

Algorithmes évolutionnistes

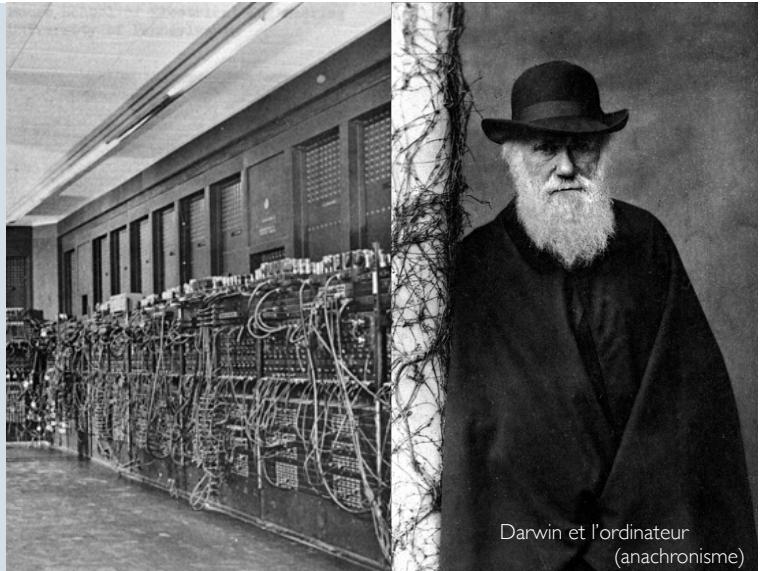
Optimisation stochastique à base de population

UE méthodes et outils pour l'IA et la RO
L3 informatique

Nicolas Bredeche

Sorbonne Université
ISIR, UMR 7222
Paris, France
nicolas.bredeche@upmc.fr

...



Darwin et l'ordinateur
(anachronisme)



rev. 2019-04-04

Préambule

2

Documents autorisés pour l'examen:
une seule et unique feuille A4 recto-verso manuscrite

Problème: $y^* = \arg\min_{y \in Y} f(y)$

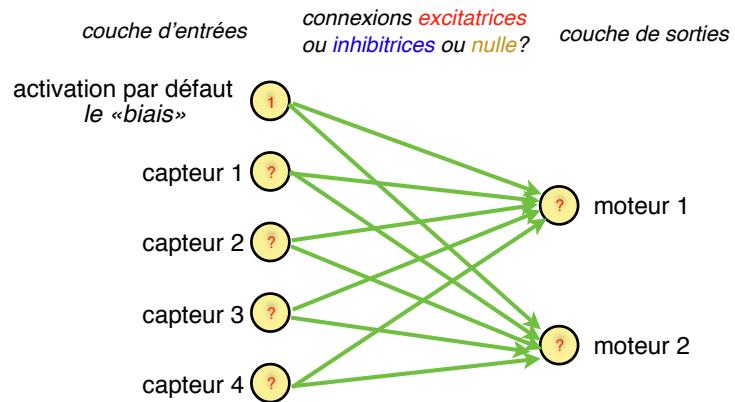
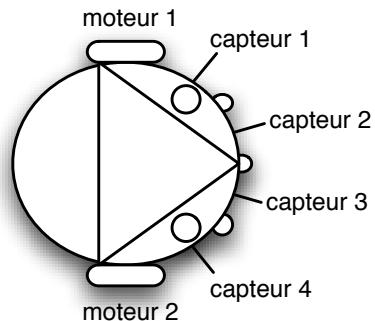
Solution candidate: $a := (y, f(y))$

Formulation simplifiée

Formulation plus générale: $a := (y, s, f(y), f'(y), f''(y))$

Optimisation

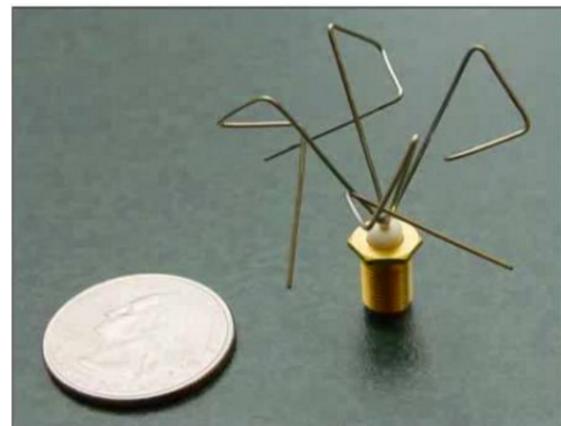
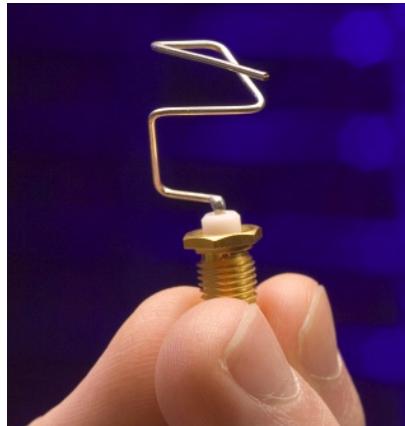
- Des méthodes pour des classes de problèmes
 - ▶ Algorithme de gradient (recherche locale, suit le gradient)
 - ▶ Hill-climbing (recherche locale, change un élément à la fois)
 - ▶ Méthodes énumératives (recherche globale, espace de recherche discret)
 - ▶ Méthodes heuristiques (espace structuré)
 - ▶ Méta-heuristique et méthodes stochastiques
 - recherche aléatoire (recherche globale, sans a priori) [Monte carlo, Tabu]
 - recuit simulé (recherche globale) ["simulated annealing"]
 - méthodes bio-inspirées (recherche globale) [DE, PSO, Algo. évol., ...]



