Utilisation d'un algorithme génétique pour la mise au point d'un contrôleur neuronal pour robot mobile

Jean-Baptiste Mouret
*jb@lrde.epita.fr**

30 décembre 2003

Résumé- *Ce rapport décrit le résultats de nos experiences sur l'utilisation d'un algorithme génétique pour la mise au point d'un contrôleur neuronal pour robot mobile. Ce dernier doit être capable de se déplacer sur le terrain de la Coupe de France de Robotique en évitant les murs et des petits poteaux disposés aléatoirement. Le robot est doté de télémètres infrarouges.*

Un réseau de neurones simple inspiré des véhicules de Braitenberg est d'abord utilisé sur un terrain simplifié afin de vérifier la validité de notre approche. Puis l'experience est reconduite en utilisant un réseau de neurones récurrents et l'ensemble des obstacles potentiels.

Les choix relatifs à la conception du réseau de neurones et à l'utilisation d'un algorithme génétiques sont detaillés et les résultats de nos experiences commentés.

Table des matières

*Ce travail a eut lieu dans le cadre de l'Epita et du LRDE (Laboratoire de Recherche et Développement de l'Epita), pour l'association Evolutek

1 Introduction

L'approche Animat En 1984, Braitenberg [Bra84] publie son court mais maintenant célèbre livre *Vehicles : Experiments in Synthetic Psychology*. Il y décrit des expériences dans lesquelles des créatures artificielles simples ont des comportements intéressants, apparemment intentionnels, comme la recherche de lumière ou l'évitement d'obstacles. En partant d'un véhicule particulièrement simple avec un unique capteur de lumière qui commande un moteur, les possibilités de ces «créatures» sont progressivement améliorées par l'addition de roues, de capteurs et de connexions entre les différents éléments. La figure 1 présente ainsi un robot inspiré du véhicule de Braitenberg capable d'éviter les obstacles. L'idée sous-jacente à ce travail est qu'un comportement jugé complexe et intentionnel peut être lié à un système extrêmement simple.

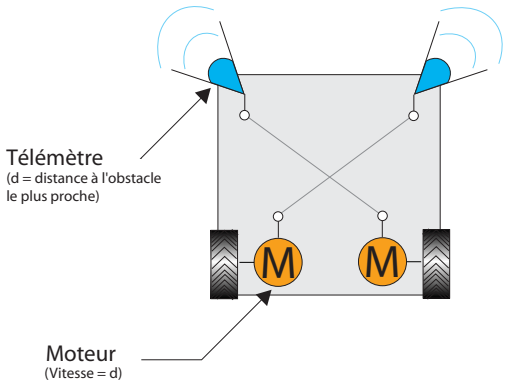

Figure 1: Véhicule inspiré du véhicule de Braitenberg 3c, capable d'éviter les obstacles

Ce principe est à rapprocher de l'approche prônée par Rodney Brooks [Bro86, Bro91] qui affirme qu'un système artificiel doit se concevoir comme un système entier qui se comporte dans un environnement réel. Il rejette ainsi l'approche «traditionnelle» de l'intelligence artificielle, dite «descendante», dans laquelle le système organise de façon centralisée les comportements, pour une approche «ascendante» dans laquelle ce sont les comportements activés en parallèle et interagissant avec l'environnement qui font émerger leur propre cohérence. Ces robots n'utilisent pas de représentation interne de leur environnement, pour paraphraser Brooks : la meilleure représentation du monde, c'est lui-même. L'approche ascendante a mené à ce que l'on appelle aujourd'hui l'approche Animat [Mey95].

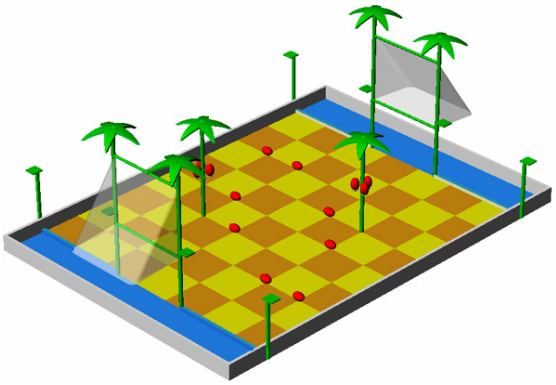

Figure 2: Aperçu du terrain utilisé lors de la Coupe de France de Robotique

Les caractéristiques des Animats en font des créatures particulièrement adaptées à la «survie» dans des situations nouvelles. En effet, en suivant l'approche classique de l'intelligence artificielle, le concepteur a toujours une bonne connaissance *a priori* des événement auxquels les robots vont être confrontés. Cependant, à cause de l'attribution trop «humaine» de connaissances, ces robots s'adaptent très mal à des situations nouvelles. Par opposition, le simple robot de Braitenberg de la figure 1 est capable, malgré sa simplicité, de réagir correctement dans la plupart des cas sans que l'on ait eut besoin de spécifier explicitement la réaction adaptée à chacune des situations.

Utilisation dans le cadre de la Coupe de France de robotique La capacité pour un robot à réagir devant des situations inconnues, dans un environnement dynamique est une caractéristique très utile dans le cadre de la Coupe de France de robotique. Pour l'édition 2004, quatre robots formant deux équipes doivent s'affronter sur une table de 3×2.1 mètres de cotés avec pour objectif le ramassage puis l'utilisation pour marquer des buts de petites balles de rugby. La figure 2 est un aperçu du terrain sur lequel s'affrontent les robots. Afin de complexifier la tâche des robots, deux poteaux appelés «palmiers» sont placés aléatoirement sur la piste de jeu.

