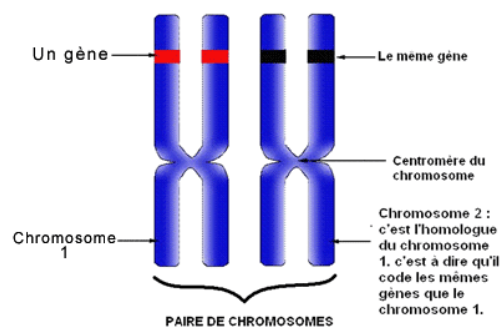


Figure 1 - Paire de chromosomes

Le premier paragraphe de cette explication sommaire des bases de la génétique est parti des chromosomes vers le plus petit, allèle. Nous allons maintenant aller vers le plus grand. L'ensemble des chromosomes d'un organisme est appelée le génome (i.e : Le génome humain) et les sous-ensembles de ce génome ont pour nom des génotypes. Au cours du développement d'un individu son génome évolue pour faire de lui ce qui le caractérisera en tant qu'organisme vivant unique, cette évolution porte le nom de phénotype.

Les êtres vivants qui, comme les humains, possèdent des chromosomes par paires font partie de la catégorie des diploïdes par opposition aux autres organismes, les apoïdes. Durant l'acte de reproduction des êtres vivants, a lieu ce que l'on nomme le crossover (enjambement, recombinaison en français) qui est un mécanisme où les gènes présents sur les paires de chromosomes des parents vont se mélanger pour ne former qu'un chromosome appelé gamète, ce gamète va ensuite se dupliquer pour reformer une paire de chromosomes complète. Pour les organismes apoïdes ce mécanisme est identique à ceci près que le chromosome résultant du mélange ne se duplique pas. Ainsi le crossover recombine, réarrange du matériel génétique présent chez les parents il n'y a donc pas de mutation ici. Les mutations apparaissent au niveau des nucléotides, qui sont les composants de l'ADN, en effet il arrive que l'enzyme en charge de la copie de l'ADN face erreur et introduise donc une mutation dans une séquence d'ADN. Cette mutation sera transmise ou non aux descendants si l'organisme mutant est suffisamment adapté à son environnement et aura tendance à devenir la norme sur une longue période d'évolution si cette mutation apporte un avantage compte tenu de l'environnement.

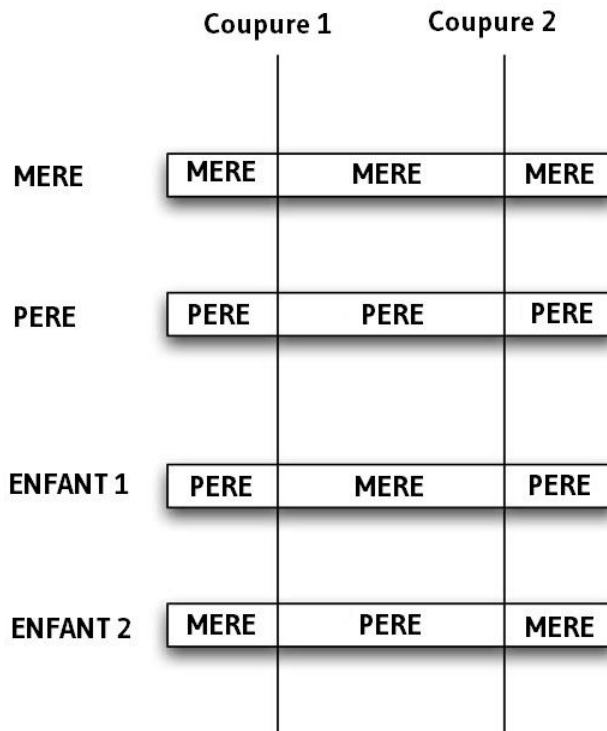


Figure 2 - Crossover

Cet avantage se mesure principalement via la viabilité de l'organisme mutant, est ce que ce dernier aura vécu assez longtemps pour se reproduire et en fonction de sa fertilité, combien de descendants directs et indirects aura-t-il ?

Il existe deux sortes d'évolution naturelle, la microévolution et la macroévolution, la première correspond à des changements de petites tailles sur la matériel génétique existant pendant une période de temps relativement courte. La macroévolution quant à elle fait référence à des changements majeurs sur une espèce sur une longue période de temps. Les algorithmes génétiques relèvent le plus souvent de la microévolution.

Si l'on transpose tout ce vocabulaire issue de la biologie aux algorithmes génétiques alors un chromosome est une solution donnée à un problème donné le plus souvent exprimée sous la forme d'une chaîne de bit (e.g : 1001). Les gènes, quant à eux, représentent une partie de la solution suscitée, par exemple si notre solution chromosomes est composée de deux gènes alors ceux si seront 10 (en rouge) et 01 (en bleu). Si nous gardons notre exemple, les allèles positionnées sur les différents locus du gène seront alors 1 et 0 pour le chromosome rouge et 0, 1 pour le chromosome bleu. Par analogie avec le monde réel le crossover consistera à recombinaison le matériel génétique de deux solutions possibles à un problème donné et la mutation au changement aléatoire d'un des bits composant un gène.

**1001**

Figure 3 - Chromosome en chaîne de caractères

Les algorithmes génétiques ressemblent donc étrangement à leur modèle : la nature. En effet, dans un algorithme génétique les différentes solutions potentielles sont des chromosomes composés de gènes. En manipulant ces gènes, de nouveaux chromosomes disposant de nouvelles caractéristiques sont créés via les mécanismes de crossover et de mutation aléatoire.

## Les algorithmes génétiques

Les algorithmes génétiques ne possèdent pas de définition universelle, néanmoins, pour pouvoir entrer dans cette catégorie, les algorithmes dits génétique doivent posséder quatre caractéristiques communes. En effet, ils doivent posséder un « pool » de chromosomes (i.e ensemble de chromosomes), un mécanisme de sélection pour détecter les mieux adaptés à leurs environnements, pouvoir générer des crossover pour créer des descendants et enfin la capacité de créer des mutations aléatoires.

Les algorithmes génétiques les plus simples fonctionnent uniquement avec trois opérateurs distincts : la sélection, le crossover et la mutation. La sélection consiste à choisir certains chromosomes dans le pool pour se reproduire et donner naissance à une nouvelle génération ; plus un chromosome maximise le résultat de la fonction qui évalue son adaptation à son environnement plus il sera choisi pour la reproduction. Le crossover échange de manière aléatoire les sous branches se trouvant après un certains locus pour créer deux descendants issu de la recombinaison du matériel génétique des parents. Le crossover est utile car il permet à des descendants d'hériter de caractéristiques de ses deux parents, cependant il n'introduit pas de nouveau matériel génétique dans les futures générations. Ainsi si nous prenons l'exemple de deux chromosomes (vert et rouge) qui sont des solutions possibles à un problème donné codé par une chaîne de bits, alors le crossover aura pour effet :

```
10001001110010010
01010001001000011

10001001101000011
01010001010010010
```

Figure 4 – Crossover à base de bits

Le troisième et dernier opérateur, la mutation, échange certains bits à l'intérieur d'un chromosome de manière complètement aléatoire mais avec une très faible probabilité. La mutation introduit donc du nouveau matériel génétique dans les nouvelles générations.

```
10001001101010011
```

Figure 5 - Mutation aléatoire

L'effet de ces mutations n'est pas prédictible, on ne peut pas savoir s'il elles seront positives ou négatives. Néanmoins les mutations négatives seront éliminées via l'opérateur de sélection puisqu'une mutation négative entrainera une réduction du score d'adaptabilité à l'environnement et donc le chromosome ne sera pas choisi pour la reproduction et la mutation éliminée du pool. Ces trois opérateurs copient rigoureusement ceux qui ont permis à tous les organismes vivants connus d'évoluer.

Une des composantes clefs de l'évolution naturelle est la sélection, en effet, les organismes les moins adaptés à leurs environnements auront tendance à mourir alors que les plus adaptés produiront de nombreux descendants qui posséderont cette caractéristique avantageuse amplifiée et qui à leur tour la transmettront.

Nous pouvons dès à présent imaginer ce que serait la cinématique d'un algorithme génétique simple mais reproductible par tous. La première étape (1) est de générer une population de chromosomes qui seront des solutions potentielles au problème donné. Les chromosomes sont composés de gènes dont les valeurs ont été initialisées au hasard. Cependant chaque chromosome doit représenter une solution complète au problème donné. (2) La population ainsi générée passe par la fonction qui calcule l'adaptation environnementale pour un chromosome donné. (3) La troisième étape consiste à sélectionner des chromosomes appartenant à la première génération pour les faire se reproduire afin de générer des descendants, cette étape correspond à l'opérateur de sélection et donc un chromosome a plus de chance d'être sélectionné pour la reproduction s'il maximise sa fonction d'adaptation environnementale et le même chromosome peut être sélectionné plusieurs fois. (4)

Une fois les paires de parents constituées il faut appliquer l'opérateur crossover qui va échanger des branches de matériel génétique entre les parents pour créer des descendants qui proviennent de la recombinaison des parents. A la suite de l'opération dite crossover, (5) il faut appliquer l'opération de mutation qui mute aléatoirement certains bits présents à certains locus de manière aléatoire. Lorsque les étapes 3, 4 et 5 ont été suffisamment répétées pour produire autant de chromosomes de seconde génération que de chromosomes de première génération (ndlr : ceux générés aléatoirement), alors le pool de chromosomes de seconde génération devient la référence et nous retournons à l'étape 2.

