Algorithmes génétiques et application au problème du voyageur de commerce

Fabien DUBOIS,
Antoine RATO,
Corentin HEMBISE
https://github.com/dut-info/Algo-Genetique

26 mai 2016

Table des matières

1 Principe des algorithmes génétiques

1.1 Introduction

Les algorithmes génétiques sont des algorithmes qui se basent sur la sélection naturelle afin de trouver des solutions à un problème d'optimisation, celle - ci est un processus de l'évolution des espèces.

Les individus les plus adaptés à leur environnement, sont plus susceptible de « survivre », alors au fil de nouvelles générations, la population devient meilleur au regard de son environnement.

En clair, les algorithmes génétiques miment le processus de sélection naturelle pour trouver des solutions efficaces à un problème à un ou plusieurs paramètres. Ce sont des algorithmes d'approximation puisqu'il ne permettent pas de trouver la solution optimale à un problème, mais de s'en rapprocher. En outre, ils ont l'avantage de trouver une solution en un temps bien inférieur aux algorithmes déterministes.

Les termes utilisés dans ce genre de problème sont empruntés de la théorie de l'évolution, ainsi, on parlera :

d'individu : c'est une solution admissible du problème

de population : c'est un ensemble d'individus

de gène : c'est une partie d'un individu, une partie du problème

de génération : c'est une itération de l'algorithme

de fonction objectif : c'est une fonction qui permet de définir qu'un individu est meilleur qu'un autre

Pourquoi utiliser des algorithmes génétiques ?
Dans les problèmes d'optimisation où le nombre de variables est grand, une solution optimale est coûteuse et n'est pas toujours nécessaire, néamoins, le temps de calcul doit être optimisé. Dans cette optique, les algorithmes génétiques sont bien adaptés, puisqu'ils permettent de traiter d'un problème à plusieurs variables efficacement. De plus, ce genre d'algorithmes est particulièrement adaptable à différents problèmes.

1.2 Principe

Les algorithmes de génétique se décomposent en 5 phases :

1. La création d'une population initiale
2. L'évaluation des individus
3. La sélection des meilleurs individus
4. Les croisement et les mutations des enfants
5. La création d'une nouvelle population

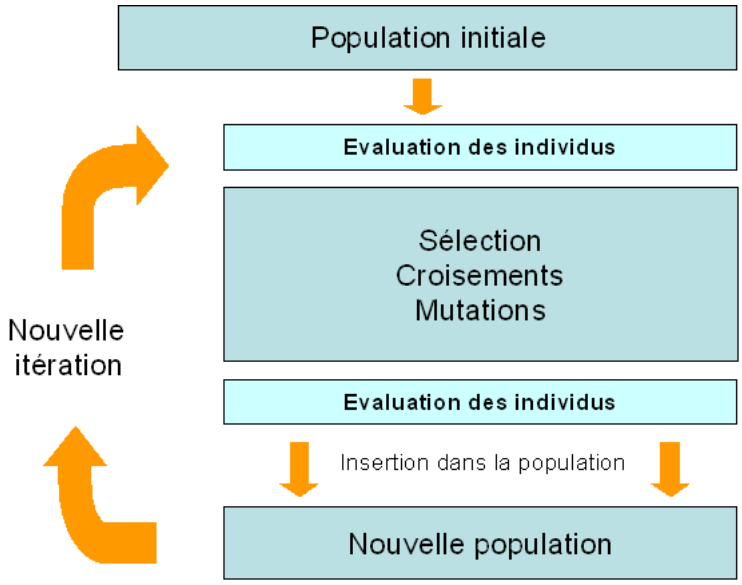


FIGURE 1 – Schéma général d'un algorithme génétique

Ensuite, on réitère à partir de la phase 2 jusqu'à la condition d'arrêt, elle peut être un nombre d'itération ou le moment ou le meilleur individu n'évolue plus.

