Identification de paramètres et optimisation
Cours de Master 2 STIM
2015-2016

Sébastien Adam

28 janvier 2016

Plan du Cours

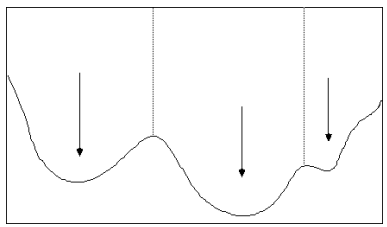
Principe général

Bilan sur les méthodes vues précédemment

1. La forme analytique de la fonction à optimiser doit être connue
2. Elle doit respecter des propriétés mathématiques de :
   - continuité/différentiabilité pour calculer le gradient voire le Hessien
   - convexité pour garantir la convergence

Or ...

- On ne dispose pas toujours la forme analytique de la fonction (Black-box and derivative-free optimization)
- La dérivée (a fortiori le Hessien) n'est pas toujours exprimable
- Les critères sont rarement convexes



Principe général

Les méthodes présentées aujourd'hui tentent de pallier ces problèmes

- Par :
  - L'exploitation de la valeur du critère en modèle boite noire
  - L'injection d'une part d'aléatoire contrôlé par le but
  - La manipulation d'ensembles de solutions (populations)
  - La coopération et/ou compétition entre solutions multiples
- On parle parfois d'heuristiques ou de métaheuristiques
- Quelques exemples pour ce cours :
  - Hill Climbing - la base
  - Recuit simulé - simulated annealing
  - Méthodes évolutionnaires - genetic algorithms - evolutionnary algorithms
  - Essaims particulaires - particle swarm optimization
- NB : un autre point commun entre ces méthodes : des analogies "marketing" parfois tirées par les cheveux

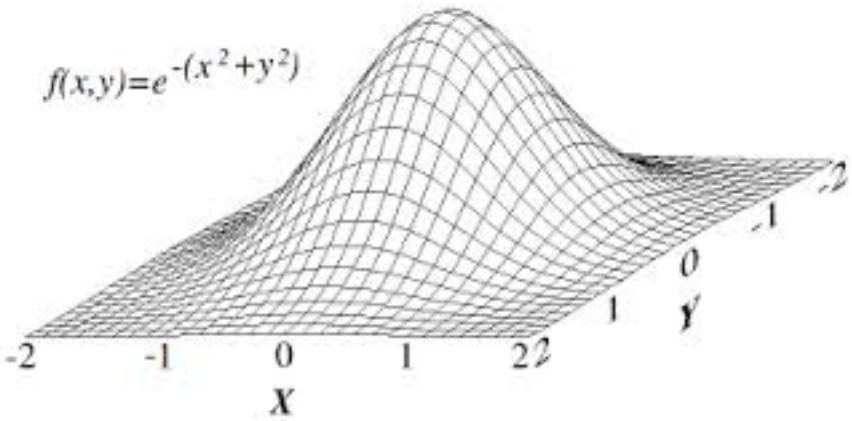
Plan du Cours

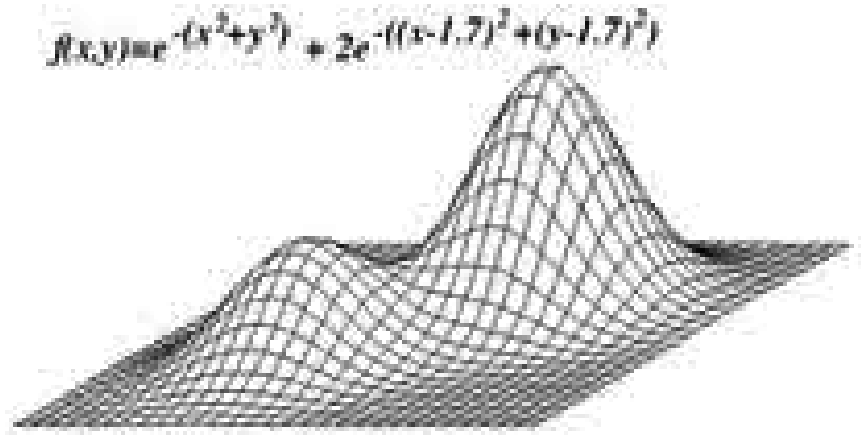
Principe général

Hill Climbing pour la résolution de Problème boîte-noire

- Principe très simple
  - Une solution initiale générée aléatoirement
  - Tant que critère d'arrêt non atteint
    - Générer un ensemble de voisins V (par exemple en modifiant la valeur d'un paramètre)
    - Sélectionner le meilleur des voisins
    - Si le voisin sélectionné est meilleur que la solution courante, le garder
    - Sinon, sortir
- Variantes :
  - Stochastic Hill Climbing : sélection aléatoire du voisin parmi ceux améliorant
  - shotgun hill climbing : multiples tentatives de Hill Climbing
- Défaut principal : extrema locaux

Recuit simulé





Recuit simulé

Principes

- La méthode tente de pallier les défauts d'un « simple » Hill Climbing
- Objectif : tolérance aux fonctions objectif multi-modales
- Analogie : le processus de recuit utilisé en metallurgie : le refroidissement naturel de certains métaux ne permet pas aux atomes de se placer dans la configuration la plus solide
  $\rightarrow$ On alterne cycles de refroidissement / réchauffage
- Historique :
  - Physique : Metropolis en 1953
  - Informatique : Kirkpatrick en 1983 (IBM) - Cerny en 1985
- Point commun : la notion de température qui diminue au cours du processus
  - Température haute : recherche aléatoire
  - Température basse : recherche locale
- Nombreuses variantes

Recuit simulé

Algorithme de principe (minimisation)

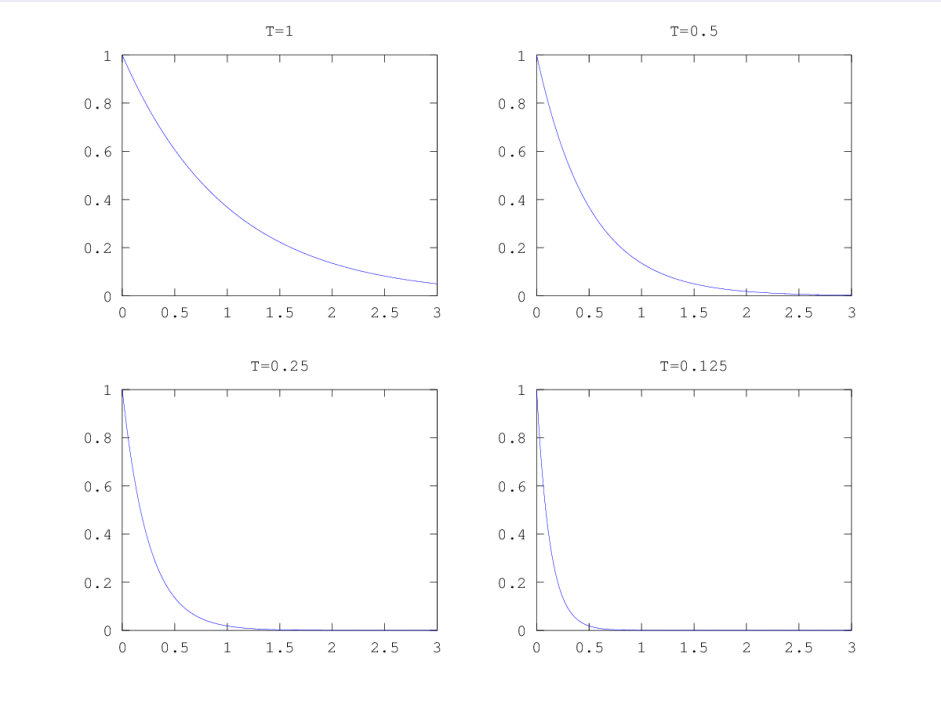
1. Choisir une initialisation θ_0 de θ
2. Choisir une température initiale $T_0 > 0$
3. Répéter
   - Effectuer un déplacement (souvent aléatoire) des paramètres tel que $\theta'_k = f(\theta_k)$
   - Calculer $\delta = J(\theta'_k) - J(\theta_k)$
   - Si $\delta \leq 0$ alors $\theta_{k+1} = \theta'_k$
   - Sinon
     - Si $Alea(0,1) < e^{\frac{-\delta}{T}}$ alors $\theta_{k+1} = \theta'_k$ [critère de Metropolis]
     - Sinon $\theta_{k+1} = \theta_k$
   - Adapter T
4. Jusqu'à condition de fin
5. Retourner la meilleure configuration trouvée

Recuit simulé

Remarques

Une solution qui améliore est toujours acceptée. Une solution qui dégrade est « parfois » acceptée :

- Si la température est élevée
- Si la dégradation est « raisonnable »



