Université de Nantes — UFR Sciences et Techniques
Master informatique parcours “optimisation en recherche opérationnelle (ORO)”
Année académique 2019-2020

Distanciel

Métaheursitiques

Nilson Toula

15 octobre 2019

Livrable du devoir distanciel 1 : Particle Swarm Optimisation (PSO)

2 - Application

Partant de $\vec{v}_A(0) = \vec{v}_B(0) = 0$, calculer les 3 premières itérations de l'algorithme PSO avec ces éléments.
Rapporter les valeurs respectives de $\vec{x}_i(t), \vec{v}_i(t), \vec{p}_i(t), \vec{g}(t)$.

t	$\vec{x}_a(t)$	$\vec{v}_a(t)$	$\vec{p}_a$	$f(\vec{x}_a(t))$	$\vec{x}_b(t)$	$\vec{v}_b(t)$	$\vec{p}_b$	$f(\vec{x}_b(t))$	$g(t)$
0	0.5	0.0	0.5	1.0	1.25	0.0	1.25	2.25	1.25
1	0.9048	0.4048	0.5	0.8642	1.25	0.0	1.25	2.25	1.25
2	1.1609	0.2561	0.5	-0.0938	1.25	0.0	1.25	2.25	1.25
3	1.1423	-0.0186	0.5	-0.109	1.25	0.0	1.25	2.25	1.25

TABLE 1 – Valeur des variables en fonction des itérations

Quelle est la meilleure valeur de f(x) au terme de l'algorithme et pour quelle valeur de x ?
— La meilleur valeurs de f(x) au terme de cet algorithme est : $f(\vec{x}_b(t)) = 2.25, \forall t \in \{0, 1, 2, 3\}$

3 - Production à rendre

Produire un court rapport sous latex qui présente de manière claire, argumentée et pédagogique vos résultats.

1 - Théorie

La métaheuristique PSO se base sur un mimétisme des essaims d'oiseaux ou des bancs de poissons.
Pour se faire on place un certain nombre de particules qui interagiront les unes avec les autres.

A chaque itération t de notre algorithme les particules changent leurs positions $\vec{x}_i(t)$ d'une certaine valeur que l'on appellera vélocité.
Celle-ci est influencé par plusieurs paramètres, parmi ceux-ci trois paramètres nous intéresse :
- Le paramètre social : qui représente la meilleurs valeur connue par l'essaim : $\vec{g}(t)$
- Le paramètre personnel : qui représente la meilleur valeur connu par la particule : $\vec{p}_i(t)$
- La vélocité lors de la précédente itération : $\vec{v}_i(t-1)$

Plus généralement, la valeur de la vélocité à une itération t+1 est :
$v_i(t+1) = w \cdot \vec{v}_i(t) + r_1 \cdot c_1(\vec{p}_i(t) - \vec{x}_i(t)) + r_2 \cdot c_2(\vec{g}_i(t) - \vec{x}_i(t))$

2 - Initialisation

Ici, notre essaim ne se compose que de deux particules A et B évoluant dans un espace unidimensionnel et ont pour but de maximiser la fonction :

$$f(x) = x \cdot sin(10 \cdot \pi \cdot x) + 1$$

Pour se faire, on à initialiser nos particules avec les valeurs suivantes :
$\vec{x}_A(0) = 0.5$
$\vec{x}_B(0) = 1.25$
$\vec{v}_B(0) = \vec{v}_B(0) = 0$

De ces position, on en déduit leurs valeurs :
$f(\vec{x}_A(0)) = 1.0000000000000002$
$f(\vec{x}_B(0)) = 2.25$

Ainsi que la meilleure position de l'essaim :
$\vec{g}(0) = \vec{v}_B(0) = 1.25$

Finalement, étant à l'initialisation, les meilleurs valeurs des particules seront leurs valeurs initiales :
$\vec{p}_A(0) = \vec{x}_A(0)$; $\vec{p}_B(0) = \vec{x}_B(0)$