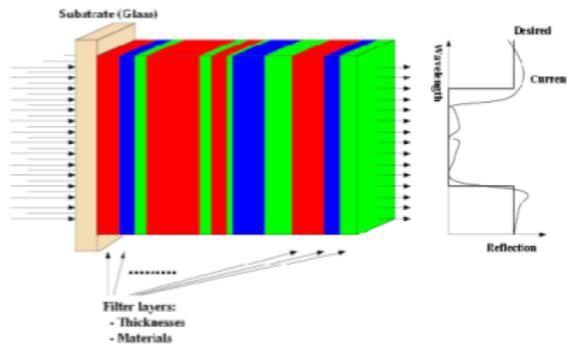
- Fonction objectif:
 - ▶ $f(y) = \text{distance parcourue pendant } n \text{ itération}$
- Description d'une solution:
 - ▶ $\{-1,0,1\}^{10}$, soit: $[x_1, x_2, \dots, x_{10}]$, avec $x_i \in \{-1,0,+1\}$
- Algorithme:
 - ▶ recherche aléatoire

exemple pratique
robot_randomsearch.py

Antenne



- Antenne de petite taille pour satellite
 - Objectif: maximisez l'efficacité, minimiser la taille
 - Espace de recherche: structure de l'antenne (graphe)
 - Difficulté: absence de modèles d'interférences



- Définir les couches d'un filtre optique
 - Objectif: contraintes fixées par un cahier des charges
 - Espace de recherche: (matériau, épaisseur)ⁿ
 - Difficulté: espace de recherche discret et continu

Image extraite de cours @ M. Schöenauer (avec permission)

[Schutz, Bäck][Martin et al., 1995]

Recherche avec un critère subjectif, problème inverse

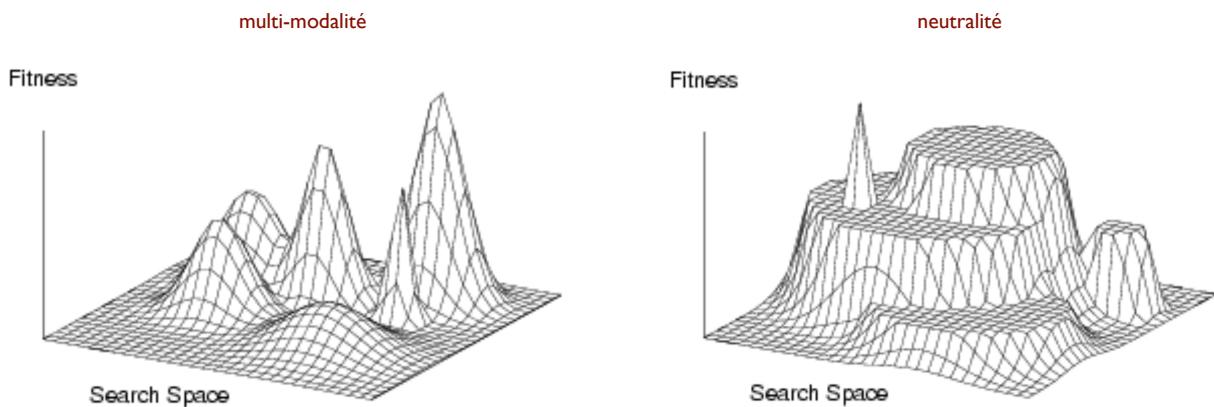


- Retrouver la composition d'un café à l'arôme
 - Objectif: maximiser la satisfaction de l'expert-évaluateur
 - Espace de recherche: mélanges de café
 - Difficulté: évaluation empirique par l'expert

- Propriétés
 - Espace de recherche
 - ▶ binaire, symbolique, continu
 - ▶ structuré ou non
 - Fonction de performance
 - ▶ lien tenu entre représentation et performance
 - ▶ évaluation potentiellement bruitée
 - Relation faible entre espace de recherche et objectif

Caractérisation de l'espace de recherche

10



- espace de recherche “complexe”
 - ▶ multi-modalité, plateaux de neutralité
 - ▶ irrégulière, non différentiable, discontinue
- espace de recherche inconnu
 - ▶ problème type “boîte noire”
 - ▶ MAIS: notion de voisinage (sinon, il ne reste plus que Monte Carlo)

Algorithmes évolutionnistes*

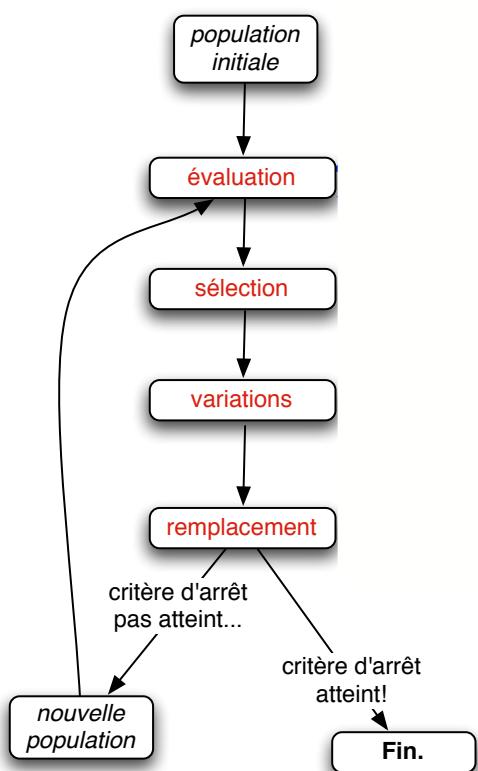
Principes généraux

*Ou: Algorithmes évolutionnaires

*Ou: Algorithmes pour l'optimisation stochastique à base de population

Optimisation par évolution artificielle

12



- Caractérisation
 - Algorithme d'optimisation stochastique à base de population
 - Famille des méta-heuristiques
- Deux mécanismes principaux
 - Pression à la sélection
 - Variations +/- aveugles
- Propriété souhaitée
 - Recherche globale et locale

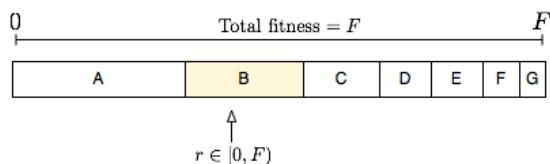


- Définition
 - ▶ Sélectionne une sous-partie des solutions candidates
- Exemple
 - ▶ Renvoie les N meilleurs individus parmi M
- Propriétés
 - ▶ Déterministe vs. stochastique
 - ▶ Compromis exploration/exploitation
 - ▶ Elitiste ou non

