

Apprentissage de comportement comme résultat de l'interaction avec la
dynamique de l'environnement

basé sur l'article de Sejal Kamani [Koza 1995] « *Behavior Learning and Individual
Cooperation in Autonomous Agents as a Result of Interaction Dynamics with the Environment*

»

Emmanuel PIERRE
epierre@e-nef.com

PRÉSENTATION DE L'ARTICLE

INTRODUCTION

Cet article est basé sur la reprise de l'article « A case study in the behavior-oriented design of autonomous agents » de Luc Steels [Steels 94] réimplémenté dans l'objectif de la Programmation Génétique dans l'article « behavior learning and individual cooperation in autonomous agents as a result of interaction dynamics with the environment » de S.Kamani [Koza 95].

Il s'agit de l'étude de l'adaptation d'un système multi-agents dans un système dynamique par Programmation Génétique (Genetic Programming).

Il y a encodage de schémas de motivation pour Steels dans ses robots, et cet article veut présenter l'apprentissage de ces schémas par les GP et le calcul de Fitness.

La coopération n'est pas explicitement programmée comme dans le robot de Luc Steels, mais doit émerger du comportement des agents à partager et agir de façon responsable dans un système dynamique.

PROGRAMMATION GÉNÉTIQUE

Elle est dérivée des Algorithmes génétiques, utilisant plutôt des programmes informatiques que des vecteurs fixes de chaînes ou structures.

La population initiale est composée de programmes aléatoires représentés sous formes d'arbres, la population est accouplée en utilisant le principe de mutation et de crossover, avec sélection par tournoi. La sélection se fait par le calcul de la Fitness sur un certain nombre de tours (*runs*).

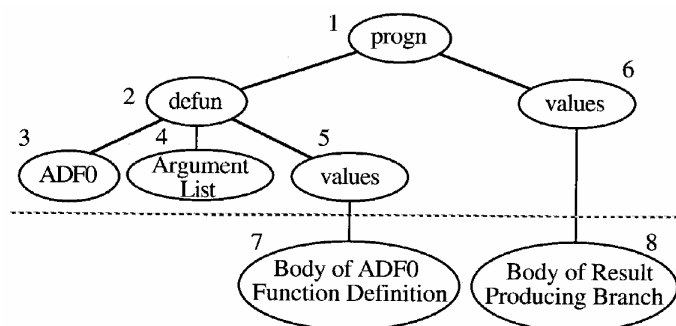
Tableau récapitulatif des valeurs les plus couramment prises par un GP:

Probabilité de crossover	90,00%
Probabilité de reproduction	10,00%
Probabilité de choix interne de points pour crossover	90,00%
Taille maximum du programme crée	17
Taille maximum pour les programmes aléatoires initiaux	6
Probabilité de mutation	0,00%
Probabilité de permutation	0,00%
Fréquence d'édition	0
Probabilité d'encapsulation	0,00%

Condition de décimation	Null
Pourcentage de décimation cible	0,00%
Méthode de génération initiale aléatoire	Rampe 1/2 1/2 (depth-grow)
Méthode de sélection basique	Tournoi par groupe de 7
Méthode de sélection d'appariement	Tournoi par groupe de 7
Fitness ajustée	Non utilisée
Sur-sélection	Non utilisée
Stratégie élitiste	Non utilisée
Valeur aléatoire (si besoin) dans la création de la fitness	Fixée une fois pour toutes
Dans le crossover préservatif, assignation des types aux points non variants	Typage de branche

Plus spécifiquement, divisé en

- Result Production Branch (RPB): c'est la partie principale (programme principal)
- Automatically Defined Functions (ADF): c'est une fonction qui évolue dynamiquement pendant un fonctionnement d'un GP, appelé par un programme (RPB) qui est évolué en même temps.

