

SYS843
Réseaux de neurones et
systèmes flous
Séance 11

Eric Granger
Ismail Ben Ayed

Hiver 2017

Sommaire

Optimisation

- 1) Introduction aux algorithmes génétiques
- 2) Optimisation conjointe (un exemple)
 - 1) Optimisation avec multiples critères (exemples)

Algorithmes Génétiques (AG)

- Une classe d'algorithmes probabilistes d'optimisation
- Inspirée par le processus d'évolution biologique
- Utilise des concepts de la sélection naturelle et de l'héritage génétique (Darwin 1859)
- Concepts introduits par John Holland (1975)

Algorithmes Génétiques (AG)

- **Quand est ce que ils sont très utiles:** Quand on a un problème difficile d'optimisation et on n'a pas beaucoup d'information sur la structure de l'espace de solutions
- Ont fait un bon impact en science, génie et business
 - Apprentissage machine (ex., réseaux de neurones)
 - Traitement de signal (ex., conception de filtres)
 - Optimisation combinatoire (ex., problème du voyageur de commerce)
 - Jeux (ex., poker)
 - Etc.

AG: Principe général

- Un AG maintient **une population de solutions (candidats)** au problème
- Évolue la population de solutions en appliquant **itérativement** un ensemble **d'opérateurs stochastiques**

Opérateurs stochastiques

Sélection: Reproduire les solutions qui ont **le plus de succès** dans la population, avec un taux proportionnel à leur **qualité**

Recombinaison: Décomposer deux solutions et, ensuite, mixer leurs parties de façon aléatoire pour former de nouvelles solutions

Mutation: Perturber de façon aléatoire une solution (un candidat)

AG vs. Nature

Algorithme genetique	Nature
Problème d'optimisation	Environnement
Solutions faisables	Individus qui vivent dans l'environnement
Qualité des solutions (fonction 'fitness')	Degré d'adaptation de l'individu à son environnement

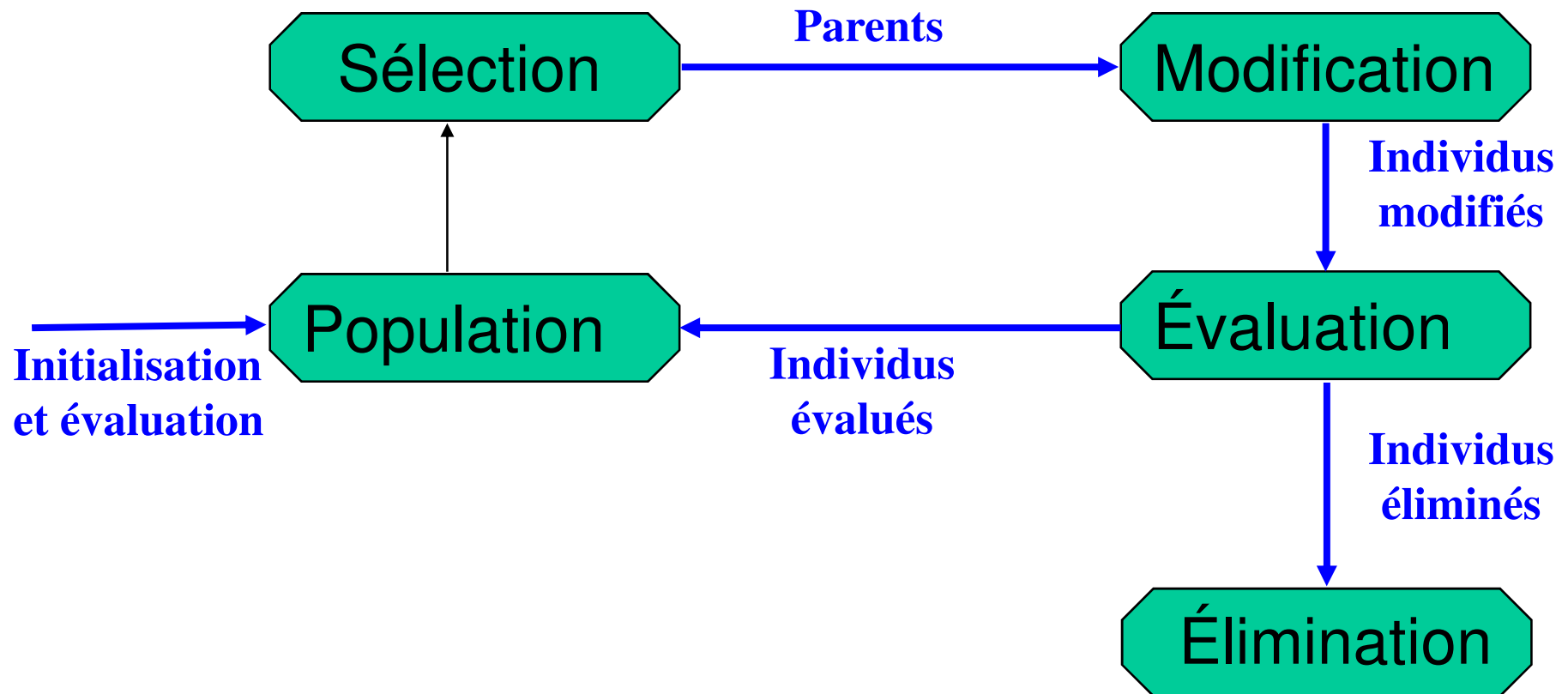
AG vs. Nature

Algorithme genetique	Nature
Un espace de solutions faisables	Une population d'organismes
Opérateurs stochastiques	Sélection, recombinaison et mutation
Itération d'opérateurs stochastiques sur un ensemble de solutions	Évolution des populations pour s'ajuster à leur environnement

AG: Étapes de base

- Initialiser une population d'individus
- Évaluer les individus (fonction 'fitness')
- Répéter jusqu'à ce qu'une condition est vérifiée
 - Sélectionner les meilleurs individus pour reproduction
 - Recombiner l'information des individus
 - Mutation des individus
 - Évaluer les individus modifiées
 - Générer une nouvelle population

Cycle d'évolution



Un exemple: Problème 'MAXONE'

- On veut maximiser le nombre des 1's dans une chaîne de l nombres binaires
- Chaque individu est une chaîne l nombres binaires
- La fonction d'évaluation ('fitness') f d'un candidat (solution possible) est le nombre de 1's dans le code génétique du candidat
- On commence avec une population of n chaînes aléatoires. Pour l'exemple, on suppose que $l = 10$ et $n = 6$.

Un exemple: Problème 'MAXONE'

- **Initialisation**

$$s_1 = 1111010101 \quad f(s_1) = 7$$

$$s_2 = 0111000101 \quad f(s_2) = 5$$

$$s_3 = 1110110101 \quad f(s_3) = 7$$

$$s_4 = 0100010011 \quad f(s_4) = 4$$

$$s_5 = 1110111101 \quad f(s_5) = 8$$

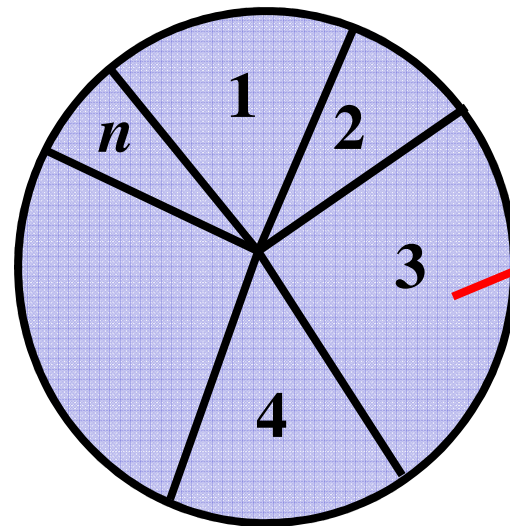
$$s_6 = 0100110000 \quad f(s_6) = 3$$

Un exemple: Problème 'MAXONE'

- **Sélection**

➤ Un individu est sélectionné avec la probabilité:

$$\frac{f(i)}{\sum_i f(i)}$$



*Surface
proportionnelle à la
fonction f*

Un exemple: Problème 'MAXONE'

- **Sélection:** Supposons on obtient la population suivante

$$s_1' = 1111010101 \quad (s_1)$$

$$s_2' = 1110110101 \quad (s_3)$$

$$s_3' = 1110111101 \quad (s_5)$$

$$s_4' = 0111000101 \quad (s_2)$$

$$s_5' = 0100010011 \quad (s_4)$$

$$s_6' = 1110111101 \quad (s_5)$$

Un exemple: Problème 'MAXONE'

- **Recombinaison:** On apparie les chaînes pour recombinaison (*'crossover'*)
- Pour chaque couple, on décide selon une probabilité si on fait la recombinaison
- Supposons qu'on a décidé de faire la recombinaison juste pour les couples (s_1', s_2') et (s_5', s_6')
- Pour chaque couple, on extrait un point de recombinaison (ex., 2 pour le premier couple et 5 pour le deuxième couple)

Un exemple: Problème 'MAXONE'

- **Recombinaison ('crossover')**

Avant recombinaison:

$$s_1' = 1111010101$$

$$s_5' = 0100010011$$

$$s_2' = 1110110101$$

$$s_6' = 1110111101$$

Après recombinaison:

$$s_1'' = 1110110101$$

$$s_5'' = 0100011101$$

$$s_2'' = 1111010101$$

$$s_6'' = 1110110011$$

Un exemple: Problème 'MAXONE'

- **Mutation:** On applique une mutation aléatoire. Pour chaque bit à copier à la nouvelle population, on permet une petite probabilité d'erreur (ex., 0.05)

Avant mutation

$s_1'' = 11101\textcolor{red}{1}0101$

$s_2'' = 1111\textcolor{red}{0}10101\textcolor{red}{1}$

$s_3'' = 11101\textcolor{red}{1}11\textcolor{red}{0}1$

$s_4'' = 0111000101$

$s_5'' = 0100011101$

$s_6'' = 11101100\textcolor{red}{1}1$

Après mutation

$s_1''' = 11101\textcolor{red}{0}0101 \quad f(s_1''')=6$

$s_2''' = 1111\textcolor{red}{1}1010\textcolor{red}{0} \quad f(s_2''')=7$

$s_3''' = 11101\textcolor{red}{0}11\textcolor{red}{1}1 \quad f(s_3''')=8$

$s_4''' = 0111000101 \quad f(s_4''')=5$

$s_5''' = 0100011101 \quad f(s_5''')=5$

$s_6''' = 11101100\textcolor{red}{0}1 \quad f(s_6''')=6$

Un exemple: Problème 'MAXONE'

- **Fin de l'exemple:** En une génération, le total de la fonction d'évaluation ('fitness') a changé de 34 à 37 (amélioration de 9%)
- Maintenant, on répète cette procédure jusqu'à ce que un critère d'arrêt est satisfait

Composantes d'un AG

- Définition du problème
- Principes de codages (gène, chromosome)
- Procédure d'initialisation (création)
- Sélection de parents (reproduction)
- Opérateurs génétiques (mutation, recombinaison)
- Fonction d'évaluation (environnement)
- Condition d'arrêt

Possibilités de codage d'individus:

- Bit chaines de bits (0101 ... 1100)
- Nombres réels (0.2 -1.1 ... 0.9 19.4)
- Listes de règles (R1 R2 ... R11 R12)
- Éléments d'un programme (programmation génétique)
- N'importe quelle structure de données

Représentation: Codage

Le choix d'une méthode de représentation doit:

- Utiliser une structure de données proche de la représentation naturelle
- Se baser sur des opérateurs génétiques qui ne donnent pas des solutions non faisables.

Initialisation

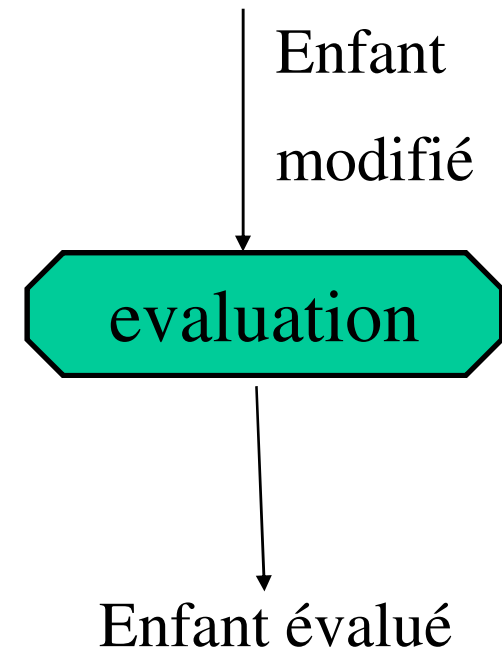
- Commencer avec une population d'individus générés de façon aléatoire ou utiliser:
 - Une population existante (sauvegardée au passé)
 - Un ensemble de solutions données par un expert humain
 - Un ensemble de solutions obtenues par un autre algorithme heuristique

Sélection

- **Objectif:** Focaliser la recherche sur les régions les plus prometteuses dans l'espace des solutions
- **Inspiration:** “Survie du plus fort” de Darwin

Fonction d'évaluation

- Le choix de la fonction d'évaluation ('fitness') est crucial: La solution dépend de ce choix.
- Choisir une fonction appropriée est la partie la plus difficile dans le design d'un AG
- Les solutions qui sont codées de la même façon doivent avoir la même fonction d'évaluation
- Le seul lien AG/problème



Recombinaison ('Crossover')

P1	(0 1 1 0 1 0 0 0)	(0 1 0 0 1 0 0 0)	E1
P2	(1 1 0 1 1 0 1 0)	(1 1 1 1 1 0 1 0)	E2

- **Aspect très important d'un AG:** Permet au processus d'évolution de bouger vers les régions les plus prometteuses dans l'espace de recherche
- Apparie les solutions partielles des bon parents pour construire des nouvelles solutions qui sont meilleures
- Accélère la recherche au début du processus

Mutation

Before: (1 0 1 1 0 1 1 0)

After: (0 1 1 0 0 1 1 0)

Before: (1.38 -69.4 326.44 0.1)

After: (1.38 -67.5 326.44 0.1)

- Mouvement dans l'espace de recherche (local or global)
- Retrouve l'information perdue

Conditions d'arrêt

Exemples:

- Un nombre prédéterminé de générations ou un certain temps algorithmique
- Une solution satisfaisante a été obtenue
- Pour un certain nombre de générations, il n'y a pas d'amélioration dans la solution obtenue

Un autre exemple

Problème du voyageur de commerce: The Traveling Salesman Problem (TSP)

- Le voyageur de commerce doit visiter chaque ville dans son territoire exactement une seule fois et puis retourner au point de départ.
- Étant donné le coût de voyage entre les villes, il doit planifier son itinéraire de façon à minimiser le coût

Représentation

- La représentation ici est une liste ordonnée de nombres de ville (AG basé sur l'ordre)

1) London 3) Dunedin 5) Beijing 7) Tokyo
2) Vénice 4) Singapore 6) Phoenix 8) Victoria
9) Toronto

Liste de villes 1 (3 5 7 2 1 6 4 8)

Liste de villes 2 (2 5 7 6 8 1 3 4)

Initialisation et sélection

- Un vecteur $v = (i_1 i_2 \dots i_n)$ représente un tour (v est une permutation de $\{1, 2, \dots, n\}$)
- La fonction d'évaluation ('Fitness') f d'une solution est l'inverse du coût de la solution
- **Initialisation:** On peut (par exemple) utiliser un échantillon aléatoire de permutations of $\{1, 2, \dots, n\}$
- **Sélection:** proportionnelle à la fonction d'évaluation

Recombinaison ('crossover')

- Construire des enfants en se basant sur:
 - Choisir une sous-séquence du premier parent
 - Préserver l'ordre relatif des villes dans le deuxième parent et s'assurer que les solution obtenues sont faisables

Exemple:

$$p_1 = (1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9)$$

$$o_1 = (x \ x \ x \ 4 \ 5 \ 6 \ 7 \ x \ x)$$

$$p_2 = (4 \ 5 \ 2 \ 1 \ 8 \ 7 \ 6 \ 9 \ 3)$$

$$o_2 = (x \ x \ x \ 1 \ 8 \ 7 \ 6 \ x \ x)$$

Recombinaison ('crossover')

À partir point de coupure ('cut point') d'un parent, les villes de l'autre parent sont copiées en respectant le même ordre

➤ La séquence de villes dans le deuxième parent est:

9 – 3 – 4 – 5 – 2 – 1 – 8 – 7 – 6

➤ En enlevant les villes du premier enfant, on obtient:

9 – 3 – 2 – 1 – 8

➤ Cette séquence est placée dans le premier enfant (On fait pareil pour le deuxième):