Recuit simulé

Nombreux choix

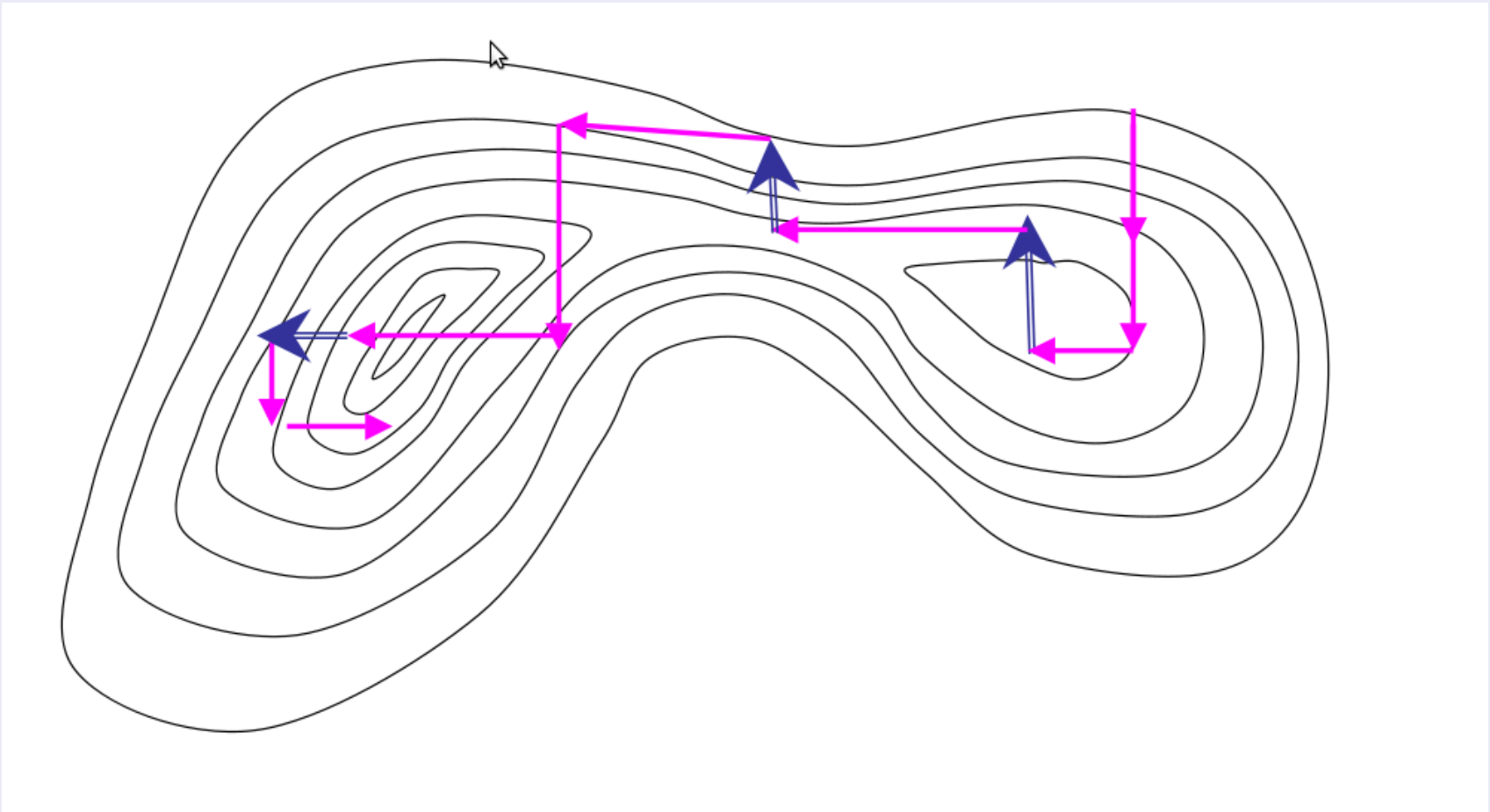
- Quel θ_0 ?
- Température initiale :
  $\rightarrow T_0 = f(J)$?
- Quels déplacements ?
- Quel *Cooling schedule* ?
  - Fixe
  - Par paliers (taux)
- Quel critère d'arrêt ?
  - Nombre d'itérations (naïf)
  - Taux d'acceptation
  - Valeurs de δJ
- Un « Outil » de réglage :
  Le taux d'acceptation

Exemple d'algorithme par paliers

- Engendrer une configuration initiale S0 ; S := S0
- T := T0
- Répéter
  - nb_moves := 0
  - Pour i := 1 à iter_palier
    - Engendrer un voisin S' de S
    - Calculer Δ = f(S') – f(S)
    - Si CritMetropolis(Δ, T), alors
      - S := S'; nb_moves := nb_moves + 1
  - acceptance_rate := i / nb_moves
  - T := T * coeff
- Jusqu'à <condition fin>
- Retourner la meilleure configuration trouvée

Recuit simulé

Illustration de principe sur des iso-critères

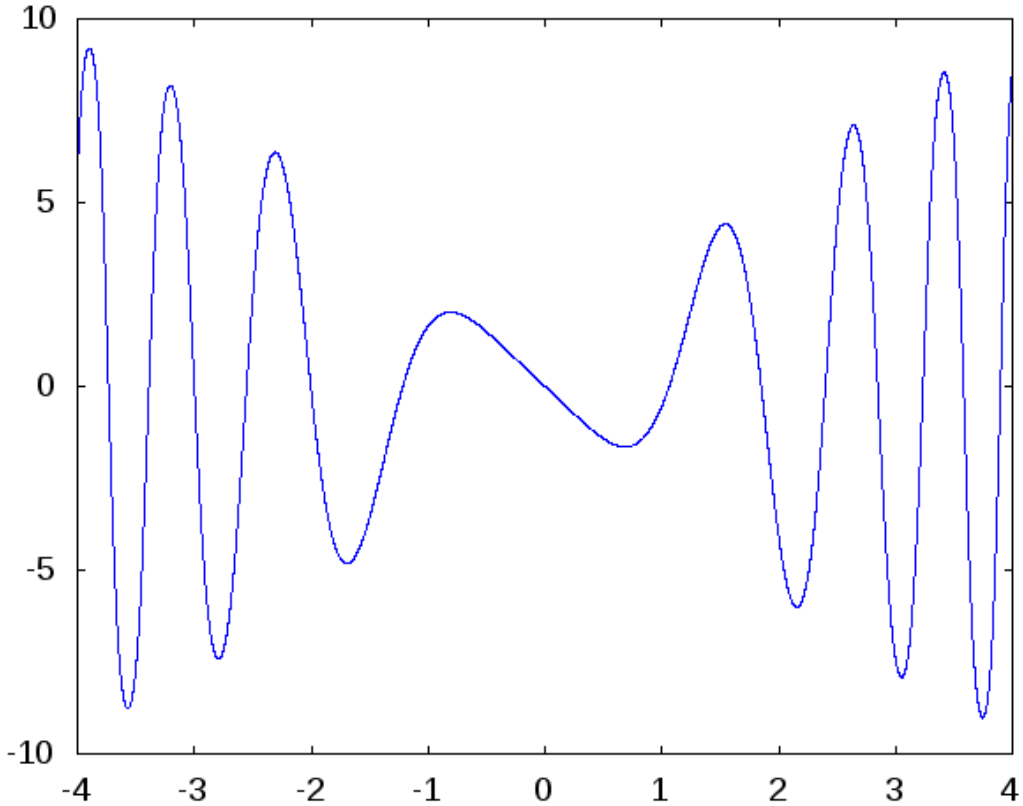


Recuit simulé

Démo 1D

$$f(x) = sin\left((0.3x)\, sin\left(1.3x^2 + 0.00001x^4 + 0.2x + 80\right)\right)$$

x : valeur rejetée, go valeur acceptée malgré dégradation, ro amélioration



Recuit simulé

Bilan

- Facile à implémenter (voir code)
- Propriétés de convergence
- Méthode importante historiquement
- Le point intéressant : introduction d'une part d'aléatoire
- Peu utilisée en pratique : difficile à "tuner"

Plan du Cours

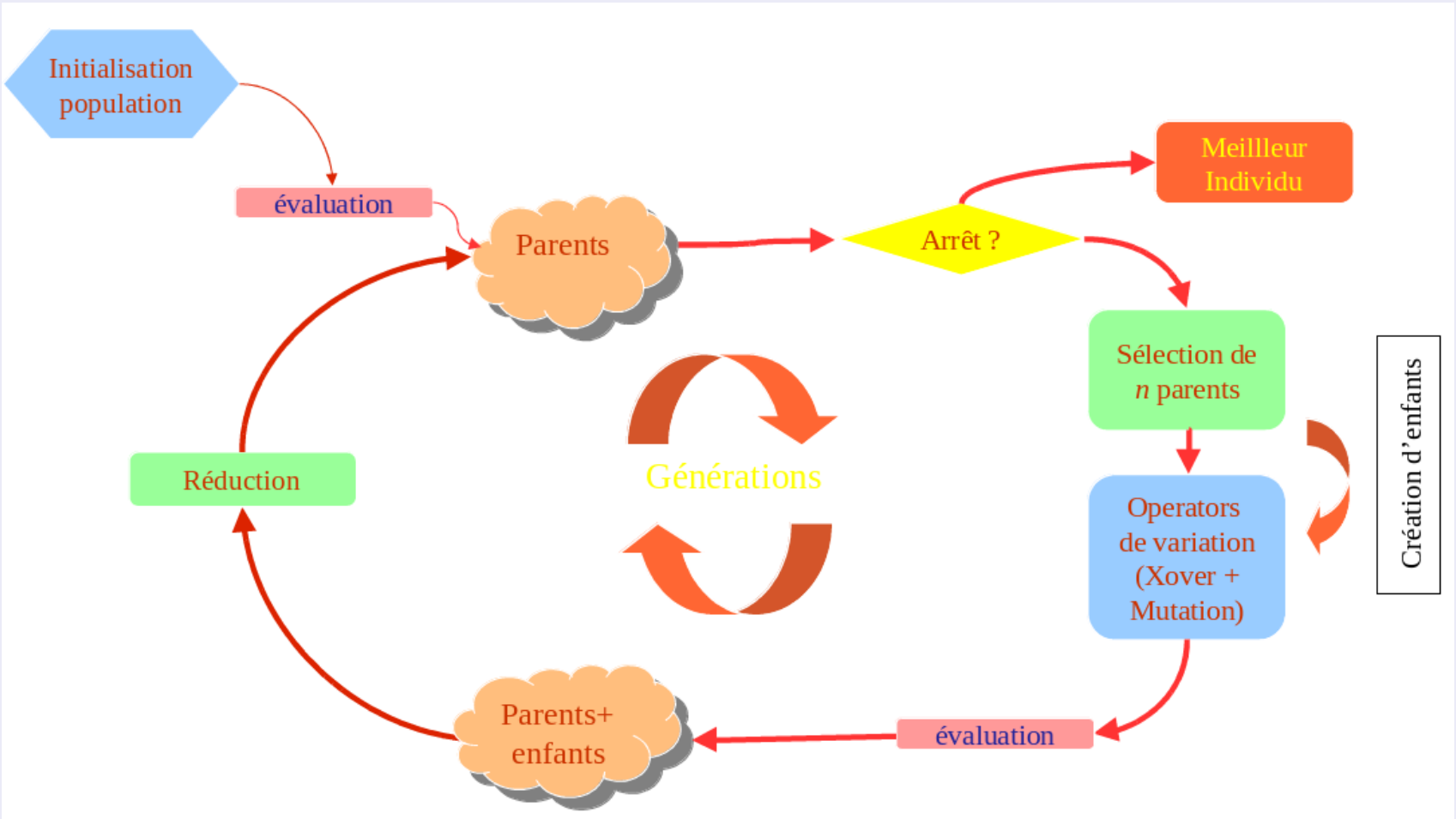
Algorithmes évolutionnaires

Principes

- Objectifs :
  - Minimiser itérativement un critère $J(\theta)$ dépendant de plusieurs paramètres
- Historique :
  - Premiers tests d'EA : 1957, dès la sortie des ordinateurs commerciaux
  - Regain d'intérêt dans les 70's, puis applications dans les 90's
- Analogie
  - Sélection naturelle dans les populations d'individus : Darwin
- Points clefs
  - Population de solutions potentielles
  - Pas d'hypothèse de dérivabilité du critère
  - Modifications aléatoires des solutions mais sous contrôle du but (adaptation)
- Un petit exemple simple combinant RN et AG en vidéo