Ces éléments au placement inconnu a priori et le comportement imprévisible de l'équipe adverse rend nécessaire la conception d'un robot particulièrement robuste, c'est à dire capable de réagir de manière adaptée – mais pas forcément optimale – à un maximum de situations. La durée de jeu courte – 1 minute et 30 secondes – et la vitesse élevée de dépla-

cement des robots [1] justifient eux aussi l'inutilité de plans. Il apparaît en effet plus adapté de concevoir un robot qui survit dans un maximum de situations qu'un robot qui cherche à planifier le comportement optimal ; dans ce dernier cas le manque d'information sur l'environnement et ses potentiels changements ont en effet de fortes chances de rendre tout plan inefficace.

Notre approche S'inspirant du véhicule de Braitenberg de la figure 1, nous commençons par le décrire en utilisant le formalisme des *réseaux de neurones formels*. L'objectif d'obtenir le robot le plus robuste possible mène au problème du choix optimal des poids associés aux connexions entre neurones. Cet objectif étant très difficile à exprimer sous la forme d'une fonction dérivable, ces poids sont calculés à l'aide d'un *algorithme génétique*. Notons de plus qu'il est peu aisé d'avoir des connaissances a priori sur les poids idéaux. L'évolution de réseaux de neurones à l'aide de ces algorithmes a été largement étudiée, [Yao99] en présente un aperçu.

Cette expérience simple sur un véhicule de Braitenberg, précédemment menée [MF95, MLN95], nous permet de mettre au point dans un cadre simple les fonctions d'adaptation capables de mener à l'obtention d'un contrôleur robuste.

Les capacités du réseau sont ensuite augmentées par l'ajout de connexions *récurrentes* et de neurones *cachés* afin d'être capable d'exploiter une information ponctuelle correspondant à la présence d'un palmier. La topologie du réseau utilisé est très proche des *Continuous Time Recurrent Neural Network (CTRNN)*. Un contrôleur doté d'un tel réseau et de onze capteurs de distance est ainsi évolué et les résultats obtenus sont commentés. Dans ce cas, l'algorithme génétique a pour mission de maximiser une fonction d'adaptation – la capacité à éviter les obstacles – non dérivable et dépendant d'environ 300 variables.

Ce travail tire la plupart de ses bases de [MF95, MLN95], cependant nos expériences montrent qu'il est possible d'utiliser dans le cadre de la Coupe de France de Robotique des fonctions d'adaptation plus simples et moins directives que celles suggérées par la littérature.

[1] La vitesse de déplacement de beaucoup de robots participants est de l'ordre du $m.s^{-1}$.

2 Évolution d'un véhicule de Braitenberg

2.1 Formulation neuronale et motivation

Les essais en simulation pour tester la capacité d'adaptation et la robustesse du robot décrit sur la figure 1 se révèlent décevants. Si on le place proche d'un obstacle, il est par exemple souvent incapable de l'éviter, étant dans l'incapacité d'effectuer une marche arrière. Un comportement souhaitable serait donc que la vitesse soit négative lorsque le robot se trouve à une distance faible d'un obstacle. Réciproquement, en l'absence d'obstacle, il paraît intéressant que le robot essaye d'aller le plus vite possible. On souhaite par conséquent utiliser une *fonction de transfert* qui permette de transformer le signal renvoyé par le capteur en une vitesse pour le moteur.

Cette fonction devra être continue et lisse, la valeur maximale de sa dérivée correspondant à l'accélération maximum du robot.

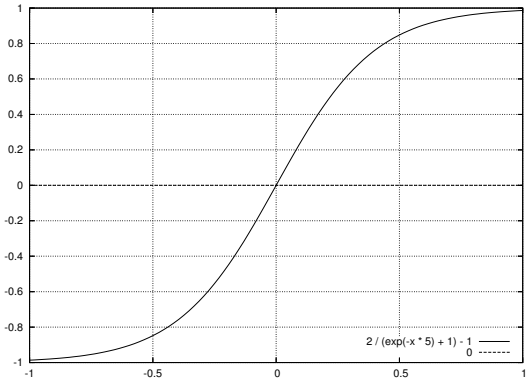

Figure 3: Sigmoide $1 - \frac{2}{1+\exp(-\lambda x)}$

Une bonne candidate est la fonction *sigmoide* (Figure 3), que l'on peut exprimer sous la forme $\varphi(x) = 1 - \frac{2}{1+\exp(-\lambda x)}$, où λ désigne un réel permettant de choisir la pente maximale de la sigmoide. Si on ramène les sorties x des capteurs dans l'intervalle $[-1; 1]$, alors $\varphi(x)$ vaut -1 (marche arrière) si x est proche de -1 et 1 (vitesse maximale en avant) si x est proche de 1.

Une autre amélioration serait la gestion de plusieurs capteurs de distances (télémètres). On peut noter que l'utilisation d'une fonction de transfert de type sigmoide implique la possibilité que les moteurs soient simultanément arrêtés, la fonction valant 0 en $x = 0$. Ainsi, lorsque les deux télémètres

renvoient 0, le robot ne se déplace plus. Ce ne serait pas un problème si cet état n'était qu'un état de transition, par nature instable.

