Projet de L2, Info 31 en Java
Optimisation stochastique : l'algorithme Jaya

October 18, 2016

Vous devez réaliser ce projet (Optimisation stochastique : l'algorithme Jaya) seul pour des problèmes 2D. Pour des problèmes 3D (par exemple, vous placez des sphères dans un tétraèdre), vous pouvez être troisau maximum. Remarquez que, en 2D (respectivement en 3D), le nombre d'inconnues peut être plus grand que 2 (respectivement, que 3).

1 La méthode Jaya

Le but est de minimiser une fonction $F : X \in \mathbb{R}^d \to \mathbb{R}$. F est décrite par une méthode qui sait évaluer $F(X)$ pour n'importe quel vecteur de d flottants.

L'algorithme Jaya (Victoire, en sanscrit) [2, 1] procède ainsi. Elle gère une population de n points de $\mathbb{R}$. Pour commencer, vous pouvez essayer $n = 100$. La population initiale de n points est soit composée de n points aléatoires dans une grande boîte de $\mathbb{R}^d$, soit obtenue en perturbant une approximation de solution $c \in \mathbb{R}^d$ (passer en paramètre le rayon, ou un vecteur R de d rayons, un pour chaque inconnue).

Voici une étape de Jaya: etape(P: un vecteur de n points de $\mathbb{R}$)

- calculer B le meilleur point de la population P : $B = \text{argmin}_{p \in P} f(p)$.
- calculer W le pire point de la population P : $W = \text{argmax}_{p \in P} f(p)$.
- définir deux matrices aléatoires diagonales D_1 et D_2 (il suffit de stocker le vecteur des termes de la diagonale). Les éléments de la diagonale de D_1 sont des nombres (pseudo-)aléatoires flottants compris entre 0.0 et 1.0. De même pour D_2.
- Pour chaque point $X \in P$, calculer :

$$X' := X + D_1\overrightarrow{WX} + D_2\overrightarrow{XB}$$

Remarquer que :

$$X' := X+D_1(X-W)+D_2(B-X) = (I+D_1-D_2)X+(-D_1W+D_2B) = DX+K$$

avec D une matrice diagonale, et K un vecteur constants (lors de l'étape). Si $f(X') < f(X)$, alors remplacer X par X' dans la population P. Ainsi, la population ne peut pas se dégrader.

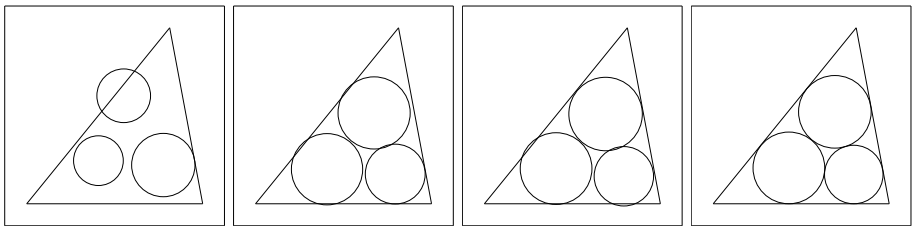

Figure 1: Trois cercles tangents entre eux et tangents aux cotés d'un triangle donné par ses trois sommets.

Note : le produit d'une matrice carrée diagonale $d \times d$ et d'un vecteur colonne se fait en $O(d)$.

Jaya enchaîne des étapes jusqu'à un nombre maximum m d'étapes fixé *a priori* (par exemple : $m = 500$), ou jusqu'à convergence (tous les points de la population sont inclus dans une boîte englobante de côté inférieur à un certain seuil ϵ, avec $\epsilon = 10^{-6}$ par exemple).

2 Utilisation de Jaya

Vous résoudrez de petits systèmes d'équations $G(X) = 0$, avec $X = (x_1, \dots x_d)$ et $G = (g_1, \dots g_d)$. Pour résoudre $G(X) = 0$, il faut minimiser soit $F(X) = \sum g_i(X)^2$, soit $F(X) = \sum |g_i(X)|$.

La figure 1 montre un exemple : un triangle ABC est donné ; il faut trouver les 3 cercles de centre (x_i, y_i) ($i \in \{1, 2, 3\}$) et de rayon r_i tels que les trois cercles soient inclus dans ABC, tangents entre eux, et tangents aux cotés de ABC.