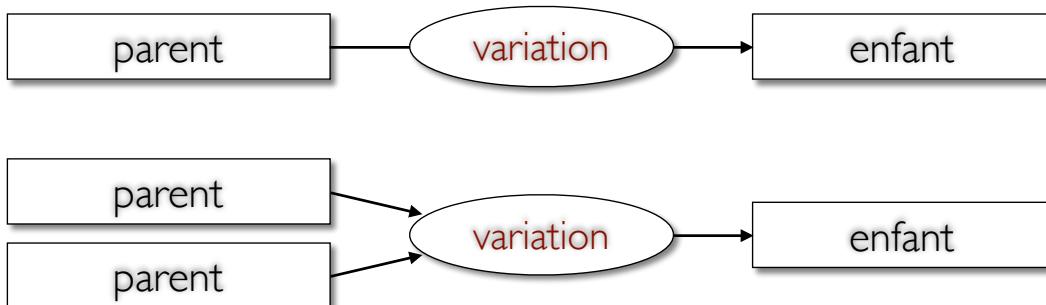


- Méthodes
 - ▶ fitness proportionate
 - ▶ *k*-tournament
 - ▶ plus-selection ($\mu+\lambda$)
 - ▶ comma-selection (μ,λ)
 - ▶ NSGA-2 (multi-objectif)

- K-tournament
 - ▶ sélectionner k individus
 - ▶ garder le meilleur
- Fitness-proportionate



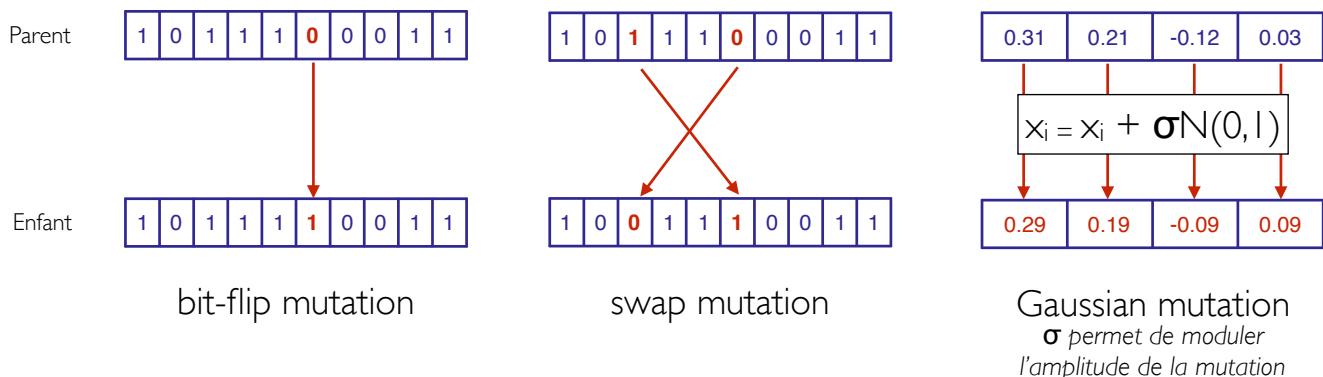
- (μ, λ) -ES
 - ▶ sélectionner les μ meilleurs, générer λ enfants, garder λ
- $(\mu + \lambda)$ -ES ("élitiste")
 - ▶ sélectionner les μ meilleurs, générer λ enfants, garder μ et λ



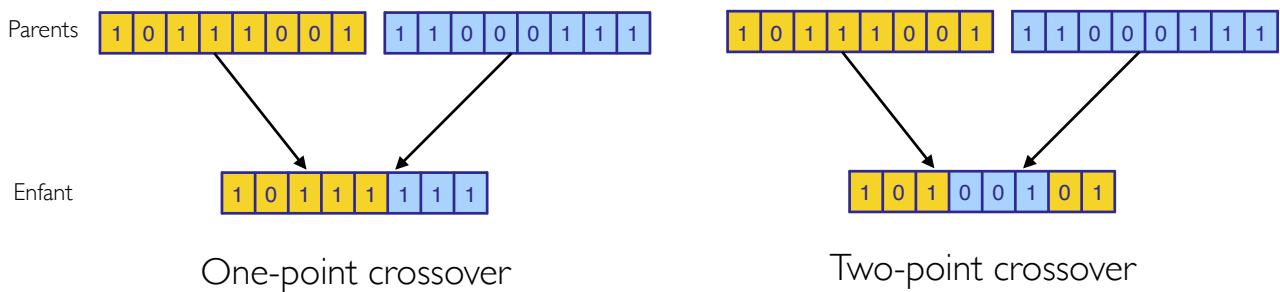
- Définition
 - ▶ Construit un nouvel individu à partir d'un (ou plusieurs) individus
- Exemple
 - ▶ Modifie aléatoirement un élément du génome
- Propriétés
 - ▶ Conservatif vs. disruptif



- Définition
 - ▶ Construit un nouvel individu à partir d'un seul individu parent
- Exemples



- Définition
 - ▶ Construit un nouvel individu à partir de 2 (ou +) individus parents
- Exemples:

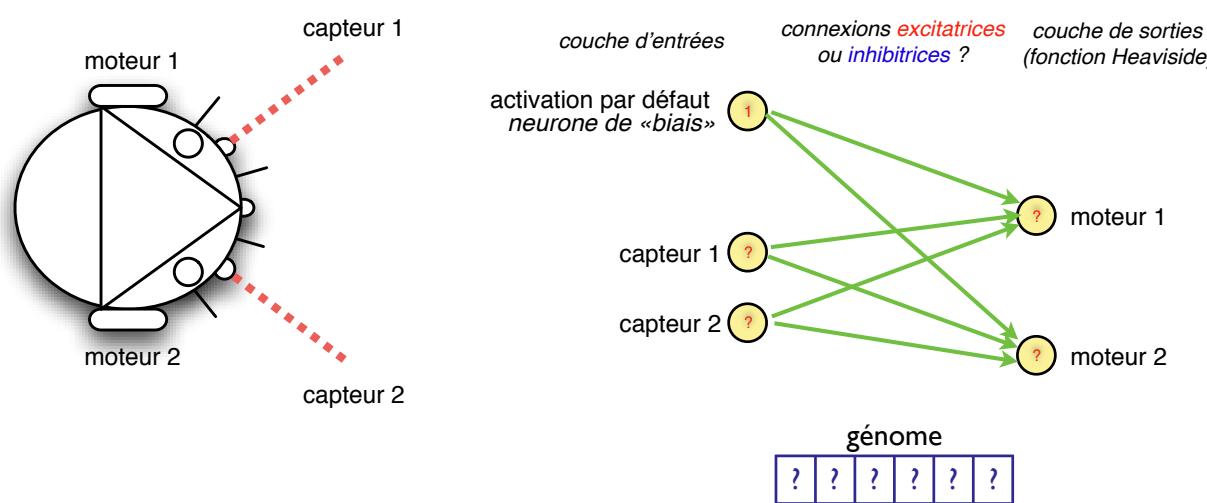


Cas d'étude

Robotique évolutionniste

Optimiser un robot éviteur d'obstacle

20



$$fitness(x) = \int_T V * (1 - \sqrt{\delta_v}) * (1 - sensor_{max})$$

- maximiser la vitesse de chaque moteur
- maximiser la vitesse de translation
- maximiser la distance au mur

$$\text{fitness}(x) = \int_T V * (1 - \sqrt{\delta_v}) * (1 - \text{sensor}_{max})$$

