L2 – Vie Artificielle

Alexandre Allauzen – Michèle Sebag
LIMSI – LRI

22 nov. 2013

Overview

Généralités

Initialisation

Opérateurs de variation

Fitness

Modularité

Programmation génétique

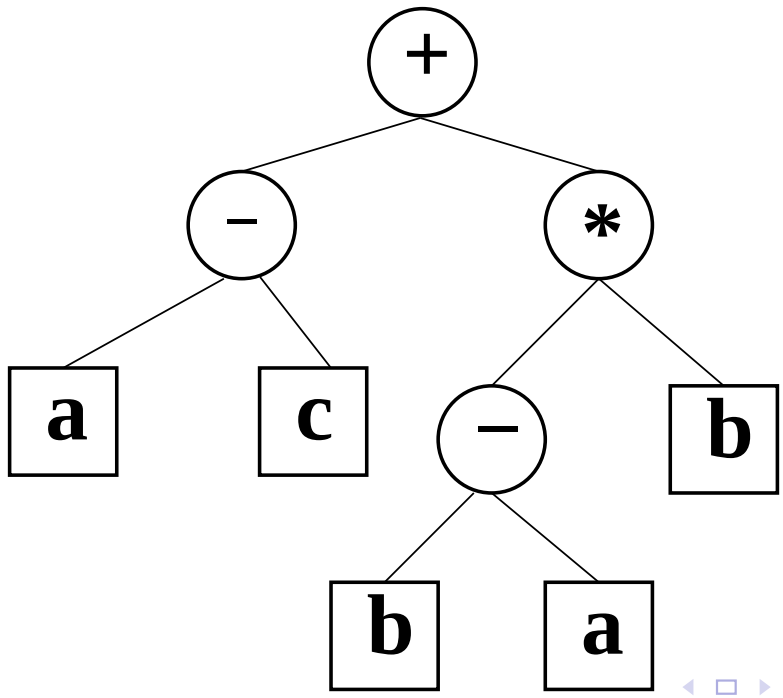
J. Koza – 1992

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

Le rêve : Le programme qui écrit le programme

$\Omega =$ 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)

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

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

Overview

Généralités

Initialisation

Opérateurs de variation

Fitness

Modularité

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)

Overview

Généralités

Initialisation

Opérateurs de variation

Fitness

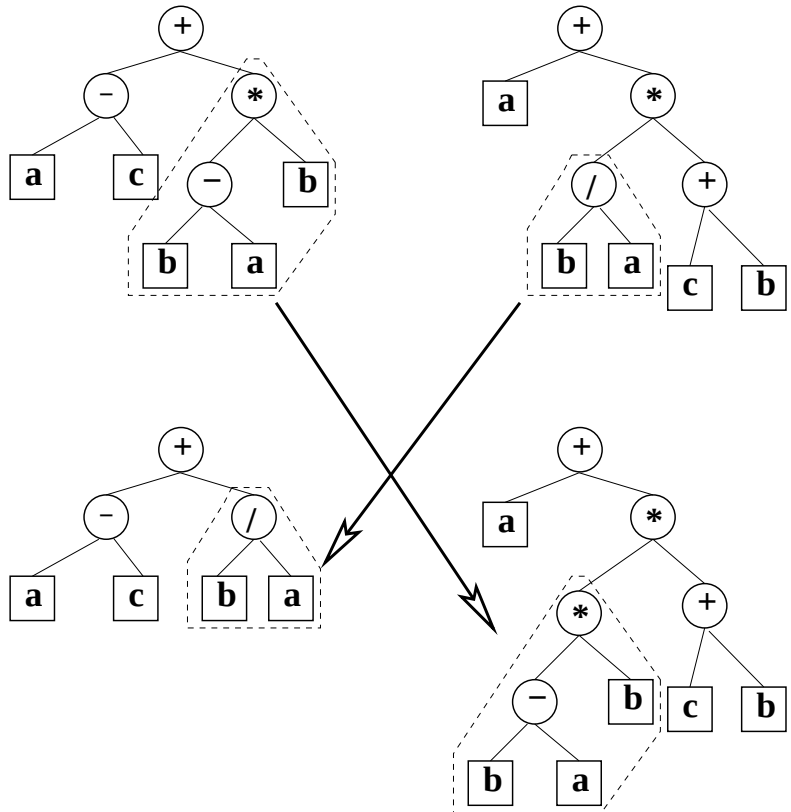
Modularité

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

Overview

Généralités

Initialisation

Opérateurs de variation

Fitness

Modularité

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

$$\approx \begin{cases} \text{Résolution de contraintes} \\ \text{Co-évolution des arbres et des cas} \end{cases}$$

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

Overview

Généralités

Initialisation

Opérateurs de variation

Fitness

Modularité

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.