1.2.1 Population initiale

La population initiale doit être créer aléatoirement, et chaque individu doit être une solution admissible du problème. Créer aléatoirement la population permet de s'assurer de la diversié de la population. La diversité est un élement important dans ce genre d'algorithmes puisqu'elle permet de ne pas converger vers une solution unique et d'explorer le maximum des solutions du problèmes.

1.2.2 Evaluation des individus

Chaque individu doit être évaluable, sur une même échelle, on va associer une note à chacun d'entre eux pour pouvoir les comparer. La méthode d'évaluation dépend du problème. L'évaluation est une phase importante lorsque l'on applique un algorithme génétique à un problème multi-critère, puisqu'il faut syntétiser l'ensemble des critères pour ne retourner qu'une "note".

1.2.3 Sélection des individus

La phase de sélection permet de choisir les individus qui vont survivre à la nouvelle génération et qui seront susceptible de se "reproduire". La méthode

de sélection doit garder une certaine diversité dans la population, en effet, un individu mauvais $au regard de sa fonction objectif$, peut donner un enfant très bon après croisement.
Plusieurs méthodes existent :

La sélection élitique
Ce type de sélection trie les individus d'une géneration par ordre de fitness et choisit les $n/2$ individus les meilleurs. C'est la méthode de sélection la plus simple à implementer. Le problème de cette méthode réside dans le fait qu'elle converge rapidement vers une unique solution, au fil des sélections, la population devient de moins en moins diversifiée.

La sélection par roulette
Dans ce type de selection, un individus avec un meilleur fitness qu'un autre à plus de chance d'être choisit. Tous les individus sont placés sur une roulette, plus l'individus à un bon fitness, plus il occupera une grande partie de la roulette. On tire ensuite un emplacement de la roulette au hasard et l'individu correspondant à cet emplacement est sélectionné pour la génération suivante. On réitère le processus jusqu'à obtenir la taille de la nouvelle population souhaitée.

La sélection par rang
Ce type de sélection est similaire à la précedente, mais dans ce cas, la probabilité d'être selectionné dépend du rang de l'individus dans la population $pralablement triepar fitness$. La sélection par roulette présente un inconvénient, si un ou des individus dominent tous les autres, la selection n'assure plus la diversité de la population, la sélection par rang permet de résoudre se problème.

La sélection par tournoi
On forme des groupes d'individus aléatoires au seins de la poulation et on selectionne dans chaque groupe, l'individu ayant le meilleur fitness.

1.2.4 Croisement des individus

La phase de croisement associe deux individus pour créer un ou plusieurs individus enfant, qui partageront les gènes des deux parents. Généralement, on choisit aléatoirement un point de césure c dans la chaine génétique des parents et on copie les c premiers gène du premier parent dans le premier enfant et les gène de c jusque la fin du second parent vers le premier enfant, pour le second enfant, les rôles des deux parents sont inversés.

1.2.5 Mutation des individus

La mutation consiste à changer aléaoirement un gène d'un individu par un autre gène. Cette phase permet d'éviter la convergence des solutions. Elle s'effectue sur les enfants nouvellements créer et dois avoir une propabilité faible, dans le cas contraire, la recherche deviendrait aléatoire.

2 Application au porblème du voyageur de commerce

2.1 Définition du problème

Le problème est le suivant :

> Un voyageur de commerce doit visiter n villes données en passant par chaque ville exactement une fois. Il commence par une ville quelconque et termine en retournant à la ville de départ.
> Quel chemin faut-il choisir afin de minimiser la distance parcourue ?

