Robotique Évolutionnaire

Marc Schoenauer

Projet TAO – INRIA Futur et LRI Orsay

Marc.Schoenauer@inria.fr

http://tao.lri.fr/

Robotique évolutionnaire

- La biologie est une source d'inspiration inépuisable

  Voir aussi les réseaux de neurones

- Modèle d'évolution Darwinienne des espèces

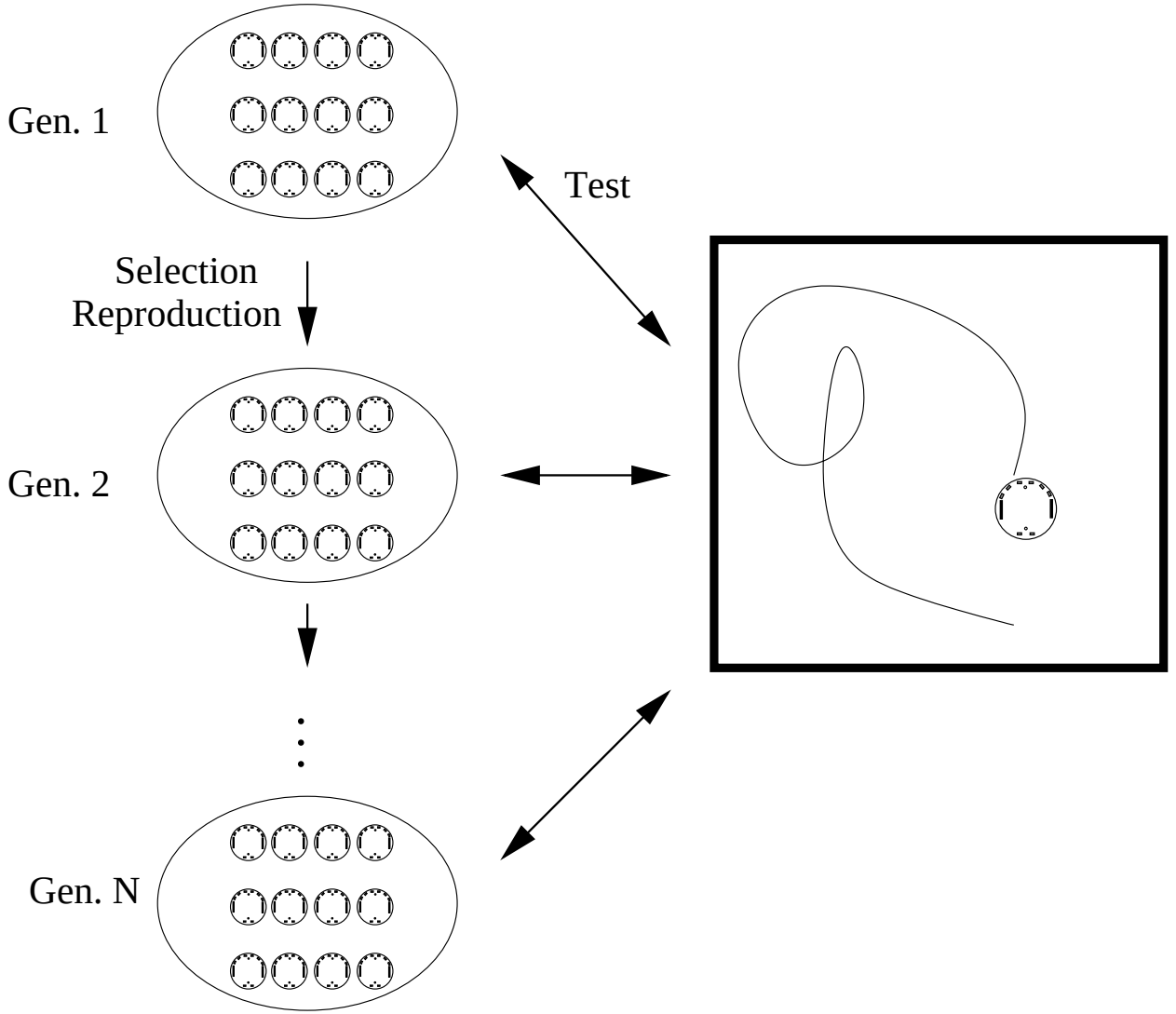
  Les plus adaptés survivent et se reproduisent

- Conjonction de la sélection naturelle et des variations apparemment non dirigées durant la reproduction

- $\longrightarrow$ "Émergence" d'espèces adaptées à leur environnement

Évolution artificielle : le programmeur crée l'environnement

Robotique évolutionnaire (2)



Contexte

- Robot autonome
- Variables d'état inaccessibles

  Position dans l'espace
- Environnement changeant et/ou inconnu Labyrinthes

  Obstacles mobiles, autres robots, . . .
- Capteurs bruités
- Plusieurs tâches différentes

  Éviter les obstacles et rassembler des objets et . . .
- Interaction avec l'environnement

Approches

But : Identifier la relation fonctionnelle capteurs – moteurs pour obtenir le comportement désiré — Optimisation

Approches descendantes — Connaissance parfaite du monde

- Contrôle optimal — connaître une trajectoire dans le “bon” espace
- Approche symbolique — IA “classique”

Approches ascendantes — Basées sur le comportement

- Robotique comportementale — Décomposition en sous-tâches
- Robotique évolutionnaire — Émergence de la décomposition

Apprentissage d'un contrôleur

- Design direct — Monde complètement connu
- Apprentissage sans modèle du monde

  En situation, i.e. en interaction avec l'environnement

Mais

- Exemples capteurs/actions en général non disponibles
- Dynamique du système souvent mal connue
- Besoin de généralisation

Un problème d'identification de fonction

- Trouver un espace de recherche pour les contrôleurs

  Une représentation, un modèle

- Trouver une fonction de performance

  comportement attendu = perfomance maximale

- Optimiser la performance dans l'espace de recherche choisi

Une solution

Les réseaux de neurones artificiels

- Approximateurs universels
- dotés de bonnes capacités de généralisation

Une performance mesurée en situation

- Réelle ou simulée
- Dépendant fortement du problème

L'évolution artificielle

- Des algorithmes d'optimisation souples et efficaces
- Mais processus encore mal compris Théorie balbutiante

Identification de fonctions

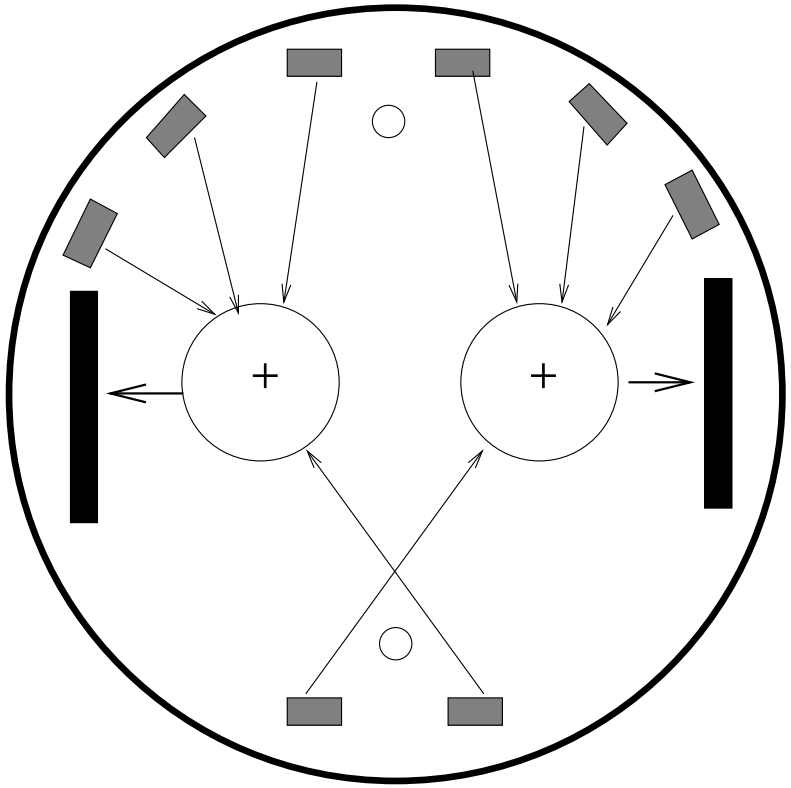
Il faut dans tous les cas un modèle pour la fonction à identifier.

- Modèles paramétriques : un nombre fixe de paramètres réels (e.g. les polynomes de degré d).
- Modèles non-paramétriques : un espace décrit par un nombre variable de paramètres (e.g. les polynômes à n variables).
- Les réseaux de neurones et la programmation génétique sont des modèles intermédiaires:
  paramètrique pour une structure fixe, non-paramétrique si l'on cherche aussi la structure.

Modèles de contrôleurs

De réactifs à symboliques, avec ou sans états internes

- Braitenbergs' vehicles (1984)

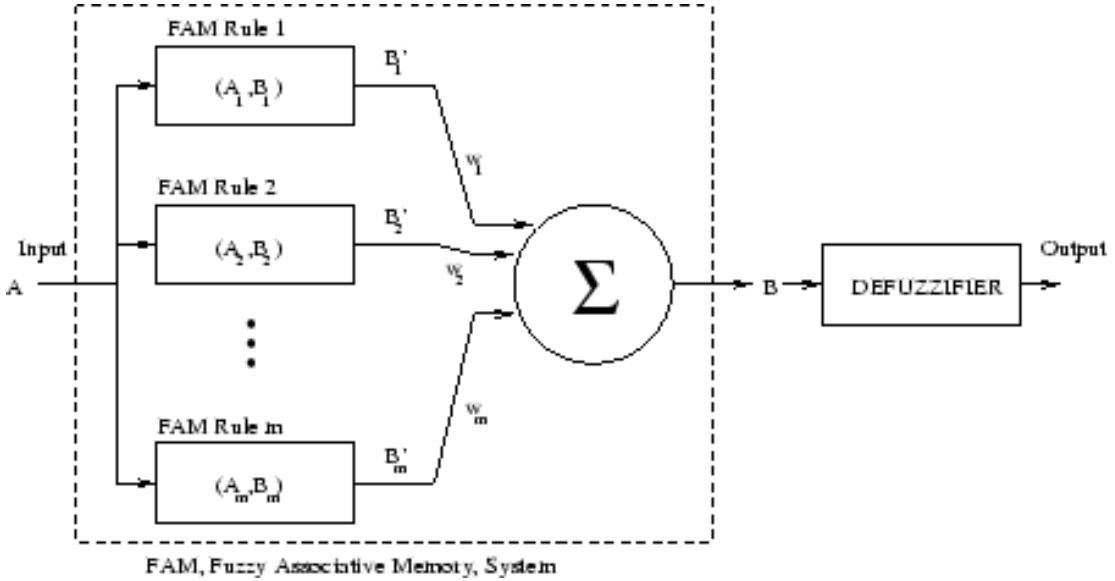


Purement réactif

Modèles de contrôleurs (2)

- Contrôleurs flous

Réactifs ou rebouclant



- Réseaux de règles symboliques

Systèmes de classeurs

```
if (true) then leftSpeed=2; rightSpeed=2;
if (leftSensor>threshold) then rightSpeed=0;
if (rightSensor>threshold) then leftSpeed=0;
if (leftSensor>threshold) and (rightSensor>threshold) then
leftSpeed=-2; rightSpeed=-2;
```

Peuvent simuler des états internes

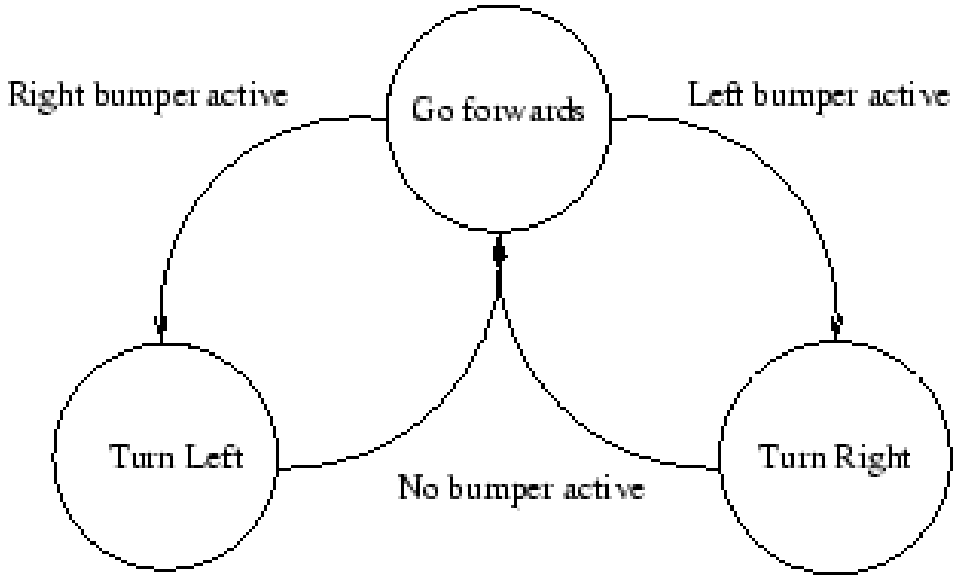
Modèles de contrôleurs (3)

- Réseaux de neurones artificiels

  Purement réactifs pour les réseaux *feedforward*

  Les réseaux récurrents ont des états internes

- Automates à états finis



- Programmation génétique

  1 programme = 1 arbre

Régression (data fitting)

On dispose d'exemples (valeurs de la fonction):

$$(\vec{X}_i, Y_i)_{i=1,...P}$$

on cherche p.ex. F de R^N dans R qui minimise

$$\sum_{i=1}^{i=P} [F(\vec{X}_i) - Y_i]^2$$

- Approximation et d'interpolation — Fonctions Splines
  Interpolation polynomiale, fractions rationelles
  Réseaux de neurones, Machines à vecteur de support (SVM)
- Le point-clé est la généralisation: que se passe-t-il pour les points non pris en compte durant l'apprentissage ?

Une mesure empirique de généralisation

Validation croisée

- Choisir un modèle — Un espace de recherche pour la fonction
- Pour chaque exemple $(\vec{X}_i, Y_i)$ apprendre F_i sur $\{(\vec{X}_1, Y_1), \ldots, (\vec{X_{i-1}}, Y_{i-1}), (\vec{X_{i+1}}, Y_{i+1}), \ldots (\vec{X_P}, Y_P)\}$
- Une mesure de généralisation est

$$\sum_{i=1}^{i=P} [F_i(\vec{X}_i) - Y_i]^2$$

Réseaux de neurones

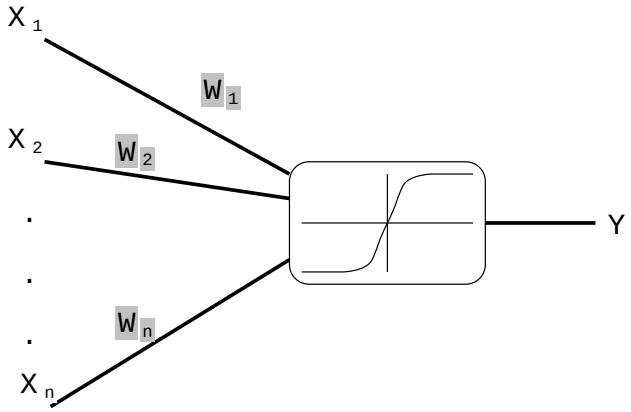
Neurones artificiels

- Perceptron (Rosenblatt 59, Widrow & Hoff 60, Minsky & Papert 69)
- Parallel Distributed Proessing (Rumelhart et McClelland 86)

Tutoriel interactif de l'EPFL en ligne localement :

`http://www.lri.fr/ marc/EEAAX/Neurones/tutorial/`

Un neurone



- Connections : des entrées et une sortie
- Poids sur les connections entrantes — Mémoire locale
- Une fonction de transfert σ

  Typiquement, la fonction sigmoide $\frac{1}{1+e^x}$
- L'activation du neurone, pour des entrées x_i, est $\sum_{i=1}^{n} w_i X_i$
- La sortie du neurone est $Y = \sigma(\sum_{i=1}^{n} w_i X_i)$

Les neurones en réseau

- Les sorties de certains neurones sont connectées aux entrées d'autres neurones.
- Les entrées non connectées sont les entrées du réseau
- Les sorties de certains neurones particuliers sont les sorties du réseau.
- Les paramètres de contrôle sont
  - l'architecture du réseau (le graphe des connections)
  - les poids des connections
- Le but recherché, le type des valeurs des entrées/sorties, le graphe des connections et l'algorithme d'optimisation des poids déterminent le type du réseau de neurones formels.

Types de réseaux

- En l'absence de boucles dans le graphe de ses connections, un RN définit une fonction de R^n dans R^m

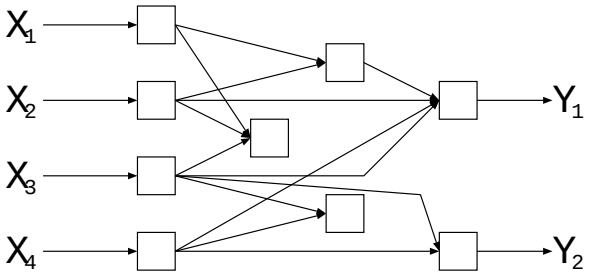
  réseaux dits à propagation directe – feed-forward

- Dans le cas contraire, on calcule les sorties de chaque neurone

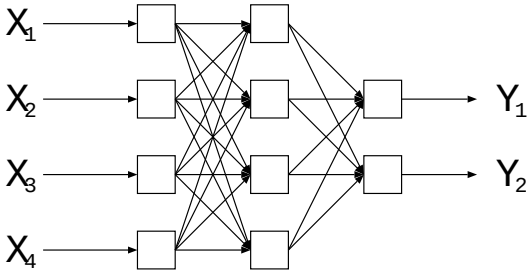
  Réseaux récurrents

  - De manière synchrone, ou
  - Dans un ordre donné

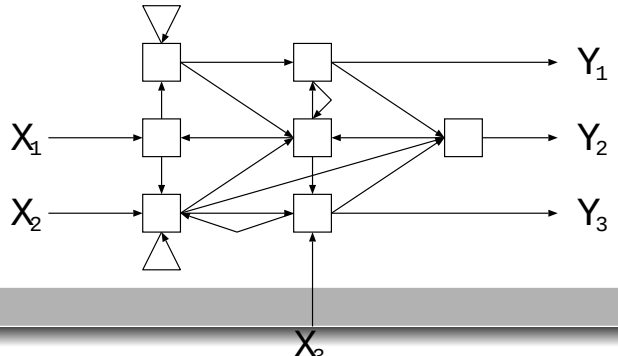
Exemples de réseaux



"Feed-forward" network



Muti-layer perceptron



Des outils d'approximation

Perceptrons multi-couches :

- L'ensemble des RN multi-couches à fonction de transfert sigmoïdale, à n entrées et m sorties (et même à 3 couches) est dense dans $L^2([0,1]^n, [0,1]^m)$.
- Pour un réseau à 3 couches ($m = 1$), majoration de l'erreur d'approximation / du nombre de neurones de la couche cachée.

De même, pour les réseaux à fonction de transfert gaussienne (RBF).

Apprentissage supervisé classique

Apprentissage des poids d'un RN de topologie fixée.

- Un ensemble d'exemples d'entrées/sorties du réseau (i.e. de $(n+m)$uplets $X_1, \ldots, X_n, Y_1, \ldots, Y_m$ est disponible Plusieurs variantes, dont la plus célèbre est une méthode de gradient stochastique, la rétro-propagation du gradient.
- Un oracle est disponible, qualifiant un réseau
  - Apprentissage par renforcement Barto et Sutton
  - Règle de Hebb

    Renforce les poids entre deux neurones actifs simultanément

Optimisation de réseaux de neurones

- Optimisation des poids d'un réseau à architecture fixée

  Éventuellement, point de départ pour le RPG

- Optimisation de la règle d'apprentissage

$$\Delta w_{ij} = \Phi(x_j, y_i, w_{ij})$$

  x_j et y_i : activités des neurones amont et aval de la connection

- Optimisation simultanée de la topologie (et des poids)

Les algorithmes évolutionnaires

- Paradigme Darwinien grossier
  - Sélection naturelle

    Les plus adaptés survivent et se reproduisent
  - Variations aveugles Indépendamment de l'adaptation

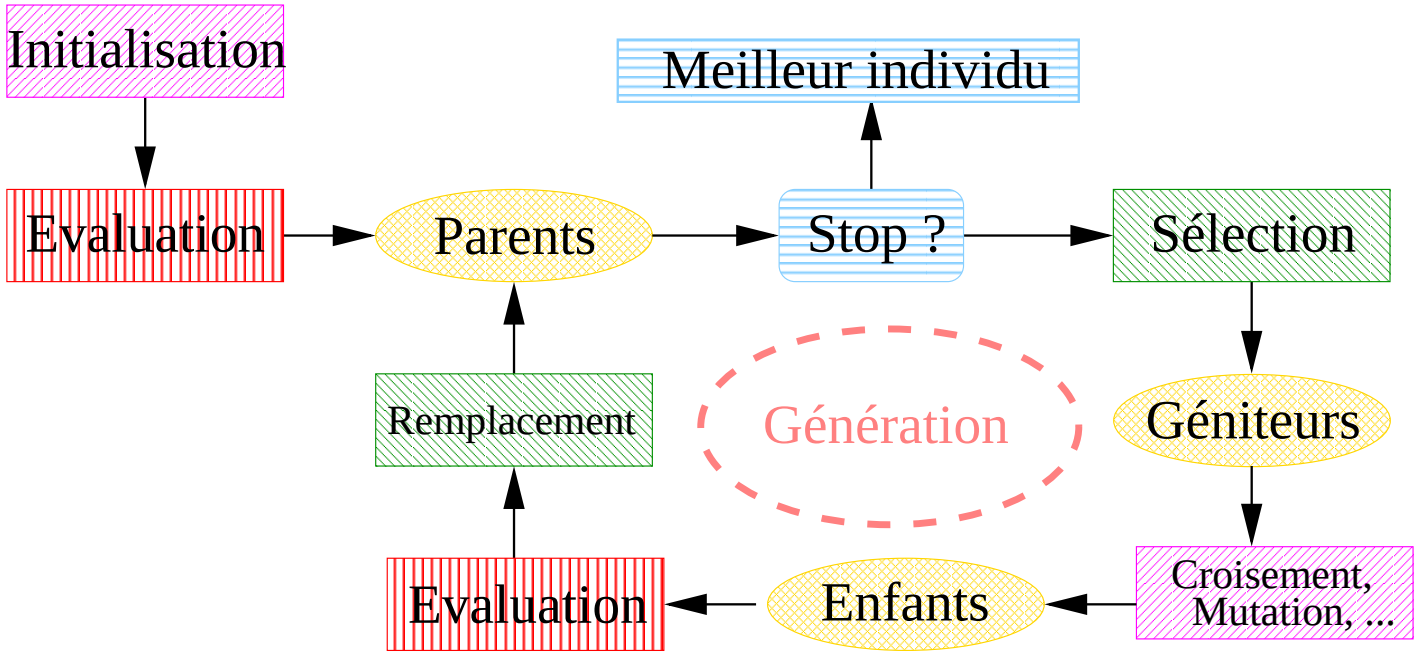
  Source inépuisable d'inspiration sans obligation de réalisme !
- Algorithmes d'optimisation stochastiques
  - Ordre 0 (cas continu) Pas de calculs de gradient

Robustesse face aux optima locaux, souplesse d'utilisation

Un algorithme “évolutionnaire”

Fonction $\mathcal{F}$ (performance, fitness) définie sur une espace Ω

Individus, population $\longleftrightarrow$ points de Ω, tuple de points de Ω



Opérateurs stochastiques: Dépendent de la représentation

"Darwinisme" (stochastique ou déterministe)

Coût calcul

Critère d’arrêt, statistiques, ...

La feuille de papier qui tombe

P. Bentley

Le problème

- Trouver la forme d'un morceau de papier
- qui met le plus de temps possible à tomber

Difficultés

- Pas de simulation
- Pas d'a priori sur la forme de la solution

Quoique . . .