Supposons que $x_l(t) = 0$ et $x_r(t) = 0$, et écartons légèrement le robot vers la droite. Le capteur du coté gauche est alors plus près de l'obstacle et on a $x_l(t+1) = x_l(t) - \varepsilon$ et $x_l(t+1) = x_l(t) + \varepsilon$. Il suffit à présent de calculer les vitesses des moteurs en appliquant la fonction $\varphi(x)$:

$$\begin{aligned} v_l(t+1) &= \varphi(-\varepsilon) \\ v_r(t+1) &= \varphi(\varepsilon) \end{aligned}$$

Puisque le moteur gauche recule et le droit avance, le robot se déplace vers la gauche et retrouve ainsi son état stable. Pour éviter ces états, on peut utiliser deux capteurs de chaque coté du robot et les placer de telle manière à ce qu'il soit très difficile pour le robot d'être dans une configuration à la fois symétrique et pour laquelle la vitesse de ses moteurs est nulle.

Une solution simple pour exploiter plusieurs télémètres est de remplacer le capteur gauche du robot de la figure 1 par la moyenne de tous les capteurs du coté gauche et réciproquement pour le coté droit. Il paraît naturel d'effectuer une moyenne pondérée, tous les capteurs n'ayant pas nécessairement la même importance.

On peut à présent calculer les vitesses des moteurs v_r et v_l en fonction des valeurs renvoyées par les capteurs gauches $x_{l,1} \cdots x_{l,n}$ et droits $x_{r,1} \cdots x_{r,n}$:

$$\begin{aligned} v_r &= \varphi\left(\sum_{i=1}^{n} w_i x_{r,i}\right) \\ v_l &= \varphi\left(\sum_{i=1}^{n} w_i x_{l,i}\right) \end{aligned}$$

où w_i représentent le poids de chaque capteur dans le calcul de la vitesse du moteur correspondant. Ces poids sont difficiles à fixer, nous verrons par la suite que nous utiliserons un algorithme génétique pour les trouver.

En ajoutant des connexions de manière à ce que tous les capteurs soient connectés à tous les moteurs, on obtient exactement la formulation classique d'un réseau de neurone à une couche : un perceptron sans couche cachée. Ce réseau est montré sur la figure 4.

Cette reformulation permet de gagner un formalisme et d'apporter de nombreuses nouvelles idées. En effet, les possibilités et limitations des réseaux

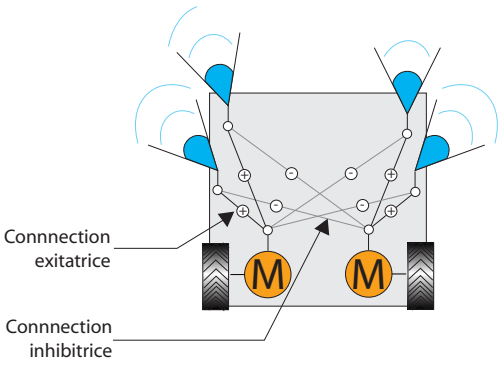

Figure 4: Contrôleur neuronal inspiré du véhicule de la figure 1

de neurones ont été largement étudiées[2] depuis une vingtaine d'années. Il existe de plus une littérature abondante autour de l'utilisation combinée de réseaux de neurones et d'algorithmes évolutionnaires de laquelle il est possible de s'inspirer pour améliorer les résultats de notre robot.

2.2 Dispositif expérimental

On se propose dans un premier temps de tenter de trouver de bons poids pour le réseau de neurones décrit sur la figure 4. Un bon ensemble de poids est un réseau permettant au robot de se déplacer le plus vite possible en évitant les murs. Cela implique qu'il lui soit très difficile d'être bloqué dans un état stable. On souhaite de plus que le comportement du robot soit le plus robuste possible, c'est-à-dire que quel que soit l'endroit où on le positionne sur le terrain, il trouve une solution pour ne pas toucher les murs.

Le terrain sur lequel le robot devra évoluer est celui de la coupe de France de robotique (Figure 2), sans les palmiers. La difficulté principale réside donc au niveau du «trou» de chaque coté du terrain permettant d'entrer dans la zone d'embut.

L'expérience étant très simple, de nombreuses solutions donnent un bon résultat. L'algorithme génétique est ici sous-exploité, une recherche aléatoire étant capable de trouver de bons résultats. Cette première expérience a pour but de :

– mettre au point des fonctions d'adaptation simples pour résoudre notre problème ;

[2] Une étude approfondie des réseaux de neurones est disponible dans [Hay99]

– vérifier qu'un algorithme génétique est bien capable de trouver une bonne solution ;
– poser des bases saines pour les expériences sur les réseaux de neurones récurrents (Section 3) et notamment définir des opérateurs génétiques adaptés.

Sous une approche énergétique, on souhaite trouver les valeurs des poids W qui maximisent la fonction objectif. Cet optimisation devra être un compromis entre le nombre de virages, qui ralentissent le robot mais permettent d'éviter les murs, et la longueur des lignes droites, qui permettent de se déplacement plus vite. Le compromis doit aussi être présent au niveau de la vitesse de déplacement sélectionnée, une vitesse trop importante pouvant mener à l'impossibilité d'éviter les obstacles à temps.

Nous disposons d'un simulateur prenant en compte l'accélération maximum des moteurs, leur vitesse et les collisions potentielles. Tous les tests que nous avons effectués l'ont été dans cet environnement simulé.

