M4202Cip

Algorithmes génétiques

23 janvier 2020 – B. Colombel

1 De quoi s'agit-il ?

Les algorithmes génétiques (AG) font partie des algorithmes évolutionnistes. Ils s'inspirent des mécanismes de la sélection naturelle pour trouver une solution à un problème d'optimisation.

La solution trouvée pourra n'être qu'une solution approchée de la solution optimale, mais les algorithmes génétiques peuvent s'appliquer dans des cas où l'espace de recherche des solutions est très important et où il peut être couteux et/ou compliqué de calculer la solution optimale.

Dans ce projet vous allez programmer les algorithmes génétiques pour résoudre plusieurs problèmes d'optimisation plus ou moins simples.

Il existe de très nombreuses variantes d'algorithmes génétiques. Ils s'appuient tous sur les mêmes principes. Nous n'utiliserons qu'une de ces variantes dans ce projet. Regardons cela plus en détails ...

1.1 Darwinisme et algorithmes génétiques

Sources : Wikipedia

Dans son livre *De l'origine des espèces*, Darwin expose une théorie selon laquelle, étant donné que tous les individus d'une espèce diffèrent légèrement entre eux et d'une génération à l'autre, et que seule une partie de ces individus réussit à se reproduire, seuls les descendants des individus les mieux adaptés à leur environnement survivront et se reproduiront en transmettant les variations utiles à leur survie.

Ainsi, comme les individus sélectionnés transmettent leurs caractères à leur descendance, les espèces évoluent et s'adaptent en permanence à leur environnement. Il baptise du nom de « sélection naturelle » cette sélection des individus les mieux adaptés.

La sélection naturelle repose sur trois principes :
- le principe de **variation** : les individus diffèrent les uns des autres, ainsi que d'une génération à l'autre,
- le principe d'**adaptation** : les individus les plus adaptés au milieu survivent et se reproduisent davantage,
- le principe d'**hérédité** : les caractéristiques d'une espèce sont héréditaires, on les retrouve donc d'une génération à l'autre.

Les algorithmes génétiques appliquent ces principes à la résolution de problème.

2 Déroulement d'un algorithme génétique

On dispose d'un problème dont on cherche à trouver une solution optimale. Cette solution optimale doit être cherché dans un espace de solutions. Celui-ci correspond à l'ensemble de toutes les valeurs possibles qui sont potentiellement la solution recherchée.

Chercher une solution *optimale* suppose que l'on est en mesure de comparer les solutions entre elles et de déterminer la meilleure. Il est donc nécessaire que l'on dispose pour chaque problème étudié d'une fonction qui permette d'évaluer la qualité d'une solution donnée. Cette fonction d'évaluation est également appelée *fonction d'adaptation* ou *fonction de fitness*. On suppose que la valeur de cette fonction est croissante (resp. décroissante) avec la qualité de la solution. La solution optimale est donc celle qui maximise (resp. minimise) cette fonction.

Les algorithmes génétiques travaillent sur un sous-ensemble de l'espace des solutions. On appelle alors *population* ce sous-ensemble et *individu* une de ces solutions.

2.1 Principe

Le principe des algorithmes génétiques est de faire évoluer la population en lui appliquant des opérateurs génétiques pour produire une nouvelle génération. En reprenant les principes de Darwin, il s'agit de favoriser les individus les mieux adaptés lors de la production de la génération suivante. Les individus les mieux adaptés sont ceux qui optimisent la *fonction de fitness*.

L'idée est qu'après n générations, il ne reste que les individus les mieux adaptés. Parmi eux, celui qui est le mieux adapté est la solution optimale recherchée ou, à défaut, en est une bonne valeur approchée.