Algorithmes évolutionnaires

Schéma général d'un algorithme évolutionnaire



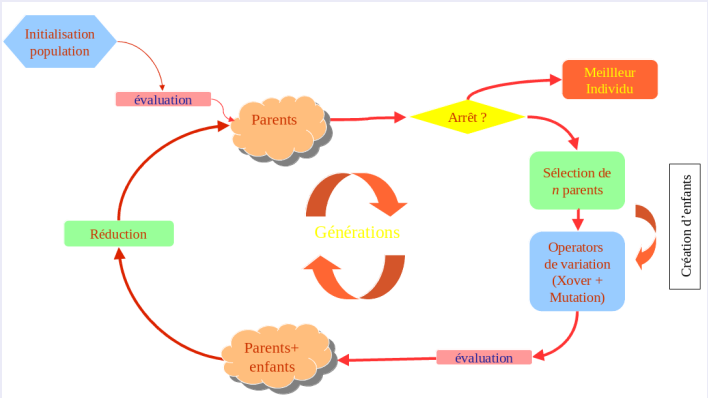
Algorithmes évolutionnaires

Eléments vitaux d'un AE

- Le codage des individus (solutions) : le génotype
  - Chaîne de bits (problèmes binaires, ou issue d'une quantification)
  - Vecteur de réels
  - Vecteurs d'entiers
  - Graphes
  - Mixtes...
- Un "outil" d'évaluation d'une solution
  - Une forme analytique
  - Une forme analytique non différentiable
  - Une boîte noire
- Un opérateur d'initialisation, le plus aléatoire possible pour une bonne diversité ! (trop guider l'algorithme conduit à une convergence prématurée)
- Des opérateurs de variation permettant une bonne exploration de l'espace et l'exploitation des itérations précédentes

Algorithmes évolutionnaires

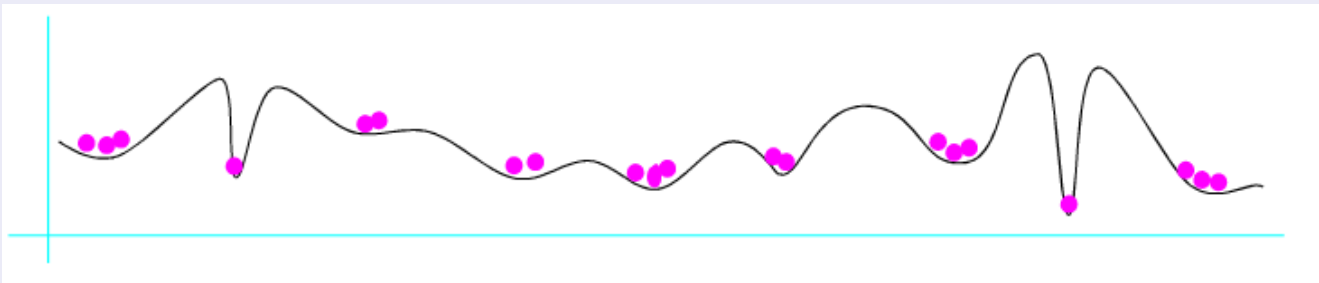
Sélection



Utilisée pour :

- former des "couples" de parents : avec remise
- réduire la population : sans remise

- Objectif : maintenir de la diversité, tout en convergeant vers les bonnes solutions

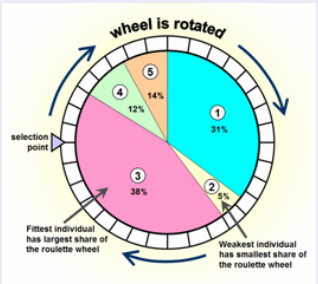


- Notion de pression de sélection
  - Trop fort : convergence prématurée
  - Trop faible : recherche aléatoire

Algorithmes évolutionnaires

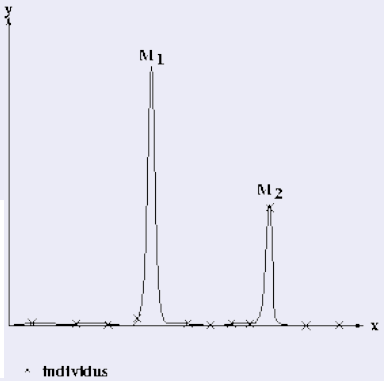
Sélection par roue de loterie

- Opérateur "historique"



Taille de la zone i : $\frac{J(i)}{\sum_i J(i)}$

- Pression dépendante du paysage de fitness

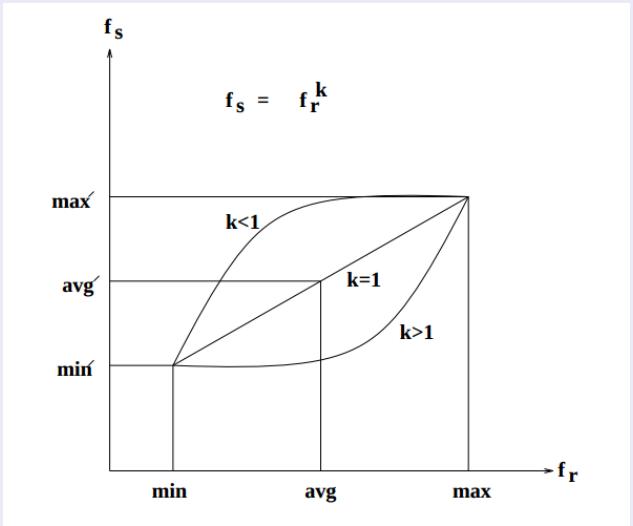


- Coûteux
- Non parallélisable : si les individus sont distribués, synchro nécessaire.

Algorithmes évolutionnaires

Sélection par roue de loterie améliorée

- Scaling linéaire : $J_s = aJ_r + b$
  - $a = \frac{max_s - min_s}{max_r - min_r}$
  - $b = \frac{min_s max_r - max_s min_r}{max_r - min_r}$
- Scaling exponentiel : $f(J) = J^k$, éventuellement $f(J) = J^{k(n)}$



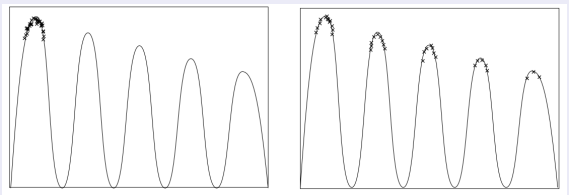
- $k < 1$: écart réduits : exploration
- $k > 1$: écarts amplifiés : exploitation
- Idéalement, k doit varier
- $k(n) = \left(tan\left[\left(\frac{n}{N+1}\right)\frac{\pi}{2}\right]\right)^p$

- Utilisation du ranking :
  - Taille de la Case i : $(TaillePop + 1 - Rang(J(i)))/\sum i$
  - Dépendant de la taille de la population

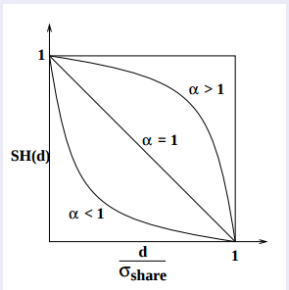
Algorithmes évolutionnaires

Sélection par roue de loterie avec sharing

- On peut aussi vouloir se répartir dans l'espace des solutions : le sharing



- On peut intégrer cela dans le fitness : $f(J_i) = \frac{J_i}{m_i}$ avec $mi = \sum_{j=1}^{N} S(d(x_i, x_j))$ où $S(d) = 1 - \left(\frac{d}{\sigma}\right)^{\alpha}$ si $d < \sigma$, 0 sinon
- σ délimite le voisinage d'un point, α définit la "force" de la pénalité



- Efficace, mais très coûteux ($O(N^2)$)

Algorithmes évolutionnaires

Sélection par roue de loterie avec sharing

- Il existe aussi des approches à base de clustering pour faire du sharing
- Exemple : initialisation de chaque individu comme centre d'un cluster
- Itérations :
  - Si deux centres sont à une distance $d < d_{min}$: fusion des cluster + calcul du nouveau centre
  - Parcours des individus :
    - si distance au plus proche centre acceptée $d < d_{max}$, agrégation
    - sinon, nouveau cluster
- On calcule alors : $f(J_i) = \frac{J_i}{m_i}$ avec $mi = n_c \left(1 - \left(\frac{d_{ic}}{2d_{max}}\right)^{\alpha}\right)$
- Avec
  - n_c : nombre d'individu dans le même cluster que l'individu i.
  - α coefficient de sensibilité
  - d_{ic} : distance entre l'individu et son centre
- Efficace, et moins coûteux ($O(NlogN)$)