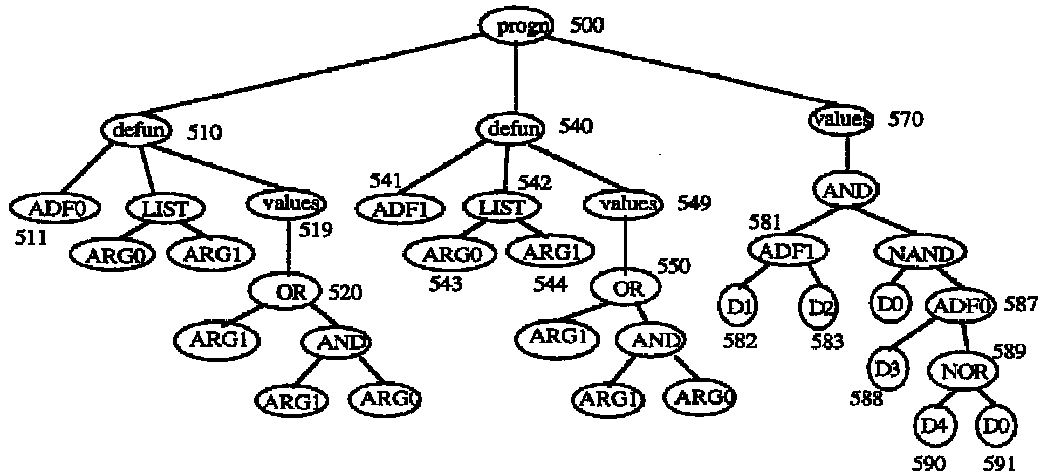


L'intérêt des ADF est qu'ils permettent:

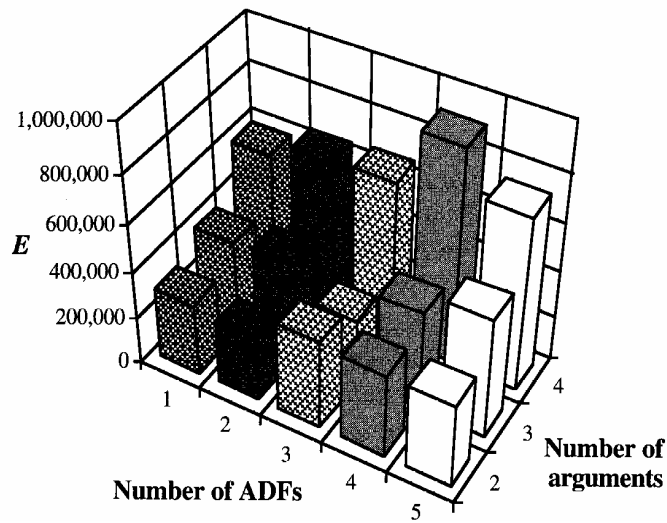
1. la décomposition de problèmes en sous-problèmes, puis réassemblage en solution du problème général
2. utilisent les régularités, symétries, homogénéités, similarités, patterns et modularisation de l'environnement
3. moins de calcul (car sous-problèmes)
4. la solution trouvée est de taille réduite

Toute fonction (ADF) peut accepter le retour d'autres fonctions ainsi que des terminaux (satisfaction de clôture (*closure*)).

La Fitness doit être complètement définie (pour tous les cas de figures possibles). Elle est calculée d'après les conséquences du fonctionnement du programme.



Un ADF peut avoir plusieurs arguments. D'après Koza, il y a un lien entre le nombre d'ADF, et d'arguments dans la rapidité de solvabilité d'un problème.



Particularités générales d'un GP:

- ensemble de terminaux
- ensemble de fonctions primitives
- mesure de Fitness
- paramètre de contrôle de fonctionnement
- méthode pour désigner le résultat et critère d'arrêt de fonctionnement.