2.2 Opérateurs génétiques

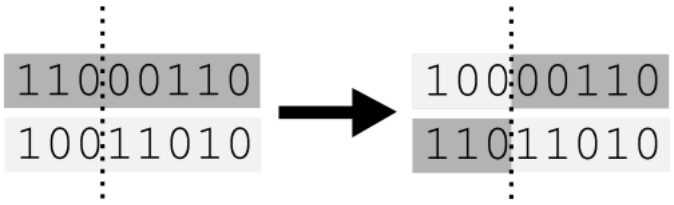
Chaque individu dépend d'un certain nombre de paramètres que l'on peut coder dans une chaîne par une séquence de symboles pris parmi un nombre fini de symboles (par exemple les lettres de l'alphabet, les chiffres 0 et 1, etc.). Ces symboles représentent les caractéristiques d'un individu, on parle également de gène. Pour tous les individus la longueur de la séquence est la même. Cette séquence est aussi appelée génome de l'individu.

La sélection : consiste à choisir les individus d'une population qui vont se retrouver à la génération suivante. La règle est que les mieux adaptés ont plus de chance de survivre. Il existe plusieurs manière de réaliser cette sélection.

le croisement : combine les caractéristiques (gènes) de deux individus pour en produire deux nouveaux.

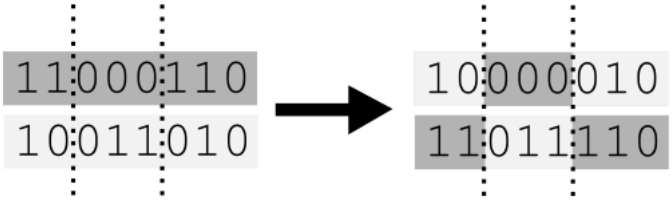
Il existe essentiellement deux types de croisements : le croisement 1 point et le croisement 2 points. Ils se réalisent à partir deux individus de la population.

Le *croisement 1 point* découpe chacune des séquences de symboles représentant un individu en deux portions. Le point de coupe est choisi au hasard mais est le même pour les deux individus. On construit ensuite deux nouveaux individus en échangeant les portions des deux individus de départ.



croisement 1 point

Le *croisement 2 points* découpe chacune des séquences de symboles représentant un individu en trois portions. Les points de coupe sont choisis au hasard mais sont les mêmes pour les deux individus. On construit ensuite deux nouveaux individus en échangeant les portions intermédiaires des deux individus de départ.



croisement 2 points

La mutation : correspond à une modification aléatoire des caractéristiques d'un individu.

Cette opération est généralement réalisée avec une probabilité faible. Son existence permet de faire apparaitre dans une population des caractéristiques qui étaient jusque là absentes et donc d'explorer des parties de l'espace de solutions qui pouvaient être délaissées.

2.3 Algorithme

Pour fixer plus simplement les idées, voici l'écriture générique d'un algorithme génétique :

```
Pour chaque Individu dans Groupe
    Initialiser Individu
FinPour

Pour nombre d'iterations
    Parent A = Sélection d'un Individu (Groupe)
    Parent B = Sélection d'un Individu (Groupe)
    Fils = Recombinaison (Parent A, Parent B)
    Si hasard > pourcentage Alors
        Appliquer une mutation à Fils
    FinSi
    Optimiser Fils // Optionnel
    Evaluer Fils
    Si Fils est accepte dans Groupe Alors
        Reinserer Fils dans Groupe
    FinSi
FinPour
```

3 Application au problème du voyageur de commerce

Le problème est le suivant : on se donne n villes, et on souhaite déterminer un chemin qui passe par chaque ville exactement une fois (et revient à son point de départ) qui soit de longueur minimale. Si les villes sont numérotées de 1 à n, un chemin est codé par une permutation des entiers $1..n$, c'est-à-dire la liste ordonnée des villes visitées, chaque ville n'apparaissant qu'une seule fois.

Question 2 : Combien y a-t-il d'individus possibles (c'est à dire de parcours) qui sont codés par une chaîne binaire de longueur $\ell = 100$?

