16–23/01/2020

Enseignant : Simon Forest – simon.forest@liris.cnrs.fr
Sujet disponible sur perso.liris.cnrs.fr/sforest/ibi/

Intelligence Bio-Inspirée
TP : Algorithmes génétiques

1 Introduction

Vous avez perdu le mot de passe permettant d'accéder à un logiciel très important, et vous n'avez aucun moyen de le récupérer ! Heureusement, le logiciel n'est pas très bien sécurisé, et il vous indique à chaque tentative à quel point vous être proches de retrouver le mot de passe. Cette mesure de proximité semblant assez complexe, vous allez utiliser un algorithme génétique pour retrouver votre mot de passe.

2 Mise en place

Ce travail est à faire par binômes. Un identifiant unique à chaque binôme vous sera attribué pendant la première séance, vous devrez le conserver pendant tout le TP et l'indiquer sur votre rapport.

Si vous étiez absent à la première séance, envoyez-moi un mail avec les noms des deux membres du binôme, et je vous enverrai un identifiant.

Vous avez à votre disposition un fichier de code Python pré-compilé, qui vous permettra de comparer vos tentatives au vrai mot de passe. Le code lui-même ne vous est pas accessible[1]. Vous pouvez importer la méthode de la façon suivante :

```
from blackbox import check
```

Elle prend deux arguments, l'identifiant de votre binôme et une chaîne de caractère.

```
similarity = check(group_id, password_attempt)
```

Elle renvoie une valeur située entre 0 et 1, où 1 signifie que la chaîne de caractères donnée en entrée est exactement identique au mot de passe à trouver. Vous ne connaissez pas la manière dont les chaînes de caractères sont comparées. La mesure de similarité n'est pas idéale, et elle peut comporter des maxima locaux, ou vous envoyer localement dans la mauvaise direction.

3 Objectifs

Vous devez utiliser un algorithme génétique pour trouver le mot de passe, ou au moins s'en approcher le plus possible. Vous en avez les informations suivantes :

- Il est composé uniquement de chiffres et de lettres majuscules.
- Il contient entre 12 et 18 caractères.

Un « individu » dans votre algorithme correspondra donc à une tentative de mot de passe, son phénotype étant une chaîne de 12 à 18 chiffres et lettres majuscules.

[1] Il vous est interdit de décompiler le fichier.

Vous ne serez pas évalués uniquement sur la découverte ou non du bon résultat, mais surtout sur la conception de l'algorithme et la justification de vos choix concernant :

- le codage du génotype ;
- la sélection (et la présence ou non d'élitisme) ;
- la mutation ;
- le cross-over ;
- les valeurs des hyper-paramètres (nombre d'individus, taux de mutation, etc.).

Vous aurez sûrement besoin de plusieurs itérations successives de votre algorithme, n'hésitez pas à observer son comportement pour l'améliorer[2].

4 Travail à rendre

Vous devrez m'envoyer, par mail à **simon.forest@liris.cnrs.fr**, votre livrable composé de :

- votre code source, commenté, compressé dans une archive ;
- votre rapport au format PDF **ou** *Jupyter Notebook*.

La date limite est fixée au 9 février 2020.

4.1 Code

Vous devez programmer un algorithme génétique en Python. Je vous recommande de structurer votre code en plusieurs classes et plusieurs fichiers. Mettez à part la définition des hyper-paramètres, pour qu'il soit plus facile de les retrouver et modifier (par exemple à l'aide d'un fichier de configuration).

Votre code doit pouvoir être lancé tel quel, sans modifier les paramètres, et donner en sortie le mot de passe trouvé (ou la solution la plus proche). Vous ne devez pas fixer la *seed* dans le code[3]. Précisez dans le rapport si vous utilisez Python 2 ou 3[4].

4.2 Rapport

Vous devez expliquer et justifier vos choix pour l'implémentation de votre algorithme et le réglage des hyper-paramètres. Certains arguments seront assez empiriques, mais vous pouvez aussi mentionner des implémentations que vous avez essayées et qui n'ont pas fonctionné comme vous l'espériez.

Vous devrez aussi analyser l'efficacité de votre algorithme. Vous pourrez prendre en compte la quantité d'opérations effectuée, le temps de convergence, etc. Vous pourrez vous appuyer sur des graphiques, par exemple :

- évolution de la meilleure *fitness* ou *fitness* moyenne au cours des générations ;
- *fitness* en fonction de la distance entre le meilleur phénotype et le mot de passe final ;
- influence d'un hyper-paramètre sur le temps de convergence ;
- etc.

Si vous avez réussi à trouver votre mot de passe, vous pouvez tester votre algorithme sur d'autres exemples, que vous obtiendrez en ajoutant des multiples de 100 à votre `group_id` dans l'appel de `check`. (Par exemple, si votre numéro de groupe est 3, vous pouvez tester `3`, `103`, `203`, etc.)

[2] Observez si la *fitness* croît, stagne, ou même re-diminue. Observez aussi comment les phénotypes se rapprochent de la solution.

[3] Vous pouvez néanmoins en prendre une pour vos tests. Et si votre algorithme ne fonctionne pas pour une *seed* aléatoire, vous pouvez spécifier celles qui fonctionnent dans le rapport.

[4] Les deux sont utilisables, mais le mot de passe attendu est différent selon la version utilisée.