$o_1 = (2 \ 1 \ 8 \ 4 \ 5 \ 6 \ 7 \ 9 \ 3)$

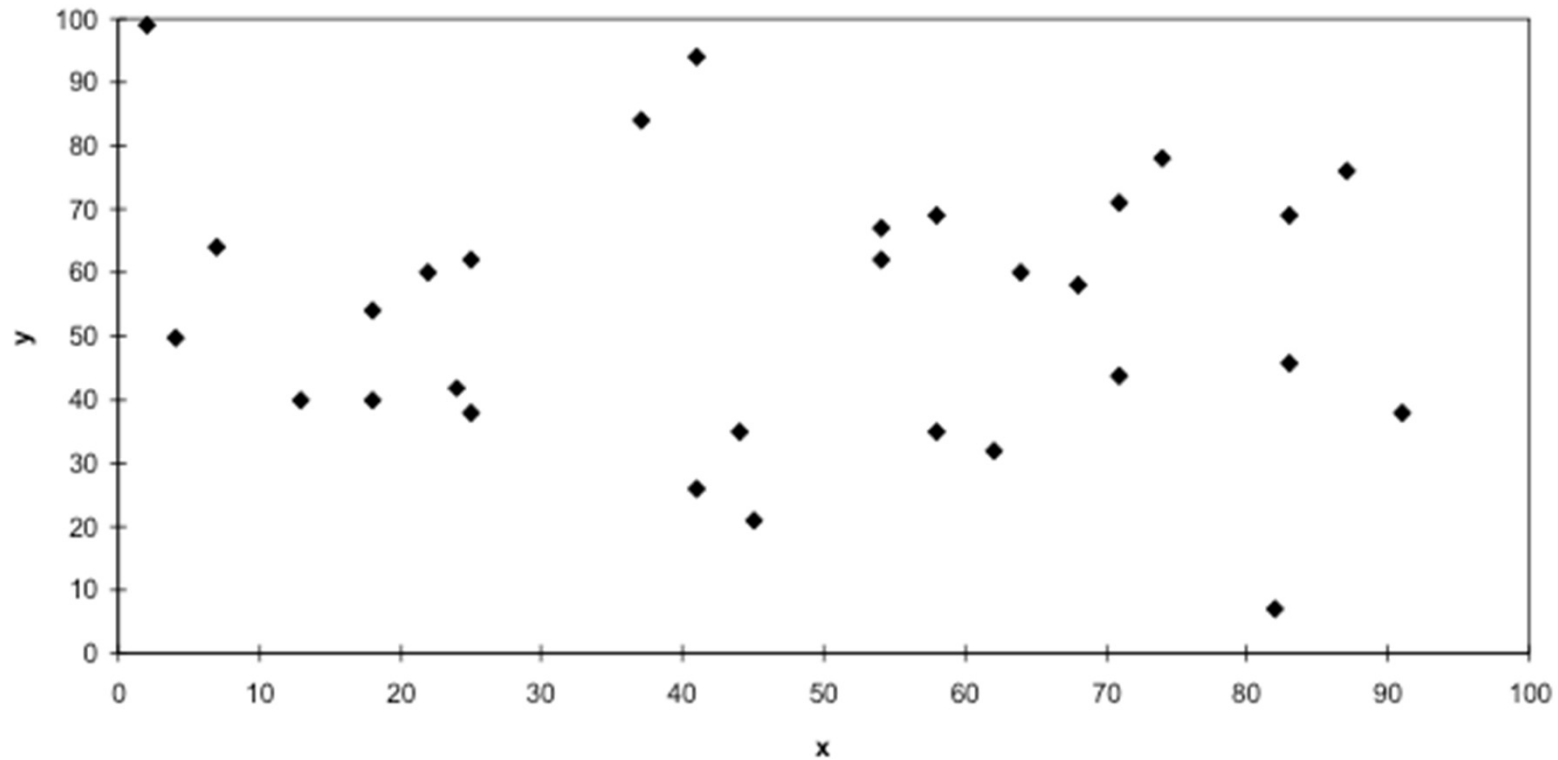
$o_2 = (3 \ 4 \ 5 \ 1 \ 8 \ 7 \ 6 \ 9 \ 2)$

Inversion

- Un sous-ensemble entre deux points sélectionnées de façon aléatoire est inversé
- Exemple:

(1 2 3 4 5 6 7 8 9) est changé à (1 2 7 6 5 4 3 8 9)

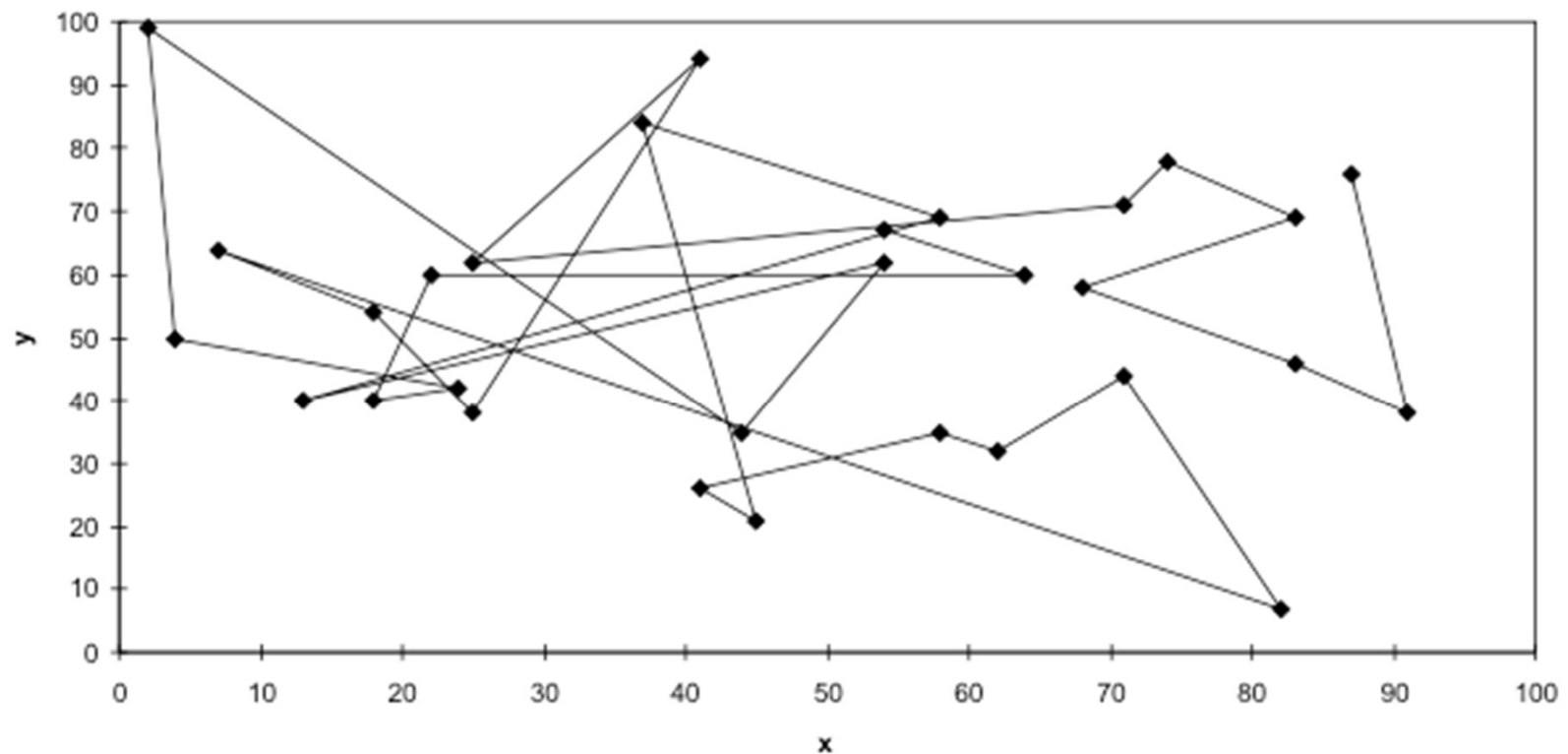
Exemple: 30 villes



Source: Wendy Williams's slides

Exemple: 30 villes

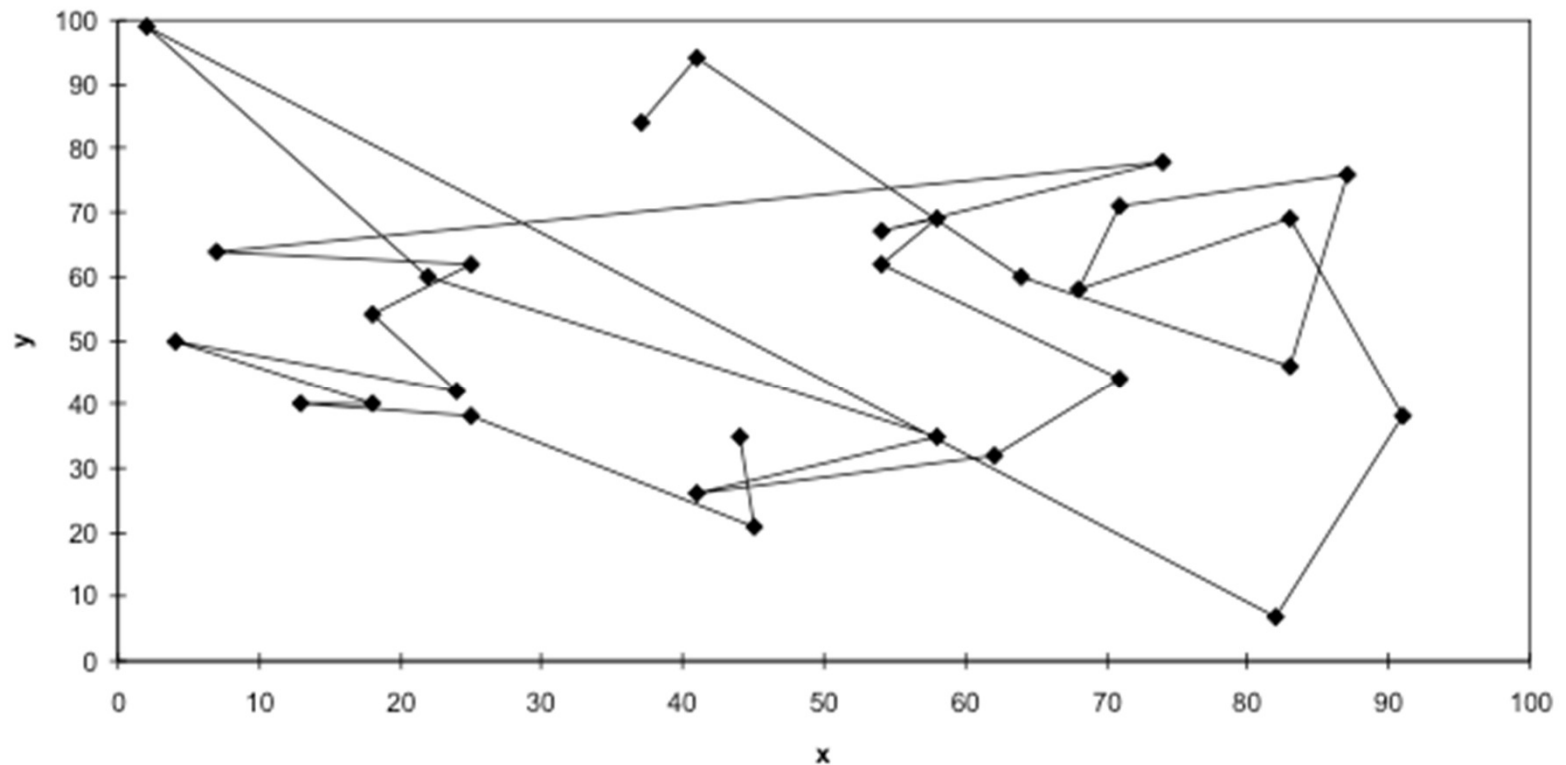
Solution $_i$ (Distance = 941)



Source: Wendy Williams's slides

Exemple: 30 villes

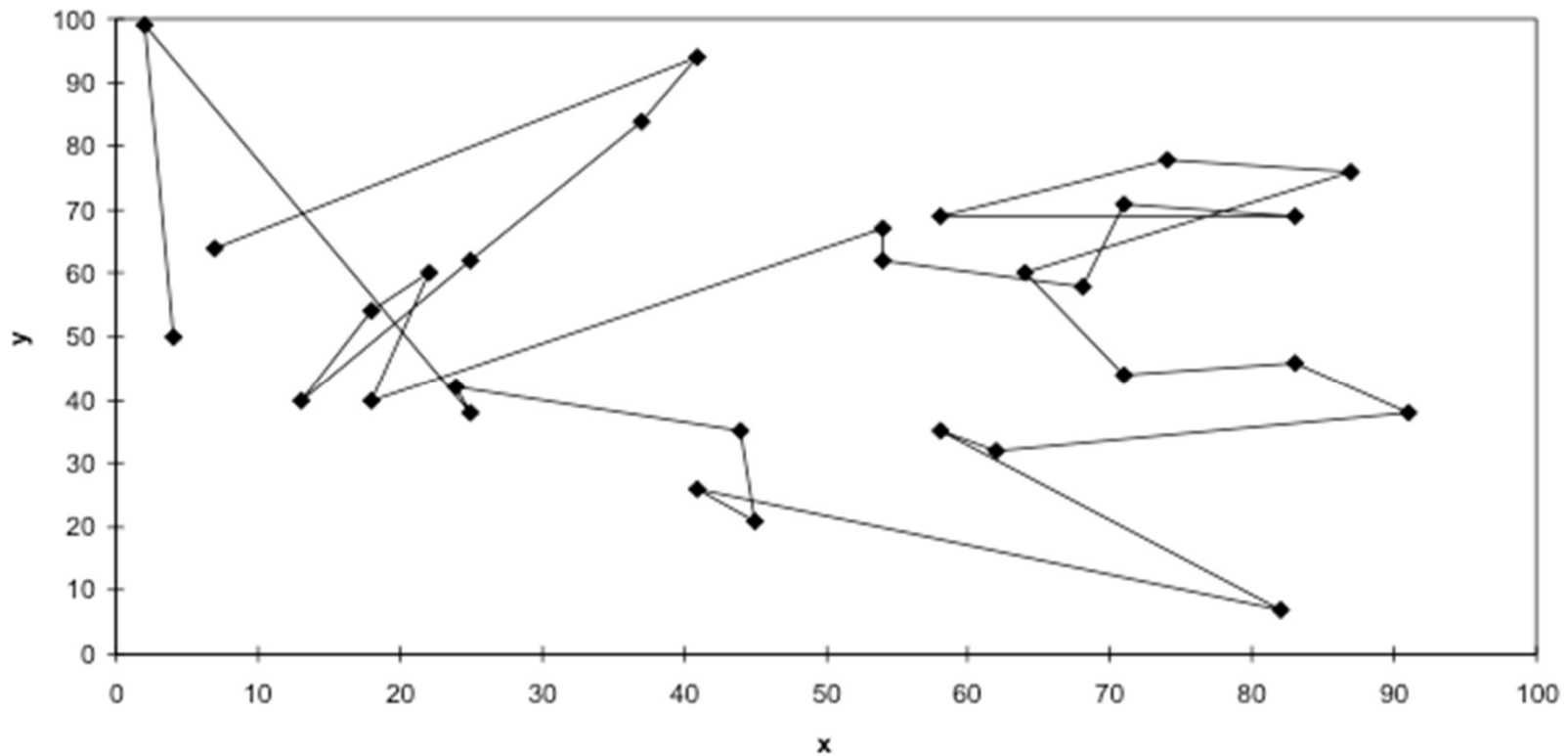
Solution j (Distance = 800)



Source: Wendy Williams's slides

Exemple: 30 villes

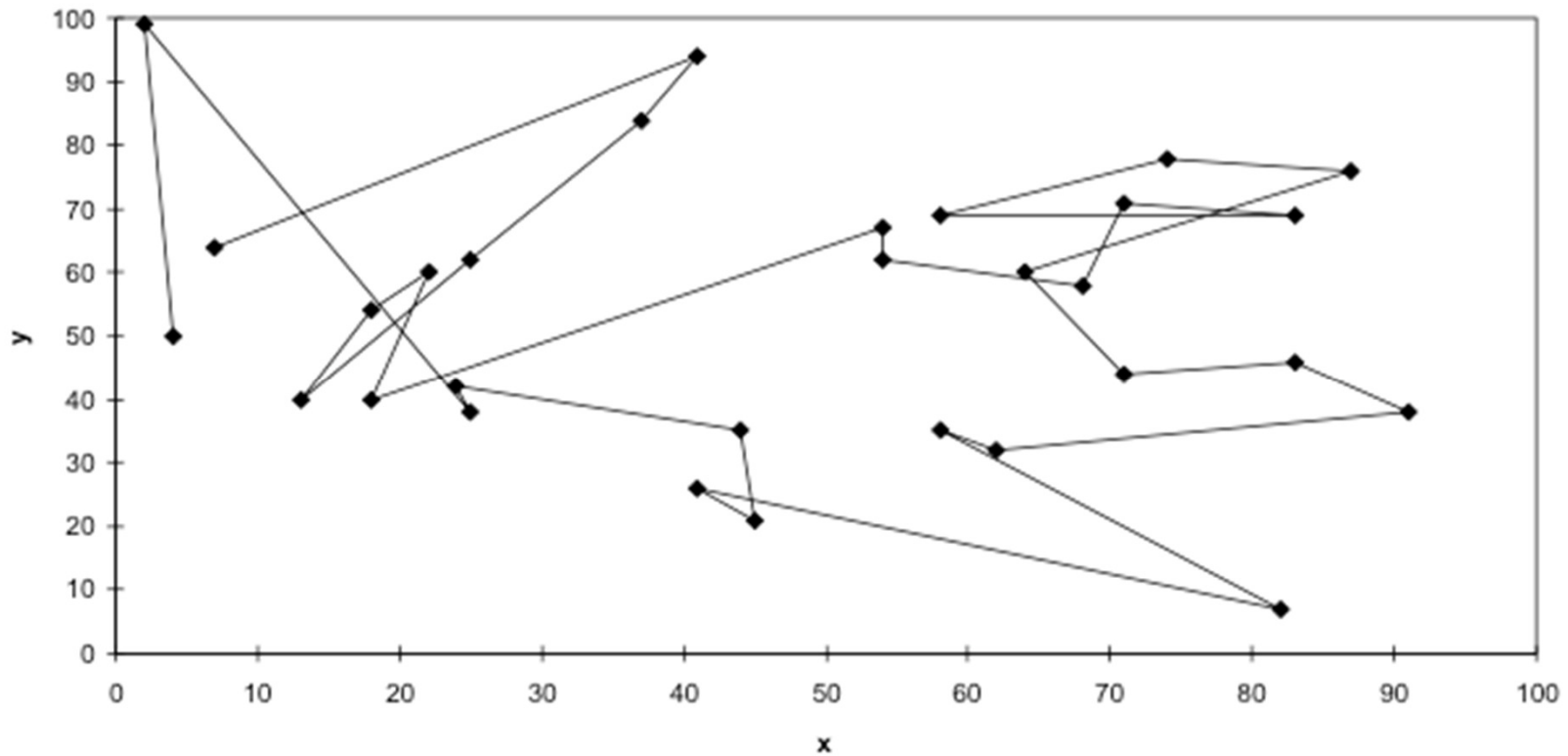
Solution $_k$ (Distance = 652)



