Algorithmes évolutionnaires pour l'optimisation numérique : Identification de fonctions et Programmation Génétique

Marc Schoenauer*

Équipe TAO – INRIA Futurs – France

http://www.lri.fr/∼marc

Juin 2005

* Au CMAP, École Polytechnique (UMR CNRS 7641) avant sept. 2001

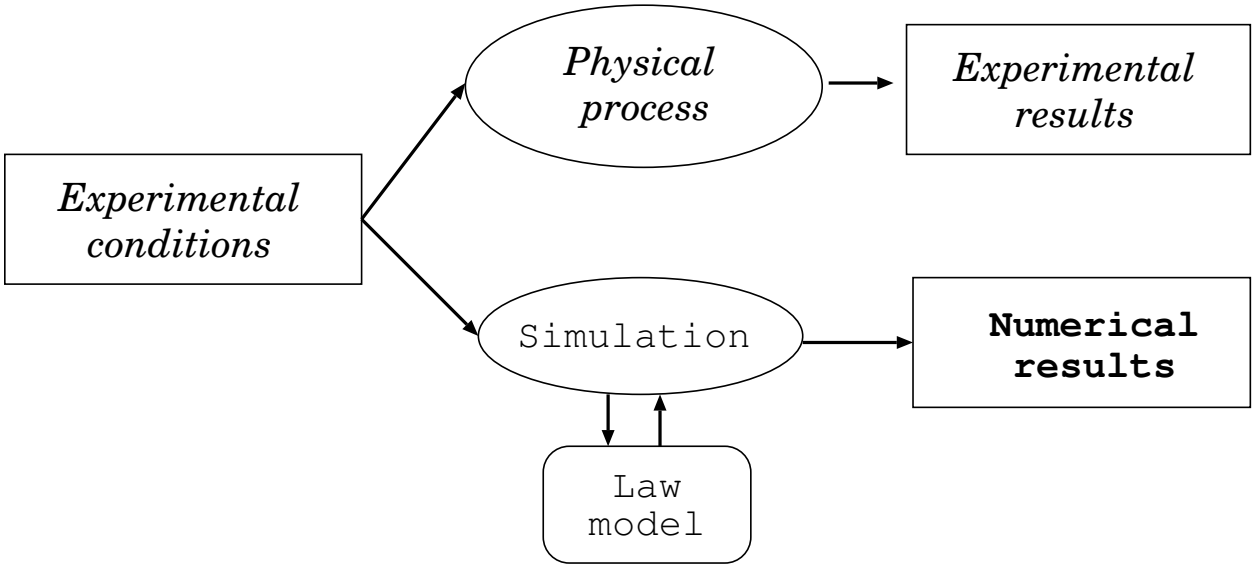
Plan

- Identification de fonction
- Planification de trajectoire — Réseaux de neurones directs
- Chromatographie — Réseaux de neurones récurrents
- Lois de comportement de matériaux hyperélastiques 3D — GP classique
- Lois de comportement de matériaux élasto-visco-platiques 1D — GP ad hoc
- Embryogénèse

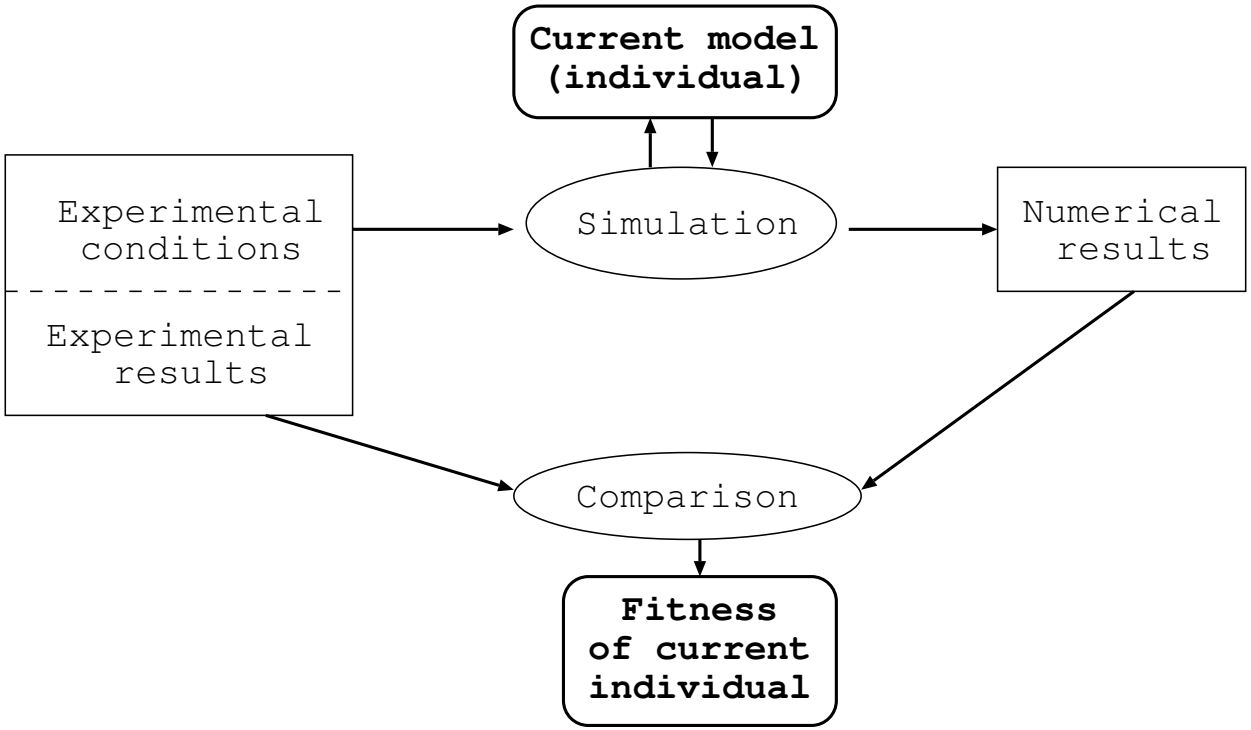
Problèmes inverses

Mal posés, numériquement difficile à résoudre

Hypothèses: Résolution **précise** et **robuste** du problème direct



Une fitness possible pour le problème inverse:



Question: quelle représentation (et opérateurs de variation) pour les fonctions?

Exemples

- Contrôle optimal : Le problème du créneau
  Par réseaux de neurones multi-couches
- Identification de loi d'état en chromatographie
  Par réseaux de neurones récurrents
- Identification de lois de comportement de matériaux
  Programmation génétique "classique" pour les matériaux hyperélastiques
  Programmation génétique originale pour les modèles rhéologiques
- Identification d'inclusion
  par diagramme de Voronoï

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.

Régression (data fitting)

On dispose d'exemples de valeurs prises par la fonction inconnue:

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

on cherche p.ex. F de $\mathbb{R}^N$ dans $\mathbb{R}$ qui minimise

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

- Il existe de nombreuses techniques d'approximation et d'interpolation

  Interpolation polynomiale, fractions rationelles

  Fonctions Splines

  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 ?

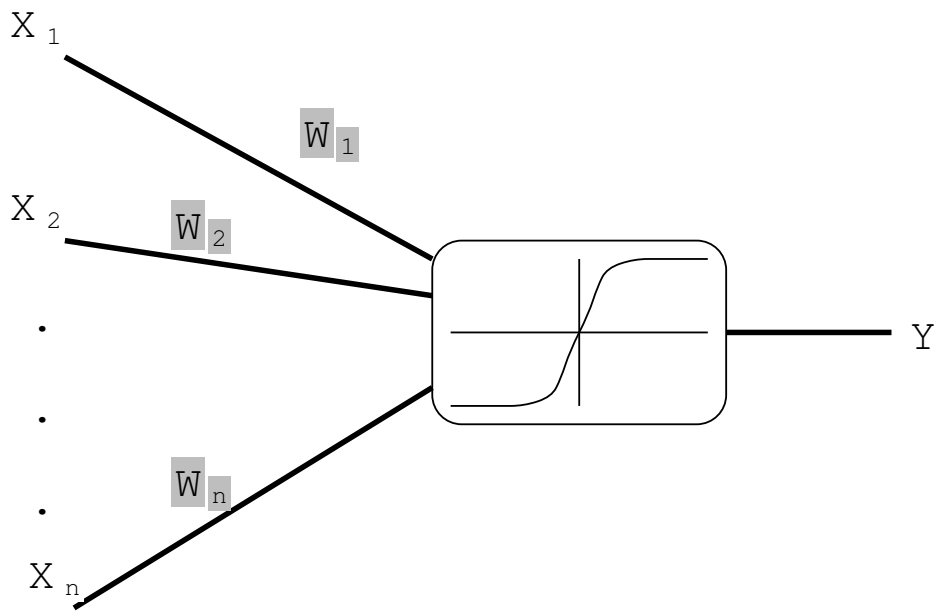
Réseaux de neurones

Neurones artificiels

- Perceptron (Rosenblatt 59, Widrow & Hoff 60, Minsky & Papert 69)

- Parallel Distributed Proessing (Rumelhart et McClelland 86)

Tutoriel en ligne à `http://www.eeaax.polytechnique.fr/Neurones/`



- Connections : des entrées et une sortie
- Poids sur les connections entrantes Mémoire locale
- Une fonction de transfert σ
- 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 $\mathbb{R}^n$ dans $\mathbb{R}^m$

