TP 7 - Programmation Robotique - INFO2

Yann Chevaleyre & Pierre Boudes

October 28, 2016

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(x)` ou `x` est un vecteur (représenté sous forme de tableau numpy ou de liste).

1. Créez une fonction `f2d(x)` qui prend en paramètre un vecteur `x` à deux dimensions, et renvoie la valeur $1.0 - |x_0 - 0.2| - |x_1 - 0.7|$
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) Soit *bruitGaussien*(), une fonction qui renvoie un nombre aléatoire tiré d'une loi normale de déviation standard égal à 0.3
   (c) Répéter *niter* fois:

i. Générer un tableau T' à partir de T, de telle sorte que$T'[i] = T[i] + bruitGaussien()$
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 qu'à partir de l'itération $\frac{niter}{2}$, la déviation standard soit de 0.01. Pourquoi faire cela ?

5. retestez cet algorithme

3 Evitement d'obstacle paramétré

On réutilisera la stratégie d'évitement d'obstacle implémentée lors d'un TP précédent. Il s'agissait d'une combinaison linéaire avec des coefficients. Ce sont ces coefficients que nous ferons évoluer avec l'algorithme 1+1-ES.

Supposons qu'on utilise le télémètre pour connaitre la distance des obstacles orientés aux angles $[90, 45, 0, -45, -90]$. Soit $d_1 \dots d_5$ la liste de ces distances. Soit $v_1 \dots v_5$ les vecteurs allant du centre du robot ces 5 points d'impact du télémètre. Ces vecteurs devront être représentés dans le plan du robot.

On cherchera à calculer 5 coefficients $w_1 \dots w_5$ tels que le vecteur $V = v_1 \times w_1 + v_2 \times w_2 + \dots + v_5 \times w_5$ décrive la direction vers laquelle le robot doit s'échapper.

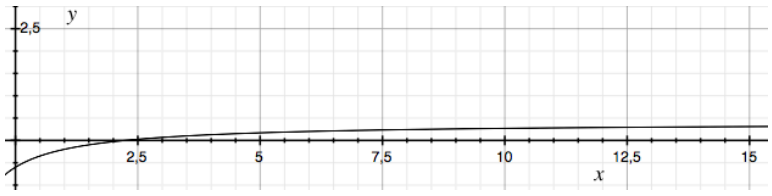
Chaque coefficient w_i sera calculé à partir de la norme d_i du vecteur v_i de cette façon:$w_i = \frac{g(d_i)}{d_i}$.

La fonction g sera définie ainsi:

$$g(d) = \alpha_0 - \frac{1}{1 + d \times \alpha_1}$$

On voit que cette fonction dépend des paramètres $[\alpha_0, \alpha_1]$. Selon la valeur de ces paramètres, les vecteurs auront une rôle d'attracteurs ou de répulseurs.

Voici par exemple la fonction d affichée avec $\alpha_1 = 0,67$ et $\alpha_0 = 0,4$. On voit qu'ici le poids sera positif pour $d > 2.5$ et négatif sinon. Donc les obstacles situés à plus de 2.5seront attractifs, et les autres répulsifs.



C'est donc ces deux paramètres $[\alpha_0, \alpha_1]$ que nous chercherons à optimiser.

1. Ecrire une fonction `calculeVecteurs` qui renvoie les 5 vecteurs $v_1 \dots v_5$.
2. Ecrire la fonction `g(d,theta0,theta1)`
3. Ecrire la fonction `VecteurEvitement` qui calcule le vecteur V.
4. Ecrire une fonction `orienteEvitement` qui oriente le robot dans la direction du vecteur V
5. Ecrire la fonction `EvalueTheta([theta0,theta1])` qui fasse ainsi:
   (a) au début, le robot est remis à sa position et orientation initiale
   (b) Répéter 150 fois:
      i. orienter le robot avec `orienteEvitement`
      ii. avancer de 5 points
   (c) x,y = position()
   (d) Renvoyer $x + y$
6. Optimiser la fonction `EvalueTheta` à l'aide de l'algorithme 1+1-ES. Le résultats sera un robot qui s'approche le plus du coin en bas à droite.

4 Evitement d'obstacle et atteinte d'un but

Proposer une solution pour qu'en plus de l'évitement d'obstacle, le robot se dirige vers un but (par exemple en bas à droite). Optimiser l'ensemble.