Source: Wendy Williams's slides

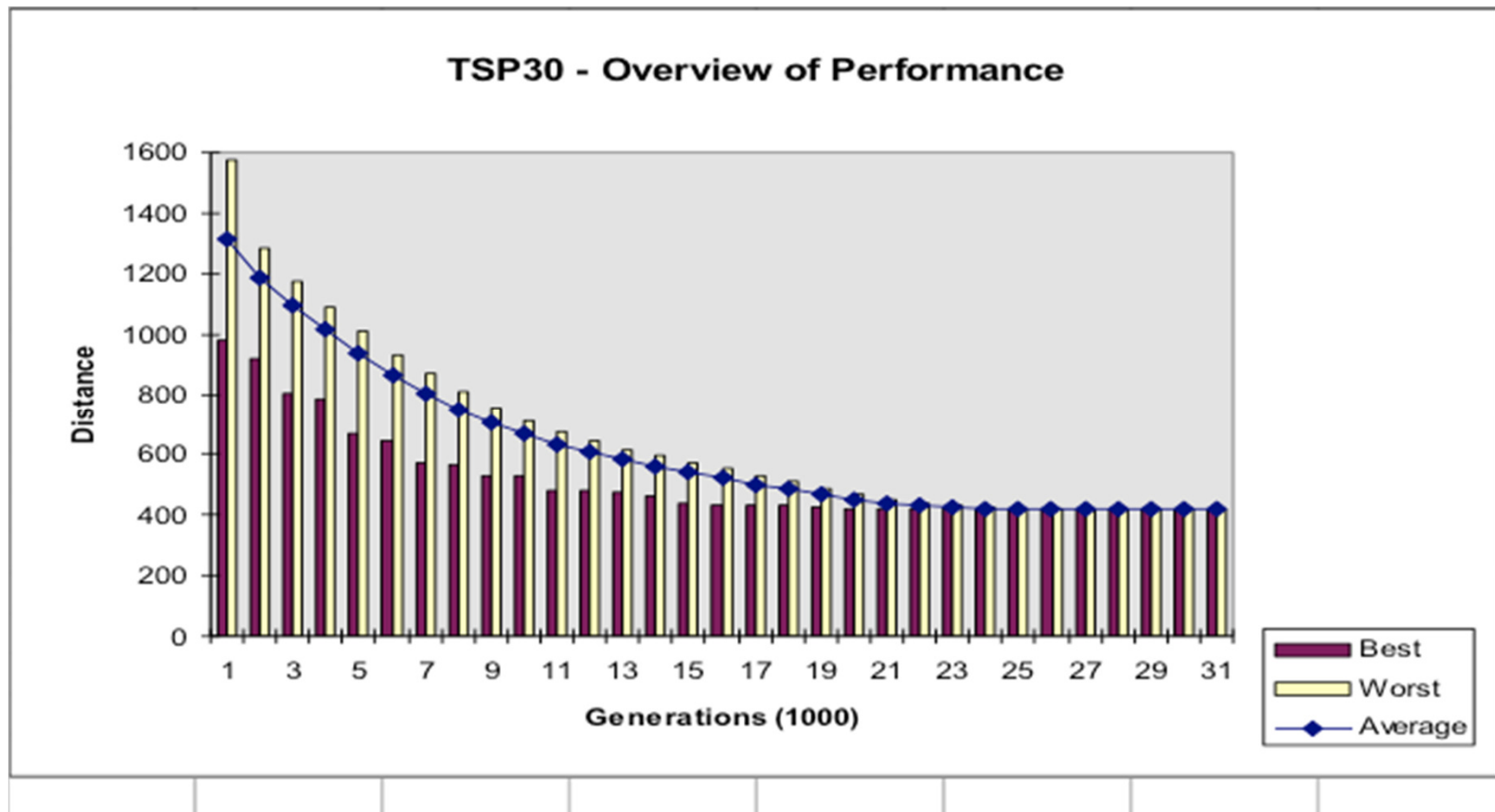
Exemple: 30 villes

Meilleure solution (Distance = 420)



Source: Wendy Williams's slides

Performance



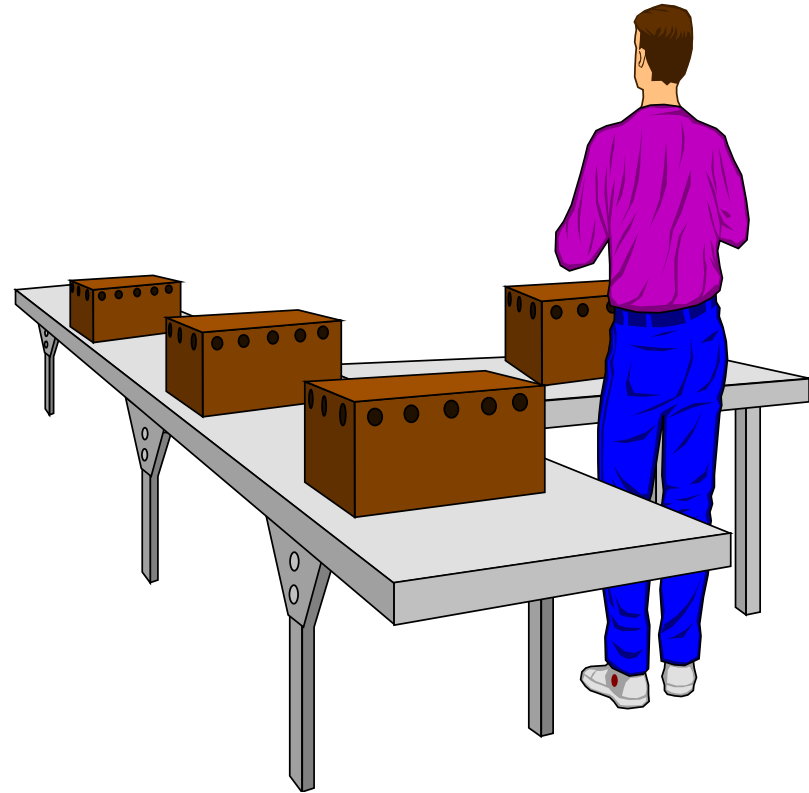
Source: Wendy Williams's slides

À propos des AGs!

“Almost eight years ago ... people at Microsoft wrote a program [that] uses some genetic things for finding short code sequences. Windows 2.0 and 3.2, NT, and almost all Microsoft applications products have shipped with pieces of code created by that system.”

□

- Nathan Myhrvold, Microsoft Advanced Technology Group, Wired, September 1995



Source: Wendy Williams's slides

AGs: Avantages

- Concepts faciles à comprendre
- Independent de l'application
- Peut être appliquer à l'optimisation avec multiples critères
- Peut aider dans des environnements bruités
- On a toujours une réponse, et la réponse s'améliore toujours avec le temps
- Peut être parallélisé
- C'est facile d'exploiter des solutions existantes

On utilise les AGs

- Quand les autres solutions sont très lentes ou très compliquées
- Quand le problème est similaire à un problème qui a été bien adressé par les AGs

Sommaire

Optimisation

- 1) Introduction aux algorithmes génétiques
- 2) Optimisation conjointe (un exemple)
 - 1) Optimisation avec multiples critères (exemples)

K-means: Rappel

1. **Partition:** Assigne la catégorie la plus *proche* à chaque
2. **Apprentissage:** Mise-à-jour du prototype de chaque catégorie

$$E(\mathbf{S}, \mu_1, \mu_2) = \sum_{\mathbf{S}} \|x_p - \mu_1\|^2 + \sum_{\bar{\mathbf{S}}} \|x_p - \mu_2\|^2$$

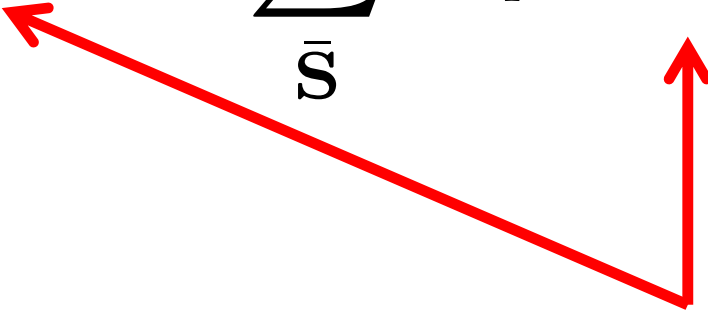
Variables de différente nature:

Optimisation conjointe?

K-means: Problème non-linéaire

1. Partition: Assigne la catégorie la plus *proche* à chaque

2. Apprentissage: Mise-à-jour du prototype de chaque catégorie

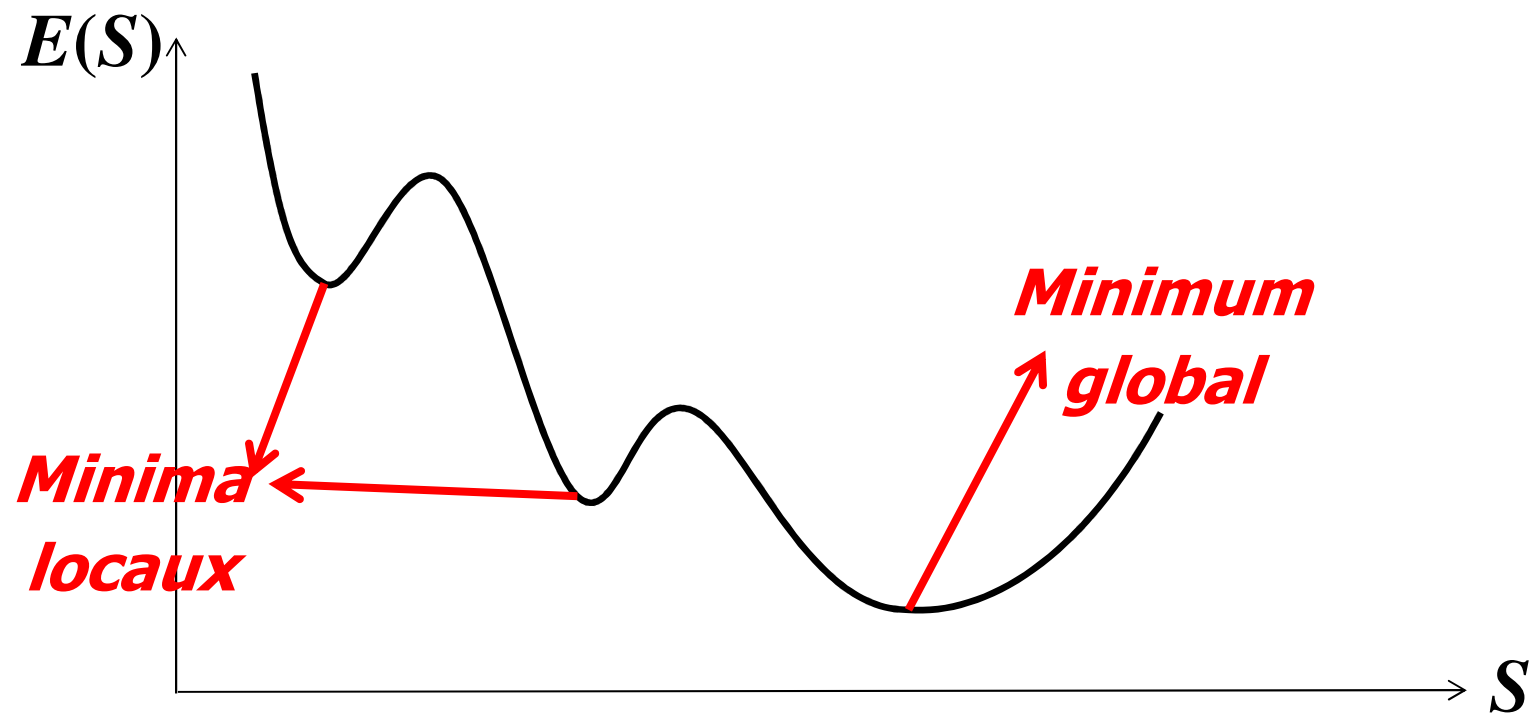
$$E(\mathbf{S}) = \sum_{\mathbf{S}} \|x_p - \mu_{\mathbf{S}}\|^2 + \sum_{\bar{\mathbf{S}}} \|x_p - \mu_{\bar{\mathbf{S}}}\|^2$$


On peut réécrire le problème avec une seule
variable

Valeurs optimales!

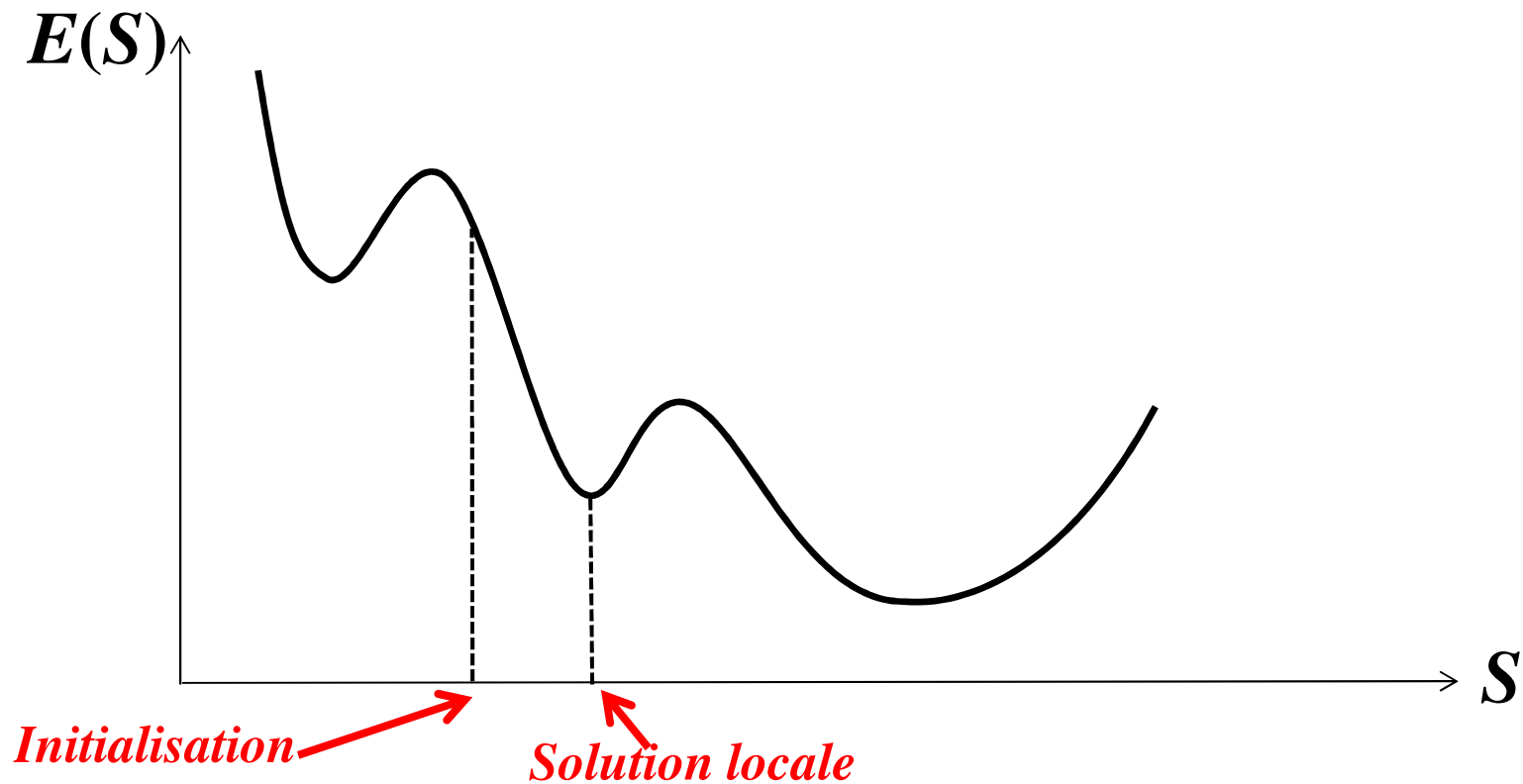
Optimum global?

- *K-means ne garantit pas d'obtenir un minimum global*



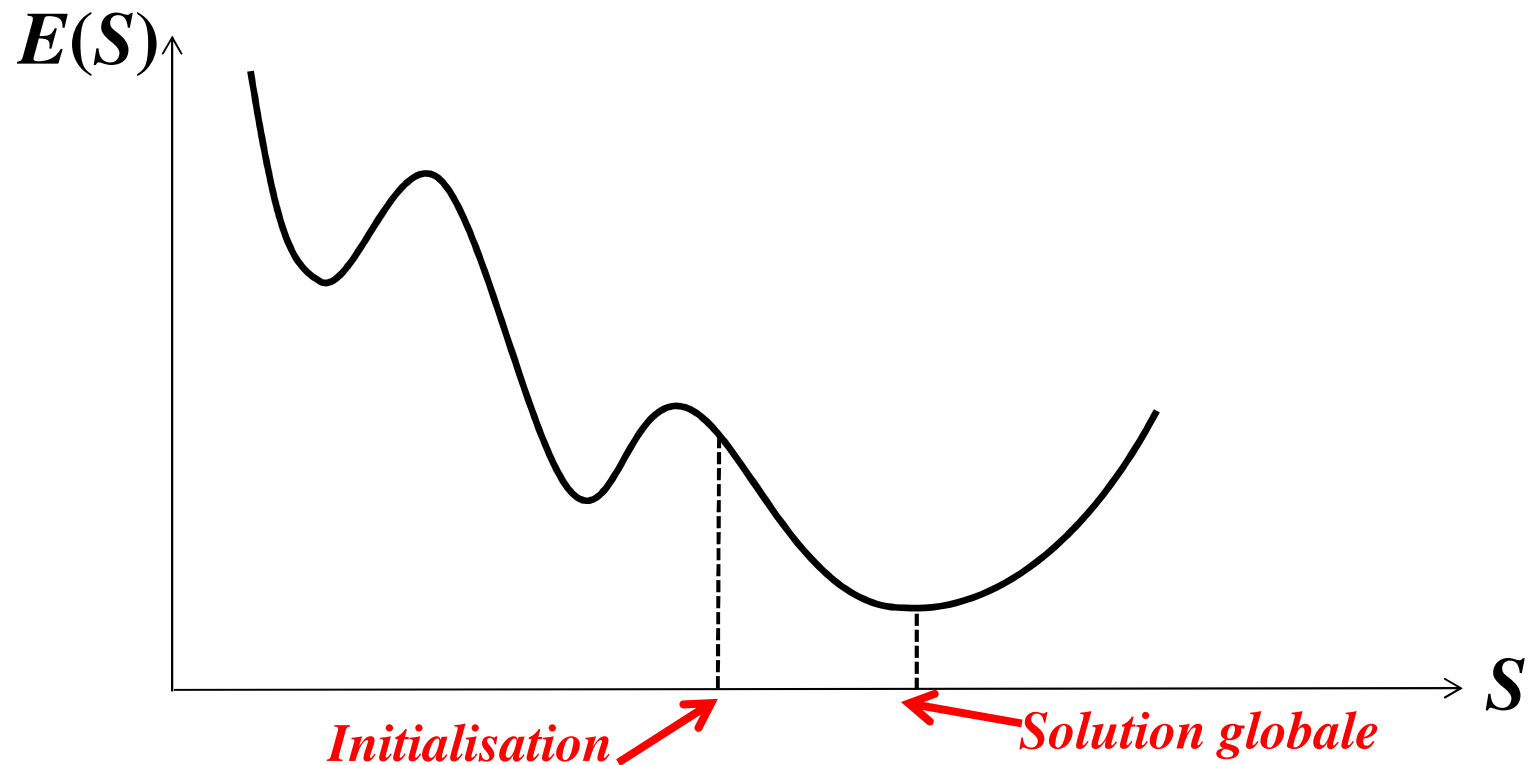
Optimum global?