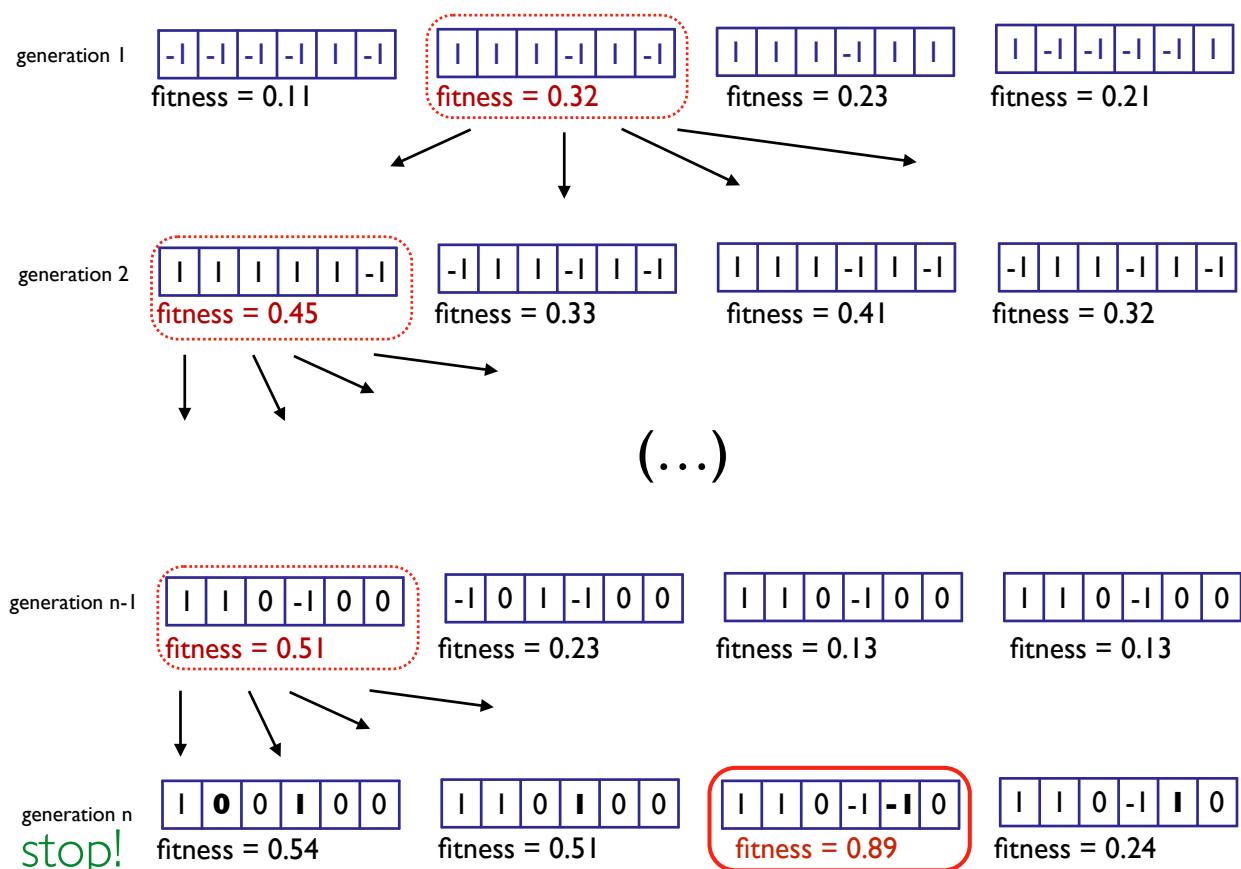
- **Objectif** : maximiser la fonction fitness
- **Représentation d'un individu** : $\{-1, 0, 1\}^m$
- **Population** : 4 individus initialement tirés au hasard
- **Opérateur de Sélection** : (μ, λ) -ES avec $\mu=1$ et $\lambda=4$
- **Opérateurs de Variation** : mutation bit-flip
 - probabilité de mutation par gène: $p = 0.2$

Remarque: on aurait pu choisir d'autres opérateurs et d'autres paramètres, bien sûr.