Un algorithme sans ordinateur

Matériel nécessaire

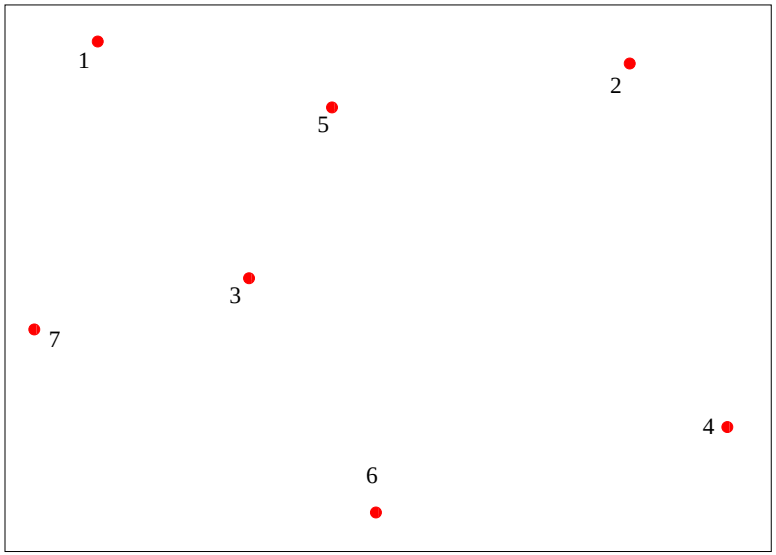
- Une ramette de feuilles A4
- Des ciseaux
- Un bac à sable format A4
- Une dizaine de petites cailloux numérotés
- Un chronomètre
- Une fléchette
- . . . de la patience

Représentation

Initialisation

- Lancer les petits cailloux en l'air au dessus du bac à sable
- Reporter les positions des cailloux tombés dans le bac sur une feuille A4

Génotype : Liste ordonnée des points

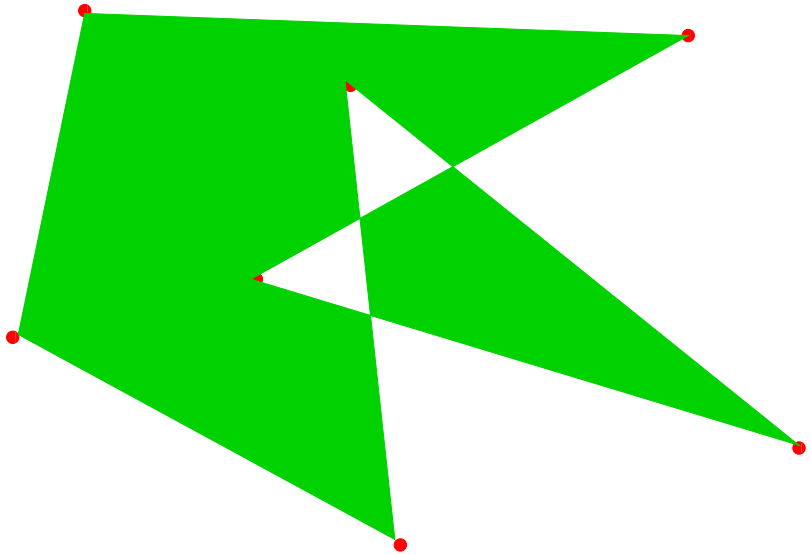


Du génotype au phénotype

Morphogénèse : passage du génotype au phénotype

- Tracer le polygone rempli correspondant à la MacPaint
- Découper la forme obtenue

Phénotype : La forme de papier



La fonction performance

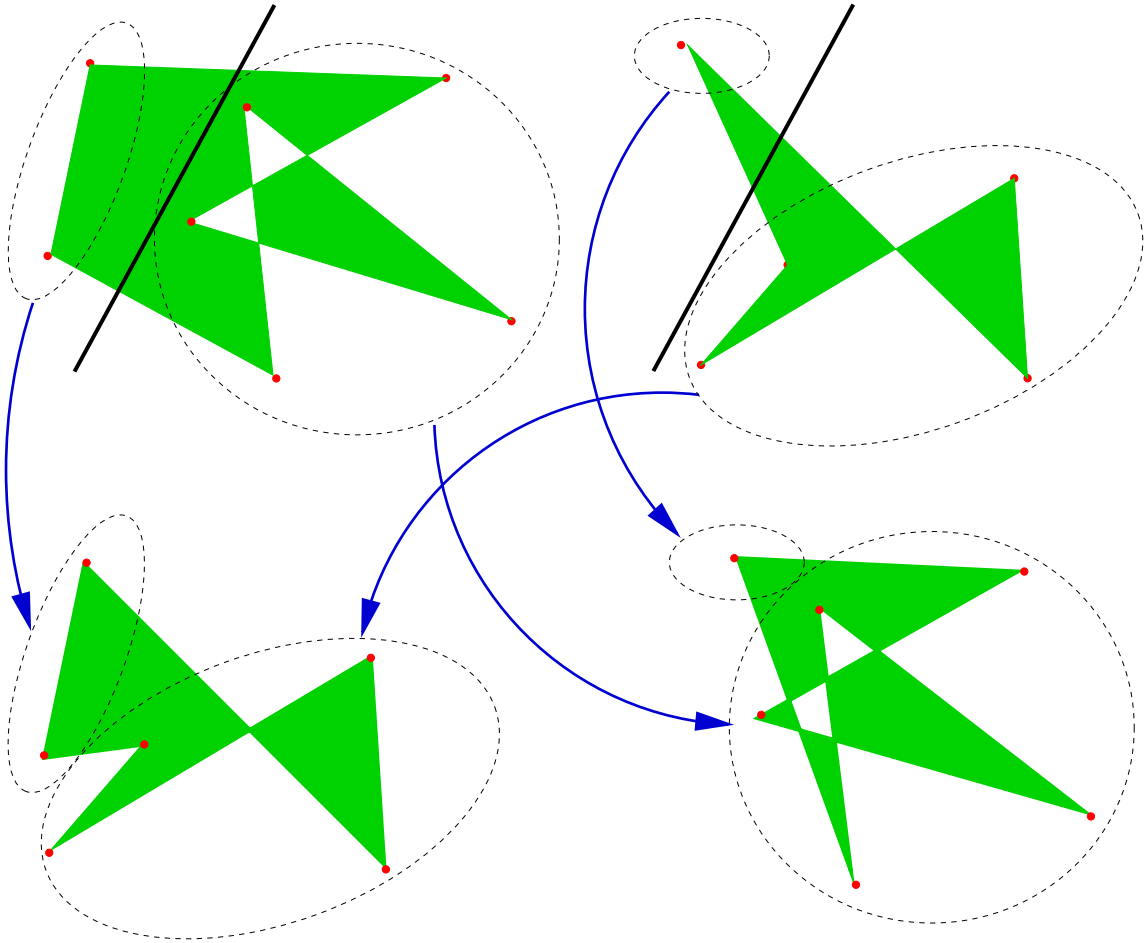
Évaluation

Pour chaque forme

- Faire 5 fois
- Lâcher la forme de 2m de haut
- Chronométrer le temps d’atteinte du sol
- Faire la moyenne des 5 temps

Opérateurs de variation

Croisement : Un croisement possible, pour lequel les ciseaux et le scotch ne suffisent pas.

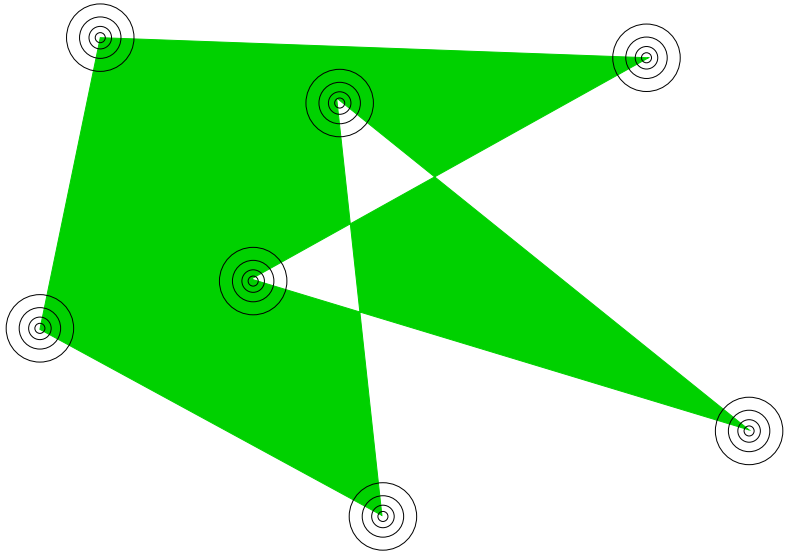


Opérateurs de variation (2)

Mutations

- Viser avec la fléchette chaque point tour à tour

  Distance à ajuster selon votre habileté



- Tracer sur une nouvelle feuille les impacts

  Si la fléchette sort du cadre, enlever le point

  Si votre portable sonne, ajouter un point

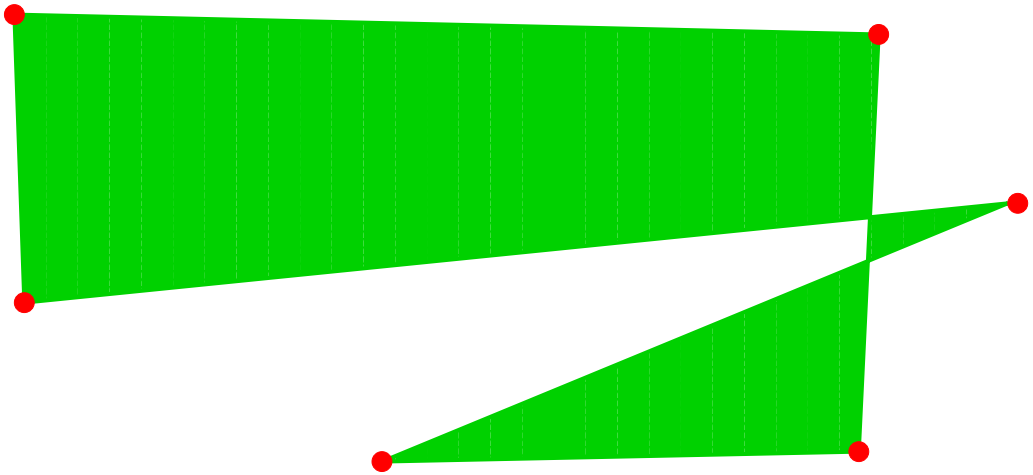
Darwinisme

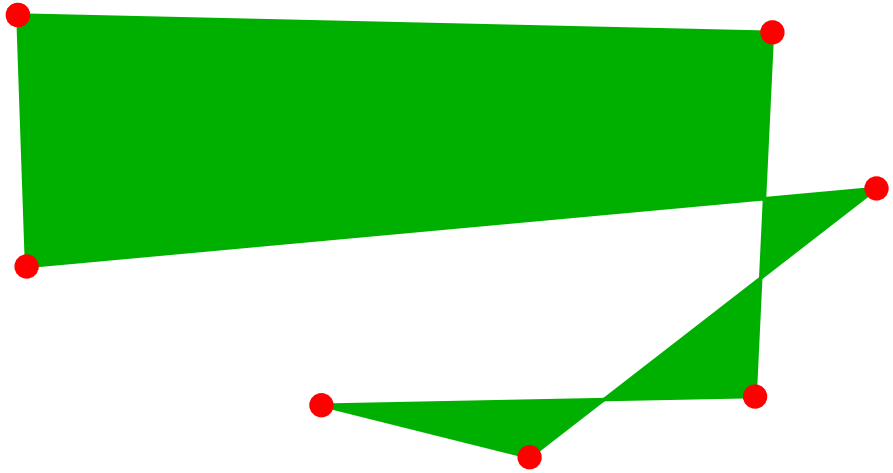
Remplacement déterministe

Déchirer la moitié des formes – les plus rapides

Résultats

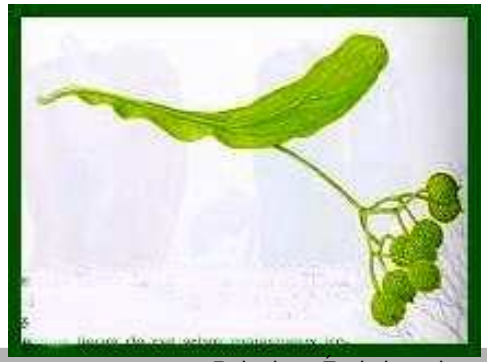
Avec 5 formes et 10 générations





Deux des meilleures formes obtenues

- Meilleure forme aléatoire : 0.8s
- Toutes les formes après 10 générations de 10 formes : $> 2s$
- Comportement “hélicoptère”

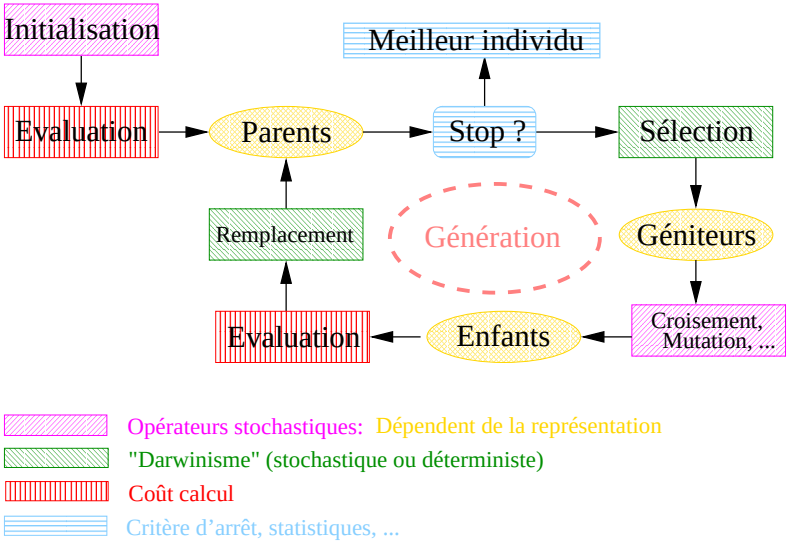


Points clé

- Une population, pas un individu
  Attention à la perte de diversité génétique

- Dilemne Exploitation vs Exploration
  - Exploitation — Recherche locale
  - Exploration — Recherche globale
  - Excès d'exploitation $\Longrightarrow$ Convergence prématurée
  - Excès d'exploration $\Longrightarrow$ Pas de convergence

Deux points de vue orthogonaux



- Le Darwinisme artificiel (sélection et remplacement) ne dépend que de la performance
- L'initialisation et les opérateurs de variation (croisement, mutation, . . .) ne dépendent que de la représentation, i.e. l'espace de recherche Ω.

Génotypes et phénotypes

- Espace génotypique $\mathcal{G}$:
  application des opérateurs de variation
- Espace phénotypique Ω: calcul de la fitness
- Morphogénèse : du génotype au phénotype

  Plusieurs génotypes peuvent donner le même phénotype
- Codage : du phénotype au génotype

L'algorithme

- manipule des génotypes (variations)
- calcule la fitness sur les phénotypes ($\rightarrow$ sélection)

Exemples suivents

History

The (3+1) roots of Evolutionary Algorithms

- Genetic Algorithms

  Holland, 75; Goldberg, 89

- Evolution Strategies

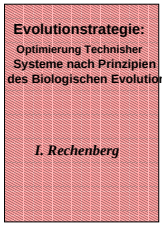
  I. Rechenberg, 73; Schwefel, 81

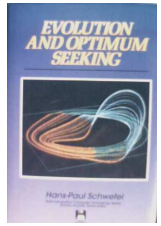
- Evolutionary Programming

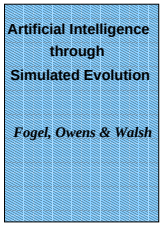
  Fogel et al., 66

- Genetic Programming

  Koza, 92



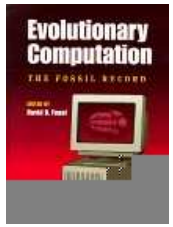


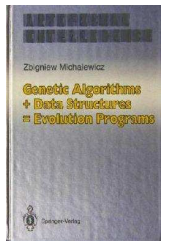
. . .

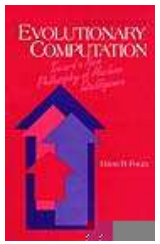
History (2)

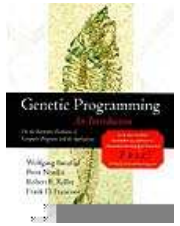
Other milestones

- The deep roots
  D. Fogel, editor, 98
- Evolution Programs Z. Michalewicz, 92 . . . and later Evolutionary Computation D. Fogel, 95
- Genetic Programming
  the textbook Banzhaf et al., 98
- Evolutionary Computation
  the textbook - best starting point
  Eiben & Smith









Quelques génotypes usuels

- Représentation “binaire” $\mathcal{G} = \{0,1\}^N$

  Algorithmes génétiques historiques

- Représentation “réelle” $\mathcal{G} = [0,1]^N$ ou $\mathbb{R}^N$ ou ...

  Algorithmes génétiques – codage réel, Stratégies d’évolution

- Représentation “combinatoire” $\mathcal{G} = \mathcal{S}_N$

  $\longrightarrow$ Genetic Local Search

- Représentation “par arbres”

  Espace d’arbres – Programmation génétique

Plusieurs représentations pour le même problème

$\longrightarrow$ Dynamiques différentes

Opérateurs de variation (non-dirigées)

Le croisement: opérateur binaire (2+ $\rightarrow$ 1)

Échange d'information "génétique" entre parents

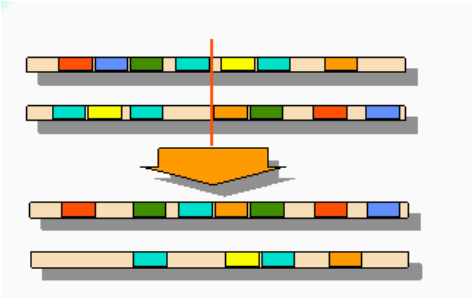
- Début d'évolution : exploration — "linéarité" de la performance ?
- Fin d'évolution : exploitation — . . . efficacité décroissante

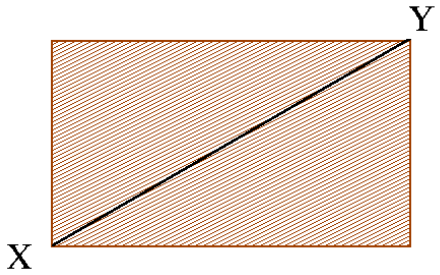
La mutation: (ré)introduction de diversité

- Ergodicité — Mais petites variations plus probables
- "Strong causality" — Continuité faible locale

Opérateurs de variation : Le croisement

Exemples classiques





Échange de gènes Croisement de paramètres réels

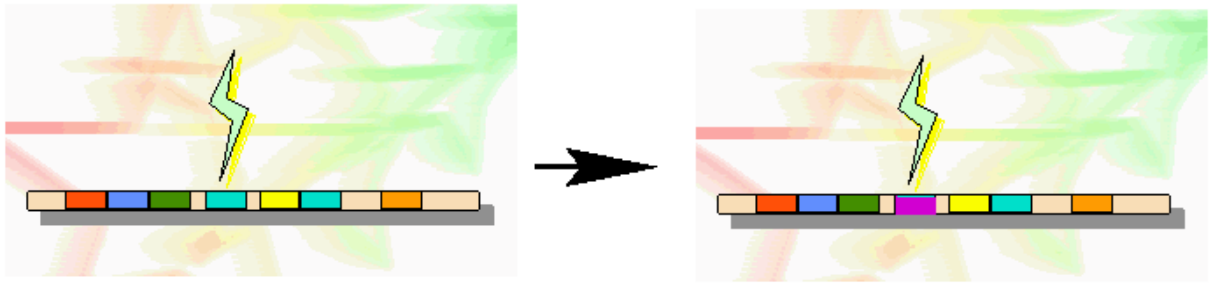
Exemple orgiaque: Cinq parents

La foule subjuguée boira ses paroles enflammées
Ce plat exquis enchanta leurs papilles expertes
L'aube aux doigts de roses se leva sur un jour nouveau
Le cadavre sanguinolent encombrait la police nationale
Les coureurs assoiffés se jetèrent sur le vin pourtant mauvais
pour un enfant surréaliste

Opérateurs de variation : La mutation

Exemples classiques

- Mutation d'un gène



- Ajout de bruit Gaussien aux paramètres réels

Un exemple sans queue ni tête

La terre est comme un orange bleue

La terre est bleue comme une orange

Pragmatisme de rigueur

Quelques résultats théoriques

- Algorithmes génétiques de base (chaînes de bits)

  Mutation et sélection couplées, convergence globale (R.Cerf, 94)

- Stratégies d'évolution adaptatives

  Convergence log-linéaire (A.Auger, 04)

Des leçons venues d'applications réussies Prochains transparents

- Prise en compte du problème
- (Trop ?) grande flexibilité

Espace de recherche – 1. Espace mixte

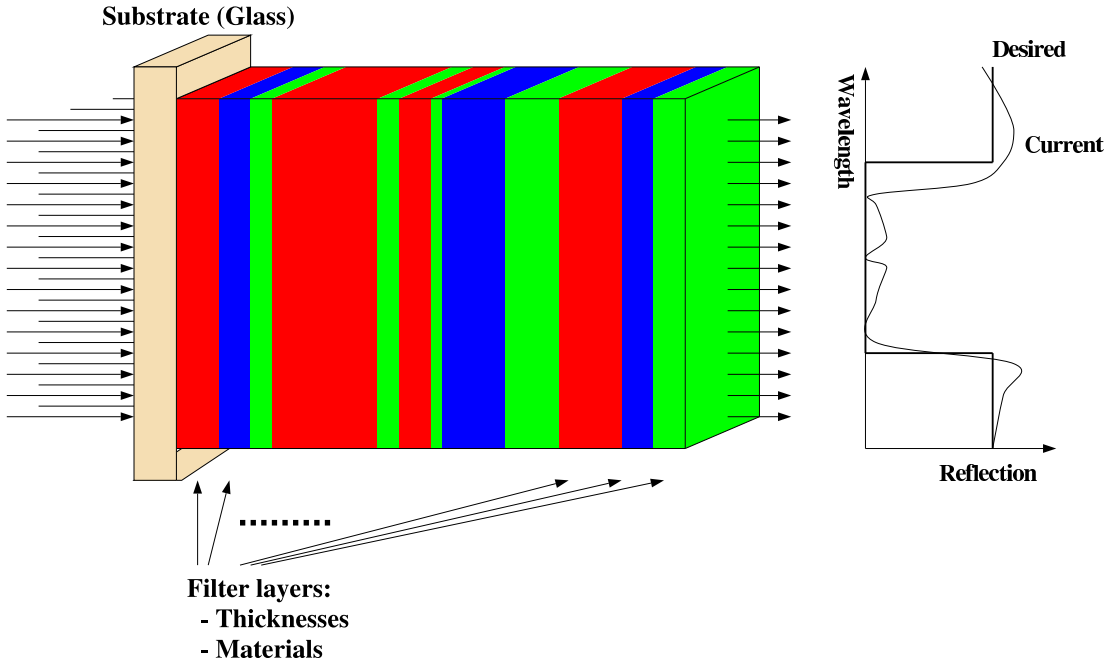
Schutz & Bäck, ICD, Dortmund.

Martin, Rivory & Schoenauer, Optique des Solides Paris VI & CMAP.

Espace de recherche : Filtres optiques

$$(\text{matériau,épaisseur})_1 \ldots (\text{matériau, épaisseur})_N$$

But : Répondre au gabarit fixé.



Espace de recherche – 2. Graphes valués

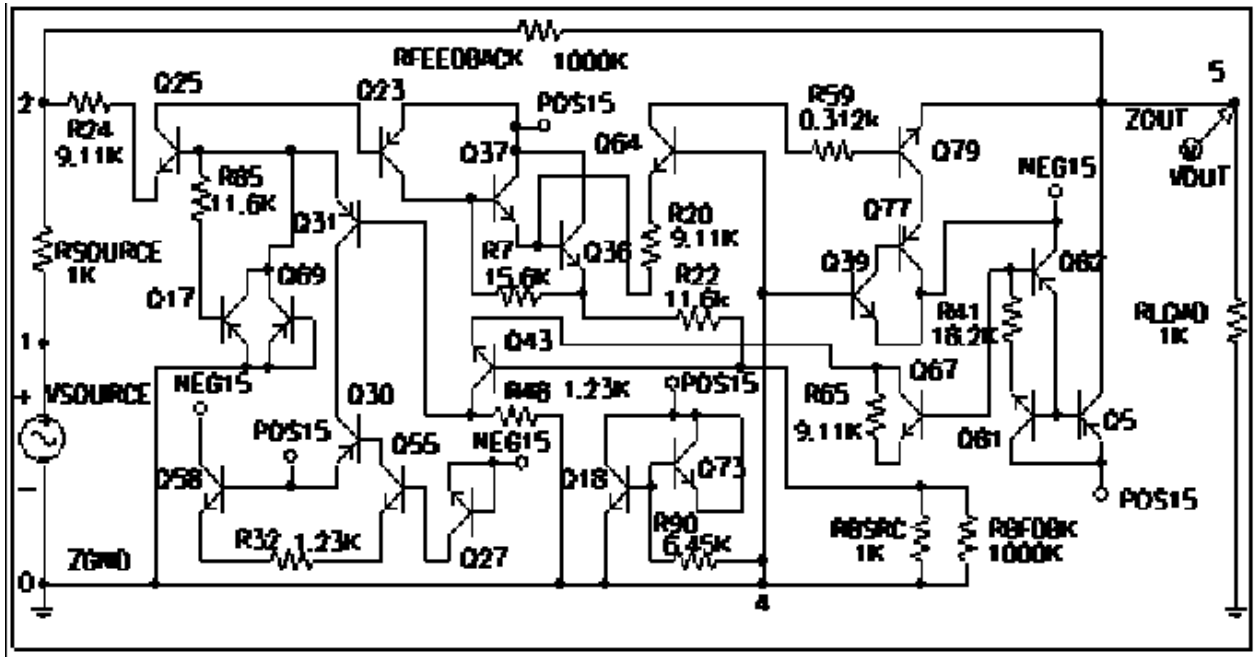
Koza et al., Stanford.