2.3 Algorithme génétique

La figure 5 rappelle le principe de base d'un algorithme génétique appliqué à notre problème. Il se déroule en cinq étapes nécessitant l'intervention d'*opérateurs génétiques* :

1. génération d'une population aléatoire ;
2. mesure de l'adaptation (*fitness*) de chacune des séquences présentes ;
3. sélection des individus les plus adaptés pour la reproduction ;
4. reproduction des individus sélectionnés, les nouveaux individus sont générés à l'aide d'un opérateur de croisement (*cross-over*) ;
5. application d'un opérateur de mutation, modifiant légèrement certains des nouveaux individus ;
6. retour à l'étape 2.

Il est maintenant nécessaire de déterminer comment coder les poids du réseau de neurone afin de constituer un génome puis de choisir des opérateurs génétiques adaptés.

2.3.1 Codage du génome

Les poids des couches des perceptrons multicouches (PMC) sont généralement stockés sous la forme d'une matrice $\mathbf{W}$ telle que w_{ij} corresponde au poids de la connexion du neurone j vers le neurone i. Ces derniers sont le plus souvent codés sous la forme d'un nombre à virgule flottante. On peut par conséquent exprimer les poids du réseau de la figure 4 sous la forme d'une matrice de nombres à virgule flottante.

Il suffit ensuite de mettre bouts à bouts chaque ligne de la matrice pour obtenir le génotype d'un individu.

2.3.2 Opérateurs génétiques

Sélection On utilise le principe de la roulette biaisée pour tirer aléatoirement les couples d'individus qui se reproduisent. Ce principe se base sur l'image d'une roulette telle que la probabilité de sélectionner un individu particulier soit proportionnelle à la valeur de sa fonction d'adaptation. Ainsi, les meilleurs individus auront une probabilité plus importante d'être sélectionnés pour la reproduction.

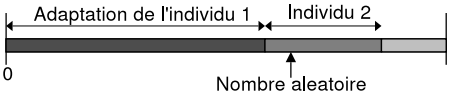

Figure 6: Principe de la roulette biaisée. Sur cet exemple, l'individu sélectionné est le deuxième.

En pratique, on se place sur une échelle dont la longueur est égale à la somme des valeurs des fonctions d'adaptation correspondant à tous les individus de la population. Cette échelle est ensuite divisée en segments mis bouts à bouts dont la longueur est proportionnelle à l'adaptation de chaque individu (Figure 6. Ainsi, à chaque segment est associé un individu. Il suffit ensuite de tirer aléatoirement un nombre compris entre 0 et la somme des valeurs des fonctions d'adaptation puis de mettre en correspondance cette valeur avec l'individu associé en utilisant l'échelle précédemment construite.

L'opération est répétée jusqu'à l'obtention de suffisamment de parents.

Croisement On emploie le croisement barycentrique. Pour générer deux enfants p'_1 et p'_2 à partir des individus p_1 et p_2, on commence par tirer un nombre α au hasard dans l'intervalle $[-0.5; 1.5]$ et on applique les formules :

$$\begin{aligned} p'_1 &= \alpha p_1 + (1-\alpha) p_2 \\ \mathrm{P}'_2 &= \alpha p_2 + (1-\alpha) p_1 \end{aligned}$$

Des expériences ont été conduites en utilisant le croisement BLX-α mais n'ont pas mené à une amélioration significative du résultat.

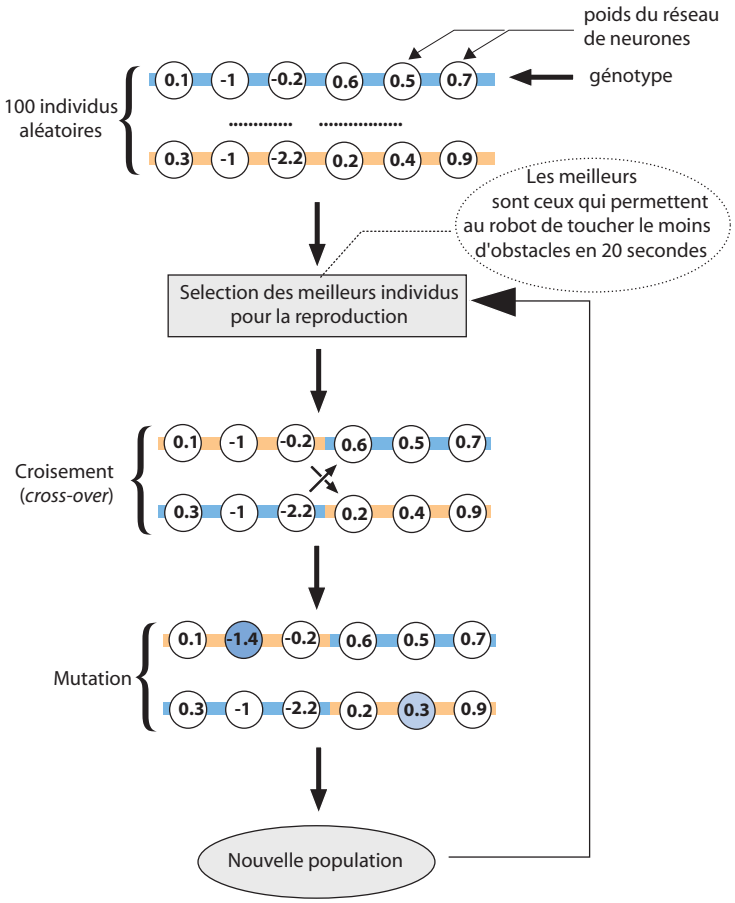

Figure 5: Principe d'un algorithme génétique

Mutation On ajoute un bruit blanc à certains poids choisis aléatoirement. Pour chaque élément de la matrice de poids, on tire un nombre aléatoire dans l'intervalle $[0;1[$ selon une loi uniforme. Si ce nombre est inférieur à une probabilité fixée γ, un autre nombre aléatoire est tiré dans l'intervalle $[-\frac{n}{2},\frac{n}{2}[$, puis ajouté à l'élément :