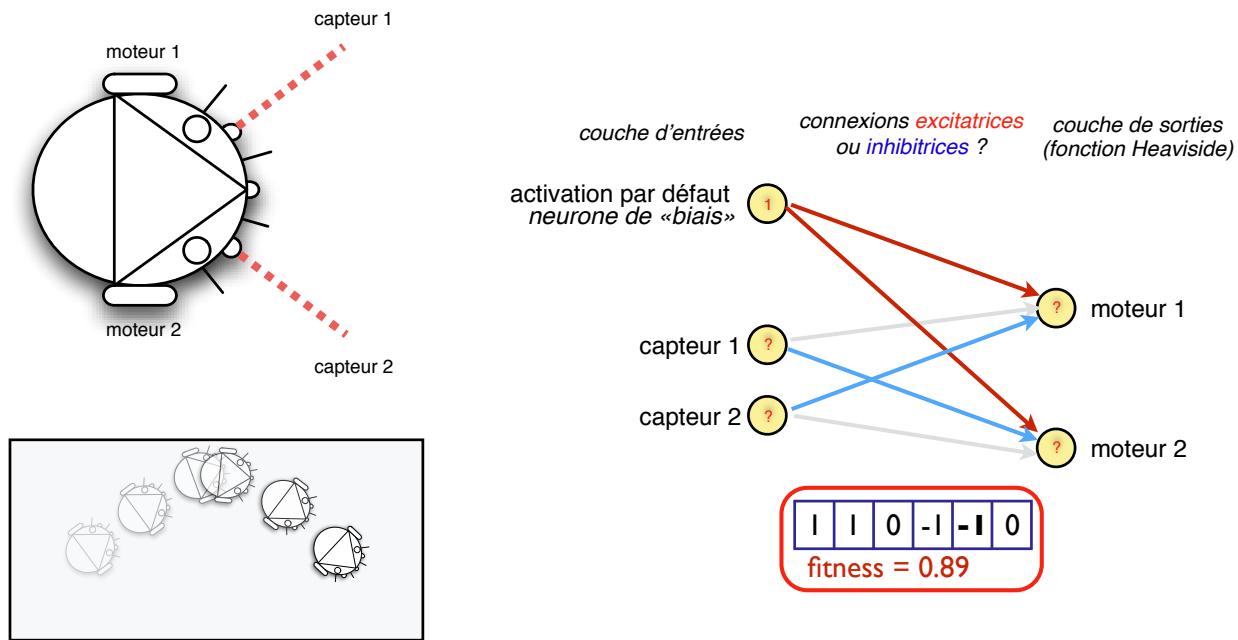
nicolas.bredeche@upmc.fr

simulation partielle du déroulement de l'algorithme

22



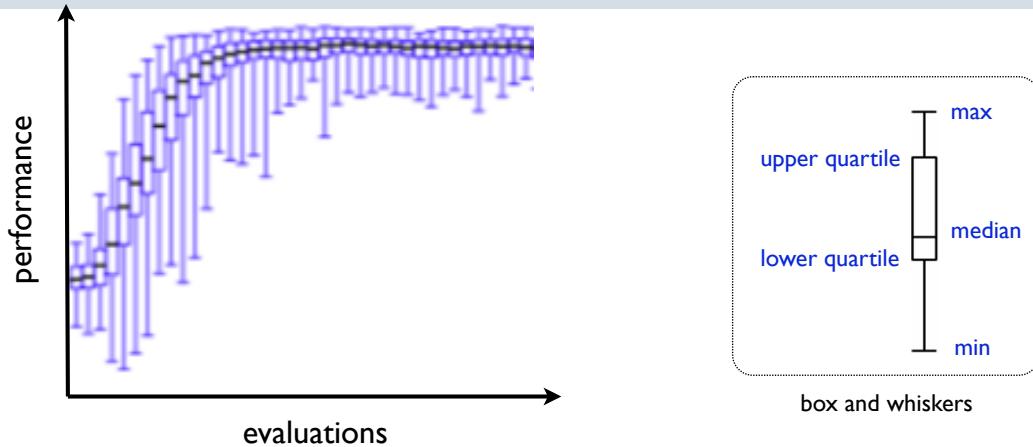
nicolas.bredeche@upmc.fr



Remarques

● En pratique

- Sur le calcul de la fitness: réévaluer pour bien estimer la qualité
- Conserver le meilleur? compromis exploration/exploitation
- Le meilleur peut apparaître avant la dernière génération: archivez!
- Généralisation: la qualité d'individu dépend du contexte
- Quand s'arrêter?
 - budget max en nombre d'évaluations
 - stagnation de l'optimisation
- Il s'agit d'une méthode stochastique, donc: faire plusieurs runs!
- La fonction fitness doit guider l'évolution. Il faut la définir avec soin



- A retenir:

- ▶ Médianes plutôt que moyennes
- ▶ Evaluations plutôt que générations
- ▶ Répéter les expériences, poursuivre jusqu'à convergence
- ▶ En pratique:
 - meilleures performances au mieux: le choix de la solution
 - meilleures performances en moyenne: le choix de l'algorithme

Cas d'étude

Le problème du max-one

exemple pratique
geneticalgorithm_maxone.ipynb

- **Objectif** : maximiser la correspondance entre une chaîne de bits et une autre (ex.: [1,1,1,1,1,1,...,1])
- **Représentation d'un individu** : $\{0, 1\}^m$
- **Population initiale** : N individus
- **Opérateur de sélection** : par tournoi de taille k
- **Opérateurs de variation** : mutation bit-flip (probabilité/bits)

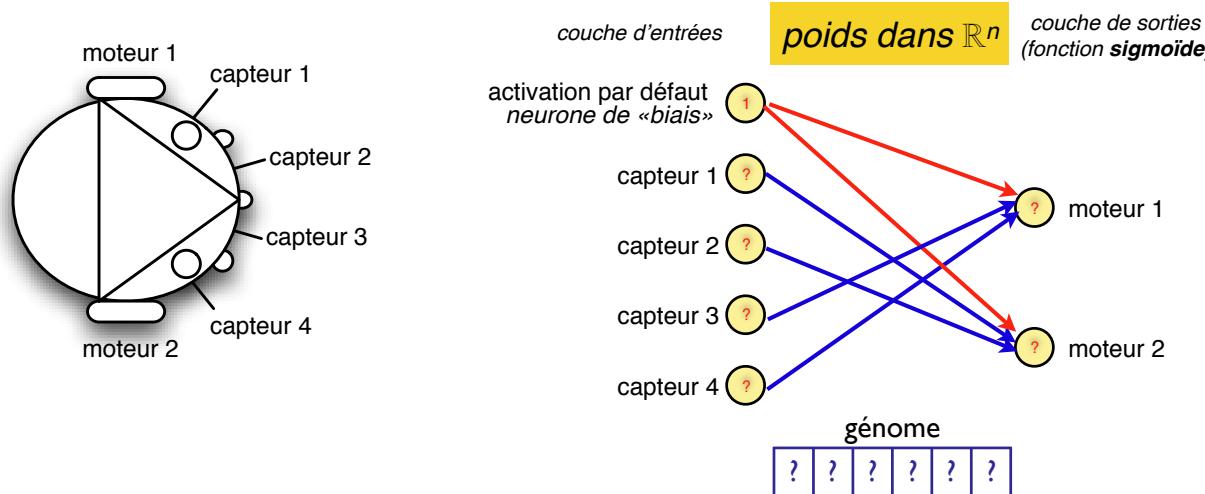
en rouge, les paramètres à régler

nicolas.bredeche@upmc.fr

Optimisation dans \mathbb{R}^n

Stratégies d'évolution

Optimiser les poids d'un réseau de neurones (défini dans \mathbb{R}^n)



$$fitness(x) = \int_T V * (1 - \sqrt{\delta_v}) * (1 - sensor_{max})$$

nicolas.bredeche@upmc.fr

Optimisation dans \mathbb{R}^n

30

Algorithme ($1+1$)-ES naïf

init: $x \in \mathbb{R}^n$

init: $\sigma > 0$

```

for i in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ )  $\geq$  fitness( $x$ ):
         $x = x'$ 