Espace de recherche : Circuits analogiques

Réseau de transistors, diodes, résistances

But : Fonctionalités fixées

e.g. extraction de racine cubique

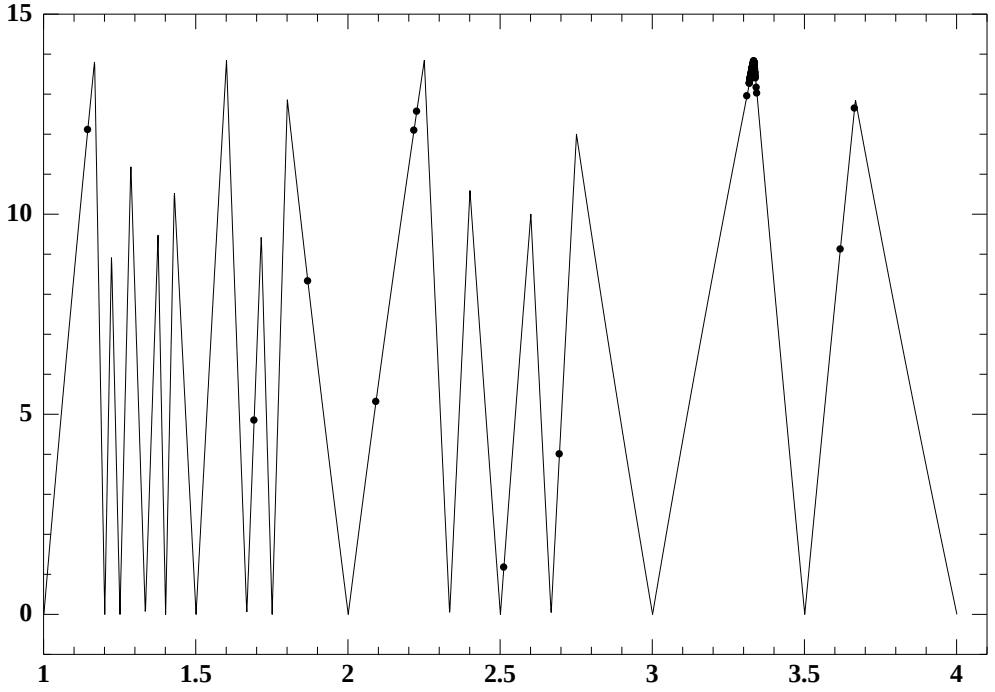


Performance – 1. très "chahutée"

L. Taïeb & MS, CMAP & Thomson

Espace de recherche : Interféromètres Positionner des antennes

But : Maximiser la tolérance en conservant la précision.



Cas de 3 antennes, $\mathcal{F}$ = Marge d'erreur (position 2ème antenne)

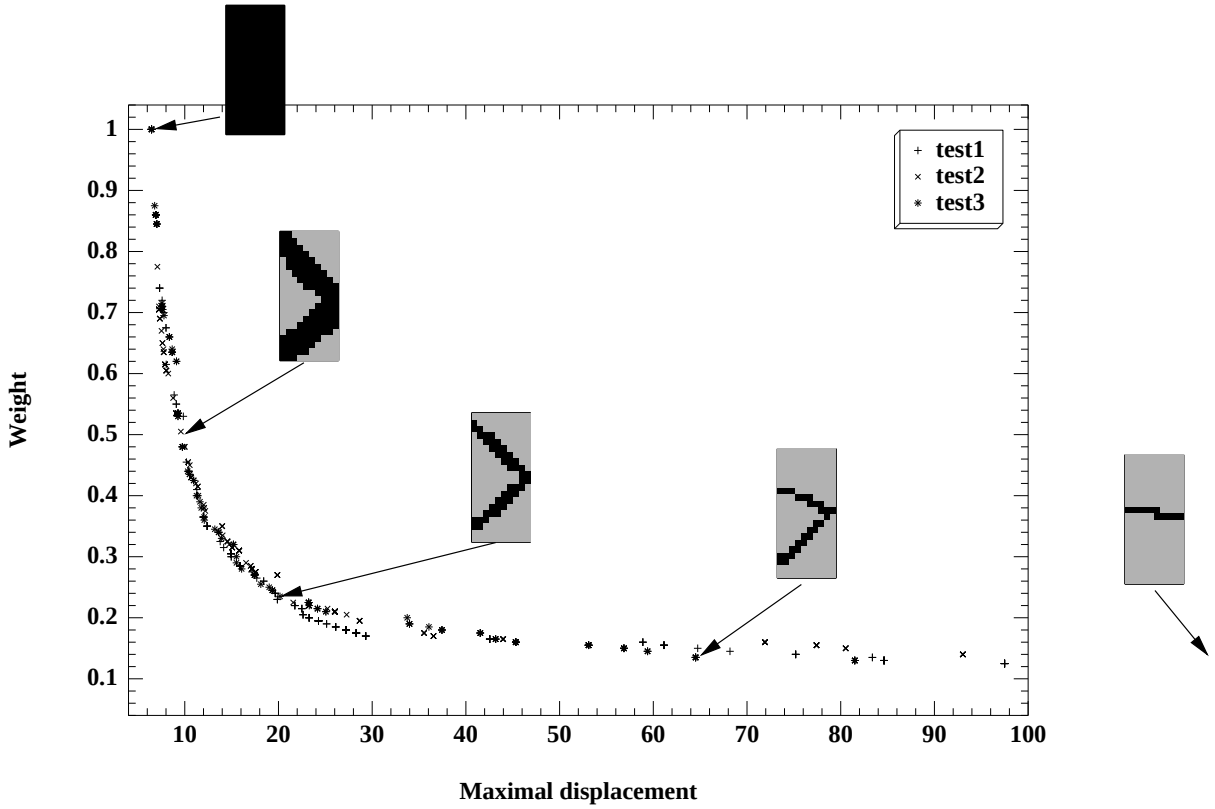
Performance – 2. Multi-objectif

Conception optimale de formes:

Trouver une forme de poids minimal et de rigidité maximale.

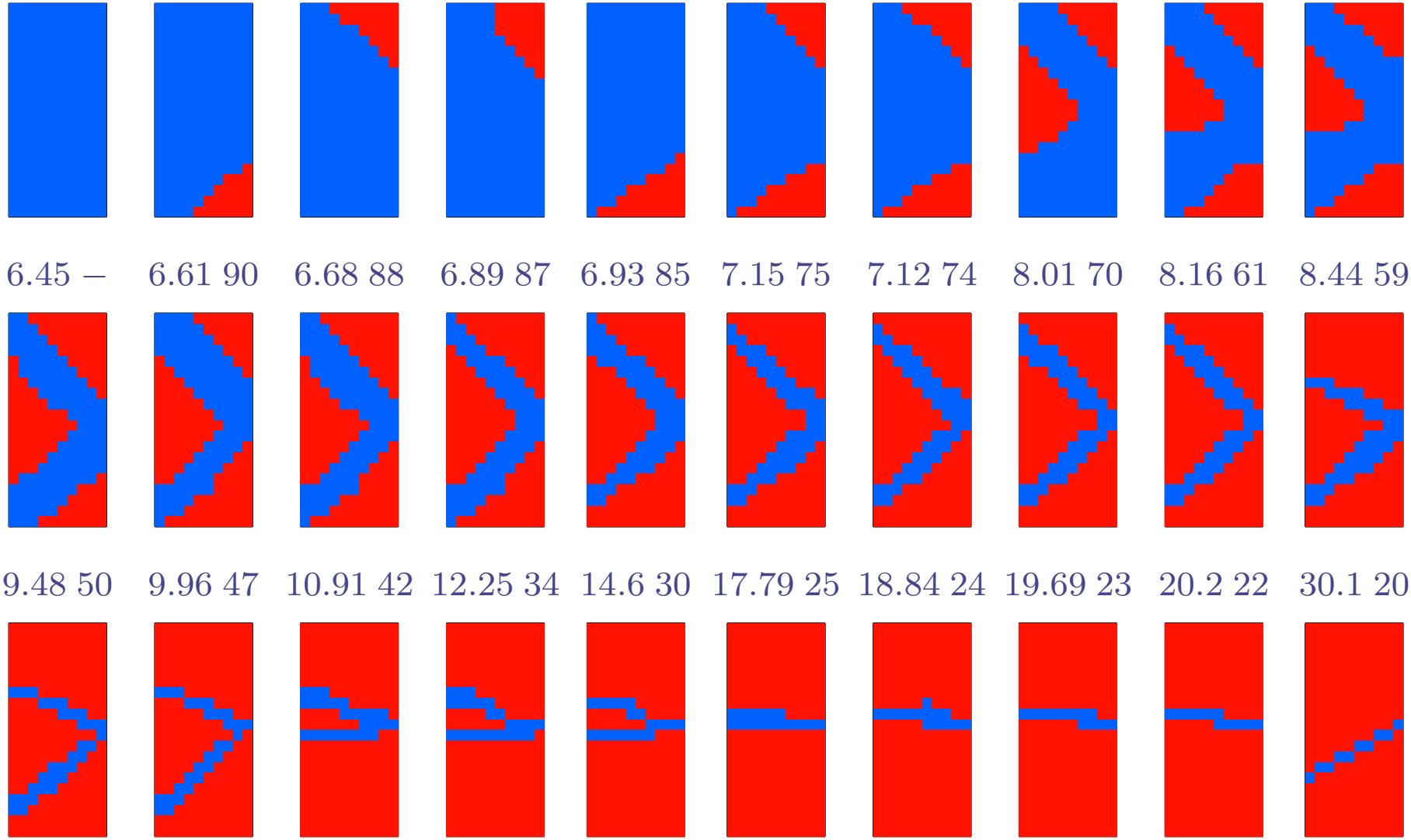
Objectifs contradictoires

Résultats dans l'espace des objectifs



Front de Pareto

Multi-objectif (2) : les solutions



Coût CPU $\equiv$ un calcul mono-objectif !

Performance – 3. non calculable

Herdy & al., Berlin, PPSN96

Espace de recherche : Mélanges de café

But : Retrouver un arôme

$\mathcal{F}$ = note de l'expert

Deux exemples plus détaillés

- Optimisation paramétrique

  Espace de recherche = $\mathbb{R}^n$

  - Poids d'un réseau de neurones
  - ou coefficients d'un polynône, d'une fraction rationelle, . . .
- Programmation génétique

  Espace d'arbres

  - Le contrôleur lui-même
  - Le programme qui contruit le contrôleur

Normal mutations for real-coded EAs

Basic mutation:

- add Gaussian noise

$$X_i := X_i + \sigma\mathcal{N}(0,1)$$

variance σ

- more general Gaussian perturbation

$$X := X + \sigma\mathcal{N}(0,C)$$

step-size σ, covariance matrix C

- Tuning of σ and C crucial

Adaptation of Gaussian mutation

History

- $\sigma \propto^{-1}$ fitness — Early EP, difficult to tune
- The 1/5th rule:
  Increase σ when more than 20% successful mutations

  Theoretical studies on sphere and corridor fns

  Can be wrong on less regular landscapes
- Self-adaptive mutations:
  - Each individual has its own mutation parameters
  - Mutate the mutation parameters
  - Mutate the individual
- Covariance Matrix Adaptation — Derandomized self-adaptation

Self-adaptive mutations

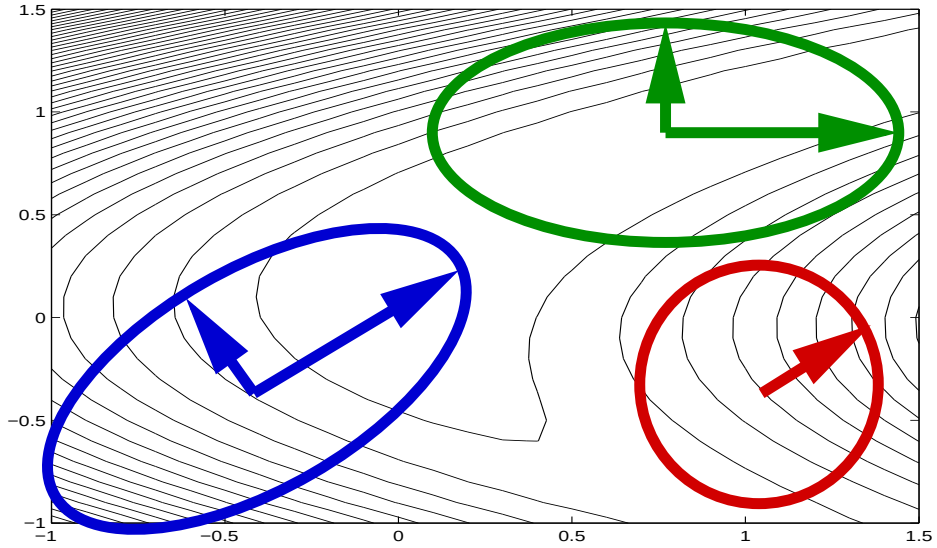
Rechenberg, 73; Schwefel, 81

- Isotropic: One σ per individual, $C = I_n$
- Non -isotropic: One σ_i per variable, $C = \text{diag}(\sigma_1, \ldots, \sigma_n)$
- Correlated: C positive definite $\quad n(n-1)/2$ rotations

Isotropic mutation

Non−isotropic mutation

Correlated mutation

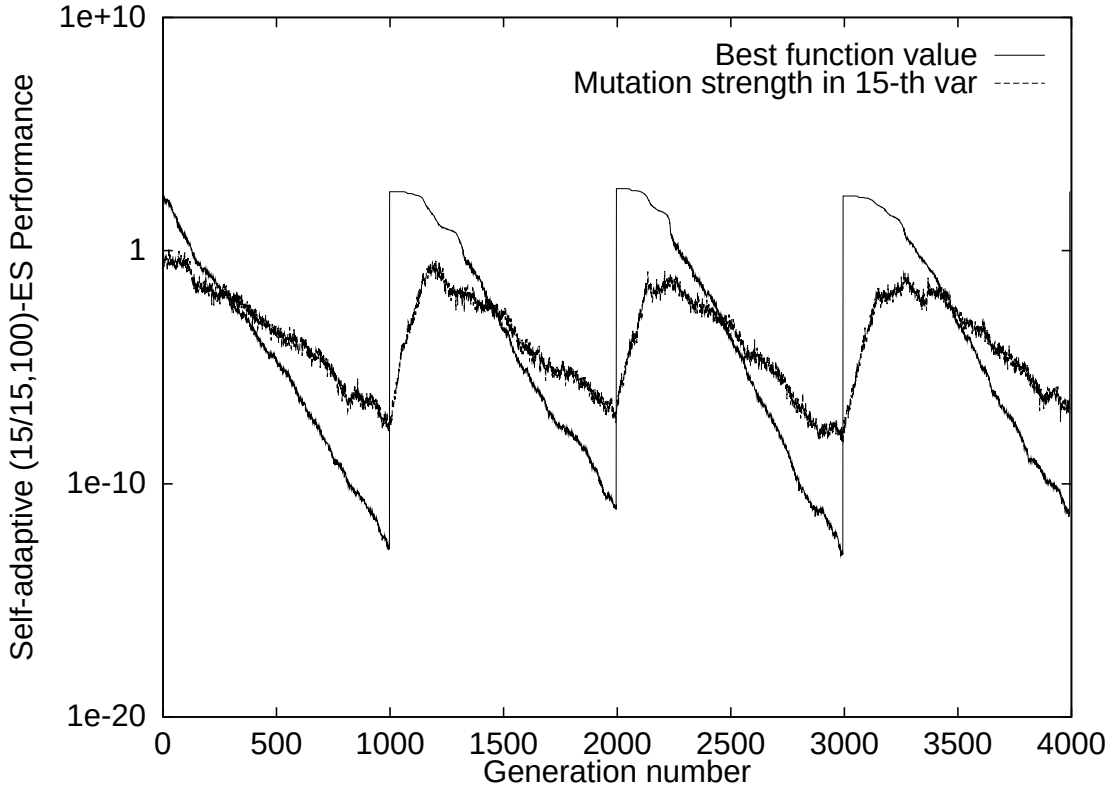


Evidence of self-adaptivity – SA-ES

Deb & Beyer 01

Experiments on dynamic landscape

Quadratic function, random moves of minimum every K generations



Back to “simple” adaptivity

Hansen & Ostermeier 95-01

- Self-adaptivity can be very slow

  e.g. for the full covariance matrix

- The actual path contains local information on the landscape
  - Consecutive steps in colinear directions $\rightarrow$ increase step-size
  - Similarly, add direction information to the covariance matrix

  Rank 1 update Hansen & Ostermeier, 96

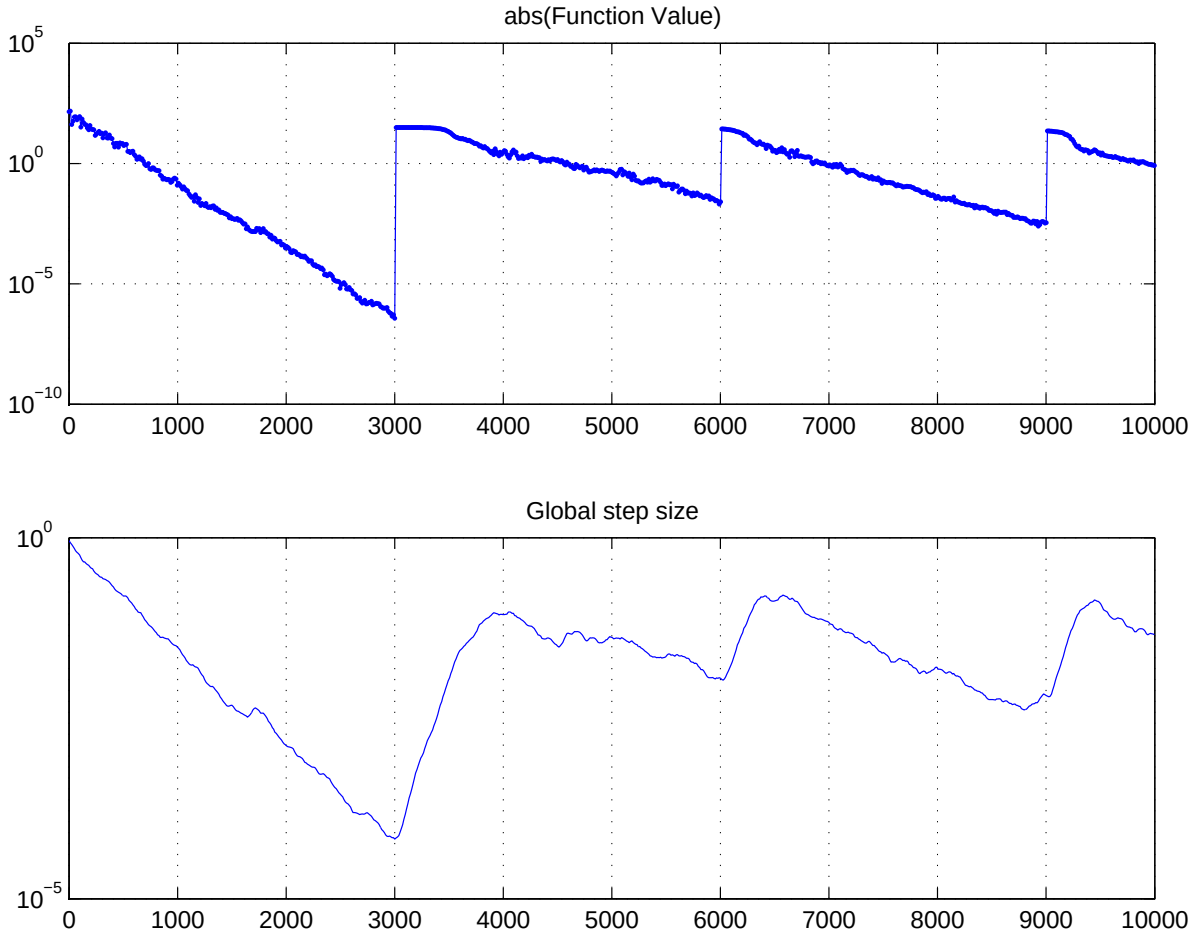
  Rank μ update Hansen, Müller, & Koumoutsakos, 2003

$\longrightarrow$ CMA-ES – Covariance Matrix Adaptation

Evidence of self-adaptivity – CMA-ES

Monitor the step-size

Same dynamic quadratic function



Closing the loop: Quasi-Newton methods

The “ideal” covariance matrix is the inverse of the Hessian

- Deterministic optimization methods
- $$X_{n+1} = X_n - \alpha \hat{H}^{-1} \nabla f(x)$$

  where $\hat{H}^{-1}$ is an approximation of H^{-1}
- Main issue: Find a good approximation of H^{-1}!
- Iteratively, based on finding conjugate directions
  - Update of rank one,
  - BFGS,
  - ...

Programmation génétique

Espaces d'arbres

Un ensemble $\mathcal{N}$ de noeuds (ou opérateurs)

Un ensemble $\mathcal{T}$ de feuilles (ou opérandes)

Ω = Arbres($\mathcal{N}, \mathcal{T}$)

Exemples :

- $\begin{cases} \mathcal{N} = \{+, \times\} \\ \mathcal{T} = \{X, \mathcal{R}\} \\ \Omega = \text{ Polynomes de } X. \end{cases}$
- $\begin{cases} \mathcal{N} = \{ \text{ if-then-else, while-do, repeat-until,..}\} \\ \mathcal{T} = \{\text{expressions, instructions}\} \\ \Omega = \text{ Programmes} \end{cases}$

Points clé :

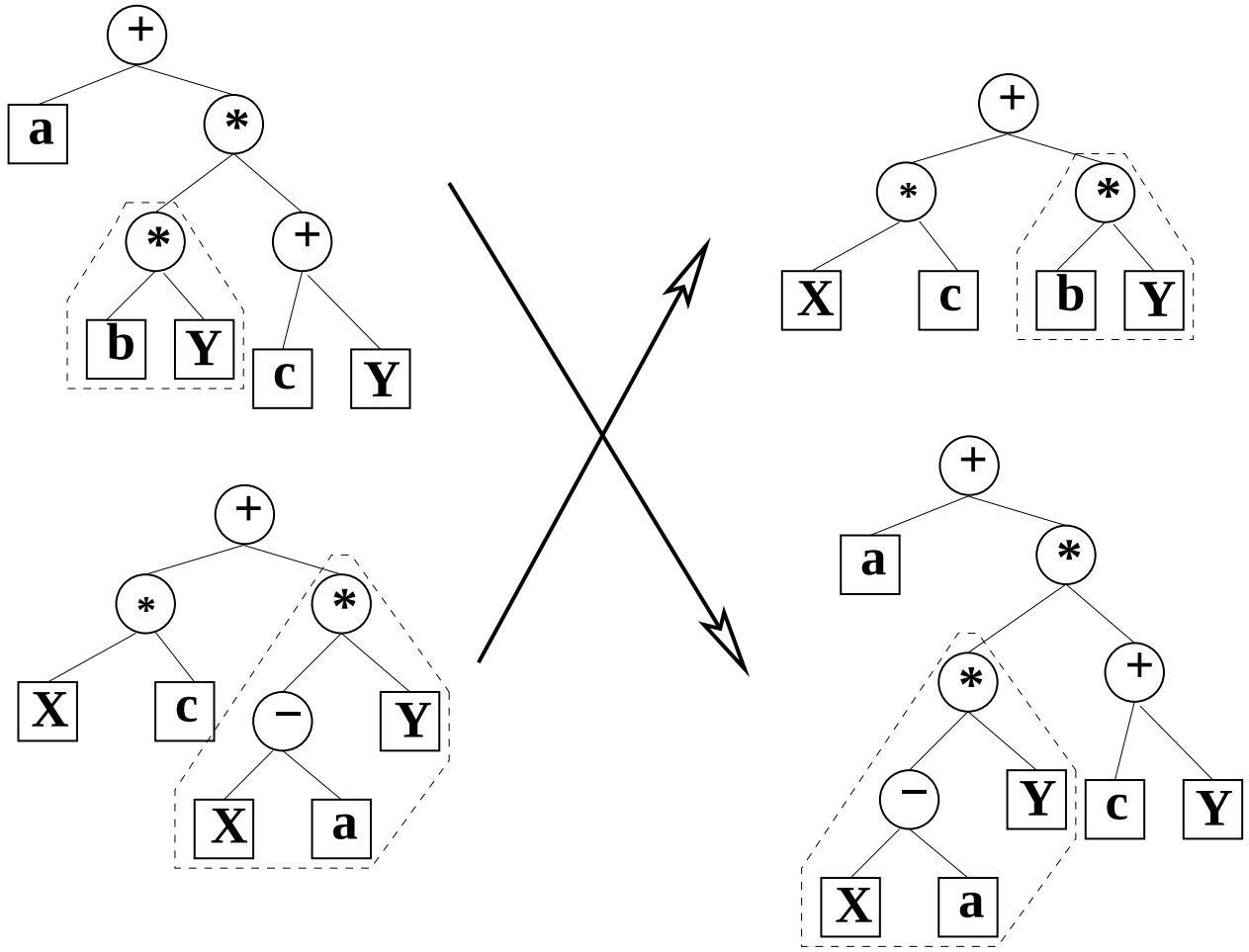
Structure d'arbre / Longueur variable

Croisement

Principe :

Deux points sont choisis dans les deux parents

Les sous-arbres sous ces points sont échangés.



Croisement (2)

Points délicats :

- Vérifier que la longueur max des arbres est respectée.
- Tout croisement est-il possible ?

Strongly-typed GP

Remarque :

1. Opérateur historiquement privilégié de GP.
2. “Le croisement suffit à produire des mutations” . . .
3. Le croisement produit un enfant long et un enfant court.