Question 2 : Combien y a-t-il de permutations des entiers $1..n$? Est-il envisageable de toutes les énumérer lorsque $n = 10$? $n = 30$?

Il faut adapter les opérations de croisement et de mutation afin d'avoir toujours des permutations.

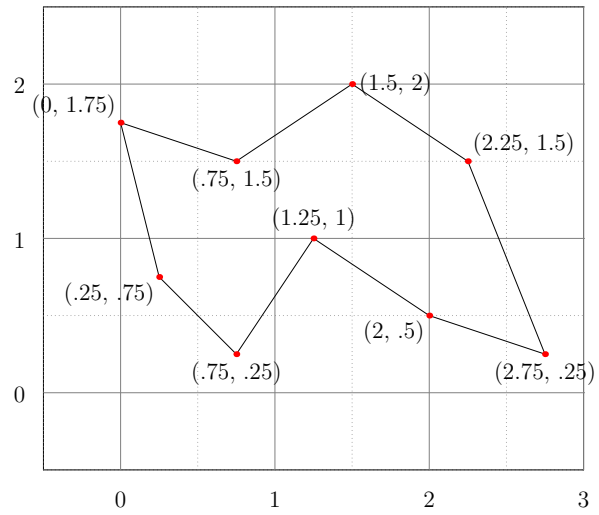
4 Travail demandé

1. Exploiter les documents fournis et ceux que vous pourrez trouver par vous même pour vous documenter sur les notions d'algorithme génétique et de voyageur de commerce.
2. Implémenter un algorithme génétique, appliqué par exemple au problème de voyageur de commerce.
   Si votre algorithme est performant, vous pourrez participer au défi des 250 villes :
   http://labo.algo.free.fr/defi250/defi_des_250_villes.html.
3. Rédiger un rapport présentant le travail effectué et incluant les réponses aux questions et en mettant en évidence les outils et les théories mathématiques utiles.

Fichiers d'entrée et de sortie

Vous reprendrez les fichiers d'entrée du TP5 du module de modélisation `20.in3` , `100.in3` , `1000.in3` et `10000.in3` pour $n = 20{,}100{,}1\,000{,}10\,000$. Les fichiers sont basés sur les coordonnées des villes de différents pays.
On rappelle que consiste de deux flottants par ligne séparés par un espace, sauf la première ligne qui contient l'entier n. Les n lignes de deux flottants (doubles) correspondent aux coordonnées x, y de chaque point.
Par exemple, pour le graphe ci-contre, on aura :

```
9
0 1.75
1.5 2
.75 1.5
.25 .75
.75 .25
1.25 1
2 .5
2.25 1.5
2.75 .25
```



Pour calculer la distance entre deux villes, vous utiliserez la distance euclidienne :

$$\text{dist}(A,B) = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

Les graphes seront ainsi considérés comme **complets**.

Le format du fichier de sortie est presque le même que ce du fichier d'entrée. La première ligne du fichier de sortie est l'entier n. La deuxième est la longueur du tour. Ensuite on liste les points de l'entrée, mais selon l'ordre du parcours obtenu. Pour l'exemple on a :

```
9
8.270864494922767
0 1.75
.75 1.5
1.5 2
2.25 1.5
2.75 .25
2 .5
1.25 1
.75 .25
.25 .75
```

Peu importe quel est le premier point utilisé, mais on ne le répète pas à la fin. Par contre, pour calculer la longueur, il faut compter le retour au premier point.

Vous pourrez également tester vos programmes avec le fichier `grader2.py` fourni avec le sujet.

5 Ressources

— Une description (en français) du problème du voyageur de commerce et de son implémentation avec un algorithme génétique.
— Une description (en anglais) du codage, des opérations de croisement (plusieurs options possibles) et de mutation adaptées au problème du voyageur de commerce.
— On pourra consulter également le site http://labo.algo.free.fr/pvc/algorithme_genetique.html