Optimisation TD 3
Algorithme Génétique

C. Frindel

18 Novembre 2016

1 Point Théorique

Un algorithme génétique est un algorithme d'optimisation dit *bio-inspiré*. Il s'inspire du principe d'évolution naturelle d'un ensemble de gènes (génome) pour converger vers une mutation optimale. Il s'agit de trouver, comme pour les méthodes étudiées jusqu'à là, l'ensemble de paramètres qui minimise une fonction de coût. La démarche est cependant différente puisqu'elle n'est plus basée sur les outils différentiels, mais sur une approche heuristique. L'algorithme génétique est particulièrement adapté pour des problèmes où l'évaluation de la fonction de coût n'est pas directe, que celle-ci n'est pas différentiable, et/ou lorsque l'ensemble des paramètres est très grand.

Dans le principe comme dans l'application, c'est un algorithme assez simple. On considère que chaque paramètre à optimiser est un gène, et que l'ensemble des gènes forme le génome d'un individu. L'optimisation du génome se fait grâce aux étapes suivantes résumé dans la Figure 1 :

1. On construit une population initiale de génomes choisis aléatoirement
2. La population est ensuite évaluée : on estime la valeur de la fonction de coût pour chaque génome.
3. Les génomes sont ensuite triés en fonction de leur coût respectif. Ceux qui ont les coûts les plus élevés sont ensuite sélectionnés. Pour chaque génome sélectionné, on applique une mutation et/ou un croisement aléatoires avec un autre génome de la population. (Les mutuations muettes ne seront pas prises en compte).
4. Cette nouvelle population est à son tour évaluée et mutée jusqu'à convergence. Le génome de la population qui minimise la fonction de coût sera considéré comme solution du problème d'optimisation.

2 Exercice 1 : Le problème de marche aléatoire

Nous allons appliquer l'algorithme génétique à un problème assez simple. On modélise une marche aléatoire en dimension 1 sur T pas de temps, dans la direction d :

$$x(t+1) = x(t) + d$$

avec d tiré uniformément sur le doublet $\{-1, 1\}$, et $t = \{0, \dots, T-1\}$.

On cherche à optimiser la trajectoire de la marche de sorte à ce qu'elle reste dans un intervalle déterminé $[-r_0; r_0]$. Le paramètre r_0 sera à faire varier. Le coût d'une trajectoire sera égal au nombre de pas situés hors de l'intervalle $[-r_0; r_0]$.

1. Écrire une fonction qui génère aléatoirement un génome. Elle prend T, la longueur de la marche, en paramètre d'entrée et sort un vecteur de longueur T prenant ses valeurs dans $\{-1, 1\}$ de façon aléatoire.
2. Écrire la fonction qui calcule le coût d'un génome. Elle prend en paramètre d'entrée : un génome et r_0, la borne de l'intervalle. Elle retourne un scalaire.

3 Exercice 2 : Codage de l'algorithme génétique

1. Initialisation : écrire une fonction qui créée aléatoirement une population initiale de N génomes de taille T.

2. Écrire une fonction qui trie les génomes d'une population en fonction de leur coût. Indice : utiliser la fonction `numpy.argsort()`.

3. Écrire une fonction qui sélectionne les N_s génomes ayant les valeurs de coût les plus importantes (sélection par rang). On prendra par exemple $N_s = N/2$. La valeur de sortie correspondra à la population à faire muter et/ou à croiser.

4. Nous allons maintenant procéder à la mutation des génomes sélectionnés. La mutation est définie par un taux T_m qui correspond à la probabilité qu'un gène a de muter. Il s'agit donc ici de parcourir les gènes de chaque génome et de les faire muter avec une probabilité T_m. Écrire la fonction qui permet d'effectuer cette mutation sur la population sélectionnée.

5. Nous allons maintenant effectuer le croisement de la population mutée. Le croisement se définit par un taux T_c qui indique la probabilité avec laquelle un croisement entre génomes peut intervenir.
   Coder la fonction qui effectue le croisement entre individus de la population sélectionnée avec la probabilité T_c. La position du croisement dans le génome, ainsi que l'individu avec lequel le croisement sera effectué seront tirés aléatoirement.

6. Mettre à jour la population courante avec la population mutée et incrémenter le compteur de génération.

7. Calculer le coût moyen de la nouvelle population, ainsi que le coût minimum obtenu.

8. Quel(s) critère(s) d'arrêt utiliseriez-vous ? Faire tourner l'algorithme avec $T = 500$, $r_0 = 4$, $N = 100$, $T_m = 0.05$ et $T_c = 0.1$. Tracer l'évolution du coût moyen au fur et à mesure des générations.

9. Faire varier les paramètres suivants (N, r_0, T_m et T_c) pour étudier leur influence.

10. Décrire le comportement de l'algorithme (évolution des coûts moyen et minimum) et interpréter la convergence en fonction des paramètres utilisés.

11. Quelles différences remarquez-vous par rapport aux méthodes différentielles étudiées jusqu'à maintenant (descente de gradient, Newton) ?

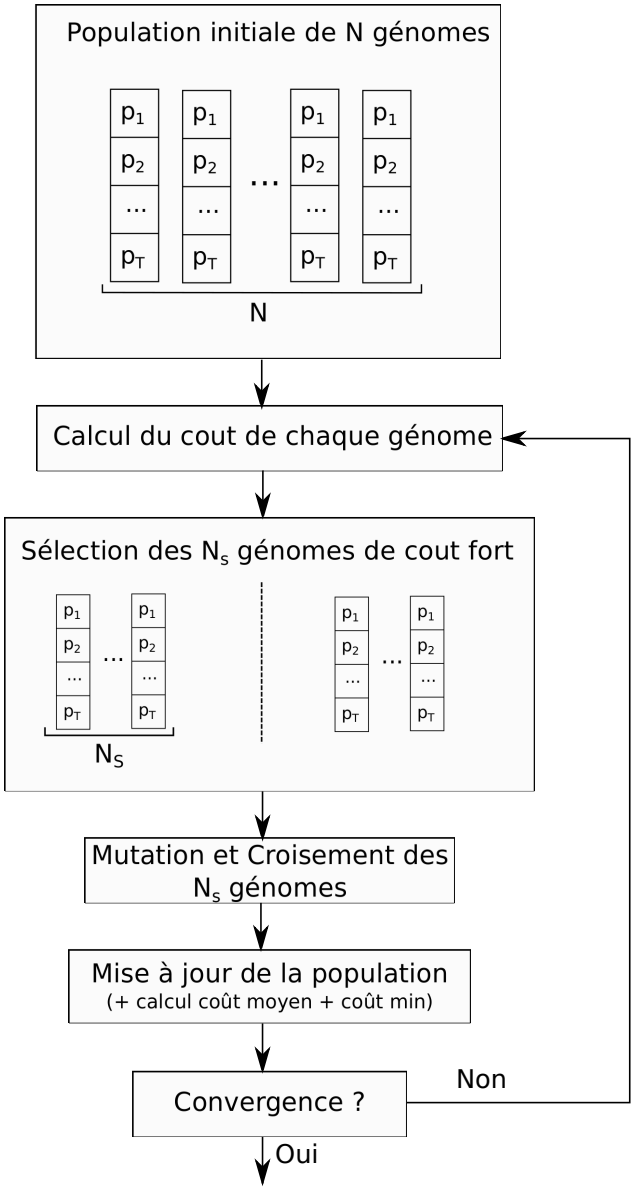

Figure 1: Schéma de principe de l'algorithme génétique