```

nicolas.bredeche@upmc.fr

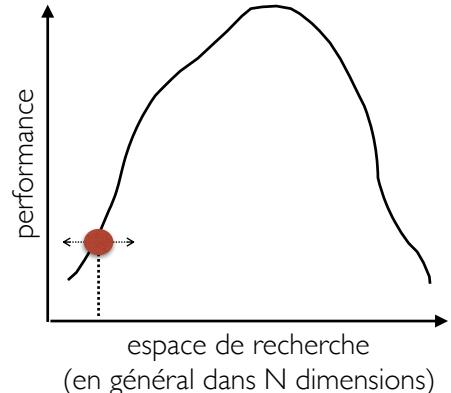
Algorithme ($1+1$)-ES naïf

```

init:  $x \in \mathbb{R}^n$ 
init:  $\sigma > 0$ 

for  $i$  in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ )  $\geq$  fitness( $x$ ):
         $x = x'$ 

```



nicolas.bredeche@upmc.fr

Algorithme ($1+1$)-ES naïf

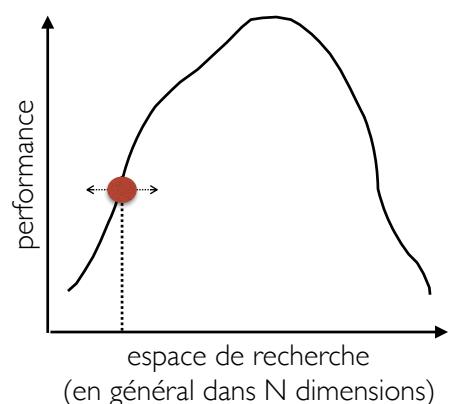
convergence lente
sigma trop petit

```

init:  $x \in \mathbb{R}^n$ 
init:  $\sigma > 0$ 

for  $i$  in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ )  $\geq$  fitness( $x$ ):
         $x = x'$ 

```



nicolas.bredeche@upmc.fr

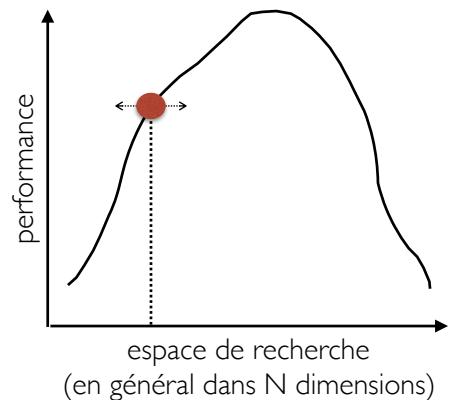
Algorithme (1+1)-ES naïf

```
init:  $x \in \mathbb{R}^n$ 
```

```
init:  $\sigma > 0$ 
```

```
for i in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ ) ≥ fitness( $x$ ):
         $x = x'$ 
```

convergence lente
sigma trop petit



nicolas.bredeche@upmc.fr

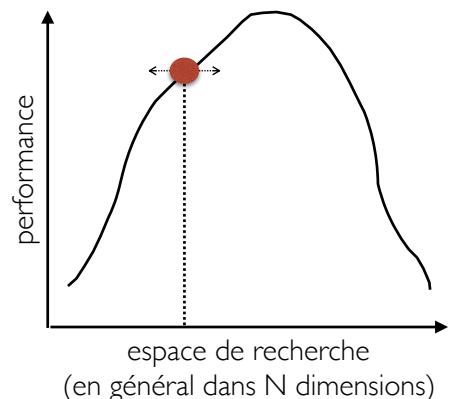
Algorithme (1+1)-ES naïf

convergence lente
sigma trop petit

```
init:  $x \in \mathbb{R}^n$ 
```

```
init:  $\sigma > 0$ 
```

```
for i in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ ) ≥ fitness( $x$ ):
         $x = x'$ 
```



nicolas.bredeche@upmc.fr

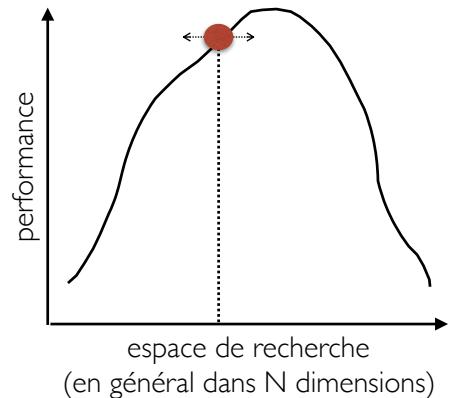
Algorithme (1+1)-ES naïf

```
init:  $x \in \mathbb{R}^n$ 
```

```
init:  $\sigma > 0$ 
```

```
for i in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ ) ≥ fitness( $x$ ):
         $x = x'$ 
```

convergence lente
sigma trop petit



nicolas.bredeche@upmc.fr

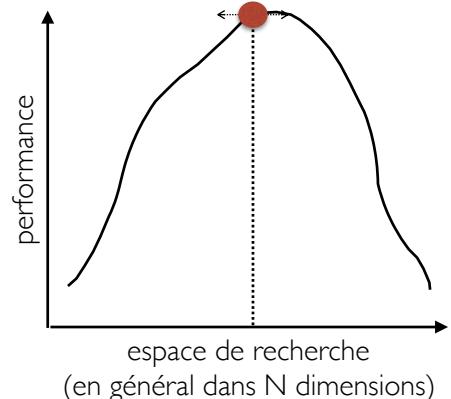
Algorithme (1+1)-ES naïf

convergence lente
sigma trop petit

```
init:  $x \in \mathbb{R}^n$ 
```

```
init:  $\sigma > 0$ 
```

```
for i in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ ) ≥ fitness( $x$ ):
         $x = x'$ 
```



nicolas.bredeche@upmc.fr

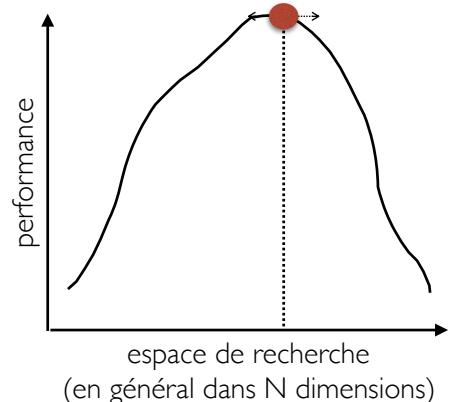
Algorithme (1+1)-ES naïf

```

init:  $x \in \mathbb{R}^n$ 
init:  $\sigma > 0$ 

for  $i$  in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ )  $\geq$  fitness( $x$ ):
         $x = x'$ 