Bloat

Mutations

Remarque :

1. Opérateur historiquement absent de GP (#popu $>$ 2000).
2. Plusieurs types de mutation.

Mutation traditionnelle

- Remplacement de sous-arbre par un arbre aléatoire
- Remplacement de noeud par un noeud de même arité

Mutation “promotionnelle”

- Insertion de noeud Faire du fils d’un noeud son petit-fils
- Promotion de noeud

Remplacer un noeud par un de ses noeud-fils

Mutation: Terminaux numériques

Points délicats :

- Mutation structurelles: Modifient la structure de l'arbre
- Très peu de "petites" modifications possibles

  $\Rightarrow$ Ajustement des constantes nécessaire

Mutation "numérique"

- Muter quelques constantes dans l'arbre très lent
- Muter toutes les constantes dans l'arbre

  peut être très destructeur
- Optimisation:
  - tirer n jeux de constantes, garder le meilleur
  - coupler optim. non paramétrique/paramétrique

    de type montée, ou . . . évolutionaire

Un premier exemple

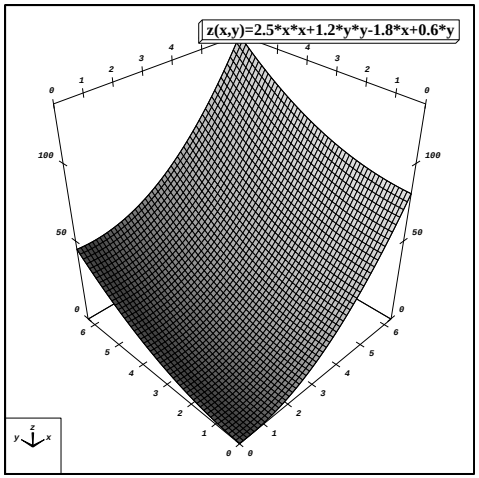
- Régression "symbolique"
- 50 points (X0,X1,P(X0,X1)), P polynôme connu
-

$$Fitness = \sum_{i=0}^{i=50} [Tree(X0, X1) - P(X0, X1)]^2$$

Remarque: Il faut avoir de bonnes raisons pour utiliser GP pour la régression !

Régression symbolique

$$P(X0, X1) = 2.5 * X0^2 + 1.2 * X1^2 - 1.8 * X0 + 0.6 * X1$$



Meilleur résultat: version postfixée

((-1.42455)+((-2.57915)+((2.32683)+(X1+(X0+((X1+((X0*(X0+X0))+(X1*X1)))+((-0.968281)+((0.0169015)*((X1*(-15.3847))+((X1+(((X0*X1)+(X1-(X0+((-18.7284)+(X0+((X0*X1)+X1))))))+(X1+((-7.21495)*(X1+((-19.0404)+((4.32199)*(((-0.381033)*(X1*X1))+(X1+X1)))))))))-(X0*((14.7915)*((11.0556)-(X0+X0))))))))))))

Après simplification par Maple

$$2.499997075 * X0^2 + 1.200819057 * X1^2 - 1.797686828 * X0 + 0.597758036 * X1 - 0.006760359$$

Retour à la robotique

- Modèle : les réseaux de neurones

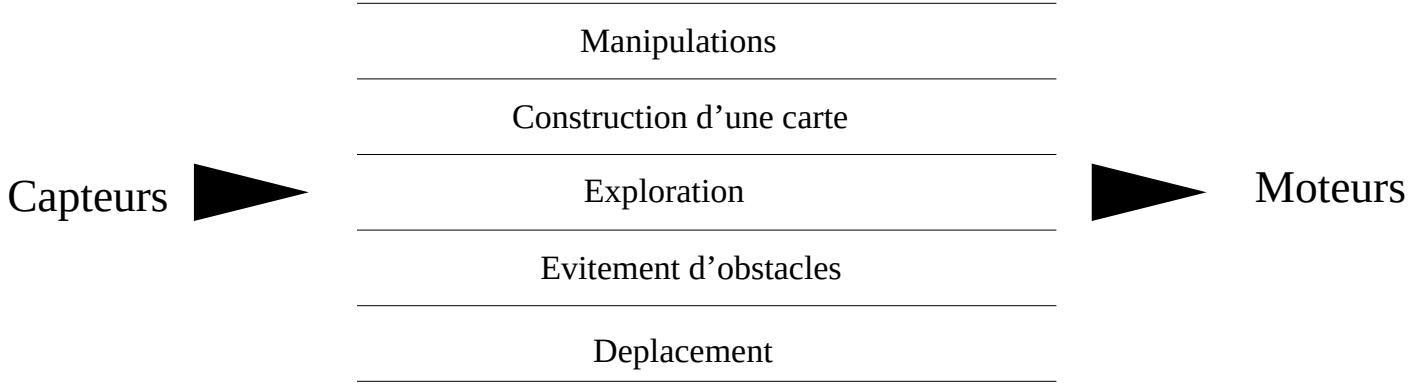
  Feedforward ou récurrents

- dont on optimise
  - soit les poids
  - soit aussi la topologie
- par évolution artificielle

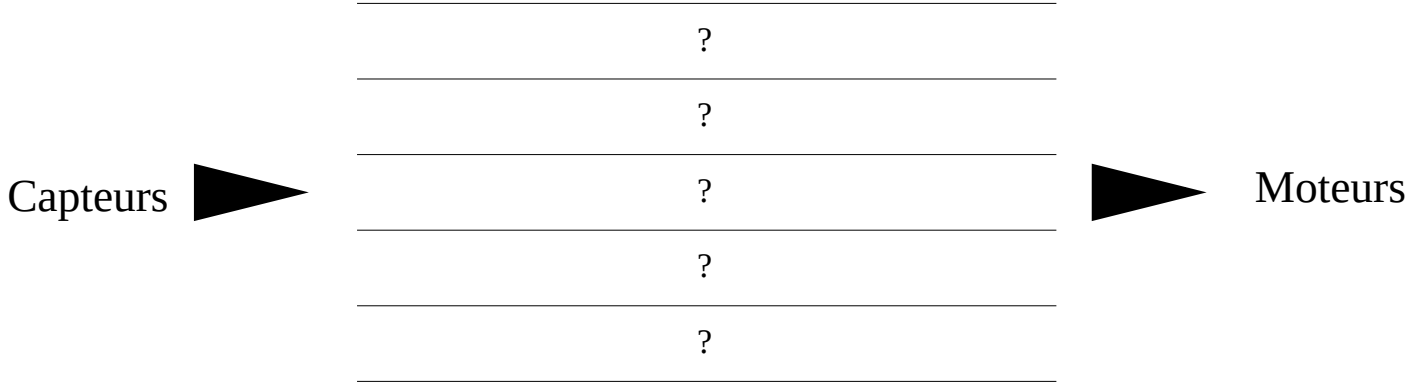
mais selon quel critère ?

La fonction performance

Décomposition comportementale



Robotique comportementale : L'utilisateur décompose le problème



Robotique évolutionnaire : La décomposition émerge durant l'évolution

La fonction performance (2)

- Fonctionnelle vs comportementale

  État du contrôleur vs Distance parcourue

- Implicite vs explicite

  Survie vs Distance à la prise de courant

- Variables internes vs externes

  Capteurs embarqués vs informations du superviseur

- Co-évolution : Évolution de plusieurs robots

  La performance dépend des autres robots

Méthodes traditionnelles $\equiv$ Fonctionnelle, explicite et externe

Robots autonomes $\equiv$ Comportementale, implicite et interne

Performance subjective $\equiv$ Comportementale, explicite et externe

Émergence de la complexité (2)

L'évolution artificielle ne peut pas tout

Quelques pistes :

- Évolution incrémentale — Problème du bootstrap
- Apprentissage durant la "vie" du robot (on-line)
- Évolution et développement

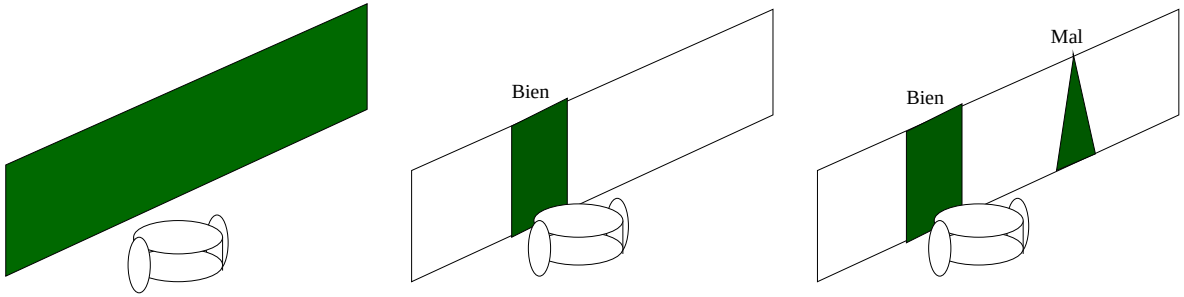
  Évolution simultanée de la morphologie et du contrôleur

Incrémentalité

- Le problème du bootstrap

  Tous les individus aléatoires sont nuls

- Partir d'une tâche simple
- La compliquer graduellement durant l'évolution
- Exemple: reconnaissance visuelle



Simulation ou expérience réelle

- Dans la réalité, les capteurs sont bruités, les roues patinent, . . .
- Mais une évaluation réelle prend 10 à 100 fois plus de temps que la même simulée,
- quoique tout dépende de la précision de la simulation cf K. Simms
- Il faut de toute façon bruiter artificiellement la simulation

Mais pas n'importe comment !

Validation finale sur robot réel

Simulation minimale Nick Jacobi, 1997

Quelle doit être la précision de la simulation ?

- Bon sens: le plus fidèle possible à la réalité

  J. Pollack et al., K. Simms, . . .

- Mais seules certaines interactions auront lieu si l'évolution réussit
- . . . et donc seules celles-là doivent être bien modélisées
- Exemple : locomotion d'un robot insecte à 6 pattes

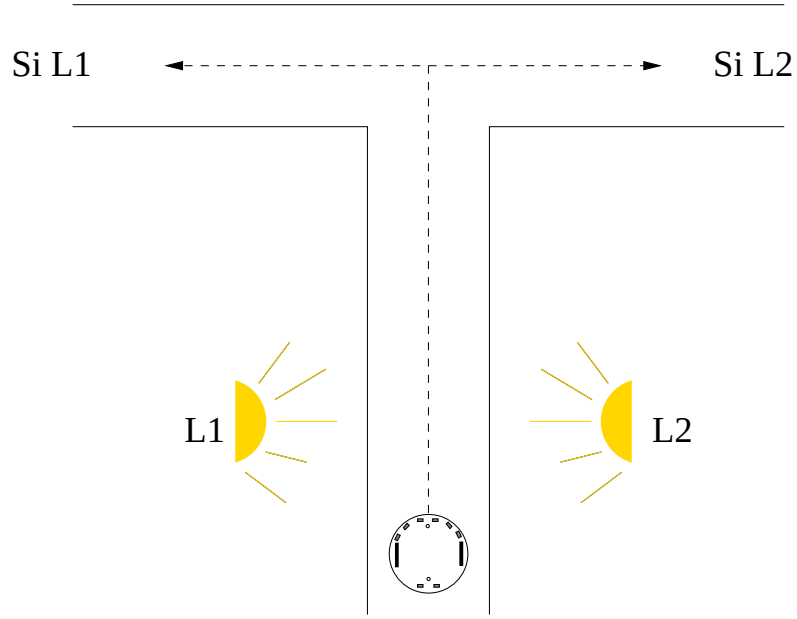
  Les chocs entre les pattes n'auront pas lieu

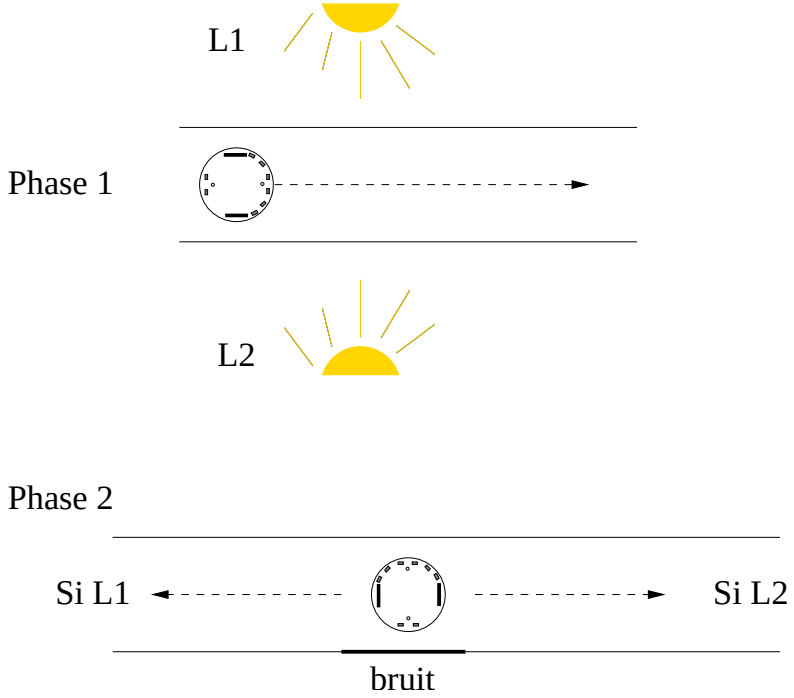
  Modélisées simplement + pénalité

Simulation minimale (2)

Le labyrinthe en T

- Choisir le bon embranchement en fonction d'un lumière





Environnement réel Environnement minimal

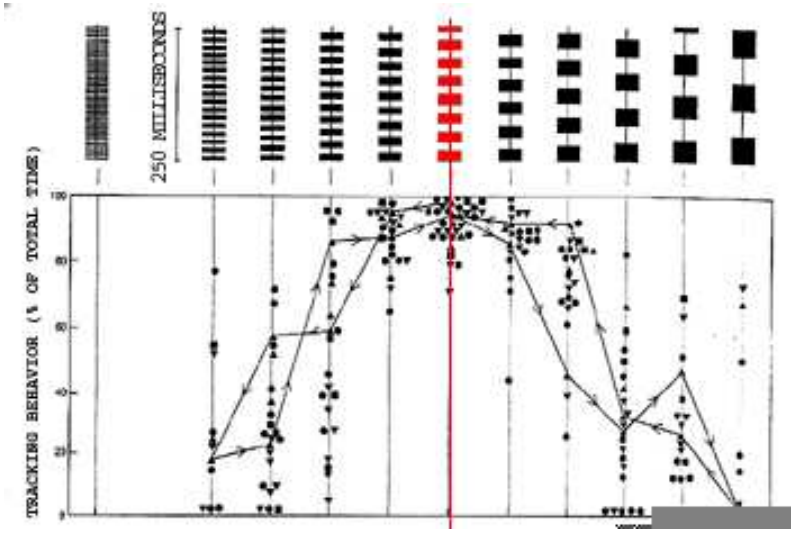
- Mais interdit certaines solutions

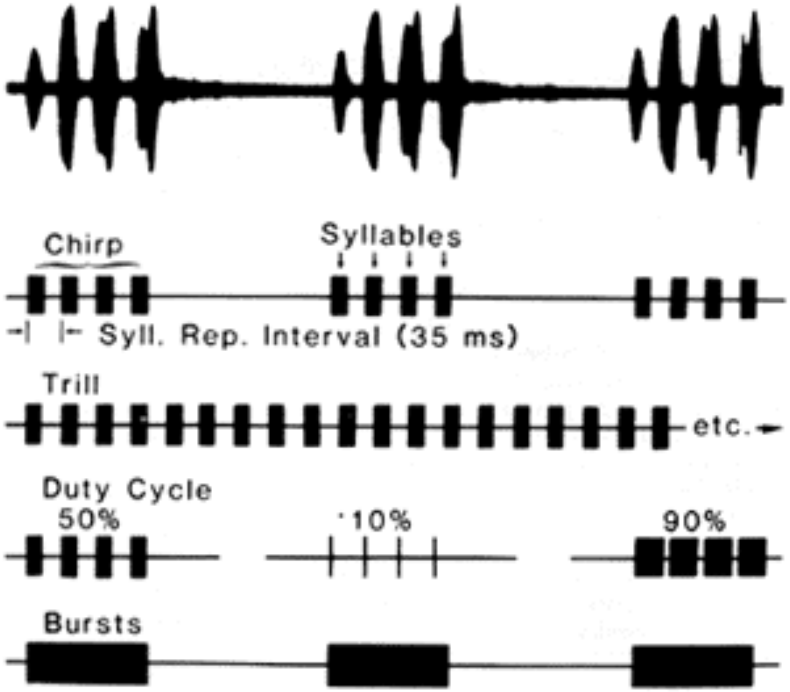
Suivre le mur après avoir "vu" la lampe

Ne pas se tromper de problème

Phonotaxie du cricket – B. Webb, 96

- Phonotaxie du cricket — Distingue les races et la direction
- Mécanisme cognitif complexe ?

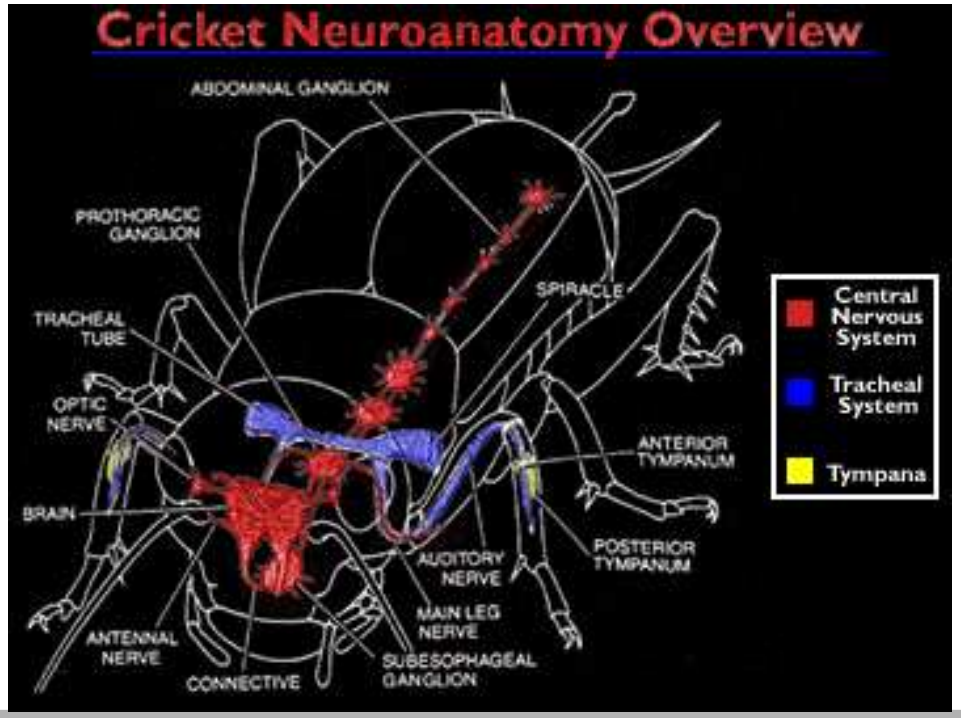


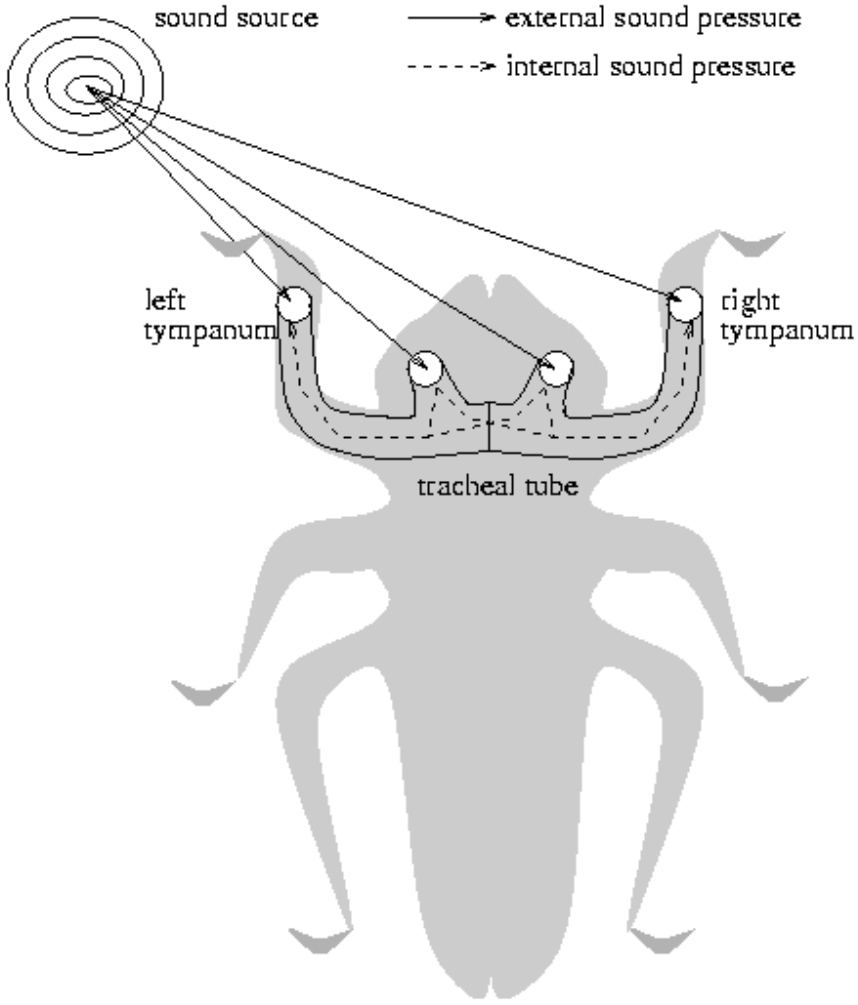


Une approche naturaliste

Mais un examen attentif de la morphologie du cricket donne

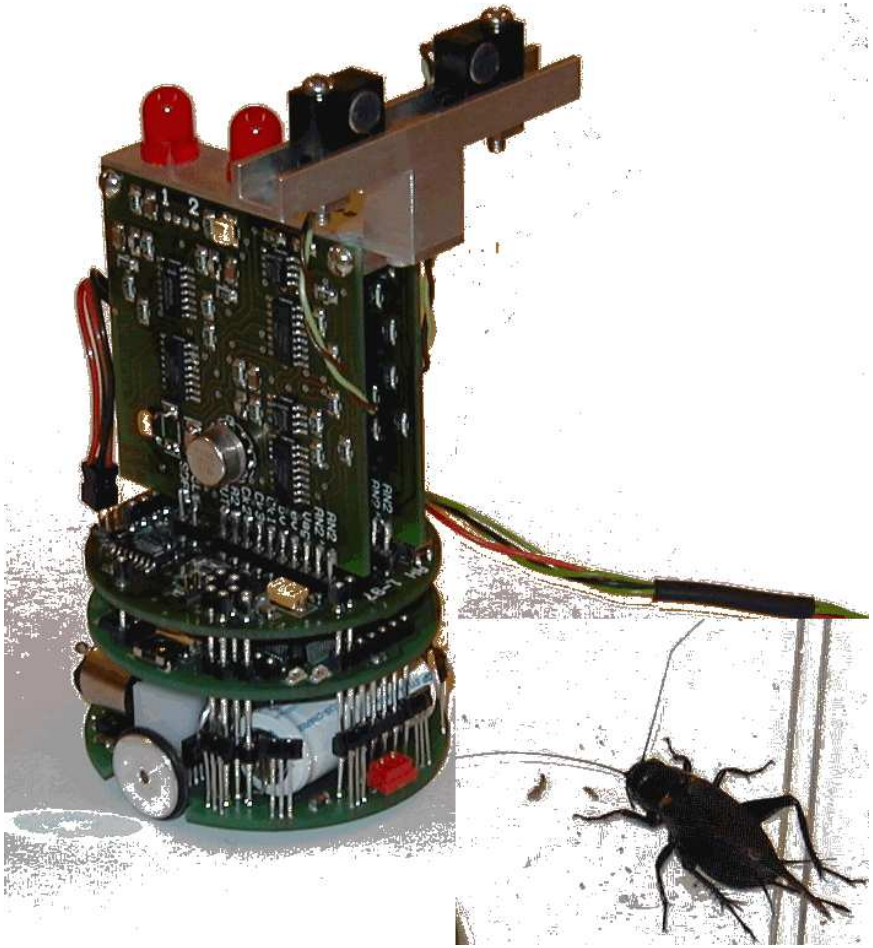
. . .





Résultats

Ce qui permet de faire des robots-cricket sans douleur !



