Evolution Artificielle

et Programmation génétique

Michèle Sebag, LRI, Orsay

Plan

- Aspects spécifiques
- Passage à l'échelle
- Problèmes ouverts
- Une application en découverte scientifique
- Connaissances du domaine : couplage GP grammaires
- Dernières avancées : estimation de distribution

Programmation génétique

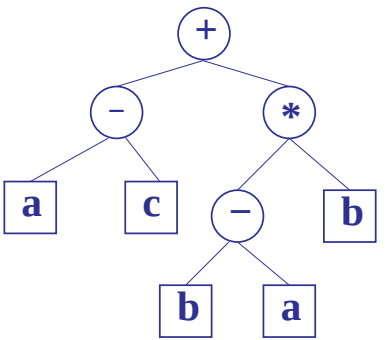
J. Koza – 1992

$\mathcal{F} : \Omega \mapsto \mathbb{R}$ Trouver argmax ($\mathcal{F}$)

Le rêve : Le programme qui écrit le programme

Ω = espace de programmes
$\mathcal{F}$ = qualité d'un programme
S-expressions :

$\mathcal{T}$ = { Vars, Cstes }
$\mathcal{N}$ = { opérateurs }



Quelques applications

- Classification
- Régression symbolique
- Prédiction séries chaotiques

- Stratégies multi-agents (e.g. jeux, . . .)
- Robotique
- Génération de plans

- Conception de circuits analogiques
- Apprentissage de réseaux neuronaux
- Modélisation mécanique

Concepts

GP = rejeton de GAs

Traits distinctifs : matériel génétique

structuré (souvent sous forme d'arbre)

de taille variable (bornée)

souvent exécutable

Historique :

Représentation: Langage LISP

Publications: Cramer 85, Koza 89, Koza 92, Koza 94

Remarque : pas d'alternative en optimisation classique

Espaces d'arbres

Etant donné :

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-clés

1. Les terminaux Variables du problème
   information accessible, constantes
2. Les noeuds PLUS $\neq$ MIEUX !
3. La fitness 99.9999% du coût de calcul !
4. Les génotypes profondeur des arbres
   initialisation
   opérateurs de variation
5. Le moteur d'évolution + application des opérateurs de variation
6. Critère de succès

Initialisation des arbres

- Choisir une profondeur max des arbres.
- Probabilités de sélection dans $\mathcal{N}$ et $\mathcal{T}$.
- Besoin d'opérateurs sécurisés (ex : $log(0)$?).
- La population initiale doit être suffisamment diverse.

Initialisation des arbres, 2

Procédure Grow : Lancer CreeArbreGrow($Prof_{Max}$)

Procédure (récursive) CreeArbreGrow(profondeur)

Si $profondeur == 1$

Retourner un élément x dans $\mathcal{T}$

Sinon

Choisir un élément x dans $\mathcal{N} \cup \mathcal{T}$

Si x est un noeud ($//x \in \mathcal{N}$), soit k l'arité de x

Pour $i = 1$ à k,

y_i = CreeArbreGrow(profondeur-1)

retourner $x(y_1, \ldots y_k)$.

Sinon retourner x.

Initialisation des arbres, 3

Procédure Full: Lancer CreeArbreFull($Prof_{Max}$)

Procédure (récursive) CreeArbreFull(profondeur)

Si $profondeur == 1$ Retourner un élément x dans $\mathcal{T}$

Sinon

Choisir un élément x dans $\mathcal{N}$, soit k l'arité de x

Pour $i = 1$ à k,

y_i = CreeArbreGrow(profondeur-1)

retourner $x(y_1, \ldots y_k)$.

Procédure "Ramped half and half", pour favoriser la diversité.

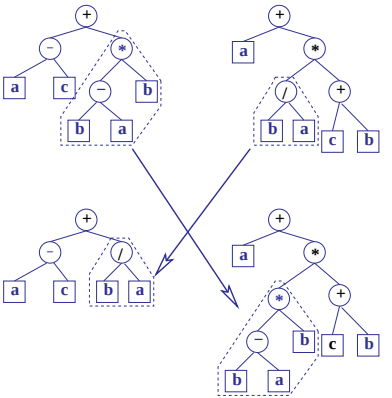
- Soit $N = \frac{P}{Prof_{Max}-1}$
- Pour i=2 à $Prof_{Max}$
- Créer $N/2$ arbres via Grow(i) et $N/2$ via Full(i)

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 ?

Remarque :

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

Mutation

Remarque

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

Mutation traditionnelle

- Remplacement de sous-arbre Choisir un point dans le parent
  Remplacer le sous arbre par un arbre aléatoire
- Remplacement de noeud Choisir un noeud/feuille dans le parent
  Remplacer ce noeud/feuille par un noeud/feuille de même arité

Mutation, 2

Mutation “promotionnelle”

- Insertion de noeud Choisir un point dans le parent
  Faire du fils le petit fils
- Promotion de noeud Choisir un noeud/feuille dans le parent
  Remplacer le noeud pere par le noeud fils

Point délicat de la mutation GP :

- Viole le “strong causality principle” : Pas de contrôle fin

Mutation: Terminaux numériques

Mutation "numérique"

- Muter toutes les constantes dans l'arbre. très destructeur
- Mise à niveau : tirer n jeux de constantes, garder le meilleur.
  coupler optim. non paramétrique/paramétrique
- Optimisation de type montée, ou ... évolutinnaire

Points délicats :

- Mutation structurelles:
  Possibilité de remettre en cause la structure de l'arbre
  $\Rightarrow$ Ajustement des constantes nécessaire
- Très peu de "petites" modifications possibles

Calcul de la fitness

- Il faut interpreter chaque arbre pour l'évaluer pour un jeu de valeurs terminales données
- On peut éventuellement le pré-compiler
  si nombreuses évaluations de l'arbre avec les mêmes données

Cas de fitness

Contexte : l'arbre doit satisfaire plusieurs objectifs
Exemples :

- jouer "bien" face à plusieurs joueurs
- trier un ensemble de séquences
- fitter plusieurs courbes

Difficulté : opportunisme de l'évolution.

- Battre le "meilleur" joueur $\not\Rightarrow$ jouer bien...
- Diminuer le désordre *moyen* crée des optima locaux...

Cas de fitness, 2

Reposer le problème :
Plusieurs cas de fitness

$\approx$ { Résolution de contraintes
Co-évolution des arbres et des cas

Heuristiques :

- Facteurs de pénalité pénaliser la dégradation sur un cas
- Poids sur les cas adaptation, techniques de boosting
- Augmenter le nombre de cas au cours de l'évolution
  la fitness doit préserver l'ordre

Plan

- Aspects spécifiques
- Passage à l'échelle
- Problèmes ouverts
- Une application en découverte scientifique
- Connaissances du domaine : couplage GP grammaires
- Dernières avancées : estimation de distribution

Modularité : la tondeuse à gazon

Contexte : une pelouse torique à $n \times n$ cases
Objectif : le programme de la tondeuse à gazon.

$\mathcal{T}$: Les feuilles, LEFT, TOND, $\mathcal{R}$

LEFT: Tourne à gauche

TOND: tond la case courante et avance

$\mathcal{R}$: couple (x, y), x, y dans $[1, n]$

$\mathcal{N}$: Les noeuds, V+, Proc, FROG

V+ : addition vectorielle modulo n	2 args.
Proc: exécute séquentiellement les deux arguments	2 args.
FROG: saute en un point et tond la case d'arrivée	1 arg.

