

Chapitre 4 : algorithmes d'optimisation globale

* Problème à résoudre : recherche de $\alpha^* \in \Omega \subset \mathbb{R}^n$ minimum global de $J : \Omega \rightarrow \mathbb{R}$, c'est à dire tel que $\forall x \in \Omega, J(\alpha^*) \leq J(x)$

(on pourra supposer J seulement C^0 sur Ω)

→ Les méthodes vues aux § 2 et 3 ne conviennent plus :

- * elles stationnent en un minimum local
- * le gradient de J n'est pas forcément défini

* L'optimisation globale est rencontrée dans de nombreuses situations réelles, où la fonction J admet de nombreux minima locaux à éviter :

→ cas de la fonction de Rastrigin :

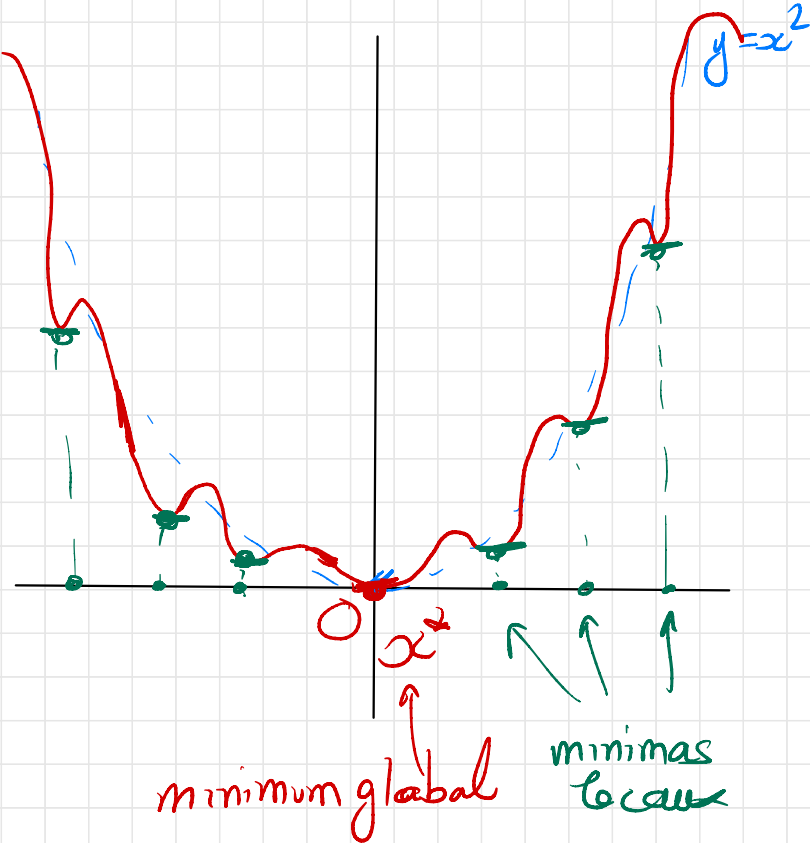
$$J(x_1, \dots, x_n) = \sum_{i=1}^n (x_i^2 - \cos(2\pi x_i)) + n$$

* J admet de nombreux minima locaux (de l'ordre de 10^n sur $[-5, 5]^n$)

* J admet un unique minimum global en $\alpha^* = (0, \dots, 0)$

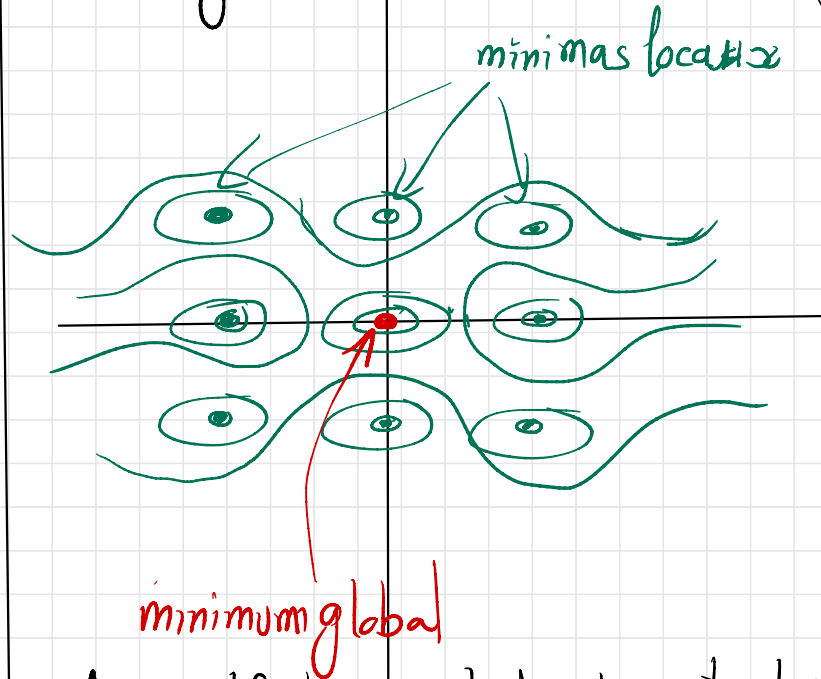
* Rastrigin en 1D ($n=1$):