H. Lund et B. Webb

LegoLab – H. Lund

Quelques expériences

Presque toutes tirées de

Evolutionary Robotics – The biology, Intelligence, and Technology of Self-Organizing Machines

S. Nolfi et D. Floreano

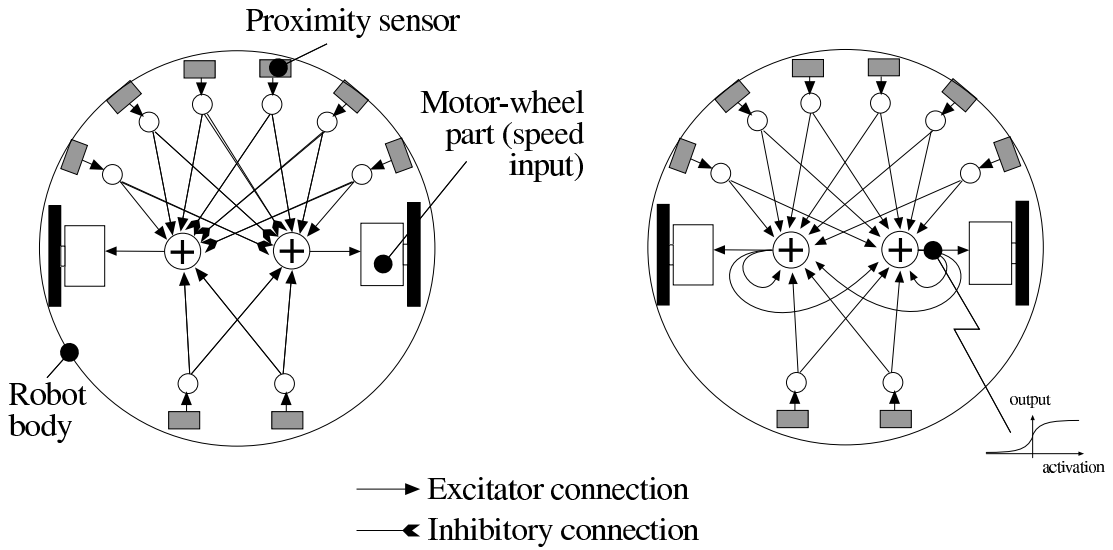
MIT Press, 2000

Nombreuses figures reproduites avec la gracieuse permission des auteurs

Vidéos gracieusement fournies par D. Floreano

Évitement d'obstacles

Floreano et Mondada, 94



Un véhicule de Braitenberg et le RN récurrent qui va évoluer

- Apprentissage des poids d'un RN fixé — Optimisation
  paramétrique
- Récurrent mais sans états internes
- Apprentissage sur robot réel — Plusieurs jours

La fonction performance

$$\mathcal{F} = \int_{T_{exp.}} V(1 - \sqrt{\Delta v})(1 - i)$$

- V somme des vitesses $r_i \in [-0.5, 0.5]$ des roues $\rightarrow$ déplacement
- $\Delta v = |r_1 + r_2|$ $\rightarrow$ ligne droite
- i maximum (normalisé) des capteurs $\rightarrow$ éloignement des obstacles

Comportementale, interne, explicite

Analyse des résultats

- Générations initiales
  - La plupart tournent sur place
  - Les meilleurs avancent lentement
  - Pas d'évitement d'obstacle
  - Perf. dépend du point de départ
- Après $\approx$ 20 gén.
  - Évite les obstacles
  - Ne tournent plus sur place
- Ensuite, accélèrent progressivement

Analyse des résultats (2)

- N’atteignent jamais plus de 48mm/s (max = 80)

  Inertie et mauvais capteurs

- Ne se bloque pas dans les coins étroits

  Comme les contrôleurs de Braitenberg

Autres expériences

- Changement d’environnement
- Changement de plate-forme (robot Koala)
- De la simulation à la réalité

Ré-adaptation très courte → Incrémentalité

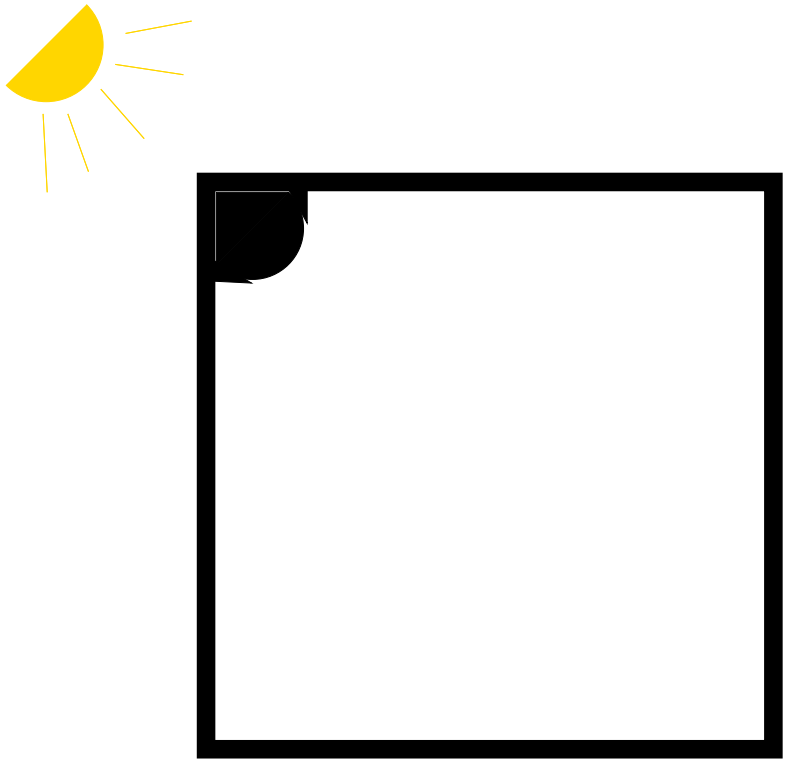
Exploration et recharge

Floreano et Mondada, 96

Au-delà d'un comportement réactif

Conditions

- La batterie se décharge en 20s dans la zone blanche
- et se recharge dans la zone noire
- Mais la zone noire n'augmente pas la performance

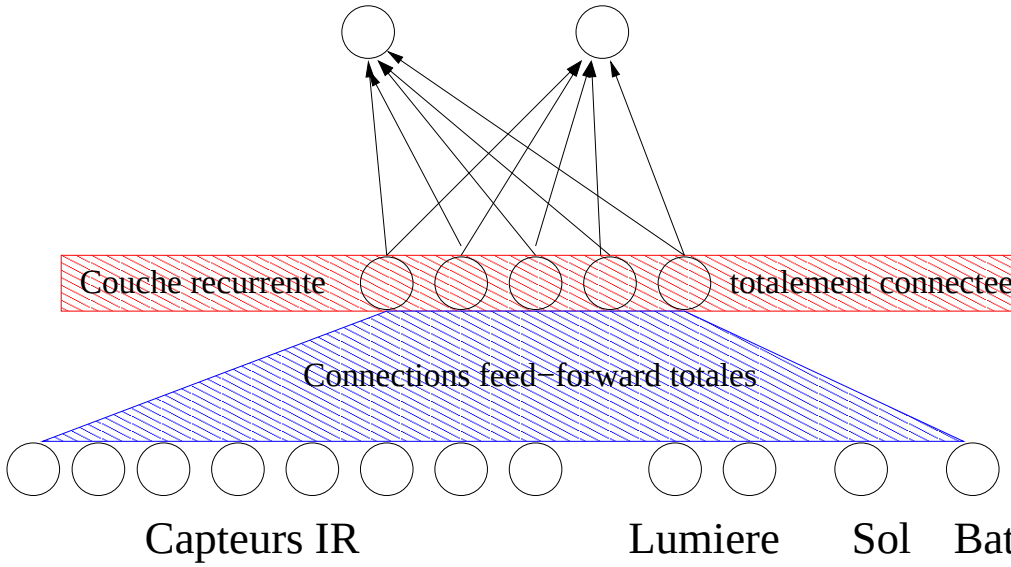


Exploration et recharge (2)

- Un capteur supplémentaire sous le robot → sol noir/blanc
- 2 capteurs sont utilisés en mode passif → lumière ambiante

Apprentissage: réseau de type Elman

- Apprentissage des poids d'un RN fixé
- Récurrent avec états internes
- Apprentissage sur robot réel



Exploration et recharge (3)

La fonction performance

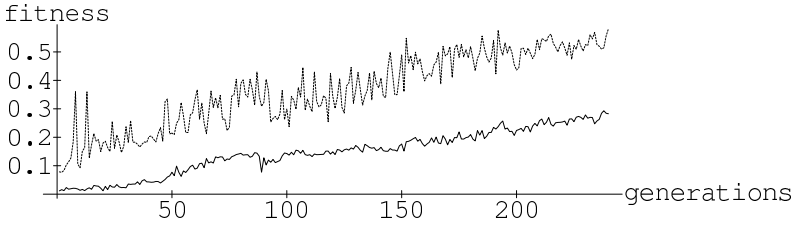
$$\mathcal{F} = \int_{\text{Zone blanche}} V(1-i)$$

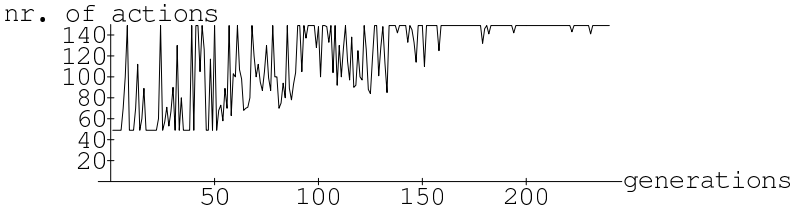
- Le temps de vie dépend de la gestion de la batterie
- décharge simulée – 3 cycles de vie maximum
- V somme des vitesses $r_i \in [-0.5, 0.5]$ des roues $\rightarrow$ déplacement
- i maximum (normalisé) des capteurs $\rightarrow$ éloignement des obstacles

Comportementale, interne, explicite avec un peu d'implicite

Analyse des résultats

Au cours de l'évolution



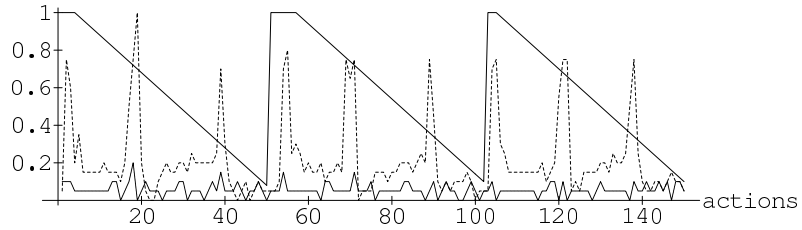


Fitness (meilleure et moyenne) Nombre de pas de moteur du meilleur

Étude du meilleur comportement

Méthodes de neurophysiologie et d'éthologie

On instrumente le robot

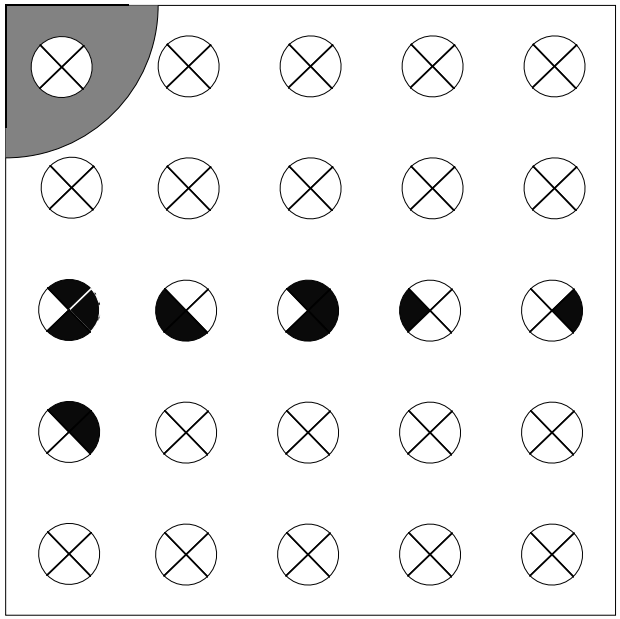


Batterie et état des moteurs au cours de la "vie" du robot

Robustesse et généralisation

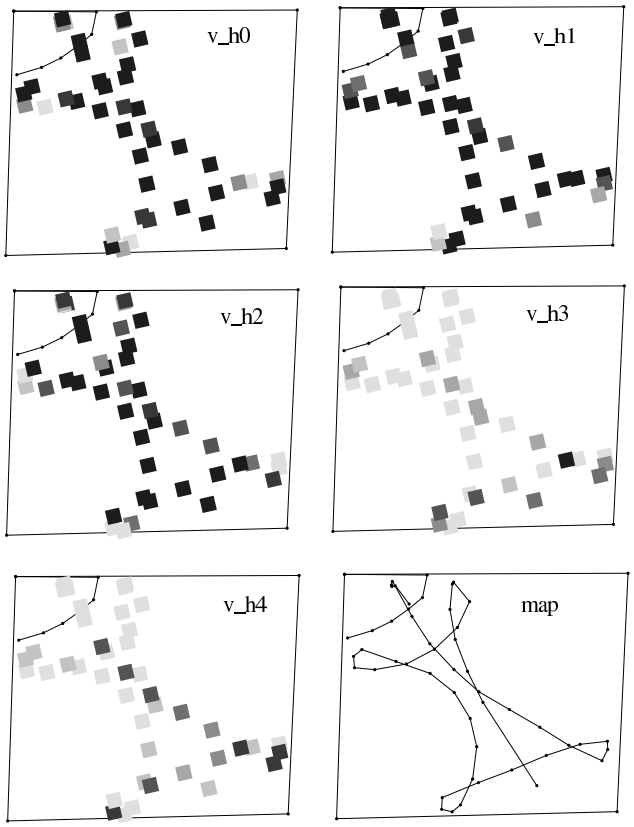
- Robot posé en différents endroits
- avec différentes orientations
- Une fois la recharge trouvée, il la retrouve toujours
- Quelquefois, ne trouve jamais la recharge
- Mais attention à la dynamique !

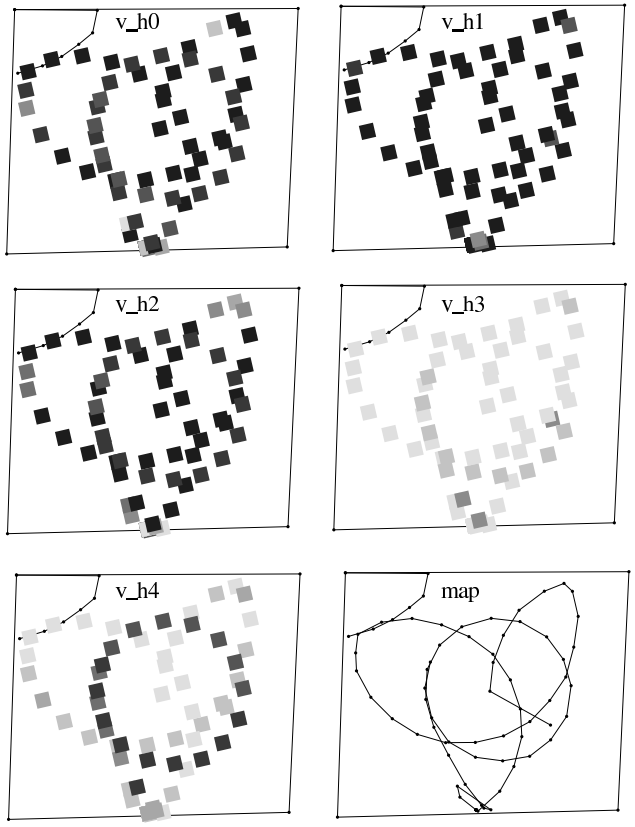
Robot arrêté



Étude des noeuds cachés

Visualisation de l'activation des neurones cachés lors d'une trajectoire

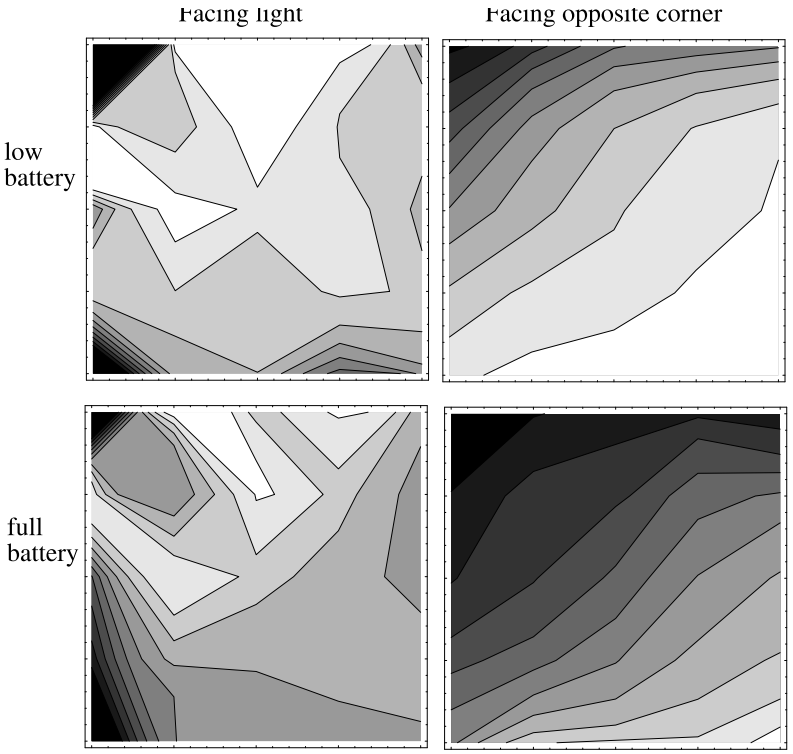




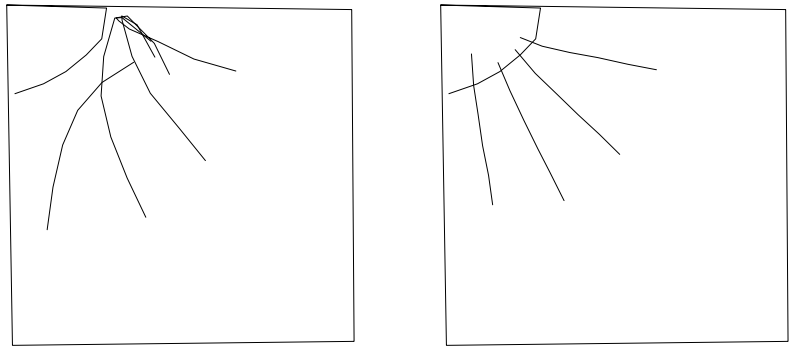
Conditions normales Sans lumière

Peu de modularité évidente . . . sauf pour le noeud H_4

Étude du neurone H_4

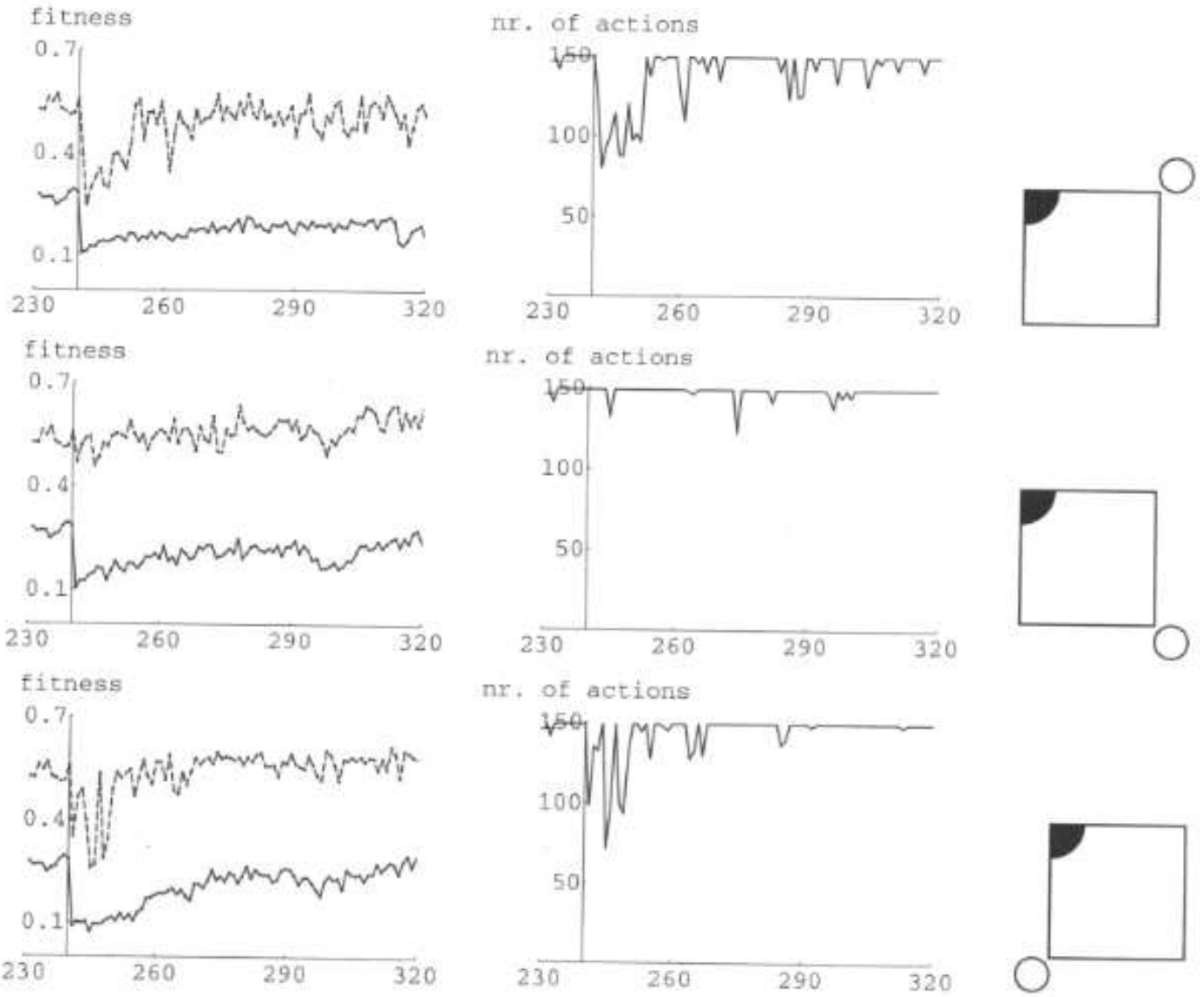


activité de H_4 / direction du robot



trajectoires batterie pleine ou vide

Changement d'environnement



Évolution de robots adaptatifs

Floreano et Mondada, 96

Introduction d'apprentissage en ligne

Pour

- Le robot s'adapte aux changement d'environnement
- L'apprentissage peut guider l'évolution

Effet Baldwin vs Lamarckisme

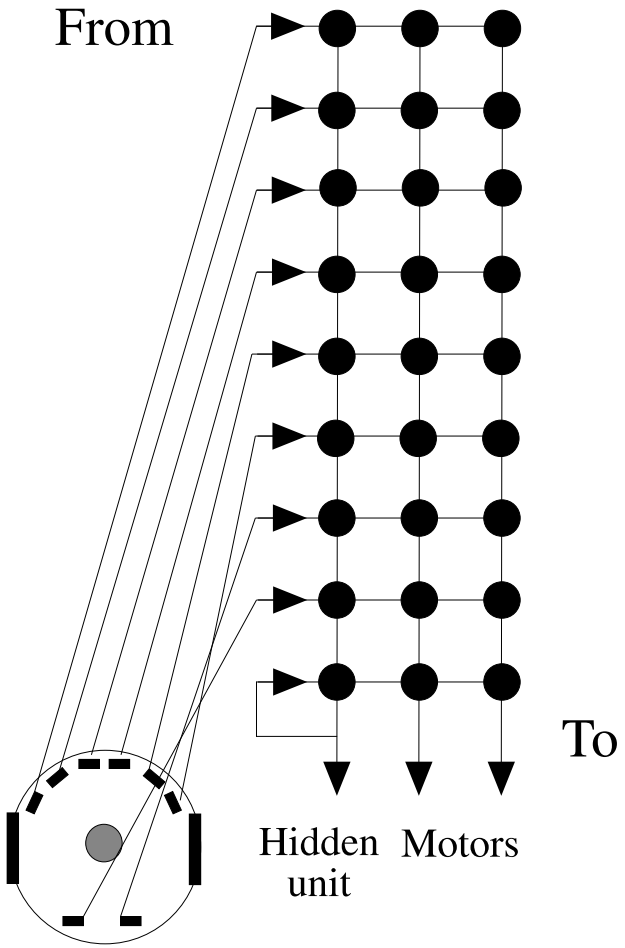
Contre

- Les premiers pas de temps risquent d'être catastrophiques
- Dans un environnement compétitif, retarde la reproduction
- Les résultats sont *a priori* moins robustes
- L'apprentissage peut coûter cher

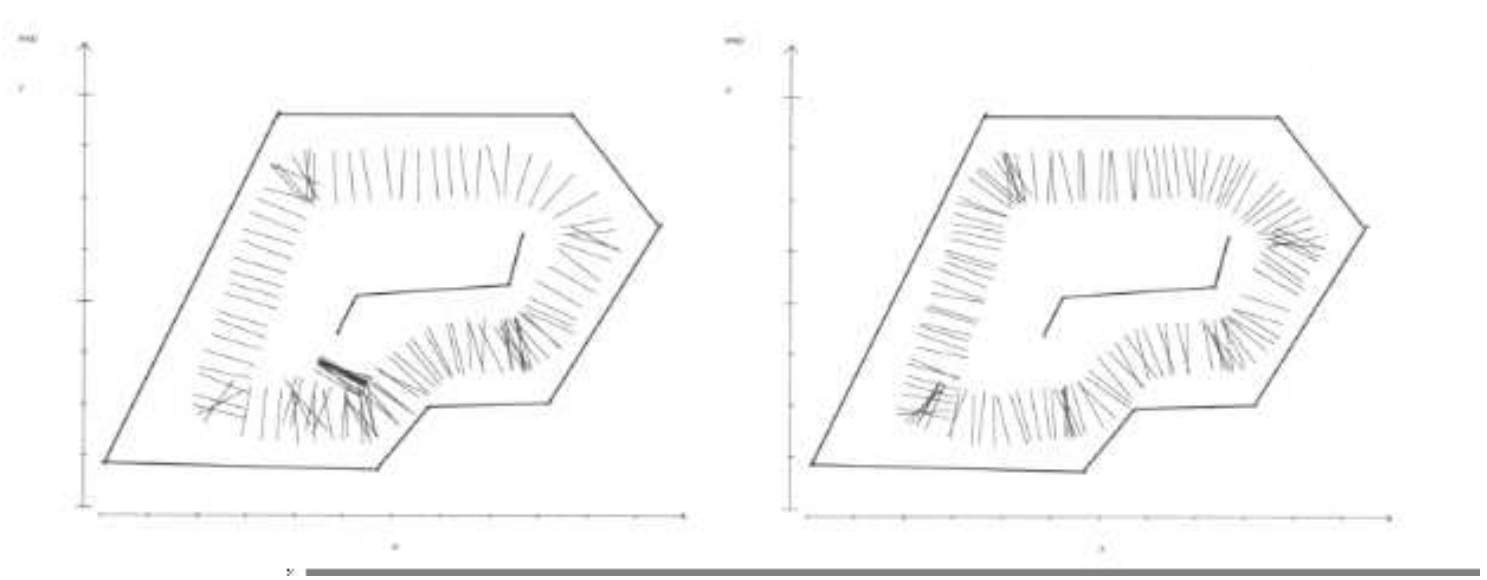
Apprentissage de règles non-supervisées

On code les synapses sur 8 bits

- Entrée ou modulation
- Excitation ou inhibition
- Règle d'apprentissage
  - Règle de Hebb
  - Règle postsynaptique
  - Règle présynaptique
  - Covariance
- η parmi {0, 0.3, 0.7, 1}