2.2 Modélisation du problème

Soit :
— n le nombre de villes
— $\{v_i\}^n$ l'ensemble des villes
— c un chemin, c'est à dire une permutation des n villes
— $\{D_{d,a}\}^{n,n}$ la distance de la ville d à a

La fonction ocjectif pour un chemin c est : $f(c) = \sum_{i=1}^{n} D_{c_i c_{(i+1) \bmod n}}$

Résoudre ce problème en trouvant une solution optimale reviendrais à parcourir l'ensemble des solutions du problème. C'est à dire l'ensemble des chemins, donc toutes les permuations de c, soit $n!$ permutations.
Par exemple, pour 10, 30 et 100 villes et si l'on suppose qu'une évaluation d'une permutation coûte 1 μs :

n	Nombre de permutations	Temps
10	3 628 800	3,6 s
30	26×10^{31}	84×10^{15} siècles
100	93×10^{156}	3×10^{142} siècles

Même si pour dix villes une solution optimale reste envisagable, à partir de 30 villes, un algorithme de ce genre serait beaucoup trop long. Dans ce cas, l'utilisation d'un algorithme génétique est approprié.

2.3 Implementation

Dans le cas du problème du voyageur de commerce, un individu représente un chemin, une ville représente un gène qui est modelisé par un entier et une position en x et y.

Création de la population initiale : les chemins de départ sont créés aléatoirement, ce qui permet de diversifier au maximum la population initial, et de ne pas se limiter à quelques possibilités.

Evaluation des chemins : la fonction objectif calcule pour chaque chemin, la distance totale à parcourir. L'objectif de l'alogorithme est de minimiser cette

fonction. Dans notre implémentation, la distance entre deux villes v_1 et v_2 est donnée par la formule $D_{v_1,v_2} = \sqrt{(v_{1_x} - v_{2_x})^2 + (v_{1_y} - v_{2_y})^2}$

Selection des chemins : nous avons implémenté plusieurs méthodes de selection : selection élitique, selection par roulette, selection par tournoi.
Ce qui nous permettera de comparer les différentes méthodes de selection.

Croisement des chemins selectionnés : le croisement de deux chemins parents créés, en fonction du taux de croisement, un ou plusieur individu. Pour un taux de croisement de 1, deux chemins créent 2 enfants, pour 1.5, deux chemins créent en moyenne 1.5 enfants. Ce paramètre nous permettera de faire évoluer la taille de la population au fil des générations. Le croisement de deux chemins de taille m s'effectue comme suit :

- On tire un nombre aléatoire *rand* entre 1 et m
- On copie les villes de 1 à *rand* du premier parent vers le premier enfant
- On copie les villes du second parent qui ne sont pas dans le premier enfant
- On effectue la même opération pour le second enfant en inversant le rôle des parents

Cette méthode de croisement nous assure que chaque enfant créé soit un enfant "correct", c'est à dire qu'il soit une permutation des n villes.

Mutation d'un chemin : la mutation intervient avec une probabilité de $x \in]0, 1[$. Le procesus de mutation consite à inverser au hasard deux villes d'un chemin.

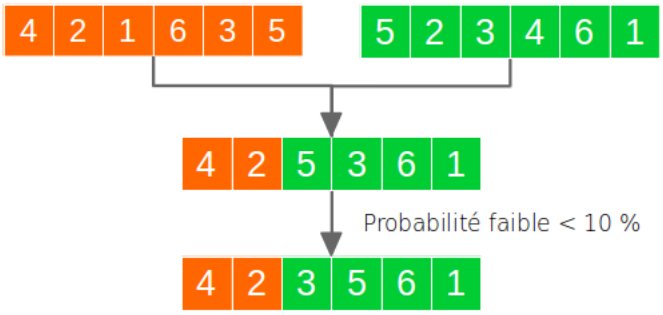


FIGURE 2 – Schéma d'un croisement et d'une mutation

2.4 Implementation technique

Nous avons implémenté l'algorithme en java, ce qui nous permet d'utiliser le polymorphisme pour par exemple, définir plusieur méthodes de sélection tout en conservant la même logique dans le reste du code. C'est également un langage facilement portable et qui nous est familier.