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

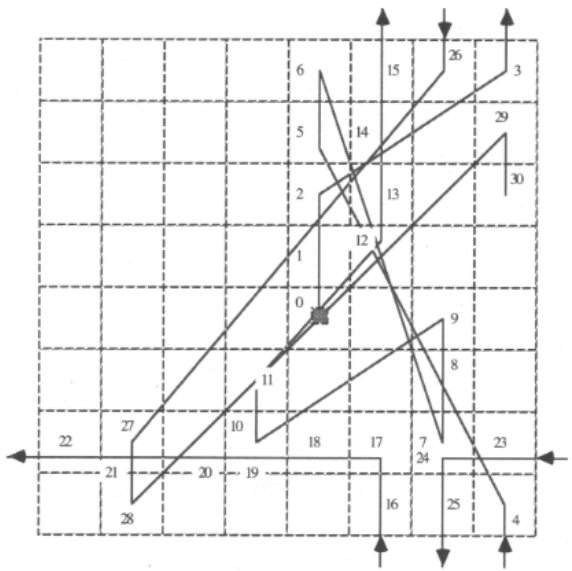
pelouse de 64 cases

population de 1000 individus

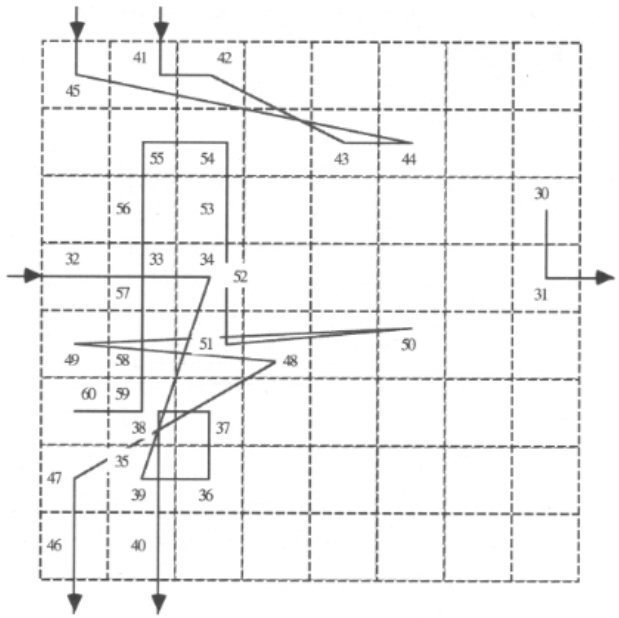
convergence en 34 générations

meilleur individu : 296 noeuds

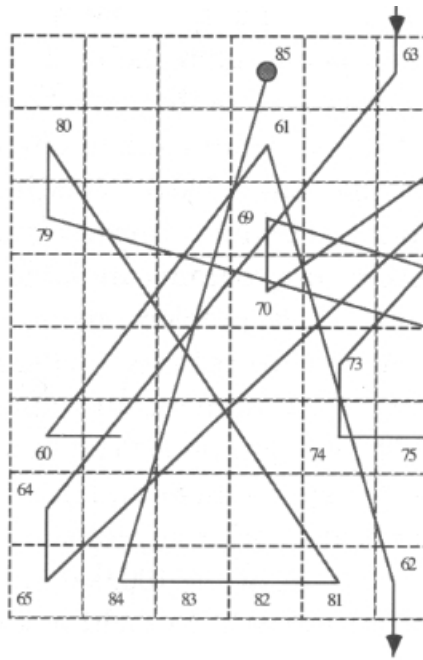
Solution, trajet (= phénotype)