Ce processus est itératif tant que la meilleure solution n'a pas changé pour un nombre donné de génération. Une itération de cette cinématique correspond à une génération et l'ensemble des générations est souvent nommé « run » ou encore « batch » en anglais, une traduction française semble inadaptée ici.

Les résultats de ces itérations contiennent de bonnes solutions qui peuvent être des solutions optimales dans le cas des problèmes complexes comme les problèmes dits NP-Difficile. Les problèmes classés dans la catégorie des NP-Difficile sont ceux dont il n'est pas possible de trouver une solution optimale dans un temps polynomial. Une courbe de type polynomial est une courbe qui peut être tracée en utilisant des exponentielles et des variables. La résolution d'un problème de type NP-Difficile n'augmente pas de manière exponentielle avec le nombre de variable mais plus que ça, en exemple mathématique d'opérateur non polynomial et le factoriel  $n!$ . Un exemple connu de problème NP-Difficile est celui du voyageur de commerce qui doit relier une série de villes données de sorte à parcourir le moins de kilomètres et en ne passant qu'une seule et unique fois par ville. Les villes de départ et d'arrivée ne sont pas fixées. Cette tâche peut sembler facile pour un programme itératif classique car la vitesse à laquelle il analysera chaque solution ne pourra pas être atteinte par un algorithme génétique. Cependant si l'on considère l'augmentation du nombre de possibilités avec l'augmentation du nombre de villes il nous apparaît clairement qu'il devrait exister une meilleure façon de faire que de tester toutes les solutions possibles. Ainsi s'il n'y a qu'une ville ou deux, il n'existe qu'une seule solution à ce problème mais si le nombre de villes passe à 10 nous sommes



déjà à plus de 39 millions de possibilité. Le nombre exorbitant de  $3041 * 10^{64}$  est atteint au bout de 50 villes, ce qui rend les algorithmes itératifs classiques complètement inutilisables face à ce problème. Les manières de résoudre ce problème NP-Difficile grâce aux algorithmes génétique seront exposées dans le développement de cette thèse.

Vous pourrez trouver, à la page suivant, un schéma récapitulatif.

Etape 1 : Générer une population initiale de manière aléatoire.

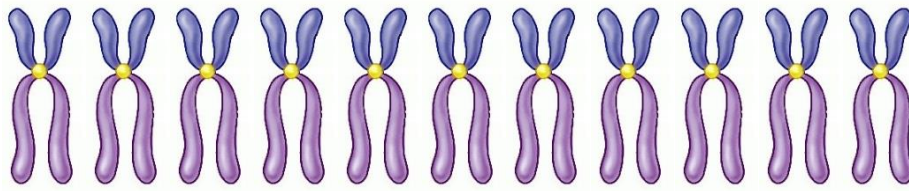
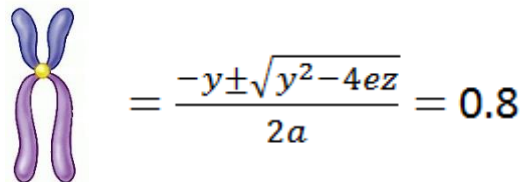


Figure 6 - Population initiale

Etape 2 : Déterminer le facteur d'adaptation à son environnement pour chaque neurone.



$$= \frac{-y \pm \sqrt{y^2 - 4ez}}{2a} = 0.8$$

Figure 7 - Equation d'adaptation à l'environnement

Etape 3 : Sélectionner une paire de chromosomes pour accouplement avec une plus forte probabilité pour ceux disposant d'un facteur fort.

Etape 4 : Recombiner le matériel d'une paire de neurones en les « accouplant » grâce au crossover

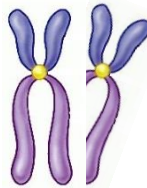


Figure 8 - Crossover

Etape 5 : Provoquer des mutations aléatoires sur chromosome créé à l'étape précédente

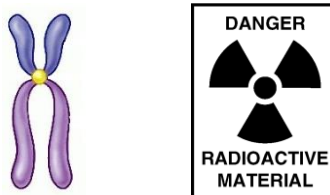
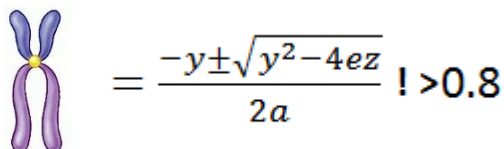


Figure 9 - Mutation aléatoire

Tant que



$$= \frac{-y \pm \sqrt{y^2 - 4ez}}{2a} ! > 0.8$$

, inclure ce chromosome à la poplation initiale puis recommencer à l'étape 3.

## Champs d'applications

Les algorithmes génétiques sont des algorithmes de recherche adaptatifs et peuvent de ce fait être utilisés dans de nombreux domaines tels que les sciences, les affaires, l'ingénierie ou encore la médecine. Les algorithmes génétiques sont conçus pour rechercher des solutions dans des espaces de recherche très larges, dans un espace dit non linéaire. Un espace de recherche qui appartient à la catégorie des espaces non linéaires contient un grand nombre de solutions potentielles et une solution optimale qui ne peut pas être trouvée dans un temps raisonnable via les techniques habituelles (itératives). Ainsi l'usage des algorithmes génétiques est préconisé quand l'espace de recherche est déraisonnablement grand, complexe ou encore non compréhensible par un humain. Ils peuvent aussi être adaptés lorsqu'il n'existe pas de méthode programmatique pour réduire l'espace de recherche ou quand les méthodes traditionnelles d'optimisation de la fouille sont inadéquates.

De nombreux problèmes concrets d'optimisation possèdent des espaces de recherche non linéaire comme celui du voyageur de commerce, le dilemme du prisonnier, ou encore l'hôte et le parasite. Les algorithmes génétiques possèdent la faculté de rechercher des solutions proches des solutions optimales dans des espaces de recherche complexes et démesurés dans un temps raisonnable en simulant l'évolution biologique. Les algorithmes génétiques ne garantissent pas de trouver la solution la plus optimisée pour le problème donné car il se peut que cette solution n'évolue pas pendant 10 à 15 générations mais que ce ne soit toujours pas la solution la plus optimisée. Ce problème, qui sera explicité lors du développement de cette thèse, est comparable à celui issu des mathématiques nommé les minima locaux malgré que la solution retournée par l'algorithme génétique puisse encore être améliorable, elle est néanmoins extrêmement proche de la solution optimale si ce n'est pas la solution optimale elle-même.

Les domaines de recherches dans lesquels les algorithmes génétiques sont le plus fréquemment utilisés sont la recherche d'une donnée stockée, la recherche de chemins, et la recherche de solutions. Afin d'illustrer ces domaines voici quelques exemples, imaginons que dans une base de données contenant plusieurs millions d'enregistrements qui recensent la population d'un pays par noms. Quel est le meilleur moyen de trouver la ligne qui correspond à un nom donné ? Knuth <sup>7</sup> en 1973 décrit un ensemble d'algorithmes génétiques qui font aussi bien voir mieux que la recherche binaire répandue sur la quasi-totalité des moteurs de base de données actuels. La recherche d'un chemin peut s'illustrer facilement par le problème du voyageur de commerce exposé plus avant mais aussi par un simple puzzle de trois cases sur trois ne contenant que huit pièces. Chaque pièce a été déplacée de sa position initiale, quel est l'ensemble de cout à jouer pour replacer les huit pièces dans leur position initiale ? On pourrait facilement résoudre ce problème avec la technique des arbres de recherche partiel, notamment grâce à l'algorithme A\*<sup>8</sup> mais si le nombre de pièces augmente de manière importante, aucun algorithme de recherche conventionnelle ne pourra nous venir en aide. Enfin la recherche de solution est un sur ensemble de la recherche de chemin, en effet certains problèmes ne nécessitent pas de tracer un chemin physique pour être résolu comme par exemple la prédiction de la structure d'une protéine à partir de la séquence d'acides aminés.

---

<sup>7</sup> Né en 1938. Est un informaticien américain de renom et professeur émérite en informatique à l'université Stanford

<sup>8</sup> est un algorithme de recherche de chemin dans un graphe entre un nœud initial et un nœud final tous deux donnés.