La tondeuse à gazon, 2

Fitness :

la tondeuse part de la case (0,0)

la fitness est le nombre de cases de gazon tondues en m pas.

Choix :

- Sélection ?
- Filtrer les individus non adaptés ?
- Taille de population ?

Résultats cités : Koza 92

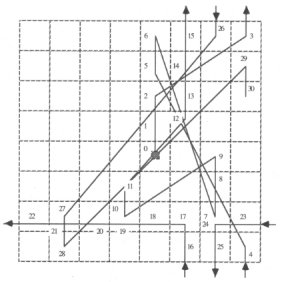
pelouse de 64 cases

population de 1000 individus

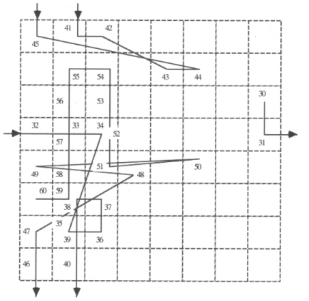
convergence en 34 générations

meilleur individu : 296 noeuds

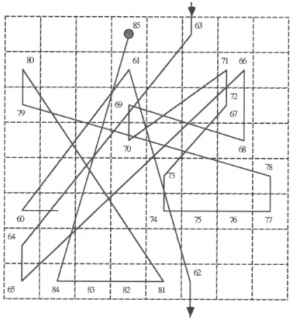
Solution, trajet (= phénotype)



Partie 1



Partie 2



Partie 3

Solution, description (= génotype)

```
(V8A (V8A (V8A (FROG (PROGN (PROGN (V8A (MOW) (MOW)) (FROG
(3,2))) (PROGN (V8A (PROGN (V8A (PROGN (PROGN (MOW) (2,4)) (FROG
(5,6))) (PROGN (V8A (MOW) (6,0)) (FROG (2,2)))) (V8A (MOW)
(MOW))) (PROGN (V8A (PROGN (PROGN (0,3) (7,2)) (FROG (5,6)))
(PROGN (V8A (MOW) (6,0)) (FROG (2,2)))) (V8A (MOW) (MOW))))
(PROGN (FROG (MOW)) (PROGN (PROGN (PROGN (V8A (MOW) (MOW)) (FROG
(LEFT))) (PROGN (MOW) (V8A (MOW) (MOW)))) (PROGN (V8A (PROGN
(0,3) (7,2)) (V8A (MOW) (MOW))) (PROGN (V8A (MOW) (MOW)) (PROGN
(LEFT) (MOW))))))))) (V8A (PROGN (V8A (PROGN (PROGN (MOW) (2,4))
(FROG (5,6))) (PROGN (V8A (MOW) (6,0)) (FROG (2,2)))) (V8A (MOW)
(MOW))) (V8A (FROG (LEFT)) (FROG (MOW))))) (V8A (FROG (V8A
(PROGN (V8A (PROGN (V8A (MOW) (MOW)) (FROG (3,7))) (V8A (PROGN
(MOW) (LEFT)) (V8A (MOW) (5,3)))) (PROGN (PROGN (V8A (PROGN (LEFT)
(MOW)) (V8A (1,4) (LEFT))) (PROGN (FROG (MOW)) (V8A (MOW)
(3,7)))) (V8A (PROGN (FROG (MOW)) (V8A (LEFT) (MOW))) (V8A (FROG
(1,2)) (V8A (MOW) (LEFT)))))) (PROGN (V8A (FROG (3,1)) (V8A
(FROG (PROGN (PROGN (V8A (MOW) (MOW)) (FROG (3,2))) (FROG (FROG
(5,0))))) (V8A (PROGN (FROG (MOW)) (V8A (MOW) (MOW))) (V8A (FROG
(LEFT)) (FROG (MOW)))))) (PROGN (PROGN (PROGN (PROGN (LEFT)
(MOW)) (V8A (MOW) (3,7))) (V8A (V8A (MOW) (MOW)) (PROGN (LEFT)
(LEFT)))) (V8A (FROG (PROGN (3,0) (LEFT))) (V8A (PROGN (MOW)
(LEFT)) (FROG (5,4)))))))) (PROGN (FROG (V8A (PROGN (V8A (PROGN
(PROGN (V8A (PROGN (PROGN (MOW) (2,4)) (FROG (5,6))) (PROGN (V8A
(MOW) (1,2)) (FROG (2,2)))) (V8A (MOW) (MOW))) (FROG (3,7)))
(V8A (PROGN (PROGN (MOW) (2,4)) (FROG (5,6))) (PROGN (V8A (MOW)
(6,0)) (FROG (2,2))))) (PROGN (PROGN (V8A (FROG (MOW)) (V8A
(1,4) (LEFT))) (PROGN (FROG (MOW)) (V8A (MOW) (3,7)))) (V8A
(PROGN (FROG (MOW)) (V8A (LEFT) (MOW))) (V8A (FROG (1,2)) (V8A
(MOW) (LEFT)))))) (PROGN (V8A (PROGN (FROG (2,4)) (V8A (MOW)
(MOW))) (V8A (FROG (MOW)) (LEFT))) (PROGN (3,0) (LEFT))))) (FROG
(V8A (7,4) (MOW)))))) (V8A (V8A (PROGN (MOW) (4,3)) (V8A (LEFT)
(6,1))) (MOW))).
```

Evaluation

Critères

- probabilité de solution
- complexité de la solution

Taille pelouse	32	64	96
Taille solution	145	280	427
Facteur	4.5	4.4	4.4
Nb évaluations t.q. Proba succès $> .99$	19 000	100 000	4 531 000

Passage à l'échelle

Remarque :

- *Nous* devons programmer structuré.
- Vrai aussi pour la machine ?

Scalable $\Rightarrow$ Réutilisation
Décomposition
Résolution hiérarchique

Approche descendante : programmation structurée
Approche ascendante : changement de représentation

Passage à l'échelle, 2

ADF : Automatically defined functions :

- ADF prédéfinies — Koza 92
- Modules emergents — Angeline 93
- ADF adaptatives — Koza 95

Intérêt :

Réutiliser des portions de code.

Rendre plus probable l'apparition d'un pgm efficace.

Principe des ADF prédéfinies

- Individu = Main
  ADF$_0$
  ...
  ADF$_K$
- ADF$_i$ = Feuilles = (sous-)ensemble de $\mathcal{T}$ +
  variables locales $arg_{i,1}, ..arg_{i,k}$
  Noeuds = (sous-)ensemble de $\mathcal{N}$ +
  ADF$_0$ + ADF$_1$ + ... ADF$_{i-1}$
- Main = Feuilles = $\mathcal{T}$
  Noeuds = (sous-)ensemble de $\mathcal{N}$ + ADF$_K$
- Fitness : évaluation de Main.