$$J(x) = x^2 - \cos(2\pi x) + 1$$



* Rastrigin en 2D ($n=2$)

2



* Les méthodes à construire devront autoriser d'éventuelles remontées d'une iteration à l'autre.

* Un aspect aléatoire est introduit dans ces algorithmes.

* On présente ici les 4 méthodes les plus répandues dans ce contexte :

- * Recuit simulé
- * Algorithmes génétique
- * Stratégies d'évolution
- * PSO (Particle Swarm Optimization)

4.1) Recuit simulé

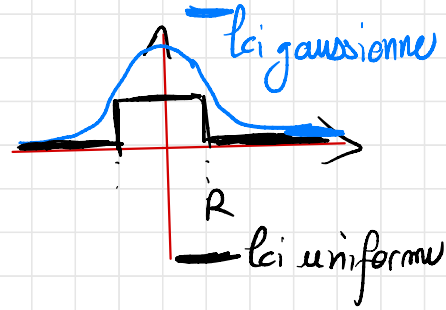
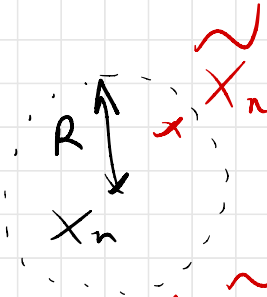
(simulated annealing)

→ Principe basé sur un procédé industriel de refroidissement progressif d'un métal pour l'amener à un état d'énergie minimal (~ 1950)

→ Mathématiquement^(avec $\Omega = \mathbb{R}^n$) on construit une suite aléatoire $X_0 \rightsquigarrow X_1 \rightsquigarrow \dots X_n \rightsquigarrow$ de la manière suivante :

* X_0 suit une loi particulière sur Ω

* Par passer de X_n à X_{n+1} :



→ on calcule X_{n+1} à partir de X_n et d'un voisinage de X_n suivant une loi aléatoire (uniforme, Gaussienne)

→ 2 cas sont possibles : $J(\tilde{X}_n) \leq J(X_n)$ ou $J(\tilde{X}_n) > J(X_n)$

* Si $\tilde{J}(\tilde{X}_n) \in J(X_n)$, on note
 $X_{n+1} = \tilde{X}_n$

* Si $\tilde{J}(\tilde{X}_n) > J(X_n)$, on garde
 quand même $\tilde{X}_n$ avec la probabilité:

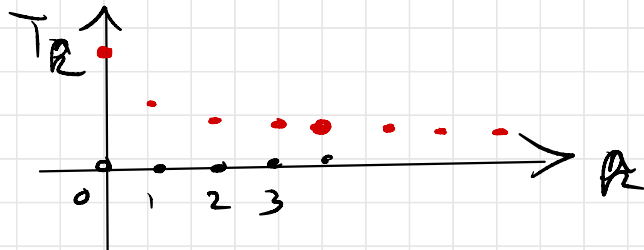
$$p = \exp\left(-\frac{\tilde{J}(\tilde{X}_n) - J(X_n)}{T_n}\right)$$

probabilité
faible si
remontée
forte

probabilité
se réduisant
avec les
itérations

où (T_n) est une suite (de températures)
 décroissant lentement vers 0.

(sinon $X_{n+1} = X_n$)



* Exemple d'implémentation sur 4
 la fonction de Raskigin:
 $\rightarrow \Omega = [-5, 5]^n$ *loi uniforme sur $[-5, 5]^n$*
 $(X_0 \sim U([-5, 5]^n))$
 $\rightarrow$ voisinage: $B(X_n, R)$
 avec $R = 0,1$ avec tirage de
 $\tilde{X}_R$ uniforme sur $B(X_n, R)$
 $\rightarrow$ décroissance lente:
 * $T_k = \frac{1}{\ln(k)}$ ou
 * $T_k = \rho^k$ (avec ρ proche de 1)
 * T_k décroissante par paliers

→ Sur cet exemple, la méthode du recuit simulé converge, avec un choix adapté de paramètres, vers le minimum global, y compris pour $n > 2$ (≈ 10).

