

Algorithmes génétiques et application au problème du voyageur de commerce

Fabien Dubois,
Antoine RATO,
Corentin HEMBISE

<https://github.com/dut-info/Algo-Genetique>

24 mai 2016

Table des matières

1	Principe des algorithmes génétiques	2
1.1	Introduction	2
1.2	Principe	2
1.2.1	Population initiale	3
1.2.2	Evaluation des individus	3
1.2.3	Sélection des individus	3
1.2.4	Croisement des individus	4
1.2.5	Mutation des individus	4
2	Application au problème du voyageur de commerce	5
2.1	Définition du problème	5
2.2	Modélisation du problème	5
2.3	Implementation	5
3	Résultats	6
3.1	Paramétrage de l'algorithme	6
3.2	Résultats	7
4	Améliorations	7
5	Références	7

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.

La sélection naturelle est un processus de l'évolution des espèces. Les individus les plus adaptés à leur environnement, sont plus susceptibles de « survivre », alors au fil de nouvelles générations, la population devient meilleure 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, néanmoins, 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

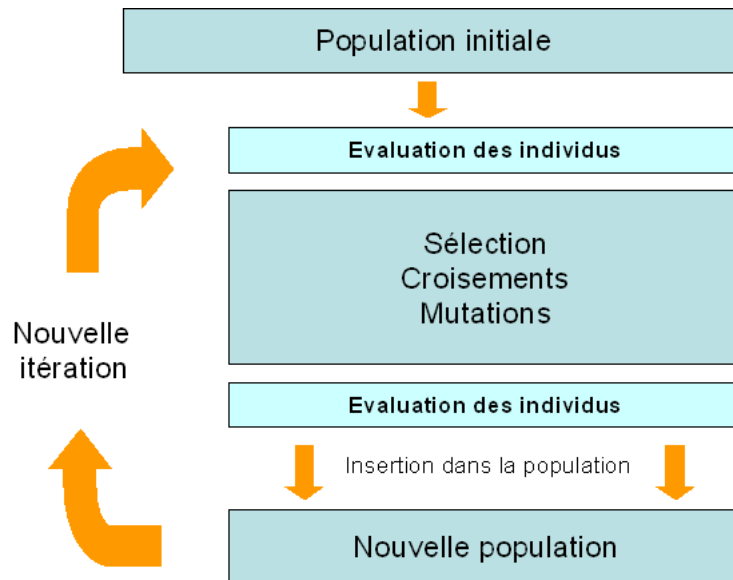


FIGURE 1 – Schéma général d'un algorithme génétique

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 où le meilleur individu n'évolue plus.

1.2.1 Population initiale

La population initiale doit être créée aléatoirement, et chaque individu doit être une solution admissible du problème. Créer aléatoirement la population permet de s'assurer de la diversité de la population. La diversité est un élément 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 problème.

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 synthétiser 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 susceptibles de se "reproduire". La méthode

de sélection doit garder une certaine diversité dans la population, en effet, un individu mauvais *auregarddesafonctionobjectif*, peut donner un enfant très bon après croisement.

Plusieurs méthodes existent :

La sélection élitique

Ce type de sélection trie les individus d'une génération 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 choisis. Tous les individus sont placés sur une roulette, plus l'individu 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 précédente, mais dans ce cas, la probabilité d'être sélectionné dépend du rang de l'individu dans la population *pralablementtrieparfitness*. La sélection par roulette présente un inconvénient, si un ou des individus domine tous les autres, la selection n'assure plus la diversité de la population, la sélection par rang permet de résoudre ce problème.

La sélection par tournoi

On forme des groupes d'individus aléatoire 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 chaîne 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 créés et doit avoir une propabilité faible, dans le cas contraire, la recherche deviendrait aléatoire.