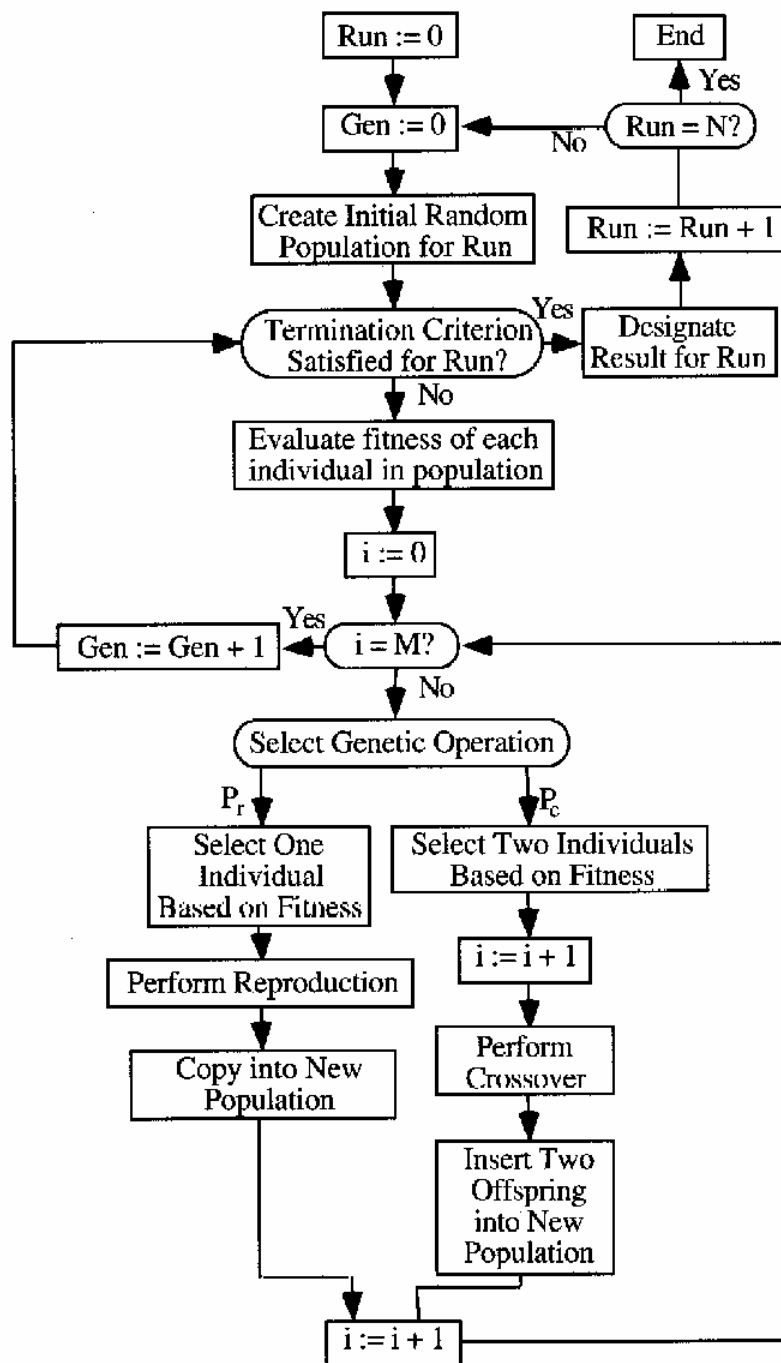


Figure 2.5 Flowchart for genetic programming.

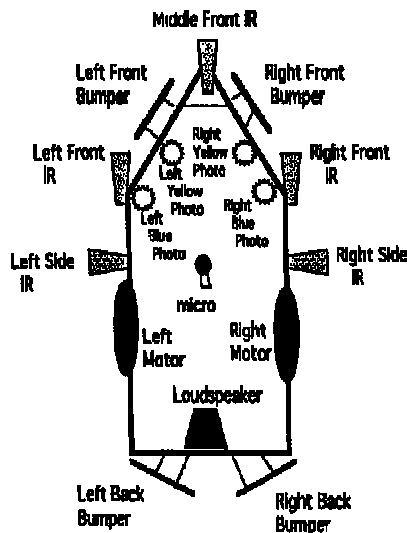
SYSTÈME PHYSIQUE

Le système est composé de deux ou plusieurs robots, comportant chacun un moyen d'autonomie (batteries rechargeables), dotés de 6 senseurs, moteur, actuateurs.

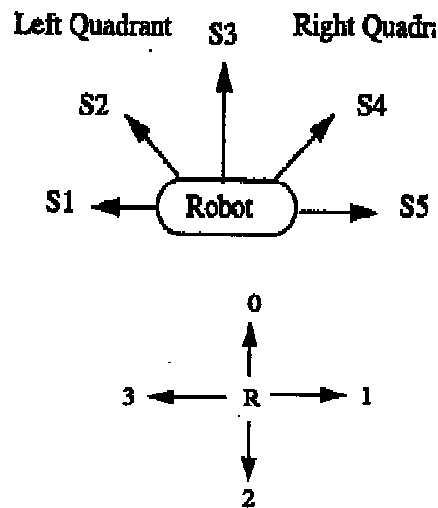
Le terrain comporte des lampes et une station de chargement (*Charging Station*) prenant leur énergie de la même source.

Le système n'a plus d'énergie si toutes les lampes sont complètement chargées, et donc la Station de Chargement est vide.

Les robots heurtant les lampes les déchargent, rechargeant d'autant la Station de Chargement, où les robots peuvent se recharger.



Robot utilisé par Steels



Equivalent en GP

MONDE SIMULÉ

Le monde simulé est toroïdal de 10 cases sur 10, composé de 3 lampes, 1 station de recharge.