* La sensibilité au choix de la suite de températures est forte :

→ si décroissance trop rapide (convergence prématurée)
→ si décroissance trop lente : pas de convergence

* Un grand nombre d'évaluations de la fonction J est nécessaire ($n=2: 2000$)
 $n=6: 20\,000$
 $\vdots$

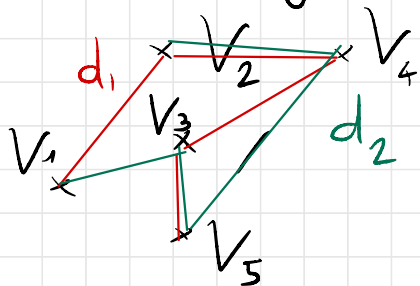
→ Mathématiquement, la convergence éventuelle d'une telle méthode doit s'exprimer de manière probabiliste, par exemple avec un résultat du type

$$\lim_{k \rightarrow \infty} P(\|X_k - \alpha^*\| \geq \varepsilon) = 0$$

→ Il existe de tels résultats, seulement sur des cas très particuliers et par des décroissances de températures pas nécessairement bien adaptées numériquement.

→ La méthode peut s'étendre à des fonctionnelles plus générales et définies sur d'autres espaces

que $\mathbb{R}^n$:
 Exemples : problème du voyageur de commerce (Traveling Salesman Problem: TSP)



On veut minimiser la distance totale parcourue en passant par toutes les villes (1 seule fois).

$$J: \begin{matrix} \mathbb{S}_n & \xrightarrow{\sigma} & \mathbb{R} \\ \left(\begin{array}{cccc} 1 & 2 & \dots & n \\ \downarrow & \downarrow & & \downarrow \\ \sigma(1) & & & \sigma(n) \end{array} \right) & \xrightarrow{\sigma} & \sum_{i=1}^n \|V_{\sigma(i+1)} - V_{\sigma(i)}\| \end{matrix}$$

$$\text{card}(\mathbb{S}_n) = n!$$

6

Le recuit simulé fonctionne par résoudre ce problème pour des valeurs de $n \approx 50$.

On définit la notion de dessin d'un parcours (voir illustration web : interslices)

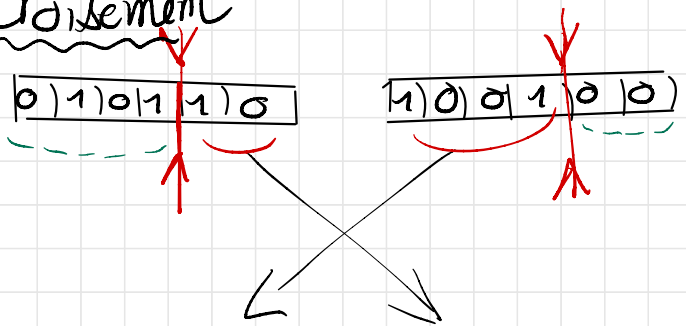
4.2) Algorithmes génétiques

Le principe de cette méthode est basé sur une analogie avec la théorie de Darwin (1854) d'évolution des espèces ("survival of the fittest")

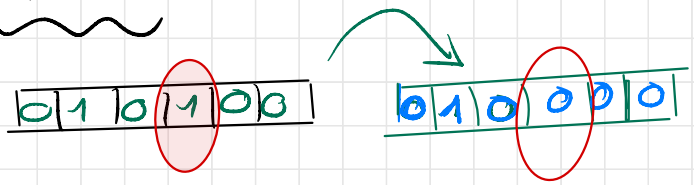
* On va définir et faire évoluer une famille de "chromosomes" (chaînes de gènes)
 $\boxed{0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0}$ suivant les 3 principes