```



nicolas.bredeche@upmc.fr

Algorithme (1+1)-ES naïf

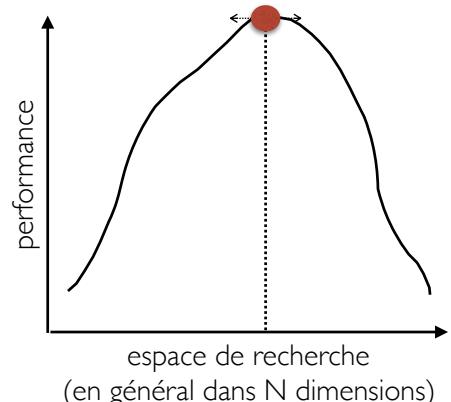
```

init:  $x \in \mathbb{R}^n$ 
init:  $\sigma > 0$ 

for  $i$  in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ )  $\geq$  fitness( $x$ ):
         $x = x'$ 

```

convergence lente
sigma trop **grand!**



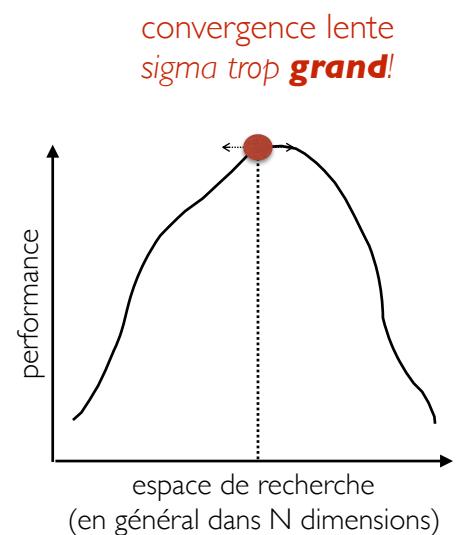
nicolas.bredeche@upmc.fr

Algorithme (1+1)-ES naïf

```
init:  $x \in \mathbb{R}^n$ 
```

```
init:  $\sigma > 0$ 
```

```
for i in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ ) ≥ fitness( $x$ ):
         $x = x'$ 
```



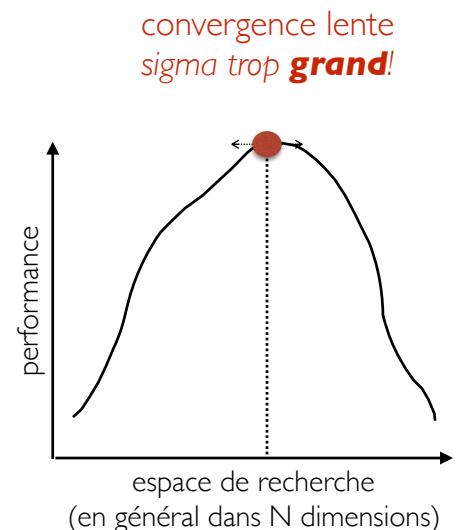
nicolas.bredeche@upmc.fr

Algorithme (1+1)-ES naïf

```
init:  $x \in \mathbb{R}^n$ 
```

```
init:  $\sigma > 0$ 
```

```
for i in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ ) ≥ fitness( $x$ ):
         $x = x'$ 
```



**oscille autour du maximum
avant de l'atteindre**

nicolas.bredeche@upmc.fr

Algorithme (1+1)-ES, règle des 1/5e

```

init:  $x \in \mathbb{R}^n$ 
init:  $\sigma > 0$ 

for  $i$  in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ )  $\geq$  fitness( $x$ ):
         $x = x'$ 
         $\sigma = 2 * \sigma$ 
    else:
         $\sigma = 2^{-1/4} * \sigma$ 

```

nicolas.bredeche@upmc.fr

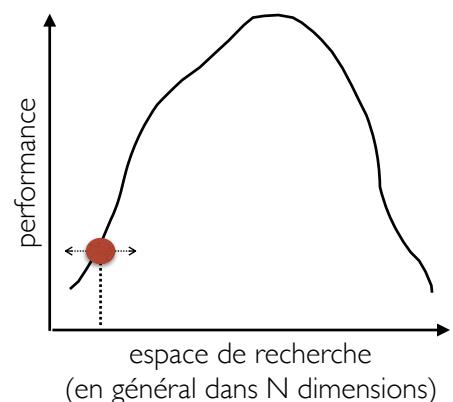
Algorithme (1+1)-ES, règle des 1/5e

```

init:  $x \in \mathbb{R}^n$ 
init:  $\sigma > 0$ 

for  $i$  in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ )  $\geq$  fitness( $x$ ):
         $x = x'$ 
         $\sigma = 2 * \sigma$ 
    else:
         $\sigma = 2^{-1/4} * \sigma$ 

```



nicolas.bredeche@upmc.fr

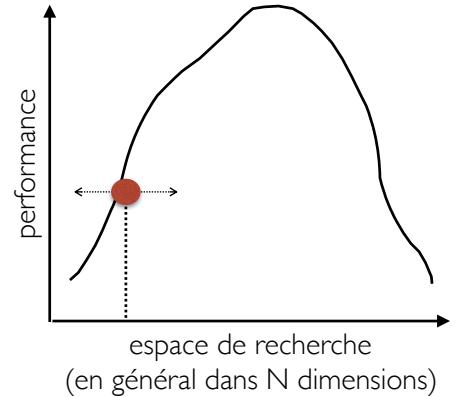
Algorithme (1+1)-ES, règle des 1/5e

```
init:  $x \in \mathbb{R}^n$ 
```

sigma augmente!

```
init:  $\sigma > 0$ 
```

```
for i in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ )  $\geq$  fitness( $x$ ):
         $x = x'$ 
         $\sigma = 2 * \sigma$ 
    else:
         $\sigma = 2^{-1/4} * \sigma$ 
```