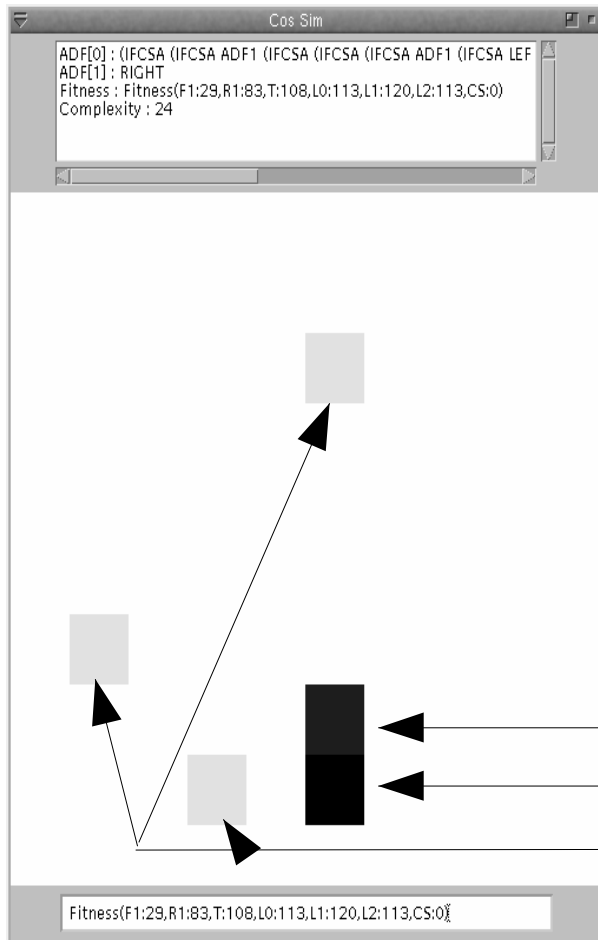
Les lampes initialement ont 12 unités d'énergie, s'accroissent de 2 par unité de temps (*timestep*) jusqu'à 30, et sont déchargées de 7 si heurtées par un robot.

Les unités originelles sont multipliées ici par 4 pour mettre en correspondances avec la simulation présentée dans la 2ème partie.

Les robots ont 80 unités d'énergie maximale, disponible initialement, perdent 1 par mouvement et peuvent se recharger de 6 unités par unité de temps à la Station de

Chargement.

Ils ont une connaissance interne de leur état et de celui du système par des fonctions en GP.



Simulateur

En Java 1.1 basé sur la librairie gpsys-1.1 de Adil QURESH (University College London)

```
> java gpsys.cos.CosSim fichier  
RandomSeed population  
génération
```

Charging Station (Blue photoaxis)
Robot
Lampes

FITNESS

Raw Fitness: temps vécu par le système + nombre de robots en vie + nombre de lampes pas complètement au maximum

Cette fonction sert à forcer le robot à ne pas avoir un comportement qui laisse le système déperir (perdre son énergie).

Le système est arrêté si un des robots meurt ou si toutes les lampes sont au maximum. Un temps de 100 unités de temps est autorisé.

La raw fitness maximale vaut donc 105.

PARAMÈTRES DU GP

Terminaux RPB	LEFT, RIGHT, FORWARD, BACK
Fonctions RPB	IFSYSUNSAFE, IFCHARGED, IFAL, IFCSA, IFLGR, IFDIE, IFOBA
Terminaux ADF	LEFT, RIGHT, FORWARD, BACK, HALT
Fonctions ADF	IFACS, IFCHARGED, IFSYSUNSAFE
Paramètres	M=10 000, G=100

DESCRIPTION DES OPÉRATEURS

- LEFT, RIGHT, FORWARD, BACK, HALT: terminaux de mouvements
- IFSYSUNSAFE: Si l'énergie du système est inférieure à une valeur fixée; renvoie un booléen.
- IFCHARGED: Si le robot est chargé à son maximum; renvoie un booléen.
- IFAL: Si le robot est devant une lampe (direction correspondante); renvoie un booléen
- IFCSA: Si la Station de Chargement est devant; renvoie un booléen.
- IFLGR: Si la luminosité est plus forte à droite qu'à gauche; renvoie un booléen.
- IFDIE: Si l'énergie du robot est inférieure à ROBOT_SAFE (constante); renvoie un booléen.
- IFOBA: Si il y a un obstacle devant le robot; renvoie un booléen.
- IFACS: Si la station de chargement est devant; renvoie un booléen.