Résultats



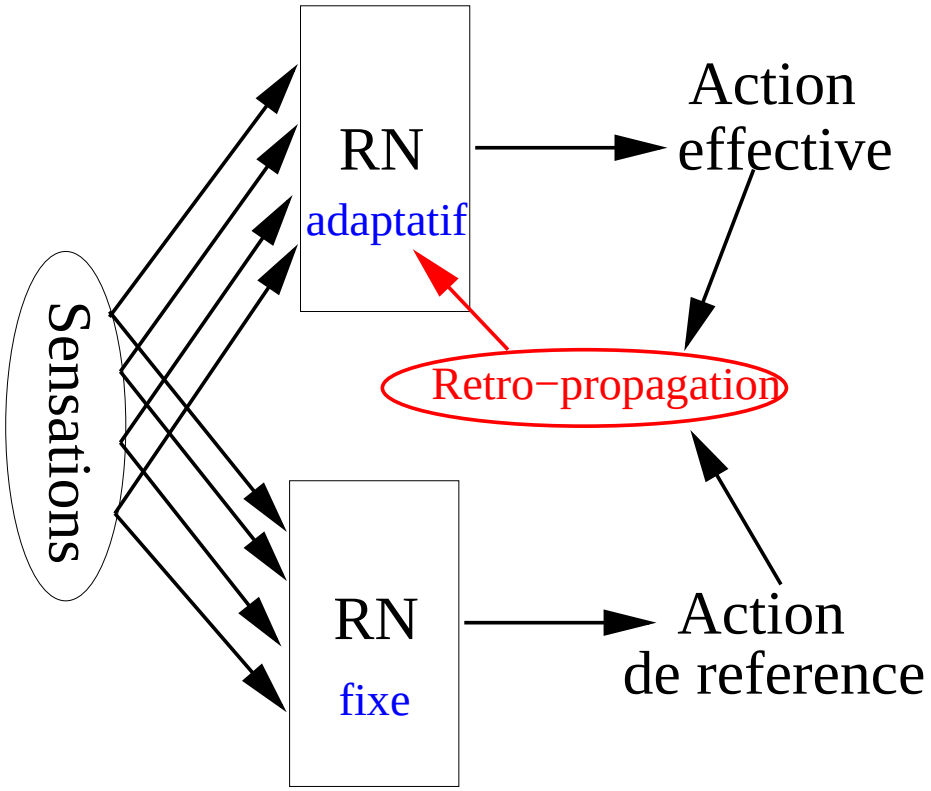
Mais apprentissage lent

- Environnement compétitif
- Environnement changeant rapidement

Apprentissage de prédiction des sorties

On apprend par rétro-propagation . . . à partir des erreurs prédites

- Deux réseaux parallèles
- L'évolution apprend
  - Les poids des 2 réseaux
  - Les paramètres de la RPG
- Poids du réseaux "moteur" ajustés en ligne
- à partir de la déviation / sorties prédites



Mur noirs-murs blancs

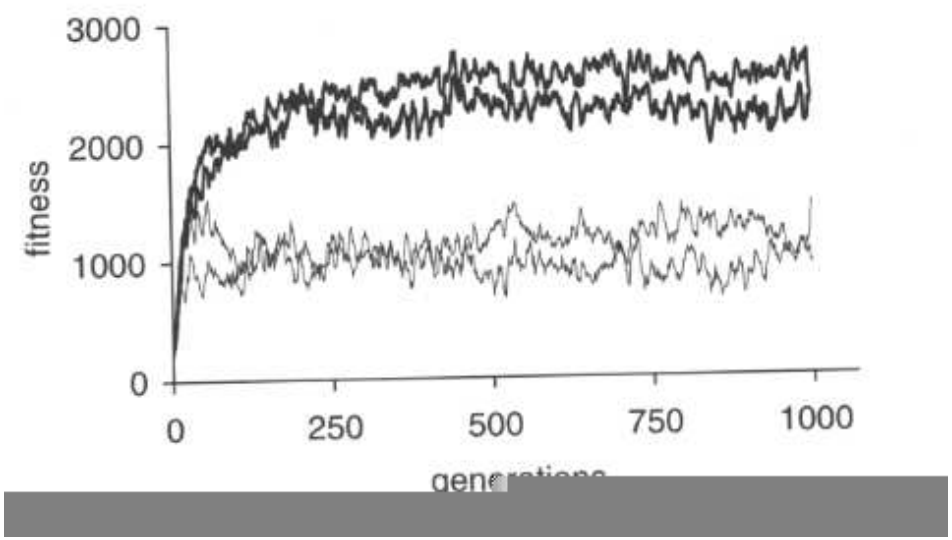
Les réponses des capteurs dépendent de la couleur des murs

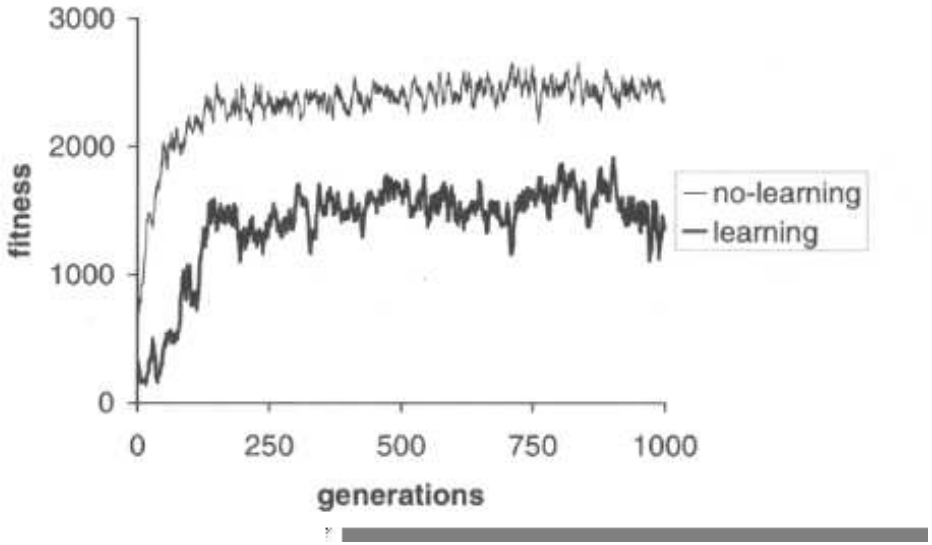
- Tâche : explorer l'arène — trouver une cible "invisible"
- 10 essais — époques
- La couleur des murs change à chaque génération

Seuls les robots capables de modifier leur comportement peuvent survivre à plusieurs changements

Analyse des résultats

- Légèrement meilleur que évolution seule et apprentissage seul . . .
- dans l'environnement d'apprentissage seulement!





En changeant d'environnement

En gelant les poids

Résultats de Nolfi et Parisi → adaptation

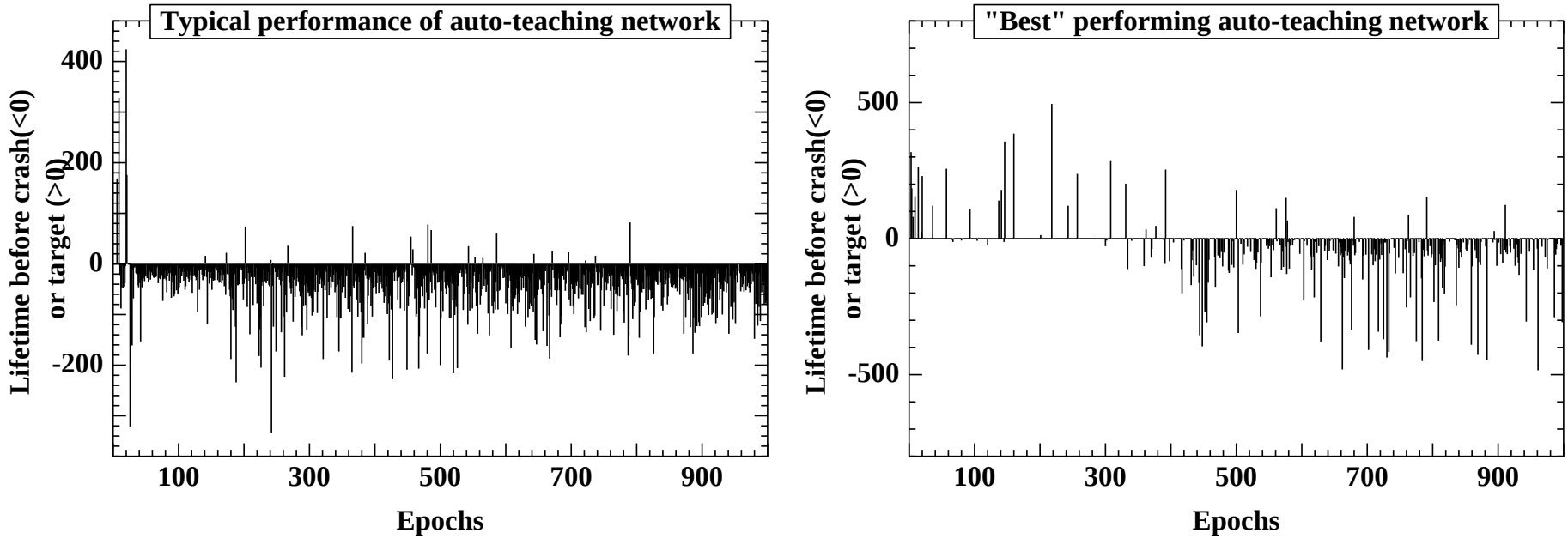
Opportunisme de l'évolution

N. Godzik, 2004

Stabilité à plus long terme

- L'évolution ne teste jamais plus de 10 époques !
- Tests des meilleurs résultats (500 générations)
- mais sur 1000 époques

avec changement de couleur toutes les 10 époques



Résultats pour le pire et le meilleur de 21 réseaux auto-teaching

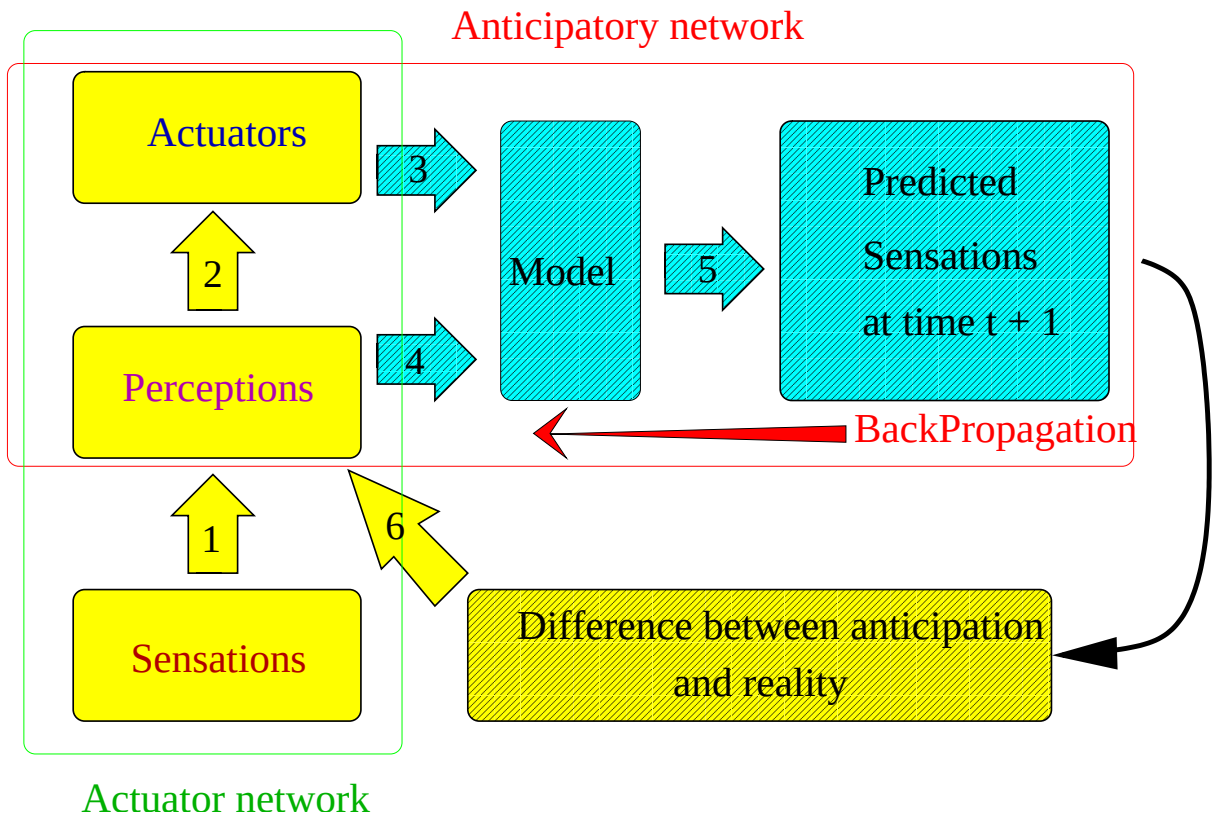
Anticipation

- Force de rappel dans l'environnement

  Prédire le résultat de ses actions

- Contingences sensori-motrices

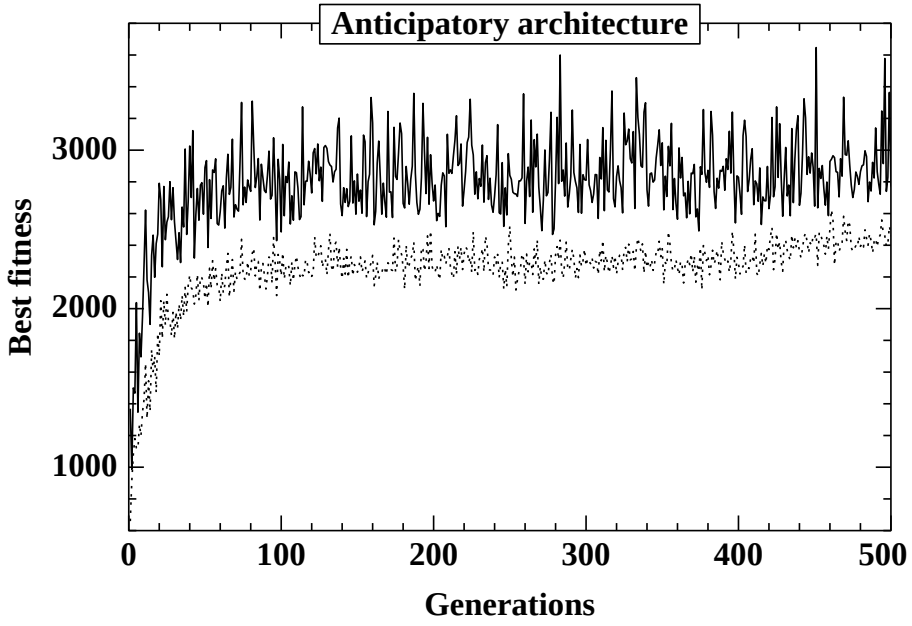
  modèle du monde procédural

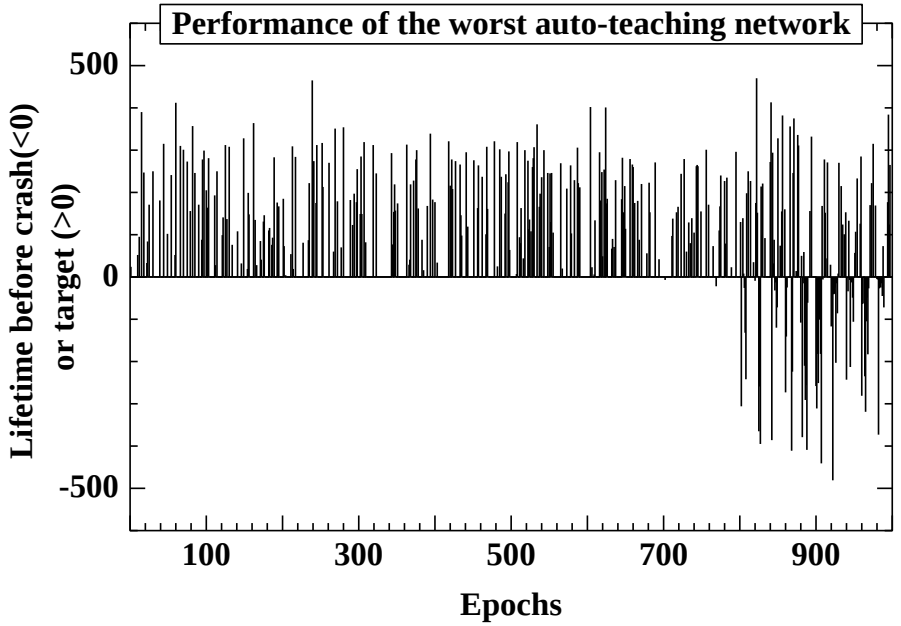


Stabilité de l'anticipation

Mêmes conditions expérimentales

durant l'évolution, puis durant les tests post-évolution





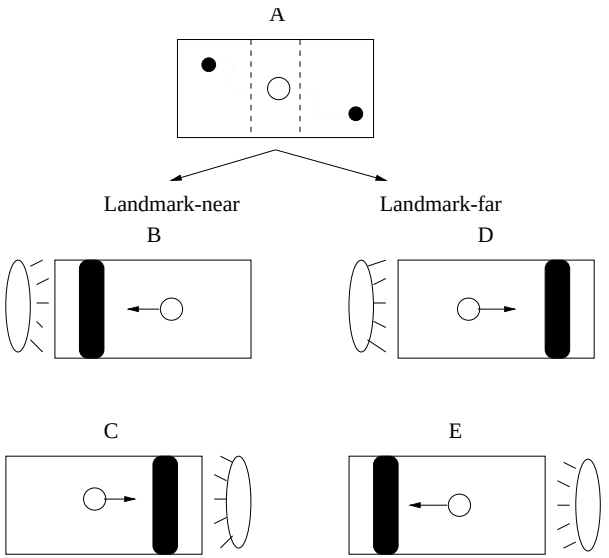
Perfomance on-line (moyenne et maximum sur 21 essais)

Test sur 100 époques du pire des 21 réseaux anticipatoires

Mais adaptation limitée

Apprentissage d'une variable aléatoire

Le robot doit apprendre le lien but/lumière au premier essai de chaque époque, et le mémoriser.



- Un calcul de fitness = 15 époques
- Une époque = – Positionnement aléatoire du but et de la lumière
  – 10 essais à partir d'une position aléatoire
- Score = nbre de fois oú le robot trouve le but ($\in [0, 150]$)

Résultats avec réseaux CTRNNs

CTRNNs : Continuous Time récurrent Neural Networks
Réseau totalement connecté, 11 entrées et 2 sorties

$$\frac{dy_i}{dt} = \frac{1}{\tau}(-y_i + \sum_{j=1}^{k} w_{ji} \frac{1}{1+\exp(y_j|\beta_j)} + gI_i$$

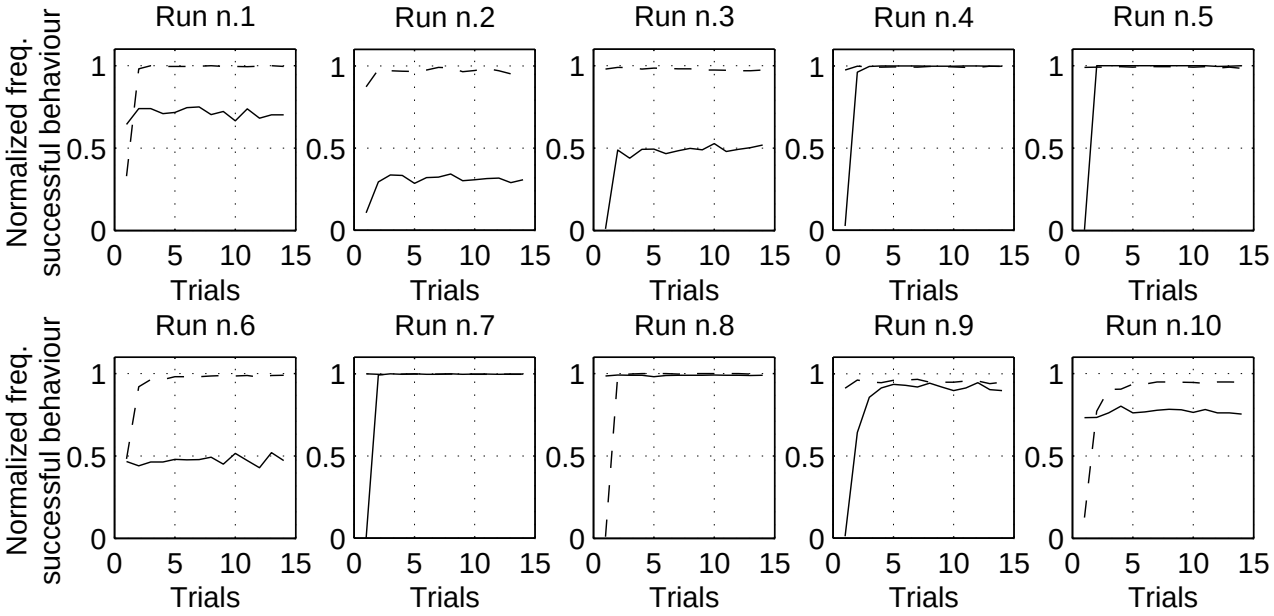
y_i sortie du neurone i, I_i: entrées du réseau
$\tau_i \in [10^{-0,7}, 10^{1,3}$, $w_{ij} \in [-4,4]$, $\beta_i \in [-2,2]$, $g \in [1,7]$ inconnues
Résolution numérique par un schéma de discrétisation standard (Euler)
Les activations du réseau sont mises à 0 au début de chaque époque

pas entre les époques !!!

Résultats

- Génotype: 196 réels, croisement arithmétique 100%, mutation gausienne (σ = 0.1) 10%
- Taille population 100, élitisme 2, sélection par roulette, 5000 générations

Résultats de 10 runs Performance du meilleur sur 5000 tests



Pas de modularité constatée dans les RNs obtenus!

Co-évolution

Compétition (pas coopération)

- Lutte pour la survie
- Modèle proie-prédateur Lotka-Volterra

$$\frac{\partial N_1}{\partial t} = N_1(r_1 - b_1 N_2), \frac{\partial N_2}{\partial t} = N_2(-r_1 + b_2 N_1)$$

- $\rightarrow$ oscillations des tailles de population
- Simulation : tailles de population fixe, performance variable

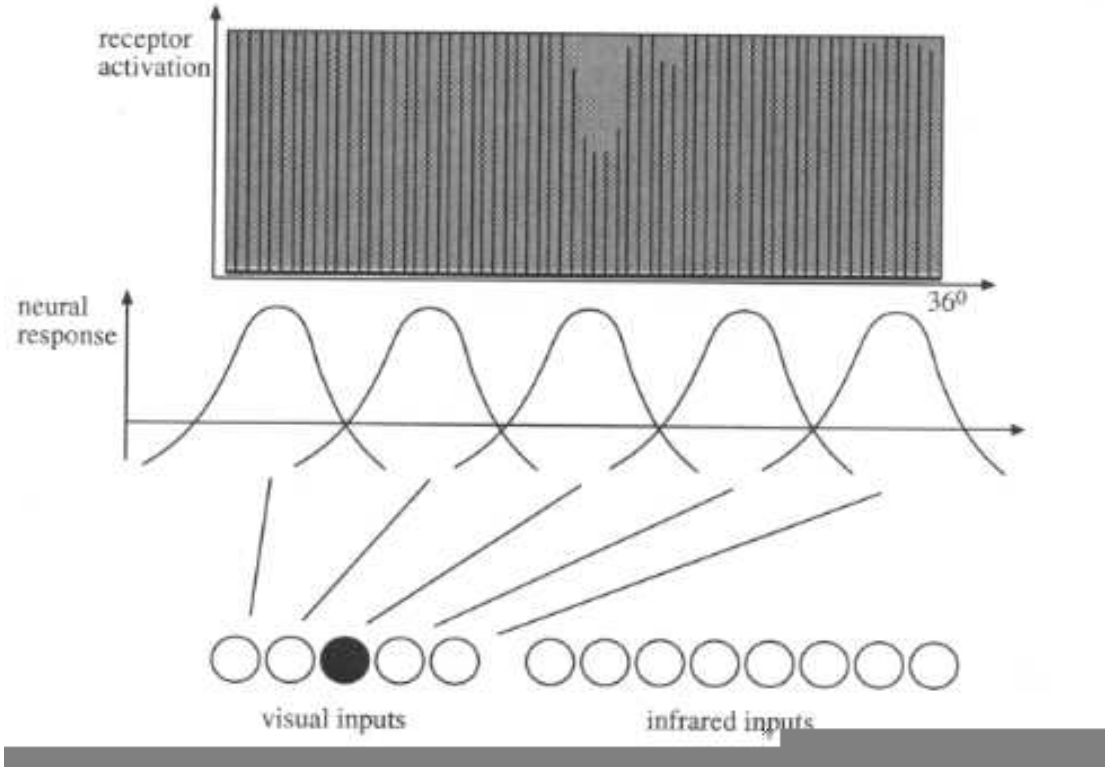
  Équivalent si le taux de reproduction est proportionnel à la performance
- Fitness calculée à partir de tournois

  Entre tous, aléatoires, avec les meilleurs seulement, ...