Des exemples de domaine plus palpables dans lesquels les algorithmes génétiques devraient ou sont utilisés sont répartis en trois familles : l'optimisation, l'architecture logicielle, et les applications financières. Ainsi nous sommes susceptibles de retrouver ce genre d'algorithme dans les planifications de production, le routage des appels téléphoniques et des véhicules, les plans électriques pour les optimisations. L'apprentissage machine pour designer des réseaux neuronaux ou encore le design et le contrôle de robot pour l'architecture logicielle. Les échanges boursiers, l'évaluation de crédit, l'allocation de budget ou encore la détection de fraude pour les applications financières.

## Le dilemme du prisonnier, un exemple de résolution classique.

Le dilemme du prisonnier fût inventé en 1950 par Melvin Dresher et Merill Flood<sup>9</sup> puis reformulé par Albert W Tucker<sup>10</sup> la même année.

Deux personnes sont arrêtées par la police, ils sont suspectés d'avoir commis un crime ensemble. Cependant les forces de l'ordre ne disposent pas de preuves irréfutables pour emprisonner les deux suspects. Les forces de l'ordre proposent alors la même offre au deux suspects qui n'ont aucun moyen de communiquer entre eux. « Si tu dénonces ton complice présumé et qu'il ne te dénonce pas tu es libre et lui ira en prison pour 10 ans. Si vous vous dénoncez mutuellement vous serez tous les deux emprisonnés pour 5 ans. Si personne ne parle alors la peine sera de 6 mois pour chacun ».

Ce problème peut aussi être exprimé sous la forme de la matrice suivante :

	Ne parle pas	Dénonce son complice
Ne parles pas	(6 mois, 6 mois)	(10 ans, libre)
Dénonce son complice	(libre, 10 ans)	(5 ans, 5 ans)

Ainsi les deux protagonistes mènent la réflexion suivante :

Si mon complice me dénonce alors :

- Si je ne le dénonce pas, j'irai 10 ans en prison.
- Si je le dénonce, j'irai 5 ans en prison.

Si mon complice ne me dénonce pas :

- Si je ne le dénonce pas, j'irai 6 mois en prison.
- Si je le dénonce, je suis libre.

Le choix le plus rationnel dans ce cas, où l'on ne peut pas communiquer avec son complice, est de le dénoncer. Cependant la réflexion menée par mon complice sera la même et il me dénoncera aussi, ce qui nous fera aller en prison 5 ans. Si les deux acteurs poursuivent leurs intérêts personnels alors le résultat total n'est pas optimal (au sens Pareto) car ils auraient pu aller seulement 6 mois en prison.

De nombreux programmes ont été confrontés les uns aux autres lors de tournois afin de déterminer la stratégie la plus efficace. Certains de ces programmes étaient composés de stratégies complexes issues de l'intelligence artificielle tels que les modèles de Markov cachés ou encore l'inférence bayésienne pour modéliser l'adversaire.

---

<sup>9</sup> Sont des mathématiciens américains (1911–1992) et (1908 – 1991) principalement connu pour l'énonciation du dilemme du prisonnier.

<sup>10</sup> (28 novembre 1905 - 25 janvier 1995) était un mathématicien américain d'origine canadienne qui a produit d'importantes contributions en topologie, théorie des jeux et optimisation non linéaire.



Malgré ce déploiement de technologies les plus complexes les unes que les autres, le gagnant de tous ces concours (calculé sur le nombre d'année en prison) possédait une stratégie des plus simple nommée la riposte (traduit de l'anglais tit-for-tat !). Cette stratégie consiste à ne pas parler lors du premier tour et dans tous les tours suivants il copie le comportement de coup d'avant de son adversaire.

Ainsi la stratégie tit-for-tat punira son adversaire jusqu'à ce que celui-ci se mettent à ne coopérer (jusqu'à ce qu'il ne parle pas). Axelrod <sup>11</sup> qui inventa cette stratégie décida en 1987 de voir si les algorithmes génétiques ne pourraient pas apporter une réponse plus « intelligente » au dilemme du prisonnier. Pour le dernier tour il n'y a donc que quatre possibilités :

Cas 1 : CC

Cas 2 : CD,

Cas 3 : DC,

Cas 4 : DD

Où C signifie coopèrent et donc ne parle pas et D signifie dénonce son complice. Si on suivait la stratégie tit-for-tat alors le tour numéro deux aurait été joué comme suit :

Cas 1 → Ne parle pas,

Cas 2 → Dénonce

Cas 3 → Coopère

Cas 4 → Dénonce

La Règle des tournois stipule que les programmes peuvent garder en mémoire les trois dernières parties ce qui nous donne 64 possibilités :

Possibilité 1 : CC CC CC

Possibilité 2 : CC CC CD

Possibilité n

Possibilité 64 : DD DD DD

Ces 64 possibilités peuvent être encodées sur une chaîne de caractère de 64 lettres. Comme chaque locus peut posséder deux allèles possibles (C ou D), le nombre de stratégies différentes est de  $2^{64}$ . Ceci représente un espace de recherche de grande taille où les algorithmes génétiques sont généralement à l'aise. Alexrod a réalisé 50 runs de 40 générations chacun, et les algorithmes ont plus ou moins évolués vers des stratégies similaires à tit for tat dans le sens où ils punissent la non coopération mais pas seulement en se basant sur le dernier tour. De ce fait les algorithmes génétiques ont obtenus des résultats sensiblement meilleurs que tit for tat.

---

<sup>11</sup> (Né en 1943) est un professeur de science politique à l'université du Michigan et est connu pour ces travaux sur la collaboration.

L'introduction des algorithmes génétiques ont donc permis de découvrir des stratégies ressemblant à tit for tat en partant de zéro, ce qui signifie que tit for tat est une stratégie quasiment optimale. Cependant les mécanismes de mutations aléatoires ont introduit de la volatilité dans le comportement de l'algorithme ce qui lui a permis de devenir plus intelligent.

## Introduction de la thèse professionnelle.

Élève de l'école eXia.Cesi Aix-en-Provence en échange à l'université du Québec à Montréal, je suis diplômé Analyste Programmeur (Niveau III) et Responsable en Ingénierie des logiciels (Niveau II) et je suis actuellement candidat au diplôme de Manager des systèmes d'informations (Niveau I) pour conclure ma formation à l'eXia.

J'ai souhaité agrémenter le cursus complet de l'eXia par des expériences prestigieuses à l'étranger, c'est donc pourquoi je me suis naturellement intéressé à l'échange et au double diplôme proposé à l'UQAM<sup>12</sup>.

Ma première année d'échange à l'UQAM a pris la forme d'un stage en recherche qui avait pour intitulé : « Métriques pour la détection automatique de défaut dans les applications à base de services » au sein du LATECE ([www.latece.uqam.ca](http://www.latece.uqam.ca)) qui est l'un des nombreux laboratoires de recherche de l'UQAM. Ce stage m'a donc entraîné dans la tâche complexe qu'est la détection de défaut de conception dans les applications à base de services divers et variés. Ce stage m'a permis de découvrir d'innombrables choses, tout d'abord le Canada et ses paysages et ses habitants fantastiques, une grande université nord-américaine et ses séances de cours qui sont très différentes de ce que j'ai connu à l'eXia.Cesi et ses recherches universitaires de pointes menées par des professeurs mondialement connus dans leurs domaines respectifs.

Je suis particulièrement enchanté et fier d'avoir pu apporter ma pierre à l'état de l'art de la détection automatique de défaut pour les applications à base de service, en effet nous avons écrit deux articles de recherche, dont je suis respectivement co-auteur ([http://mathieu-nayrolles.com/media/Moha-et-al\\_ICSOC2012.pdf](http://mathieu-nayrolles.com/media/Moha-et-al_ICSOC2012.pdf)), et auteur ([http://mathieu-nayrolles.com/media/Nayrolles-et-al\\_ICSOC2012ToolDemo.pdf](http://mathieu-nayrolles.com/media/Nayrolles-et-al_ICSOC2012ToolDemo.pdf)). Nous avons soumis ces deux articles pour publication à l'ICSOC 2012 (International Conference on Service Oriented Computing, <http://www.icsoc.org/>) et la communauté scientifique a réservé un accueil plus que chaleureux à notre approche et à nos travaux puisque les deux articles ont été acceptés pour publication et présentation avec un taux d'acceptation moyen de plus de 85%<sup>13</sup>. De plus, nous avons reçus l'award du second meilleur article lors de sa présentation à Shanghai le 14 Novembre 2012.

Depuis le 1 Janvier 2013 et pour une période d'un an je poursuis mes recherches dans le cadre de l'obtention de ma maîtrise en Informatique à l'UQAM. Néanmoins je vais explorer de nouvelles pistes basées sur certaines de mes nouvelles idées en accord avec le but recherché par l'équipe de ma directrice de recherche (ndlr : La détection et la correction automatique de défauts dans les applications à base de service), notamment liés à théorie de la fouille d'association, aux algorithmes génétiques et l'intelligence artificielle.