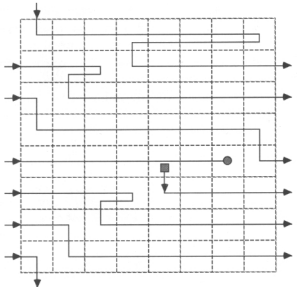
Tondeuse avec ADF

$ADF_0 : \mathcal{T} = \{ LEFT, TOND, \mathcal{R}\}$
$\mathcal{N} = \{ V+, Proc\}$

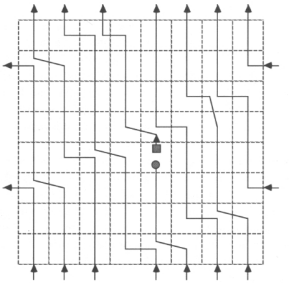
$ADF_1 : \mathcal{T} = \{ LEFT, TOND, \mathcal{R}, arg_1\}$
$\mathcal{N} = \{ V+, Proc, FROG, ADF_0\}$

Main : $\mathcal{T} = \{LEFT, TOND, \mathcal{R}\}$
$\mathcal{N} = \{ V+, Proc, FROG, ADF_0, ADF_1\}$

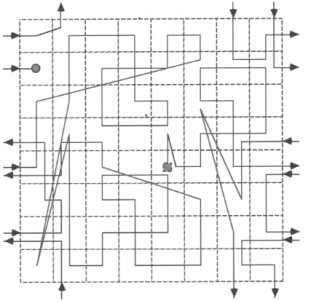
Solution avec ADF, trajet (= phénotype)



Run 1

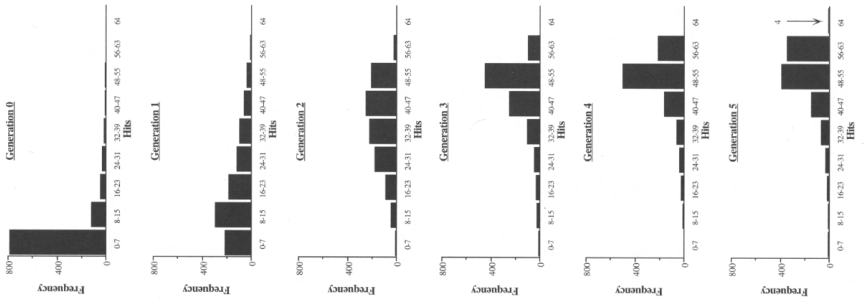


Run 2



Run 3

Solution avec ADF, un run



Evaluation

Sans ADF :

Taille pelouse	32	64	96
Taille solution	145	280	427
Nb évaluations t.q. Proba succès $> .99$	19 000	100 000	4 531 000

Avec ADF :

Taille pelouse	32	64	96
Taille solution	66.3	76.8	84.3
Nb évaluations t.q. Proba succès $> .99$	5 000	11 000	16 000

Embryogénèse : Synthèse de réseaux de neurones

F. Gruau, 95

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

Arbre $\rightarrow$ graphe de connexion

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

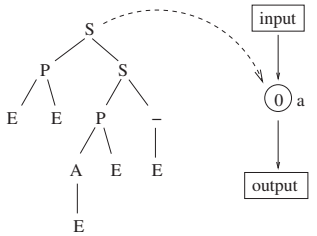
Sur un exemple, dans le contexte des réseaux de neurones booléens:

Embryogénèse, 2

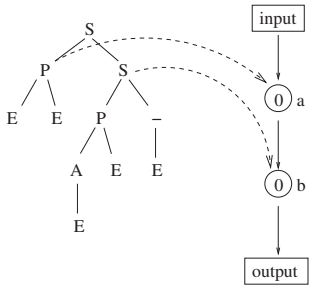
$\mathcal{T}$	E	symbole de fin
$\mathcal{N}$	S	Division séquentielle
	P	Division parallèle
	A	Incrémenter le seuil du neurone
	O	Décrémenter le seuil du neurone
	$+$	Incrémenter le poids d'un lien
	$-$	Décrémenter le poids d'un lien
	C	Couper le lien courant
	I	Incrémenter le numéro du lien courant
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

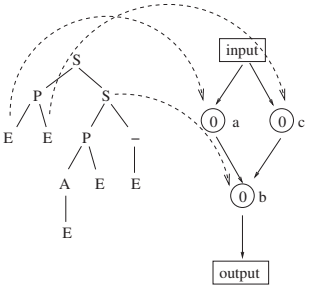


L'arbre et l'embryon

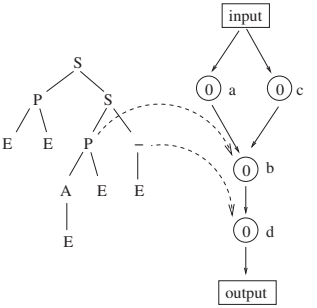


Étape 1

Développement d'un réseau de neurones, 2

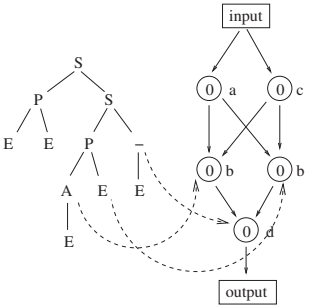


Étape 2

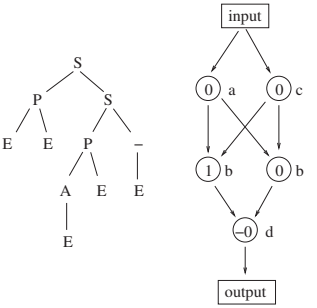


Étape 3, 4 et 5

Développement d'un réseau de neurones, 3



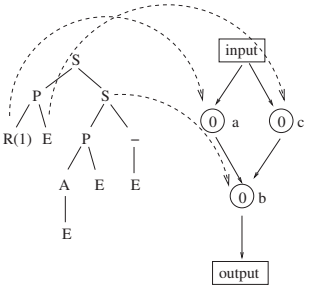
Étape 6



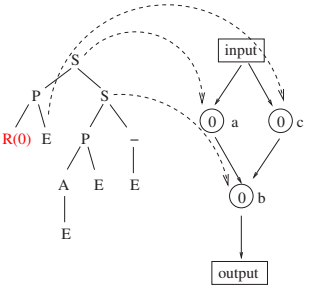
État final (réseau XOR)

Développement d'un réseau de neurones, 4

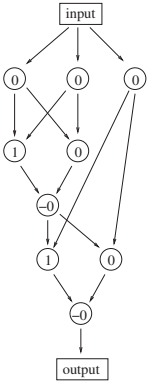
Avec récursion:



Étape 2



Étape 3



État final

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)

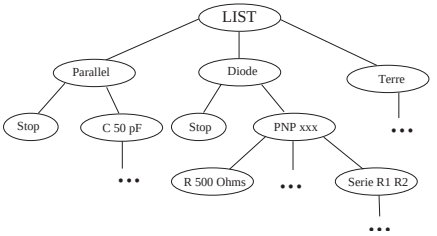
Réalisations :

Filtres passe-bande asymétriques

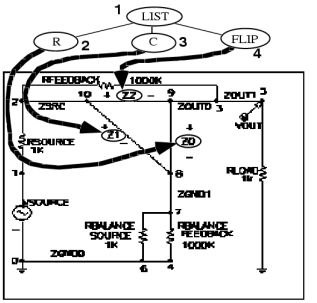
Extracteur racine cubique

Amplifieurs,...

