TP 3

Algorithmes génétiques

Dans un algorithme génétique associé à un problème d'optimisation (non-linéaire, avec ou sans contraintes) les individus d'une population (solutions réalisables) évoluent en appliquant itérativement des opérateurs de sélection, croisement et mutation. Un tel algorithme s'écrit *génériquement* de la façon suivante.

Algorithme génétique

Population initiale P_0 composée de m individus
$k \leftarrow 0$

Tant que (condition d'arrêt non vérifiée)
- Sélection de p individus dans P_k.
- Croisement des p individus sélectionnés pour engendrer p enfants.
- Sélection de q enfants parmi les p.
- Mutation des q enfants
- Ajout des enfants (mutés ou non) à la population
- Sélection des m meilleurs individus pour former la nouvelle population P_{k+1}.
- $k \leftarrow k + 1$

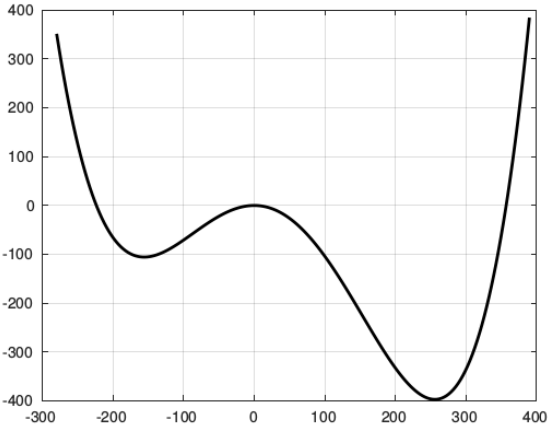
Fin Tant que

Le test d'arrêt peut porter sur la variation S de la performance moyenne de la population entre deux itérations : $S = |\overline{f}_{k+1} - \overline{f}_k|/|\overline{f}_k|$ où $\overline{f}_k$ est la performance moyenne de la population totale à l'itération k.

Dans ce TP, il vous est demandé d'implémenter un algorithme génétique pour deux problèmes d'optimisation nécessitant la construction d'opérateurs de sélection, croisement, mutation différents.

Partie 1. Minimisation d'une fonction non convexe

Soit $f : \mathbb{R} \to \mathbb{R}$ la fonction définie par $f(x) = \alpha\left(\frac{x^4}{4} + (n_2 - n_1)\frac{x^3}{3} - n_1 n_2 \frac{x^2}{2}\right)$ avec $\alpha = 5 \cdot 10^{-7}$, $n_1 = 256$, $n_2 = 156$ (cf. figure ci-dessous).



Le but de cet exercice est de résoudre le problème de minimisation (sans contrainte)

$$\min_{x \in \mathbb{R}} f(x) \tag{1}$$

par un algorithme génétique.

1. Déterminer les minima locaux de f en utilisant la fonction `fminunc` de MATLAB. Cette fonction utilise une méthode de Newton (quasi-newton en fait) avec un point initial x_0. Prendre différentes valeurs de x_0 pour obtenir tous les minima de f.
2. **Algorithme génétique**. Pour la recherche du minimum global, on se restreint à un intervalle fermé, borné $[a, b]$ suffisamment grand pour contenir tous les extrema de f. On prendra $a = -280$ et $b = 390$.
   Dans l'algorithme génétique, les individus de la population seront représentés par un nombre réel dans $[a, b]$. Les différents opérateurs utilisés dans cet algorithme sont :
   - *Opérateur de sélection* : La fonction de performance (fitness) coïncide avec la fonction f et la sélection d'individus consiste à prendre ceux qui ont les plus petites valeurs f.
   - *Opérateur de croisement* : A partir de deux individus x_1, x_2 (parents), 2 autres individus sont engendrés (enfants) en prenant aléatoirement deux nombres réels compris entre x_1 et x_2.
   - *Opérateur de mutation* : Choisir aléatoirement un nombre réel dans $[a, b]$.

   (a) Compléter le script MATLAB `minGA.m` (à récupérer sur ARCHE), en implémentant les opérateurs génétiques définis ci-dessus.

   (b) Tester la convergence en faisant varier le taux de mutation et le nombre d'individus de la population initiale

Partie 2. Algorithme génétique pour le TSP

Problème du voyageur de commerce (TSP) : Etant données N villes, on cherche le plus court trajet passant par toutes les villes une et une seule fois, en revenant à la ville de départ. La ville de départ sera toujours la ville 1. Les distances entre villes sont regroupées dans une matrice D de taille $N \times N$.

Un trajet est représenté par une permutation de la liste $(1, 2, \cdots, N)$. Dans l'algorithme génétique, un trajet correspond à un individu de la population. Les opérateurs utilisés dans l'algorithme sont les suivants :

- *Opérateur de sélection* : La fonction de performance (fitness) d'un individu est la distance totale du trajet correspond.
- *Opérateur de croisement* : Croisement PMX (Partially Matching Crossover) adapté aux permutations.
- *Opérateur de mutation* : Permutation aléatoire de deux villes dans un trajet.

1. Ecrire une fonction MATLAB `fitness.m` qui calcule la distance totale d'un trajet à partir de la matrice D. Cette fonction aura comme entête
   ```
   function F=fitness(P,D)
   ```
   où `P` est un tableau de taille $p \times N$ représentant les p individus de la population (permutations de N éléments).
2. Tester la fonction MATLAB `offspring.m` qui vous est fournie (à récupérer sur ARCHE), qui calcule un croisement PMX entre deux individus. Donner des exemples.
3. Compléter le script MATLAB `TSPga.m` (à récupérer sur ARCHE), en implémentant les opérateurs de sélection et de mutation.
4. Tester la convergence avec les données contenues dans les fonctions `data1D.m` et `data1D.m`.
5. Engendrer aléatoirement des données ($N \leq 70$) et comparer avec le solveur de programmation linéaire (cf. TP1).