

ALGORITHMES EVOLUTIONNISTES

(et autres heuristiques d'inspirations biologiques)

Pr. Fabien Moutarde
Centre de Robotique (CAOR)
MINES ParisTech (ENSMP)
PSL Research University

Fabien.Moutarde@mines-paristech.fr
<http://people.mines-paristech.fr/fabien.moutarde>

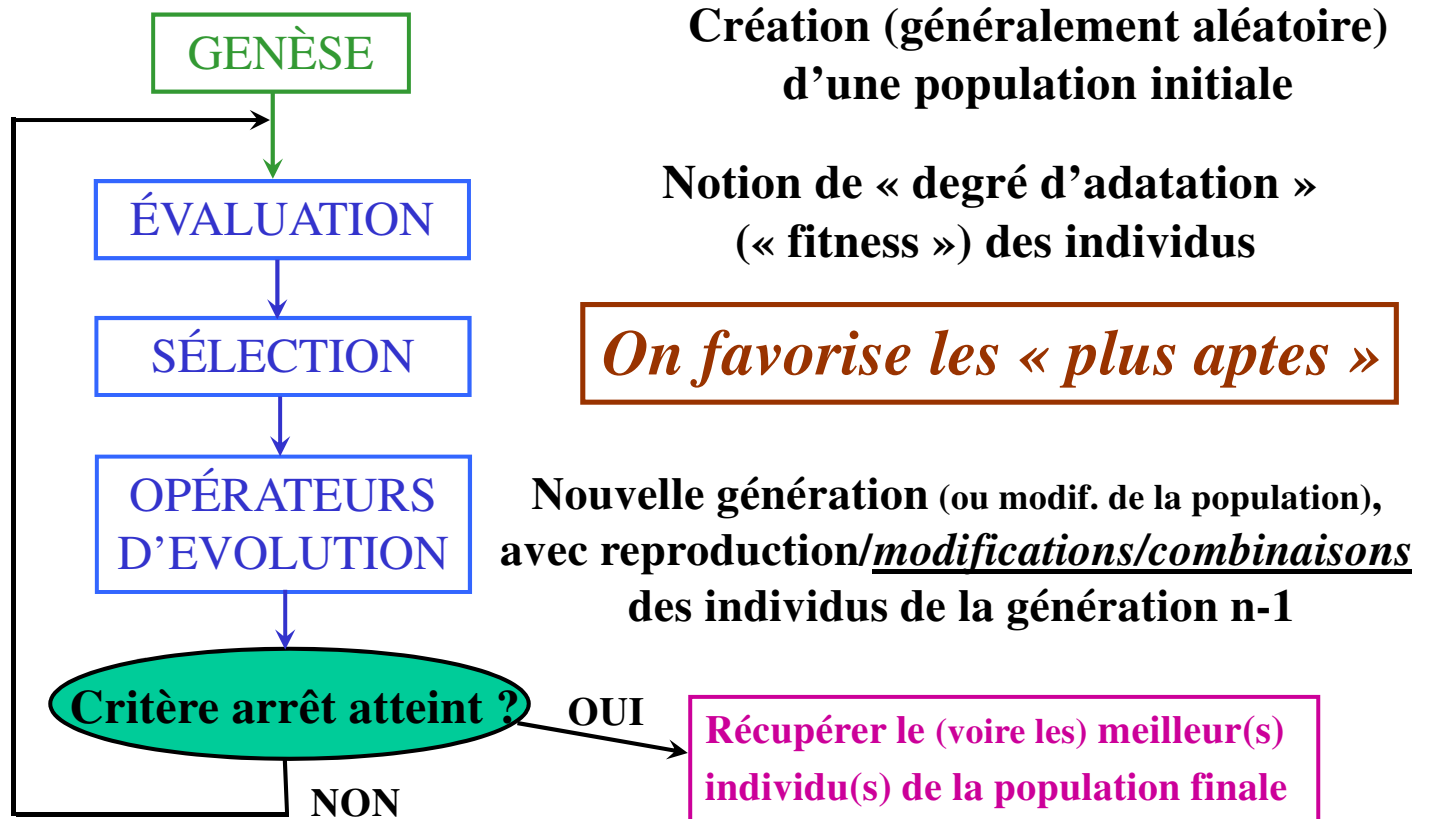
Algorithmes évolutionnistes Pr. Fabien Moutarde, CAOR, MINES ParisTech, PSL Fév.2017 1

