ECOLE CENTRALE DE NANTES

Option Réalité virtuelle - DS C++

3 novembre 2014- durée 2h - tous documents autorisés

Recommandation : Lisez l'énoncé jusqu'au bout avant de vous lancer !

1 ALGORITHMES GÉNÉTIQUES

1.1 GÉNÉRALITÉS

Un algorithme génétique est un algorithme dont le fonctionnement est basé sur les idées de reproduction et de sélection issues de la théorie de l'évolution. Etant donné un problème à résoudre, on commence par déterminer l'espace des solutions de ce problème (c'est-à-dire à coder cet ensemble de solutions). On teste ensuite un grand nombre de solutions possibles en les choisissant au hasard. Parmi ces tentatives, certaines seront plus proches que d'autres de la solution. On effectue donc un classement des solutions proposées et on sélectionne les meilleures. En effectuant des opérations de *croisement* et de *mutation* parmi cette sélection, on génère un nouvel ensemble de solutions. On itère ensuite ce processus de sélection et de transformation jusqu'à trouver (ou pas) la bonne solution.

1.2 EXEMPLE

On s'intéresse ici au problème du mot mystère (il sera abordé dans la seconde partie du DS, la première restant très générale) : il consiste à trouver un mot choisi au hasard de longueur n en disposant pour seule information du nombre de lettres correctes à chaque tentative (pas forcément dans l'ordre). Par exemple, pour $n = 4$, on cherche le mot `alea`. L'ensemble des solutions est l'ensemble des mots de 4 lettres de taille 26^4. On commence par générer au hasard les 4 mots suivants :

— erty
— akil

— bler
— hyyy

Parmi ces 4 mots, `akil` et `bler` sont ceux qui ont le plus de lettres en commun (2) avec notre mot mystère. Nous allons donc les sélectionner et les croiser et muter... On mélange alors les lettres de ces deux mots et on en change certaines au hasard. On dispose alors d'une nouvelle liste de mots comme celle-ci :

— akil
— bler
— aley
— bljr

Grâce à l'opération de croisement / mutation, on a généré ici le mot `aley` qui s'approche de la solution. Il suffit ensuite d'itérer le processus...
La première partie de ce problème est consacrée à la programmation générique des algorithmes génétiques.

2 TRAVAIL À EFFECTUER

2.1 ALGORITHME GÉNÉRIQUE

Cette première partie est totalement indépendante du problème du mot mystère qui sera abordé plus loin.

QUESTION 1 Définissez une classe abstraite `genome` qui permettra des opérations de croisement / mutation. La mutation est une opération interne à un génome tandis que le croisement nécessite un autre génome avec lequel effectuer ce croisement. Vous devrez également fournir une opération qui vérifie si un génome est meilleur qu'un autre (elle retournera un nombre positif dans ce cas, négatif si moins bon et nul si on ne peut pas juger). Il est recommandé de **pas** surcharger l'opérateur < pour ceci.
Nous allons maintenant définir les éléments d'une classe `population` qui permettra de manipuler des listes de `genome`.

QUESTION 2 Expliquez pourquoi il vaut mieux que la classe `population` manipule des listes de pointeurs vers des génomes que des listes de génomes à proprement parler ? Définissez la classe `population` avec ses attributs.

QUESTION 3 Écrivez une méthode renvoyant la sélection de n meilleures solutions (meilleurs génomes) au sein d'une liste de `genome`.

QUESTION 4 Écrivez maintenant une méthode renvoyant une liste de n `genome` issus de la reproduction d'une liste d'individus passée en paramètre. Pour cela, vous effectuerez n fois l'opération suivante : choisir deux individus au hasard (2 à 2 distincts) et effectuer leur croisement.

QUESTION 5 Écrivez une méthode renvoyant une liste **mutée** des individus passés en paramètres.

QUESTION 6 Enfin, programmez une méthode `generation()` qui créera à partir d'une liste d'individus et des paramètres m et n, une nouvelle liste de solutions de taille n : pour créer cette nouvelle génération, vous sélectionnerez les m meilleures solutions puis générerez $n - m$ croisements et enfin chacune des solutions sera mutée.

2.2 ÉTUDE DE CAS : RÉSOLUTION DU PROBLÈME DU MOT MYSTÈRE

Le but de cette partie est de programmer une classe `motmystere` qui va permettre de résoudre (à l'aide de ce qui a été fait précédemment) le problème du mot mystère. On suppose que les mots sont stockés sous forme de chaînes de caractères. Le mot à trouver sera stocké sous forme de variable de classe, tandis que la tentative (le mot essayé) utilisera une variable d'instance. Le taux de mutation sera également stocké sous forme de variable de classe (initialisée à 5%).

QUESTION 7 Écrivez la structure de la classe `motmystere` et ajoutez-y un constructeur permettant de créer le mot tentative au hasard. Le constructeur prendra comme seul argument la longueur du mot recherché.

QUESTION 7 Écrivez des accesseurs en lecture et écriture sur le mot à rechercher.

QUESTION 8 Écrivez la méthode de mutation transformant la chaîne de caractères abritant la tentative par mutation. Pour chaque caractère, vous effectuerez une mutation (remplacement au hasard) si un nombre tiré au hasard est supérieur au taux de mutation.

QUESTION 9 Écrivez la méthode de croisement avec un autre `genome`. Cette méthode permettra de croiser la tentative appelante avec une autre tentative. La nouvelle tentative sera formée avec les n premiers caractères de l'appelant tandis que les autres caractères seront les derniers caractères du mot passé en paramètre. n est un nombre tiré au hasard compris entre 0 et la longueur du mot recherché.

QUESTION 10 Écrivez la méthode de comparaison permettant de comparer deux mots tentatives : si n est le nombre de lettres de la tentative appelante coïncidant avec celles du mot mystère et si m est le symétrique pour la tentative paramètre, la méthode doit renvoyer $n - m$.

QUESTION 11 Écrivez une fonction principale permettant de rechercher le mot "alea" par un algorithme génétique. Vous pouvez fixer les paramètres comme vous le souhaitez.