Algorithmes Evolutionnaires

N. Durand

Exposé du sujet :

On va successivement optimiser 3 fonctions :

1. La fonction td_{fun} en dimension n=1, puis n=2, puis n=3.

$$td_{fun}(x) = \prod_{i=1}^{n} \Sigma_{k=1}^{5} k\, cos[(k+1)\, x_i + k]$$

   sur l'intervalle $[-10, 10]^n$. Comme le montre la figure 1 en dimension n=1, cette fonction a 3 optima équivalents que l'on va essayer de trouver avec l'algorithme génétique. On s'interessera ensuite à cette même fonction en dimension 2, puis 3.

2. La fonction de Griewank en dimension n=2, puis n=10, puis n=20.

$$griewank(x) = \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) - \frac{1}{4000} \sum_{i=1}^{n} x_i^2$$

   sur l'intervalle $[-100, 100]^n$. Comme le montre la figure 2 en dimension n=2, cette fonction a 1 optimum global en 0 et de très nombreux optima locaux.

3. La fonction de Michalewicz en dimension n=2, puis n=5, puis n=10.

$$michalewicz(x) = n + \Sigma_{i=1}^{n} \sin(x_i) \sin^{20}(\frac{i\, x_i^2}{\pi})$$

   sur l'intervalle $[0, \pi]^n$. Comme le montre la figure 3 en dimension n=2, cette fonction a 1 optimum global et beaucoup d'optima locaux.

Travail à faire :

1. Récupérer le fichier agtd.tgz et le décompresser (tar zxf agtd.tgz). Le répertoire AG_TD contient plusieurs fichiers qui permettent d'exécuter une algorithme génétique. Pour compiler l'algorithme il suffit d'éxecuter la commande *make*. Pour exécuter l'algorithme, il suffit de taper *agnorm* ou *agopt* (mode optimisé). **Le but du TD est d'optimiser différentes fonctions en ajustant les paramètres dans le fichier general.cfg**.

2. Parcourir le fichier *local.ml* qui définit toutes les fonctions utiles pour l'algorithme génétique.

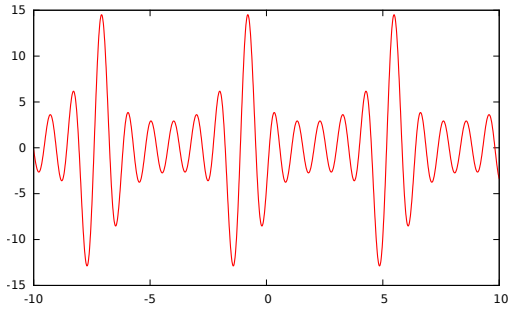

Figure 1: Fonction td_{fun} en dimension n=1.

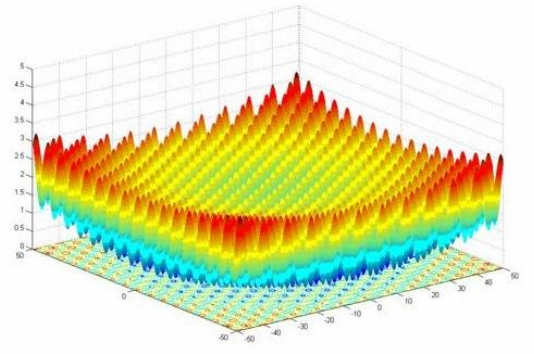

Figure 2: Fonction *griewank* en dimension n=2.

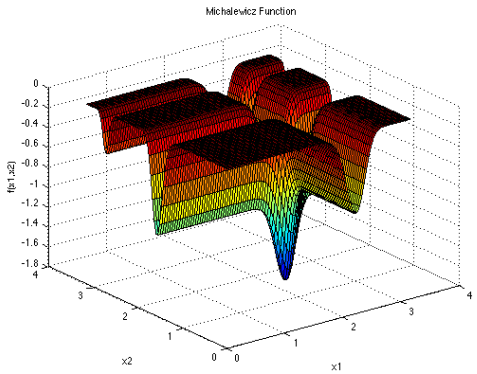

Figure 3: Fonction *michalewicz* en dimension n=2.

- le type *data* des données
- les bornes du problèmes
- la fonction d'évaluation
- la fonction qui initialise un individu dans l'espace de recherche
- la fonction de *mise à l'echelle*
- les opérateurs de *croisement* et de *mutation*
- la fonction d'affichage
- la fonction *datadistance* pour l'opérateur de *sharing*
- la fonction *dataBarycenter* pour l'opérateur de *sharing clusterisé*

3. Parcourir le fichier *general.cfg* définit les paramètres de l'algorithme génétique:
   - la fonction à optimiser ($typefun$)
   - la gestion des contraintes de bord ($typeconst$)
   - le type de croisement ($typecross$)
   - l'amplitude de la mutation ($typemut$)
   - le nombre de générations de l'algorithme génétique ($nbgens$)
   - la taille de la population ($nbelems$)
   - le pourcentage de la population croisé ($pcross$)
   - le pourcentage de la population muté (pmut). (la somme $pcross + pmut$ doit rester inférieure à 1)
   - le type de mise à l'échelle ($scaling$)
   - l'utilisation ou non de l'élitisme ($elitist$)
   - l'utilisation du sharing ($sharing$). Le coefficient $sharing$ doit appartenir à $[0, 1[$. Lorsqu'il vaut 0 il n'y a pas de $sharing$. Losqu'il vaut $0 < x < 1$, le programme sauvegarde le meilleur élément de chaque cluster si sa fitness vaut au moins x fois la valeur du meilleur élément courant.
   - la valeur de d_{max} ($complex\_sharing$) qui définit la taille d'un cluster. Si elle est trop grande, on ne distingue pas deux sommets différents. Si elle est trop petite, elle devient trop selective et considère que des points sur le même sommet sont dans des clusters différents.
   - le paramètre ($evolutive$) est utilisé lorsqu'on traite des problèmes où la fonction d'évaluation est évolutive au cours du temps.
   - la racine du générateur de nombre aléatoire est définie par le paramètre ($seed$).
4. Choisir la fonction td_{fun} en dimension n=1. En modifiant les paramètres de sharing, essayer de trouver les trois optima de la fonction.
5. Passer en dimension n=2. Combien d'optima peut-on théoriquement trouver ? En jouant avec tous les paramètres de l'algorithme génétique, essayer de trouver tous ces optimas. En jouant sur le paramètre $sharing$, essayer de trouver des optima locaux secondaires.

6. En dimension n=3, le nombre d'optimas augmente à ... Essayer de les trouver tous.

7. Choisir la fonction *griewank* en dimension n=2. Vérifier que l'algorithme génétique converge vers $(0, 0)$ et passer en dimension n=10 pour trouver l'optimum $(0, .., 0)$.

8. Toujours avec la fonction *griewank*, passer en dimension n=20 pour trouver l'optimum $(0, .., 0)$.

9. Tester la fonction *michaelewicz* en dimension n=2. L'optimum est atteint quand $f(x) = 3.8013$. Passer en dimension n=5 (9.687658), puis en dimension n=10 (19.66015).