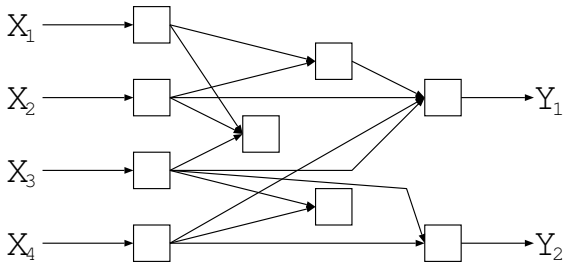
  réseaux dits à propagation directe – **feed-forward**

- Dans le cas contraire, Réseaux **récurrents**

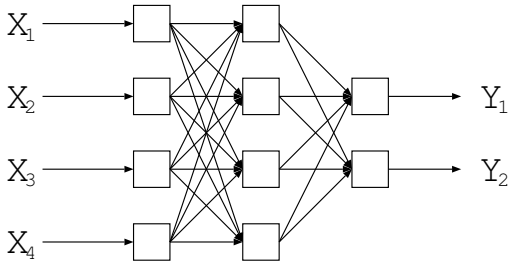
  on calcule les sorties de chaque neurone

  – De manière synchrone, ou

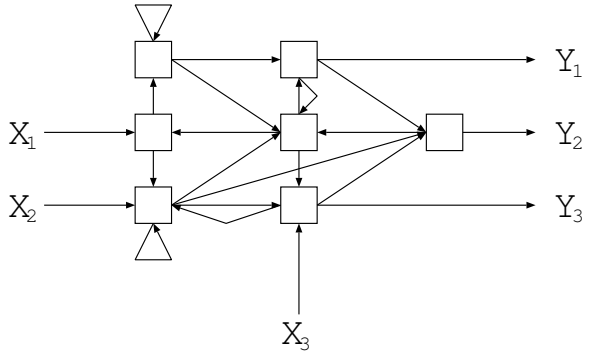
  – Dans un ordre donné



"Feed-forward" network



Muti-layer perceptron



Recurrent Network

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

Évolution de réseaux de neurones

- Évolution des poids d'un réseau à architecture fixée

  Éventuellement, point de départ pour le RPG

- Évolution de la règle d'apprentissage

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

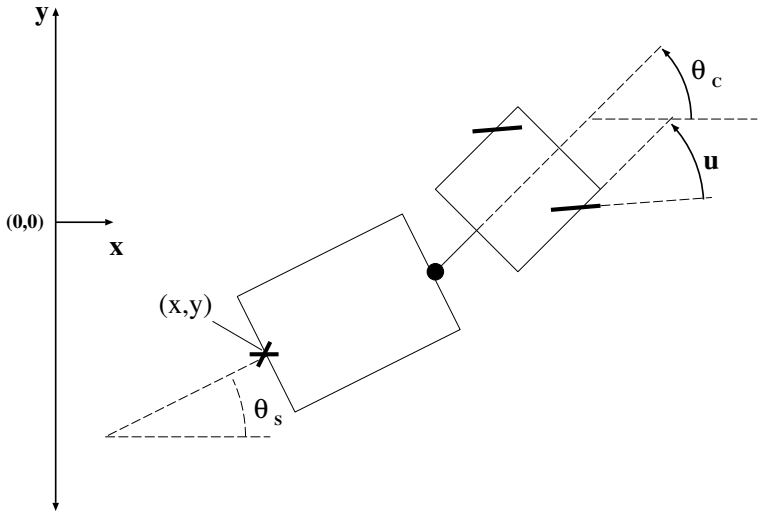
  x_j et y_i sont les activités des neurones amont et aval de la connection

- Évolution de la topologie (et des poids)

Planification de trajectoire

E. Ronald et MS, 94

Variables d'état : $(x, y, \theta_c, \theta_r)$, commande : angle u des roues.



La dynamique du système est connue (discrètisée)

Problème: trouver la commande $u(x, y, \theta_c, \theta_r)$ pour garer le véhicule au point $(0, 0, 0)$ depuis toute position de départ.

Perceptron multi-couches

- Réseaux totalement connectés d'architecture fixe.
- Régle empirique: réseau 4–9–1
- 55 poids à ajuster

Pas d'exemple de comportement du réseau
Les algorithmes d'apprentissage usuels (e.g. la rétro-propagation) ne s'appliquent pas

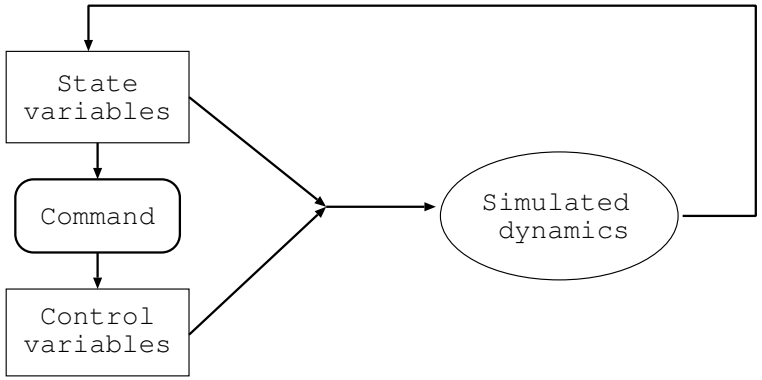
$\longrightarrow$ **Évolution artificielle**

L'algorithme d'évolution

- Représentation : vecteur de 55 réels
- Taille population ≈ 100
- Initialisation $U[-1, 1]$
- Sélection par roulette ou tournoi de taille 2
- Remplacement générationnel
- Croisement arithmétique, proba ≈ 0.8
- Mutation Gaussienne, écart-type décroissant, proba ≈ 0.1
- Quelques centaines de générations

Performance

Pour une fonction $u(x, y, \theta_c, \theta_r)$ (e.g. un perceptron multi-couches) et un point de départ donnés, calcul en boucle ouverte



Distance au but:

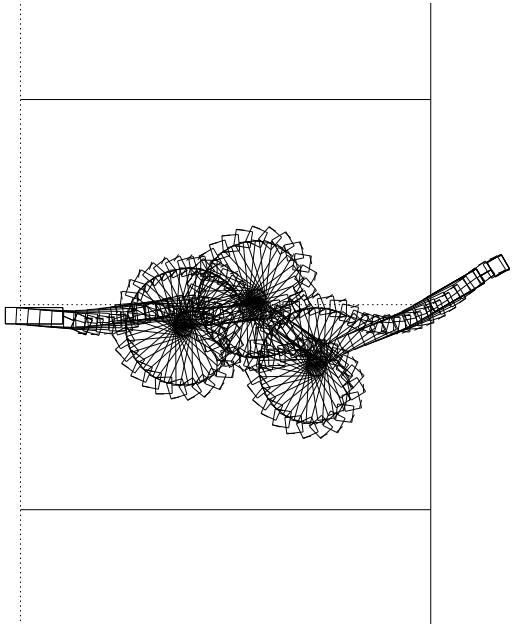
$$d^2 = x^2 + y^2 + min({\theta_S}^2, (\theta_c - 2\pi)^2, (\theta_c + 2\pi)^2)$$

$$Perf(u) = \min_{trajectoire} d^2$$

Variables **externes** – sauf GPS ultra-précis!

Performance (2)

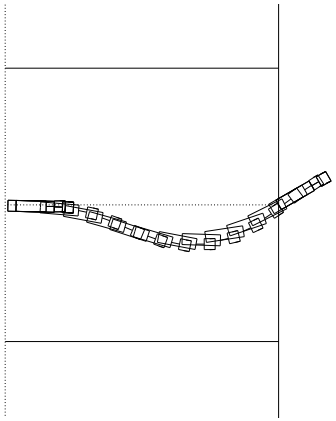
Le syndrome du “chauffeur de taxi”



Performance (3)

Prise en compte de la distance:

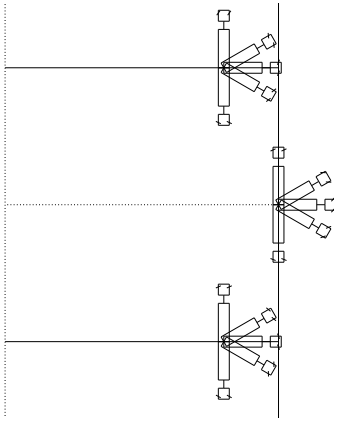
$$Perf(u) = L_{totale} + \min_{trajectoire} d^2$$



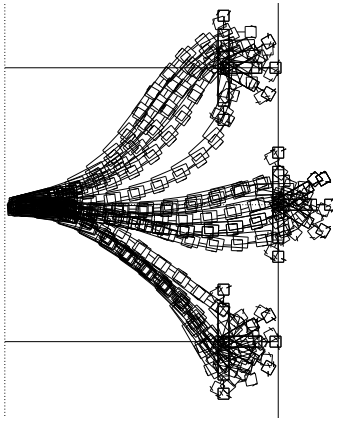
Généralisation

plusieurs points de départ $P_0, P_1, \ldots, P_n$: $P_i \rightarrow$ trajectoire T_i

$$Perf(u) = \sum_{i=0}^{i=n} \min_{T_i} d^2$$



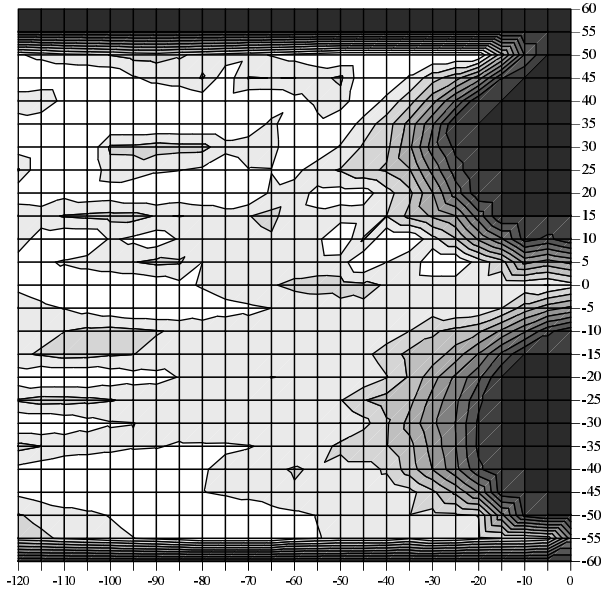
Les 15 points de départ



Le traffic résultant

Limites de l'approche

Les réseaux multi-couches à architecture fixée sont soit trop limités soit numériquement



Performance en généralisation du meilleur réseau

$\longrightarrow$ apprendre aussi l'**architecture** du réseau.