Pour conclure cette partie scolaire de ma présentation, mes nouvelles recherches à l'UQAM sont donc directement liées au contenu de cette thèse professionnelle puisqu'elles abordent de nombreux domaines de l'intelligence artificiel et notamment celui des algorithmes génétique.

---

<sup>12</sup> L'université du Québec à Montréal est une université publique de langue française qui dispose d'un rayonnement internationale.

<sup>13</sup> Pour décider parmi les centaines d'articles soumis lesquels seront présents lors de la conférence, ils sont soumis à lecture de quatre experts qui note les articles sur 100 en suivant une grille de notation.

Mes ambitions professionnelles sont d'exercer des fonctions assimilables à celle d'un directeur technique dans des milieux à haute criticités comme l'énergie, la finance ou l'armement. Ces milieux, notamment la finance pour la prédiction de valeurs ouvrent de plus en plus leurs portes à des courants de pensées issues de l'intelligence artificielle. Ainsi cette thèse est tout à fait en accord avec mon plan de formation individualisé car elle me permettra d'acquérir un avantage concurrentiel non négligeable sur les postulants aux postes à responsabilités dans les milieux à haute criticité.

## Résolution de problèmes

Comme toutes les autres méthodes de programmation les algorithmes génétiques ont été utilisés de deux manières bien distinctes. La première étant la simplification de modèle scientifique dans l'espoir (atteint !) de répondre à des questions à propos de la nature et la seconde étant de résoudre des problèmes technologiques complexes. La frontière entre les deux branches de recherches suscitées semble inébranlable et pourtant nous verrons qu'il n'est pas toujours aisé de déterminer de quel côté de cette frontière se trouve une application. En effet d'une part les réponses à des questions à propos de la nature peuvent être assimilées à la résolution d'un problème d'ingénierie. D'autre part la solution d'un problème d'ingénierie peut nous donner des réponses sur les théories évolutionnistes si l'on s'attarde sur les mécanismes, empruntés à la nature, que l'algorithme génétique a mis en œuvre pour solutionner le problème. La frontière entre les algorithmes génétiques utilisés pour répondre aux besoins de ces deux axes de recherche ne semble donc plus si inébranlable mais bel et bien floue. Ce flou peut être déroutant mais il est la preuve de l'adaptabilité des algorithmes génétiques à tout type de problème, ce qui les rend d'autant plus intéressants.

## Programmation automatique

La programmation automatique ou l'art de construire un programme qui est capable d'écrire des programmes suffisamment complexes et robuste pour être utilisés à des fins réelles est un des objectifs les plus anciens poursuivis par les chercheurs en Intelligence Artificielle. Les premiers essais ont été réalisés par Fogel<sup>14</sup> et *al.* en 1966 via l'évolution de programmes simples représentant des machines états. Le manque de résultats de ces chercheurs reconnus a failli enterrer cette application des algorithmes génétiques, cependant les travaux de John Koza<sup>15</sup> sur la programmation automatique à travers le langage LISP<sup>16</sup> ont relancé l'intérêt de la communauté scientifique. Pour beaucoup des personnes curieuses de nouvelles techniques de construction de programmes l'idée de construire des programmes qui sont capables, à partir de l'énoncé d'une problématique d'écrire le programme qui y répondra le mieux est plus qu'existante. La programmation automatique prend tout son intérêt dans les architectures massivement parallèles<sup>17</sup> qui regorgent de problématiques spécifiques qui sont souvent complexes à comprendre par l'humain.

---

<sup>14</sup> Dr Lawrence J. Fogel (1928-2007). Pionnier des théories de l'évolution naturelle dans l'informatique et de l'analyse du facteur humain.

<sup>15</sup> John Koza est un professeur de l'université de Stanford. Ces travaux les plus connus portent sur l'utilisation de méthodes de programmation génétique et l'optimisation de problèmes complexes.

<sup>16</sup> Le langage Lisp fut inventé par John McCarthy en 1958 alors qu'il était au Massachusetts Institute of Technology (MIT). Lisp est la plus ancienne famille de langages impératifs et fonctionnels.

<sup>17</sup> Les ordinateurs parallèles sont des machines qui comportent une architecture parallèle, constituée de plusieurs processeurs identiques concourant tous au traitement d'une seule application.

Le terme de massivement parallèle est couramment utilisé lorsque le système à architecture parallèle comporte de quelques dizaines de processeurs à plusieurs dizaines de milliers de processeurs, ou plus encore. Ces systèmes sont souvent scalables, c'est-à-dire que leur puissance est extensible, dans une certaine plage, à peu près proportionnellement au nombre de leurs processeurs.



La programmation génétique (GP – genetic programming) a, selon son créateur John Koza, la capacité de créer des programmes qui répondent aux critères de complexité et de robustesse des applications à usage réel. Cette méthode de programmation automatique peut facilement être exprimée sous la forme d'une structure en forme d'arbre, composée de nœuds représentant les opérations et de branches les liants entre elles. C'est sur cette structure de l'algorithme génétique qui va construire le programme va travailler.

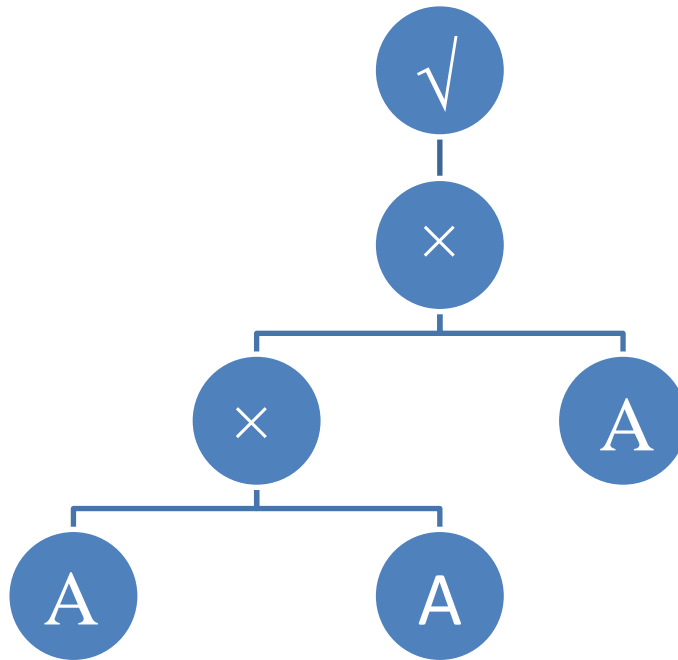


Figure 10 - Arbre d'opérations pour l'expression  $\sqrt{A^3}$

Un exemple de programmation automatique peut prendre place en un programme qui soit capable de calculer la période orbitale<sup>18</sup> d'une planète donnée en redécouvrant la troisième loi de Kepler<sup>19</sup> :

$$P^2 = cA^3$$

Figure 11 - Troisième loi de Kepler

Où  $P$  représente la période orbitale,  $A$  la distance moyenne entre la planète donnée et le soleil et enfin  $c$  une constante. L'algorithme de Koza (i.e. La programmation génétique) appliquée à ce problème peut être défini comme suit :

<sup>18</sup> En astronomie, la période orbitale désigne la durée mise par un astre (étoile, planète, astéroïde) pour effectuer une orbite complète. Par exemple, la Terre a une période orbitale de 365,25 jours.

<sup>19</sup> En astronomie, les lois de Kepler décrivent les propriétés principales du mouvement des planètes autour du Soleil. Elles ont été découvertes par Johannes Kepler à partir des observations et mesures de la position des planètes faites par Tycho Brahe, mesures qui étaient très précises pour l'époque.

1 – Choisir un ensemble de fonctions possible pour la résolution du problème donné. Le paradigme de la programmation génétique est de créer, via l'évolution, des programmes difficile à écrire et dont on ne connaît pas à l'avance quelles sont les fonctions qui vont être employées pour résoudre le problème.

Ainsi, les utilisateurs de ce paradigme doivent choisir intelligemment un ensemble de fonctions de taille raisonnable. Pour l'exemple de la troisième loi de Koza, l'ensemble de fonction parfait serait  $\{\sqrt{\phantom{x}}, \times\}$  mais un ensemble correct pourrait être  $\{\sqrt{\phantom{x}}, \times, \div, \pm\}$ .

2 – Générer aléatoirement un ensemble d'arbre d'opérations en utilisant les différentes fonctions possible définies dans l'étape 1.

3 – Calculer l'exactitude de chaque programme en appliquant une fonction spécifique, pour cet exemple, calculer le pourcentage d'erreur sur les valeurs de A et P trouvées par le programme et celle issue de mesures empiriques.

4 – Appliquer les principes de sélection, crossover, et mutation expliqués dans l'état de l'art. Koza a proportionnés ces principes d'une manière très précise. En effet, 10% de la population initiale (les arbres générés aléatoirement) sont copiés sans modification dans la population suivante (les 10% les plus précis sur les valeurs de A et P). Les 90% restants sont quant à eux soumis au crossover via deux parents eux aussi choisis en fonction de leurs précisions sur les valeurs A et P. Le crossover consiste à choisir aléatoirement un point de coupe sur une branche d'un arbre des deux parents et de former deux arbres enfants qui sont issus des deux possibilités.