- *K-means ne garantit pas d'obtenir un minimum global*



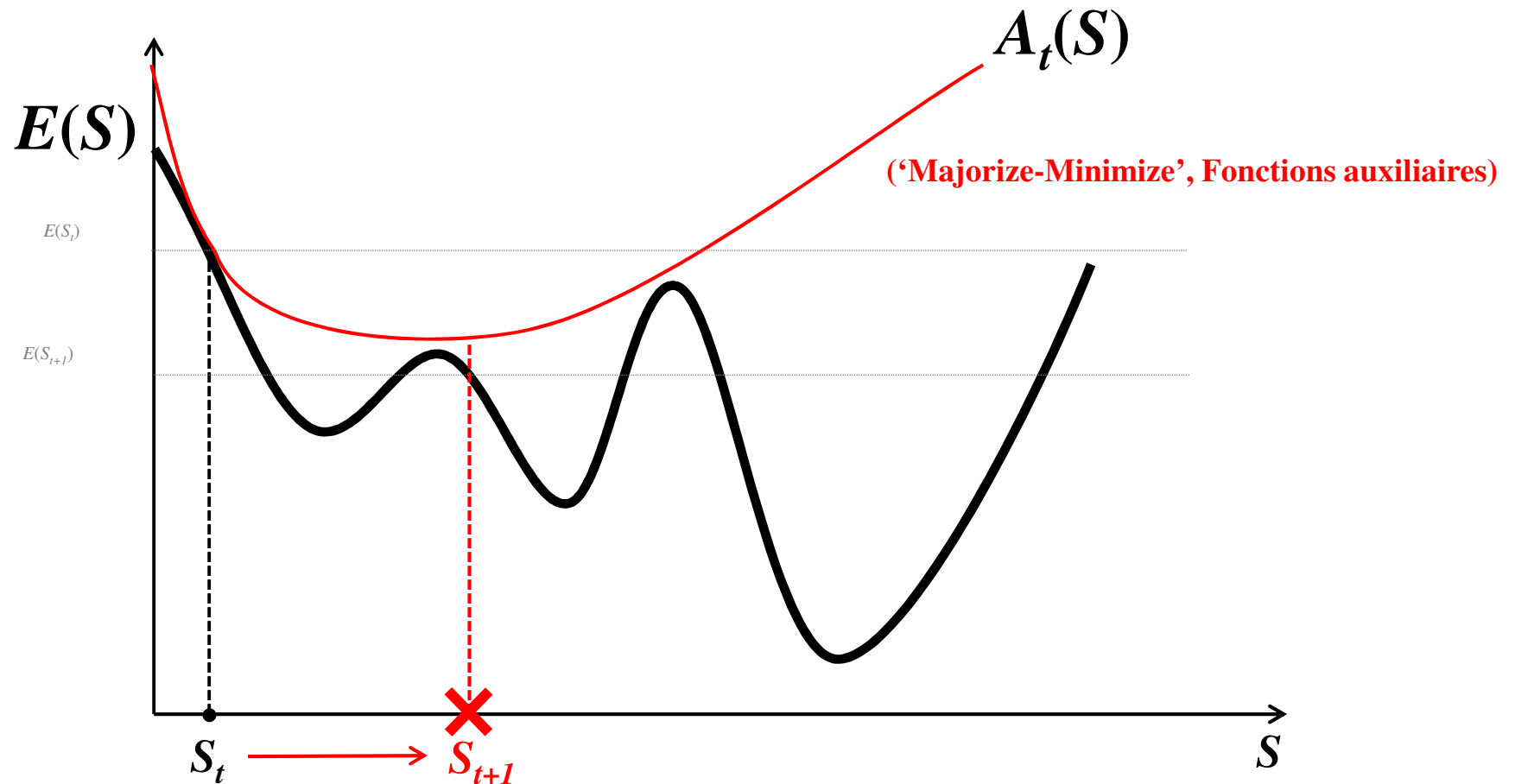
Optimum global?

- *K-means ne garantit pas d'obtenir un minimum global*



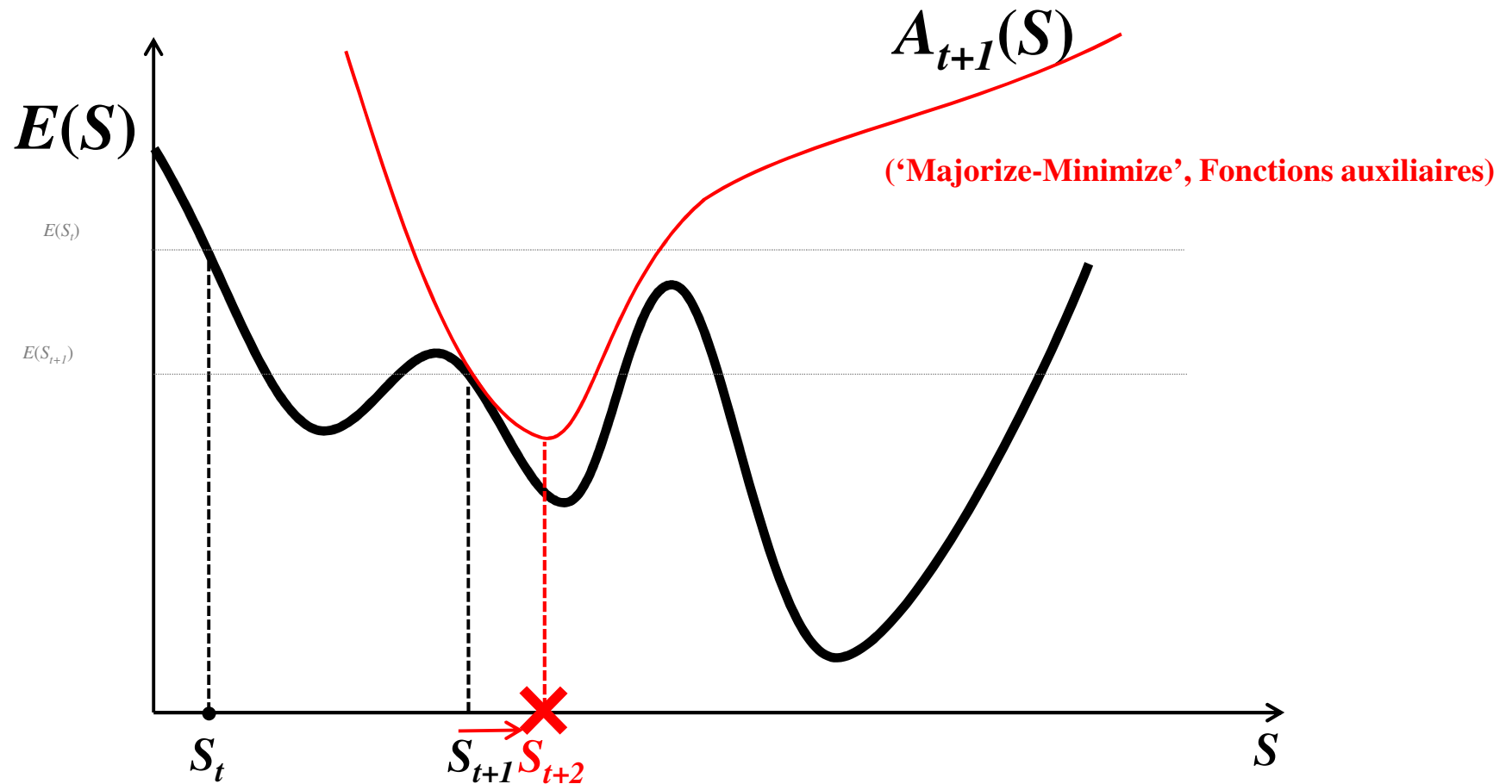
Optimum global?

Pourquoi: Optimisation de fonction auxiliaire



Optimum global?

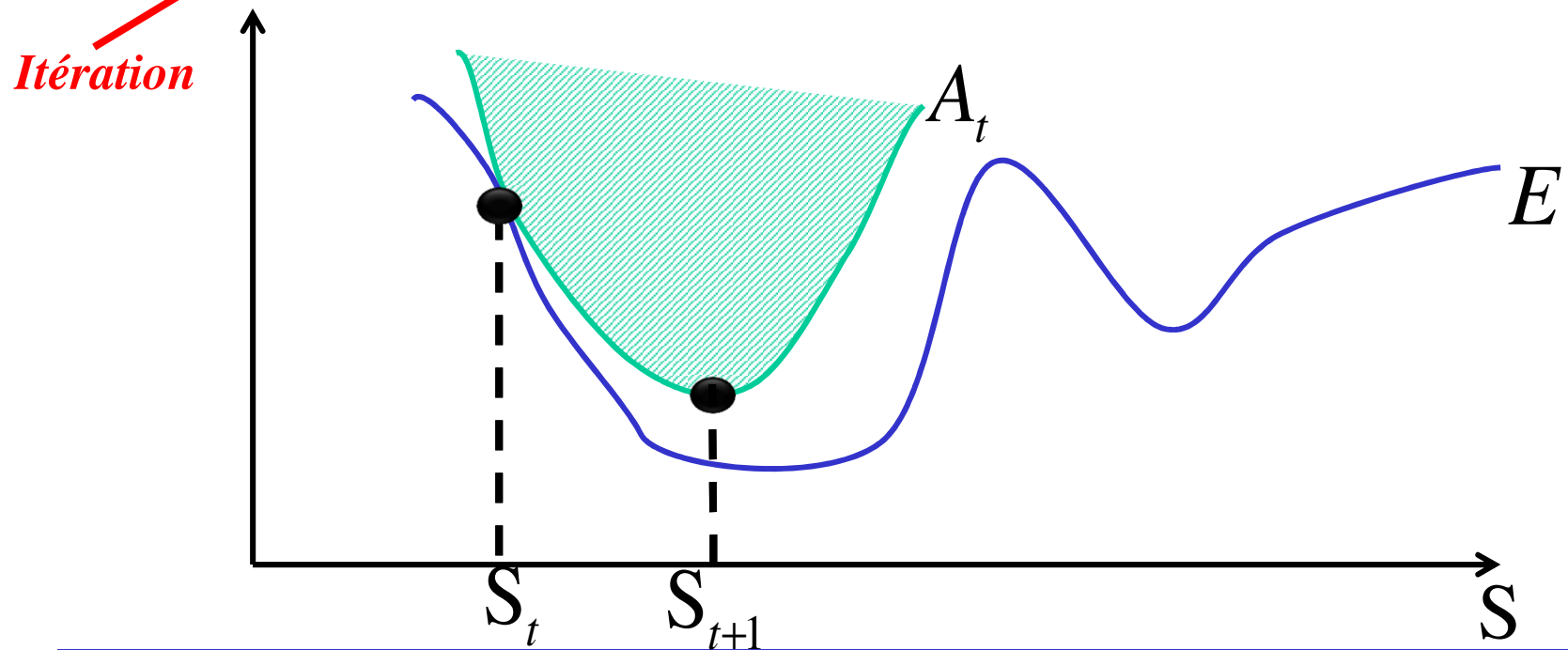
Pourquoi: Optimisation de fonction auxiliaire



Optimum global?

Pourquoi: Optimisation de fonction auxiliaire

$$S_{t+1} = \min_S A_t(S) \geq E(S)$$

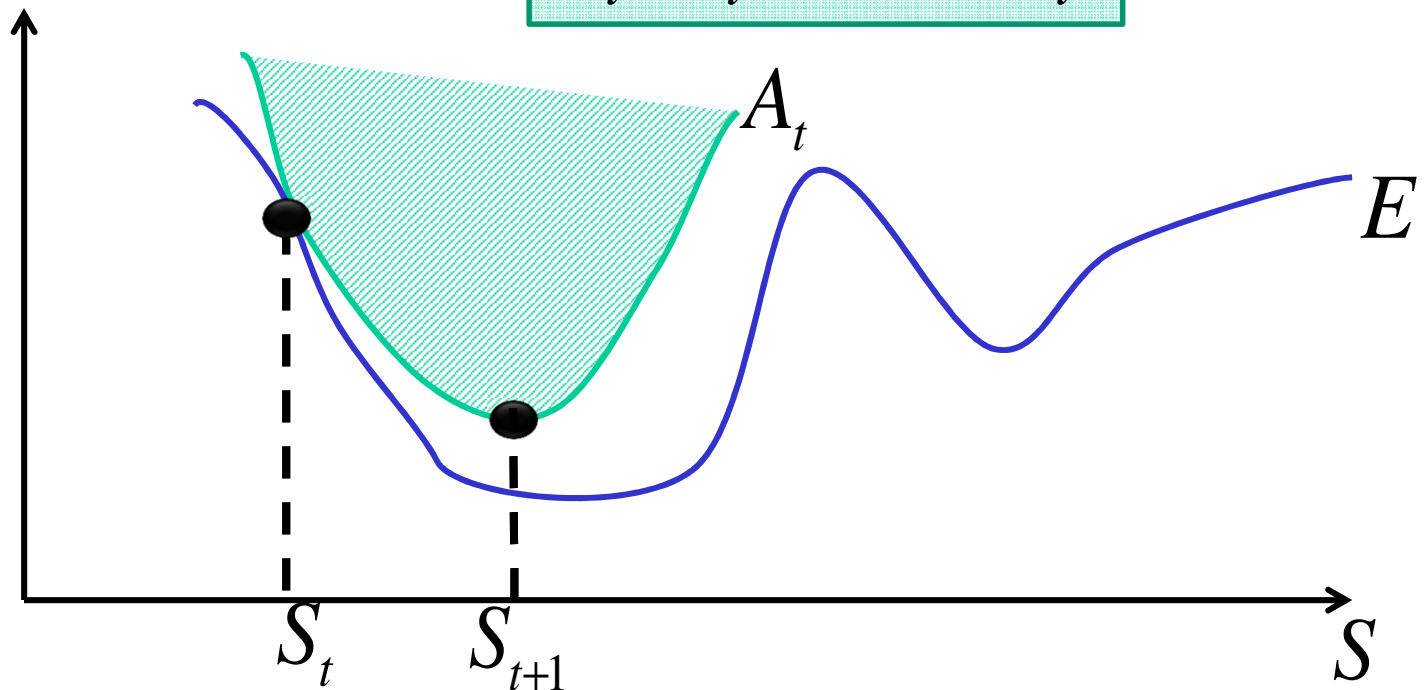


Optimum global?

Pourquoi: Optimisation de fonction auxiliaire

$$S_{t+1} = \min_S \begin{cases} A_t(S) \geq E(S) \\ A_t(S_t) = E(S_t) \end{cases}$$

Itération



Optimum global?

Pourquoi: Optimisation de fonction auxiliaire

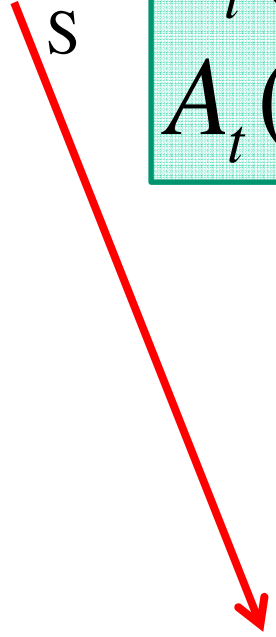
$$S_{t+1} = \min_S \begin{cases} A_t(S) \geq E(S) \\ A_t(S_t) = E(S_t) \end{cases}$$


$$E(S_{t+1}) \leq A_t(S_{t+1})$$

Optimum global?

Pourquoi: Optimisation de fonction auxiliaire

$$S_{t+1} = \min_S \begin{array}{l} A_t(S) \geq E(S) \\ A_t(S_t) = E(S_t) \end{array}$$


$$E(S_{t+1}) \leq A_t(S_{t+1}) \leq A_t(S_t)$$

Optimum global?