3 - Itérations

Par la suite, les positions de $\vec{x}_A$ seront :
$\vec{x}_A(1) = 0.9047959999999999 \Rightarrow f(\vec{x}_A(1)) = 0.8640758869635389$
$\vec{x}_A(2) = 1.160872187984 \Rightarrow f(\vec{x}_A(2)) = -0.0934984950928357$
$\vec{x}_A(3) = 1.1422906699066246 \Rightarrow f(\vec{x}_A(3)) = -0.10904074273173747$

Ces position ne correspondant pas à des valeurs améliorantes pour $\vec{g}(t)$, alors B ayant une vélocité initiale nulle n'effectue aucun déplacement lors de toutes les itérations. Ainsi, il n'y aura aucune modification des autres variables tels que $\vec{g}$, $f\vec{x}_B$ ou $\vec{v}_B$.

4 - Code source

```
nb = 4

#initialisation variables
x_a = zeros(Float64 , nb) ; v_a = zeros(Float64 , nb) ; p_a = zeros(Float64 , nb)
f_a = zeros(Float64, nb)
x_b = zeros(Float64 , nb) ; v_b = zeros(Float64 , nb) ; p_b = zeros(Float64 , nb)
f_b = zeros(Float64, nb)
g = zeros(Float64 , nb)

#intiialisation valeurs
rng = [0.679923 0.269864 0.88061 0.722799 0.566828 0.44927 0.935181 0.1669 0.29829
    0.670977 0.369223 0.927932]
x_a[1] = 0.5 ; v_a[1] = 0
x_b[1] = 1.25 ; v_b[1] = 0
w = 1 ; c1 = 2 ; c2 = 2

function swarm(x_a,v_a,p_a,x_b,v_b,p_b,g,rng,w,c1,c2)
    cpt = 1
    for t in 1:nb
        push!(f_b, round(f(x_b[t]); digits = 4))
        if t != 1
            v_a[t] = velocite(w, v_a[t-1], rng[cpt], c1, p_a[t-1], x_a[t-1], rng[cpt +
                1], c2, g[t-1])
            cpt += 2
            v_b[t] = velocite(w, v_b[t-1], rng[cpt], c1, p_b[t-1], x_b[t-1], rng[cpt +
                1], c2, g[t-1])
            cpt += 2
            x_a[t] = position(x_a[t-1], v_a[t])
            x_b[t] = position(x_b[t-1], v_b[t])
        end
        f_a[t] = f(round(x_a[t];digits=4))
        f_b[t] = f(round(x_b[t];digits=4))
        p_a[t] = perso_max(x_a)
        p_b[t] = perso_max(x_b)
        g[t] = pos_max(x_a[t],x_b[t])
    end
    println("x_a: ", x_a) ; println("v_a: ", v_a) ; println("p_a: ", p_a)
    println("f_a: ", f_a)
    println("g: ", g)
    println("x_b: ", x_b); println("v_b: ", v_b); println("p_b: ", p_b)
    println("f_b: ", f_b)
end

function f(x)
    return(x*sin(pi*10*x)+1)
end

function velocite(w, v, rng1, c1, p, x, rng2, c2, g)
    return((w*v) + (rng1*c1*(p - x)) + (rng2*c2*(g - x)))
end

function position(x,v)
    return(x+v)
end

function pos_max(x1,x2)
    if f(x1)-f(x2) > 0
        return(x1)
    else
        return(x2)
    end
end
```

```
60
61  function perso_max(x)
62      max = f(x[1])
63      pos = 1
64      for i in 2:length(x)
65          if f(x[i]) > max
66              max = f(x[i])
67              pos = i
68          end
69      end
70      return(x[pos])
71  end
72
73  swarm(x_a, v_a, p_a, x_b, v_b, p_b, g, rng, w, c1, c2)
```

Code source de ce projet disponible via le lien:
https://github.com/sofros/Distanciel_Metaheuristique1