$$w_{ij}(t+1) = w_{ij}(t) + \mathrm{U}(-\frac{n}{2},\frac{n}{2})$$

2.3.3 Fonction d'adaptation

Principe Le principe de la fonction d'adaptation utilisée est simple : chaque individu est lancé pendant une période de temps T pendant laquelle la distance parcourue est mesurée. Lorsque le robot rencontre un obstacle, il est arrêté. Pour parcourir une distance maximale, il est donc nécessaire d'éviter ces obstacles. Afin de permettre l'émergence de robots robustes, l'expérience est répétée n fois en positionnant le robot à une position initiale aléatoire. On peut donc écrire la fonction d'adaptation sous la forme :

$$\mathrm{F} = \frac{1}{\mathrm{T}} \sum_{n} \sum_{t=0}^{\mathrm{T}} \|\overrightarrow{p(t) - p(t-1)}\| \qquad (1)$$

où $p(t)$ désigne la position du robot à l'instant t et T la durée de chaque test. Cette fonction correspond à la vitesse moyenne du centre du robot pendant la durée T.

La répétition de l'expérience n fois est requise pour obtenir des contrôleurs non sur-adaptés. Lors de nos tests, nous avons par exemple constaté l'apparition d'individus se servant uniquement des capteurs du coté gauche, car tournant toujours dans le sens des aiguilles d'une montre. Cette répétition

ne permet la survie que des individus réellement capables de gérer le maximum de situations.

Dans [MF95], F. Mondada et D. Floreano utilisent une fonction d'adaptation différente pour un problème similaire – l'évitement d'obstacle – suite à l'observation de nombreux individus exécutant un cercle. En effet, un cercle d'un rayon suffisant permet au robot de ne toucher aucun obstacle tout en se déplaçant à une vitesse raisonnable. Pour favoriser les déplacement en ligne droite, ils écrivent la fonction d'adaptation sous la forme :

$$\mathrm{F} = \frac{1}{\mathrm{T}} \sum_{\mathrm{T}} \mathrm{V}(1 - \sqrt{\Delta \mathrm{V}})(1 - i) \qquad (2)$$

où V correspond à la vitesse moyenne des roues, $\sqrt{\Delta \mathrm{V}}$ à la valeur absolue de la différence algébrique entre la vitesse des roues et i à la valeur renvoyée par le capteur avec la plus grande activité. Cette fonction est beaucoup plus «directive» que celle de l'équation (1) car elle spécifie explicitement la nécessité de faire des lignes droites et d'utiliser des capteurs.

Les tests que nous avons réalisés avec la fonction d'adaptation (2) n'ont pas révélé d'amélioration significative. Dans la suite nous utiliserons donc uniquement la fonction (1).

Énergie de régularisation Un raffinement est nécessaire, directement lié au choix de notre méthode de croisement et au fait que nous utilisions un réseau de neurones à base de sigmoides.

On peut d'abord constater que si les poids deviennent trop importants[3], la sortie du réseau dépendra uniquement du signe des entrées, sans tirer partie de la transition douce de la sigmoide. En effet, la fonction sigmoide tend rapidement vers -1 pour les valeurs négatives et vers 1 pour les valeurs positives. Imaginons par exemple un poids d'une valeur de 100, la partie «transitoire» de la sigmoide ne sera utilisée que pour approximativement des entrées comprises entre -0.01 et 0.01 ; pour toutes les autres entrées, la sortie prendra des valeurs très proches de -1 ou de 1. On décrit généralement dans ce cas le réseau comme *saturé*.

Or la méthode de croisement choisie a tendance a renforcer le nombre de poids forts dans la population. Ainsi, si par exemple $p_1 = 0.1$ et $p_2 = 6$, la méthode du croisement barycentrique donne avec $\alpha = 0.5 : p'_1 = p'_2 = 0.5p_1 + 0.5p_2 = 3.25$. Le réseau de l'enfant est aussi saturé, et p_2 risque de «contaminer» tous les individus avec lesquels il se reproduit.

[3] dans le cas de la fonction de transfert choisie, si ils sont supérieurs à environ 2

Afin de limiter ce problème, on utilise une technique souvent utilisée sur des réseaux de neurones appelée *énergie de régularisation* : on ajoute un terme à la fonction d'adaptation pénalisant les individus possédant des poids forts. Cette dernière devient alors :

$$\mathrm{F} = \frac{1}{\mathrm{T}} \sum_{n} \sum_{t=0}^{\mathrm{T}} \|\overrightarrow{p(t) - p(t-1)}\| - \sum_{i,j} w_i j^2 \qquad (3)$$

Temps d'évaluation et complexité , ∂t l'intervalle de temps entre t et $t+1$, n le nombre de tests par individus et k_{T} le nombre de télémètres.

L'évaluation de l'adaptation de chaque individu nécessite l'évaluation du réseau de neurones $\frac{\mathrm{T}}{\partial t}$ fois. Chaque évaluation nécessite environ $2k_{\mathrm{T}}$ additions, $2k_{\mathrm{T}}$ multiplications et 2 exponentielles. Il faut ajouter à cette complexité celle de la simulation, qui doit tester à chaque instant si le robot n'entre pas en collision avec les murs, simuler les télémètres – via une sorte de lancer de rayons –, simuler les moteurs et calculer la nouvelle position.