Les étapes 3 et 4 sont répétées jusqu'à l'obtention d'un pourcentage d'erreur sur les valeurs A et P fixé par l'utilisateur ou encore par l'atteinte d'un nombre de génération maximum.

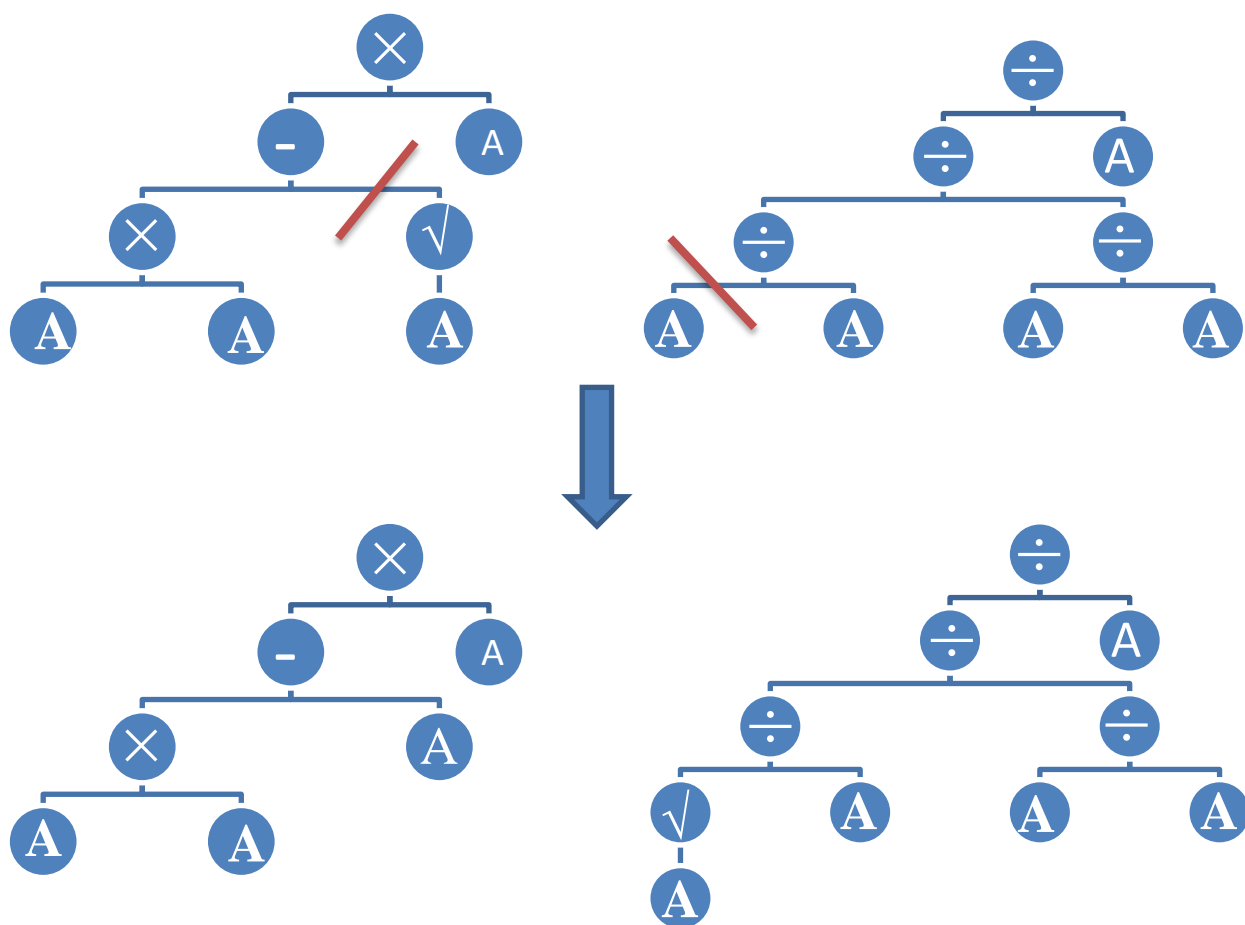


Figure 12 - Exemple de crossover dans le cadre de la programmation génétique

La figure 12 représente un crossover entre deux arbres parents qui ont été sélectionnés en fonction de leurs pourcentages d'exactitude ce qui a pour effet la création de deux descendants. Les parents sont représentés en haut et les enfants en bas. Les points de coupe sont indiqués par les traits rouge sur les arbres parents.

La programmation automatique et son représentant, la programmation génétique, peut ressembler à du pur hasard et elle a longtemps été comparé à l'exemple connu des singes tapant sur une machine à écrire et produisant l'œuvre de Shakespeare. Néanmoins il y a eu des essais concluants et complexes dans des domaines aussi variés que la planification, les séquences d'induction<sup>20</sup>, la compression d'image ou encore la robotique.

<sup>20</sup> Au sein d'un composé chimique, l'effet inductif consiste en la propagation d'une polarisation électronique au fil des liaisons chimiques, due à la différence d'électronégativité des différents éléments liés entre eux.

## Analyse et prédiction

Un des freins majeur à la progression scientifique dans de nombreux domaines de recherche est l'incapacité d'analyser et de donner un sens aux données, sans cesse plus importantes, générées par les expériences et autre simulations. Les domaines de recherches liés aux statistiques et à l'apprentissage machine se concentrent depuis de nombreuses années sur des méthodes pour trouver des patrons<sup>21</sup> qui soient suffisamment significatifs pour pouvoir prédire les futures données. Néanmoins les résultats apportés par ces domaines restent assez pauvre et la prédiction de données reste un sujet de recherche ouvert.

Une des domaines qui utilise avec le plus de résultats les algorithmes génétiques en vue de prédire des données est la biologie moléculaire. Les AG ont été utilisés avec succès pour analyser les données provenant de la résonance magnétique nucléaire<sup>22</sup>, réordonner des fragments d'ADN et la prédiction de structure de protéine. Dans ce dernier domaine, la prédiction des structures des protéines, il existe des exemples assez accessibles que nous allons aborder ensemble.

Les protéines sont la base du fonctionnement de toutes les cellules et donc de la vie. Le principal objectif de l'ADN est de coder les instructions qui vont permettent de créer des protéines et des acides aminés. Les protéines seront responsables de toutes les fonctions accomplies par la cellule, ces protéines sont composées d'acides aminés (de l'ordre d'une centaine) reliés par des peptides. A cause de l'électro statisme et de d'autre force physique comme la gravité les acides aminés « s'organisent » d'une manière très particulière et unique à chaque protéine. Cette organisation est tridimensionnelle et pour l'instant aucun chercheur en biologie ne peut déterminer quelle sera la structure tridimensionnelle d'une protéine à partir d'un ensemble de protéines, en effet cette mise en forme tridimensionnelle et unique des protéines restes un des plus grand mystères de la biologie.

Un algorithme capable de prédire l'organisation tridimensionnelle d'une protéine aiderais sans aucun doutes les chercheurs du domaine à comprendre les mécanismes à l'origine de cette structure, mais il serait désormais possible de construire des protéines dans le but qu'elles aient des fonctionnalités précises. Les biologistes pourraient donc créer des molécules dédiées à certaines fonctions comme par exemple la guérison de certaines maladies aujourd'hui encore incurable.

Depuis les années 1990 de nombreux chercheurs en biologies et en informatique ont unis leurs efforts dans l'espoir de pouvoir créer des algorithmes génétique capable de prédire la structure d'une protéine.

Les travaux de Steffen Schulze-Kremmer publiés en 1992 illustrent l'une des possibilités d'utilisation des algorithmes génétique dans la prédiction de structure de protéine. Son idée fût de prendre une séquence d'acide aminé d'une protéine et d'utiliser un algorithme génétique pour rechercher dans l'espace toutes les possibilités d'organisation jusqu'à en trouver une qui ressemble de près à celle d'origine.

---

<sup>21</sup> Dans ce contexte un patron est un modèle qui se répète dans les données de manière prévisible.

<sup>22</sup> Pratique qui consiste à déterminer la structure de l'ADN. Lucasius et Kateman l'ont expérimentée en 1989.