Évolution de la topologie des RN

A. Fadda et MS, 1996, Salvi et Trianni, 1999

- Optimise à la fois la topologie et les poids
- Pas d'opérateur de croisement sémantiquement significatif
- $\longrightarrow$ Evolutionary Programming:
  - tous les parents ont un enfant,
  - par mutation seulement,
  - les meilleurs des parents+enfants forment la génération suivante
- Mutations structurelles (topologie) et paramétriques (poids).

Les mutations

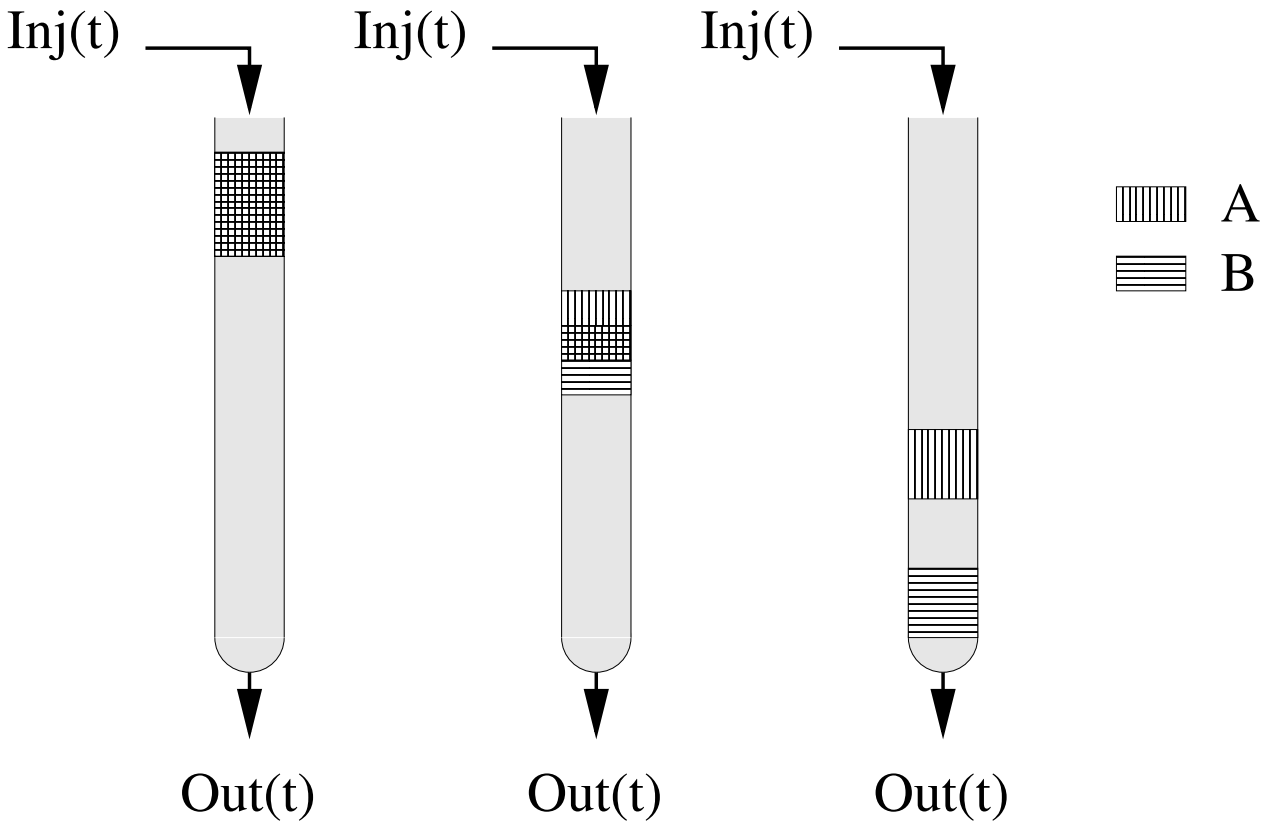
- les mutations structurelles — Les plus “douces” possibles
  - Ajouter ou enlever un neurone
  - Ajouter ou enlever une connection
  - Changer de fonction de transfert
- les mutations paramétriques : mutations gaussiennes
  - de tous les poids
  - des poids d’entrée d’un neurone
  - des poids de sortie d’un neurone
  - des coefficients des fonctions de transfert

Stratégie : une mutation structurelle est suivue d’un nombre minimum de mutations paramétriques

Chromatographie analytique

A. Fadda 97

La chromatographie est un procédé de **séparation** des composants d'un mélange par absorption sélective dans un milieu poreux.



Le système d'EDP

Si $w(z,t) = (w_1(z,t), w_2(z,t), ..., w_M(z,t))$ sont les concentrations des composantes du mélange,

$$\begin{cases} \frac{\partial w(z,t)}{\partial z} + \frac{\partial f(w(z,t))}{\partial t} = 0 \text{ for } (z,t) \in [0,L] \times [0,T] \\ w(0,t) = w^{inj}(t) \\ w(z,0) = w^{init}(z) \end{cases}$$

w^{inj}, w^{init} : **conditions expérimentales**,
f : fonction **isotherme**.

$w(L,t)_{t\in[0,T]}$: **chromatogramme** observé.

Il s'agit d'un système hyperbolique non-linéaire bien posé. (P. Valentin, F. James) qui peut se résoudre numériquement (e.g. schéma de Godunov standard).

État de l'art: Méthode déterministe

M. Sepulveda 96

- Il faut un modèle paramétrique (e.g. fractions rationelles),
- avec peu de paramètres.
- Il faut poser et résoudre le problème adjoint,
- et partir près de la solution.
- Ne peut utiliser qu'un seul chromatogramme expérimental.

Résultats

Un composant, un chromatogramme

Données expérimentales:

w^{inj} et w^{init}, les conditions expérimentales,

$w_c^{exp}(t)_{t\in[0,T]}$ le chromatogramme observé.

Fitness:

$$\sum_{c=1}^{M}\left\{\int_0^T \left|w_c^{NET}(L,t)-w_c^{exp}(t)\right|^2 dt\right\}^{1/2}$$

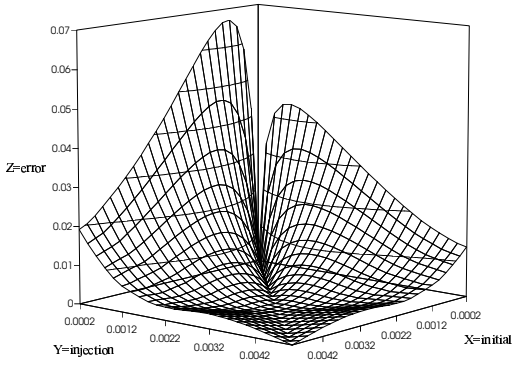
oú w_c^{NET} est solution de

$$\begin{cases} \frac{\partial w^{NET}}{\partial z}+\frac{\partial NET(w^{NET})}{\partial t}=0 \text{ in } \Omega=[0,L]\times[0,T] \\ w^{NET}(0,t)=w^{inj}(t) \\ w^{NET}(z,0)=w^{init}(z) \end{cases}$$

Remarque: Godounov $\equiv$ 2500 à 10000 calculs de la sortie du réseau !

- Résultats excellents — 150 generations de 25 réseaux
- Robuste par rapport au schéma numérique — → discrétisation grossière
- **mais** dépend des conditions expérimentales

Mauvaises propriétés de généralisation



Erreur sur une plage de conditions expérimentales.

⟶ Il faut prendre en compte plusieurs chromatogrammes dans la performance.

Un composant, plusieurs chromatogramme

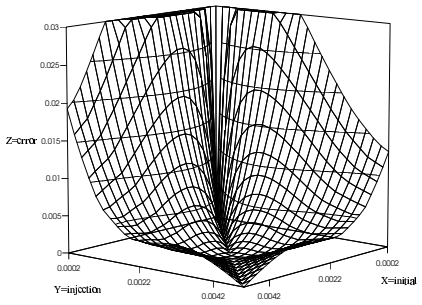
Données expérimentales:
w_i^{inj} et $w_i^{init}, i \in [1, N]$, les N conditions expérimentales,
$w_{c,i}^{exp}(t)_{t\in[0,T]}$ les N chromatogrammes correspondants.

Fitness:

$$\sum_{i=1}^{N}\sum_{c=1}^{M}\left\{\int_0^T \left|w_{c,i}^{NET}(L,t) - w_{c,i}^{exp}(t)\right|^2 dt\right\}^{1/2}$$

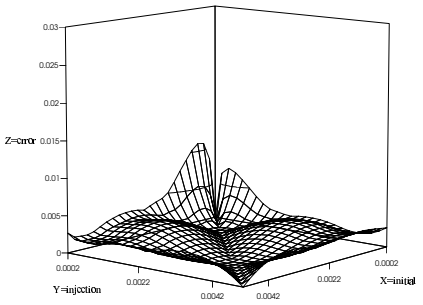
oú $w_{c,i}^{NET}$ est solution du système d'EDP de conditions expérimentales w_i^{inj} et w_i^{init} et de fonctions isotherme le réseau considéré.