Pourquoi: Optimisation de fonction auxiliaire

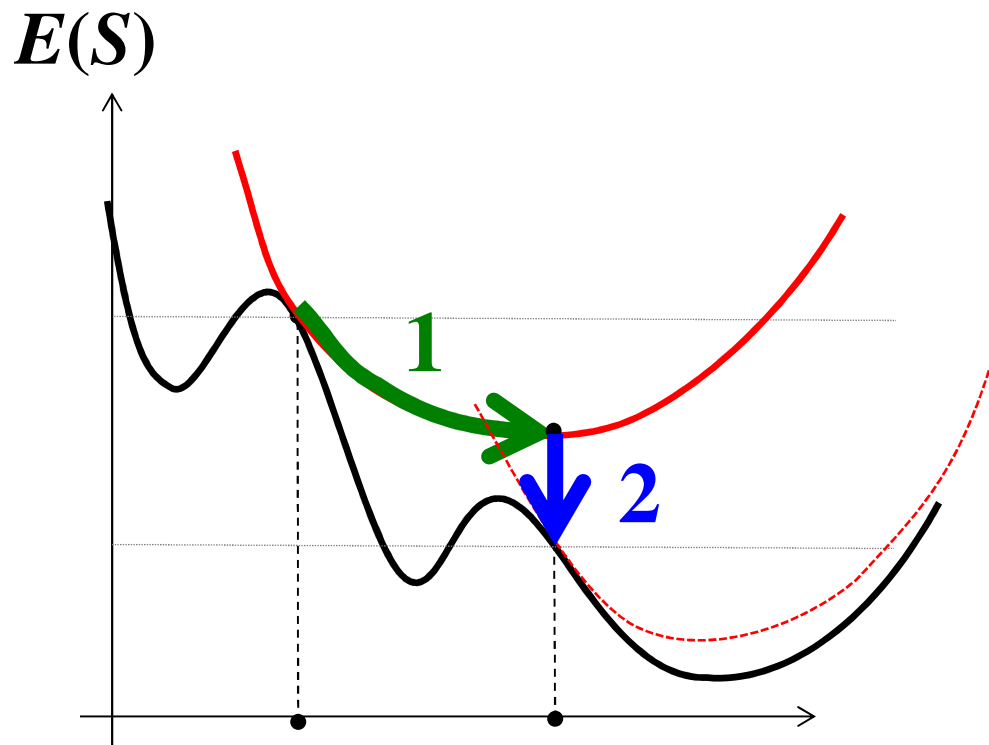
$$S_{t+1} = \min_S \begin{cases} A_t(S) \geq E(S) \\ A_t(S_t) = E(S_t) \end{cases}$$

$$E(S_{t+1}) \leq A_t(S_{t+1}) \leq A_t(S_t) = E(S_t)$$

Fonction coût est diminuée à chaque itération

K-means optimise une fonction aux.?

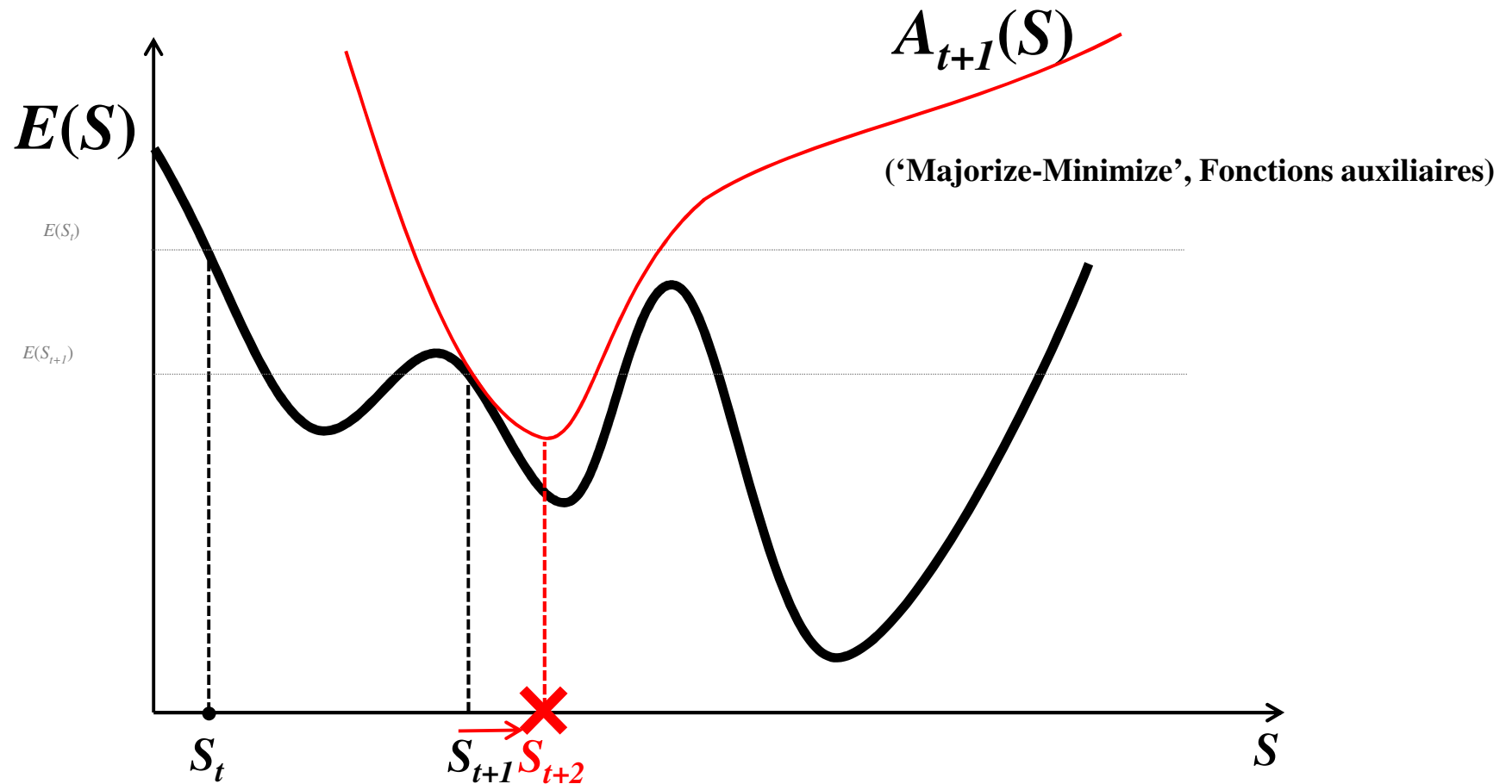
1. **Partition:** Assigne la catégorie la plus *proche* à chaque
2. **Apprentissage:** Mise-à-jour du prototype de chaque catégorie



[Tang *et al.*, 2014]

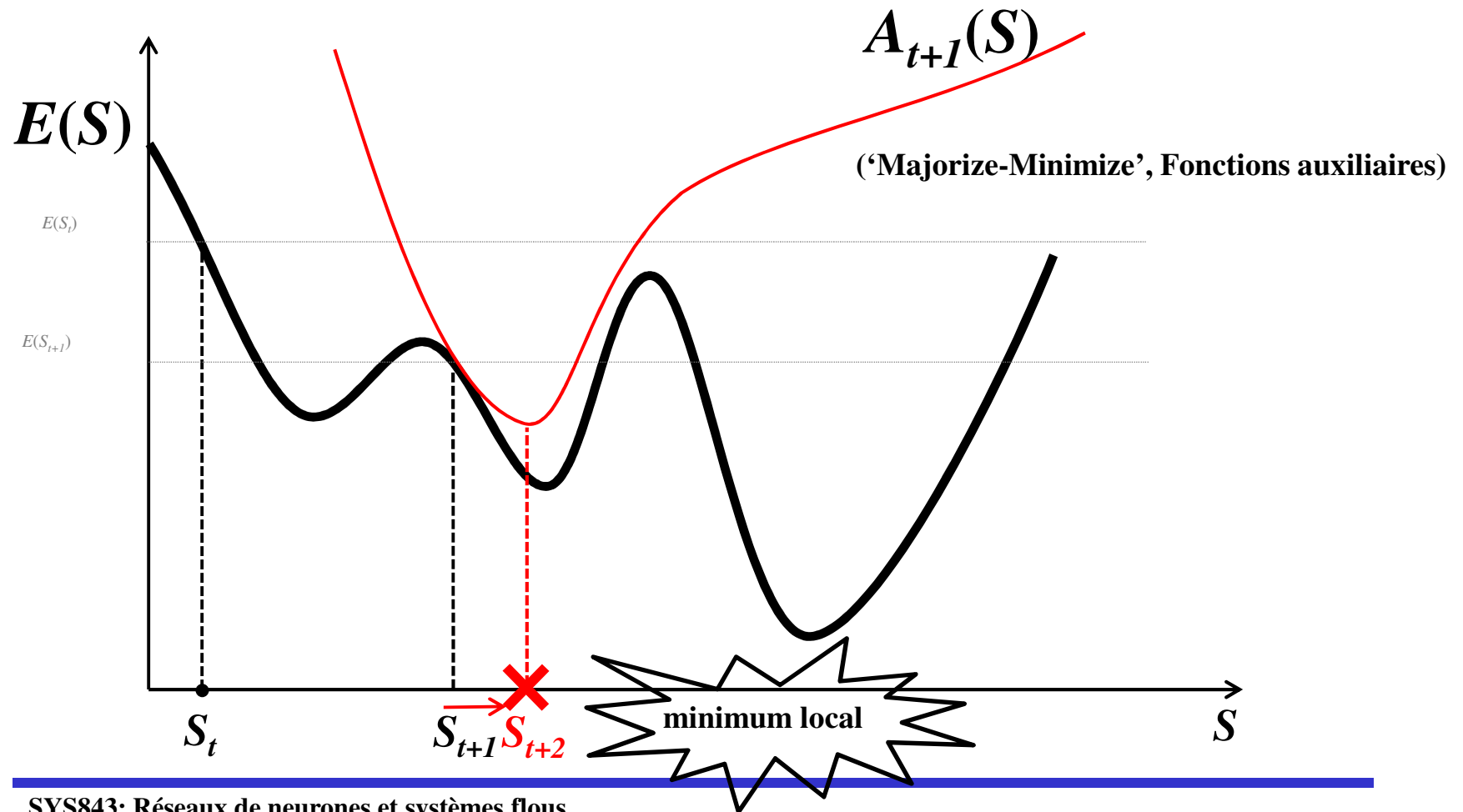
EM/K-means: Fonctions aux.

Pas de garanti d'optimum global!

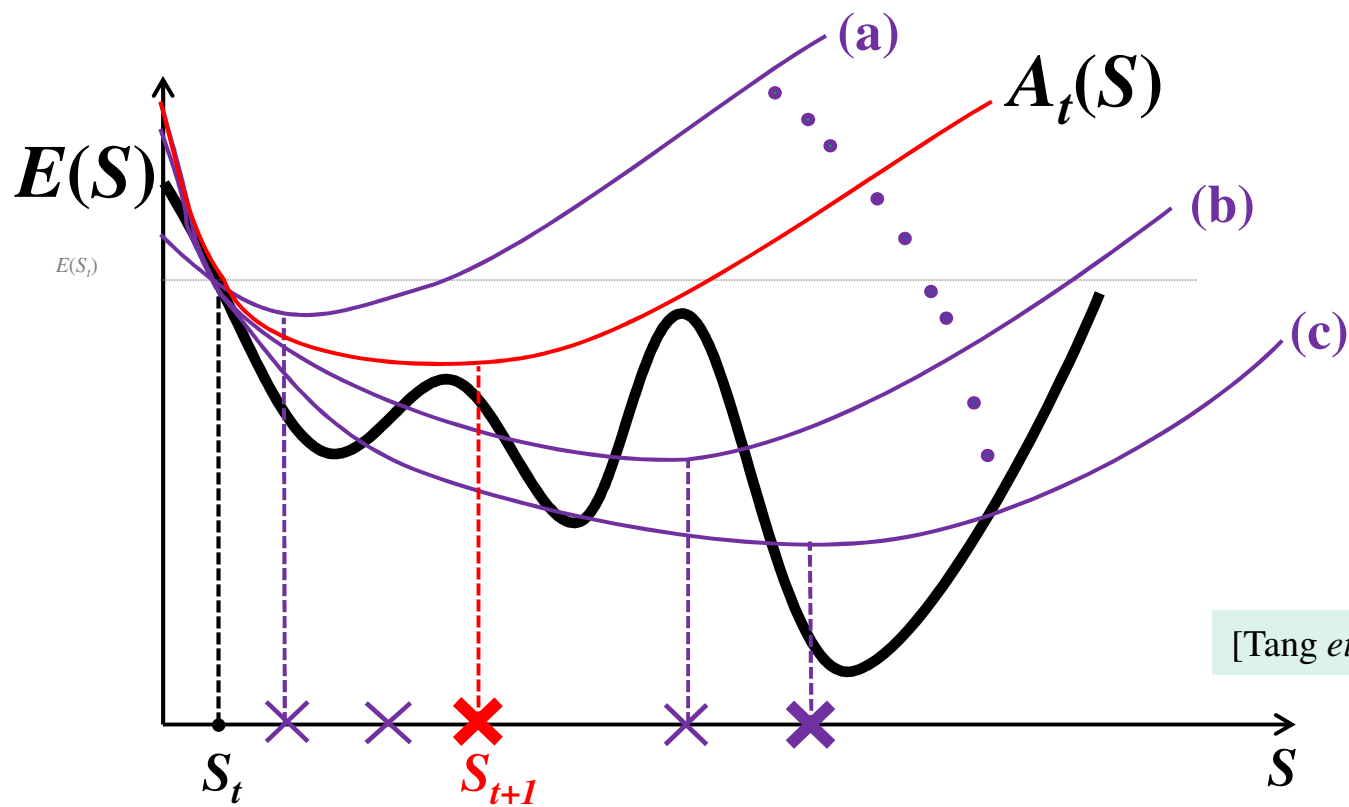


EM/K-means: Fonctions aux.

Pas de garanti d'optimum global!

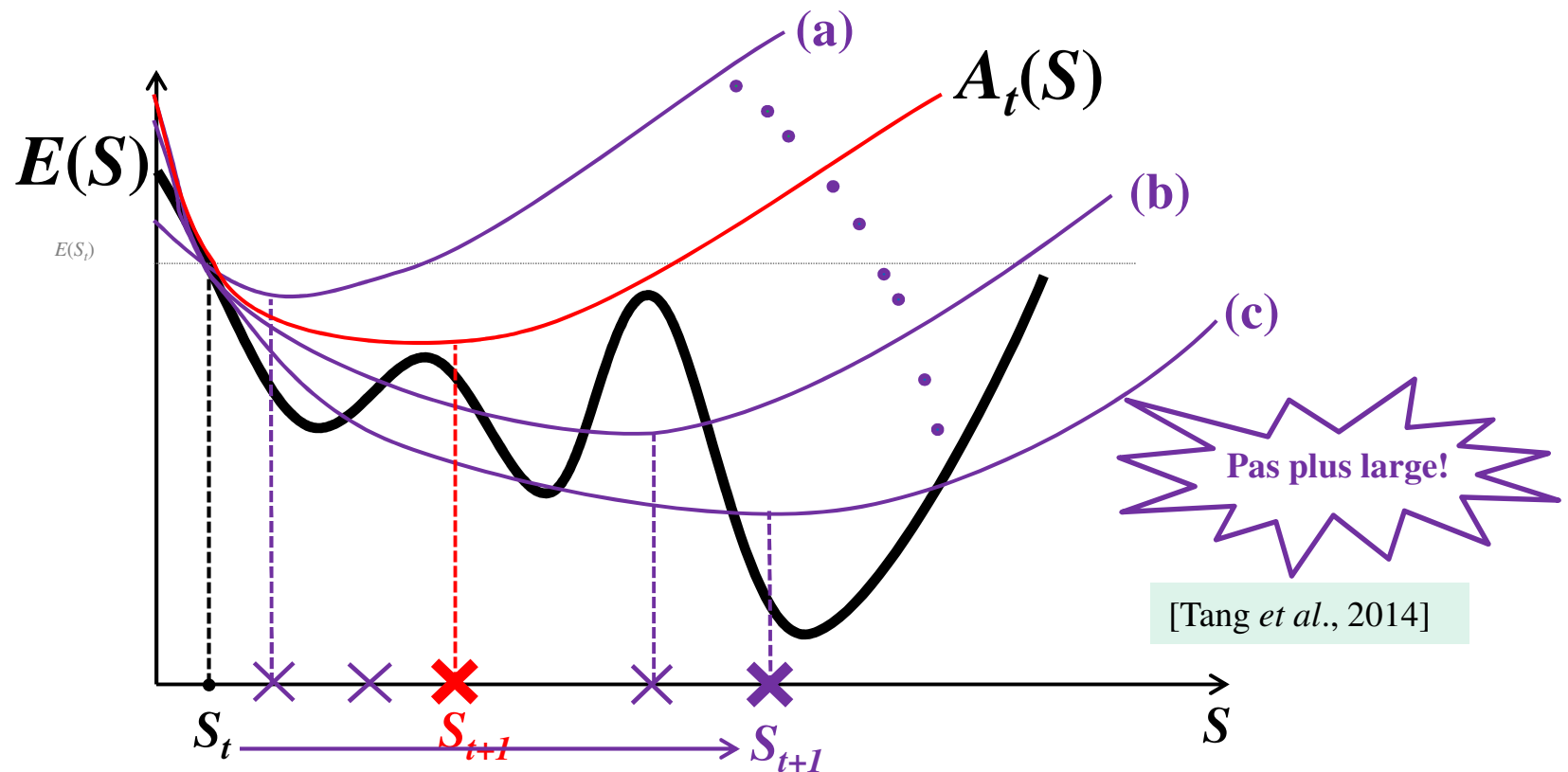


Fonctions *Pseudo-auxiliaires*



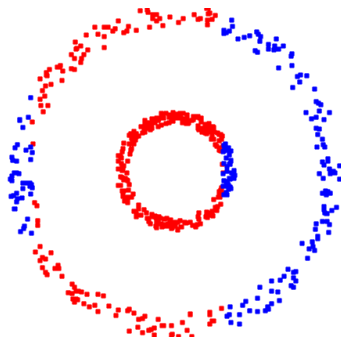
[Tang *et al.*, 2014]

Fonctions *Pseudo-auxiliaires*

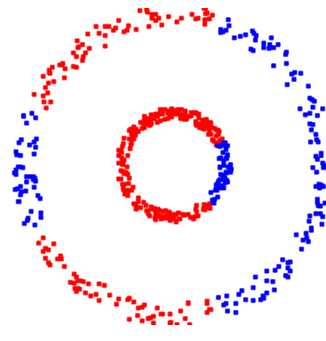


En pratique

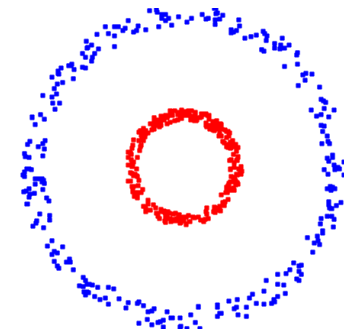
Une mauvaise optimisation peut changer totalement les résultats (ex., kernel K-means)



(a) Initialization
(energy: 0.404)



(b) Bound optimization
(energy: 0.350)



(d) Pseudo-bound
(energy: 0.094)

[Tang *et al.*, 2015]

Généralisation: K-means prob.