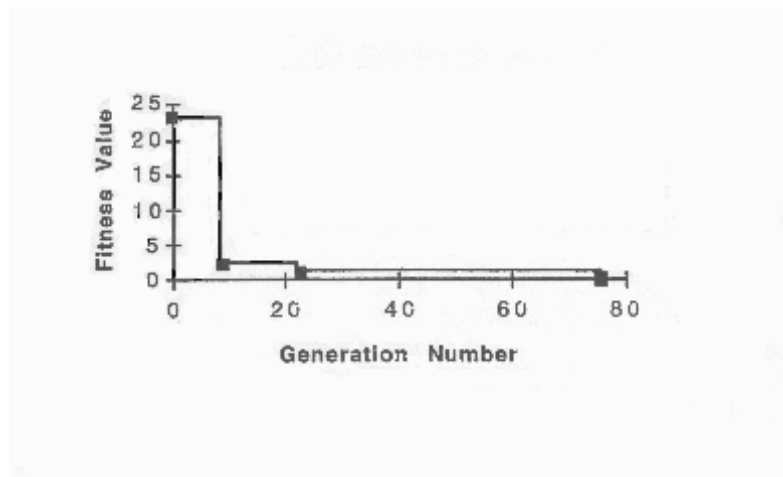
RÉSULTATS

Dans le cas de deux robots, l'algorithme semble converger rapidement si l'on en croit le graphe donné dans l'article (Figure 3), avec comme solution optimale:

Robot1: (IFAL LEFT ADF0)

Robot2: (IFCSA ADF0 ADF0)

ADF0: (IFSYSUNSAFE FORWARD RIGHT)



L'auteur signale que la résolution avec un seul RPB est aussi possible, c'est ce qui est tenté plus loin. Mais de si beaux résultats ne sont pas trouvés.

D'ailleurs un opérateur cité dans un des exemples (IFAVLENDRCR) apparaît sans jamais n'avoir été défini !

La simulation suivante a été faite avec un seul RPB et un seul ADF, faute de n'avoir pas trouvé de description sur la méthode pour évaluer deux RPB simultanément; ceci aussi bien pour les opérations de mutation, que pour les opérations de calcul de fitness qui suppose une interactivité des robots, et donc une évaluation simultanée des deux arbres !

Un certains nombres de paramètres n'ont pas été rappelés par l'auteur, ce qui rend d'autant plus la reproduction de sa simulation difficile. On peut supposer qu'il reprend les paramètres donnés par défaut par Koza:

Probabilité de crossover	90,00%
Probabilité de reproduction	10,00%
Probabilité de choix interne de points pour crossover	90,00%
Taille maximum du programme crée	17
Taille maximum pour les programmes aléatoires initiaux	6
Probabilité de mutation	0,00%
Probabilité de permutation	0,00%
Fréquence d'édition	0
Probabilité d'encapsulation	0,00%
Condition de décimation	Null
Pourcentage de décimation cible	0,00%
Méthode de génération initiale aléatoire	Rampe 1/2 1/2 (depth-grow)

Méthode de sélection basique	Tournoi par groupe de 7
Méthode de sélection d'appariement	Tournoi par groupe de 7
Fitness ajustée	Non utilisée
Sur-sélection	Non utilisée
Stratégie élitiste	Non utilisée
Valeur aléatoire (si besoin) dans la création de la fitness	Fixée une fois pour toutes
Dans le crossover préservatif, assignation des types aux points non variants	Typage de branche

La simulation effectuée ci-après ajoute des améliorations dans la recherche d'un comportement optimal en ajoutant un taux de mutation.