$\longrightarrow$ Meilleure robustesse et meilleure précision.



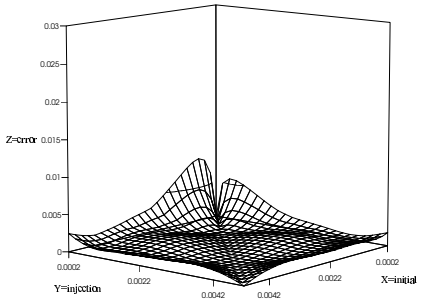
$N = 1$

$err_{max} = 0.06689$



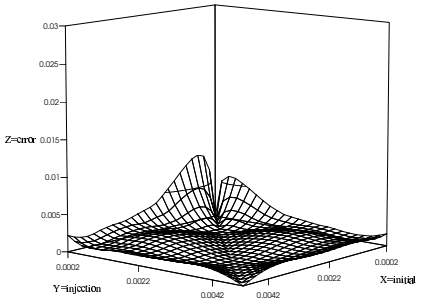
$N = 4$

$err_{max} = 0.01106$



$N = 9$

$err_{max} = 0.00876$



$N = 25$

$err_{max} = 0.00929$

Erreurs sur une plage de conditions expérimentales

Choix des conditions expérimentales

- Sur une grille — Impossible pour un nombre de corps important
- Aléatoirement choisis et modifiés en cours d'évolution

  Avec quelle fréquence ?
- Par co-évolution — Conditions expérimentales – fonctions isothermes (RN)

  Fitness des RN $\equiv$ min (erreurs sur CI)

  Fitness des CI $\equiv$ max (erreurs des RN)

Chromatographie et RN: conclusions

- Pas de modèle *a priori*
- Pas de problème adjoint
- Méthode coûteuse
- Permet de prendre en compte plusieurs données expérimentales (robustesse).
- Comparaison avec la méthode déterministe en cours sur données réelles (chères).
- Comparaison avec la Programmation génétique ???

Identification of macro-mechanical models

coll. F. Jouve, B. Lamy (CMAP), and M. Sebag, H. Maitournan (LMS)

Behavioral law of materials

- needed for accurate CAD;
- ill-known for new materials (e.g. polymers).

Art of macro-mechanical modeling:

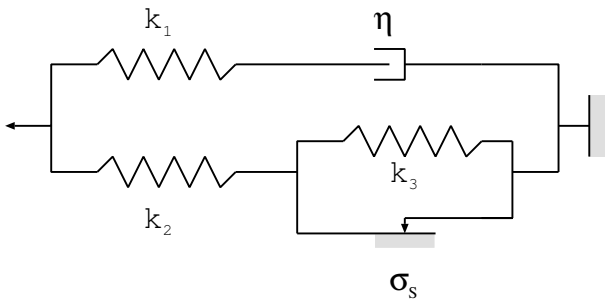
- Adapting the model of another material;
- Designing a brand new model;
- Starting with a micro-mechanical analysis.

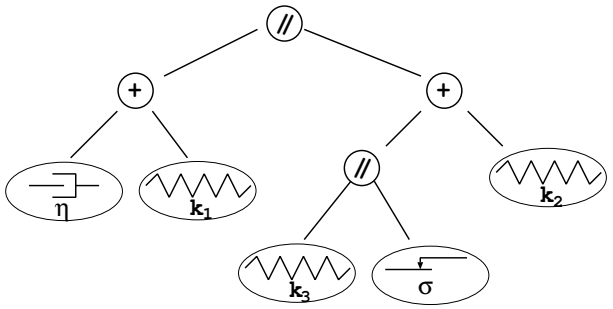
Fails when the current material:

- does not resemble other materials;
- does not fit expert's guesses;
- is not provided a tractable model by μ-M analysis.

Two different backgrounds:

- 3-D **hyperelastic materials** (e.g. rubber) are described by the *strain energy function* $E(I_1, I_2, I_3)$ (I_i are the invariants of the local strain tensor). GP representation is straightforward.
- 1-D **rheological models** describe general visco-elasto-plastic materials via assembly of elementary elements (springs, sliders and dashpots). Specific GP representation was designed:





Strain energy function identification

The direct problem

For a given constitutive law and a given loading, the Finite Element Method allows to compute the simulated behavior of the structure (displacements, stress).

$$\left.\begin{array}{l}\text{Boundary conditions}\\ \text{Forces}\\ E(I_1, I_2, I_3)\end{array}\right\} \xrightarrow{FEM} \text{Displacements.}$$

The inverse problem

$$\left.\begin{array}{l}\text{Boundary conditions}\\ \text{Forces}\\ \text{Dusplacements}\end{array}\right\} \xrightarrow{GP} E(I_1, I_2, I_3)?$$

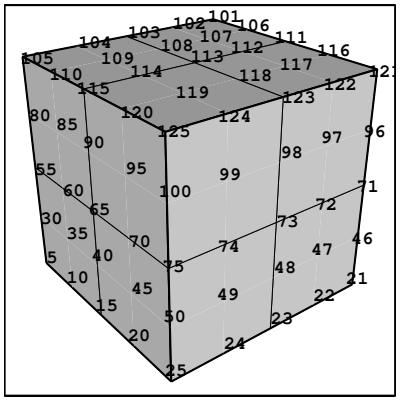
Why GP[a] ?

- Values of the second derivatives of the strain energy function are used in the FEM.
- Numerical derivation is unstable and inaccurate.
- Derivation of RNN is impossible.
- Derivation of GP-trees is straightforward.

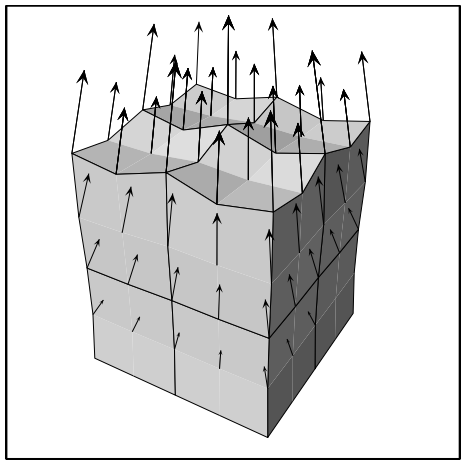
[a] We wanted to try GP on a non-regression problem

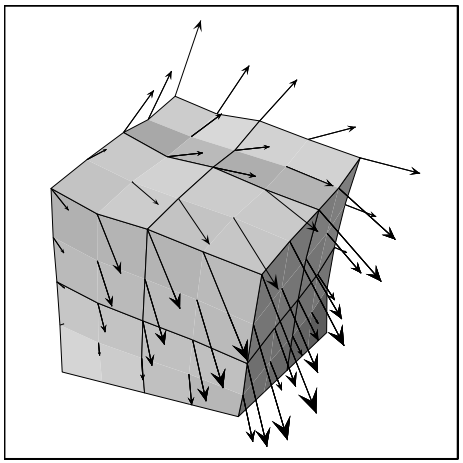
Fitness

- $2 \times 2 \times 2$ Q2-mesh.



- Three loading cases (the structure is fixed on its lower surface).





- Sum over the boundary points of the displacement errors.

One fitness evaluation for function E thus implies

- For each loading case
- For each iteration over the nonlinearity
  - For each element of the mesh
  - For each of the 27 Gauss points of the element
  - the computation of values of E, $\frac{\partial E}{\partial I_i}$, $\frac{\partial^2 E}{\partial I_i \partial I_j}$ at that point
- the numerical solution of the linear system (358×358).

and finally the computation of the displacements.

$\rightarrow$ More than 20000 tree-evaluations on average (for the simplissimus $2 \times 2 \times 2$ mesh !)

Worst complexity = when nonlinear iterations DO NOT converge

Tableau for the strain energy function identification

Objective:	Energy function fitting the experiments
Terminal set:	x, y, $\mathcal{R}$
Function set:	+, -, *
Fitness cases:	Three basic loading cases (different directions of forces)
Fitness:	Mean square error on the fitness cases
Parameters	Population size: 100
Operator rates	Crossover: 0.3 – Mutation: 0.5 – Copy: 0.2
Selection method	Tournament (2)
Replacement method	Generational
Wrapper	Compute α to meet the *no-load* condition.
Termination criterion	Final generation reached

First NON-results

- Disastrous first trials :
  $< 2\%$ of random trees are valid energy functions (i.e. nonlinear iterations in FEM converge).
- Test limits when variables go to $+\infty$:
  get rid of 80% of invalid functions.
- Use of Ogden model for strain energy function:

$$F(I_1, I_2, I_3) - \alpha log(I_3)$$

  where α is *a posteriori* computed such that

$$\frac{\partial E}{\partial I_1} + 2\frac{\partial E}{\partial I_2} + \frac{\partial E}{\partial I_3} = 0$$

  at the no-loading point.

Further NON-results

- The convergence of the FEM method for "weird" functions depends on the strength of the loadings.
- The smaller the forces, the more random (Ogden) functions have non-zero fitness.
- But all function behave like their linear part for small forces.

No significant result with small forces, no result at all with large forces.

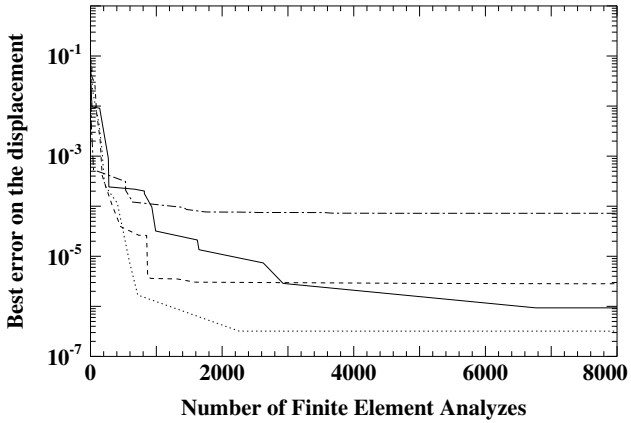
$\longrightarrow$ Need to gradually increase the intensity of the loading forces.