Un ordre de grandeur de T est 30 secondes. Nous avons choisi ∂t de l´ordre de 30×10^{-3} secondes. L'évaluation de l'adaptation d'un individu prend donc beaucoup de temps. On peut noter cependant qu'il est tout à fait possible d'évaluer l'adaptation de plusieurs individus en même temps et de tirer ainsi parti de machines disposant de plusieurs CPUs. A titre d'exemple, l'évaluation d'une centaine de générations nécessite environ cinq heures de calcul sur un Bi-Xeon à 2.4 Ghz.

Évolution incrémentale Afin de réduire les temps d'évaluation, on peut utiliser une évaluation incrémentale. Cette technique, couramment utilisée pour l'évolution de robots [CHH92] prend la forme dans notre cas d'une variation de T et de n en fonction de la génération. En effet, il apparaît inutile de tester les individus des premières générations dans des positions difficiles ou sur de nombreuses positions aléatoires alors qu'ils sont généralement incapables d'éviter les murs. De même, il n'est pas nécessaire d'évaluer l'adaptation des individus des premières générations pendant une période de temps importante pour distinguer les individus ayant un bon comportement de ceux incapables d'éviter les murs. Cependant, afin d'obtenir des contrôleurs robustes, il est souhaitable de tester les individus des dernières générations dans un maximum de situations difficiles.

On peut enfin noter que l'évolution incrémentale est le plus souvent utilisée pour réduire l'espace de recherche et non pas le temps d'évaluation. Dans

notre cas, cet espace est aussi réduit mais le gain est surtout important au niveau du temps requis pour calculer l'adaptation des individus.

2.4 Résultats

Paramètres Étant donné la simplicité de l'expérience, et en s'inspirant des paramètres utilisés par [MF95], nous avons choisi d'utiliser les paramètres suivant :

- Nombre d'individus : 60
- Nombre d'enfants par génération : 60
- Taux de mutation : 0.4
- Intervalle des mutations : 0.6

Analyse du déroulement de l'algorithme La figure 7 montre l'évolution de la valeur de la fonction d'adaptation correspondant au meilleur individu et sa valeur moyenne pour toute la population.

On observe qu'une cinquantaine de générations est suffisante pour obtenir uniquement des individus performants. Dans les premières générations, certains individus performants apparaissent ponctuellement, puis semblent disparaître. Cette disparition s'explique par la nature non-déterministe de l'évaluation de la fonction d'adaptation. Cette dernière fait en effet intervenir des positions aléatoires et il peut arriver que certaines positions se révèlent particulièrement favorables pour un individu particulier alors que ce dernier se comporte très mal dans le cas général.

La stabilité de valeur de la fonction d'adaptation pour les meilleurs individus – on pourrait s'attendre à ce qu'elle continue de croître – masque le fait que chaque génération est soumise à de plus en plus de tests. En conséquence, un maintien à la même valeur maximale entre deux générations peut être considéré comme une amélioration.

L'adaptation moyenne de la population croit régulièrement pour atteindre une valeur proche de celle du meilleur individu.

Une mesure intéressante pour estimer l'homogénéité de la population est le calcul de distance entre chaque individu. Il peut se calculer en utilisant les formules :

$$d(p,q) = \frac{1}{\mathrm{N}} \sum_{i}^{\mathrm{N}} |p_i - q_i| \quad (4)$$

$$\mathrm{D} = \frac{1}{\mathrm{P}} \sum_{i=0}^{\mathrm{P}} \sum_{j=0}^{\mathrm{P}} d(\mathrm{P}_i, \mathrm{P}_j) \quad (5)$$

où $d(p,q)$ désigne la distance entre les individus p et q, p_i et q_i le i-ème poids des individus q et p, D la distance moyenne, P le nombre d'individus dans la population et P_i le i-ème individu.

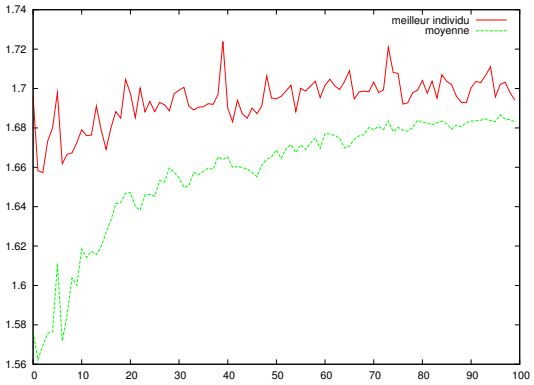

Figure 7: Évolution de la valeur de la fonction d'adaptation en fonction de la génération

La figure 8 présente pour chaque génération la valeur de D. On observe que D décroît rapidement jusqu'à un premier seuil en environ 20 générations. Un deuxième seuil est atteint au niveau de la 80-ème génération. Ce dernier seuil correspond à une valeur de 0.4 ; considérant que la mutation permet un déplacement de 0.6 au maximum et que le le taux de mutation choisi est élevé (0.4), ce deuxième seuil correspond à un ensemble d'individus très proches. Le premier seuil correspond quant à lui à l'élimination des solutions complètement mauvaises.