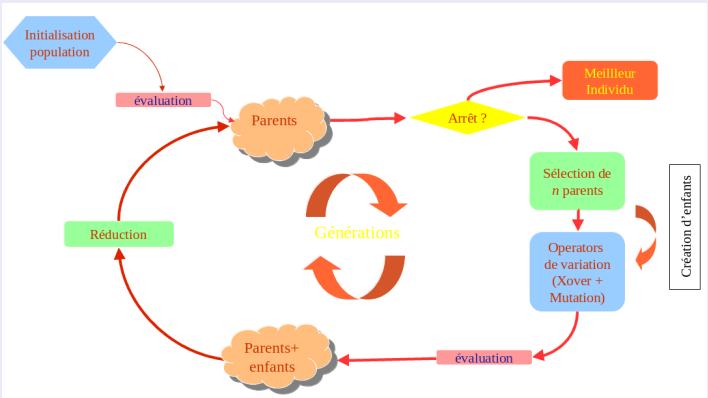
Algorithmes évolutionnaires

Sélection par tournois

- Tournois déterministes
  - Choix uniforme de n individus parmi T, le meilleur est sélectionné
  - Pression sélective croissante avec n (1 : exploration, T :déterministe)
  - Parallélisable, rapide, indépendant du paysage
  - Valeurs classiques : de 2 à 7
- Tournois stochastiques
  - Permet d'ajuster plus finement la pression sélective
  - Le meilleur est pris parmi les n avec une proba t : $t = 0.5 \rightarrow$ choix aléatoire, $t = 1 \rightarrow$ tournoi binaire
- Tournois "Evolutionnary programming"
  - Permet d'affecter un rang stochastique à chacun des individus d'une population
  - On fixe le nombre de rangs à affecter : T
  - Pour chaque individu, tournoi de taille T en tirant $T - 1$ concurrents
  - $rang(i) = T - NbIndividusBattus$

Algorithmes évolutionnaires

Opérateurs génétiques



- Deux principaux opérateurs historiques
  - Croisement
  - Mutation

Croisement

- Objectif : partage de bon gènes
- Algorithmes dépendant du codage
- Cas historique binaire : choix aléatoire d'un « locus » + permutations
- Nombreuses variantes : mono-point, multi-points, uniforme ...
- Attention à l'épistasie (correlation entre gènes) : peut être mesurée en comparant différents opérateurs

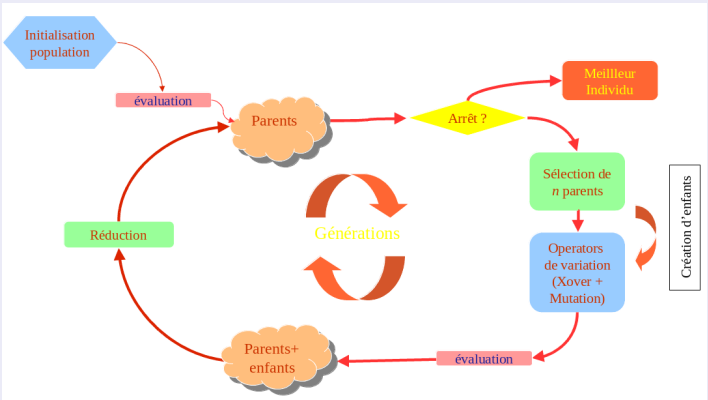
Algorithmes évolutionnaires

Croisement entre gènes réels

- Croisement barycentrique :
  - $enfant[x] = (1 - \alpha) * p1[x] + \alpha * p2[x]$ avec α entre 0 et 1
  - l'enfant est entre les parents
- $BLX - \alpha$ (BlendingXover) : autorise de sortir de l'intervalle
  - $enfant[x] = (1 - \gamma) * p1[x] + \gamma * p2[x]$ avec $\gamma = (1 + 2\alpha)u - \alpha$ et u dans $[0, 1]$
  - $\alpha = 0$: croisement barycentrique.
  - Valeur courante : 0.5
  - Plus les parents sont proches, plus les enfants seront proches.
- *SBX* (Simulated Binary Crossover)
  - Proche du crossover monopoint
  - Voir site de Kalyanmoy Deb

Algorithmes évolutionnaires

Opérateurs génétiques



- Deux principaux opérateurs historiques
  - Croisement
  - Mutation

Mutation

- Objectif : sortir des extrema locaux
- Version historique sur gènes binaires : sélection aléatoire d'un gène et modification éventuelle de sa valeur (probabilisée)
- Nombreuses variantes : bruit Gaussien, adaptatives...

Algorithmes évolutionnaires

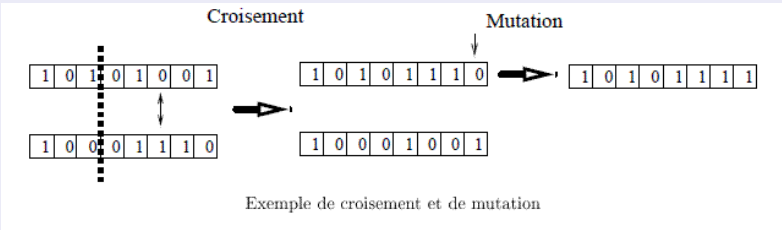
Paradigmes évolutionnaires

- Remplacement générationnel :
  - p enfants générés, ils remplacent les p parents
  - Nécessite de l'élistisme pour garantir la convergence : le meilleur parent est préservé
- Steady-State :
  - Sélection de 2 individus par tournoi
  - Création d'1 enfant par opérateurs génétiques
  - L'enfant créé remplace le plus mauvais des 2 parents, ou un parent sélectionné par tournoi ou le plus vieux, ou le pire, ou...
  - Pas besoin d'élitisme
- Stratégies d'évolution $(\mu+, \lambda)$
  - sélection de μ parents + mutation pour produire λ enfants (hist)
  - Sélection $(\mu + \lambda)$-ES : nouvelle pop = μ meilleurs parmi les $\mu + \lambda$
  - Sélection (μ, λ)-ES : nouvelle pop = μ meilleurs parmi λ
- CMA-ES (Covariance Matrix Adaptation) : meilleur algorithme actuel

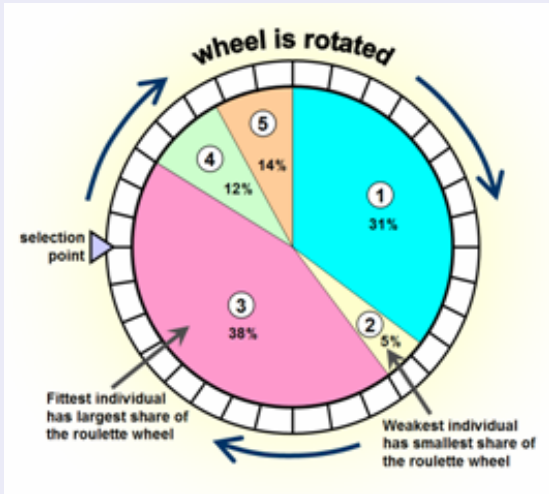
Algorithmes évolutionnaires

Exemple : l'algorithme simpl(ist)e de Goldberg

- Sélection : roue de loterie biaisée
- Individus : chaînes de bits issues de la discrétisation de paramètres
- Croisement : permutation de sous-chaînes
- Mutation : modification aléatoire de gènes



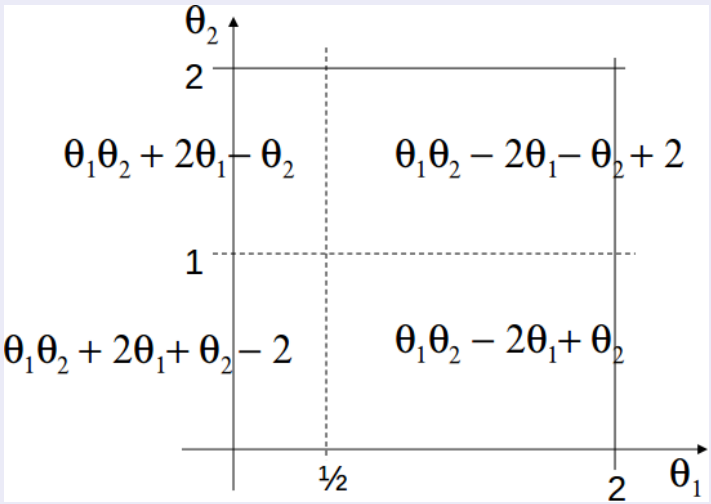
Exemple de croisement et de mutation

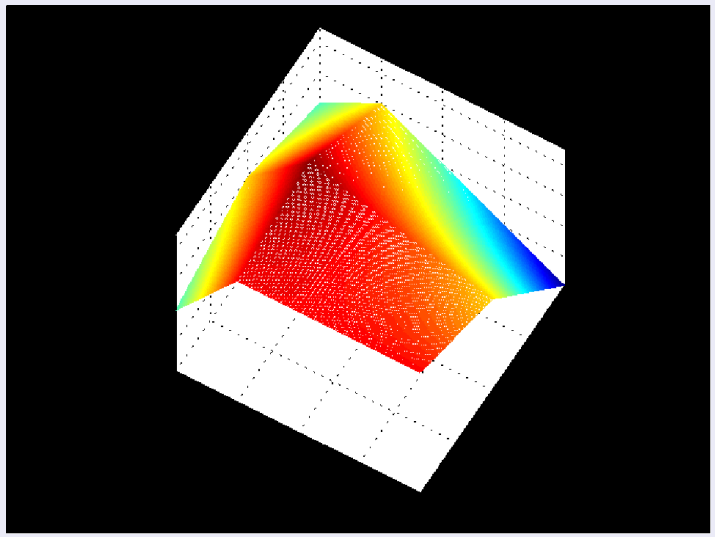


Algorithmes évolutionnaires

Sur un exemple jouet

- Objectifs :
  - Maximiser $J(\theta) = \theta_1\theta_2 - 2|2\theta_1 - 1| - |\theta_2 - 1|$
  - Domaine de définition : $(\theta_1, \theta_2) \in [0, 2]^2$





- Choix :
  - Codage binaire des paramètres
  - Opérateurs sur gènes binaires

Algorithmes évolutionnaires

Codage

- Codage binaire :
  - Correspond à une discrétisation de l'espace des paramètres
  - Exemple : 8 bits par paramètres

$$\theta_1 = \frac{b_{17}2^7 + b_{16}2^6 + b_{15}2^5 + b_{14}2^4 + b_{13}2^3 + b_{12}2^2 + b_{11}2^1 + b_{10}2^0}{2^7}$$

$$\theta_2 = \frac{b_{27}2^7 + b_{26}2^6 + b_{25}2^5 + b_{24}2^4 + b_{23}2^3 + b_{22}2^2 + b_{21}2^1 + b_{20}2^0}{2^7}$$