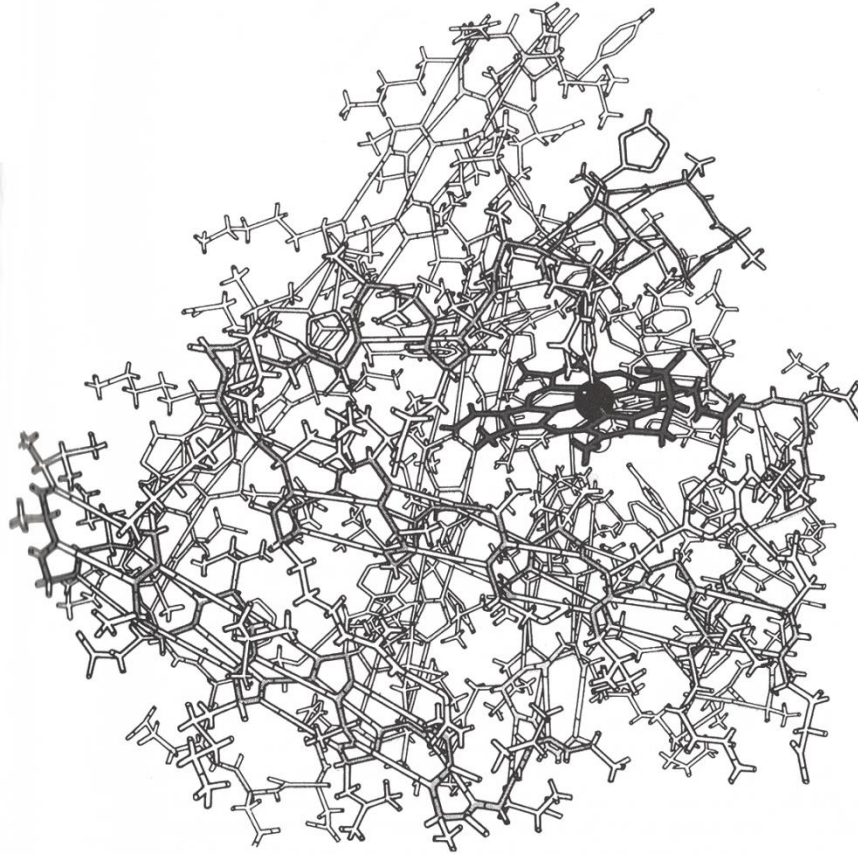


Figure 13 - Projection de la structure tridimensionnelle de la myoglobine<sup>23</sup>. (Gray & Haight, 1967, p. 419)

Pour décrire de manière rapide une structure tridimensionnelle, la manière la plus simple connue à ce jour et utilisée par Schulze est de créer une liste de tous les aminoacides associés à leurs coordonnées x, y, z. L'algorithme de Schulze se comporte comme suit :

- Convertir les coordonnées en vecteurs qu'il pourra faire évoluer pour créer une nouvelle structure.
- Evaluer laquelle des structures est la plus probable en mesurant le potentiel énergétique<sup>24</sup>.
- Créer des descendants via les mécanismes de crossover et de mutation.
- Recommencer les étapes 3 et 4 jusqu'à l'obtention d'une structure correcte ou l'atteinte d'un nombre maximum de génération.

L'algorithme génétique présenté a généré des structures possédant un potentiel énergétique très faible (i.e : et donc proche de la réalité) et même quelque fois plus faible que la protéine de base. L'algorithme génétique simplifie donc trop les structures, ce qui est sans doute dû à une mauvaise fonction d'évaluation. Cette fonction d'évaluation n'est sans doute pas assez restrictive... Néanmoins les expériences de Schulze sont un premier pas vers les prédictions de structures protéiniques qui aideront sans doute les biologistes dans un futur proche.

<sup>23</sup> Protéines de stockage / transport de l'oxygène dans les muscles apparentée à l'hémoglobine du sang.

<sup>24</sup> Il existe une hypothèse qui stipule qu'une séquence aminoacide s'assemblera de manière à réduire au maximum les conflits entre les différentes forces physique en présence. Ici plus le potentiel énergétique est faible, plus la structure est proche de la réalité.



## Ajustement de réseaux neuronaux

Les ordinateurs peuvent accomplir un nombre de tâches toujours de plus en plus grand et de plus en plus vite. Pour un certain nombre de ces tâches la vitesse d'exécution de l'ordinateur sera sans commune mesure par rapport à celle de l'humain, par exemple les opérations arithmétiques. Néanmoins il existe des problèmes pour lesquelles l'humain bat aisément les ordinateurs d'aujourd'hui. Prenons par exemple le problème souvent soumis aux enfants de maternelle, à savoir faire la différence entre une photo de chat et une photo de chien, il ne faudra pas longtemps aux enfants avant d'acquérir la compétence pour effectuer cette différenciation et ce même si l'on remplace le chat et le chien par d'autres. Pourtant ce problème reste insoluble pour un programme conventionnel.

Pour construire des programmes capables de résoudre des problèmes comme les humains, les chercheurs ont utilisés le seul modèle connu à leurs dispositions : le cerveau humain. Bien entendu le cerveau humain, même si l'on en connaissait toutes les spécificités, serait bien trop complexe à modéliser, c'est pourquoi les chercheurs ont voulu modéliser les neurones, qui sont les plus petites cellules connus composant le cerveau humain.

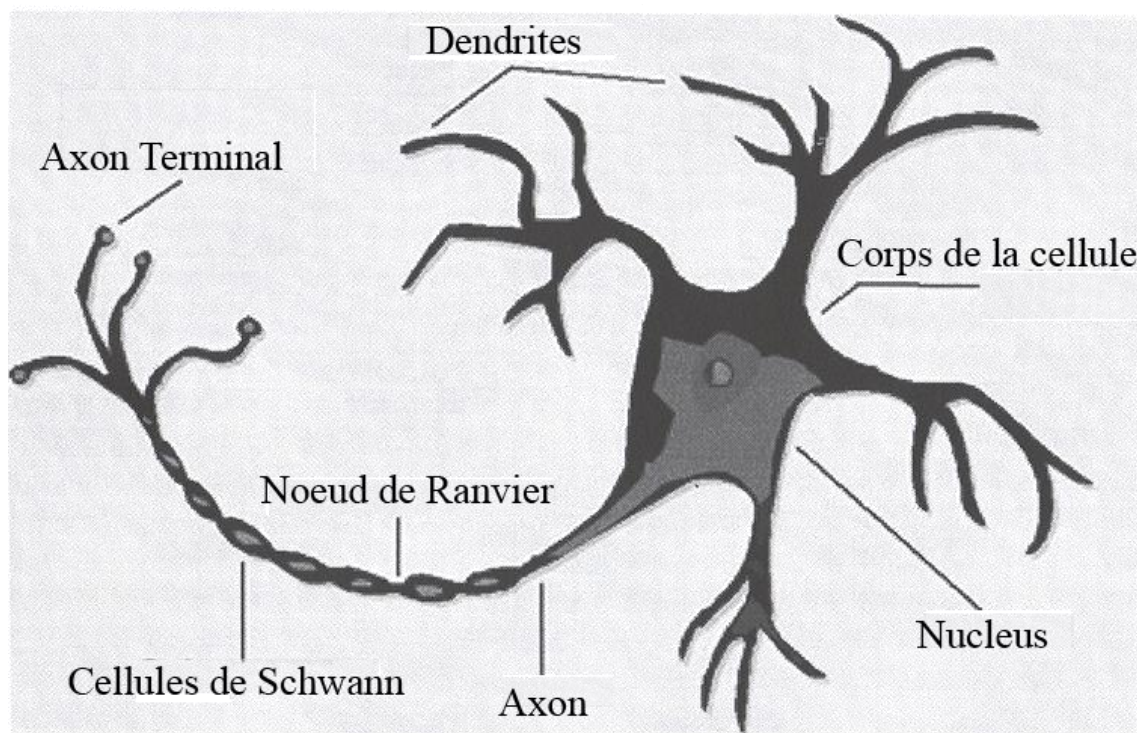


Figure 14 – Représentation schématique d'un neurone. Modifié et réimprimé depuis (Heaton, p. 40)

Un neurone, comme celui exposé à la figure 14, reçoit des signaux depuis ses dendrites. Lorsqu'un signal est reçu, le neurone peut le retransmettre en fonction de la force du signal reçu. Quand le signal est retransmis il passe par les axons et il est transmis aux autres neurones ou nerfs par le terminal axone. Ce terminal est composé de connections synaptiques.

Un réseau de neurones artificiels<sup>25</sup> est un ensemble d'unités<sup>26</sup> activable et connectés. Les connexions entre les neurones disposent d'un poids. Si un signal atteint un neurone et que ce signal est supérieur en intensité au poids d'une connexion entre ce neurone et un autre neurone connecté alors le signal est retransmis au neurone connecté. Le signal ne peut se propager que dans un seul sens et les résultats sont fournis par la dernière couche de neurones. En fonction de l'erreur commise par la dernière couche de neurone, les poids sont modifiés.