Dorminien de selection, croisement et mutation //

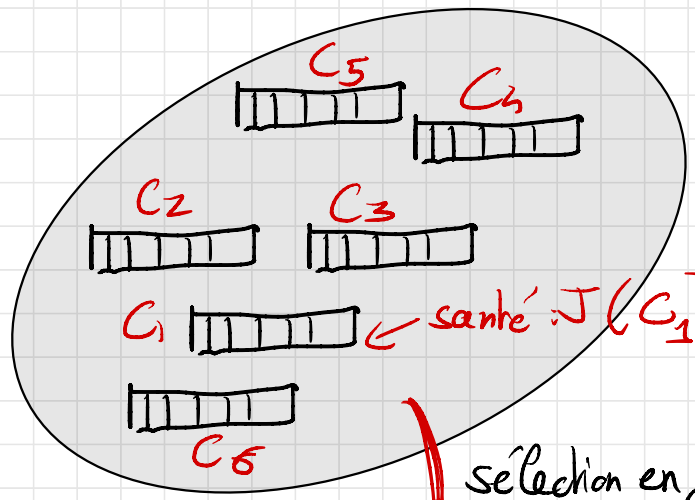
* Croisement



* Mutation :

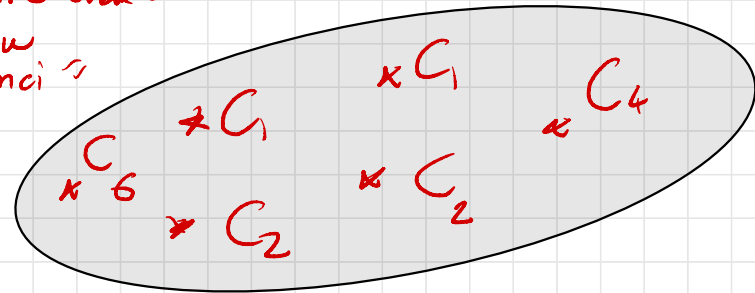


* Selection



"loterie suivant la
santé des chromosomes"
ou
"tournoi"

sélection en fonction
de la "santé"
 $\sum J(C_i)$



Schématiquement, l'évolution d'une population de N chromosomes se déroule ainsi :

$$C_i \in \{0,1\}^L$$

Initialisation aléatoire de N chromosomes $(C_i)_{1 \leq i \leq n}$

Exemple :

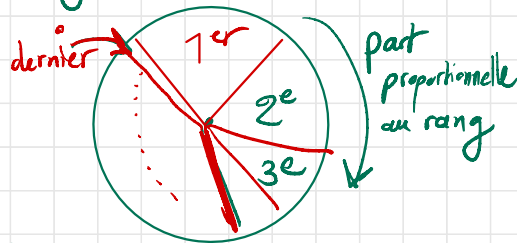
$$J(x_1, \dots, x_L) = \sum_{i=1}^L x_i$$

Évaluation de $(C_i)_{1 \leq i \leq n}$
 $\leadsto$ "santé" $J(C_i)$

(santé $\equiv$ inversement proportionnel à J)

E

sélection aléatoire par le rang :



itération

Sélection d'une nouvelle population "mieux adaptée"
 $\leadsto$ population $(C'_i)_{1 \leq i \leq n}$

S

Croisement (par paires) de la population

C

échange (aléatoire) de 2 séquences de chromosome avec probabilité p_c
 échange (aléatoire) d'un gène du chromosome (préba. p_m)

M

"Meilleur chromosome"
 $(0, \dots, 0)$

Ce premier algorithme ^(*) a été introduit par John Holland (~ 1970). Il a ensuite été étendu à l'optimisation des fonctions à variables réelles. ^(**)

(*) A.G. binaire

(**) A.G. réel

Pour cela, il suffit d'adapter au cas des variables réelles, les principes de croisement et de mutation (le reste de l'algorithme peut rester inchangé).

On introduit les nouveaux principes de croisement et mutation suivants :

* Croisement réel (barycentrique) / 9

$X_1 \in \mathbb{R}^n$ $X_2 \in \mathbb{R}^n$

$(1-\alpha)X_1 + \alpha X_2$
 $\alpha X_1 + (1-\alpha)X_2$

$\alpha \sim U([0,1])$

* Mutation : on reprend la mutation du recuit simulé.



Exemple d'implémentation :

* Fonction du Rastrigin ($n=2$ et $n=10$)

Remarques

1) Il existe de multiples variantes d'AG qui diffèrent par les choix des 3 principes (sélection, croisement, mutation).

2) En général, on rajoute un 4^{ème} principe, appelé élitisme^(*), qui consiste à garder à la génération suivante le meilleur élément de la génération actuelle (au détriment d'un élément choisi aléatoirement).