SIMULATION

PARAMÈTRES AJOUTÉS

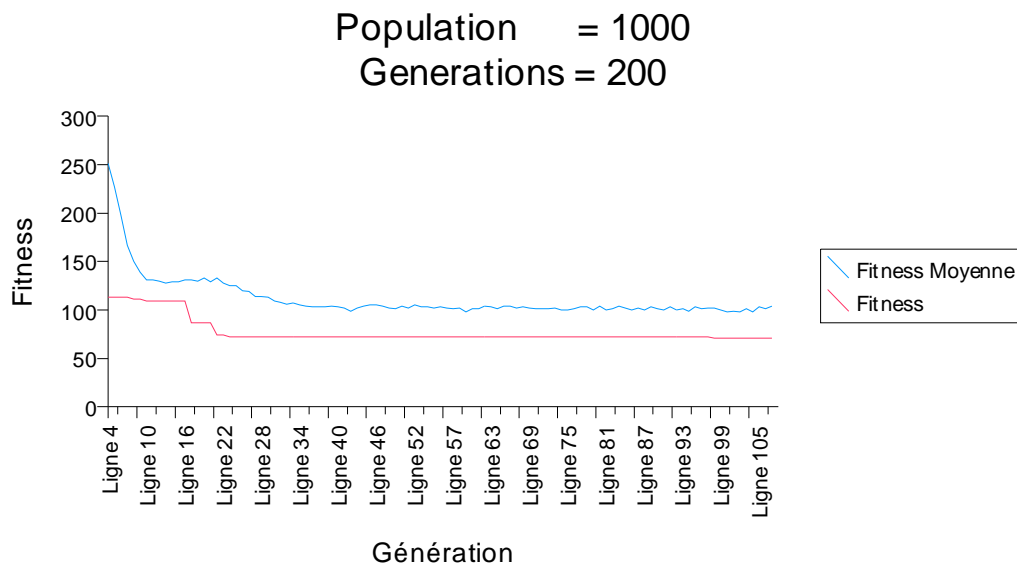
Les paramètres suivants ont été ajoutés pour la simulation. Ils ne sont pas explicitement donnés dans l'article et ont été utilisés ici.

Taille Tournoi	7
Mutation	5,00%
Arbre	<i>taille 9 maximum</i> <i>5 maximum à la création</i> <i>3 maximum pour le croisement</i>

Le temps a été étendu à 300 au lieu de 100 pour avoir une meilleure simulation du comportement du couple (robot, système).

N.B. Le répertoire best contient toutes les simulations présentées ci dessous.

Invoquer `java gpsys.cos.CosSim NomDeSim` pour relancer la simulation sauvegardée.



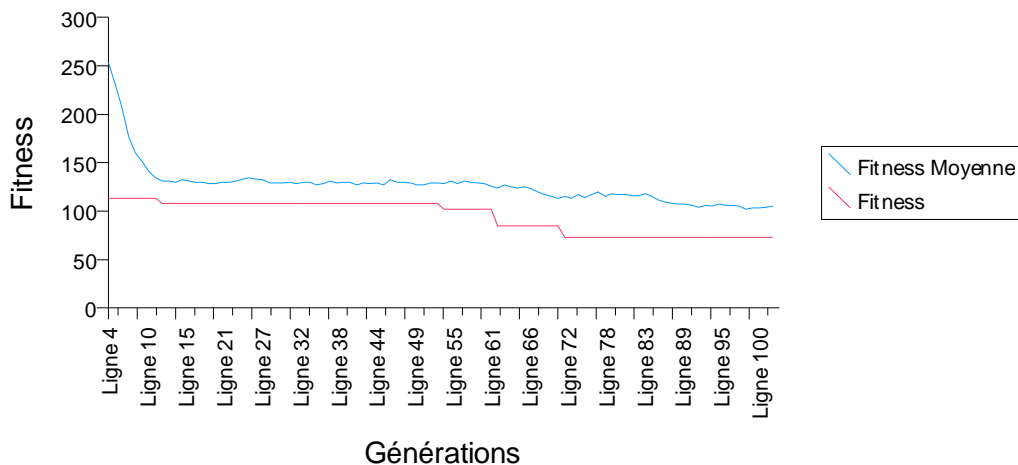
RPB : (IFSYSUNSAFE (IFCHARGED (IFDIE HALT FORWARD) (IFCHARGED (IFCSA (IFCHARGED ADF1 FORWARD) FORWARD) (IFDIE HALT (IFAL LEFT RIGHT)))) (IFSYSUNSAFE (IFSYSUNSAFE (IFCHARGED (IFSYSUNSAFE