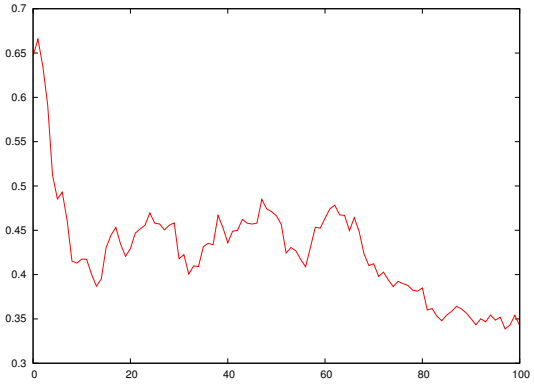

Figure 8: Évolution de la distance moyenne pour chaque composante entre les individus en fonction de la génération

Analyse des contrôleurs obtenus Seulement une dizaine de générations est nécessaire pour obtenir des individus capable à la fois d'éviter les obstacles et de maintenir une vitesse moyenne élevée. Les meilleurs contrôleurs utilisent tous la ligne droite, celle-ci correspondant à la trajectoire permettant la vitesse la plus élevée. Les robots correspondant cherchent aussi a minimiser le nombre de virages, ceux-ci les ralentissant. La figure 9 présente la trajectoire du meilleur robot de la génération 100.

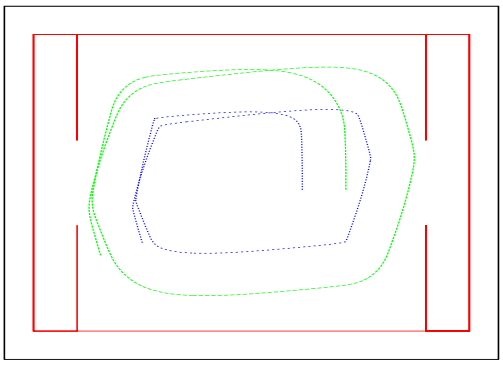

Figure 9: Trajectoires de chaque roue correspondant au meilleur individu de la génération 100

La figure 10 montre les variation de vitesse des moteurs pour un robot de la génération 5 et pour un de la génération 100. On peut tout d'abord constater que dans les deux cas, la même stratégie est utilisée : le moteur droit est bloqué à sa vitesse maximum et seul la vitesse du moteur gauche varie. L'individu de la génération 100 n'utilise jamais la marche arrière contrairement à celui de la génération 5 ; il atteint ainsi une vitesse moyenne bien meilleure.

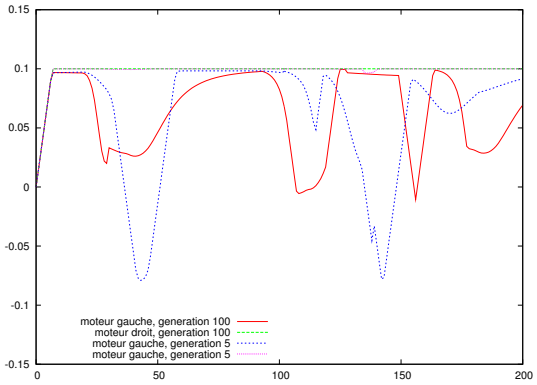

Figure 10: Variation de la vitesses des moteurs pour le meilleur individu de la génération 5 et pour le meilleur de la génération 100.

3 Évolution des poids d'un réseau de neurones complètement récurent

3.1 Motivation

Les résultats obtenus avec un réseau de neurones simple sont encourageants et laissent présager qu'un contrôleur plus évolué pourrait être utilisé. Ajouter des *neurones cachés* et des connexion récurrentes peuvent améliorer beaucoup les possibilités du réseau.

Parmi les problèmes que le précédent contrôleur est incapable de résoudre, celui de l'évitement des palmiers est particulièrement important. En effet, deux poteaux sont disposés aléatoirement sur le terrain. Leur faible épaisseur (2.4 cm) et l'impossibilité d'utiliser suffisamment de capteurs pour «couvrir» tout l'avant du robot[4] nécessite l'emploi d'un dispositif particulier.

Nous avons choisi d'équiper notre robot de 11 capteurs, dont certains ont un faisceau croisé. Ces croisements permettent de garantir que si le robot se déplace suffisamment lentement, un des faisceaux sera coupé. Nous sommes cependant en présence d'une information *ponctuelle*, le contrôleur doit donc «mémoriser» cette information pour amorcer un virage suffisamment large. Ce problème est assez proche du problème classique dénommé «T-Maze», où un robot doit décider de tourner vers la droite ou vers la gauche lors d'un croisement en T en fonction d'une information ponctuelle délivrée par une lumière avant le croisement.

Les 11 capteurs de distance sont positionnés autour du robot de la manière suivante :

- 4 capteurs sont disposés de manière similaire à la configuration du robot étudié dans la section précédente (Section 2) ;
- 2 capteurs sont positionnés à l'arrière du robot, leurs faisceaux se croisent ;
- 5 capteurs sont placés à l'avant du robot, à une hauteur de 15 cm, avec des faisceaux croisés.

Récurrence Les connexions récurrentes introduisent une notion de boucle de rétroaction, où la sortie du réseau permet de régler l'entrée, menant ainsi à un système auto-régulé. Parmi les structures

[4] En dehors des considérations financières, les capteurs risquent de se perturber mutuellement.

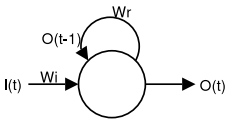

Figure 11: Neurone récurent

particulièrement intéressantes, on peut noter la possibilité d'émuler une mémoire et de lisser les données.

Pour réaliser une sorte de «mémoire», il suffit en effet de créer une connexion récurrente avec un poids important (Figure 11). Le tableau suivant présente une simulation pour un ensemble particulier d'entrées, en supposant le poids de la connexion récurrente w_r proche de 1.