Afin d'avoir une visualisation de l'avancée de la recherche et un résultat final parlant, nous avons représenté les villes par un graphe, ce qui nous permet de faire évoluer ce graphe au cours des génerations. Les villes peuvent être generée aléatoirement ou en cercle. La distance minimale pour un cercle est son périmètre, le chemin minimal est donc celui qui passe par le périmètre, ce qui nous est utile lors des phases de tests.
Pour executer le programme sous Linux, OSx et Windows, il suffit d'executer la commande :

```
java -jar Algo-Genetique.jar
```

3 Résultats

3.1 Paramètrage de l'algorithme

Liste des paramètres sur lesquels influer :
— Nombre d'individus initiaux $]1, 1000[$
— Nombre d'itérations max $]1, 5000[$
— Nombre de villes $]10, 50[$
— Méthode de selection (Elistisme, roulette, tournoi)
— Taux de mutations $]0, 1[$
— Taux de selection $]0, 1[$

3.2 Résultats

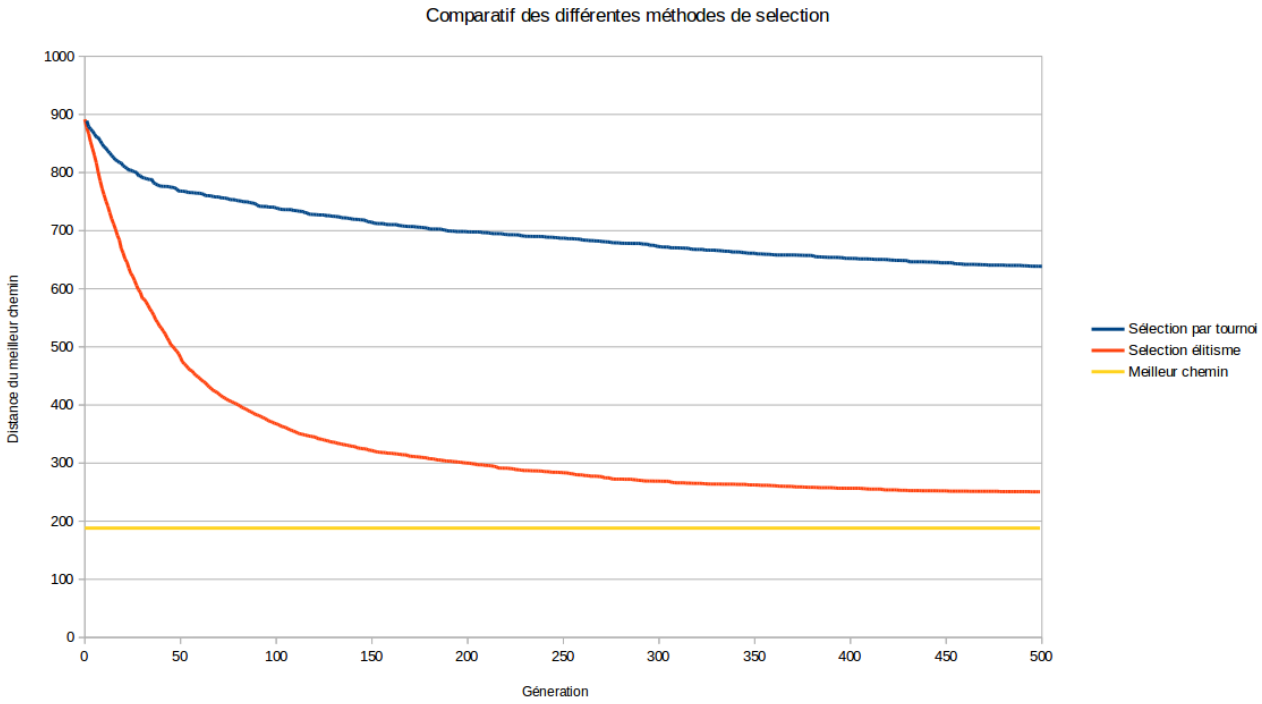
Afin de comparer nos résultats, nous testé notre programmme sous différentes conditions en faisant varier les paramètres. Le graphique ci-contre montre la convergence de l'algorithe au fil des génerations suivant deux méthodes de sélections. Les autres paramètres ont été fixés : 500 individus initiaux, 500 itérations max, 30 villes, taux de mutation à 10% et taux de selection à 50%

On remarque que la méthode élitique converge assez rapidement.