Synthèse de circuits analogiques, 2



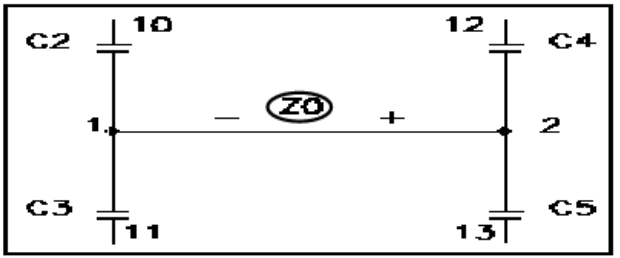
Un exemple de programme



et d'embryon

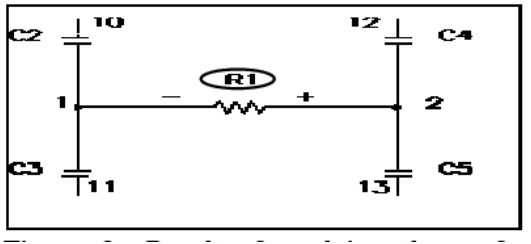
Exemples de noeuds et terminaux

Fil générique



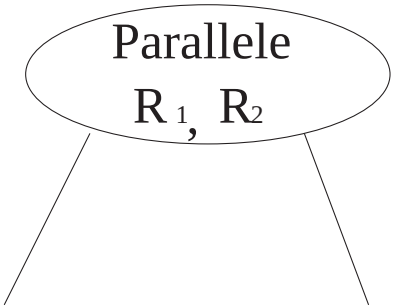
Développement :

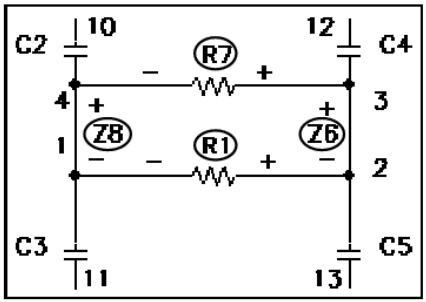
Resistance
R_1 Ω

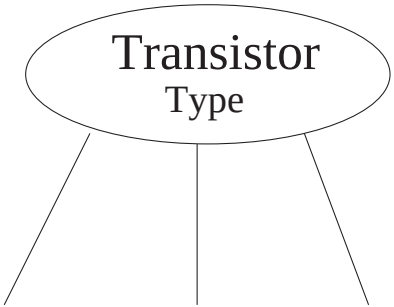


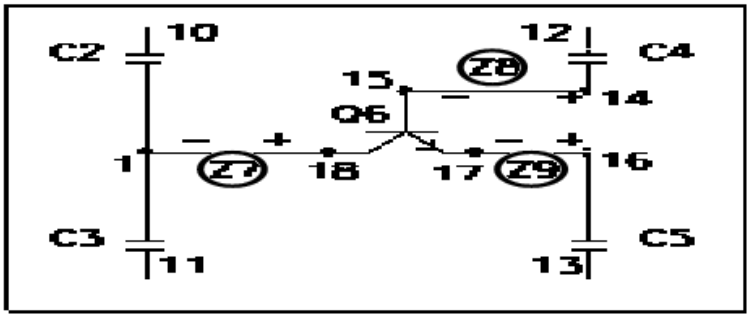
Un terminal

Application au fil générique









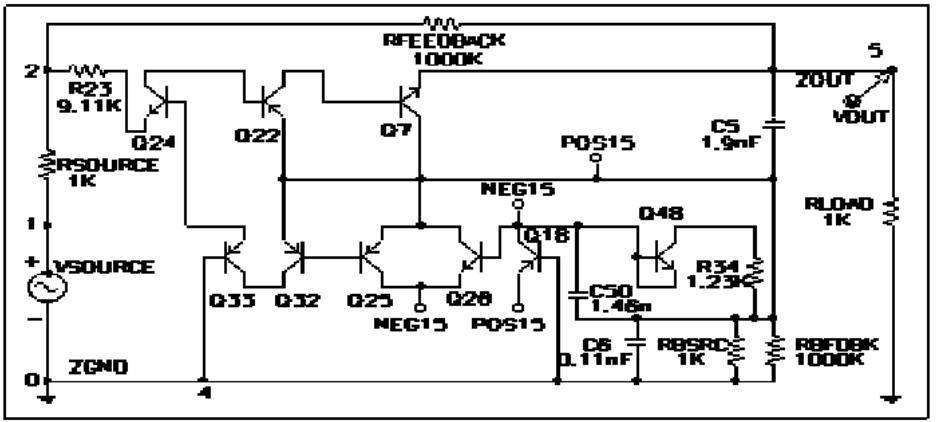
Noeuds

Application au fil générique

+ mémoire, récursion et subroutines

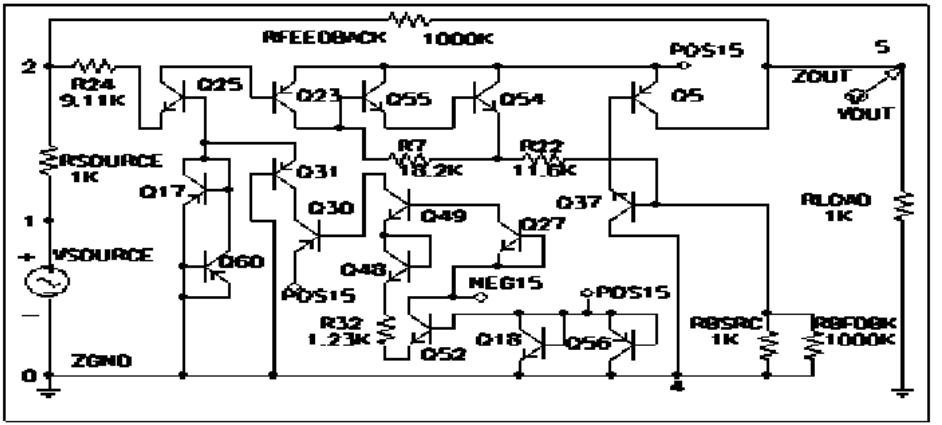
Un résultat: Ampli opérationnel 60dB

Meilleur initial



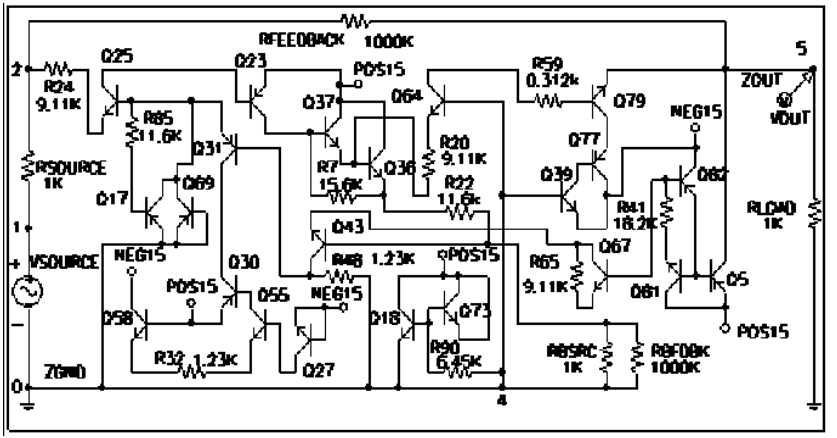
Un résultat: Ampli opérationnel 60dB, 2

Meilleur, génération 49



Un résultat: Ampli opérationnel 60dB, 3

Meilleur, génération 109



ATTENTION : taille population = 640 000

Plan

- Aspects spécifiques
- Passage à l'échelle
- Problèmes ouverts
- Une application en découverte scientifique
- Connaissances du domaine : couplage GP grammaires
- Dernières avancées : estimation de distribution

Particularités et problèmes ouverts

- Evaluation : presque toujours en contexte de conception
  objectif = au moins un run avec résultat excellent
  un critère = nb évaluations t.q. P(solution) $>$ .99
- Heuristiques pour préserver la diversité
  à revoir
- Contrôle de la taille mémoire de la population
  le phénomène du bloat
- Qu'est-ce qu'un schéma ?...