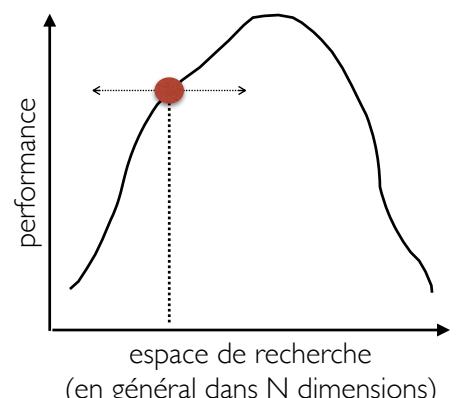
nicolas.bredeche@upmc.fr

Algorithme (1+1)-ES, règle des 1/5e

```
init:  $x \in \mathbb{R}^n$ 
```

```
init:  $\sigma > 0$ 
```

```
for i in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ )  $\geq$  fitness( $x$ ):
         $x = x'$ 
         $\sigma = 2 * \sigma$ 
    else:
         $\sigma = 2^{-1/4} * \sigma$ 
```



nicolas.bredeche@upmc.fr

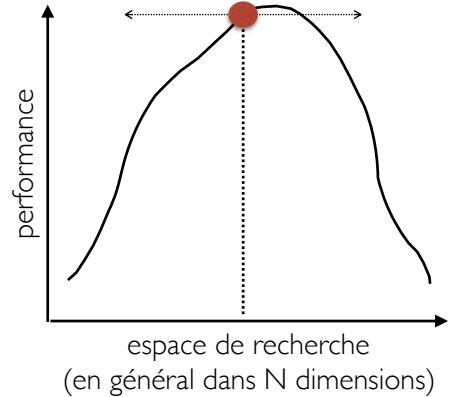
Algorithme (1+1)-ES, règle des 1/5e

```

init:  $x \in \mathbb{R}^n$ 
init:  $\sigma > 0$ 

for  $i$  in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ )  $\geq$  fitness( $x$ ):
         $x = x'$ 
         $\sigma = 2 * \sigma$ 
    else:
         $\sigma = 2^{-1/4} * \sigma$ 

```



nicolas.bredeche@upmc.fr

Algorithme (1+1)-ES, règle des 1/5e

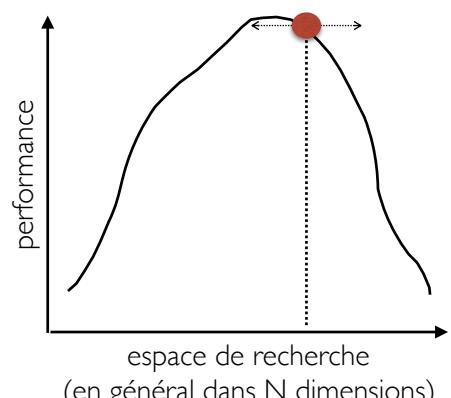
```

init:  $x \in \mathbb{R}^n$ 
init:  $\sigma > 0$ 

for  $i$  in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ )  $\geq$  fitness( $x$ ):
         $x = x'$ 
         $\sigma = 2 * \sigma$ 
    else:
         $\sigma = 2^{-1/4} * \sigma$ 

```

sigma diminue!



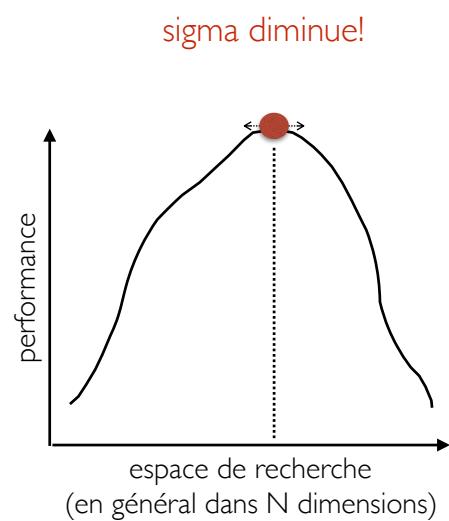
nicolas.bredeche@upmc.fr

Algorithme (1+1)-ES, règle des 1/5e

```
init:  $x \in \mathbb{R}^n$ 
```

```
init:  $\sigma > 0$ 
```

```
for i in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ ) ≥ fitness( $x$ ):
         $x = x'$ 
         $\sigma = 2 * \sigma$ 
    else:
         $\sigma = 2^{-1/4} * \sigma$ 
```



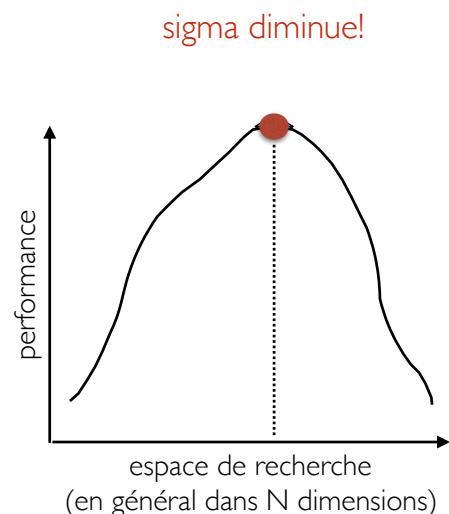
nicolas.bredeche@upmc.fr

Algorithme (1+1)-ES, règle des 1/5e

```
init:  $x \in \mathbb{R}^n$ 
```

```
init:  $\sigma > 0$ 
```

```
for i in range(evaluations):
     $x' = x + \sigma N(0, I)$ 
    if fitness( $x'$ ) ≥ fitness( $x$ ):
         $x = x'$ 
         $\sigma = 2 * \sigma$ 
    else:
         $\sigma = 2^{-1/4} * \sigma$ 
```



convergence adaptative

nicolas.bredeche@upmc.fr

Conclusions

Bonnes pratiques

50

- Evaluation
 - ▶ Une “bonne” fonction fitness vise un objectif et facilite le chemin
 - ▶ Il faut parfois réévaluer plusieurs fois une solution candidate pour avoir une bonne estimation de sa qualité
- Validité des résultats
 - ▶ Il s'agit d'un algorithme stochastique
 - ▶ Il faut faire plusieurs runs
- Utilisation dans le projet
 - ▶ vous pouvez entraîner vos robots dans une monde contrôler, en dirigeant l'évolution vers un comportement souhaité
 - ▶ Exemple: explorateur, parasite, anti-parasite, etc.
 - ▶ Vous pouvez combiner dans une architecture de Subsomption des modules évolués préalablement

Fin du cours