Un second teste nous permet de mesurer l'influence du taux du mutation sur vitesse de l'algorithme. On fait varier le taux de mutation de 0 à 1 avec un pas de 0.01, pour chaque pas, on effectue 60 tests et on fait leur moyenne.
On remarque que plus le taux de mutation augmente, plus l'on converge vite vers la solution optimale. Néamoins, pour la sélection par tournoi, on remarque une cassure un peu avant 0.5. La mutation permet de diversifier la population en ajoutant un facteur aléatoire à la recherche.

4 Questions

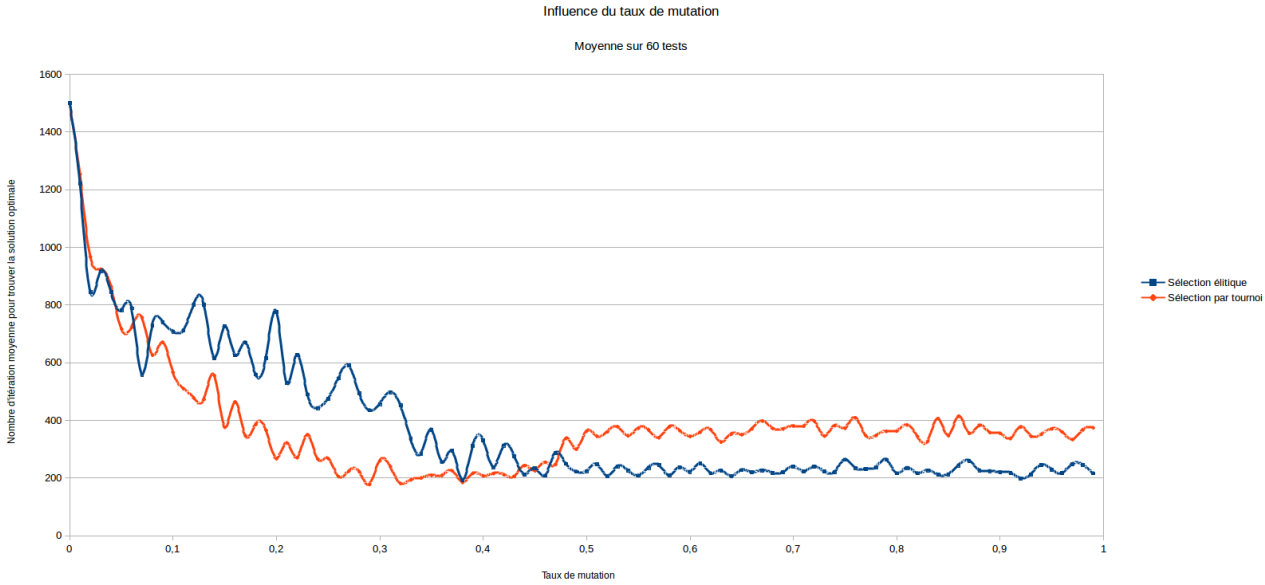
1. Quelles sont les 4 étapes d'un algorithme génétique ?
2. Quels sont les avantages des algorithmes génétiques ?



3. En quoi les algorithmes génétiques sont adaptés au problème du voyageur de commerce ?

5 Références

http://khayyam.developpez.com/articles/algo/genetic/
http://sis.univ-tln.fr/~tollari/TER/AlgoGen1/node2.html
http://www.recherche.enac.fr/opti/papers/thesis/HABIT/main002.html


Influence du taux de mutation
Moyenne sur 60 tests
Nombre d'itération moyenne pour trouver la solution optimale
1600
1400
1200
1000
800
600
400
200
0
0
0,1
0,2
0,3
0,4
0,5
0,6
0,7
0,8
0,9
1
Taux de mutation
Sélection élitique
Sélection par tournoi