La diversité

GP $\neq$ GA ! $\left\{ \begin{array}{l} Sharing \\ Niching \\ Clearing \\ \ldots \end{array} \right.$ $\Rightarrow$ Distance sur Ω !

distance entre arbres = problème ouvert

Parade possible : distance phénotypique.

Si Fitness $\equiv$ { performance cas 1, ... performance cas K }

alors $\Omega \mapsto \mathbb{R}^K$

$T \mapsto g(T) = (perf(T,1), \ldots, perf(T,$

$\text{distance}(T, T') = ||g(T) - g(T')||$

Bloat et Introns

Définitions :

- Intron: matériel génétique sans effet sur la fitness
  Pas spécifique à GP

- Bloat: augmentation (exponentielle) du nb d'introns

Exemple :

IF $\{1\}$ then $\{\ldots\}$ else $\{\ldots\}$

$0 \times \{\ldots\}$

Bloat et Introns, 2

Le débat :

- Les introns paralysent l'évolution
  taille mémoire, temps de calcul de la fitness
- Les introns "aident" l'évolution
  quand on les enlève, ça marche moins bien...
  - Mécanisme de mémoire ?
  - Permettre "petites" modifications ?

Mécanisme d'apparition :

Le croisement fait apparaitre les introns
La sélection les multiplie

Limiter la taille

- Un facteur de parcimonie

$$\mathcal{F}(T) \leftarrow \mathcal{F}(T) + c.\textit{longueur}(T)$$

- Post-process de simplification
  1. Trouver T ayant une bonne performance
  2. Fixer la performance et minimiser la longueur

Plan

- Aspects spécifiques
- Passage à l'échelle
- Problèmes ouverts
- Une application en découverte scientifique
- Connaissances du domaine : couplage GP grammaires
- Dernières avancées : estimation de distribution

Découverte, exemple

	$Battery$	$Wire$	I	C	I/C
e_1	A	X	3.4763	3.4763	1.0000
e_2	A	Y	4.8763	4.8763	1.0000
e_3	A	Z	3.0590	3.0590	1.0000
e_4	B	X	3.9781	3.4763	1.1444
e_5	B	Y	5.5803	4.8763	1.1444
e_6	B	Z	3.5007	3.0590	1.1444
e_7	C	X	5.5629	3.4763	1.6003
e_8	C	Y	7.8034	4.8763	1.6003
e_9	C	Z	4.8952	3.0590	1.6003

But : Inférer des lois physiques (chimiques,..)

$$U = RI$$

Identification of macro-mechanical models

coll. M. Schoenauer, H. Maitournam

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.

Machine Discovery

- First Era (1983)

  Langley, Falkenhainer, Nordhausen,...

  Heavy assumptions
  Heuristic construction of new terms

  PV, PV/T, PV/nT,..

- Second Era (1995)

  Dzeroski, Valdes-Perez,...

  Exhaustive exploration
  Challenge : restricting the search space

- On the ILP side

  Srinivasan, Camacho, Simon, Frisch,...

  Fixed equations
  Parametric optimization

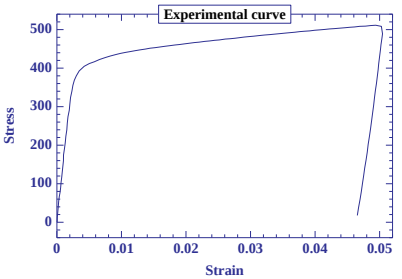
Dilemma

- Strong background knowledge
  $\longrightarrow$ exhaustive exploration is feasible

- Reasonable background knowledge
  + smart optimization criterion
  = greedy search

- "Light" background knowledge
  + stochastic search
  = global optimization

Identification of Behavioral Laws



Input: Experimental curves

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

Identification of Behavioral Laws, 2

Output: Behavioral law

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

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

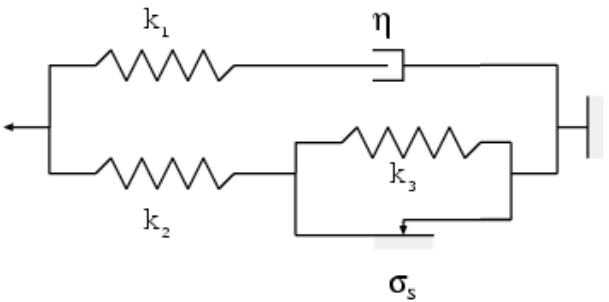
Criteria: the law must fit the experiments and be comprehensible.

Search space: Rheological models

Dynamic 1-D laws

Assembly in series or parallel of

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



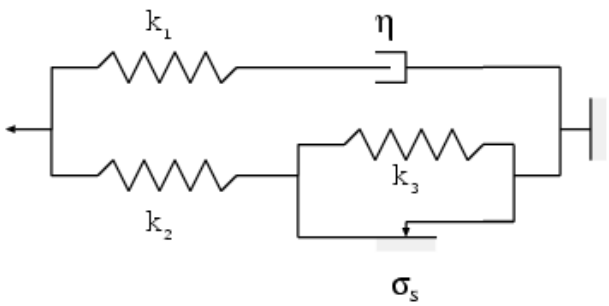
Identification Goals:

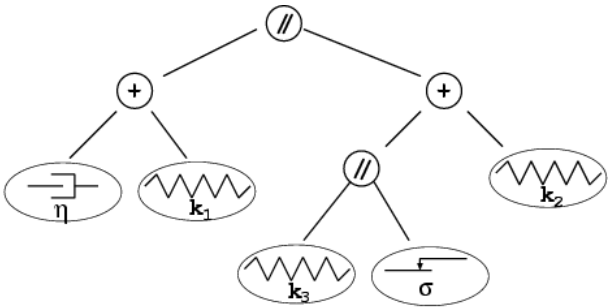
- For a given model, adjust the parameters
  $\Longrightarrow$ Parametric optimization
- Optimize both the model and the parameters
  $\Longrightarrow$ Non-parametric optimization

Rheological GP

Rheological models $\equiv$ Trees built from

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





Initialisation

Creer Arbre:

Choisir Noeud dans { +, //, ressort, patin, amort }

Si Noeud dans { ressort, patin, amortisseur }

Tirer Constante k, σ_S, η dans $\mathbb{R}^+$

Retour (Noeud, constante)

Si Noeud dans { +, // }

$Fils_1$ = Creer Arbre

$Fils_2$ = Creer Arbre

Retour (Noeud, $Fils_1$, $Fils_2$)

Evaluation

Compilation

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

- 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

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

Evaluation

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

Critère d'arrêt

Sources d'erreur

- ED $\longrightarrow$ 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

$$f(H) \approx Err$$

Conclusion partielle

- Identification de modèle rhéologique par GP
  Premiers résultats positifs.
  Passage à l'échelle difficile.

- C'est un cas favorable :
  Tout élément de l'espace de recherche est acceptable...

Plan

- Aspects spécifiques
- Passage à l'échelle
- Problèmes ouverts
- Une application en découverte scientifique
- Connaissances du domaine : couplage GP grammaires
- Dernières avancées : estimation de distribution

GP and Background Knowledge

- EC: same evolution as AI
  I. A universal tool 1965
  II. Knowledge makes the difference 1991

- GP: the closure assumption
  any subtree is a valid operand for any operator.