First results (Sigh!)

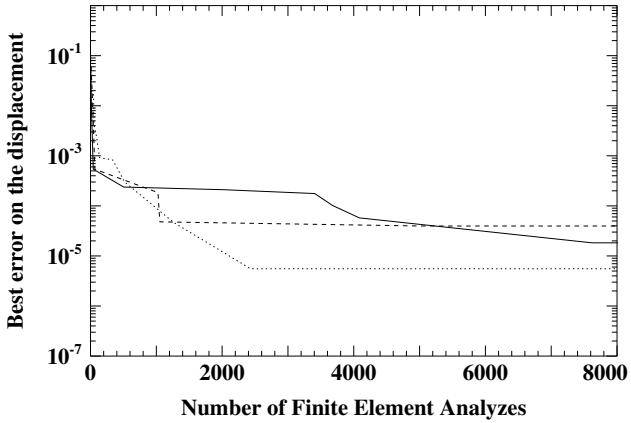
- Experimental results simulated using Mooney-Rivlin law:

$$E = aI_1 + bI_2 + cI_3 - (a + 2b + c)log(I_3)$$

- $(a, b, c) = (1, 1, 1)$: Law found in < 5 generations ... as good as random search !
- $(a, b, c) = (1.2, 1.1, 1.3)$: three out of four trials find out the linear form. Best result is
  $(a, b, c) = (1.2785, 1.0652, 1.3276)$



Results of four GP runs limited to 8000 FEA. Best errors: $3.2\,10^{-7}$, $9.33\,10^{-7}$, $2.83\,10^{-6}$ and $7.22\,10^{-5}$.



Results of iterated stochastic hill-climbing. Total number of FEAs: 32000. Best overall error: $5.55\,10^{-6}$

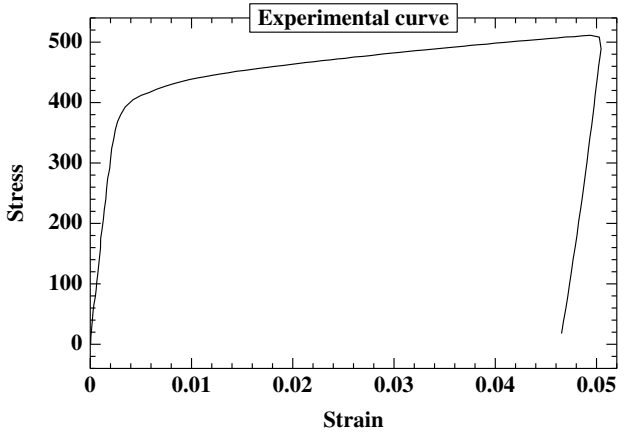
GP and hyperelastic materials: Conclusion

- Feasibility results
- Specific handling of real-valued terminals (partial local hill-climbing).
- Need for mechanical expertise all along
- Careful design of test experimental conditions
- Coupling with symbolic computation
- Post-GP optimization of real-values in best GP-trees (ES or hill-climbing).

1D Elasto-visco plastic materials

Input: Experimental curves

- observed strain $\epsilon(t)$ for applied stress $\sigma(t)$;
- observed stress $\sigma(t)$ for applied strain $\epsilon(t)$;



Output: Behavioral law

Differential equations linking $\epsilon(t)$, $\sigma(t)$ and their derivatives, e.g.

$$\begin{aligned} &\text{if} && \sigma(t) < \sigma_1 \text{ then } \sigma(t) = a.\epsilon(t) + b.\dot{\epsilon}(t) \\ &\text{else if} && \sigma(t) < \sigma_2 \text{ then } \sigma(t) = c.\epsilon(t) + d.\dot{\epsilon}(t) \end{aligned}$$

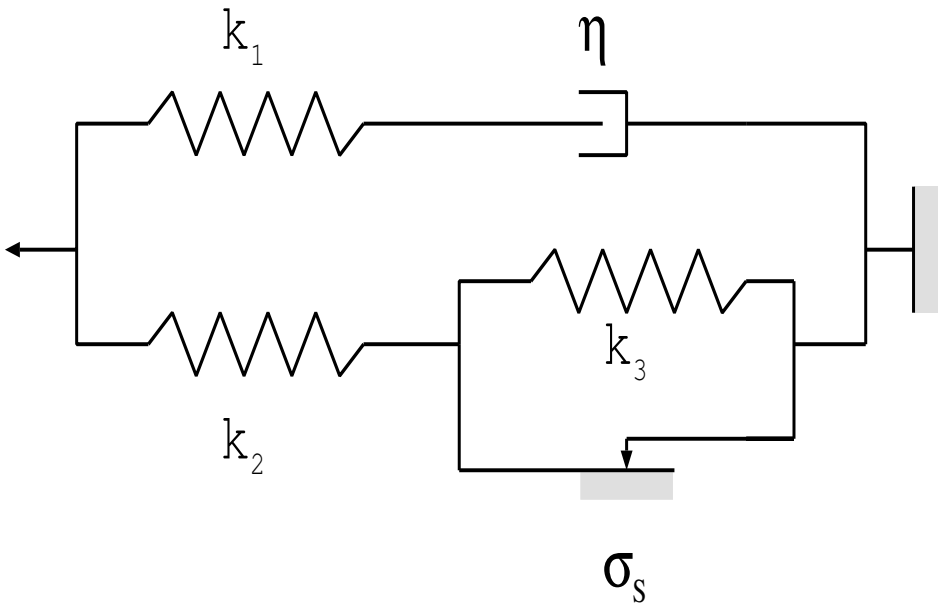
Criteria: the law must fit the experiments **and** be comprehensible.

Search space: Rheological models

Dynamic 1-D laws.

Assembly in series or parallel of

- springs (elastic behavior)
- sliders (plastic behavior)
- dashpots (viscous behavior)



Identification Goals:

- For a given model, adjust the parameters
  $\Longrightarrow$ **Parametric optimization**
- Optimize both the model and the parameters
  $\Longrightarrow$ **Non-parametric optimization**

Simulation of a Rheological model

Elementary equations:

- Spring(k) $\sigma(t) = k \ . \ \epsilon(t)$
- Slider(η) $\sigma(t) = \eta \ . \ \dot{\epsilon}(t)$
- Dashpot(σ_S) $(\dot{\epsilon}(t) = 0) \ OR \ (|\sigma(t)| = \sigma_S)$

Connection equations:

- Series
  $\epsilon_{parent}(t) = \epsilon_{child_1}(t) + .. + \epsilon_{child_m}(t)$
  $\sigma_{parent}(t) = \sigma_{child_1}(t) = .. = \sigma_{child_m}(t)$
- Parallel $\epsilon_{parent}(t) = \epsilon_{child_1}(t) = .. = \epsilon_{child_n}(t)$
  $\sigma_{parent}(t) = \sigma_{child_1}(t) + .. + \sigma_{child_n}(t)$

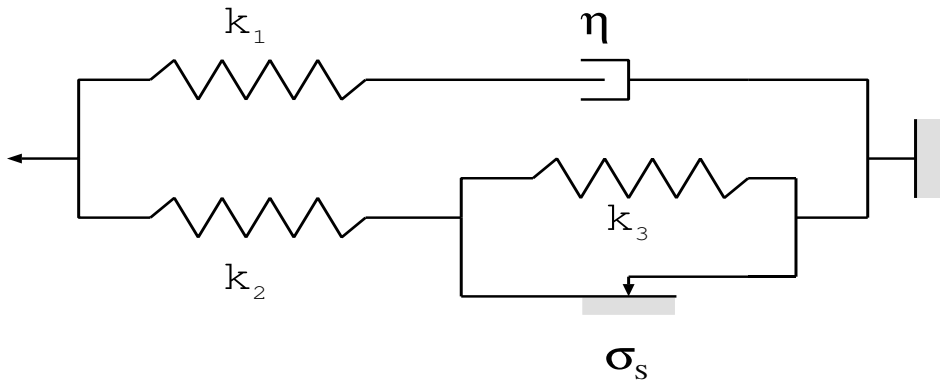
Solve $\{ \ equations \ \} \cup (\epsilon_{arrow}(t) = \epsilon_{exp}(t))$

$\rightarrow$

$$\sigma_{sim}(t) = Fn(\epsilon_{exp}(t), \dot{\epsilon}_{exp}(t))$$

Parametric identification

A given model (e.g. polyethylene)



$$\sigma_{sim}(t) = \mathcal{F}(\epsilon(t), \dot{\epsilon}(t); k_1, k_2, k_3, \eta, \sigma_S)$$

The unknown are $k_1, k_2, k_3, \eta, \sigma_S$, and the goal is to minimize

$$\sum_i |\sigma_{exp}(t_i) - \sigma_{sim}(t_i)|^2$$

Measures are available at discrete times t_i only ($\approx$ 30-300 values).

Ill-posed optimization problem

Methodology

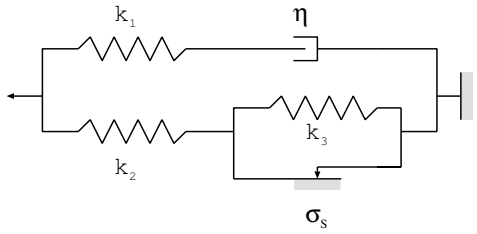
- Write the program computing σ_{sim} (by finite differences approximation of the equations). The parameters of that program are $k_1, k_2, k_3, \eta, \sigma_S$.
- Use trial-and-errors, or iterated hill-climbing to adjust the parameters.