Idée générale

Algorithmes d'optimisation
(au sens large, i.e. par forcément numérique)
conçus par *inspiration/imitation du processus d'évolution des espèces* (théorie Darwinienne)

- 1962 : John Holland → base des Algos génétiques binaires
- 1989 : David E. Goldberg → popularisation des AG

Sortes de recherche par « marche aléatoire dirigée »

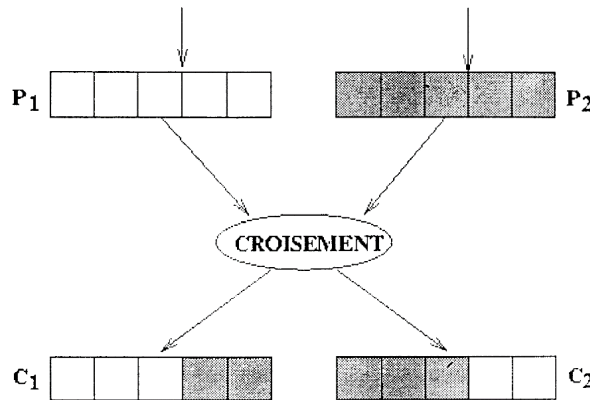


Algorithmes génétiques

- Cas particulier où chaque individu est représenté par un « génome » (séquence de bits, ou d'entiers, de réels,...)
- Les opérateurs d'évolution sont :
 - Mutation (remplacement aléatoire d'un, ou plusieurs, élément(s) du génome par une valeur aléatoire)
 - Croisement (recombinaison des génomes de 2 individus)

➔ Hyper-paramètres principaux :
taux de mutation et taux de croisement

- Paire de « parents » tirés au hasard (mais avec proba proportionnelle à leur fitness)



→ 2 « enfants » (ou 1 seul) combinant des morceaux des génomes des 2 parents (diverse façon de combiner...)

Quelques résultats théoriques

- **R. Cerf (1994) : avec « Théorie des schémas » + processus stochastiques (chaînes de Markov), a montré dans cas des AG à génomes binaires :**
 1. Quelle que soit la population initiale, l'AG converge (si la taille de population est « assez grande »)
 2. Si on ne conserve pas systématiquement à l'identique le meilleur individu d'une génération dans la suivante, alors la limite n'est PAS l'optimum !!
 3. Si au contraire on impose cet « élitisme », alors l'AG converge bien vers l'optimum GLOBAL
[mais on ne sait pas en combien de temps...]
 4. La convergence est assurée par les mutations, même sans croisement (le rôle de ceux-ci seraient donc surtout d'accélérer la convergence)

- Dans cas réels, la recherche peut « stagner » dans un minimum local, notamment à cause d'une trop grande uniformisation de la population. Solutions possibles :
 - Variation (automatique ou manuelle) du taux de mutation (et éventuellement de croisement)
 - Éliminations des doublons
 - « prohibition de l'inceste » (interdire à des individus trop similaires de se reproduire entre eux...)

Et aussi de nombreuses variantes :

- Croisement à N points
- Croisement « arithmétique » (cas génome NON binaire) :
au lieu que chaque gène vienne soit du père soit de la mère, faire pour chaque gène à la position i : $g_i(\text{fils}) = \alpha \cdot g_i(\text{mère}) + (1 - \alpha) \cdot g_i(\text{père})$
- « orgies » (reproduction à plus que 2...)
- Longueur génome variable, etc...