^(*) ou notion d'immunité ✓

3) Même si le coût de la méthode reste élevé (comme pour le recuit simulé),

il est possible de paralléliser les évaluations pour une génération

4) Comparativement aux méthodes de descente vues au § 2.3) on peut faire le comparatif :

Avantages

* Minimum global

* Pas de calcul de gradient

* Parallélisable

Inconvénients

* Coût de calcul

* Vitesse de convergence "fine"

* Choix des paramètres

4.3) Stratégies d'évolution

Il s'agit d'une variante d'un
algorithme génétique où les 3
opérateurs (sélection, croisement,
mutation) sont repensés :

- * priorité à la mutation
(croisement non indispensable)
- * nombre d'enfants supérieur au
nombre de parents
- * sélection déterministe sur la population
d'enfants et de parents

Plus précisément, on note :

μ : nombre de parents à la génération k

λ : nombre d'enfants à la génération k

(on a $\lambda \geq \mu$ et très souvent $\lambda \gg \mu$)

Une stratégie d'évolution se présente ainsi :
(voir page suivante)

Initialisation
(μ parents)

← Tirage aléatoire

(croisement éventuel)

(C_n) ← $(X_1, \dots, X_\mu)$ parents : $X = \frac{1}{\mu} \sum_{i=1}^{\mu} X_i$
(croisement isobarycentrique, déterministe)

Mutation
(λ enfants)

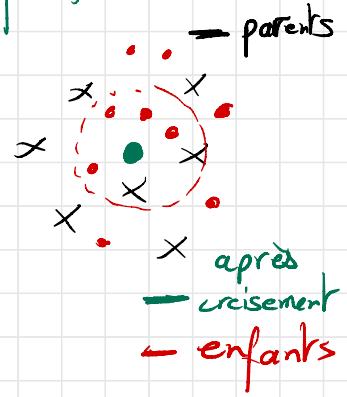
Mut. idem S.A et G.A
(tirage aléatoire dans un voisinage)

Évaluation
(λ enfants)

E_v ← λ évaluation avec J
(parallelisables)

Sélection
(μ nouveaux parents)

S_{el} ← choix des μ meilleurs éléments (au sens de J)
parmi → la famille des λ enfants
→ la famille des λ enfants + μ parents



Itération
 $k \rightarrow k+1$

Pour le premier type de sélection, on parle de stratégie "cemma" (ou virgule). On note l'algorithme $ES(\mu, \lambda)$.
Pour le second type, on parle de stratégie "plus". On note l'algorithme $ES(\mu + \lambda)$.

Remarque : Seule la stratégie "plus" contient le principe de l'élitisme.

Exemple d'implémentation :

→ Rastrigin ($n=2$ ou 10)

→ $ES(5, 50)$ si $n=2$, 20 générations

On note : la faible importance du croisement et

du type de sélection, mais aussi la sensibilité aux paramètres de mutation :

→ mutation trop faible : convergence ralentie

↘ mutation trop forte : perte d'information du point courant.

Pour remédier à cette sensibilité forte, des stratégies d'auto-adaptation des paramètres ont été développées (voir § 4.5)

Remarques :

1) Aucune démonstration générale et rigoureuse de convergence.

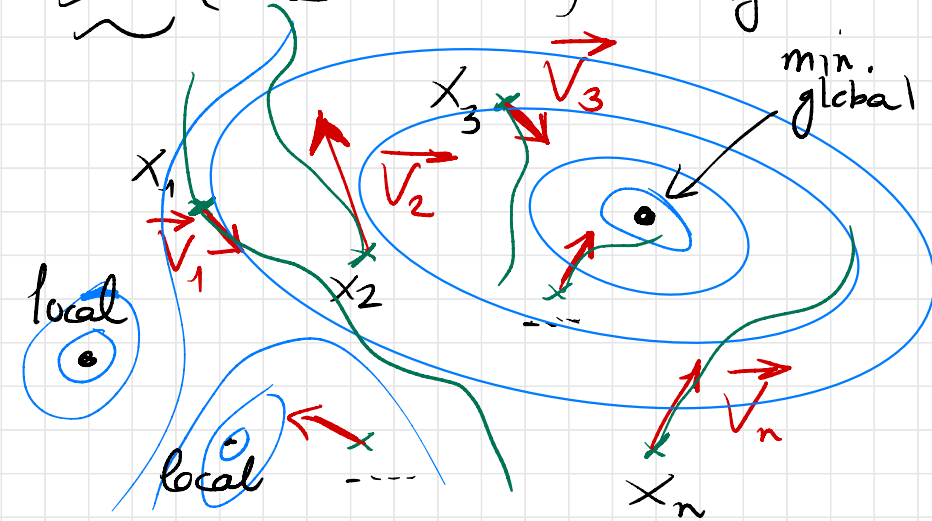
2) Création ~ 1950 mais améliorations toujours en cours et

très large utilisation pour des problèmes
à variables réelles.