Pros	simple crossover
	simple mutation
Cons	Search space size

GP and Background Knowledge, 2

- Syntactic constraints

  Gruau 96, Keijzer Babovic 99

- Strongly typed GP

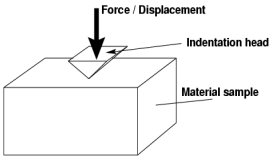
  Montana 97

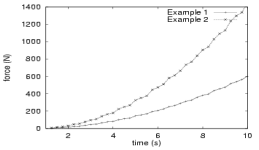
- BNF Grammars

  Horner 96, Ryan et al. 98

Dimension aware GP
The mechanical problem

Indentation experiments on unknown material





Goal: Find expression $\mathcal{F}$ s.t.

$$\text{Force} = \mathcal{F}(\text{displacement, time, material parameters})$$

Ratle, Sebag, 2000

GP and Machine Discovery

Trivial BK: Dimension-consistency

meters + seconds ? Oh !

Assumption:

finite set of units	$\{m, s, kg\}$
compound units	$U_{ijk} : m^i s^j kg^k$
limited combinations	$i, j, k \in [-2, 2]$

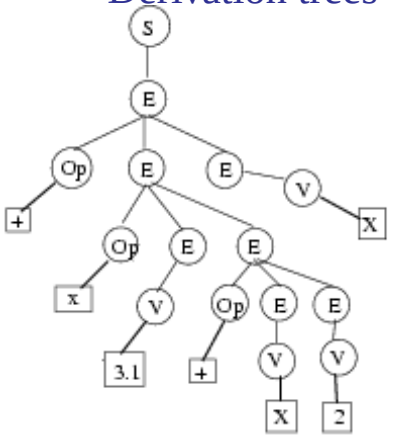
Representation: BNF grammars

S	start symbol	$U_{1,-2,1}$
N	non-terminals	$\{U_{ijk}\}$
T	terminals	$\{Vars, \mathcal{R}, +, -, *, /, exp\}$
P	production rules	

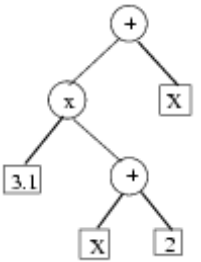
$$
\begin{aligned}
U_{ijk} := \quad & U_{ijk} + U_{ijk} \mid U_{ijk} - U_{ijk} \mid U_{ijk}\ exp^{U_{000}} \\
& \mid_{abc+def=ijk} U_{abc} * U_{def} \\
& \mid_{abc-def=ijk} U_{abc} / U_{def} \\
& \mid_{unit(var)=ijk} Var
\end{aligned}
$$

Enforcing constraints through grammars

Derivation trees ⟶ Parse trees



Derivation tree



Parse tree

Beware !	CFG	GP
Terminals	variables, constants, operators	variables and constants
Non-Terminals	typed expressions	operators

GP on derivation trees

Gruau 96

- Initialization: uniform selection among derivations in a production rule

  filter out trees with depth $> D_{max}$

- Crossover: swap nodes with same non-terminal symbol

  $\equiv$ Strongly Type Genetic Programming

  Montana 1995, Haynes et al. 1996

- Mutation: select another derivation

Dimensional grammar

Physical units			
Quantity	mass	length	time
Variables			
K (Elastic element)	+1	0	–1
n (Viscous element)	+1	0	–1
t (time)	0	0	+1
u (displacement)	0	1	0
Solution			
F (Force)	1	1	-2

Automatic generation of the grammar

each compound unit → a non-terminal symbol

admissible combinations → production rules

N	non-terminals	$\{U_{ijk}\}$
T	terminals	$\{Vars, \mathcal{R}, +, -, *, /, exp\}$
P	production rules	

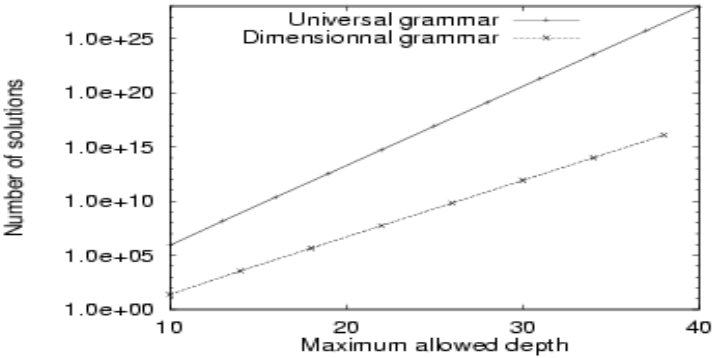
$$
\begin{aligned}
U_{ijk} := \; & U_{ijk} + U_{ijk} \mid U_{ijk} - U_{ijk} \mid U_{ijk}\, exp^{\,U_{000}} \\
& \mid_{abc+def=ijk} U_{abc} \; * \; U_{def} \\
& \mid_{abc-def=ijk} U_{abc} \; / \; U_{def} \\
& \mid_{unit(var)=ijk} Var
\end{aligned}
$$

$$
\mathcal{F} := \quad mass \times length \times time^{-2}
$$

Automatically generated

First Results

Good Reduction of the search space



First Results, 2

but Poor performances ...blamed on Initialization

Uniform initialization: $\mathcal{P}$(non-terminal) $\gg$ $\mathcal{P}$ (terminal)

deep trees, most are filtered out

Note :

Similar to constrained optimization with sparse feasible region

Ryan et al, 1998

Poor initial population $\longrightarrow$ poor performances

Initialization in Grammar Guided GP

Biased initialization fails

- Set $\mathcal{P}$(terminals) $\gg$ $\mathcal{P}$(non-terminals)
- Population poorly diversified, premature convergence

Constraint resolution for initialization

- Minimal tree depth for each non-terminal or derivation
- On-line filtering out of derivations
  incompatible with maximum depth
- GP initialization = constraint solver

$\longrightarrow$ Diversified initial population within depth D_{Max}

Constrained Initialization for Grammar-Guided GP

- Compute $d_{min}(U) = U$ minimal depth

$$U := deriv_1 \mid \ldots \mid deriv_N$$
$$d_{min}(U) = min_i \; d_{min}(deriv_i)$$
$$d_{min}(U_1 \; op \; U_2) = 1 + max(d_{min}(U_1), d_{min}(U_2))$$

- Construct Exp with maximal depth D_{Max}

$Exp = S$; $d_{max}(S) = D_{Max}$

While (exists non terminal symbols in Exp)

Select U in Exp $U = |_i \; deriv_i$

Select $deriv_i$ / $d_{min}(deriv_i) \leq d_{max}(U)$

$deriv_i = U_1 \; op \; U_2$

Set $d_{max}(U_1) = d_{max}(U_2) = d_{max}(U) - 1$

- Result: **admissible and diversified** individuals

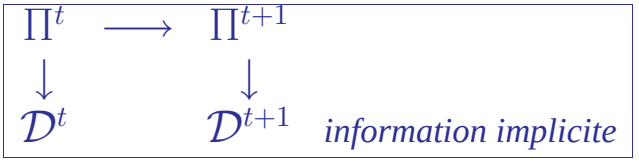
Plan

- Aspects spécifiques
- Passage à l'échelle
- Problèmes ouverts
- Une application en découverte scientifique
- Connaissances du domaine : couplage GP grammaires
- Dernières avancées : estimation de distribution

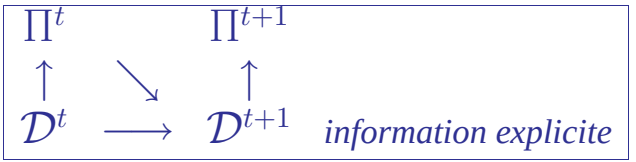
Algorithmes d'Estimation de Distribution

Principe : Extension (population) $\longrightarrow$ Intension (distribution).