Applications des Algos Génétiques

**Tous les problèmes où il faut
« chercher une aiguille dans une botte de foin »**

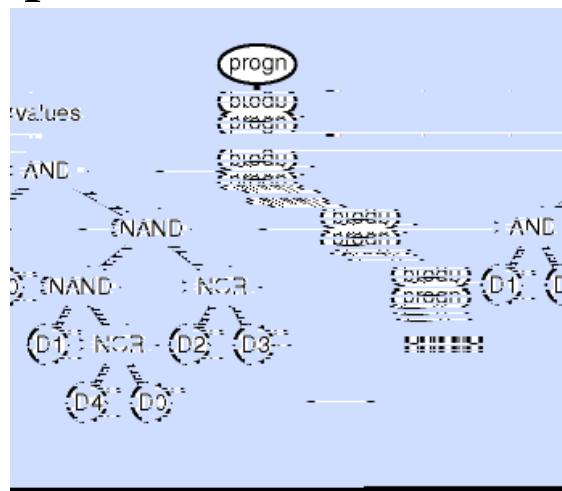
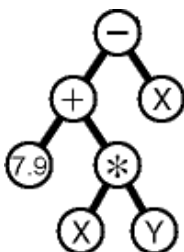
**(surtout si critère à optimiser n'est PAS différentiable
par rapport aux paramètres)**

- Optimisation numérique : trouver maximum d'une fonction, principalement dans cas implicite ou non-dérivable (sinon, méthodes type gradient possibles...)
- Optimisation combinatoire (style « problème du voyageur de commerce », « sac à dos », ...) [*mais pas toujours évident de trouver bonne représentation génétique pour gérer au mieux les éventuelles contraintes*]
- Apprentissage NON-supervisé

- **Génome** = poids des connexions $\{W_{ij}\}$ d'un réseau
 → permet de trouver un $\{W_{ij}\}$ optimal même si neurones NON dérivables, ou bien pour certaines types d'apprentissages NON-supervisés (où on ne connaît pas sorties désirées, ni critère mathématique qu'elles devraient vérifier, mais on peut évaluer « performance » par simulation)
- **Génome** = codage des paramètres d'apprentissage → AG pour trouver meilleur réglage de ceux-ci sur un problème, ou classe de problèmes
- **Génome** = codage de l'algo lui-même → recherche d'algos d'apprentissage plus efficaces (« evolution of learning »)

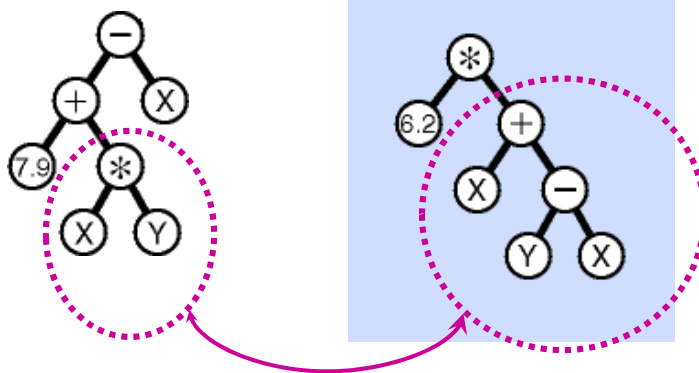
Programmation génétique (Genetic Programming, GP)

- Variante assez récente (Koza, 1992, Stanford) dans laquelle les gènes sont non plus des valeurs organisées en séquence, mais des opérations ou instructions organisées en arbre



- Permet de la « régression symbolique », voire de la génération automatique de programme...

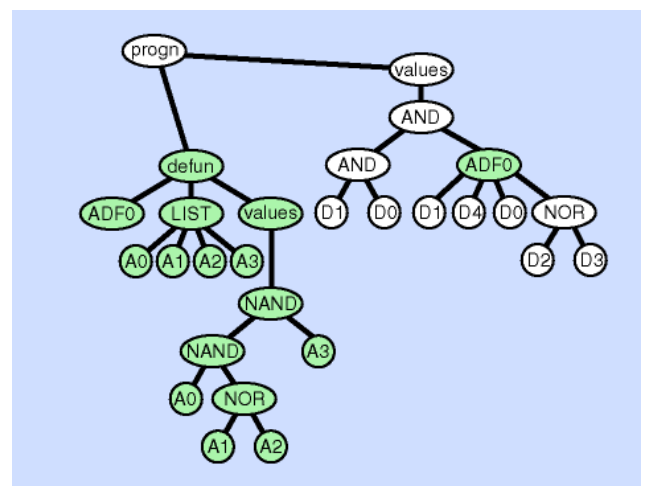
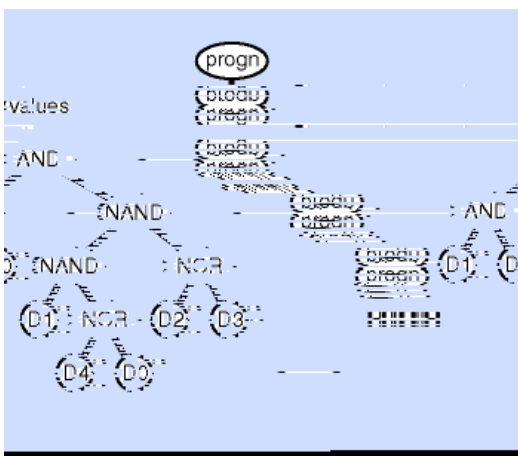
- **Mutation** : remplacement de sous-arbre par un autre généré aléatoirement
- **Croisement** → échange de sous-arbres (compatibles)