4.4) P.S.O (Particle Swarm Optimization)

("essaim")

Idee (1995 Eberhart, Kennedy)



$$(x_i, v_i) \in \mathbb{R}^n \times \mathbb{R}^n \quad (1 \leq i \leq N)$$

le principe consiste à faire
évoluer une famille dynamique
de N éléments ("individus") qui
ont une mémoire individuelle et une
mémoire collective permettant globalement
à chacun de se diriger vers des
régions "prometteuses" (au sens de)
L'algorithme se présente ainsi :

Initialisation aléatoire
 $(X_i^0, V_i^0)_{1 \leq i \leq n}$

Evaluation

$(J(X_i))_{1 \leq i \leq n}$
 $\rightarrow$ mise en mémoire

$(k \rightarrow k+1)$

Evolution de la vitesse
 en fonction des informations

$V_i^k \rightarrow V_i^{k+1}$

Actualisation des positions

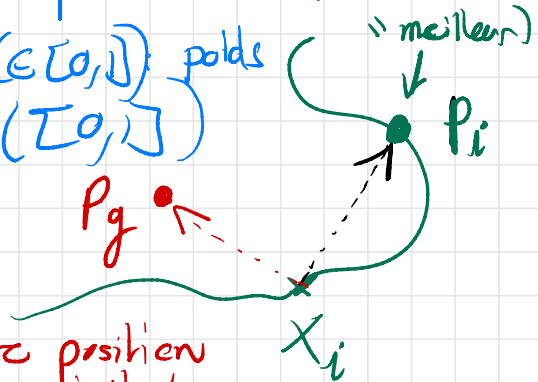
$X_i^{k+1} = X_i^k + \Delta T V_i^k$

La vitesse est calculée par la formule :

$$V_i^{k+1} = \underbrace{w}_{\text{inertie}} V_i^k + \underbrace{C_1 \alpha_1}_{\text{rappel individuel}} (p_i - X_i^k) + \underbrace{C_2 \alpha_2}_{\text{rappel collectif}} (p_g - X_i^k)$$

où $w \in]0, 1[$

$\left\{ \begin{array}{l} p_i : \text{meilleure position individuelle visitée} \\ C_1 \in \mathbb{R} (\in]0, 1[) \text{ poids} \\ \alpha_1 \sim U([0, 1]) \end{array} \right.$

