Intelligence Artificielle
TP3: Le Compte Est Bon - Instructions

1) Contexte

Nous souhaitons réaliser une IA basée sur les algorithmes génétiques permettant de résoudre le problème du compte est bon.

1.1) Objectif - Le compte est bon

Etant donné les chiffres de 0 à 9 et les opérateurs +, -, * et /, trouver une expression qui, une fois évaluée, se réduit au nombre cible donné. Les opérateurs seront appliqués de gauche à droite (pas de priorité des opérations).

Par exemple, pour la cible 23, l'expression `6+5*4/2+1` est une solution possible. Si la cible est 75.5, alors `5/2+9*7-5` est une autre solution possible.

1.2) Encodage d'un individu

Pour commencer, nous devons encoder le chromosome d'un individu (i.e., une solution possible) à l'aide de gènes. Chaque gène représentera alors un symbole, c'est-à-dire un chiffres de 0 à 9 ou un opérateur +, -, * et /. Une expression sera constituée de plusieurs gènes.

Une manière d'encoder un gène est d'utiliser un codage binaire, avec quatres bits pour tous les symboles possibles :

```
0:      0000
1:      0001
2:      0010
3:      0011
4:      0100
5:      0101
6:      0110
7:      0111
8:      1000
9:      1001
```

```
+:          1010
-:          1011
*:          1100
/:          1101
```

Les gènes `1110` & `1111` resterons inutilisés et devront être ignorés par l'algorithme.

En reprenant l'exemple ci-dessus de la cible 23, l'expression `6+5*4/2+1` serait représentée par quatres gènes de cette manière :

```
0110 1010 0101 1100 0100 1101 0010 1010 0001
  6    +    5    *    4   /    2     +    1
```

Le chromosome de cet individu est donc:

```
01101010010111000100110100101010000 1
```

Remarque concernant le décodage

Puisque l'algorithme utilise des arrangements aléatoires de bits, il rencontrera probablement des expressions telles que :

```
00100010101011101011011100 10
```

Qui, décodée représente :

```
0010  0010  1010  1110  1011  0111  0010

  2     2     +    n/a    -     7     2
```

Cette expression ne pas pas être évaluée ! Proposition: en décodant, l'algorithme ignorera simplement tout gène qui ne se conforme pas à la règle attendue :

```
chiffre -> opérateur -> chiffre -> opérateur -> etc...
```

Nous ignorant donc les symboles qui sont en surplus de notre règle, en partant de gauche à droite. Cela donnera:

```
2   +   7
```

2) Le projet

Le but du projet est de se baser sur l'exemple du module `TP3.py` pour permettre de générer et d'évaluer des solutions. Votre module sera appelé automatiquement avec la fonction `run_ag` pour évaluer la performance de votre solution.

Votre module devra garder les noms des fonctions proposées, en développant les points suivants:

- Le point d'entrée principal, à savoir `run_ag`.
- Une fonction d'encodage

- Des opérateurs de croisement et de mutation
- Une fonction de fitness
- Une fonction d'affichage d'un individu (individu->str)

Le but du projet étant d'optimiser votre AG, il conviendra de tester différentes valeurs pour les paramètres possibles et d'explorer les alternatives. La version finale que vous soumettrez sera la version que vous avez optimisée.

3) Organisation

- Le projet est réalisé par groupe de deux personnes
- Les semaines de cours du 03.11, du 10.11 et du 17.11 seront consacrées à ce projet
- Un rapport qui explique votre démarche doit être rendu. Il faut au minimum une introduction, une conclusion, une analyse, et expliquer les choix spécifiques pour votre IA, en particulier les opérations de croisement et de mutation, le fitness et les paramètres. Un plot sur la convergence du fitness en fonction des itérations devra être fourni et interprété.
- À rendre : `Nom1_Nom2.py`
- Ajouter un fichier README.md avec quelques commentaires sur ce qui fonctionne, ce qui ne fonctionne pas, et les éventuels autres points notables de votre implémentation.
- **Le projet est à rendre pour le dimanche 19 novembre à 23:59 sur Cyberlearn**

4) Évaluation

Le projet donnera lieu à une note. Il sera évalué selon les critères suivants :

- Qualité de l'implémentation de votre IA (représentation d'un individu, recherche de paramètres optimaux, …) (50%)
- Qualité du code (maîtrise du langage, lisibilité, commentaires), respect des critères (20%)
- Rapport (30%)

De plus, il est judicieux de faire particulièrement attention au nettoyage de votre code avant de le rendre :

- Pas de code inutile ! (Retirer les import et fonctions qui ne servent plus à rien, …)
- Évitez les blocs de 50 lignes de code mis en commentaire "pour l'instant" !
- Vérifier qu'une solution est toujours retournée, et que le critère d'arrêt est respecté
- Est-il encore besoin de le préciser : Commentez votre code !

/FAL (26.10.23)