Principal problème avec GP : contrôler la croissance de la complexité des arbres (toujours le rasoir d'Ockham...)

Opérateurs spécifiques à la programmation génétique

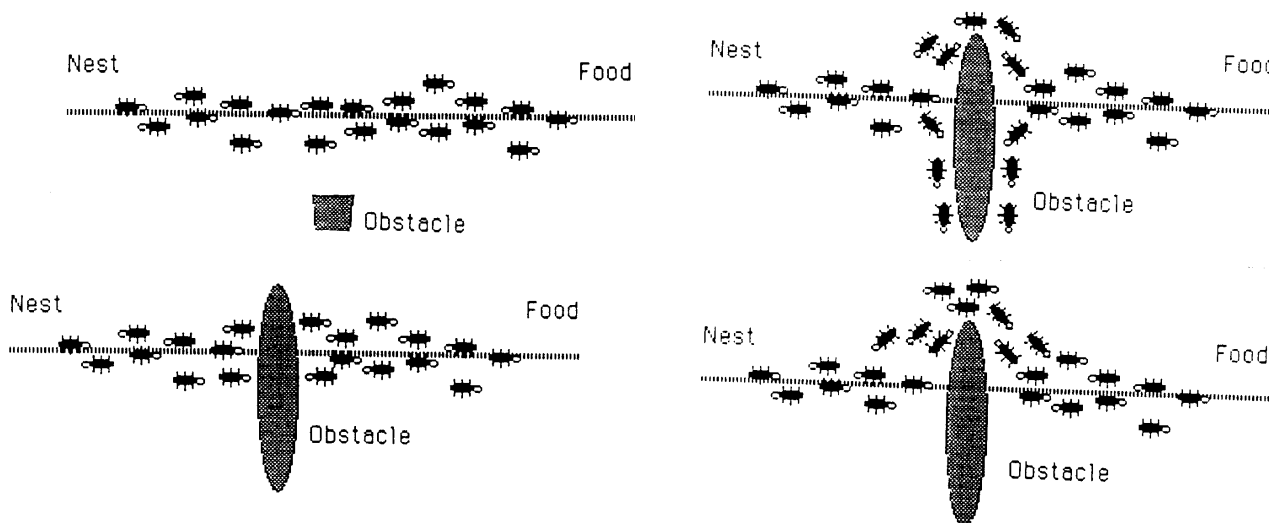
- **Création de fonction**



- **Création de boucle d'itération**
- **etc...**

Autres heuristiques d'inspiration biologique (1)

- Colonies de fourmis (Ant Colony Optimization, ACO)