Optimisation mixée

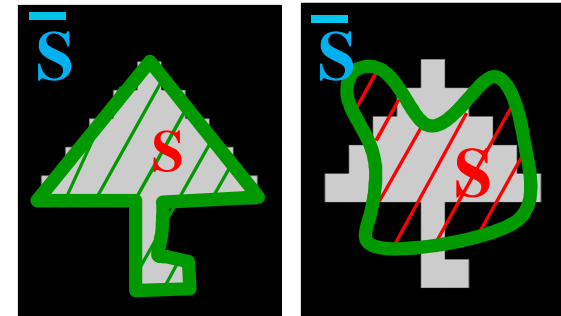
$$F(\mathbf{S}|\theta_1, \theta_0) = \sum_{p \in \mathbf{S}} -\log \mathbf{Pr}(I_p|\theta_1) + \sum_{p \in \bar{\mathbf{S}}} -\log \mathbf{Pr}(I_p|\theta_0)$$

$\min_{\theta_1, \theta_2}$

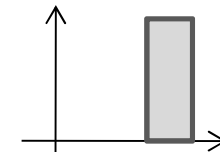


Problème non-linéaire

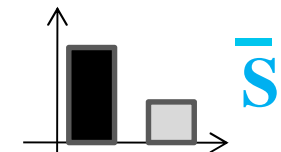
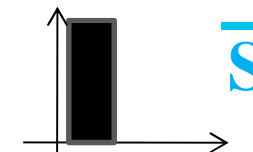
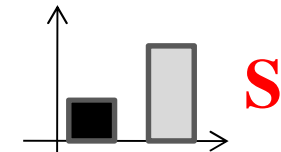
$$E(\mathbf{S}) = \langle \mathbf{1}, \mathbf{S} \rangle H(\mathcal{P}_{\mathbf{S}}) + \langle \mathbf{1}, \bar{\mathbf{S}} \rangle H(\mathcal{P}_{\bar{\mathbf{S}}})$$



low entropy



high entropy



K-means prob.: Optimisation

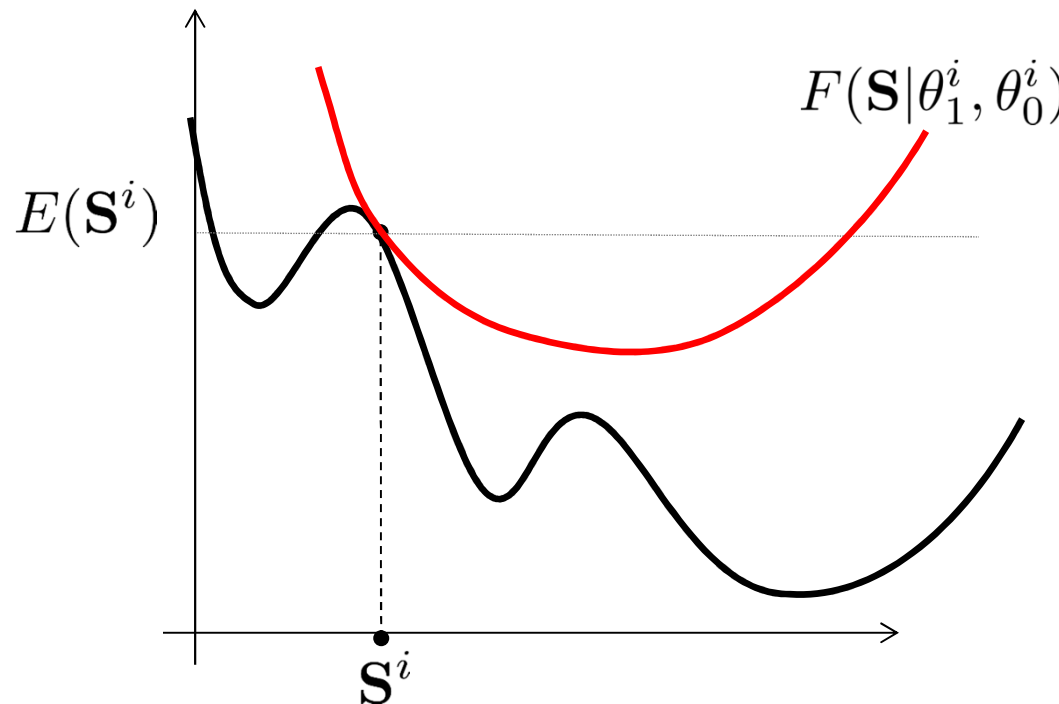
$$F(\mathbf{S}|\theta_1, \theta_0)$$

Optimisation mixée

$$\geq$$

$$E(\mathbf{S})$$

Problème non-linéaire



K-means prob.: Optimisation

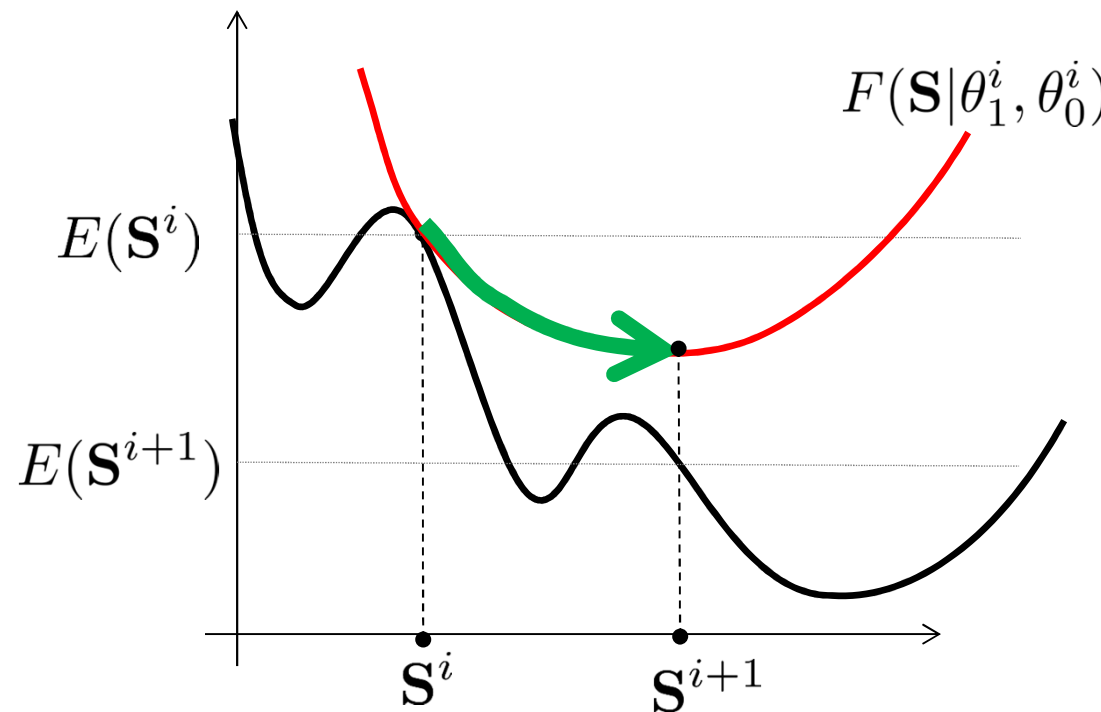
$$F(\mathbf{S}|\theta_1, \theta_0)$$

Optimisation mixée

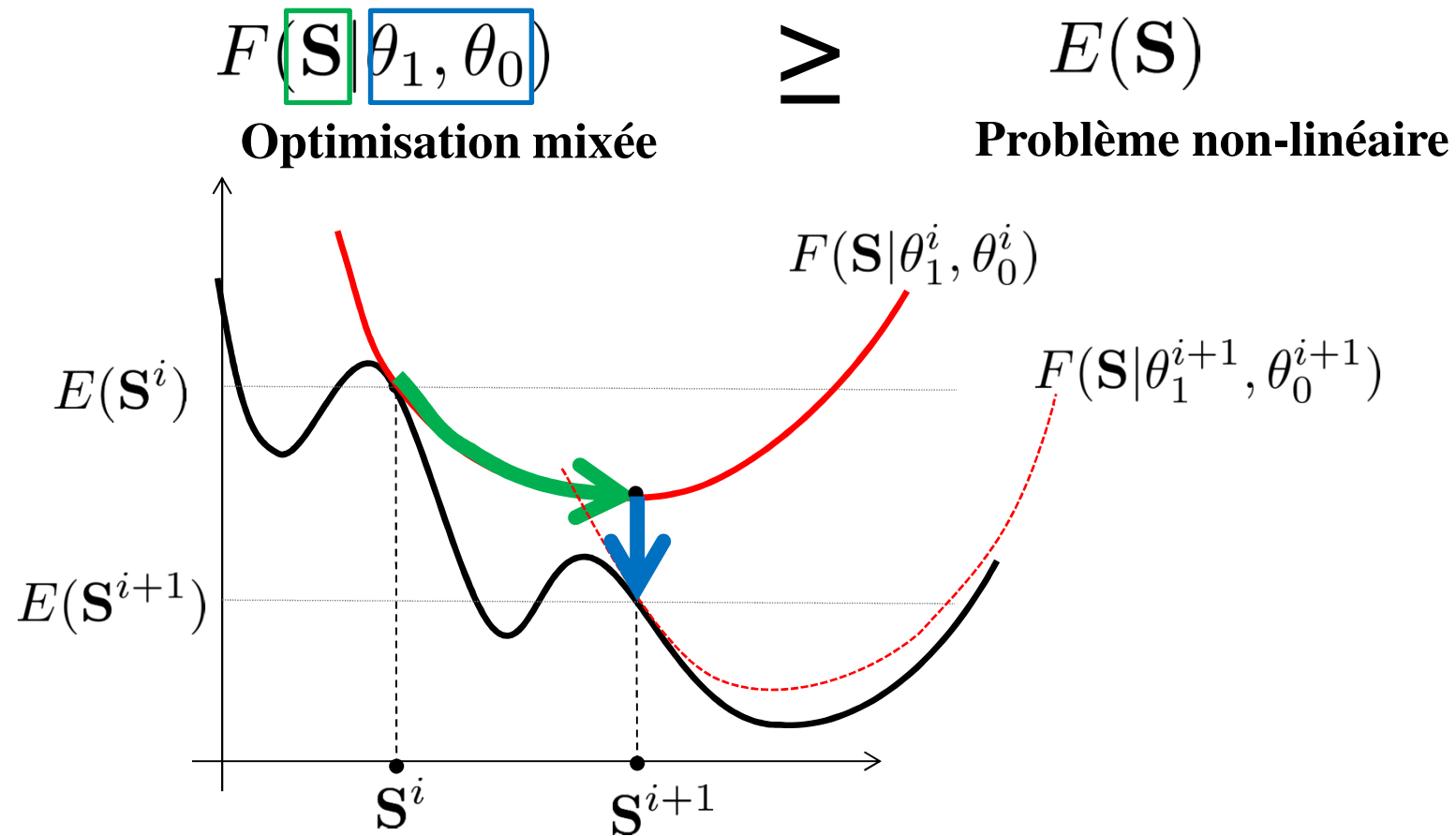
$$\geq$$

$$E(\mathbf{S})$$

Problème non-linéaire

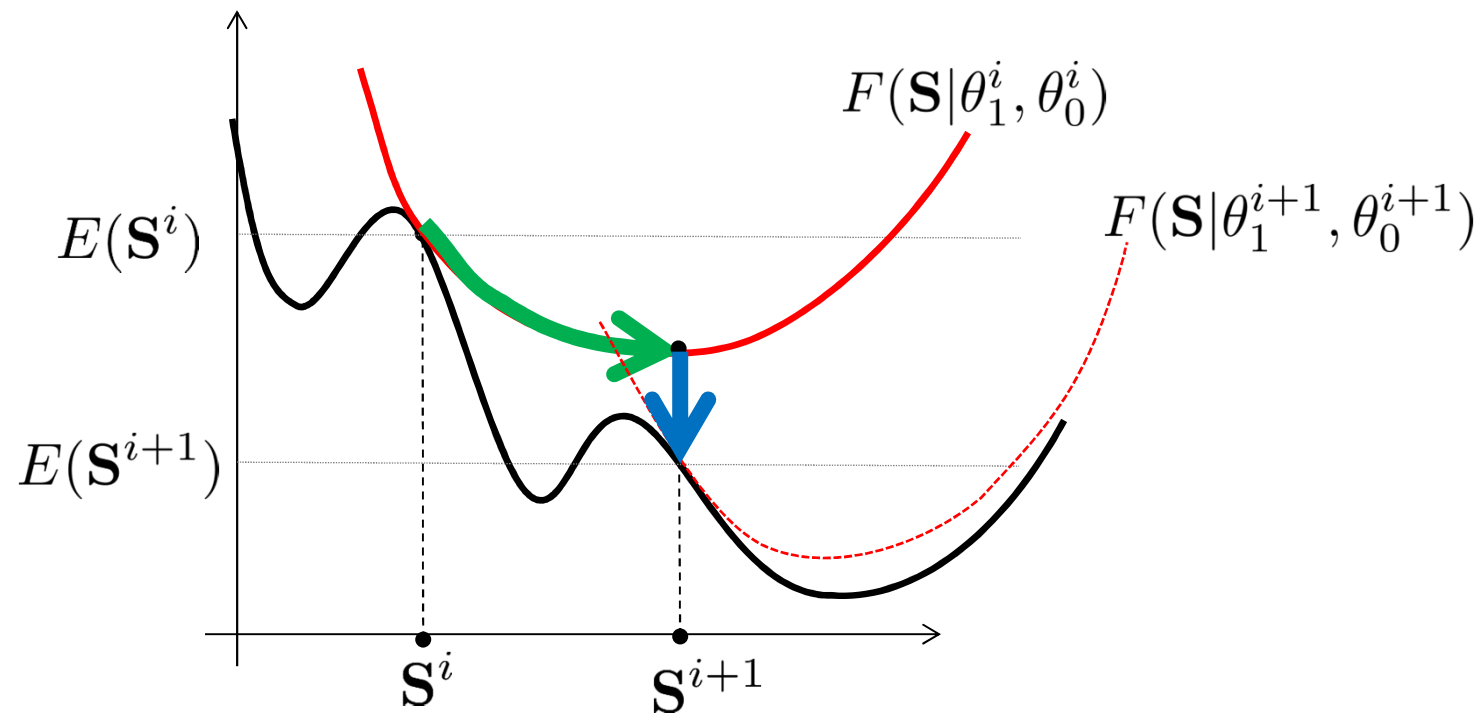


K-means prob.: Optimisation

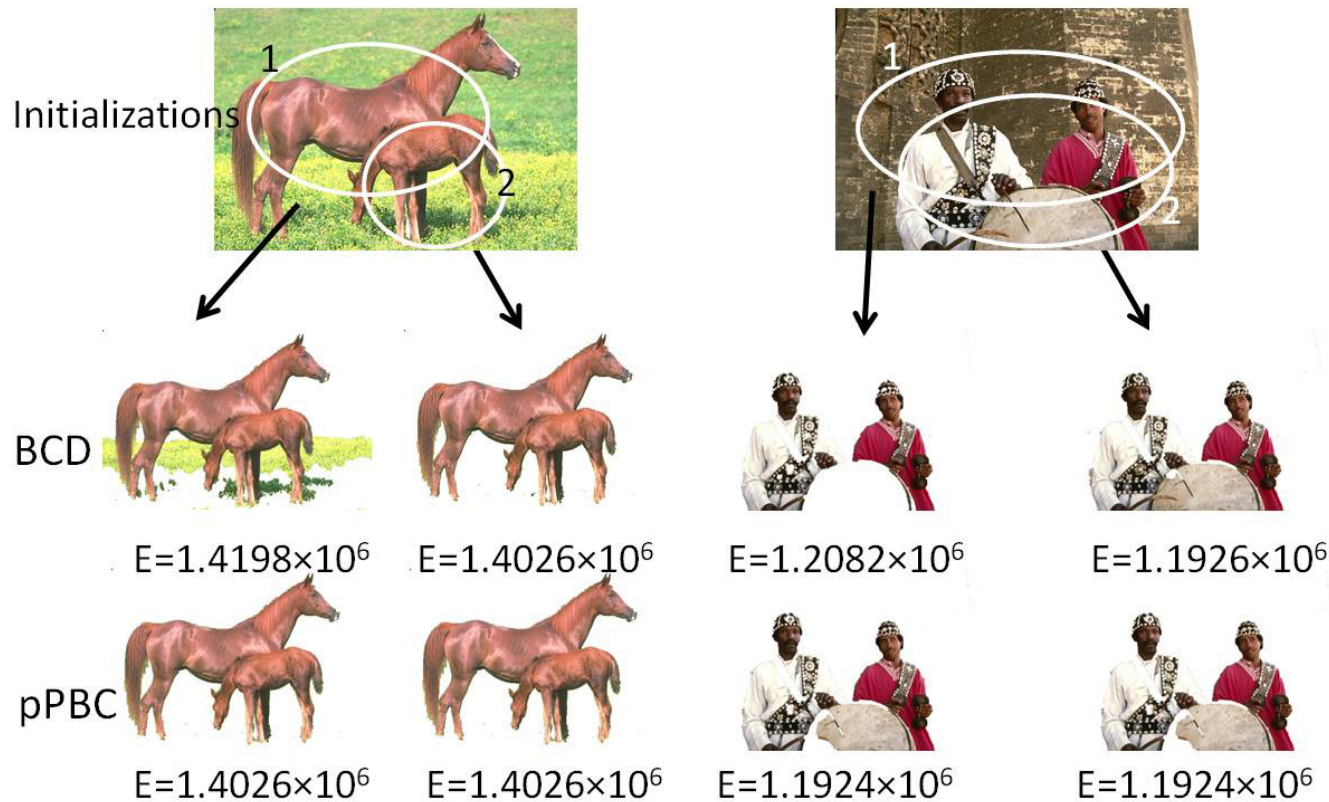


K-means prob.: Optimisation

$$F(\mathbf{S}|\theta_1, \theta_0) + \lambda \langle 1, \mathbf{S} \rangle \leftarrow \text{Perturbation}$$