  - Les b_{ik} sont des variables binaires
  - Un individu (solution potentielle) :

$$b_{17}\, b_{16}\, b_{15}\, b_{14}\, b_{13}\, b_{12}\, b_{11}\, b_{10}\, b_{27}\, b_{26}\, b_{25}\, b_{24}\, b_{23}\, b_{22}\, b_{21}\, b_{20}$$

Algorithmes évolutionnaires

Initialisation

- Tirage aléatoire : exploration
- Exemple avec évaluation :

		θ_1	θ_2	$J(\theta_1, \theta_2)$
1	(10010111 10000100)	1,18	1,03	-0,175
2	(00011011 00011010)	0,21	0,20	-1,33
3	(01011100 01011011)	0,72	0,71	-0,21
4	(11101001 10110110)	1,82	1,42	-0,472
5	(00001011 10000111)	0,08	1,05	-0,784
6	(11101110 10001110)	1,86	1,11	-0,767
7	(00101110 10000101)	0,36	1,04	0,053
8	(11100111 01110110)	1,80	0,92	-0,986
			$\Sigma J =$	-4,671
			J min =	-1,33

Algorithmes évolutionnaires

Sélection pour la génération suivante

- Transformation des valeurs de fonction objectif en adaptation (fitness) :

		θ_1	θ_2	$J(\theta_1, \theta_2)$	Adapt
1	(10010111 10000100)	1,18	1,03	-0,175	18,24%
2	(00011011 00011010)	0,21	0,20	-1,33	2,02%
3	(01011100 01011011)	0,72	0,71	-0,21	17,75%
4	(11101001 10110110)	1,82	1,42	-0,472	14,07%
5	(00001011 10000111)	0,08	1,05	-0,784	9,69%
6	(11101110 10001110)	1,86	1,11	-0,767	9,93%
7	(00101110 10000101)	0,36	1,04	0,053	21,45%
8	(11100111 01110110)	1,80	0,92	-0,986	6,85%
			$\Sigma J =$	-4,671	100%
			$J\,min\ =$	-1,33	

- Tirage biaisé par l'adaptation : 1, 1, 3, 4, 5, 6, 7, 7

Algorithmes évolutionnaires

Variations génétiques

- Croisement
  - formation de couples
  - choix aléatoire d'un point de coupure
  - permutation des deux fins de gènes (situés après le point de coupure)
  - (1, 7, c6), (1, 5, c2), (3, 7, c12), (4, 6, c9))

				θ_1	θ_2
1	(100101	11	10000100)	1,18	1,03
7	(001011	10	10000101)	0,36	1,04

- Mutation : Changement de valeur de bits choisis aléatoirement

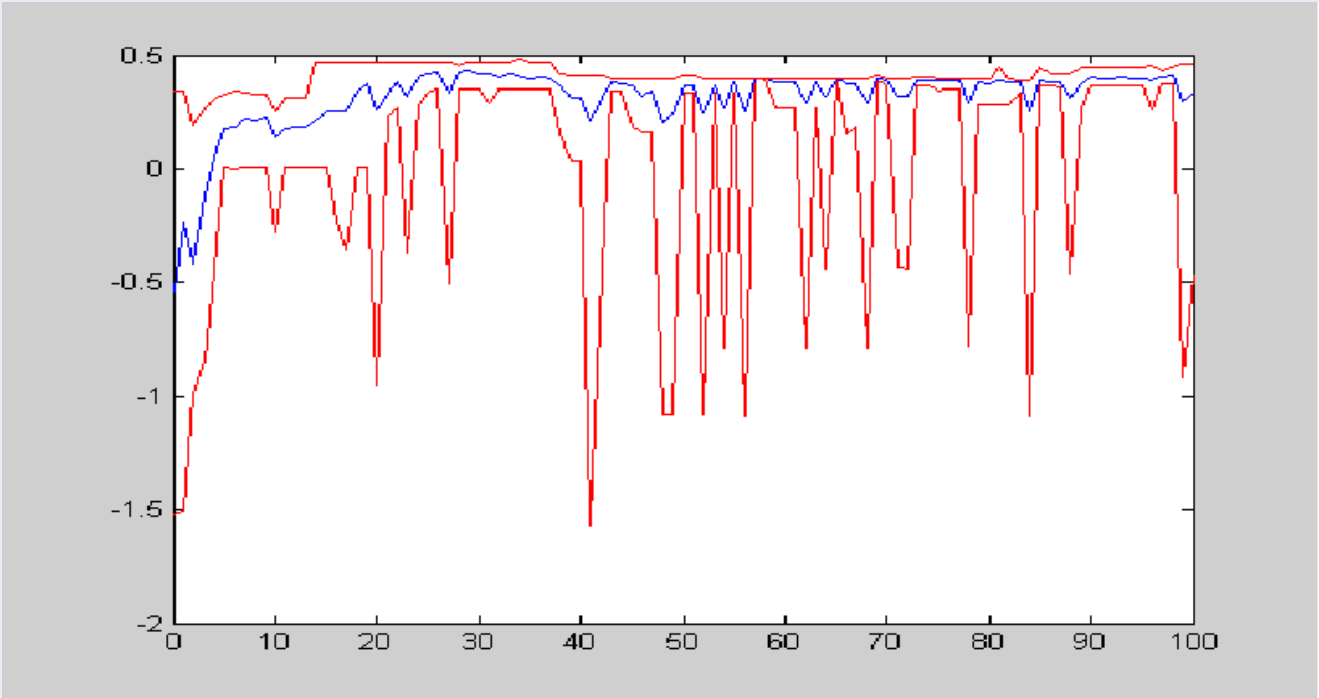
Algorithmes évolutionnaires

Résultats à l'issue d'une génération

	θ_1	θ_2	$J(\theta_1, \theta_2)$	Adapt
(10010110 10000101)	1,17	1,04	-0,163	13,88%
(00101111 10000100)	0,68	1,03	0,310	22,40%
(10010111 10000111)	1,086	1,055	-0,08127	15,35%
(00010111 10000100)	0,18	1,03	-0,4846	8,09%
(01011100 01001101)	0,719	0,664	-0,2961	11,48%
(00101110 10001011)	0,36	1,086	0,02307	17,23%
(11101001 00001110)	1,82	1,11	-0,7306	3,65%
(11101110 00110110)	1,86	1,42	-0,4968	7,87%
		$\Sigma J =$	-1,9193	
		Jmin=	-0,7306	

Algorithmes évolutionnaires

Comportement de l'algorithme simple



Algorithmes évolutionnaires

Améliorations possibles

- Élistisme vs Sauvegarde
- Mutation adaptative = fct(itérations, distrib. gènes, distrib obj.)
- Niches écologiques

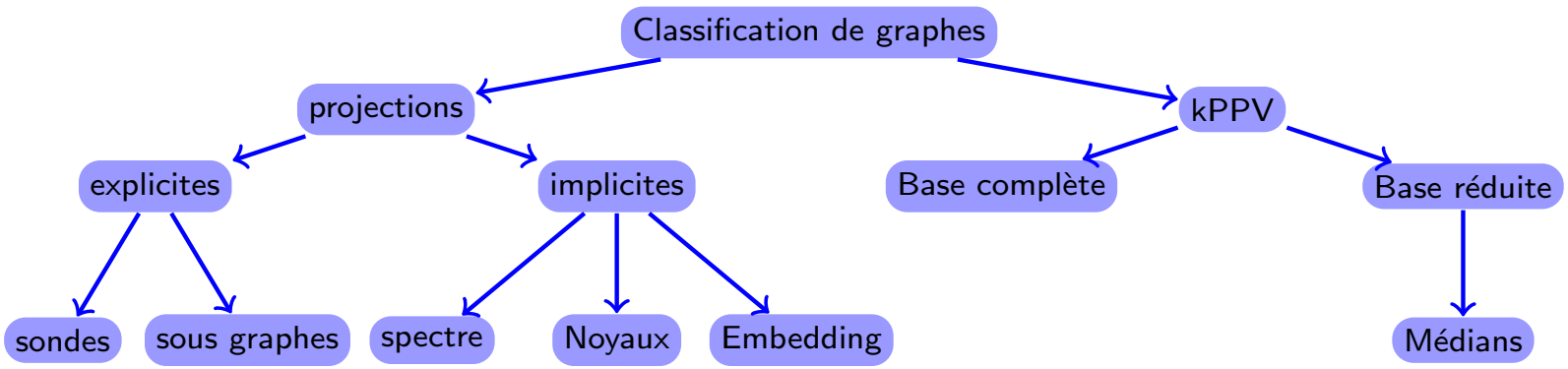
Extensions possibles

- Métaheuristiques hybrides
  - Lamarckisme : évolutions locales déterministes des gènes (ex : un/des pas de gradient comme opérateur)
  - Baldwinisme : évaluation sur le potentiel de modifications locales (on regarde ce que donne un gradient, mais on ne le fait pas)
- Programmation génétique, differential evolution ($D = C + A\vec{B}$) où A, B et C sont 3 bons individus...
- Démos : http ://www.rennard.org/alife/french/gav.html, http ://math.hws.edu/xJava/GA/

Algorithmes évolutionnaires

Un problème plus complexe

Soit $G = \{g_i, y_i\}_{i=1,\dots,n}$ un ensemble de graphes avec $(g, y) \in \mathcal{G} \times Y$. On cherche une fonction $f : \mathcal{G} \to Y$ telle que $f(g) = \hat{y}$



