Les algorithmes génétiques

Présentation

- Les *algorithmes génétiques* font partie des algorithmes évolutionnistes.
- S'inspirent des mécanismes de sélection naturelle pour approcher une solution d'un problème *d'optimisation*.
- S'appliquent bien dans le cas où l'espace des solutions est grand.

Darwin

- *De l'origine des espèces*
- les individus les mieux adaptés à leur environnement survivent et se reproduisent davantage en transmettant les variations utiles à la survie.
- L'espèce évolue en s'adaptant à son environnement.
- C'est le principe de *sélection naturelle*

Sélection naturelle

Trois principes :

- Variation : les individus diffèrent les uns des autres et d'une génération à l'autre ;
- Adaptation : les individus les mieux adaptés survivent et se reproduisent davantage ;
- Hérédité : les caractéristiques sont héréditaires et se transmettent de génération en génération.

Principes

Problèmes et solution

On dispose d'un problème d'optimisation :

$$\max_{s \in R} \{f(s)\}$$

- R est l'espace des solutions *réalisables*
- f est la fonction *objectif*, où encore :
    - fonction d'évaluation,
    - fonction d'adaptation,
    - fonction de fitness.

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 (valeur de f plus élevée) lors de la production de la génération suivante ;
- Après n générations, on choisit un individu dans la population.

génome

- Les individus sont représentés par leur génôme : suite finie de k symboles;
- Ces symboles représentent les caractéristiques d'un individu, c'est-à-dire un gène ;

Chercher la solution optimale consiste à chercher une séquence de symboles optimale.

Idée à respecter : plus l'adaptation est élevée, plus l'individu est proche de la solution.

Opérateurs génétiques

Il existe trois opérateurs génétiques :

la *sélection*

Choix des individus d'une population qui vont survivre à la génération suivante :

- Les individus les mieux adaptés ont plus de chance de survivre ;
- Méthode du *tournoi*.

le *croisement*

Combine les gènes de deux individus pour en produire deux nouveaux :

- croisement **1 point** : découpe des deux génome au même endroit, puis échange ,
- croisement **2 points** : découpe chacun des génomes en trois portions , on échange les gènes des parties médianes.

La *mutation*

- Mutation *aléatoire* des caractéristiques d'un individu ;
- Réalisée avec une probabilité faible ;
- Permet de faire apparaitre dans une population de nouvelles caractéristiques

L'algorithme génétique

Notre algorithme travaillera avec une population constante d'une génération à l'autre.

Entrées

- la fonction de *fitness* du problème ;
- le nombre d'individus n pair ;
- le nombre de génération g ;

- la probabilité de mutation p.

L'algorithme

Algorithme : Algorithme génétique

Entrées :
- $fitness$: une fonction de fitness,
- n : le nombre d'individus,
- g : le nombre de générations,
- p : la probabilité de mutation.

Sorties : l'individu le mieux adapté
C.U : n est pair

début

1 population ← n individus créés aléatoirement.
2 calculer pour chaque individu de population son degré d'adaptation.
répéter g **fois**
3 next_gen_1 ← sélectionner par tournoi $\frac{n}{2}$ individus de la population
4 next_gen_2 ← créer $\frac{n}{2}$ individus par croisement entre individus de population
5 next_gen ← next_gen_1 + next_gen_2
6 faire muter chacun des individus de next_gen
7 calculer pour chaque individu de next_gen son degré d'adaptation
8 meilleurs ← les 5 individus les mieux adaptés de population
9 retirer de next_gen les 5 individus les moins adaptés
10 population ← neilleurs + next_gen
fin
11 solution ← individu le mieux adapté de population
retourner *solution*

fin

Sélection par tournoi

- La sélection par tournoi dans une population de taille n consiste à choisir aléatoirement deux individus de la population pour ne garder que l'individu le mieux adapté des deux.
- On répète ce processus de sorte que chaque individu est choisi exactement une fois. On obtient donc $\frac{n}{2}$ individus à la fin de cette phase.

Croisements

- Croisement à un point
- On choisit aléatoirement deux individus de la population pour construire deux nouveaux individus
- On applique ensuite un tournoi entre les deux individus produits pour ne garder que le meilleur
- On répète ce processus de sorte que chaque individu est choisi exactement une fois. On obtient donc $\frac{n}{2}$ individus à la fin de cette phase.

Mutations

- Pour chacun des individus considérés, il y a une probabilité p que chacune de ses caractéristiques mute. C'est-à-dire que la valeur de cette caractéristique est modifiée aléatoirement. L'individu est donc remplacé par sa *version mutée*.
- À la fin de la population la taille de la population est la même.

Programmation

Plusieurs classes :

- l'algorithme génétique,
- le problème traité et
- les individus.

les problèmes

Comme indiqué, les classes pour représenter les problèmes devront partager certaines méthodes. Nous suggérons les méthodes suivantes comme méthodes communes aux problèmes. Evidemment, il peut exister d'autres méthodes spécifiques à chacun des problèmes. Notamment la manière de calculer l'adaptation d'un individu variera.

les individus

Les classes pour les individus devront partager certaines méthodes. Nous suggérons les méthodes suivantes comme méthodes communes aux individus. Evidemment, il peut exister d'autres méthodes spécifiques à chacun des individus, et notamment, les valeurs des génomes vont différer.

Liste de problèmes

Problème 1 : Calcul de la valeur maximale d'une fonction

Problème : trouver dans un intervalle $[x_{\min}; x_{\max}]$ une valeur de x pour laquelle $f(x)$ est maximale.

Problème 2 : Recherche d'un message "secret"

Cette fois, il ne s'agit pas à proprement parler d'un problème à résoudre, puisqu'on en connaît la solution au départ... Ce second exemple a pour objectif d'illustrer que les algorithmes génétiques peuvent trouver une solution dans un très grand espace de recherche.

Le problème est celui de faire trouver une chaîne de caractères à un algorithme génétique. On suppose cette chaîne construite 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 chaine *secret* = 'ceci est un secret a retrouver'.

Problème 3 : Recherche d'un chemin dans un labyrinthe

Soit un labyrinthe défini sur 7×7 cases.

```
+-+-+-+-+-+-+-+
|       |     |
+ + +-+ + +-+ +
```

```
| |     | |   |
+ + +-+ + + + +
|     |   | | |
+ +-+ +-+-+ + +
| |     |     |
+ + +-+-+ +-+-+
| |     |     |
+ +-+-+ + +-+ +
| |   |   |   |
+ + + +-+-+ +-+
|   |         |
+-+-+-+-+-+-+-+
```

Le problème consiste à trouver dans un tel labyrinthe de taille N (ici $N = 7 \times 7 = 49$) donné, un chemin entre la case d'entrée située en haut à gauche et celle de sortie située en bas à droite.

Pour simplifier on considère que les largeur et hauteur du labyrinthe sont identiques.

Problème 4 : le TSP

Utiliser un algorithme génétique pour approcher une solution du Problème du voyageur de commerce.