TP - Programmation Robotique - L1

Yann Chevaleyre & Pierre Boudes

November 25, 2015

Aujourd'hui nous allons traiter le problème de l'**évitement d'obstacle** d'une façon plus subtile qu'auparavant: nous allons créer un module d'évitement d'obstacle optimisé par **évolution artificielle**.

On utilisera le package `robosim` pour ce TP.

1 Introduction

L'évolution artificielle est une famille d'algorithmes d'optimisation "black-box" allant des algorithmes génétiques au stratégies évolutionnaires. Ils sont tous basés sur une perturbation aléatoire des solutions candidates dans le but d'obtenir une solution meilleurs.

2 Algorithme Evolutionnaire 1+1-ES

Dans cette section, vous implémenterez l'algorithme très simple d'optimisation nommé "1+1-ES" (voir `https://en.wikipedia.org/wiki/Evolution_strategy`). Pour tester son fonctionnement, vous l'utiliserez pour trouver le maximum d'une fonction jouet `f(L)` ou `L` est une liste.

1. Créez une fonction `f2d(L)` qui prend en paramètre une liste à deux éléments, et renvoie la valeur $1.0 - |L[0] - 0.2| - |L[1] - 0.7|$
2. Créer une fonction `optim(f,N,niter)` qui prenne en paramètre une fonction `f`. Ensuite, `N` doit désigner l'arité de `f`, et enfin le nombre d'itération pour l'optimisation.
   Cette fonction doit implémenter l'algorithme suivant:
   (a) Soit $T = [0.5, ..., 0.5]$, un tableau de N éléments ne contenant que des valeurs 0.5
   (b) Répéter *niter* fois:
      i. Générer un tableau T' à partir de T, de telle sorte que pour chaque $i \in \{0..N-1\}$, on ait $T'[i] = T[i] + random.gauss(0.0, 1.0)$
      ii. si $f(T') > f(T)$ alors
         A. $T \leftarrow T'$

3. Testez l'algorithme pour trouver le maximum de la fonction `f2d`.
   Par exemple, on appellera la fonction `optim(f2d,2,100)`.

4. Modifiez l'algorithme pour que les nombres de la liste soient tous positifs.

5. retestez cet algorithme

3 Evitement d'obstacle

Lancez le simulateur avec `from robosim import *; init()`

On veut créer un comportement tel que le robot atteigne le plus vite possible la position en bas à droite de l'écran.

PS: La fonction `telemetre()` renvoie la distance à l'obstacle droit devant le robot.

1. créez une fonction `teleangle(a)` qui tourne le robot sur lui-même d'un angle de `a`, lance `telemetre()`, puis revient à son orientation initiale en tournant de `-a`.

2. créez une fonction `teleliste()` qui renvoie une liste de distances aux obstacles situés en face du robot et à 60 et -60 degrée par rapport à la direction du robot

3. créez une fonction `evitement(x1,x2,x3,x4,x5)` qui fasse cela

   (a) si `teleangle(0) < x1` alors `tg(180)`

   (b) sinon si `teleangle(-60) < x2` alors `td(x3)`

   (c) sinon, si `teleange(60) > x4` alors `tg(x5)`

   (d) `avance(5)`

4. Créez une fonction `test_evitement(x1...x5)` qui fasse cela:

   (a) placer le robot en position 144, 112 avec la méthode `set_position`

   (b) répéter

      i. `evitement(x1...x5)`

   (c) tant que le robot n'est pas bloqué par un obstacle, et pour au maximum 150 itération

   (d) A la fin, renvoyer la distance du robot au bas de l'écran (position 512,512). Pour cela, vous utiliserez la fonction `position()`

5. Essayez `test_evitement` avec différents jeux de paramètres. Entre autres, essayez `test_evitement(5,20,20,20,20)`.

6. Par optimisation, trouvez le meilleur jeu de paramètres pour `test_evitement`

7. Si possible, trouvez d'autres algorithmes d'évitement d'obstacle plus malins à optimiser.