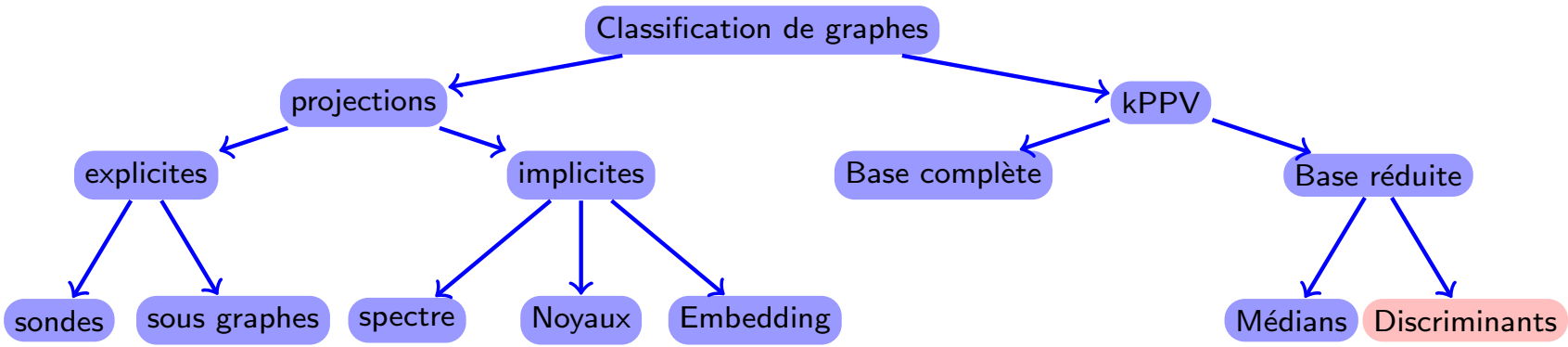
Base réduite : un processus d'optimisation

- Prototypes médians [Jiang 01] : minimisation des distances intra-classes

Algorithmes évolutionnaires

Un problème plus complexe

Soit $G = \{g_i, y_i\}_{i=1,\ldots,n}$ un ensemble de graphes avec $(g, y) \in \mathcal{G} \times Y$. On cherche une fonction $f : \mathcal{G} \rightarrow Y$ telle que $f(g) = \hat{y}$

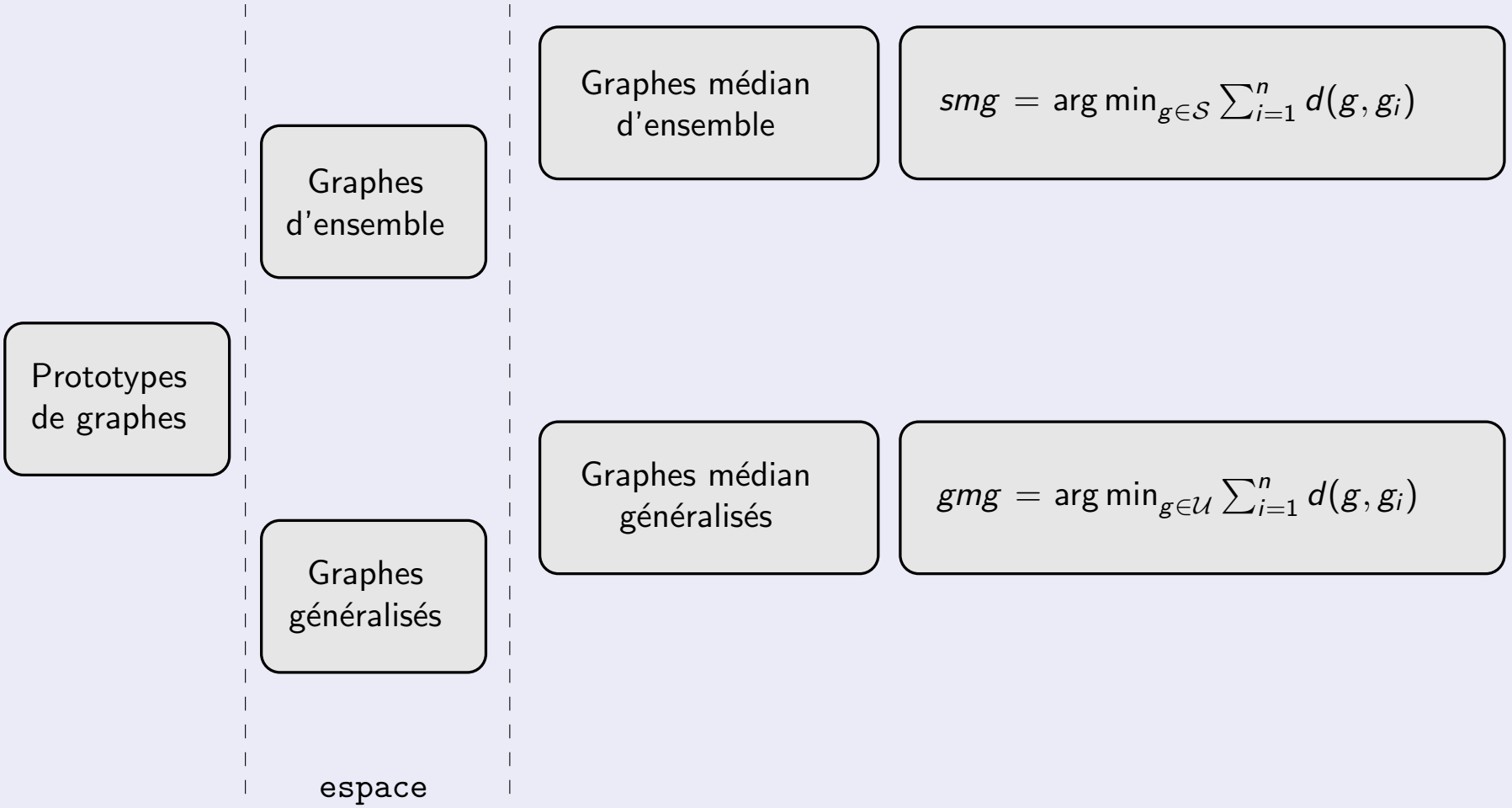


Base réduite : un processus d'optimisation

- Prototypes médians [Jiang 01] : minimisation des distances intra-classes
- Prototypes discriminants [CVIU'11] : optimisation d'un critère d'erreur