Algorithmes d'évolution



Une approche IA



Removing genetics from standard genetic algorithms, S. Baluja and R. Caruana, ICML 95

Population-Based Incremental Learning

Algorithme

Init :

$\mathcal{D} = (.5, \ldots, .5)$

Boucle

1. Génération de la population

$$X_i = \begin{cases} 1 & \text{si } r < \mathcal{D}_i \\ 0 & \text{sinon} \end{cases} \qquad r \text{ uniforme dans } [0, 1]$$

2. Evaluer et trier la population
3. Mettre à jour la distribution

$$\mathcal{D} \leftarrow (1 - \alpha)\mathcal{D} + \alpha X_{best}$$

Population-Based Incremental Learning, 2

Discussion

Peu de paramètres :

- Taille de la population
- Taux de mise à jour (learning rate) α

Pas de transmission du matériel génétique

Simplicité

Limitations

Converger ?

si $\mathcal{D}_i = 1$ ou $0 \rightarrow$ perte de diversité

si $\mathcal{D}_i \neq 1 \Rightarrow$ sur OneMax, on ne trouve jamais l'optimum

Toujours Exploration vs Exploitation

- Prise en compte de X_{worst} :

si $X_{best,i} \neq X_{worst,i}$

$$\mathcal{D}_i \leftarrow (1-\alpha)\mathcal{D}_i + \alpha X_{best,\,i}$$

sinon ($\beta << \alpha$),

$$\mathcal{D}_i \leftarrow (1-\beta)\mathcal{D}_i + \beta X_{best,\,i}$$

- "Muter" la distribution

si $\tau < .005$ (taux de perturbation)

$$\mathcal{D}_i \mathrel{+}= \mathcal{N}(0,\epsilon)$$

(ϵ = amplitude de perturbation)

Limitations, 2

Distribution produit $\equiv$ composantes indépendantes

Echec sur les problèmes trompeurs (deceptive)

Exemple :

$$\mathcal{F}(x_1, \ldots, x_5) = \begin{cases} 5 & \text{si } \sum x_i = 5 \\ 4 - \sum x_i & \text{sinon} \end{cases}$$

Problèmes trompeurs concaténés :

$$\mathcal{G}(x_1, \ldots, x_{20}) = \sum_{i=1}^{16} \mathcal{F}(x_i, \ldots x_{i+4})$$

Mutual Information Maximization for Input Clustering

Distribution chainée

Soit $\pi = (i_1, i_2, ...i_n)$ une permutation

On cherche

$$P(X) = p(X_{i_1}) \times p(X_{i_2}|X_{i_1}) \times \ldots \times p(X_{i_n}|X_{i_{n-1}})$$

- Trouver i_1 de plus faible entropie
- Trouver i_j de plus faible entropie conditionnelle par rapport aux variables déjà choisies.

De Bonet et al., NIPS 97.

Autres algorithmes EDAs

- COMIT, Combining Optimizers with Mutual Information Trees Baluja Davies 1997
- EBNA, Estimation of Bayesian Network Algorithm, Etxeberria Larranaga, 1999.
- PBIL_c, PBIL for continuous domains, Ducoulombier Sebag 1998.
- BOA, Bayesian Optimization Algorithm, Pelikan et al. 1999

Probabilistic Grammar-Guided GP

Principle: Setting weights on each derivation

Scal: $deriv_i \longrightarrow w(deriv_i)$

Vect: $deriv_i \times depth\ k \longrightarrow w(deriv_i,\ k)$

Salustowicz & Schmidhuber, 1998

Initializing Weights

$\forall\, i, \forall\, k,\ \ w(deriv_i,\ k) = 1$

Using Weights

for U at depth k, if $d_{min}(deriv_i) \leq d_{max}(U)$

Prob(Select $deriv_i$) $\propto w(deriv_i,\ k)$

Probabilistic Grammar-Guided GP, 2

Updating Weights

Loop on the best individuals

if $deriv_i$ is chosen at depth k

$w(deriv_i,\ k) *= (1 + \epsilon)$

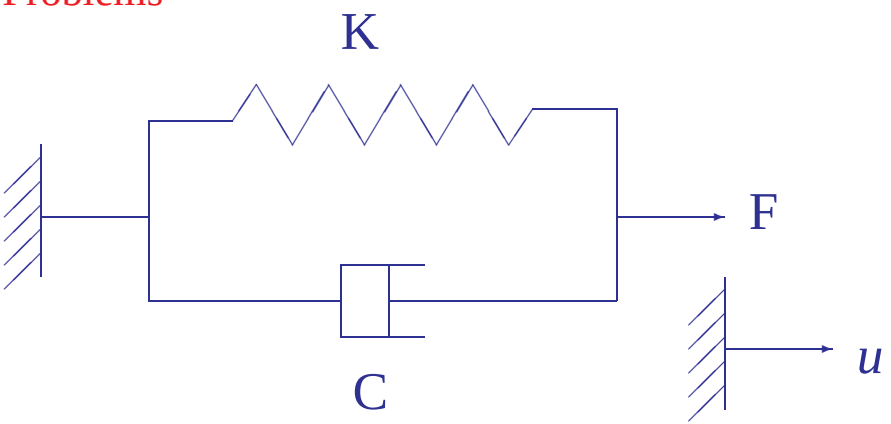
Loop on the worst individuals

if $deriv_i$ is chosen at depth k

$w(deriv_i,\ k) *= (1 - \epsilon)$

Empirical Validation

Problems



$$x(t) = \frac{F}{K}\left(1 - e^{\frac{-Kt}{C}}\right) \quad (1)$$

$$x(t) = \frac{F}{K_1} + \frac{Ft}{C_1} + \frac{F}{K_2}\left(1 - e^{\frac{-Kt}{C_2}}\right) \quad (2)$$

Empirical Validation, 2

Grammars

Universal

$$S := NT$$
$$NT := T \mid OP\ NT\ NT$$
$$OP := + \mid - \mid * \mid \div \mid exp$$
$$T := F \mid K \mid C \mid t \mid 1 \mid 2 \mid 3 \mid 4$$