Evolutionary Algorithms are a better choice!

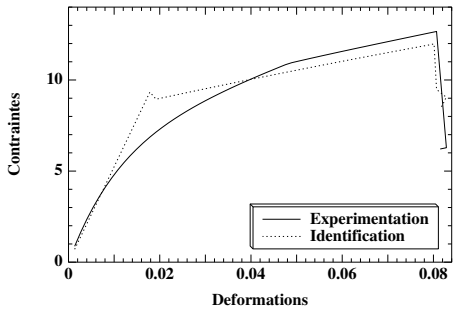
Standard $(10 + 30) - ES$ was used.

Results

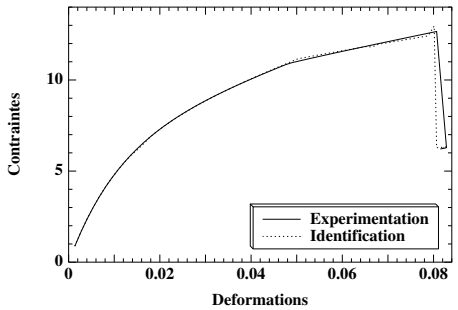
Model for the polyethylene



Responses of the best model in the population



after 20 generations



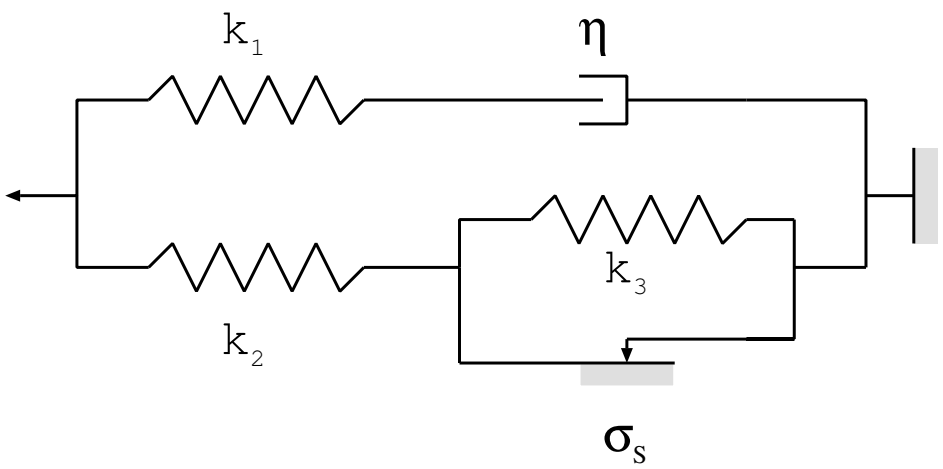
after 200 generations

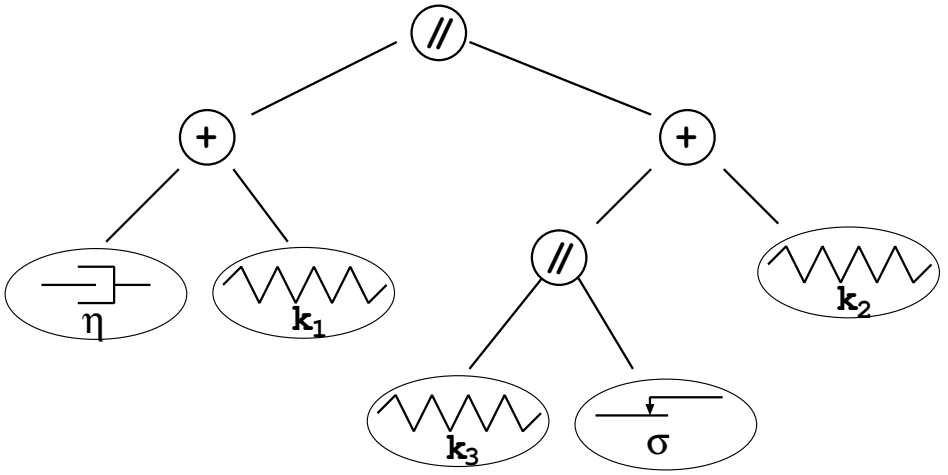
	k_1	η	k_2	k_3	σ_S
Identification	627.2	4748.	98.3	73.8	4.94
"Experimental"	587.9	4914.	93.1	116	4.49

Rheological GP

Rheological models ≡ Trees built from

- $\mathcal{N} = \{$ series +, parallel // $\}$
- $\mathcal{T} = \{$ Spring(k), Slider(σ_S), Dashpot(η)$\}$





Fitness computation

- Need for an **interpreter** of rheological model
- The sliders raise many difficulties (2 modes depending on σ w.r.t. the threshold.
- Complexity: $T \times 2.(3N)^3/3$, where T is the number of time steps of the loading history and N the size of the model
- Much slower than the compiled program used for parametric identification.

Evaluation

Compilation

$$H \quad \rightarrow \quad \textit{Système d'équations } \mathcal{S}_H$$

- Ressort(k) $\sigma(t) = k \ . \ \varepsilon(t)$
- Amortisseur(η) $\sigma(t) = \eta \ . \ \dot{\varepsilon}(t)$
- Patin(σ_S) $(\dot{\varepsilon}(t) = 0) \ OR \ (|\sigma(t)| = \sigma_S)$
- Série $\varepsilon_{parent}(t) = \varepsilon_{fils_1}(t) + \varepsilon_{fils_2}(t)$
  $\sigma_{parent}(t) = \sigma_{fils_1}(t) = \sigma_{fils_2}(t)$
- Parallèle $\varepsilon_{parent}(t) = \varepsilon_{fils_1}(t) = \varepsilon_{fils_2}(t)$
  $\sigma_{parent}(t) = \sigma_{fils_1}(t) + \sigma_{fils_2}(t)$

Simulation

$$\mathcal{S}_H \cup \ (\varepsilon_H(t) = \varepsilon_{exp}(t)) \quad \rightarrow \quad \sigma_H(t)$$

Evaluation

$$f(H) = Distance(\sigma_H, \sigma_{exp})$$

Critère d'arrêt

Sources d'erreur

- ED → Différences finies
- Erreurs expérimentales
- Bruit de résolution

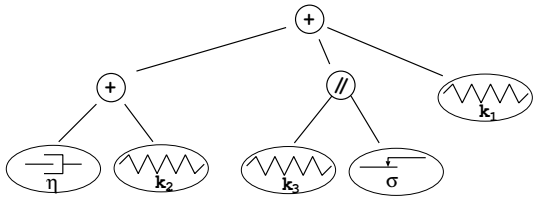
Estimation de l'erreur

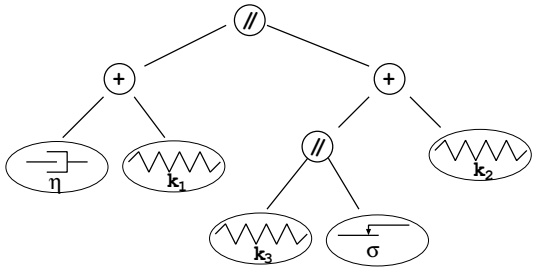
$$Err = ||\sigma_H(t_{exp} = t_1, t_2, t_3, ...) - \sigma_H(t_{exp} = t_1, t_3, t_5, ...)||$$

Critère de succès

$$f(H) \approx Err$$

Results

- 20% of successful runs (w.r.t. error criterion)
- 



Repeatedly found (wrong!) structure Compare to actual one

→ due to the absence of *creep* in the experiments.

- Best values of the parameters:

	k_1	η	k_2	k_3	σ_S
"Exp."	790.45	6248.60	150.20	41.60	7.25
Res.	998.89	8698.78	133.08	39.66	19.04

GP in law identification : Parametric vs non-parametric

Parametric identification

- gives more accurate results more rapidly
- ... if the guess of the model is good.
- otherwise, the bias can be misleading.

Non-parametric identification

- looks for solution in a much larger space
- ... but can easily get lost
- and may require heavier computational skills.

In both cases, the experimental data are crucial:
Use EC to discover discriminant experiments for similar models.
e.g. creep in the polyethylene case above

Embryogénèse : Synthèse de réseaux de neurones

F. Gruau, 95

Idée: Faire évoluer un programme dont l'exécution donne une solution.

Arbre → graphe de connexion

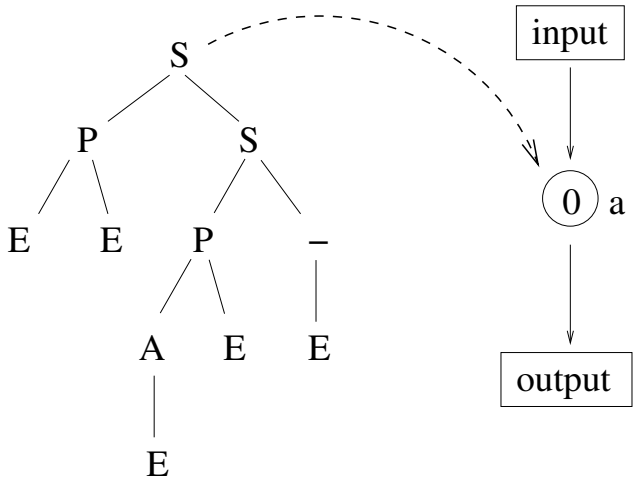
Embryogenèse : un "embryon" se développe selon un arbre de règles.