K-means prob.: Optimisation



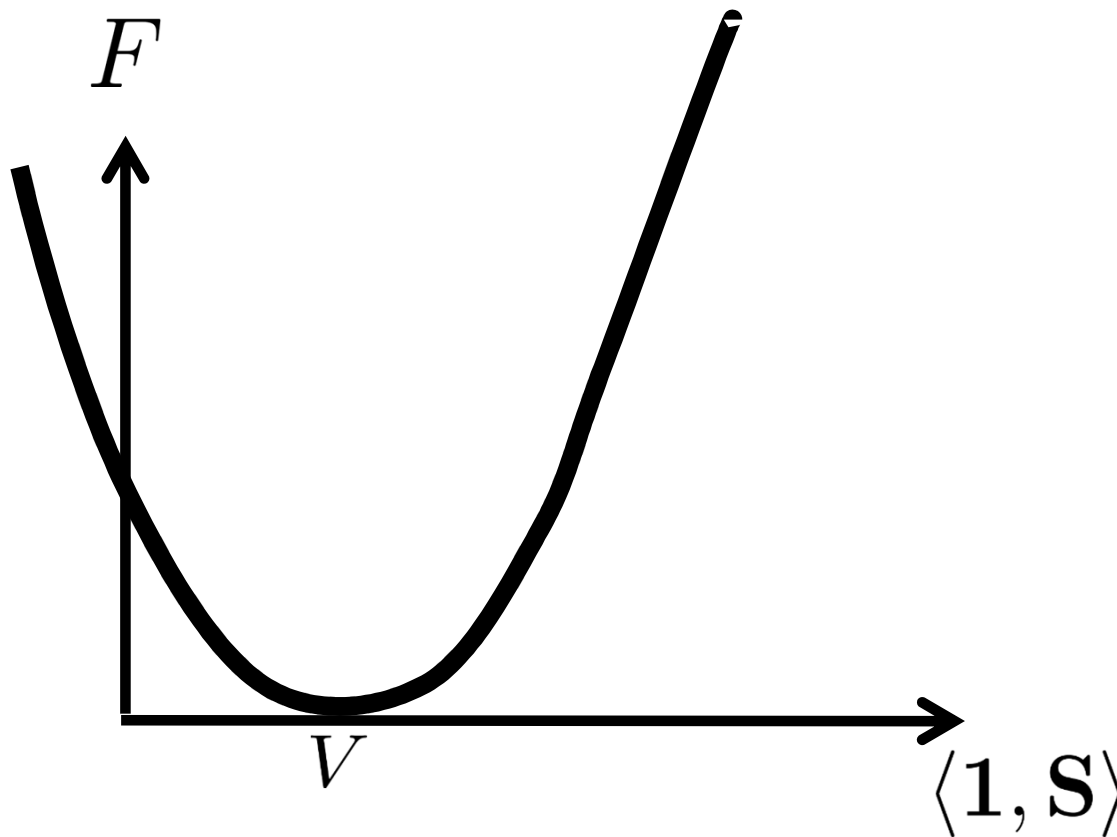
Sommaire

Optimisation

- 1) Introduction au algorithmes génétiques
- 2) Optimisation conjointe (un exemple)
 - 1) Optimisation avec multiples critères (exemples)

Exemple

$$F(\langle \mathbf{1}, \mathbf{S} \rangle) = (\langle \mathbf{1}, \mathbf{S} \rangle - V)^2$$



K-means prob.



**Ajouter une
contrainte de
volume**

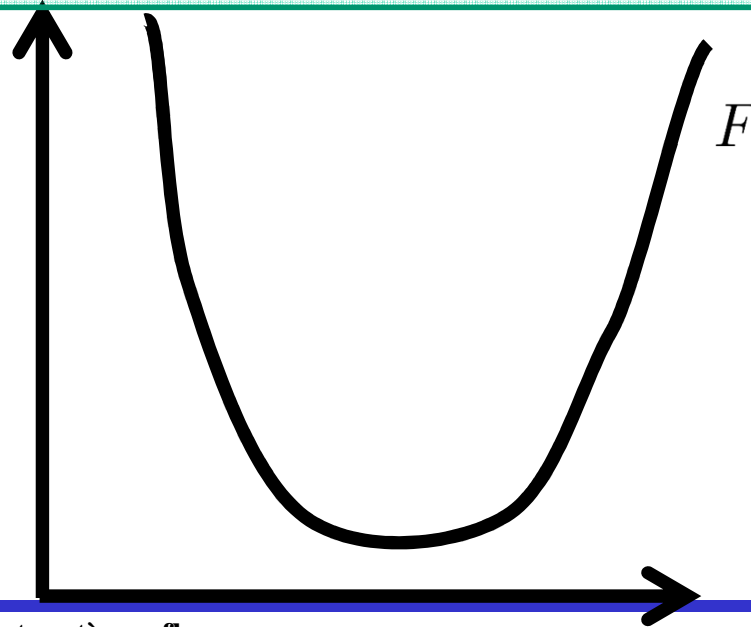
Fonction auxiliaire?

Pas toujours facile d'en trouver une

- 'Cauchy-Schwarz inequality'
- 'Quadratic bound principle'
- 'First-order expansion'
- 'Jensen's inequality'
- 'Entropy inequality'
- 'Linear bounds on fractional terms'

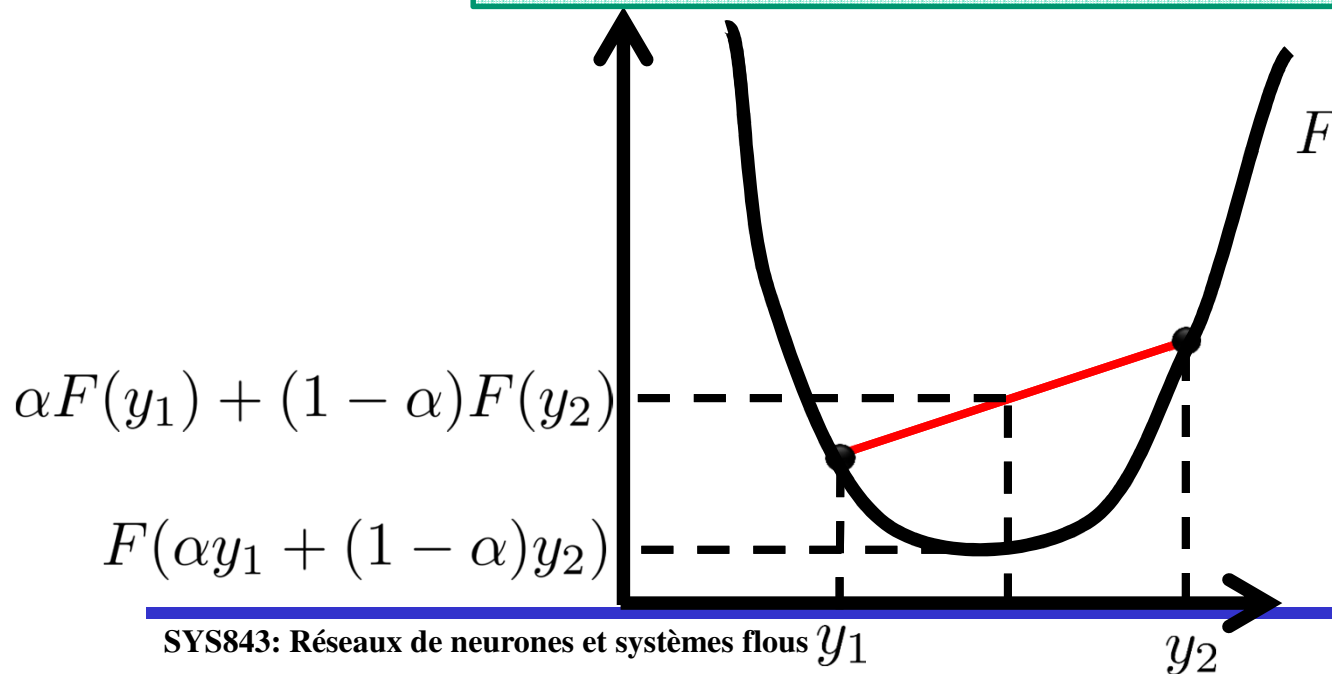
Inégalité de Jensen?

$$F\left(\sum_p \alpha_p y_p\right) \leq \sum_p \alpha_p F(y_p)$$
$$\sum_p \alpha_p = 1; \quad \alpha_p \geq 0$$



Inégalité de Jensen?

$$F\left(\sum_p \alpha_p y_p\right) \leq \sum_p \alpha_p F(y_p)$$
$$\sum_p \alpha_p = 1; \quad \alpha_p \geq 0$$



Fonction auxiliaire?

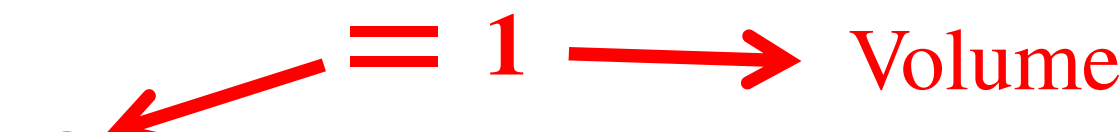
[Ben Ayed et al., CVPR 13]

$$\langle \mathbf{g}, \mathbf{S} \rangle = \sum_{p \in \mathbf{S}} g_p$$

Fonction auxiliaire?

[Ben Ayed et al., CVPR 13]

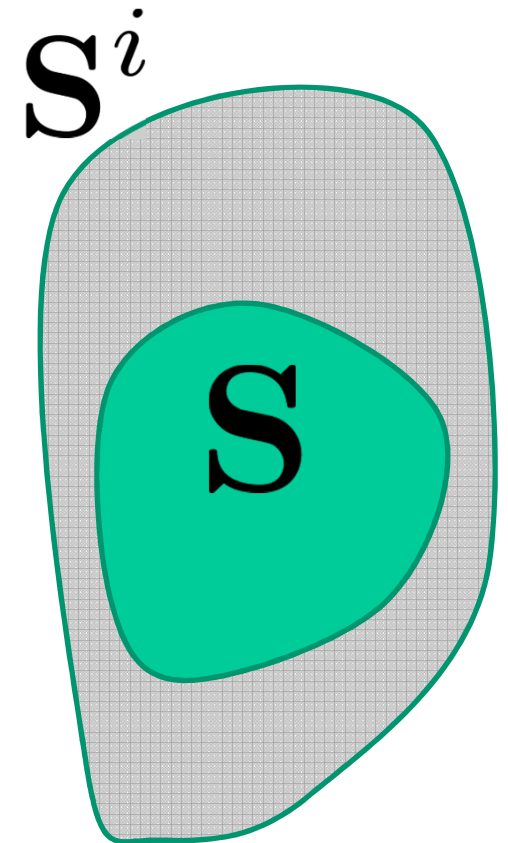
$$\langle \mathbf{g}, \mathbf{S} \rangle = \sum_{p \in \mathbf{S}} g_p$$

 $= 1 \longrightarrow \text{Volume}$

Fonction auxiliaire?

[Ben Ayed et al., CVPR 13]

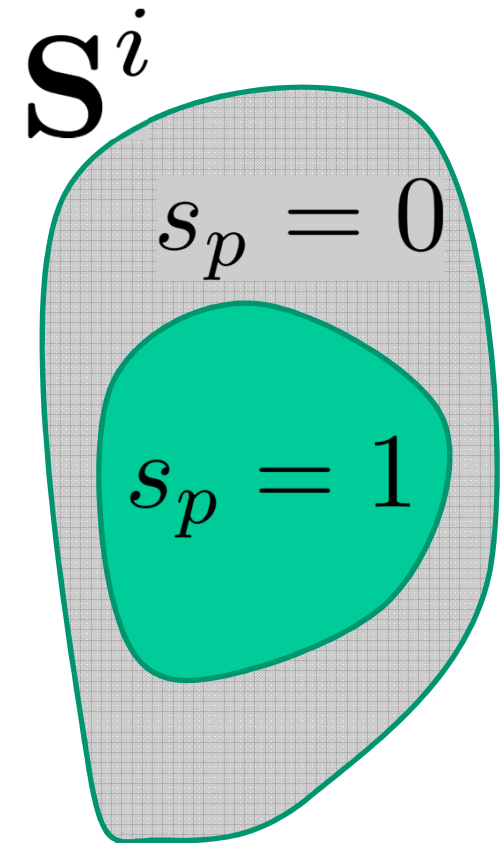
$$\langle \mathbf{g}, \mathbf{S} \rangle = \sum_{p \in \mathbf{S}} g_p$$



Fonction auxiliaire?

[Ben Ayed et al., CVPR 13]

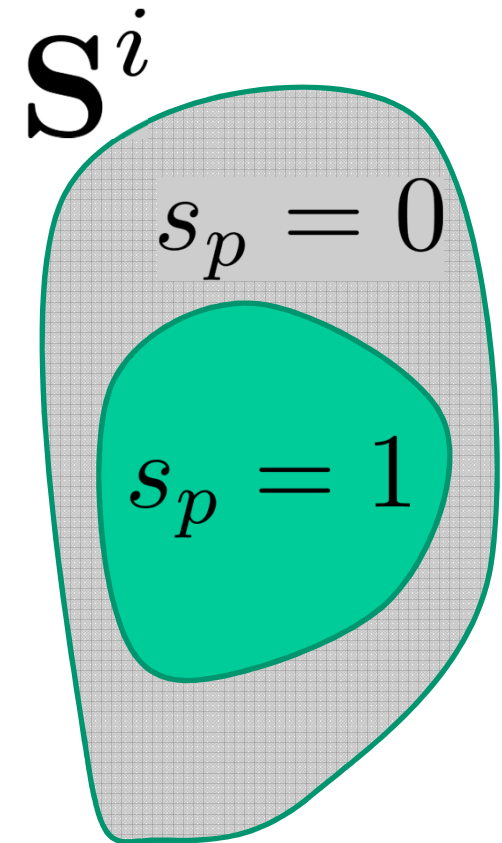
$$\begin{aligned}\langle \mathbf{g}, \mathbf{S} \rangle &= \sum_{p \in \mathbf{S}} g_p \\ &= \sum_{p \in \mathbf{S}^i} g_p s_p\end{aligned}$$



Fonction auxiliaire?

[Ben Ayed et al., CVPR 13]

$$F(\langle \mathbf{g}, \mathbf{S} \rangle) = F\left(\sum_{p \in \mathbf{S}^i} g_p s_p\right)$$



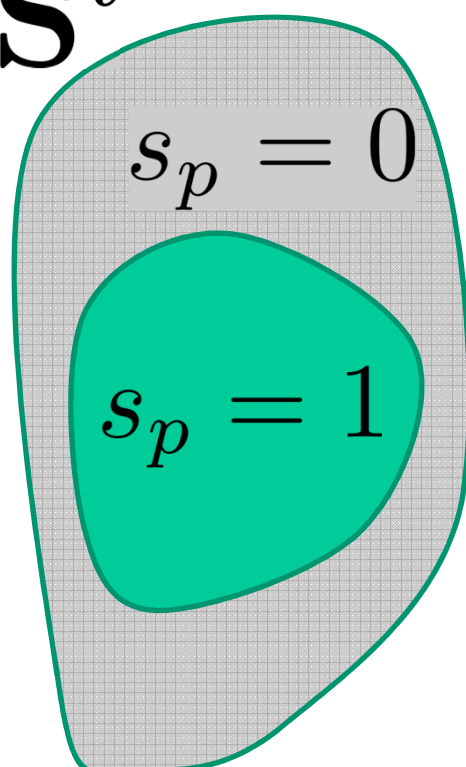
Fonction auxiliaire?

[Ben Ayed et al., CVPR 13]

$$F(\langle \mathbf{g}, \mathbf{S} \rangle) = F\left(\sum_{p \in \mathbf{S}^i} g_p s_p\right)$$

$$= F\left(\sum_{p \in \mathbf{S}^i} \frac{g_p}{\langle \mathbf{g}, \mathbf{S}^i \rangle} \langle \mathbf{g}, \mathbf{S}^i \rangle s_p\right)$$

Constante



The diagram illustrates the set \mathbf{S}^i as a collection of points. A green shaded region represents the subset where $s_p = 1$, and the surrounding grey shaded region represents where $s_p = 0$. The entire set is labeled \mathbf{S}^i .

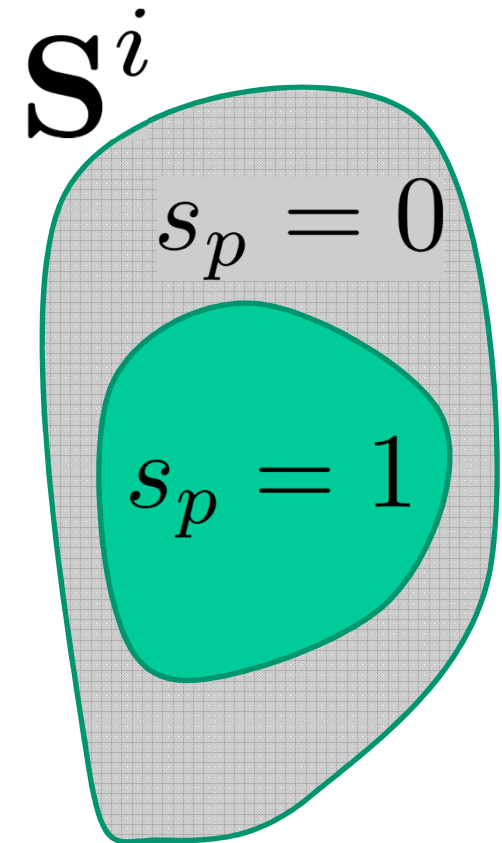
Fonction auxiliaire?