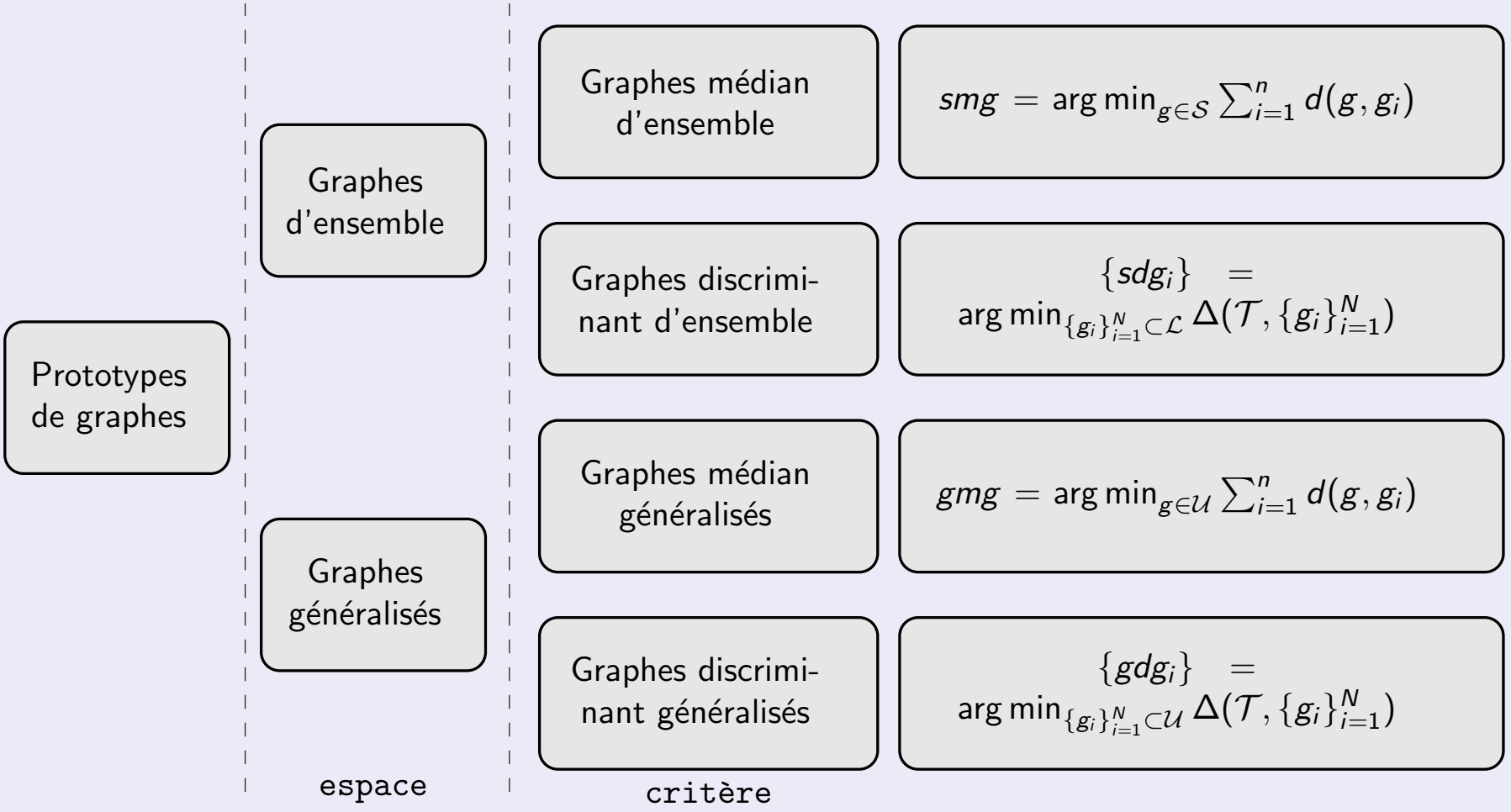
Algorithmes évolutionnaires

Prototypes de graphes



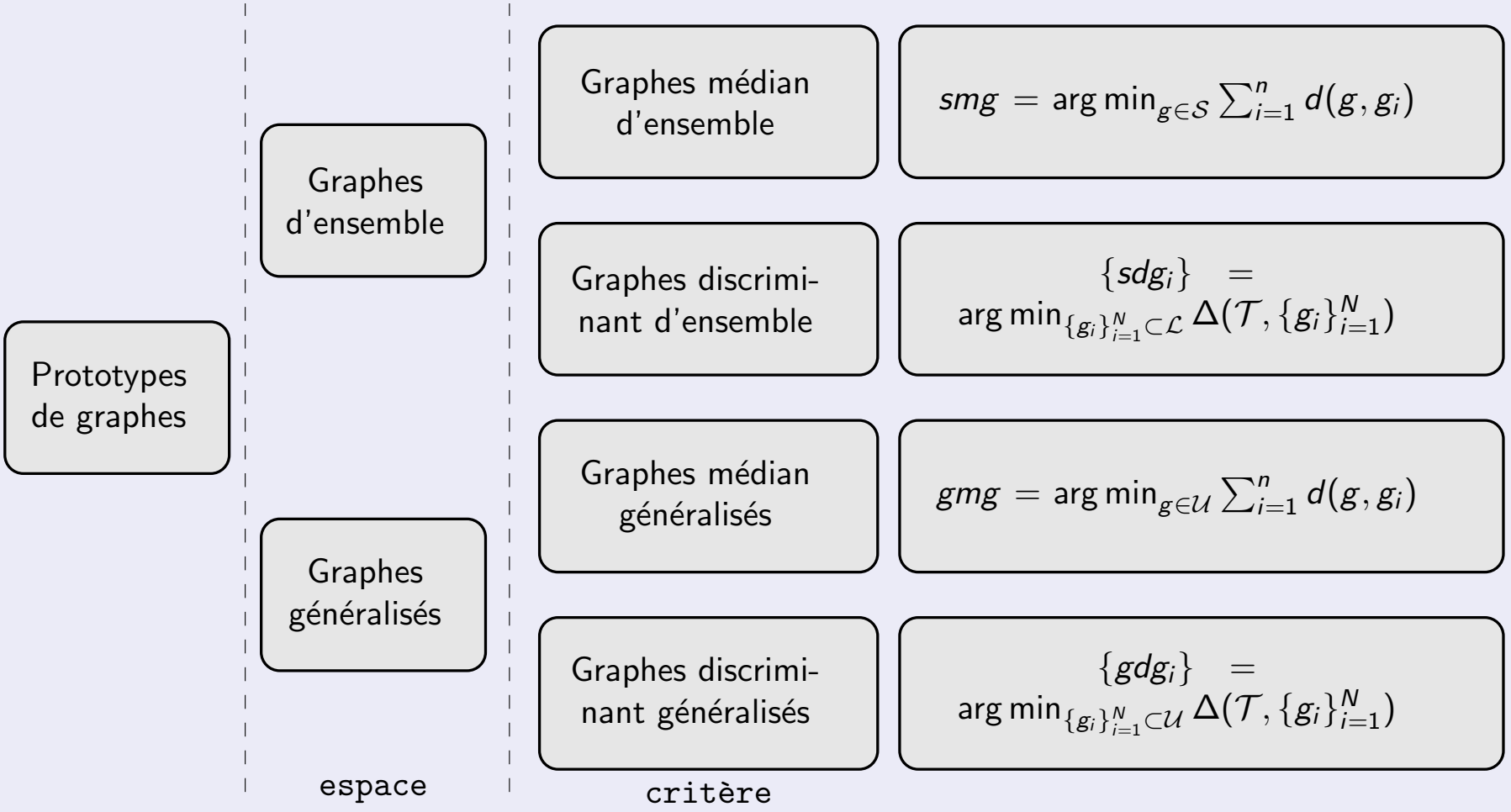
Algorithmes évolutionnaires

Prototypes de graphes



Algorithmes évolutionnaires

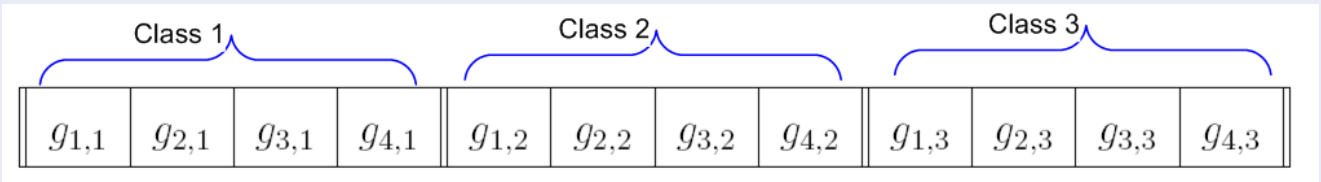
Prototypes de graphes



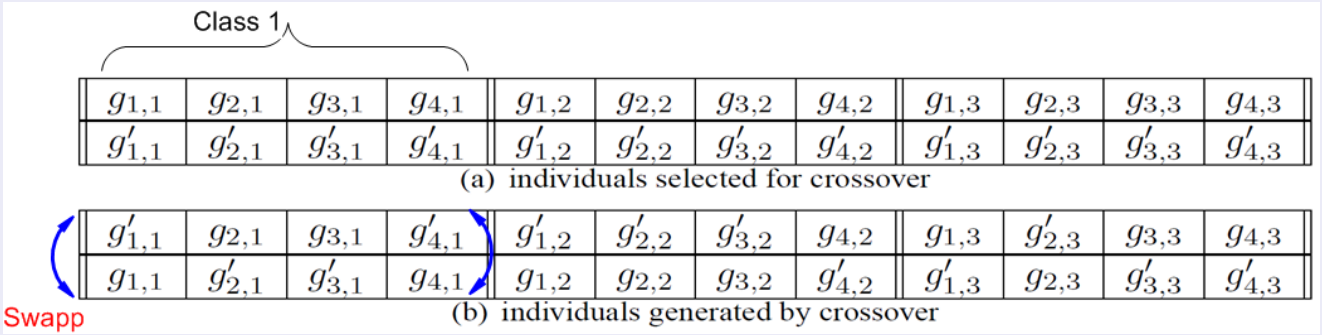
$\rightarrow$ Un problème d'optimisation