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)))
```

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

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 $\mathcal{N}$ +
  $ADF_0 + ADF_1 + \ldots ADF_{i-1}$
- Main = Feuilles = $\mathcal{T}$
  Noeuds = (sous-)ensemble de $\mathcal{N}$ + ADF_K
- Fitness : évaluation de Main.

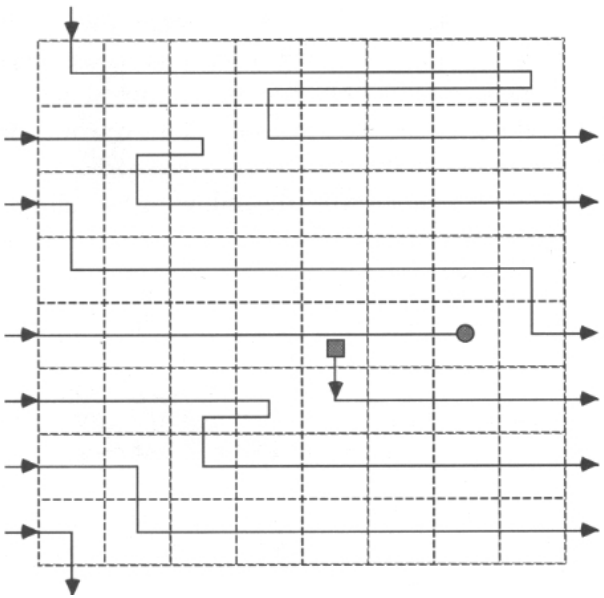
Tondeuse avec ADF

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

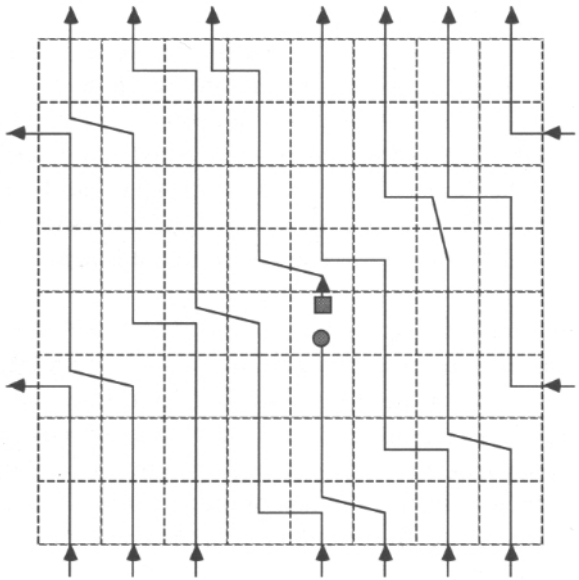
ADF_1 : $\mathcal{T} = \{ LEFT, TOND, \mathcal{R}, arg_1\}$
$\mathcal{N} = \{ V+, Proc, FROG, ADF_0\}$

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

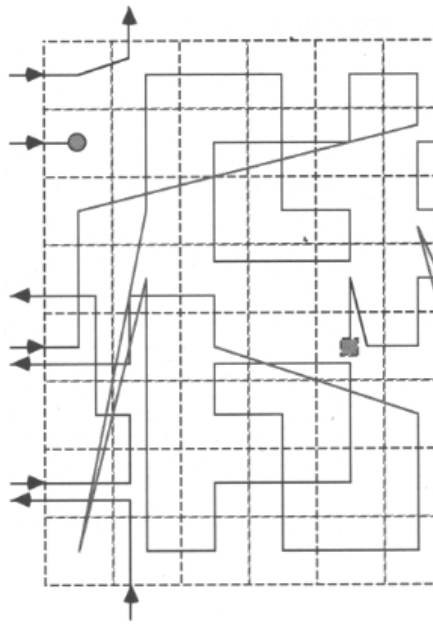
Solution avec ADF, trajet (= phénotype)



Run 1

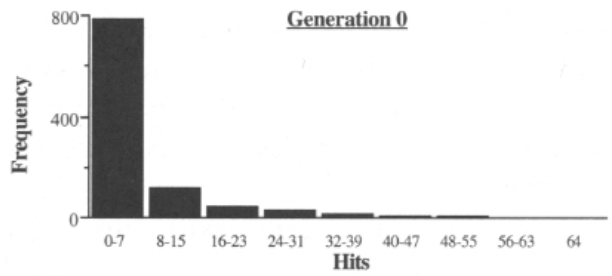


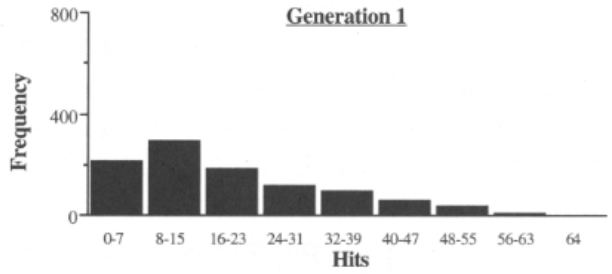
Run 2

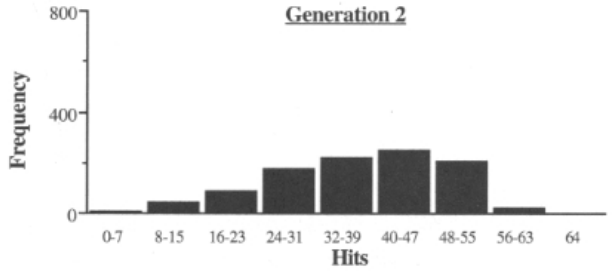


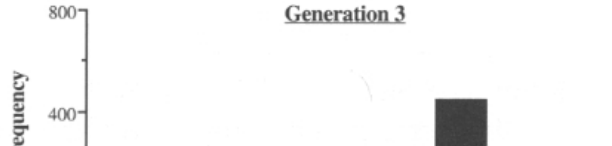
Run 3

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