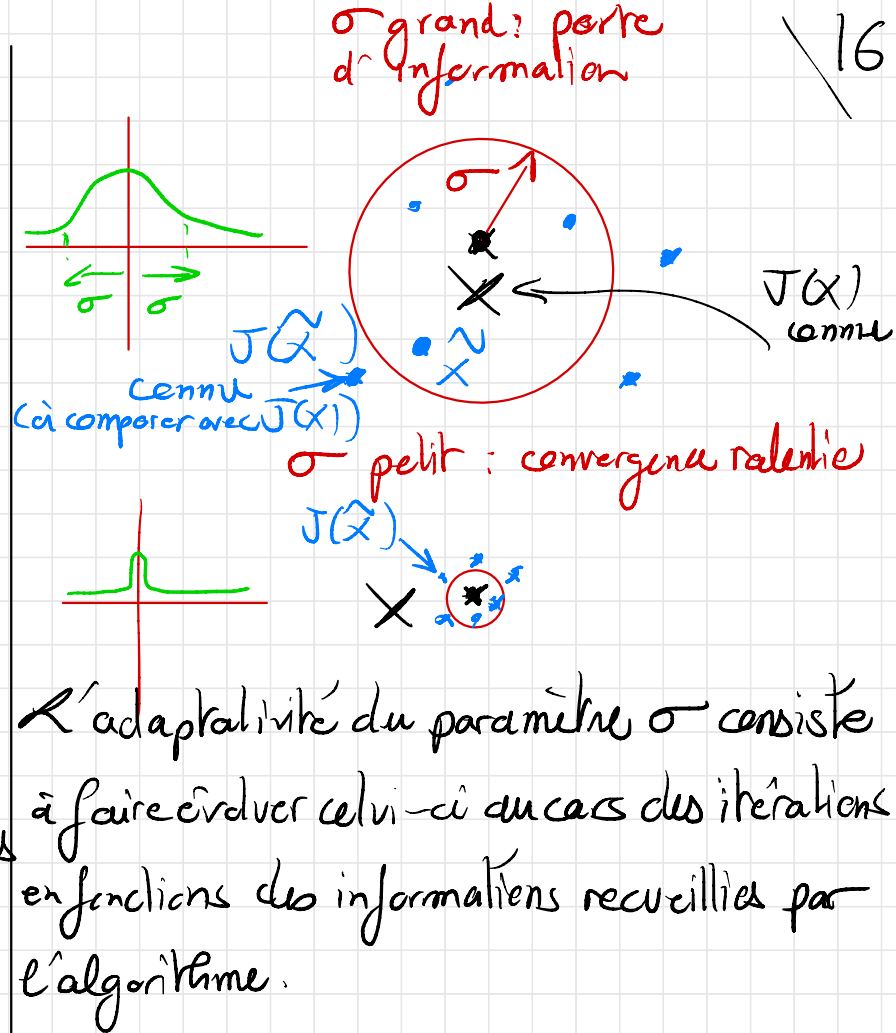


$\left\{ \begin{array}{l} p_g : \text{meilleure position globale visitée} \\ C_2 : \text{poids} \\ \alpha_2 \sim U([0, 1]) \end{array} \right.$

4.5) Amélioration et extensions des méthodes précédentes

4.5.1) Adaptativité des paramètres

Une des difficultés des méthodes précédentes est leur sensibilité au choix de certains paramètres, en particulier pour le recuit simulé, les algorithmes génétiques et les stratégies d'évolution, le choix des paramètres de mutation :



A chaque nouveau $\tilde{X}$, on définit la notation de succès/échec de la mutation :

$\rightarrow$ succès si $J(\tilde{X}) < J(X)$

$\rightarrow$ échec si $J(\tilde{X}) \geq J(X)$

En calculant un taux de succès $\bar{\sigma} \in [0, 1]$

après un certain nombre de mutations effectuées (> 100), on va modifier la valeur de σ en fonction du $\bar{\sigma}$:

$\rightarrow$ si $\bar{\sigma} \geq \bar{\sigma}_0$ (beaucoup de succès)
 $\in [0, 1]$

alors, on augmente σ

$\rightarrow$ si $\bar{\sigma} \leq \bar{\sigma}_0$ (peu de succès)
alors on réduit σ .

En général, on prend

$$\bar{\sigma}_0 = \frac{1}{3}$$

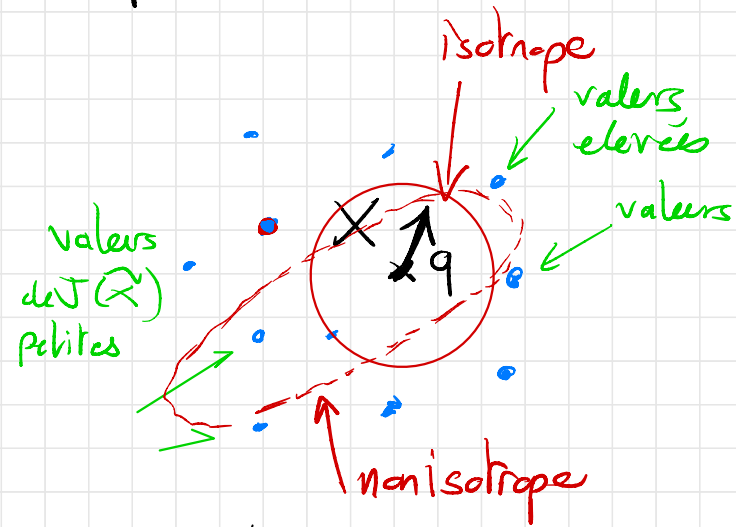
et on parle de la règle des $\frac{1}{3}$

(Exemple : stratégie d'évolution)

On peut remplacer σ par σ^α ou $\frac{\sigma}{\alpha}$ par exemple ($\alpha > 1$) ou utiliser une formule analytique du type :

$$\sigma \rightarrow \exp\left(\frac{1}{3} \cdot \frac{(\bar{\sigma} - \frac{1}{3})}{1 - \frac{1}{3}}\right) \sigma$$

Il existe un 2^{ème} niveau d'adaptativité de la mutation qui tient compte de la valeur $J(\tilde{x})$ mais aussi de la position de $\tilde{x}$:



La mutation adapte non seulement sa "intensité" (valeur de σ)

mais aussi sa forme. Elle devient non isotrope et est définie par une matrice de covariance : $\sigma \text{Id} \rightarrow C \in S_n^{++}(\mathbb{R})$

• Dans le cas d'une stratégie d'évolution, cette adaptativité conduit à la méthode

CMA-ES (gratuit)

(Covariance Matrix Adaptation)

(•) $C \in \mathcal{M}_n(\mathbb{R})$ définie positive

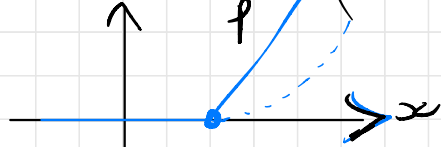
$$q \rightarrow \frac{1}{\sqrt{2\pi} \det(C)} \exp\left(-\frac{1}{2} q^T C q\right) = p(q)$$

4.5.2) Cas de l'optimisation avec contraintes

La méthode de traitement des contraintes la plus fréquemment utilisée est la méthode de pénalisation.