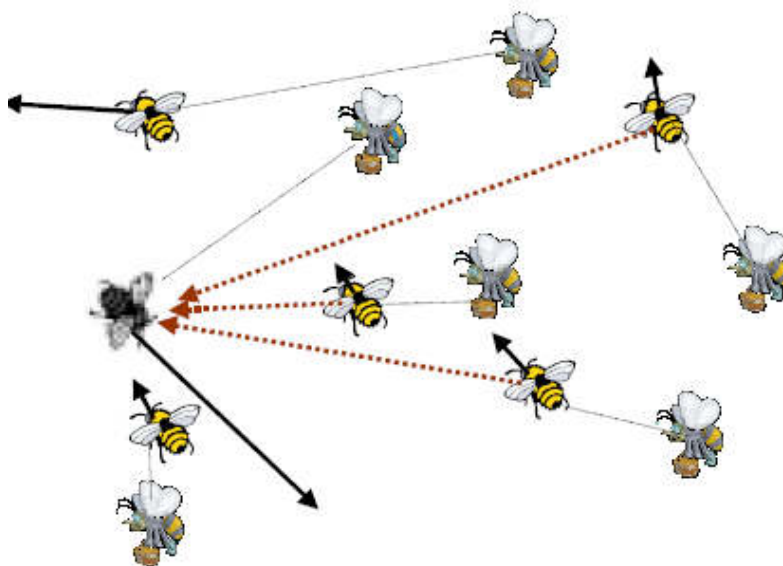


Dépose individuelle de « phéromone » et comportement « bête »
[suivre trace de max(parfum)]

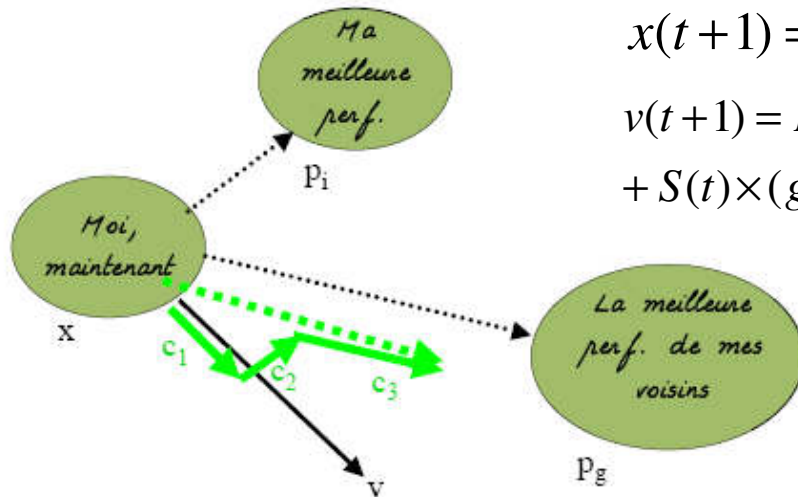
➔ **comportement collectif émergent « intelligent »**

Autres heuristiques d'inspiration biologique (2)

Optimisation par Essaim Particulaire, OEP (Particle Swarm Optimization, PSO)



- Inspiration : comportement d'essaim (abeilles, ...)
- Principe = agents à mémoire coopérant par communications



$$x(t+1) = x(t) + v(t+1)$$

$$v(t+1) = I(t) \times v(t) + N(t) \times (h_{best}(t) - x(t)) + S(t) \times (gh_{best}(t) - x(t))$$

$$I(t) = I \times u[0;1]$$

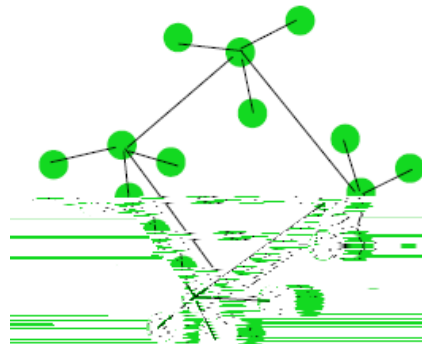
$$N(t) = N \times u[0;1]$$

$$S(t) = S \times u[0;1]$$

- A chaque itération, chaque particule se déplace selon un compromis entre les 3 tendances suivantes :
 - répéter son précédent mouvement ;
 - se diriger vers sa meilleure position passée ;
 - se diriger vers la meilleure position (passée) de son groupe d'informatrices.