Universal + exp-neg: same as above, except

$$OP := + \mid - \mid * \mid \div \mid exp \mid exp-$$

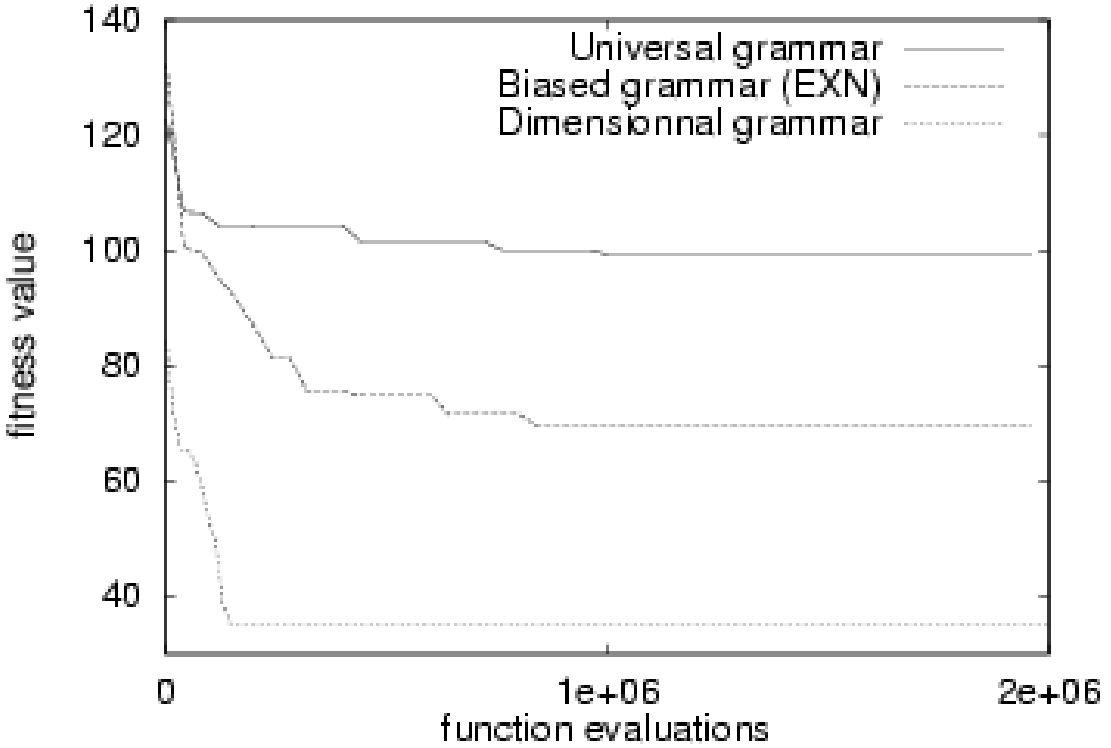
Dimensional: dimension-consistent grammar

Physical units			
Quantity	mass	length	time
Variables			
E (Force)	+1	+1	–2
K (Elastic element)	+1	0	–1
n (Viscous element)	+1	0	–1
t (time)	0	0	+1
Solution			
x (displacement)	0	+1	0

Empirical Validation, 3

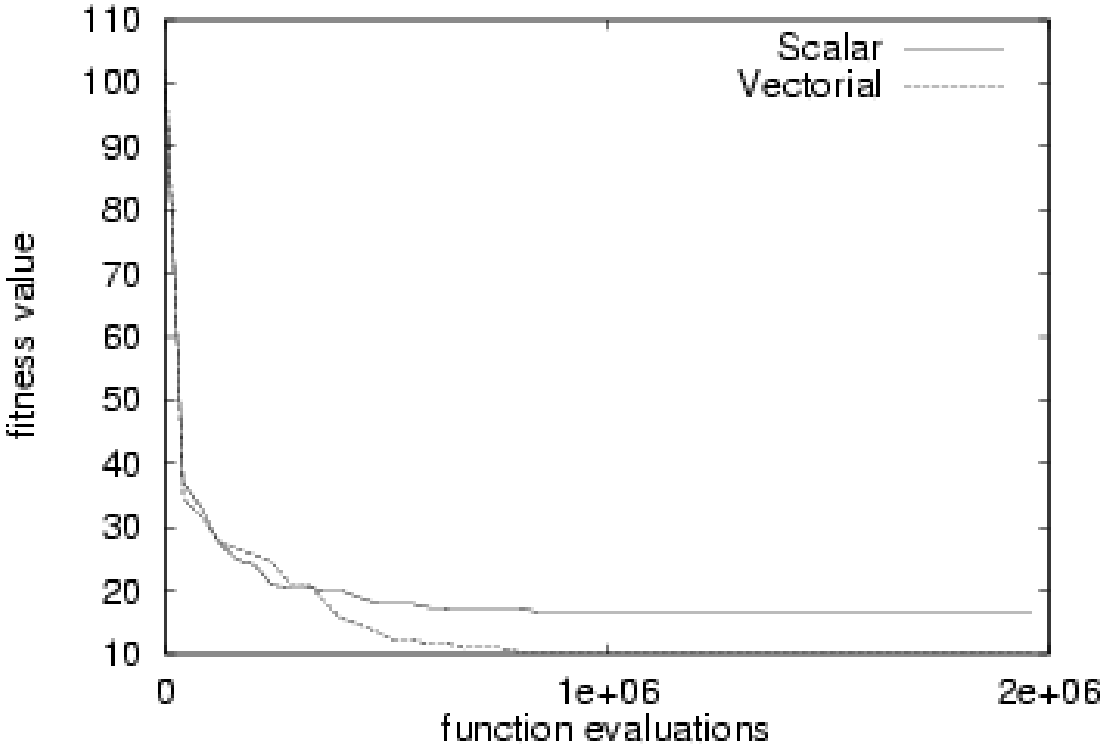
Parameter	Value	
Algorithm	GP	GPwPG
Population size	2000	500
Max. number of generations	1000	4000
Probability of Crossover	0.8	—
Probability of tree mutation	0.2	—
Probability of point mutation	0.8	—
Nb of best individuals for learning	—	2
Nb of worst individuals for learning	—	2
Learning rate (ϵ)	—	0.001
Probability of perturbation	—	0.001
Amplitude of perturbation	—	0.001
Number of training examples	20	20
Number of independent runs	10	20

Results



Convergence: Impact of grammars

Results

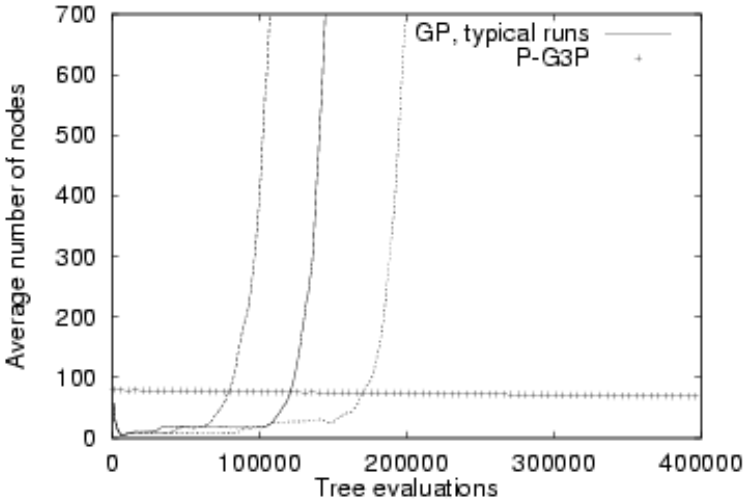


Convergence: Impact of learning distributions

Overcoming the bloat

The bloat phenomenon:

- protection against destructive crossover ?
- some crossover side effect ?



Distribution-guided GP:

No bloat.

Conclusion 1 : Learning

- Is learning an optimization problem ... what is the criterion ?
- Background Knowledge $\equiv$ Restricting the search space $\mathcal{H}$
- Evolutionary Computation $\equiv$ Generate and test
- Induction: Feedback from trials
  Capitalized as a distribution on $\mathcal{H}$

En guise de conclusion finale
Evolution de l'évolution

EC $\approx$ IA :

1er temps : outil universel GPS

2eme temps : pouvoir $\neq$ savoir KBS

Recommandations :

- Aider le programme
- Renforcer les récompenses
- Pas trop de punitions
- Ne pas imposer la destination et le chemin