(IFCHARGED (IFSYSUNSAFE RIGHT HALT) (IFSYSUNSAFE (IFDIE LEFT FORWARD) (IFLGR LEFT HALT))) (IFCHARGED FORWARD FORWARD) (IFCHARGED BACK (IFDIE HALT (IFAL LEFT RIGHT))) (IFSYSUNSAFE (IFCSA LEFT LEFT) (IFCHARGED (IFLGR (IFDIE HALT FORWARD) BACK) (IFSYSUNSAFE (IFDIE HALT HALT) (IFCSA (IFLGR FORWARD BACK) (IFCSA LEFT LEFT)))))) (IFCHARGED (IFLGR FORWARD (IFCHARGED (IFLGR FORWARD BACK) (IFSYSUNSAFE (IFCSA (IFCHARGED ADF1 FORWARD) (IFCHARGED ADF1 FORWARD)) (IFCSA (IFLGR FORWARD BACK) (IFCSA LEFT LEFT)))))) (IFSYSUNSAFE (IFLGR FORWARD BACK) (IFCSA (IFLGR FORWARD BACK) (IFCSA LEFT LEFT))))))

ADF1 : RIGHT

Fitness : Fitness(F1:71,R1:26,T:277,L0:120,L1:120,L2:110,CS:0)

Population = 1000
Génération = 100

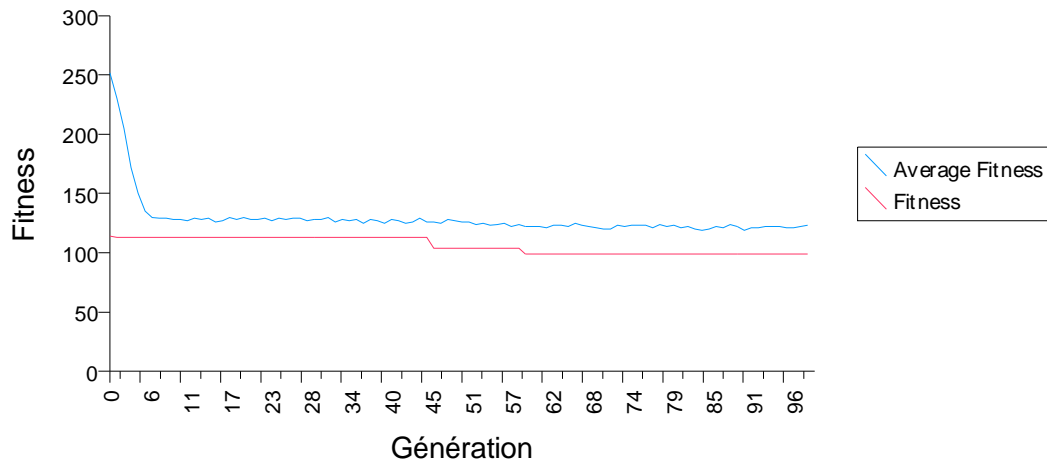


RPB : (IFSYSUNSAFE (IFSYSUNSAFE (IFDIE (IFLGR FORWARD (IFAL HALT (IFCSA ADF1 (IFOBA (IFAL BACK ADF1) (IFOBA HALT RIGHT)))))) (IFAL FORWARD (IFCSA FORWARD FORWARD))) FORWARD) (IFCHARGED BACK (IFCSA FORWARD RIGHT)))

ADF1 : (IFSYSUNSAFE RIGHT HALT)

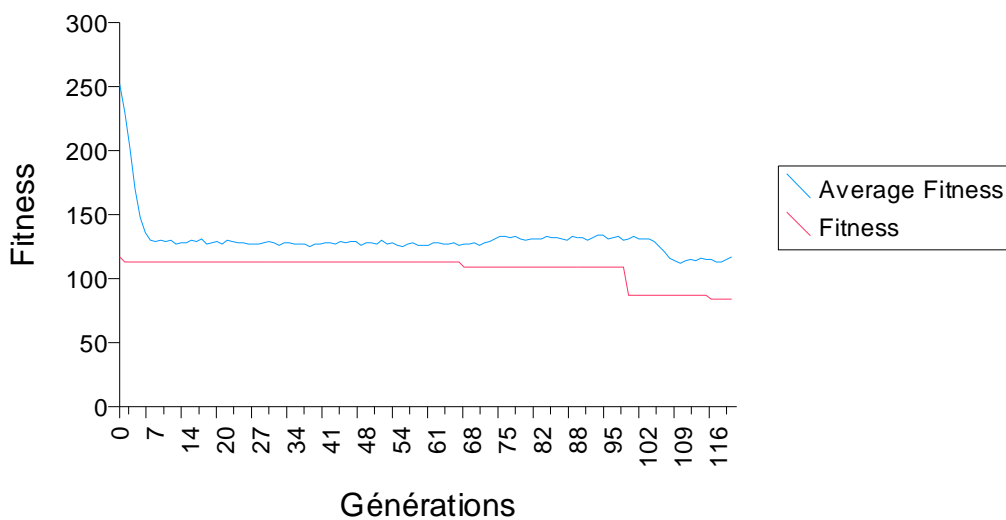
Fitness : Fitness(F1:73,R1:26,T:236,L0:120,L1:120,L2:120,CS:0)