Proie/prédateur

Floreano et Nolfi, 97-99

- Le prédateur voit, mais est lent — RN 8+5 → 2 récurrentes
- La proie est aveugle, mais deux fois plus rapide — RN 8 → 2 récurrentes



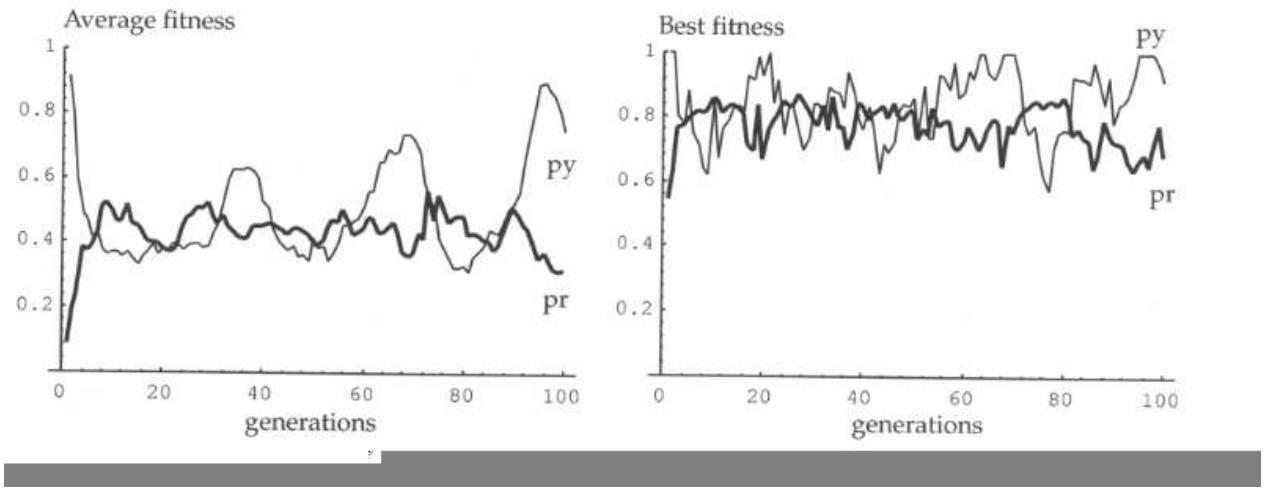
La fonction performance

- Tournoi “round robin” entre tous les prédateurs et toutes les proies
- Arrêt lorsque le prédateur touche la proie (capteur spécial)
- ou après un temps donné (500 cycles, $\approx$ 50s)
- La durée du tournoi est ajoutée à la performance des 2 adversaires

Les proies doivent minimiser la somme des temps de tournoi
Les prédateurs doivent la maximiser

Comportementale, interne/externe, implicite

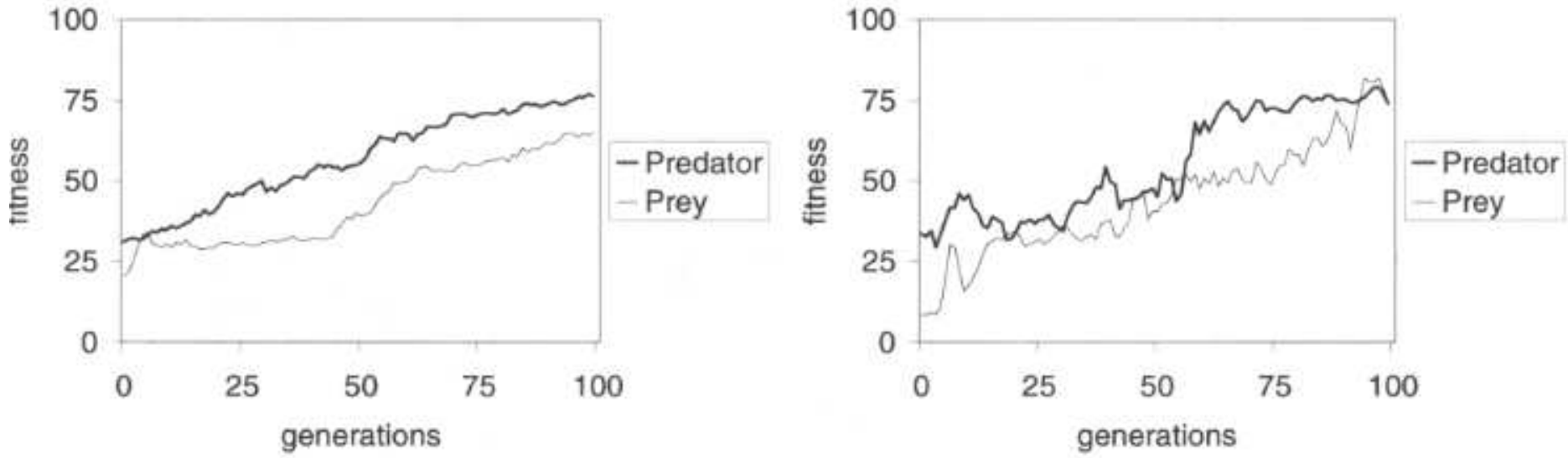
Premiers résultats



- Prédateurs initiaux très mauvais
- Effet “Red Queen” bien visible
- Les meilleurs en fin d’évolution peuvent être (facilement) battus par des meilleurs intermédiaires !

Le tableau d'honneur (hall of fame)

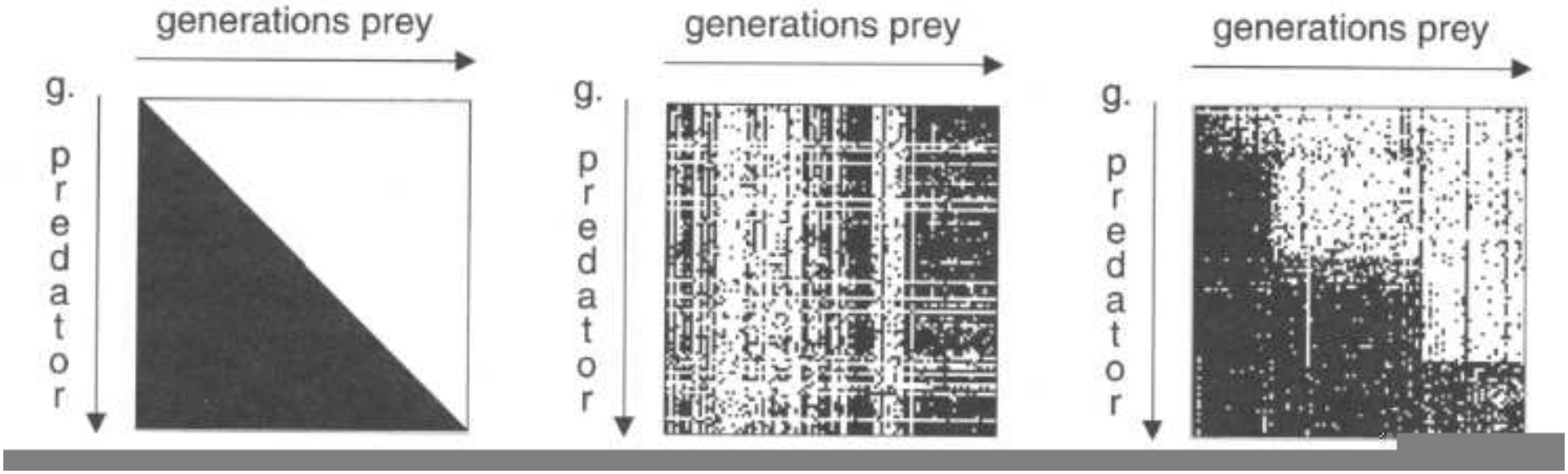
Idée: affronter aussi les champoins des générations passées



Plus d'effet "Red Queen"

Hall of fame (2)

Tournoi entre tous les individus rencontrés à toutes les générations.
Noir ≡ prédateur gagne. Blanc ≡ proie gagne.



Situation idéale / Sans tableau d'honneur / Avec tableau d'honneur

Les meilleurs en fin d'évolution sont meilleurs que presque tous les individus intermédiaires

Contrôleurs symboliques et superviseurs

Trouver un compromis entre

- L'expert dicte le comportement

  e.g. l'architecture de subsumption de Brooks

- L'évolution découvre tout toute seule

  e.g. Floreano & Nolfi, Tuci & al., . . .

Contrôleurs symboliques

- L'expert donne les briques de base

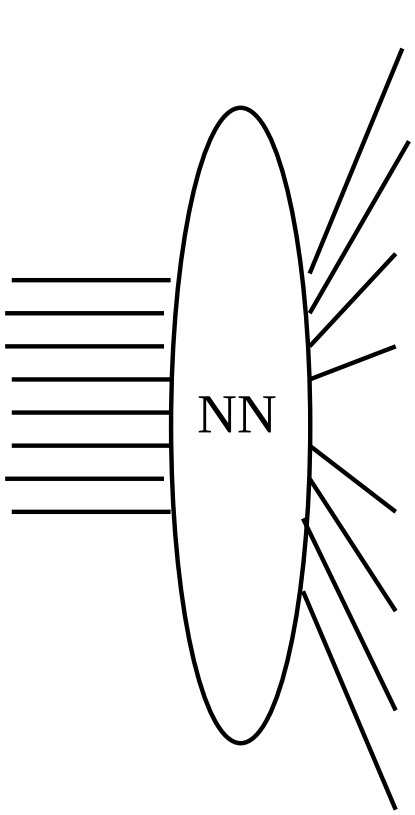
  e.g. aller tout droit, tourner

- L'évolution les assemble

  Sortie du contrôleur = quel brique utiliser

Superviseur

INPUTS ACTIONS MOTORS



A1 FORWARD

A2 BACKWARD

A3 LEFT

A4 RIGHT

I1

I2

I3

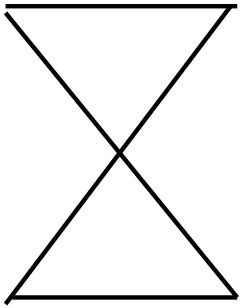
I4

Choose ACTION Ai

= Arg Max {Aj}

Execute with

INTENSITY Ii



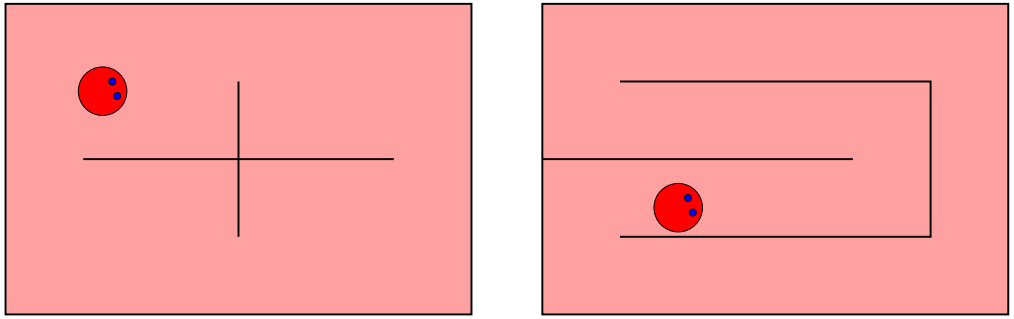
LEFT

RIGHT

LEARNED KNOWN

Évitement d'obstacles

Contrôleurs classiques vs symboliques

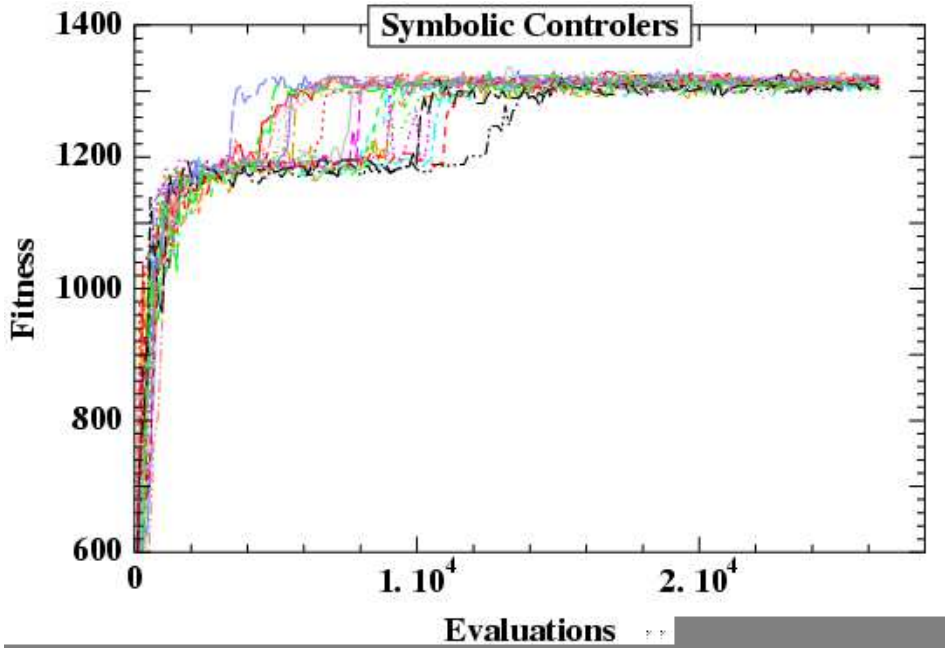


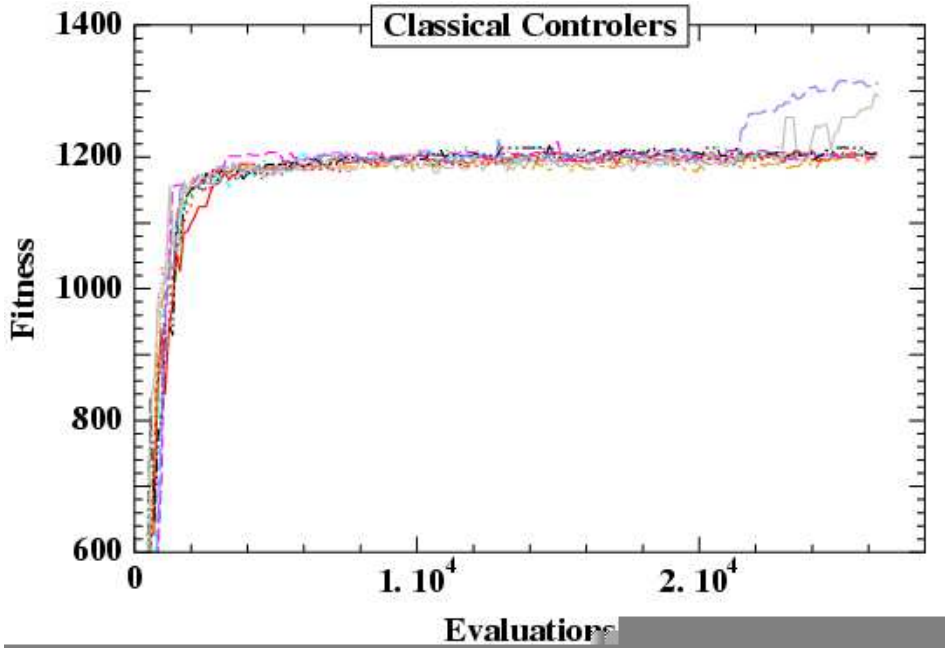
*

Architecture	CC	CS
8-2 / 8-6	861 ± 105	1030 ± 43
8-8-2 / 8-8-6	1042 ± 100	1094 ± 55
8-20-2 / 8-14-6	1220 ± 41	1315 ± 6
8-20-2 / 8-14-6*	1132 ± 55	1197 ± 16

Moyennes sur 10 essais indépendants

Évitement d'obstacles (2)





10 contrôleurs symboliques

10 contrôleurs classiques

Les contrôleurs symboliques trouvent la vitesse max. plus rapidemen

Note: *Punctuated Equilibrium ?*

Les superviseurs

Passage à l'échelle

Donner á l'évolution une bibliothèque de briques de base

Bilbiothèque =

{ Évitement d'obstacles, Balayage, Stop, Suivi de lumière }

Robustesse: Ajouter des briques non pertinentes, voire dangereuses

Architecture des superviseurs :

RN Elman (une couche cachée totalement connectée)

Nb sorties = |bibliothèque|

À chaque pas de temps, exécuter i, $i = Argmax\{Sortie(j)\}$.

Conditions expérimentales

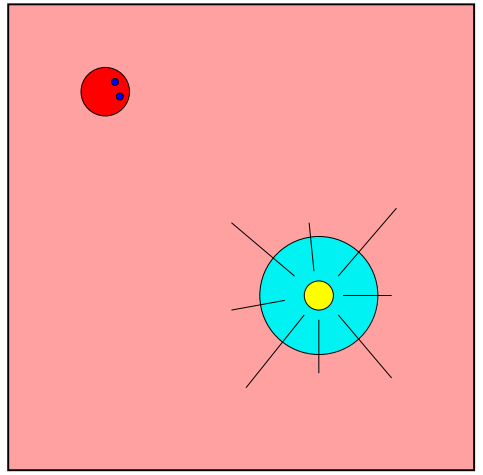
Réseau Elman, avec pour sorties

1. Les actuateurs — contrôleur classique
2. ToutDroit, Droite, Gauche — contrôleur symbolique
3. Bibliothèque de contrôleur classiques — superviseur classique
4. Bibliothèque de contrôleur symboliques — superviseur symbolique

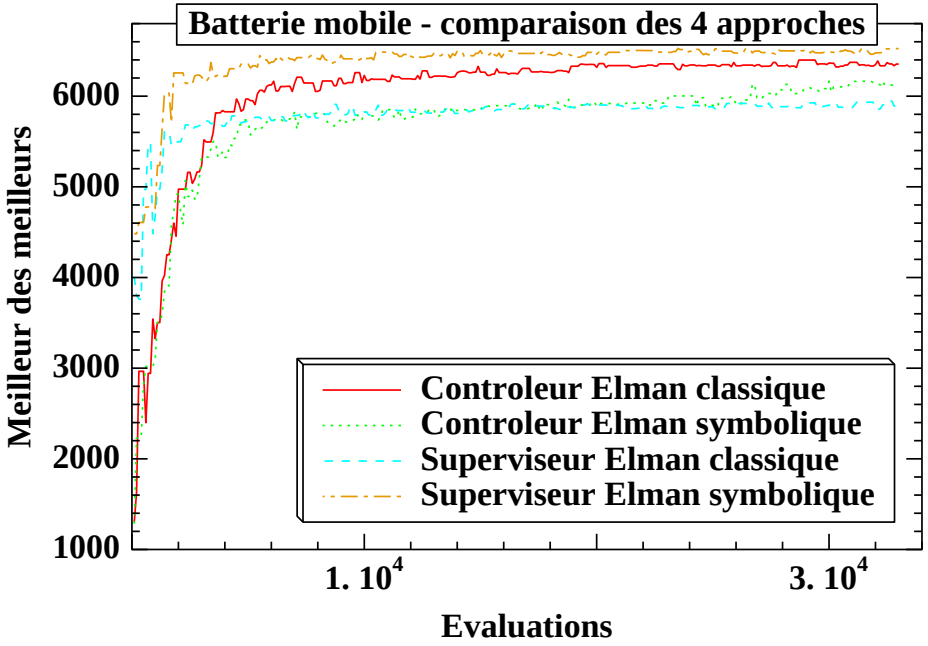
Exploration et survie

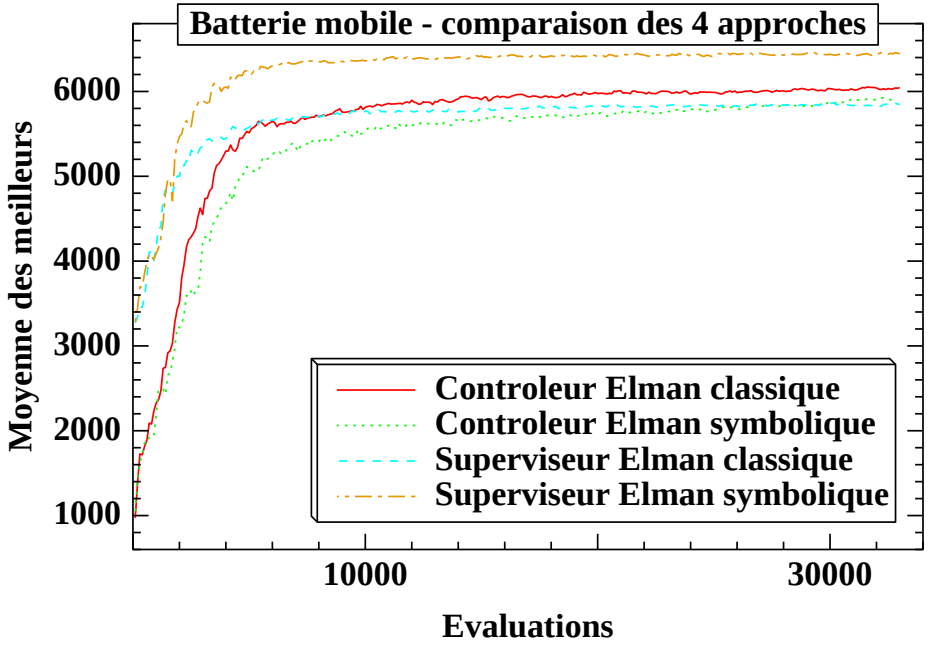
Inspiré par Floreano et Nolfi

- Le robot a une batterie (virtuelle)
  décharge en 685 pas de temps
- Sous le lumière, la zone de recharge
  mais pas de capteur de sol
- La recharge n'est pas instantanée
  100 pas de temps
- 17 entrées
  8 capteurs IR actifs, 8 passifs, et le niveau de batterie
- Fitness $\propto$ vitesse
  en dehors de la zone de recharge



Exploration et survie, Résultats





Meilleur des meilleurs

Moyenne des meilleurs

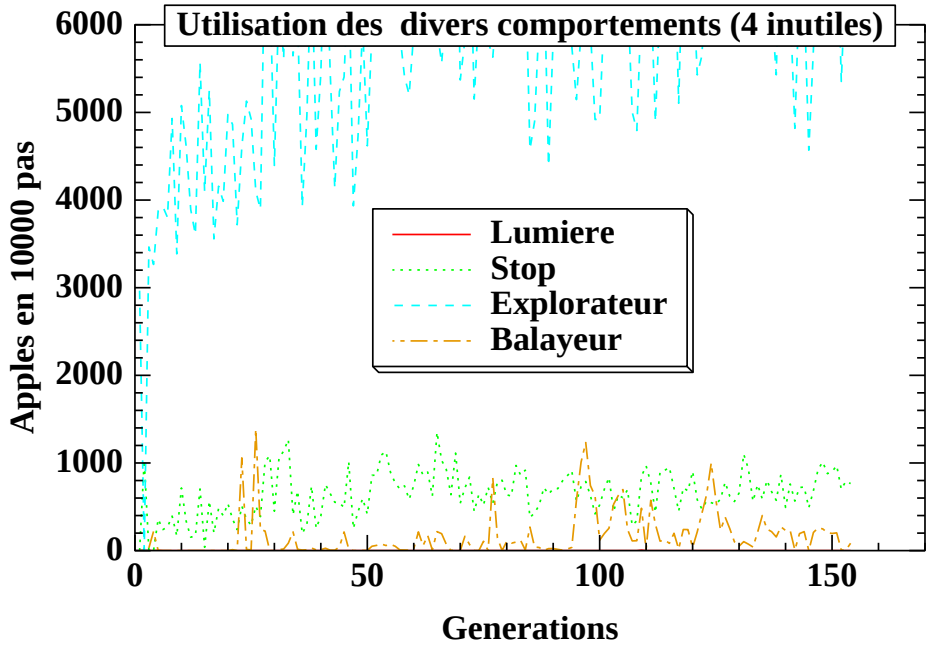
Statistiques sur 10 runs indépendants

Exploration et survie, Résultats (2)

Robustesse

La vitesse est importante

- Balayeur jamais utilisé
- Suivi de lumière rarement utilisé



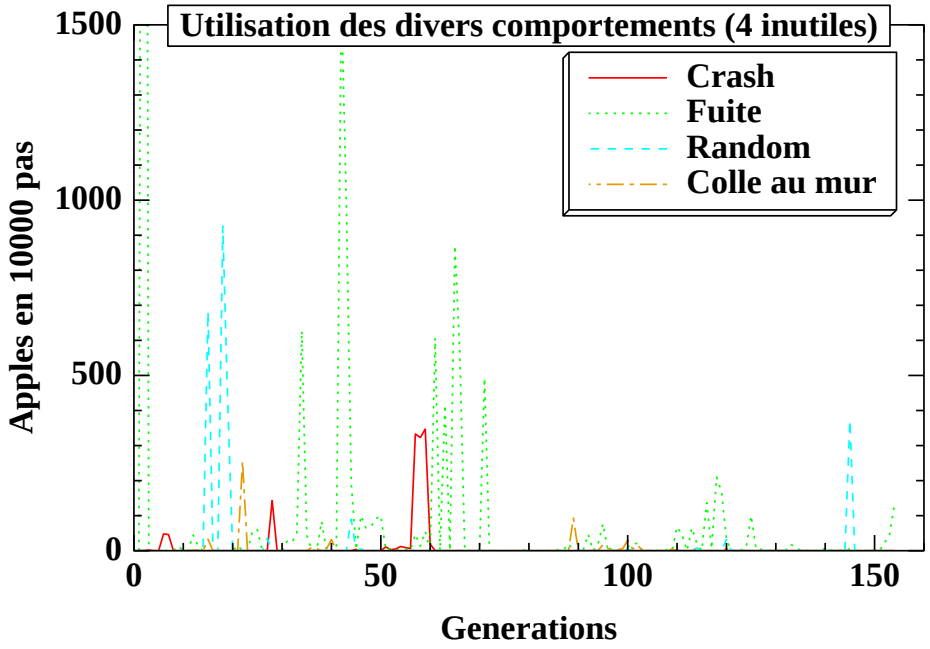
Questions ouvertes

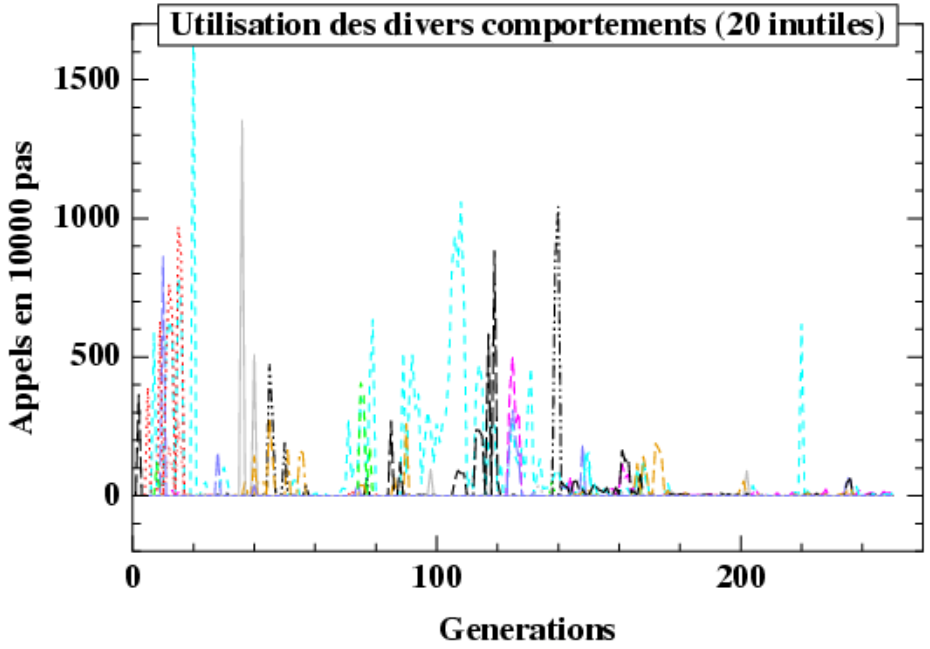
- Faiblesse des performances du contrôleur symbolique
- Quoi qu'il continue à s'améliorer si on continue l'évolution

Neutralité?