Elle peut être utilisée dans les mêmes conditions et de la même façon qu'au chapitre 3 :

en notant $p : \mathbb{R}^n \rightarrow \mathbb{R}$

$$x \mapsto \begin{cases} 0 & \text{si } x \in \Omega \\ > 0 & \text{sinon} \end{cases}$$


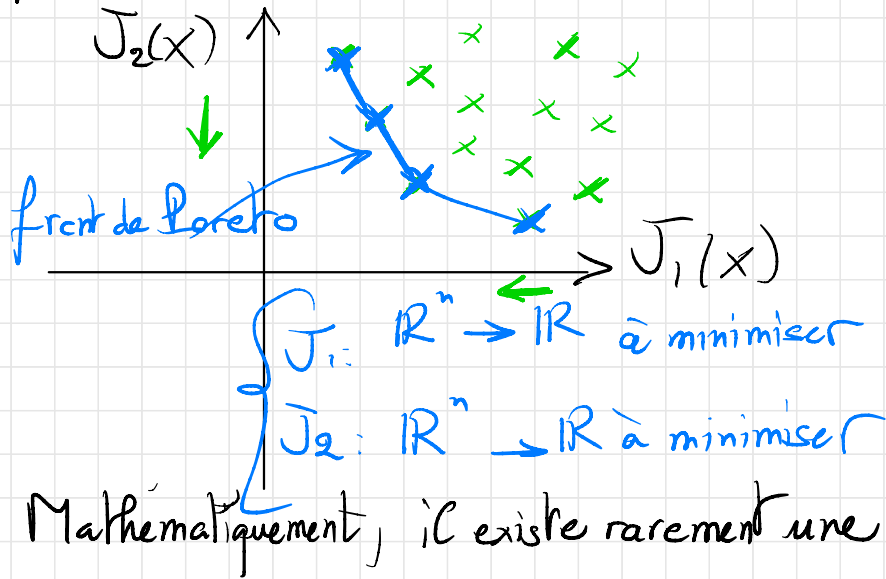
on minimise $x \mapsto J(x) + \rho p(x)$

sur $\mathbb{R}^n$ par la méthode choisie
(SA, GA, ES, PSO...)

Le paramètre ρ doit être
choisi de manière "judicieuse"
(ρ trop grand : pénalisation trop forte
mais perte d'information sur J ;
 ρ trop petit : solution obtenue hors du
domaine admissible)
(voir aussi Exercice TD6)

4.5.3) Cas de l'optimisation multi-objectif

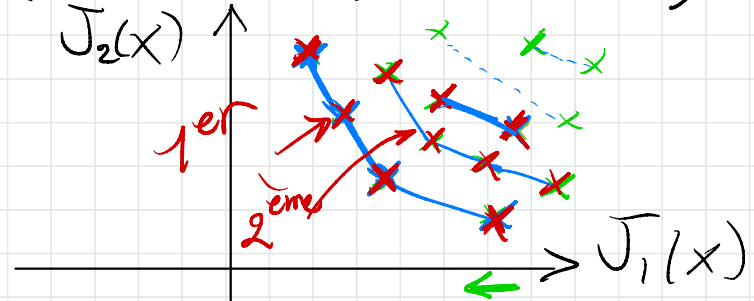
Dans la pratique, il est intéressant de minimiser simultanément plusieurs fonctions (exemple dans l'aéronautique : maximiser la portance, minimiser la traînée)



unique solution : on définit une famille de solutions optimales : le front de Pareto.

Les 3 méthodes basées sur l'évolution d'une famille (GA, ES, PSO) sont parfaitement adaptées à la recherche d'un front de Pareto.

Le seul processus à changer est la sélection dans le cas d'un AG ou d'une ES :



Les éléments sont classés par rapport à leur position sur les fronts de Pareto discrets successifs.

→ A G } multi objectif sans surcôt.
ES }