Le programme suivant (en ocaml) définit la fonction F pour ce problème :

```
let pbm (xA,yA) (xB,yB) (xC,yC)=
let f xs=
(
assert (9=A.length xs);
match xs with
|       [| x1; y1; r1; x2; y2; r2; x3; y3; r3 |] ->
        let (x1ab,y1ab)=proj_pt_dte (x1,y1) (xA,yA) (xB,yB) in
        let (x1ac,y1ac)=proj_pt_dte (x1,y1) (xA,yA) (xC,yC) in
        let (x2bc,y2bc)=proj_pt_dte (x2,y2) (xB,yB) (xC,yC) in
        let (x2ab,y2ab)=proj_pt_dte (x2,y2) (xA,yA) (xB,yB) in
        let (x3bc,y3bc)=proj_pt_dte (x3,y3) (xB,yB) (xC,yC) in
        let (x3ac,y3ac)=proj_pt_dte (x3,y3) (xA,yA) (xC,yC) in
        (square ( dist_pt_pt (x1,y1) (x2,y2) -. (r1+.r2) ))
+.      (square ( dist_pt_pt (x1,y1) (x3,y3) -. (r1+.r3) ))
+.      (square ( dist_pt_pt (x2,y2) (x3,y3) -. (r2+.r3) ))
+.      (square ( dist_pt_pt (x1,y1) (x1ab,y1ab) -. r1 ))
```

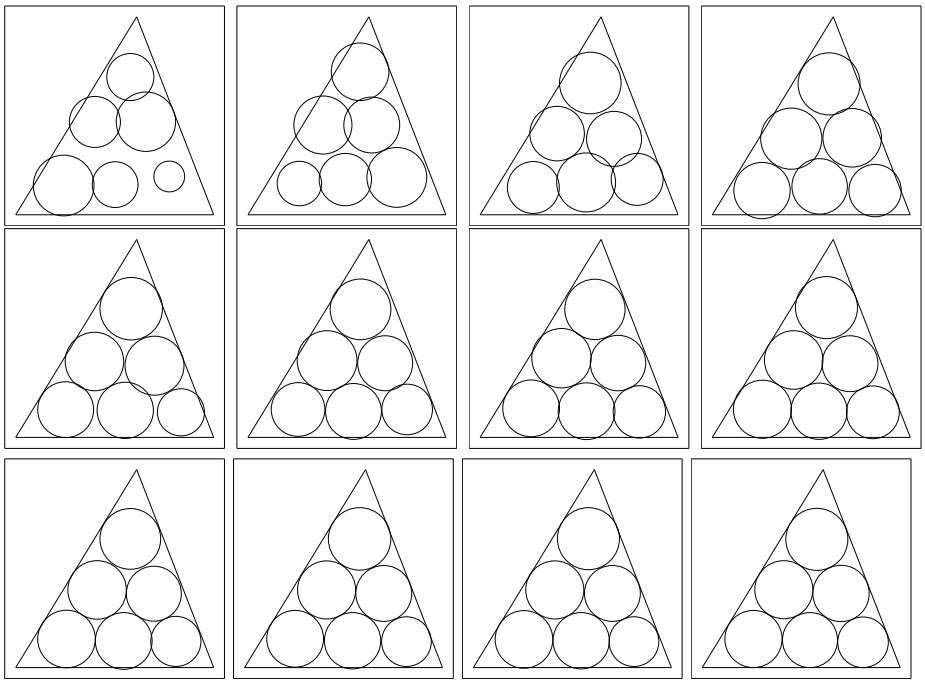

Figure 2: Six cercles tangents entre eux et tangents aux cotés d'un triangle donné par ses trois sommets.

```
+.      (square ( dist_pt_pt (x1,y1) (x1ac,y1ac) -. r1 ))
+.      (square ( dist_pt_pt (x2,y2) (x2ab,y2ab) -. r2 ))
+.      (square ( dist_pt_pt (x2,y2) (x2bc,y2bc) -. r2 ))
+.      (square ( dist_pt_pt (x3,y3) (x3bc,y3bc) -. r3 ))
+.      (square ( dist_pt_pt (x3,y3) (x3ac,y3ac) -. r3 ))
| _ -> failwith "bug" (* for compiler to be happy *)
) in f ;;
```

La fonction `proj_pt_dte (x1,y1) (xA,yA) (xB,yB)` calcule la projection orthogonale du point (x1,y1) sur la droite passant par les deux points (xA,yA) et (xB,yB). La fonction `dist_pt_pt (x,y) (x',y')` calcule la distance entre deux points. Il faudra au minimum écrire ces fonctions (ou méthodes) en Java.

Il faudra afficher la figure du meilleur point de la population à chaque étape.

La figure géométrique que vous construirez peut être différente.

La figure 2 montre un autre exemple, avec six cercles.

References

[1] R Rao. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of*

Industrial Engineering Computations, 7(1):19–34, 2016.

[2] R Venkata Rao and Vivek Patel. An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems. *Scientia Iranica*, 20(3):710–720, 2013.