Population=1000
Génération=100



RPB : (IFDIE (IFOBA (IFOBA (IFLGR HALT LEFT) (IFDIE (IFOBA (IFLGR HALT LEFT) (IFCSA (IFCSA FORWARD FORWARD) (IFOBA LEFT LEFT)))) (IFSYSUNSAFE (IFCSA FORWARD FORWARD) HALT))) (IFCSA (IFCSA FORWARD FORWARD) (IFOBA LEFT LEFT))) (IFSYSUNSAFE FORWARD HALT))
 ADF1 : BACK
 Fitness : Fitness(F1:99,R1:24,T:250,L0:120,L1:120,L2:120,CS:0)
 Complexity : 34

Population = 1000
Génération = 120



ADF[0] : (IFCHARGED (IFCHARGED (IFSYSUNSAFE (IFCSA (IFSYSUNSAFE FORWARD HALT) RIGHT) (IFSYSUNSAFE FORWARD HALT)) LEFT) (IFDIE RIGHT (IFSYSUNSAFE (IFSYSUNSAFE FORWARD HALT) (IFCSA (IFDIE LEFT LEFT) (IFLGR RIGHT RIGHT))))))

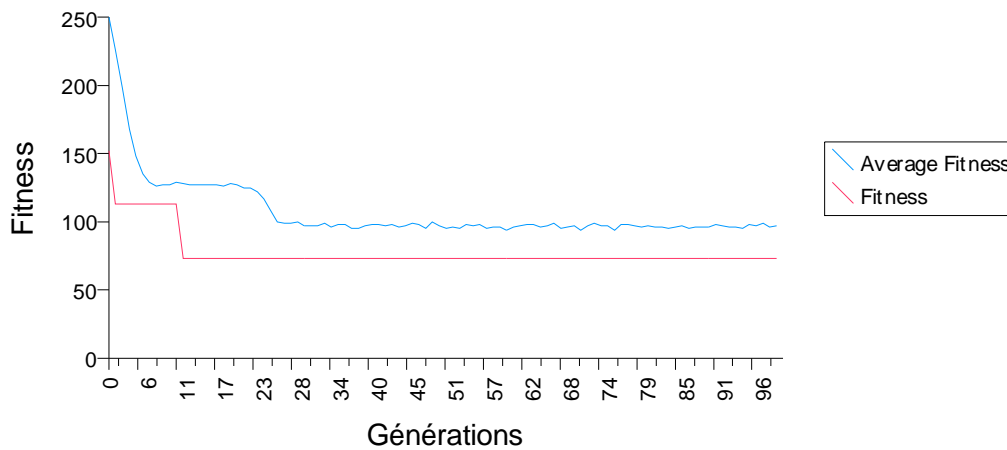
ADF[1] : FORWARD

Fitness : Fitness(F1:84,R1:0,T:150,L0:120,L1:92,L2:120,CS:0)

Complexity : 26

NOTA: Cet exemple va à l'encontre de l'article puisqu'il présente des résultats très appréciables au delà de la génération 76 qui est décrite comme maximale dans l'article !

Population = 1000
Génération = 100



ADF[0] : (IFSYSUNSAFE (IFSYSUNSAFE (IFDIE (IFOBA (IFLGR ADF1 LEFT) RIGHT) FORWARD) (IFSYSUNSAFE BACK ADF1)) (IFSYSUNSAFE (IFSYSUNSAFE (IFSYSUNSAFE (IFDIE (IFOBA (IFLGR ADF1 LEFT) RIGHT) FORWARD) (IFLGR (IFSYSUNSAFE ADF1 BACK) (IFSYSUNSAFE BACK ADF1))) (IFSYSUNSAFE (IFAL ADF1 LEFT) (IFCSA FORWARD ADF1))) (IFSYSUNSAFE RIGHT (IFCSA FORWARD ADF1))))

ADF[1] : (IFACS (IFACS (IFCHARGED BACK BACK) RIGHT) (IFSYSUNSAFE HALT RIGHT))

Fitness : Fitness(F1:73,R1:25,T:275,L0:120,L1:97,L2:120,CS:0)

Complexity : 50