2 Application au problème 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 objectif 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 reviendrait à parcourir l'ensemble des solutions du problème. C'est à dire l'ensemble des chemins, donc toutes les permutations 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 us :

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 envisageable, 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 modélisé 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 initiale, 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'algorithme est de minimiser cette

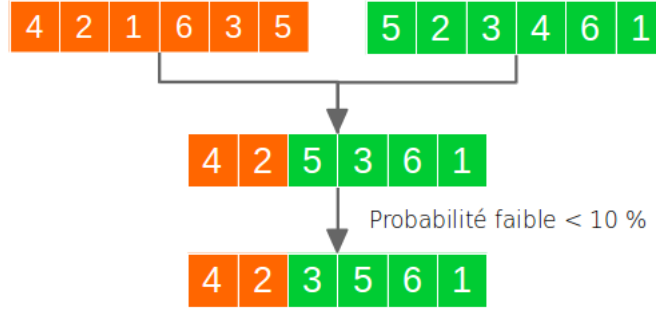


FIGURE 2 – Schéma d'un croisement et d'une mutation

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 permettra de comparer les différentes méthodes de selection.

Croisement des chemins sélectionnés : le croisement de deux chemins parents créer, en fonction du taux de croisement, un ou plusieurs individu. Pour un taux de croisement de 1, deux chemins créent 2 enfants, pour 1.5, deux chemins créer en moyenne 1.5 enfants. Ce paramètre nous permettra 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

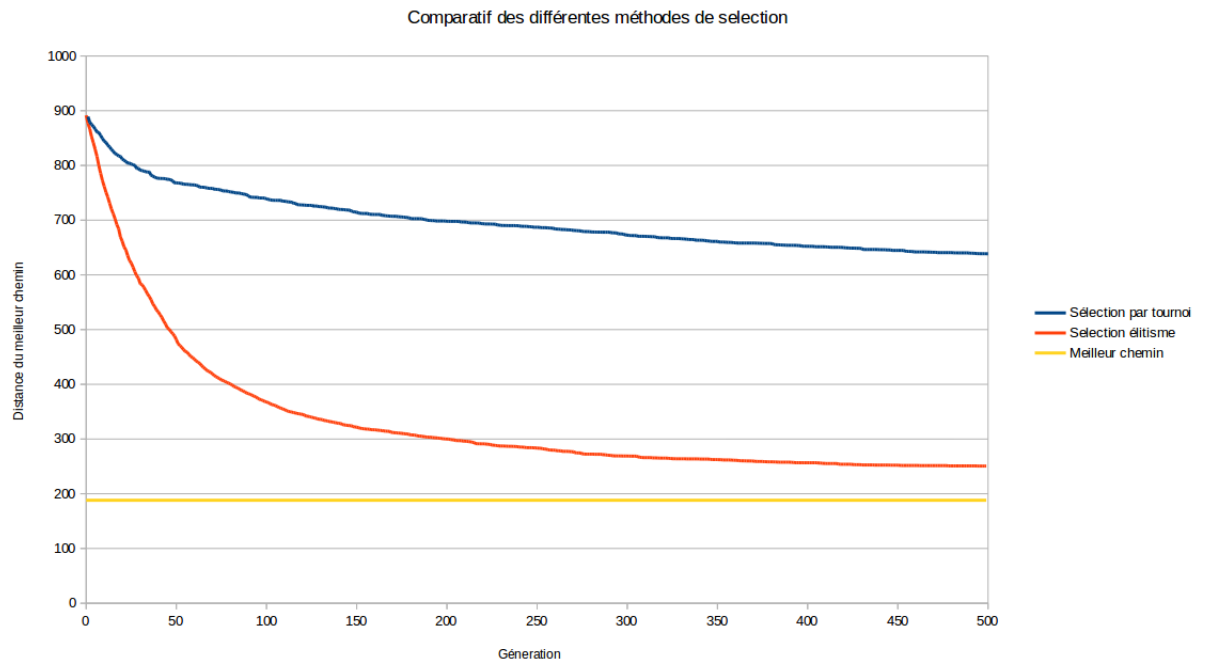
Cette méthode de croisement nous assure que chaque enfant créer soit un enfant "correct", c'est à dire qu'il soit une permutation des n villes.

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.

3 Résultats

3.1 Paramétrage de l'algorithme

Liste des paramètres sur lesquels influencer :



- Nombre d'individus initiaux]1, 1000[
- Nombre d'itérations max]1, 5000[
- Nombre de villes]10, 50[
- Méthode de sélection (Elistisme, roulette, rang, tournoi)
- Taux de croisement]1, 3[
- Taux de mutations]0, 1[
- Taux de selection]0, 1[

3.2 Résultats

4 Améliorations

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>