Algorithmes évolutionnaires

Codage

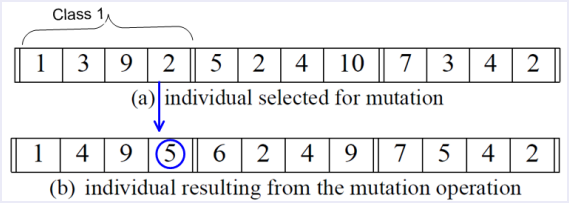


Croisement

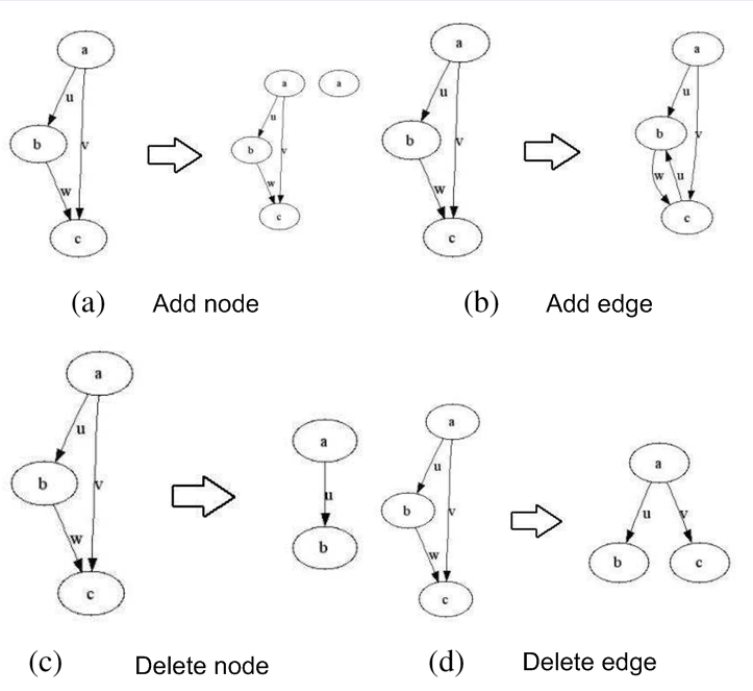


Algorithmes évolutionnaires

Mutation cas GxE

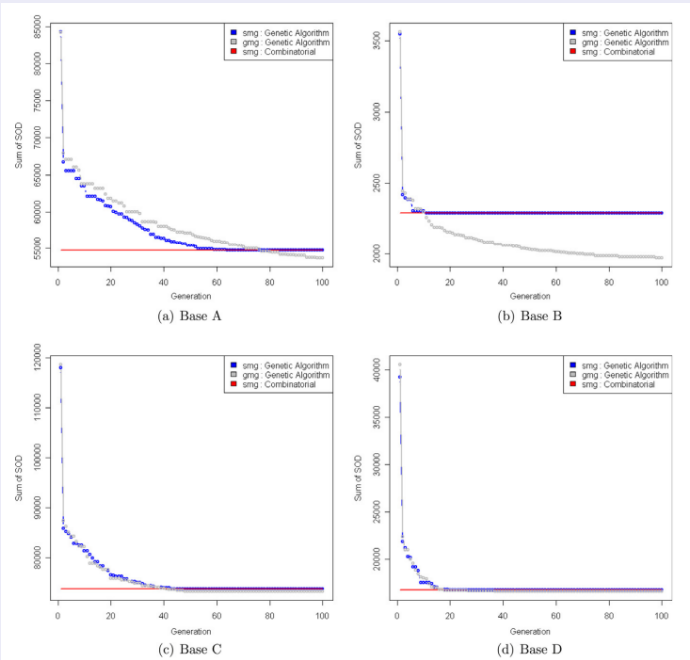


Mutation cas GxG

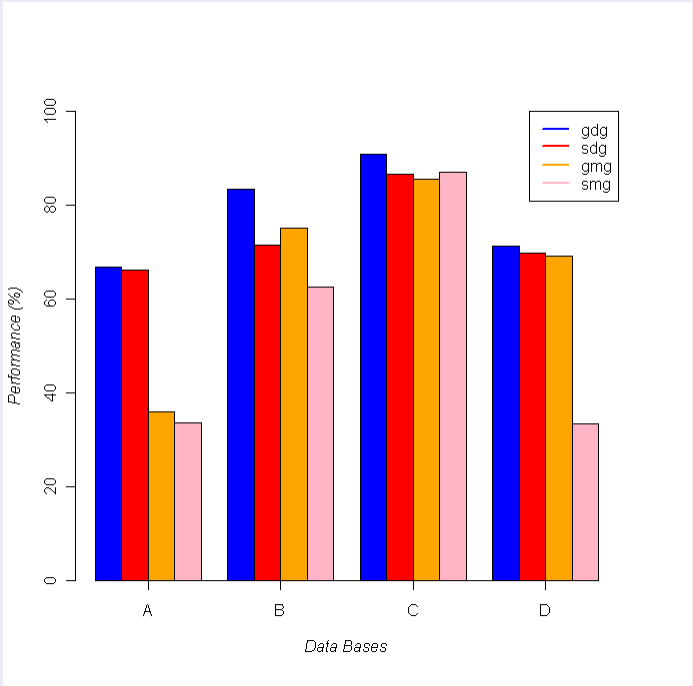


Algorithmes évolutionnaires

Convergence



Comparaison prototypes



Plan du Cours

Essaims particulaires

Principes

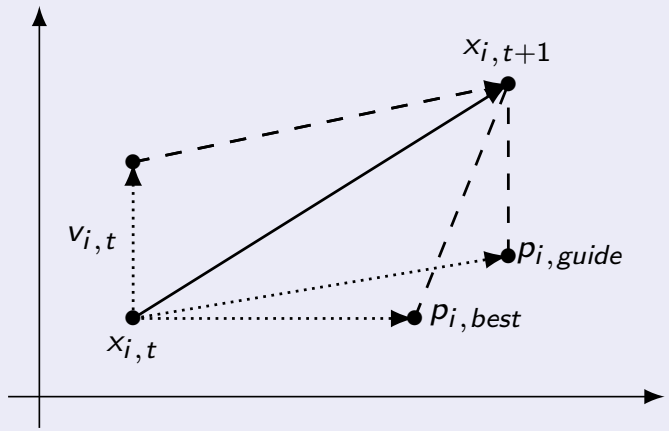
- Objectifs :
  - Minimiser itérativement un critère $J(\theta)$
  - Ne pas se « bloquer » dans les minimus locaux
- Historique :
  - Russel Eberhart (ingénieur en électricité) et James Kennedy (socio-psychologue) en 1995
  - Spécialiste Français : Maurice Clerc (FT R&D)
- Analogie
  - Intelligence en essaims (comme les colonies de fourmis) : « le tout est plus que la somme des parties » : émergence de comportements sociaux
- Points clefs
  - Population de solutions (ici des particules)
  - Coopération entre particules régies par des règles simples
    $\rightarrow$ Pas d'hypothèse sur le critère

Essaims particulaires

Principe de fonctionnement

Les particules se déplacent dans l'espace des paramètres en fonction de (i) leur inertie, (ii) leur mémoire et (iii) leur voisines

$$
\begin{aligned}
x_{i,t+1} = \quad & x_{i,t} + \chi(v_{i,t+1}) \\
v_{i,t+1} = \quad & \omega . r_0 . v_{i,t} + \\
& c_1 . r_1 . (p_{i,best} - x_{i,t}) + \\
& c_2 . r_2 . (p_{i,guide} - x_{i,t})
\end{aligned}
$$



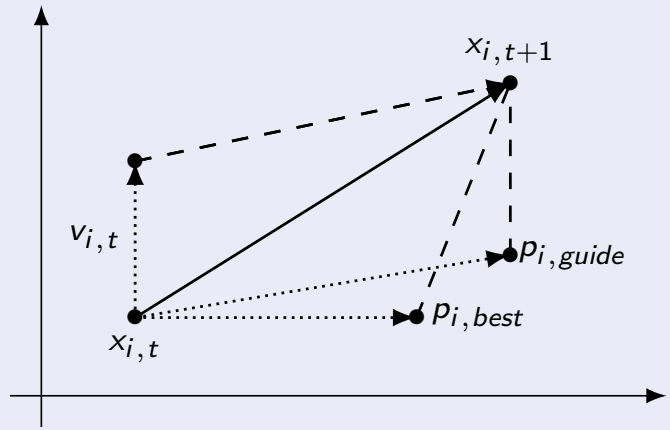
- $x_{i,t}$ est la position de la i^{eme} particule à l'instant t
- $v_{i,t}$ est la vitesse de la i^{eme} particule à l'instant t
- $p_{i,best}$ est la meilleure position visitée (mémoire individuelle)
- $p_{i,guide}$ la position d'un guide (mémoire collective)
- r_0, r_1 et r_2 sont des valeurs aléatoires

Essaims particulaires

Principe de fonctionnement

- Population de particules

$$
\begin{aligned}
v_{i,t+1} = \quad & \omega . r_0 . v_{i,t} + \\
& c_1 . r_1 . (p_{i,best} - x_{i,t}) + \\
& c_2 . r_2 . (p_{i,guide} - x_{i,t}) \\
x_{i,t+1} = \quad & x_{i,t} + \chi(v_{i,t+1})
\end{aligned}
$$



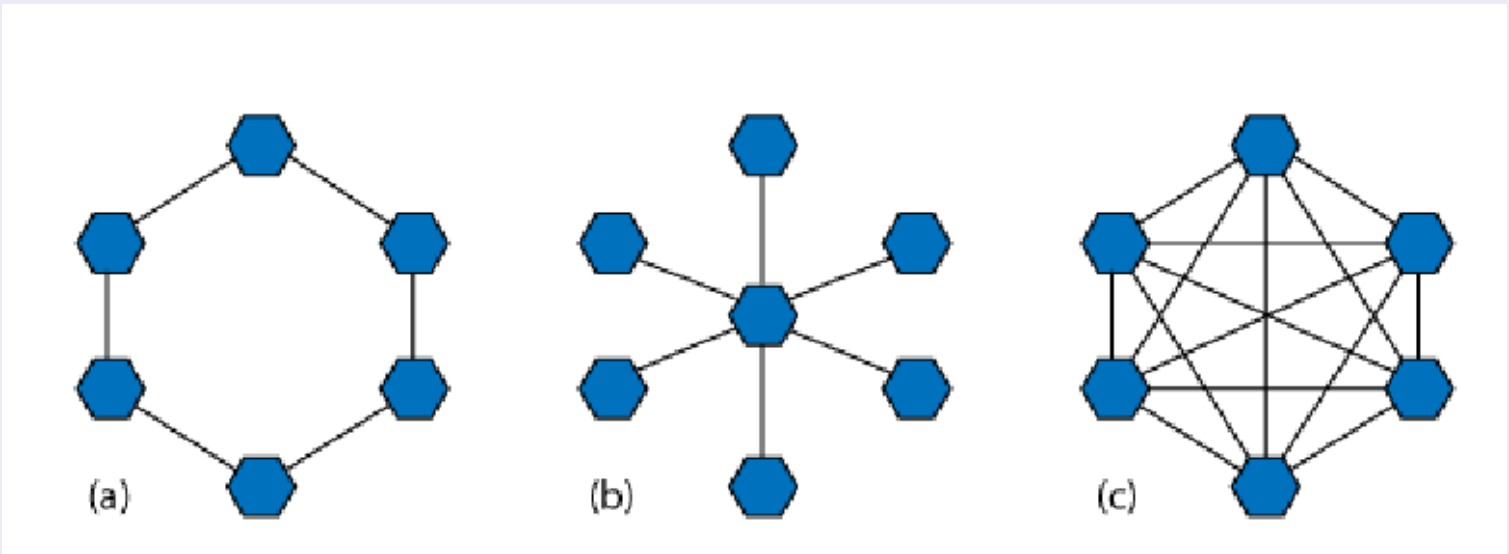
Explications

- $c_1 r_1$ est appelé facteurs individuel (r_1 est aléatoire)
- $c_2 r_2$ est appelé facteurs social (r_2 est aléatoire) : panurgisme
- ω est appelé facteur d'inertie
- La fonction $\chi()$ est généralement implémentée comme un simple facteur de turbulence, ou pour éviter une divergence de la vitesse

Essaims particulaires

Réseau social

- Les particules communiquent dans un ≪ voisinage ≫
- Il existe différents types de voisinnage
  - En Anneau : chaque particule est reliée à n particules
  - En Rayon : les particules ne communiquent qu'avec une particule centrale
  - En Etoile : chaque particule est reliée à toutes les autres



Essaims particulaires

Exemple d'algorithme

[Les variables et paramètres de l'algorithme]
N nombre de particules
$\overrightarrow{x_i}$ position de la particule P_i
$\overrightarrow{v_i}$ vitesse de la particule P_i
$pbest_i$ meilleure fitness obtenue pour la particule P_i
$\overrightarrow{x}_{pbest_i}$ position de la particule P_i pour la meilleure fitness
$\overrightarrow{x}_{gbest}$ position de la particule ayant la meilleure fitness de toutes
ρ_1, ρ_2 valeurs aléatoires positives
[———————————————————————————————]

[Initialisations]
Initialiser aléatoirement la population
...

[Traitement]
Répéter
  Pour i de 1 à N **faire**
    Si $(\mathcal{F}(\overrightarrow{x_i}) > pbest_i)$ **Alors**
      $pbest_i \leftarrow \mathcal{F}(\overrightarrow{x_i})$
      $\overrightarrow{x}_{pbest_i} \leftarrow \overrightarrow{x_i}$
    Fin Si
    Si $(\mathcal{F}(\overrightarrow{x_i}(t)) > gbest)$ **Alors**
      $gbest \leftarrow \mathcal{F}(\overrightarrow{x_i})$
      $\overrightarrow{x}_{gbest} \leftarrow \overrightarrow{x_i}$
    Fin Si
  Fin Pour
  Pour i de 1 à N **faire**
    $\overrightarrow{v_i} \leftarrow \overrightarrow{v_i} + \rho_1(\overrightarrow{x}_{pbest_i} - \overrightarrow{x_i}) + \rho_2(\overrightarrow{x}_{gbest} - \overrightarrow{x_i})$
    $\overrightarrow{x_i} \leftarrow \overrightarrow{x_i} + \overrightarrow{v_i}$
  Fin Pour
jusqu'à ce que (le processus converge)

Algorithme 2 – Algorithme avec un voisinage en étoile

Nombreux choix

- Nombre de particules
- Valeurs de c_1 et c_2
- Le voisinage
- Le facteur d'inertie
- Le critère d'arrêt
- Le voisinage
- Éventuellement une vitesse maximale

Essaims particulaires

Nombreux rafinements possibles

- Exemple avec une fonction objectif dynamique
- Exemple avec plusieurs essaims et plusieurs objectifs