

Algorithme génétique - Secret

Contexte

Le problème est celui de faire trouver une chaîne de caractères *secret* à un algorithme génétique.

On suppose que le *secret* est constituée à partir du caractère espace et des 26 lettres de l'alphabet (minuscules et non accentuées). Si elle est le longueur 30, il existe alors 27^{30} chaînes possibles.

Nous allons donc vérifier qu'en explorant qu'une petite partie de toutes ces possibilités, un algorithme génétique est capable de trouver l'une d'entre elles.

Par exemple, nous allons considérer la chaîne *secret* = 'ceci est un secret a retrouver'.

Modélisation

Pour rappel sur le principe des algorithmes génétiques :

- travailler avec un sous-ensemble P de R appelé *population* ;
- On fait évoluer P en lui appliquent des opérateurs génétiques pour produire une nouvelle génération ;
- On favorise les individus les mieux adaptés lors de la production de la génération suivante ;
- Après n générations, on choisit un individu dans la population.

Génome

Un individu représente une chaîne de n caractères (n étant pair)

Son génome étant la suite de n caractères et un gène représente un caractère parmi les 27 possibles.

Travail à Effectuer

1. Écrire la fonction `get_gene`, qui prend en paramètre un génome g et l'indice i d'un gène, et renvoie sa valeur.
2. Écrire la fonction `set_gene`, qui prend en paramètre un génome g , l'indice i d'un gène, la valeur v du gène et a pour effet de bord de modifier la valeur du gène i à v .

Sélection

La fonction *fitness* correspond à la distance de *Hamming* entre 2 génomes.

La distance de Hamming entre deux éléments a et b de F est le nombre d'éléments de l'ensemble des images de a qui diffèrent de celle de b .

Exemple de calcul de la distance de Hamming :

Soit le secret g_s :

g_s	r	a	m	e	u	r
-------	---	---	---	---	---	---

Et les deux génomes g_1 et g_2 :

g_1	c	a	s	s	e	r
g_2	r	a	l	e	u	r

$fitness(g_s, g_1) = 4$, car g_1 a 4 lettres qui diffèrent par rapport à g_s (c, s, s, e).

$fitness(g_s, g_2) = 1$, car g_2 a 1 lettre qui diffère par rapport à g_s (l).

Plus une combinaison a un *score* faible, plus l'individu associé est adaptée pour l'algorithme génétique.

L'objectif étant que l'algorithme génétique génère à une itération i un génome g_i où $fitness(g_s, g_i) = 0$, autrement dit que g_i ne possède aucune lettre qui diffère par rapport à g_s .

Travail à Effectuer

1. Écrire la fonction `fitness`, qui prend en paramètres 2 génomes g_1 et g_2 et renvoie , la distance de Hamming entre g_1 et g_2 , sous la forme d'un entier, comme indiqué ci-dessus.

Croisement

Le croisement de génome de 2 individus consiste à produire 2 nouveaux individus en découpant les 2 deux génomes au même endroit et échanger les gènes.

Dans notre implantation, on opère un croisement à 1 point, l'échange s'effectuant sur la moitié du génome.

Exemple pour les génomes g_1 et g_2 :

g_1	c	a	s	s	e	r
g_2	r	a	l	e	u	r

Le croisement produit 2 nouveaux génomes g_3 et g_4 :

g_3	c	a	s	e	u	r
g_4	r	a	l	s	e	r

Travail à effectuer

1. Écrire la fonction `croiser` qui prend en paramètre 2 génomes g_1 et g_2 et renvoie 2 nouveaux génomes correspondants au croisement de g_1 et g_2 , comme indiqué ci-dessus.

Mutation

La mutation consiste à changer la valeur d'un gène, selon une probabilité p donnée.

La mutation symbolise la substitution d'un caractère par un autre dans la liste des caractères disponibles.

Travail à effectuer

1. Écrire la fonction `muter` qui prend en paramètre un génome g et une probabilité p et a pour effet de bord de muter chaque gène de g selon la probabilité p .