Sensibilité à la bibliothèque

Les briques inutiles sont filtrées – ralentit (un peu) l'évolution





4 comportements inutiles

20 comportements inutiles

Statistiques d'utilisation moyenne des comportements par le meilleur contrôleur de la population au cours des générations

Généralisation

- Vitesse de charge/décharge:
  Avantage net au superviseur symbolique
- Augmentation de la taille de l'arène
  - Léger avantage au . . . contrôleur classique Elmann
    Mais il ne fait qu'entrer et sortie de la zone de recharge
  - La superviseur symbolique n'a besoin que de quelques générations de plus
- Ajout de murs: Pas de différence Tout le monde échoue :-(

Contrôleurs symboliques : Conclusion

Meilleur choix : Superviseur symbolique

- exploite les comportements de base fournis
- très petite variance des résultats

Contrôleur classique: quelquefois aussi bon, grande variance

Contrôleur symbolique: évolution lente (pourquoi ?)

Superviseur classique: stagne rapidement

Premiers résultats : montrent

- le passage à l'échelle
- les capacités de généralisation
- la robustesse: les comportements inutiles ne sont pas utilisés

Embryogénèse

F. Gruau, 95

Arbre $\rightarrow$ graphe de connexion

Embryogenèse : un “embryon” se développe selon un arbre de règles.

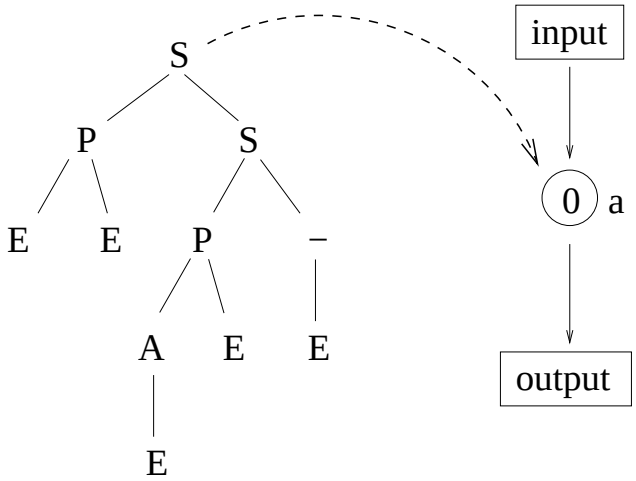
Sur un exemple, dans le contexte des réseaux de neurones booléens:

Les primitives GP

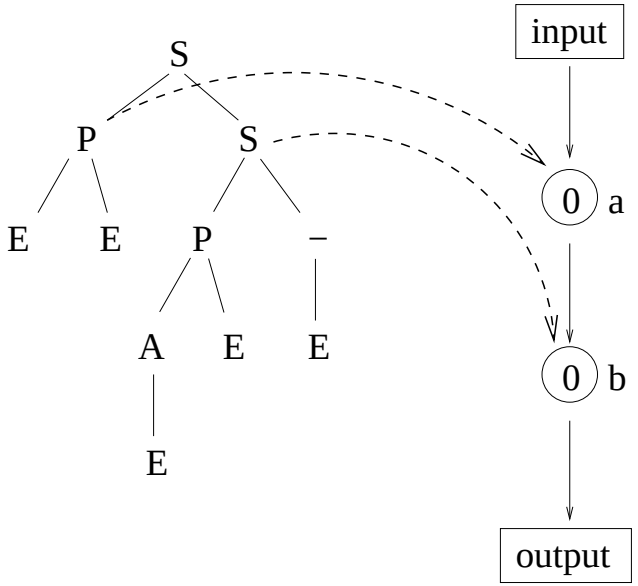
$\mathcal{T}$	E	symbole de fin
$\mathcal{N}$	S	Division séquentielle
	P	Division parallèle
	A	Incrémenter le seuil du neurone
	O	Décrémenter le seuil du neurone
	$+$	Incrémenter le poids d'un lien
	$-$	Décrémenter le poids d'un lien
	C	Couper le lien courant
	I	Incrémenter le numéro du lien courant
	D	Décrémenter le numéro du lien courant
	R	Retour au sommet de l'arbre
	W	Wait

Remarque : un seul terminal $\Rightarrow$ besoin de mutation !

Développement d’un réseau de neurones

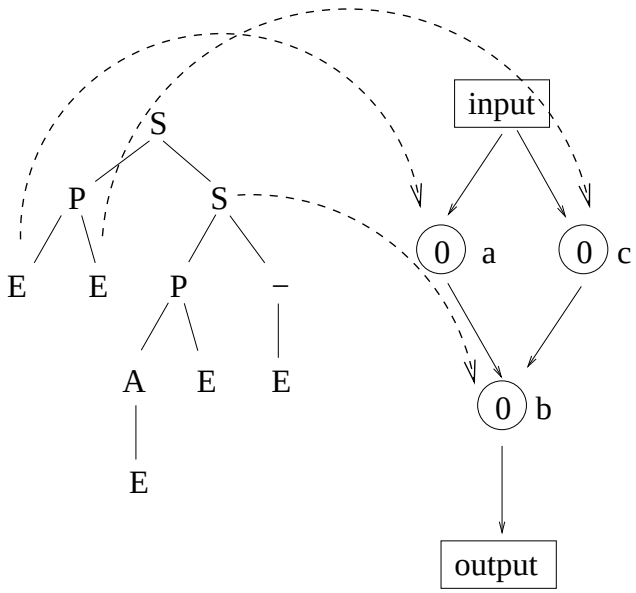


L’arbre et l’embryon

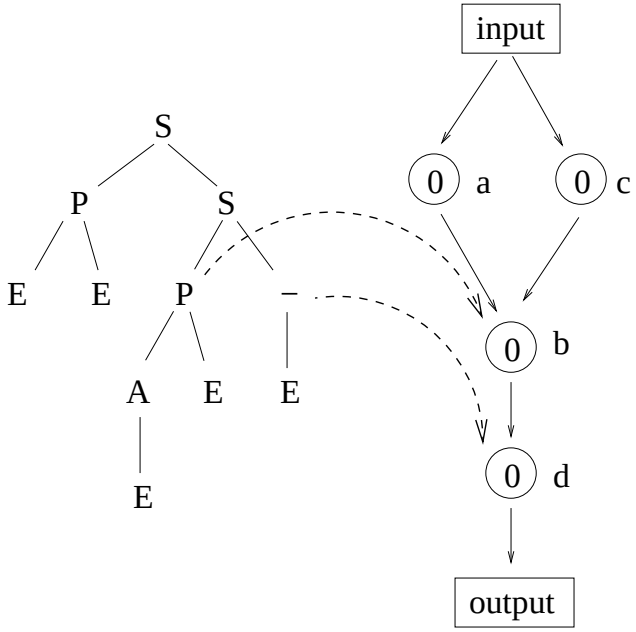


Étape 1

Développement d'un réseau de neurones (2)

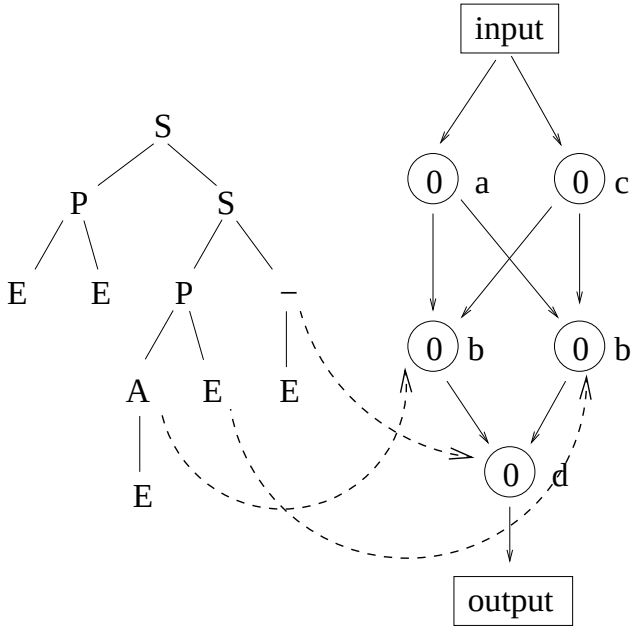


Étape 2

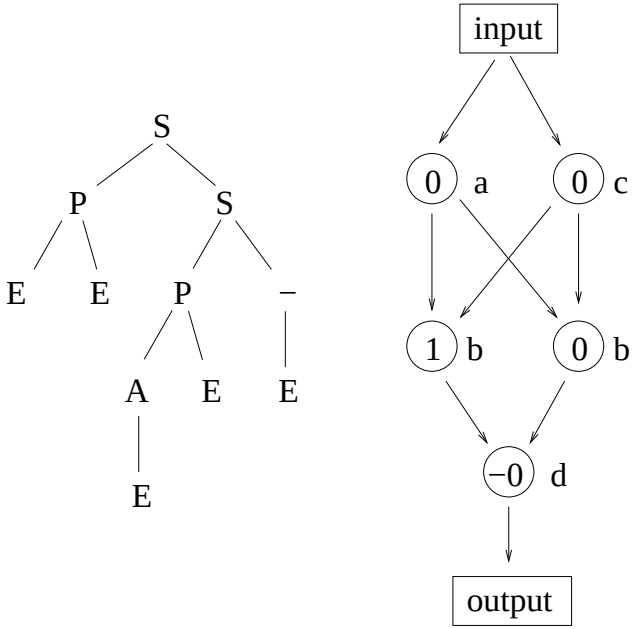


Étape 3, 4 et 5

Développement d'un réseau de neurones(3)



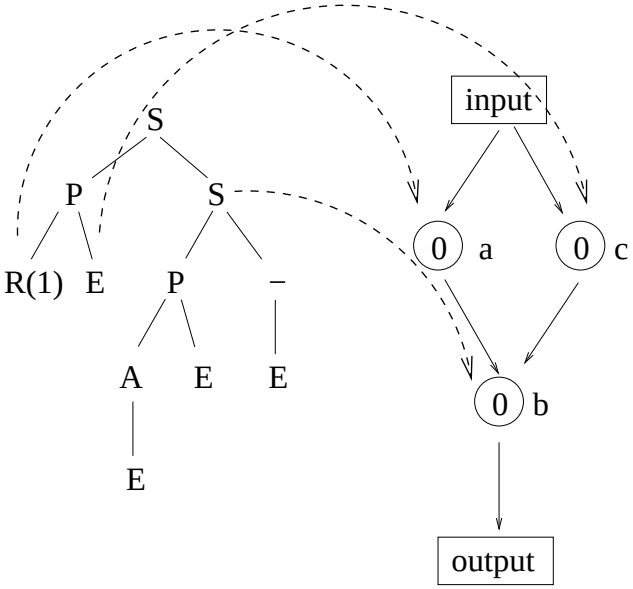
Étape 6



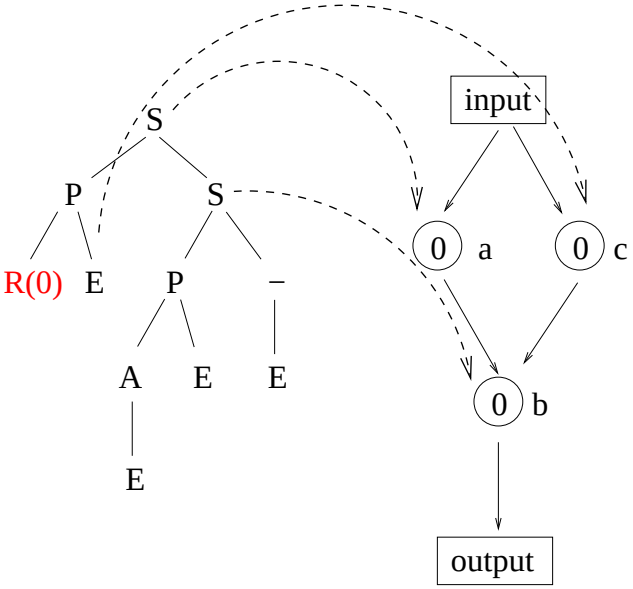
État final (réseau XOR)

Développement d'un réseau de neurones (4)

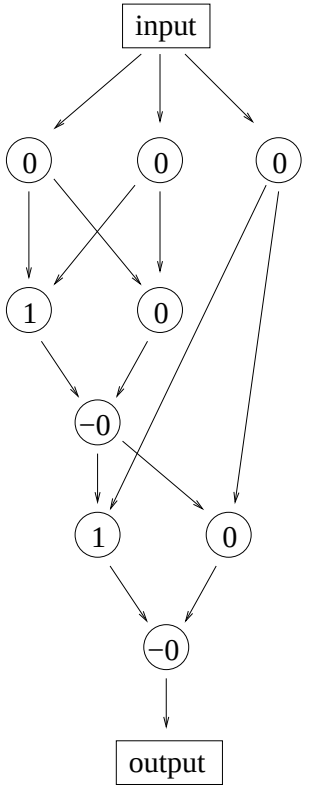
Avec récursion:



Étape 2



Étape 3



État final

Synthèse de circuits analogiques

Koza et al. 95, 98

Embryon :

une entrée, une sortie

2 cellules initiales : connection source | sortie

connection terre | sortie

Acquis :

Simulateur de circuits analogiques ROBUSTE

SPICE (217 000 lignes, Berkeley)

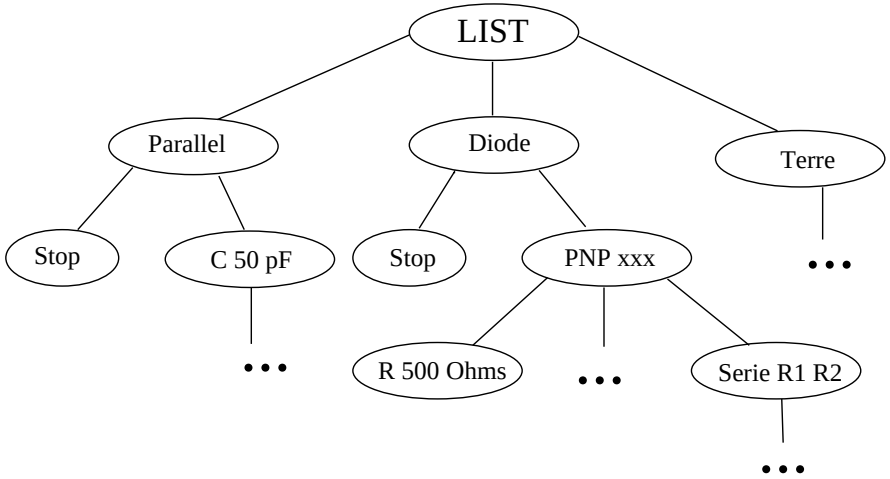
Réalisations :

Filtres passe-bande asymétriques

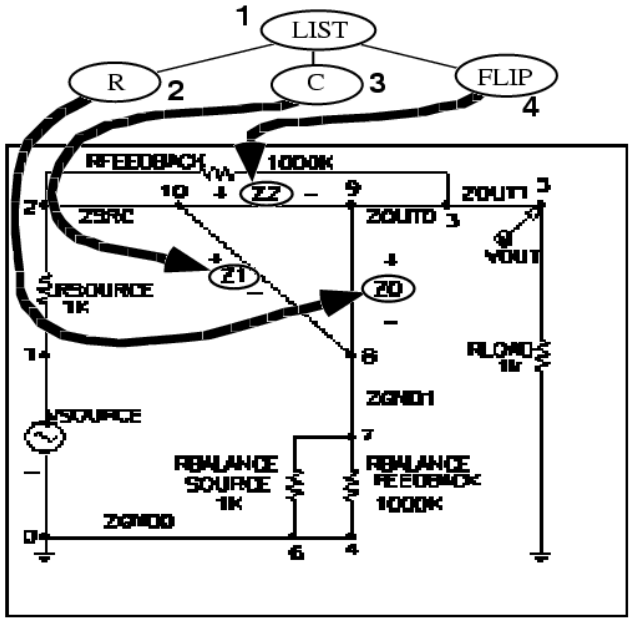
Extracteur racine cubique

Amplifieurs,...

Synthèse de circuits analogiques (2)

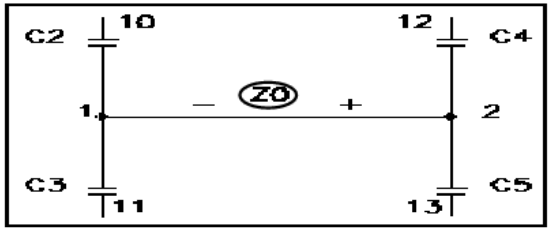


Un exemple de programme



et d'embryon

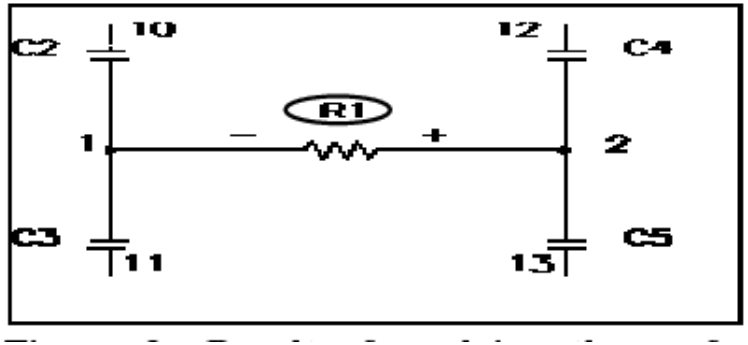
Exemples de noeuds et terminaux



Fil générique

Développement :

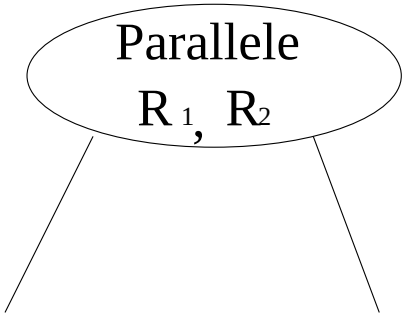
Resistance
R_1 Ω

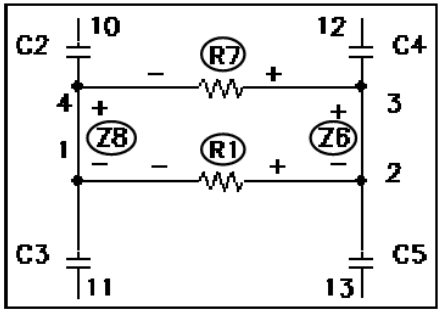


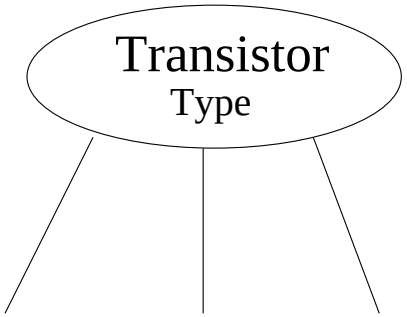
Un terminal

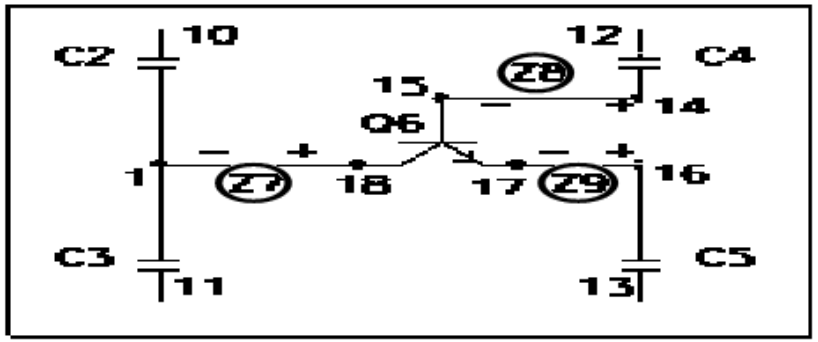
Application au fil générique

Exemples de noeuds (2)







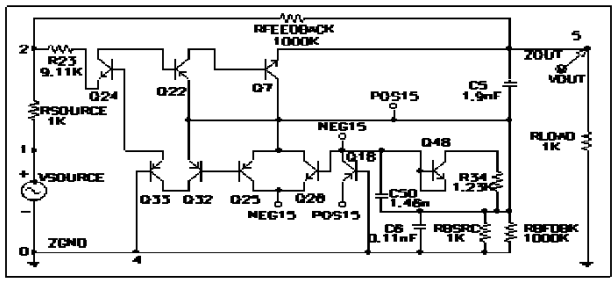


Noeuds

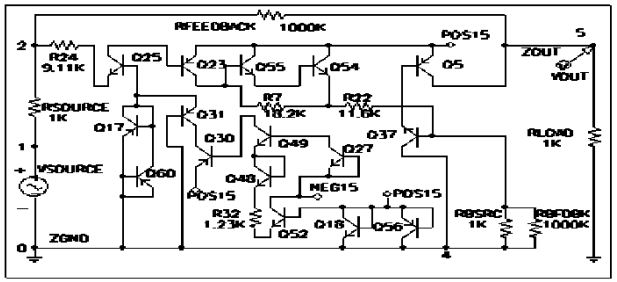
Application au fil générique

+ mémoire, récursion et subroutines

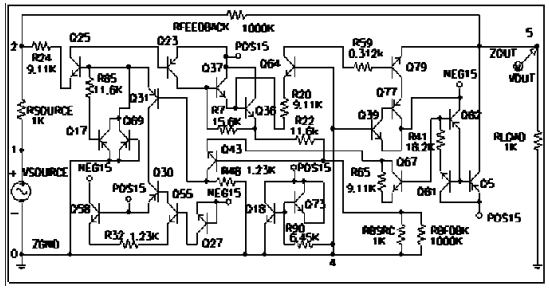
Un résultat: Ampli opérationnel 60dB



Meilleur initial



Génération 49



Génération 109

Mais taille population = . . . 640 000

Apprentissage de la topologie d'un contrôleur n

- F. Gruau (95) – Robot à six pattes – marche, simulation seulement
  Décomposition (par l'algorithme) en 3 sous-réseaux identiques

  1 million évaluations . . .
- J. Kodjabachian et J.-A. Meyer (98) – marche + obstacles + but

  Suite des travaux de Gruau
- N. Jakibi (98) – marche + évitement d'obstacles
  Simulation minimale pour un robot octopode

  3500 générations, passage au robot réel
- T. Gomi et K. Ide (98) – sur robot réel ("homard" de 3.5kg)

  40s par robot, 100 générations de 50 robots

Évolution de morphologie

Nombreux travaux . . . préliminaires
R. Dawkins (89), K. Simms (94), J. Pollack (99 . . .), O. Chocron (99)

- La morphologie du robot est un facteur essentiel
- Pourquoi ne pas la faire évoluer aussi ?
- Deux approches
  - Simulation seulement – tout est permis
  - Réalisme – il faut générer des robots constructibles

Les créatures de Karl Simms

- Évolution de la morphologie et du “cerveau” neuronal
- suivant une grammaire
- représentée à l’aide de graphes orientés
- Simulation lourde

  Plusieurs jours de Connection Machine – 65000 proc.
- Évolution de locomotion (marcher, nager, sauter)
- et co-évolution compétitive (attraper un objet le premier)

Les créatures de Karl Simms (2)