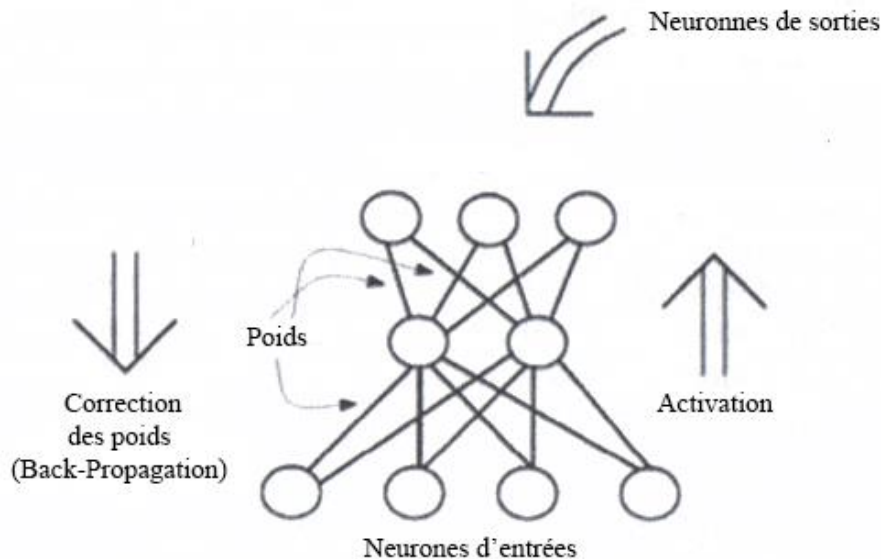


Figure 15 - Schéma d'un réseau dit feedforward<sup>27</sup> à backpropagation<sup>28</sup>. Traduit de (Whitley, p. 50)

Il existe plusieurs axes sur lesquelles les algorithmes génétiques peuvent influencer, comme l'architecture (i.e. le nombre de neurones) ou encore le poids entre les connexions. Ici j'ai choisi de vous présenter une utilisation des algorithmes génétiques qui vise à mettre à jour ces poids pour plus de précision qui remplace le mécanisme de correction habituel (i.e. la backpropagation). L'algorithme génétique créé par David Montana et Lawrence David en 1989 procède comme suit :

- Fixer aléatoirement les poids entre 0 et 1
- Créer des chromosomes qui contiennent 126 gènes et où chaque gène est une des possibilités.
- Evaluer la solution. Dans ce cas la performance du réseau de neurones est évaluée via le pourcentage d'exactitude de la couche de sortie.
- Sélectionner les réseaux les plus performants pour y appliquer les opérateurs de mutations et crossover et créer des descendants.

<sup>25</sup> Une explication sommaire de ce qu'est un réseau de neurones artificiel et de son fonctionnement est donnée ici, néanmoins le lecteur pourra trouver de plus amples détails dans mon approfondissement technique (Nayrolles, 2012) disponible ici : <http://mathieu-nayrolles/media/app-tech-reseaux-neuronnaires.pdf>

<sup>26</sup> De neurones artificiels.

<sup>27</sup> De façon plus générale, un réseau de neurone permet l'approximation d'une fonction. Ici, il s'agit d'une fonction de classification : à chaque n-uplet d'informations en entrée la fonction associe une classe.

<sup>28</sup> La technique de rétropropagation du gradient (*Backpropagation* en anglais) est une méthode qui permet de calculer le gradient de l'erreur pour chaque neurone d'un réseau de neurones, de la dernière couche vers la première.

- Répéter les étapes 3 et 4 jusqu'à l'obtention d'un taux d'erreur acceptable ou l'atteinte d'un nombre de génération maximum.

La performance de leur algorithme génétique a été comparée à celle de la backpropagation, pour obtenir les mêmes résultats il aura fallu créer 200 générations via l'algorithme génétique et 10000 itérations de l'algorithme de backpropagation. Ils ont aussi déterminés que la création d'une génération était équivalente à 12 itérations de l'algorithme backpropagation. Ainsi l'algorithme génétique arrivera aux mêmes résultats en 4 fois moins de temps<sup>29</sup>.

## Problème du voyageur de commerce : Un modèle d'efficacité pour les algorithmes génétiques

### Enoncé

Le problème du voyageur de commerce est un problème complexe (NP-Difficile<sup>30</sup>) qui est énoncé comme suit :

*Etant donné un ensemble de villes reliées par des routes dont les distances sont connues ; quel est le chemin le plus court pour parcourir l'ensemble des villes et retourner à la ville de départ ?*

La ville de départ peut être choisie sans aucune contrainte.

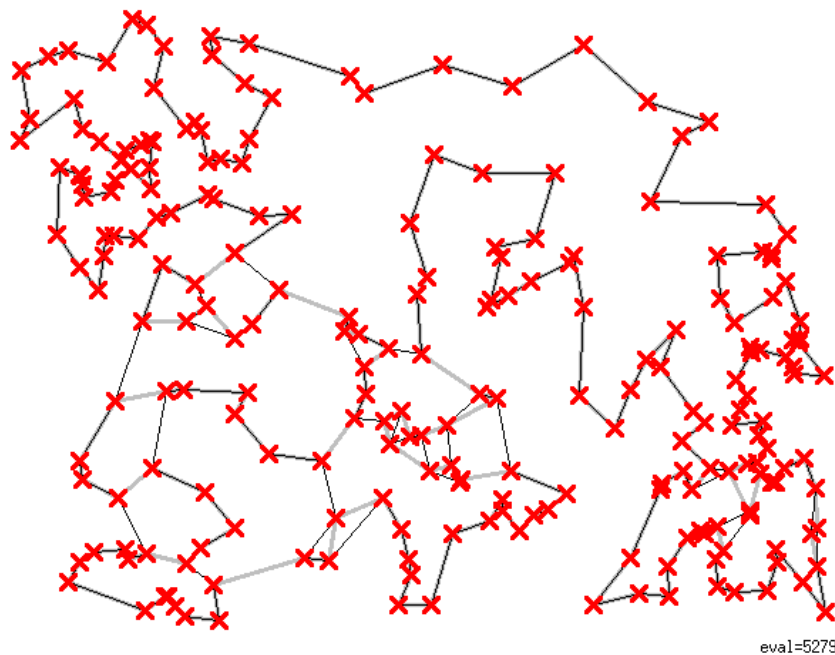


Figure 16 - Problème du voyageur de commerce résolu (src <http://morihi.net/>)

<sup>29</sup> 200 générations x 12 = 2400 et 10000 / 2400 = 4.1

<sup>30</sup> Dont la solution ne peut pas être trouvée dans un temps polynomial.

## Historique

Le problème du voyageur de commerce a été énoncé par des mathématiciens Irlandais et Anglais, respectivement nommé W. R Hamilton et Tomas Kirkman en 1832. Ce problème a ensuite été formellement introduit à la communauté scientifique en 1930 à Harvard par Karl Menger qui proposa des algorithmes visant sa résolution.

La connaissance scientifique s'étoffant chaque année, les chercheurs ont découverts que ce problème étant en fait la base de problèmes plus complexes tels que le séquençage d'ADN. Il est donc primordial de disposer de techniques rapides et fiables pour résoudre le problème du voyageur de commerce.

## L'existant

Cette sous-section présente les algorithmes de résolution du problème du voyageur de commerce les plus répandus.

### *Dijkstra*

L'algorithme de Dijkstra, du nom de son inventeur Néerlandais Edsger Dijkstra a été dévoilé en 1959 comme une alternative beaucoup plus complexe à l'approche proposée par Floyd (non décrite ici) mais aussi beaucoup plus rapide.

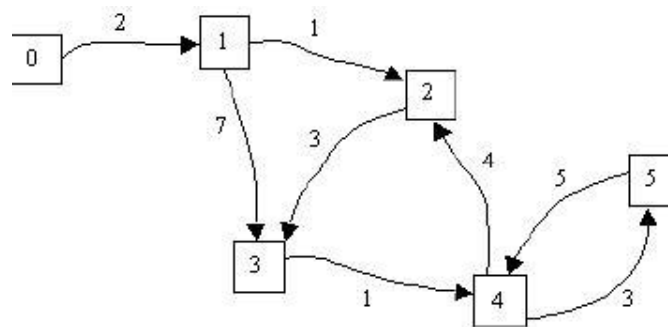
Cet algorithme fait partie de la famille des « gloutons » ; c'est-à-dire qu'à chaque étape ; il traite un nouveau sommet—une nouvelle ville. L'initialisation de l'algorithme consiste à parcourir tout le graphe ; de façon non-ordonnée dans le but de construire un tableau de tous les nœuds—villes—ainsi que de toutes les arrêtes—routes. Une fois l'étape d'initialisation terminée, la façon dont il traite les sommets est toujours la même ; il analyse les routes qui connues partant des sommets déjà visités menant au sommet en cours. S'il existe une route totale plus courte et déjà explorée alors la ville courante est retirée des sommets restant à visiter. Dans le cas contraire, la ville courante et la dernière route empruntée pour s'y rendre sont ajoutées à la solution.

```

Pour i=1 à n Faire
s=Extraire_Minimum(C,inf)
Pour tous les sommets t reliés à s //
    Si (il le faut)
        Mettre_a_jour(distances,parcours)
        Percolation(t)
    Fin Si
Suivant i

```

**Algorithme 1 - PseudoCode Dijkstra** (src :<http://www.nimbustier.net/>)



**Figure 17 - Problème du voyageur de commerce**

```

Source : 0
Parcours a l'envers de 0 a 1 :
1 0
Distance : 2
Parcours a l'envers de 0 a 2 :
2 1 0
Distance : 3
Parcours a l'envers de 0 a 3 :
3 2 1 0
Distance : 6
Parcours a l'envers de 0 a 4 :
4 3 2 1 0
Distance : 7
Parcours a l'envers de 0 a 5 :
5 4 3 2 1 0
Distance : 10

```

**Algorithme 2 - Résolution suivant l'algorithme de Dijkstra**

## **BellMan-Ford**

L'algorithme éponyme BellMan-Ford introduit en 1958 est un algorithme très similaire à Dijkstra. En effet, il dispose des mêmes mécanismes d'initialisation et d'ajout/suppression de partie de solution lorsque qu'une route plus courte vers un nœud est trouvée. La seule différence notable est que Dijkstra vise le chemin le plus prometteur pour changer de nœud (le plus court) et fait marche arrière si finalement il s'est trompé ; alors que le BellMan-Ford test toutes les routes. Ces deux algorithmes ont des performances similaires sur des graphes dont la répartition est aléatoire. Néanmoins, sur des graphes représentant un vrai système autoroutier ; Dijkstra se classera toujours devant.