[Ben Ayed et al., CVPR 13]

$$F(\langle \mathbf{g}, \mathbf{S} \rangle) = F\left(\sum_{p \in \mathbf{S}^i} g_p s_p\right)$$

$$= F\left(\sum_{p \in \mathbf{S}^i} \underbrace{\frac{g_p}{\langle \mathbf{g}, \mathbf{S}^i \rangle}}_{\alpha_p} \underbrace{\langle \mathbf{g}, \mathbf{S}^i \rangle s_p}_{y_p}\right)$$

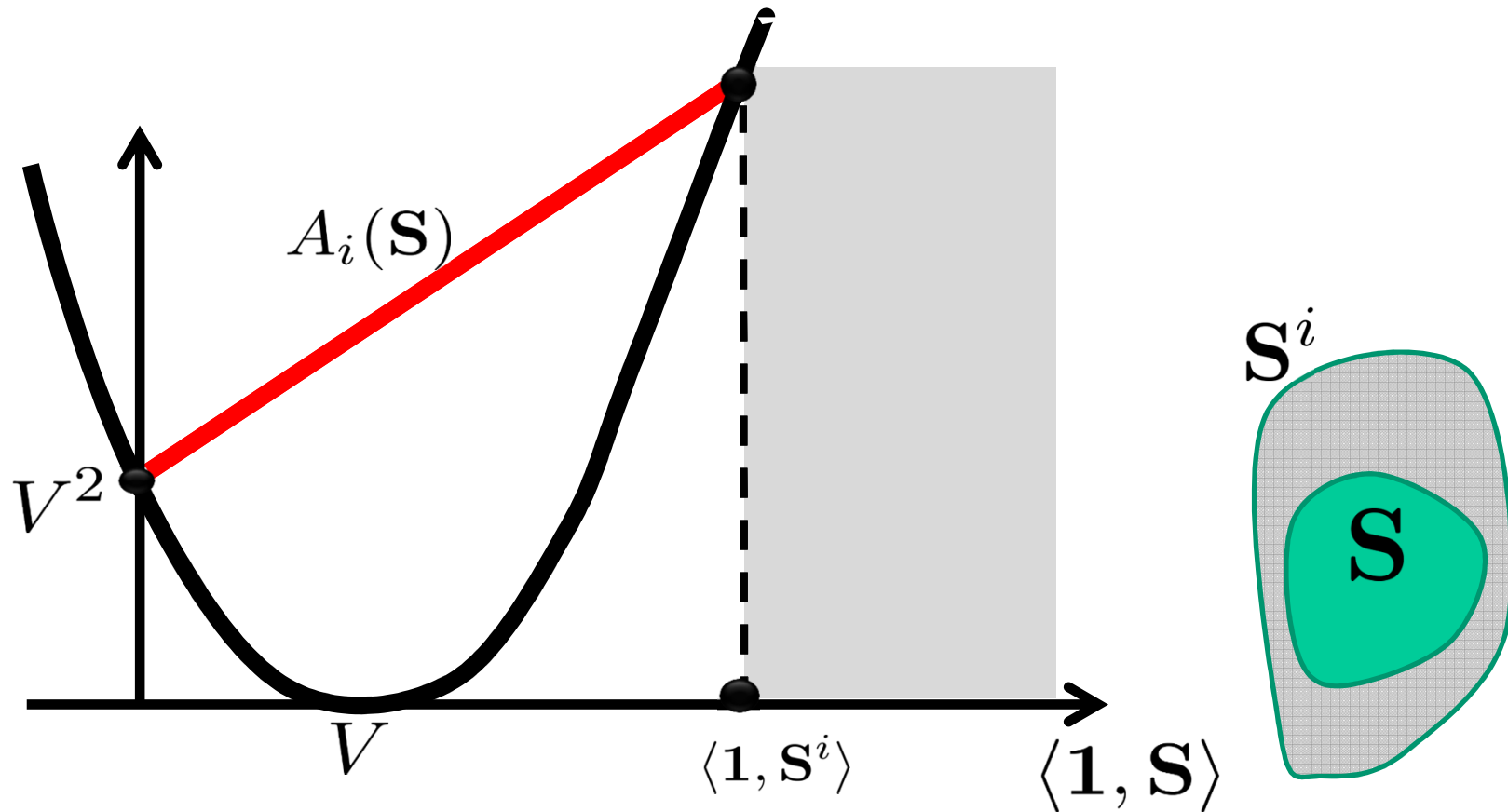
Somme= 1 ←



Fonction auxiliaire?

[Ben Ayed et al., CVPR 13]

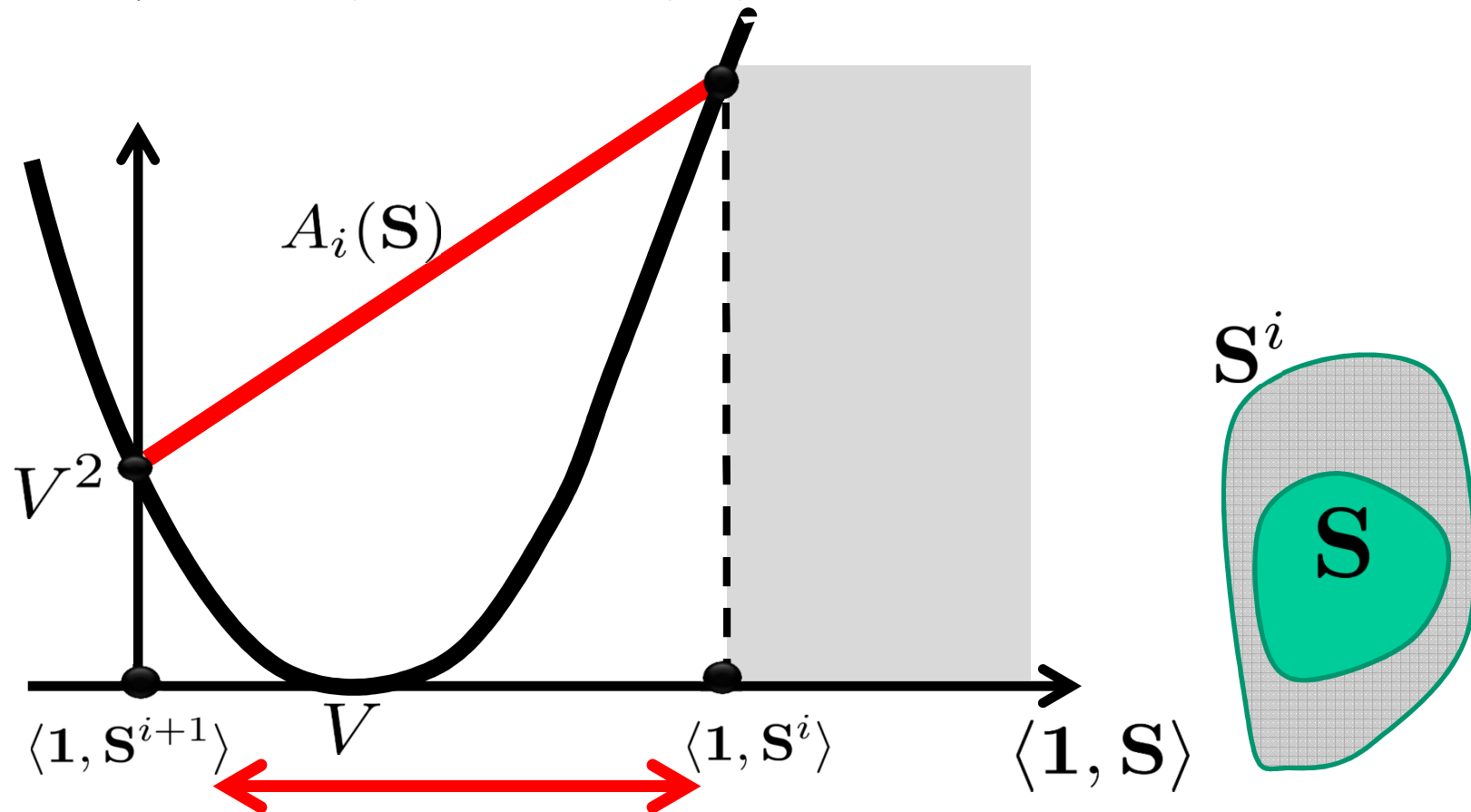
$$(\langle \mathbf{1}, \mathbf{S} \rangle - V)^2 < A_i(\mathbf{S})$$



Fonction auxiliaire?

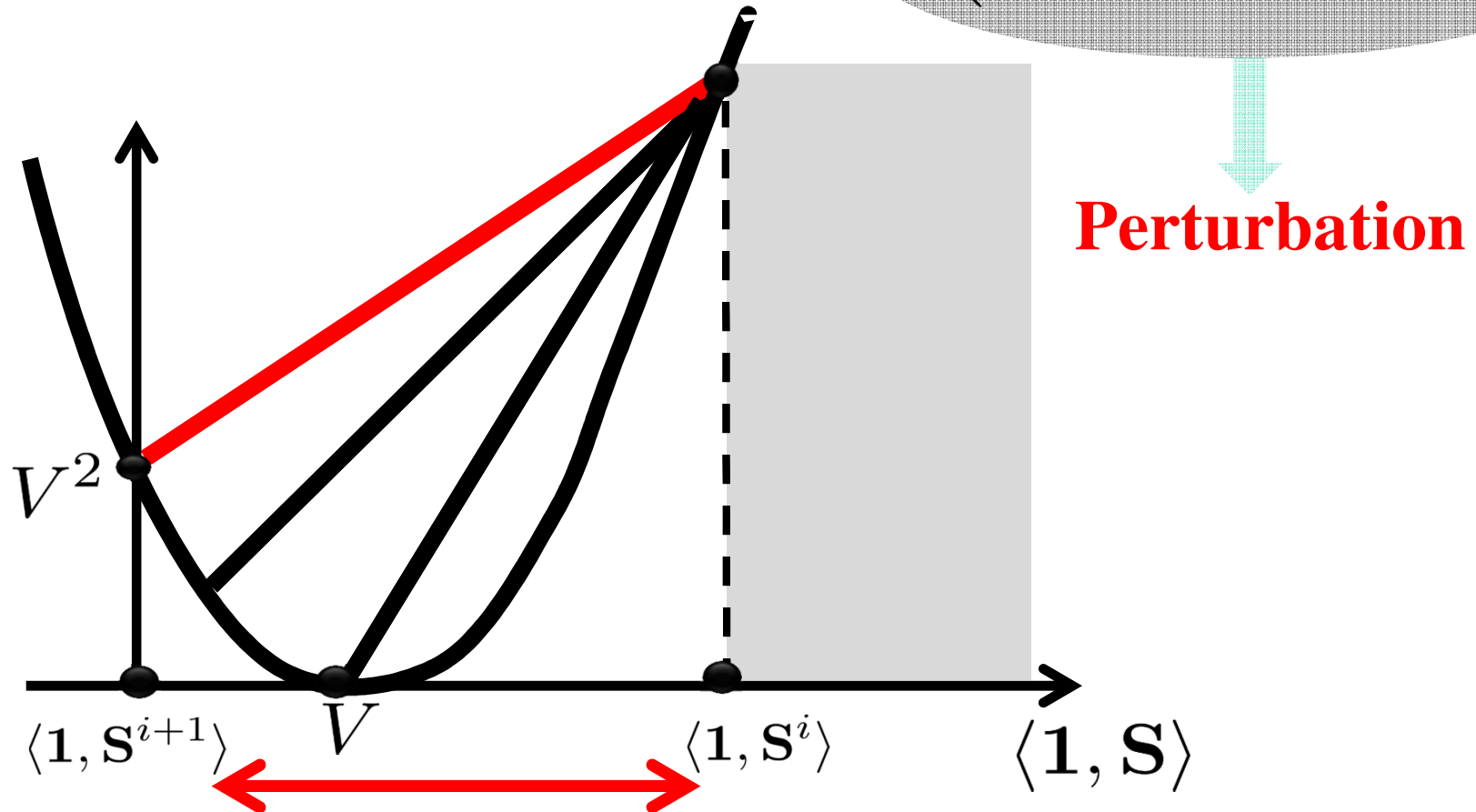
[Ben Ayed et al., CVPR 13]

$$(\langle \mathbf{1}, \mathbf{S} \rangle - V)^2 < A_i(\mathbf{S})$$



Fonction pseudo-auxiliaire?

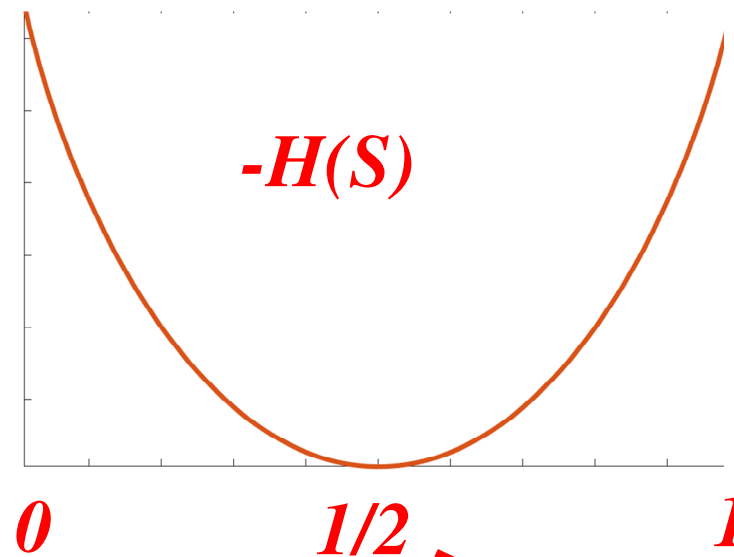
$$(\langle \mathbf{1}, \mathbf{S} \rangle - V)^2 < A_i(\mathbf{S}) + \lambda (\langle \mathbf{1}, \mathbf{S} \rangle - \langle \mathbf{1}, \mathbf{S}^i \rangle)$$



Exemple: L'entropie de partition

K-means: Préfère des solutions équilibrées en terme de cardinalité des clusters

[Kearns *et al.*, UAI 97]

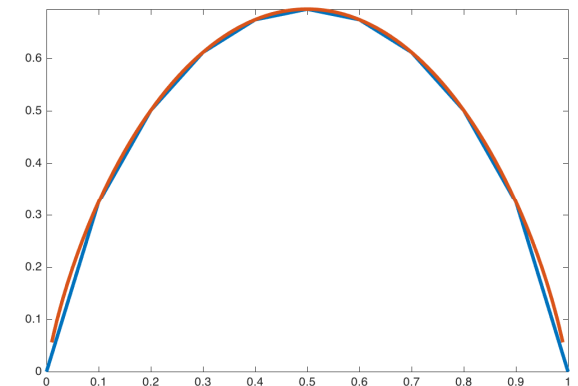
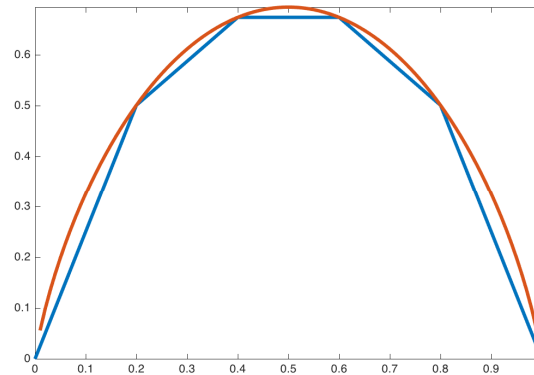
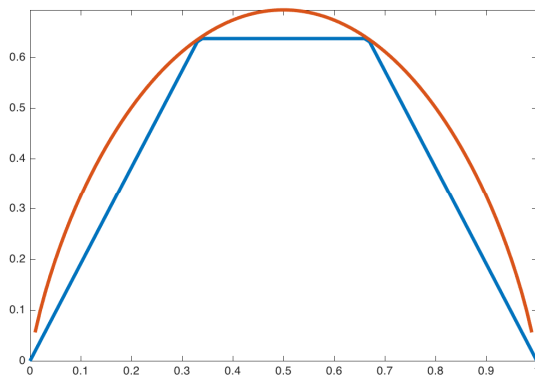


***Les deux clusters ont la même
cardinalité (proportion)***

Entropie de partition: optimisation

- Solution globale peut être obtenue en utilisant:

$$\min_{\mathbf{S}} [\min_i a_i \langle 1, \mathbf{S} \rangle + b_i)]$$

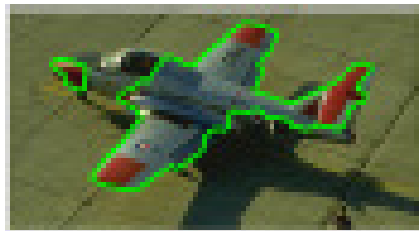


[Boykov *et al.*, 2015]

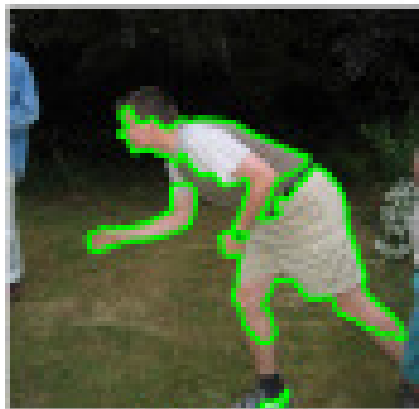
Entropie de partition: optimisation

[Boykov *et al.*, 2015]

Optimi. globale



error: 6.85%

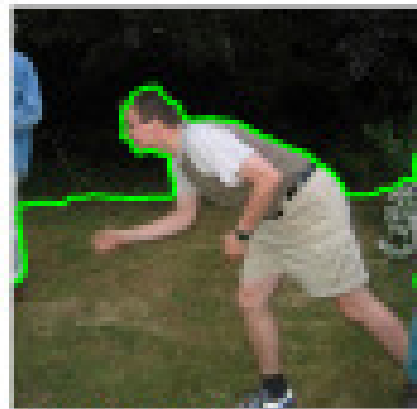


error: 4.95%

Fonction aux.



error: 9.64%



error: 41.20%

Références

- Wendy Williams, Genetic Algorithms: A tutorial
- Assaf Zilitsky, Introduction to Genetic Algorithms
- M. Tang et al., Pseudo-bound optimization for binary energies, ECCV 14
- C. Bishop, Pattern recognition and machine learning
- Ben Ayed et al., Auxiliary Cuts for General Classes of High-Order Functionals, CVPR 13
- Boykov et al., Volumetric bias in segmentation and reconstructions: secrets and solutions, ICCV 15
- M. Kearns et al., An Information-Theoretic Analysis of Hard and Soft Assignment Methods for Clustering, UAI 97.