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

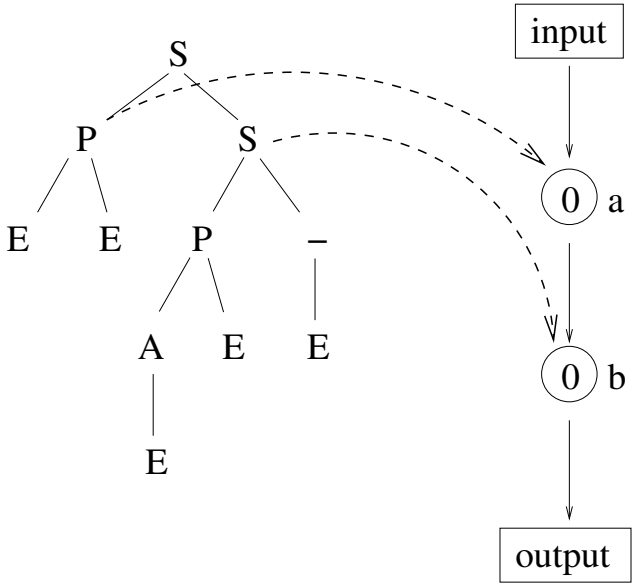
$\mathcal{T}$	E	symbole de fin
$\mathcal{N}$	S	Division séquentielle
	P	Division parallèle
	A	Incrémenter le seuil du neurone
	O	Décrémenter le seuil du neurone
	+	Incrémenter le poids d'un lien
	−	Décrémenter le poids d'un lien
	C	Couper le lien courant
	I	Incrémenter le numéro du lien courant
	D	Décrémenter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

Remarque : un seul terminal ⇒ besoin de mutation !

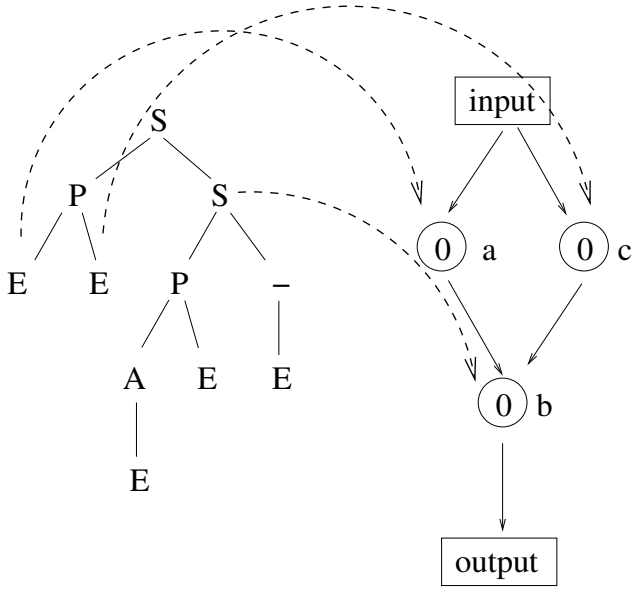
Développement d'un réseau de neurones



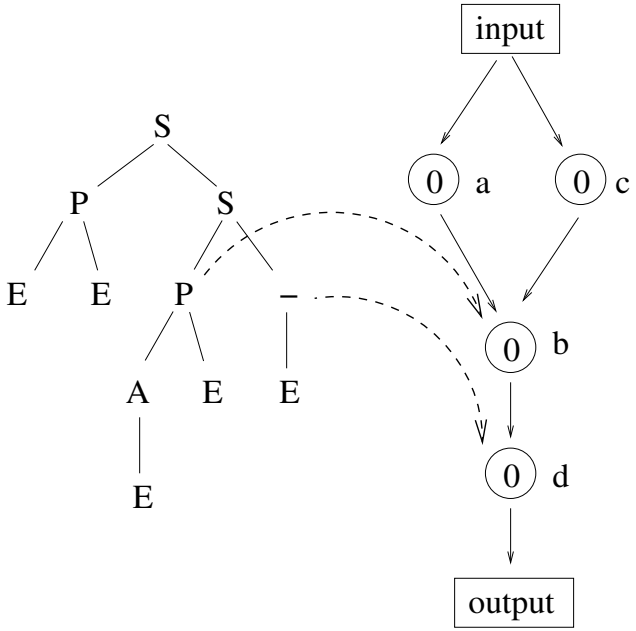
L'arbre et l'embryon



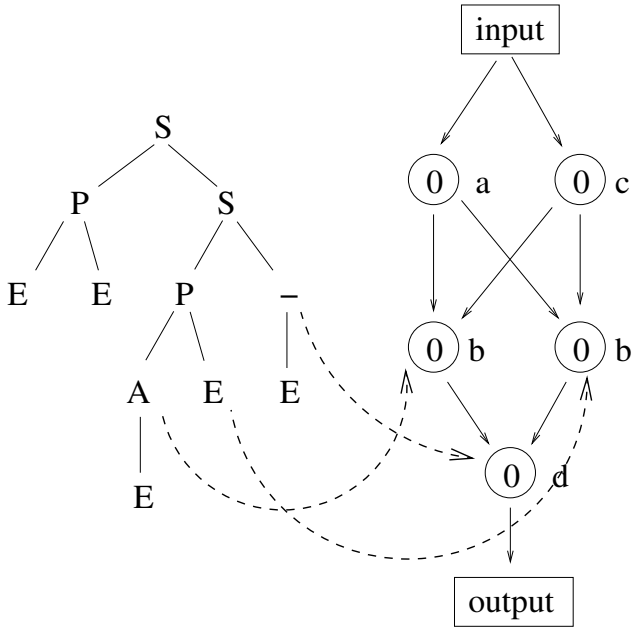
Étape 1



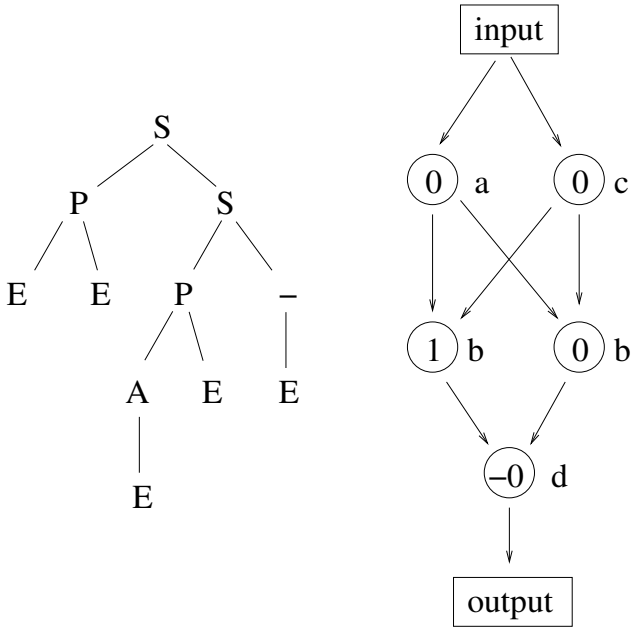
Étape 2



Étape 3, 4 et 5

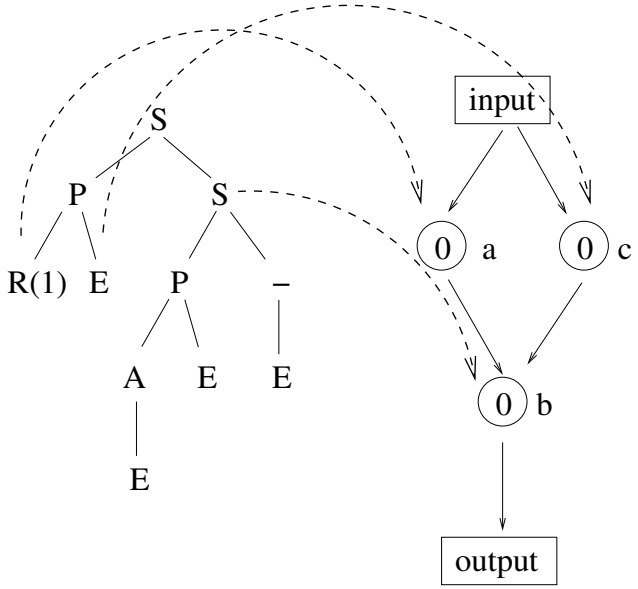


Étape 6

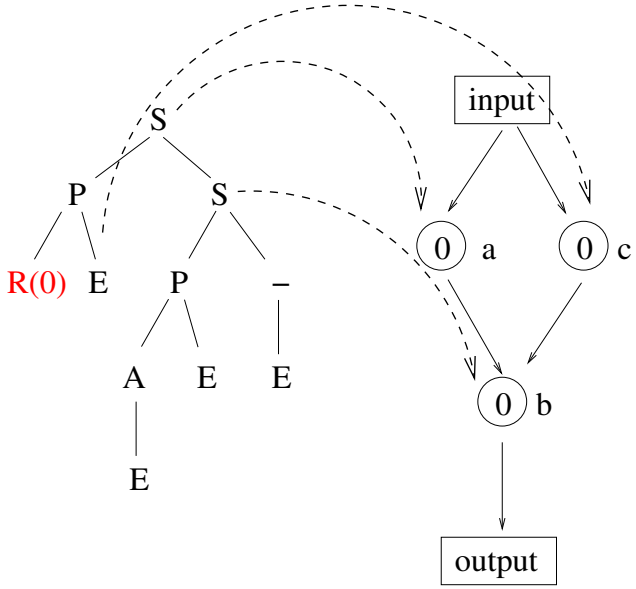


État final (réseau **XOR**)