## A\*

Cet algorithme proposée en 1968 par Peter Hart, Nils Nilsson et Bertram Raphael et nommé A\* (prononcer A étoile) est une évolution de l'algorithme de Dijkstra. Il est largement utilisé par la communauté scientifique en vertu de ses performances accrue liées à l'utilisation d'heuristique. Une heuristique est une estimation du chemin restant à parcourir. Ainsi chaque branche de notre problème dispose maintenant d'une valeur réelle et d'une valeur donnée par l'heuristique qui estime le chemin restant à parcourir jusqu'au but si l'on emprunte cette route<sup>31</sup>.

## Algorithme génétique

Les trois algorithmes précédemment exposés ont tous été introduits avant 1970 et ils sont les plus utilisés pour résoudre le problème du voyageur de commerce. En résumé nous sommes en présence d'un problème vieux de 180 ans résolu par des algorithmes vieux de 45 ans au minimum. De plus, ces algorithmes ne sont pas parfaits puisqu'ils requièrent une puissance computationnelle exponentielle en fonction du nombre de ville. Le tableau ci-dessous recense le nombre d'étapes nécessaires à la résolution du problème par ces algorithmes en fonction de nombre de ville :

Nombre de villes	Nombre d'étapes requises
1	1
2	1
3	6
4	24
5	120
6	720
7	5 040
8	40 320
9	362 880
10	3 628 800
11	39 916 800
12	479 001 600
13	6 227 020 800
..	..
50	$3041 * 10^{64}$

Tableau 1 - Nombre d'étapes par ville

L'impraticabilité du problème est criante ici ; il nous faudra effectuer plus de 6 milliards de comparaison pour un problème comprenant seulement 13 villes... Certes, une comparaison de poids entre un chemin et un autre est une opération que les processeurs peuvent effectuer dans un laps de temps très court<sup>32</sup> ; cependant la multiplication de ces opérations rend le problème insolvable dans un temps acceptable. En effet, en utilisant Dijkstra sur un problème de 50 villes ; le résultat ne serait connu qu'après plusieurs jours (semaines ?) de calcul dépendant de la machine utilisée.

<sup>31</sup> Le calcul et la génération des heuristiques ne seront pas abordé ici. En effet, même pour des problèmes bien connus comme celui du voyageur de commerce ; la génération des heuristiques reste un domaine de recherche très actif. Par conséquent, c'est un sujet à part entière qui mérite sa propre thèse professionnelle.

<sup>32</sup> Un cycle d'horloge peut suffire.

L'implémentation de l'algorithme génétique mis en concurrence avec les algorithmes couramment utilisés peut être définie comme suit<sup>33</sup> :

- Nombre de chromosomes : 1000
- Pourcentage de mutation aléatoire : 0.2%
- Pourcentage de crossover : 0.2%
- Population éligible pour la prochaine génération : 0.5%
- Arrêt du calcul après : 50 générations stables.

Le tableau ci-dessous recense les résultats obtenus par l'algorithme génétique et ceux des algorithmes classiques :

<i>Nombre de villes</i>	<i>Nombre d'étapes requises</i>
<b>1</b>	<b>53 (1)</b>
<b>2</b>	<b>53 (1)</b>
<b>3</b>	<b>53 (6)</b>
<b>4</b>	<b>53 (24)</b>
<b>5</b>	<b>53 (120)</b>
<b>6</b>	<b>53 (720)</b>
<b>7</b>	<b>53 (5 040)</b>
<b>8</b>	<b>53 (40 320)</b>
<b>9</b>	<b>62 (362 880)</b>
<b>10</b>	<b>65 (3 628 800)</b>
<b>11</b>	<b>64 (39 916 800)</b>
<b>12</b>	<b>72 (479 001 600)</b>
<b>13</b>	<b>111 (6 227 020 800)</b>
..	..
<b>50</b>	<b>304 (3041 * 10<sup>64</sup>)</b>

Tableau 2 - Nombre d'étapes requises par ville / GA

Comme nous pouvons le voir ; les problèmes triviaux (1, 2, 3 et 4 villes) nécessitent plus d'étapes pour être résolus. Ceci est dû à la nécessité d'obtenir 50 générations stables afin de retourner le résultat. En réalité, il n'a fallu que 3 étapes à l'algorithme pour trouver le résultat ; mais celui-ci continu de faire muter le pool de chromosomes afin d'identifier toutes autres solutions. Nous pouvons aussi constater que l'augmentation du nombre d'étapes nécessaires à la résolution est beaucoup moins exponentielle que pour les autres algorithmes. En effet, 304 étapes auront suffi à résoudre un problème à 50 villes ; qui était auparavant ; complètement impraticable. Bien entendu, une étape d'un algorithme génétique est beaucoup plus complexe qu'une simple comparaison puisqu'elle est composée de tri, sélection, mutation et enfin comparaison. Cependant, à 13 villes ; il faudrait qu'une étape génétique soit 56 millions de fois plus longue à exécuter. Ce n'est évidemment pas le cas, car la complexité d'une étape génétique est *seulement* 12 fois supérieure à celle d'une comparaison classique.

<sup>33</sup> Pour rappel; les données mentionnées ici ont été explicitées dans la section « Les algorithmes génétiques » de l'état de l'art.



## Conclusion

La résolution de problème inspiré par la nature est une activité cruciale si nous voulons pouvoir résoudre des problèmes de plus en plus complexes dans un temps computationnel acceptable. Dans cette thèse professionnelle, nous avons présentés des techniques modernes et innovantes pour la résolution de problème inspiré par la nature. Ces techniques sont issues—plus ou moins directement—de la théorie de l'évolution de Charles Darwin ; « La survie du plus fort<sup>34</sup> ». Ces techniques utilisent des réseaux de neurones artificielle que l'on peut entraîner à une tâche spécifique ; de la même manière que l'on entraîne son cerveau aux mathématiques ou des pools de chromosomes que l'ont fait évoluer—muter—afin de voir lequel est le plus adapté à son environnement—le problème. Les résultats exposés dans cette thèse professionnelle ; sont plus qu'encourageant quant à la résolution du problème du voyageur de commerce qui est une base à des problèmes beaucoup plus complexe tel que le séquençage d'ADN.

Cette thèse professionnelle a aussi permis de créer des algorithmes génétiques dédié à la découverte de chemins optimaux—compromis entre qualité (présence de défauts de conception) et temps d'exécution—dans les applications à base de services dans le cadre de mes recherches au sein de l'UQAM. Les résultats préliminaires de l'approche supportée par ces nouveaux algorithmes ont démontrés qu'il est possible de réduire le temps d'exécution de 35%<sup>35</sup> d'une application à base de service tout en réduisant le nombre de défaut de conception présent.

---

<sup>34</sup> Cette traduction officielle est une traduction approximative de « Survival of the fittest ». Une traduction plus précise serait : « La survie du plus adapté ». Cette traduction permet d'entrevoir plus facilement le lien entre Charles Darwin et les algorithmes présentés.

<sup>35</sup> Sur un système jouet nommé HomeAutomation et développé par l'INRIA Lille.

## Bibliographie

Gray, & Haight. (1967). *Basic Principles of Chemistry*. W.A Benjamin, Inc. New York.

Heaton, J. (n.d.). *Introduction to Neural Network*.

Holland. (n.d.). *Adaptation in Natural and Artificial Systems*.

<http://www.ai-junkie.com/ga/intro/gat1.html>. (n.d.).

Stuart, R., & Peter, N. (n.d.). *Intelligence artificielle*.

Tort, P. (n.d.). *Darwin, théorie de l'évolution*. Futura Sciences.

Whitley, D. (n.d.). *A Genetic Algorithm Tutorial*.