Paramètres de l'OEP

- Nombre de particules
- Topologie des voisinages
 - complet
 - circulaire
 - « small world »



- **I (inertie), N (« nostalgie »), S (suivisme)**
analyse théorique de la dynamique $y(t+1) = Ay(t) + Bp$ avec $y = (x, v)$
→ préférer $I \sim 0,7$ et $N \sim S \sim 1.5$ pour convergence

OEP simplifié

- Dimensions traitées indépendamment
- Variables aléatoires remplacées par valeurs moyennes

• Notation $b = (b_1 + b_2) / 2$

$$p = \frac{b_1}{b_1 + b_2} p_1 + \frac{b_2}{b_1 + b_2} p_2$$

- Sans perte de généralité $c = 1, d = 1$

Reste :

$$v_{k+1} = av_k + b(p - x_k)$$

$$x_{k+1} = x_k + v_{k+1}$$

Analyse dynamique OEP (2)

• Ecriture matricielle

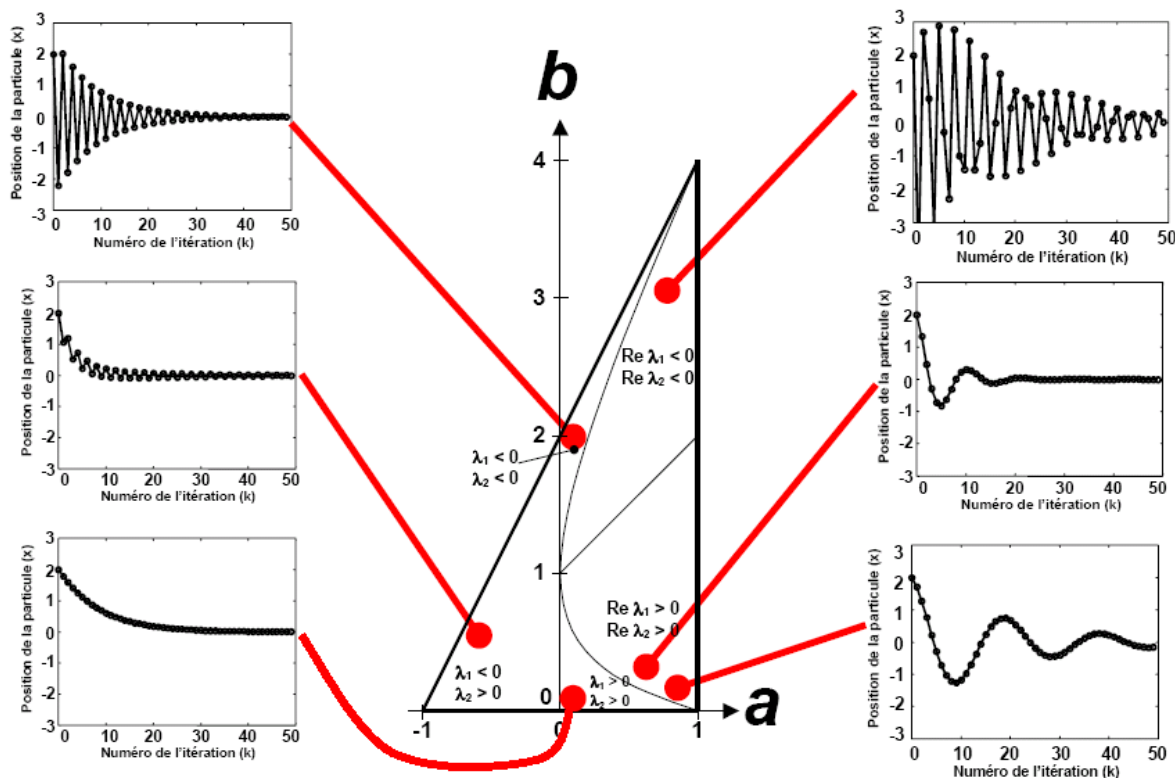
Notation : $y_k = \begin{bmatrix} x_k \\ v_k \end{bmatrix}$ $A = \begin{bmatrix} 1-b & a \\ -b & a \end{bmatrix}$ $B = \begin{bmatrix} b \\ b \end{bmatrix}$

$$y_{k+1} = Ay_k + Bp$$

Point d'équilibre : $v^{eq} = \begin{bmatrix} p \\ p \end{bmatrix}$

Equation caractéristique :

$$\lambda^2 - (a - b + 1)\lambda + a = 0$$



[Trelea, I. C., "The particle swarm optimization algorithm: convergence analysis and parameter selection", *Information Processing Letters*, vol. 85, no. 6, pp. 317-325, Mar.2003]