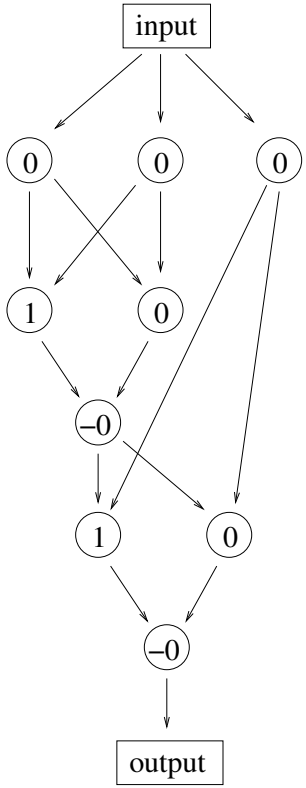
Avec récursion:



Étape 2



Étape 3



État final

Embryogenèse : 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)

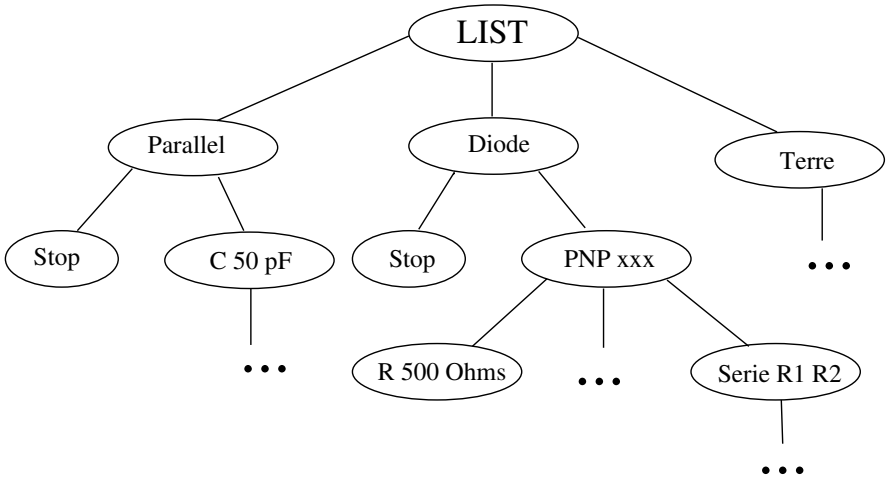
Réalisations :

Filtres passe-bande asymétriques

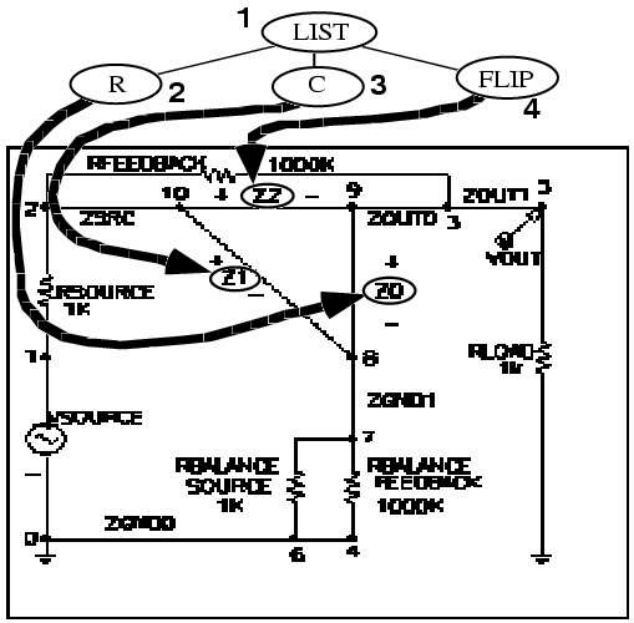
Extracteur racine cubique

Amplifieurs,...

Synthèse de circuits analogiques (2)

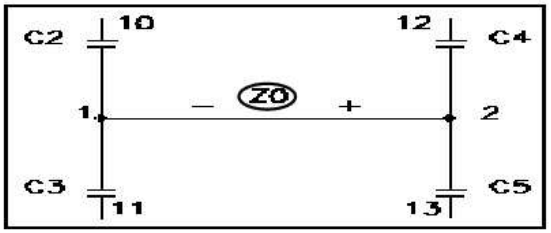


Un exemple de programme



et d'embryon

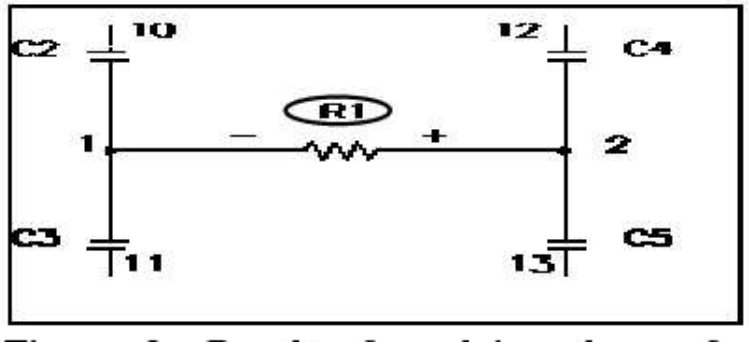
Exemples de noeuds et terminaux



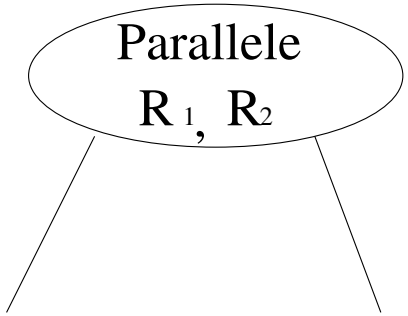
Fil générique

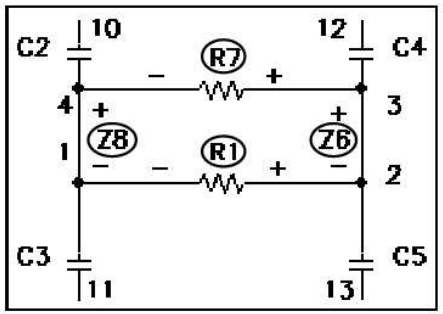
Développement :

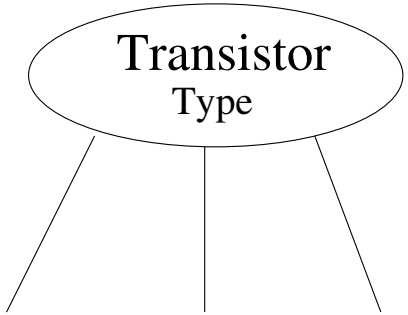
Un terminal

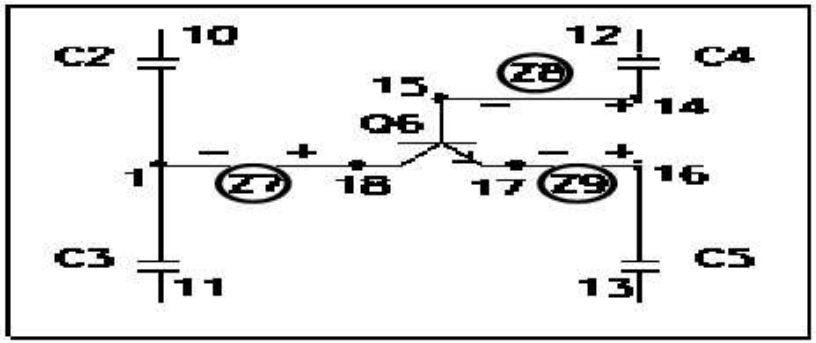


Application au fil générique







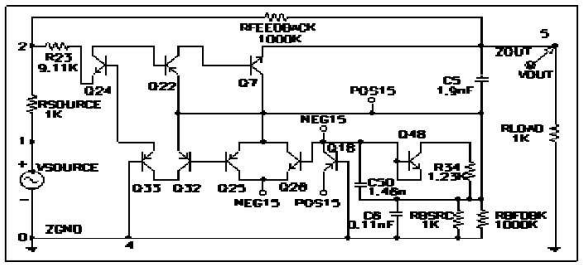


Noeuds

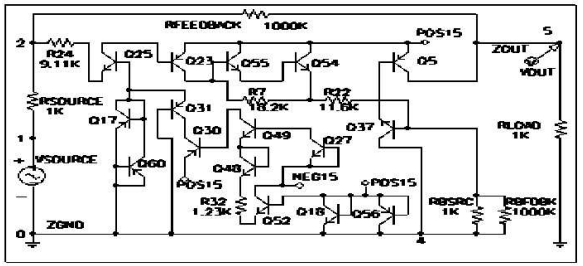
Application au fil générique

+ mémoire, récursion et subroutines

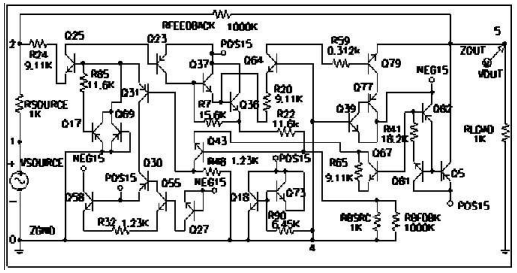
Un résultat: Ampli opérationnel 60dB



Meilleur initial



Génération 49



Génération 109

Mais taille population = . . . **640 000**

Identification de fonctions: Conclusion

- Les problèmes inverses mal posés sont des candidats à l'optimisation évolutionnaire
- Paramétrique / non-paramétrique — Pas de choix "naturel"
- Prendre en compte la connaissance du domaine
  - dans la représentation ;
  - dans les opérateurs de variation ;
  - dans la définition de la fitness ;
  - dans l'interprétation des résultats.
- Attention au choix des expériences

Ni les AEs ni l'application ne sont à traiter en boîte noire