Date	Entrée	Sortie
0	0	0
1	1	$\varphi(w_r \times 0 + w_i \times 1) \rightarrow 1$
2	0	$\varphi(w_r \times 1 + w_i \times 0) \rightarrow 1$
⋮	⋮	⋮
n	0	$\varphi(w_r \times 0.1 + w_i \times 0) \rightarrow 0$

On constate que la valeur 1 a été mémorisée pendant une durée dépendant du poids w_r ; plus ce dernier poids tendra vers 1, plus la durée de mémorisation sera importante. Le robot peut ainsi ainsi commencer une manoeuvre d'évitement suite à une information ponctuelle.

Parmi les autres possibilités offertes par un neurone récurrent, on peut aussi citer le «lissage» des données. En effet, si par exemple $w_r = \frac{1}{4}$, on a $\mathrm{O}_t = \varphi(w_i \mathrm{I}_t + \frac{1}{4}\mathrm{O}_{t-1})$, ce qui permet d'amortir les changements brusques dans les valeurs d'entrée.

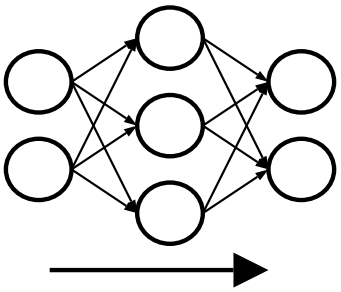

Figure 12: Un perceptron à trois couches, dont une cachée

Neurones cachés Les neurones cachés sont des neurones qui ne sont pas reliés directement aux entrées ; il s'intercalent entre les entrées et la sortie. Grossièrement, on peut les voir comme des unités capables de fusionner et combiner des données. Mathématiquement, il a été montré qu'un perceptron multi-couches (Figure 12) avec suffisamment de neurones cachés pouvait approximer n'importe quelle fonction avec une précision arbitraire. On peut par conséquent envisager d'ajouter des neurones cachés dans notre réseau afin d'en accroître les possibilités.

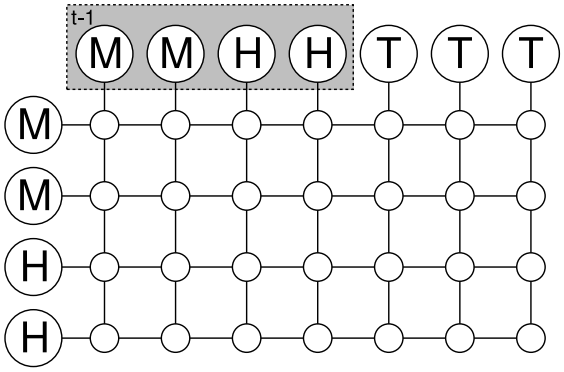

Figure 13: Réseau de neurone complètement récurent. T désigne les capteurs, H les neurones cachés et M les moteurs, chaque petit cercle désigne une synapse.

Topologie choisie et encodage La figure 13 présente le réseau choisi. Il s'agit d'un réseau complètement récurrent où chaque neurone est connecté à tous autres et où toutes les sorties au temps t sont données en entrée au temps $t + 1$. Ce type de réseau est généralement dénommé Continuous Time Recurrent Neural Network (CTRNN) dans la littérature.

L'ensemble des poids de chaque synapse se représente très bien sous la forme d'une matrice de nombres à virgule flottante. Pour évaluer la valeur de sortie $\mathrm{O}_i(t)$ du neurone i, il suffit alors de calculer :

$$\mathrm{O}_i = \varphi\left(\sum_j w_{ij}\mathrm{O}_j(t-1) + \sum_k w_{ik}\mathrm{T}_k\right)$$

où j désigne l'indice des entrées «récurrente», k celui des capteurs et w_{ij} le poids de la synapse reliant le neurone de la ligne i à celui de la colonne j.

Le génome est constitué de la suite des lignes de la matrice $\mathbf{w}$ mises bout à bout.

Nous utilisons 11 capteurs répartis sur le robot et 10 neurones cachés. En ajoutant les deux moteurs et les biais pour chaque neurone, il est donc nécessaire de travailler sur $18 \times 12 = 216$ variables. Des tests ont été conduits avec seulement 5 neurones cachés et les résultats semblaient légèrement moins bon. Il n'existe actuellement à notre connaissance aucune méthode pour déterminer le nombre «idéal» de neurones cachés.

3.2 Dispositif expérimental

Si le robot n'est sensé savoir éviter que deux palmiers, nos premiers tests avec cet environnement se sont montrés décevant : aucuns contrôleurs ne montraient d'aptitude à l'évitement des palmiers. La raison de cet échec est simplement que lors de la plupart des évaluation, le robot ne rencontrait pas de palmiers. La collision avec cet obstacle restait donc un évènement exceptionnel, qui ne permettait pas aux individus de se distinguer.

La solution adoptée consiste à placer aléatoirement huit palmiers sur le terrain. Ainsi les robots incapable d'éviter les palmiers ont une probabilité importante d'entrer en collision avec un de ces obstacles.

3.3 Fonction d'adaptation

L'ajout de récurrence dans le réseau mène à un optimum local non désiré : un comportement oscillant de type «avant-arrière». En effet, si le robot possède une accélération suffisante, qui lui permet ainsi de changer de direction rapidement, une bonne solution pour maximiser la distance parcourue entre l'instant t et $t+1$ est une oscillation «avant-arrière». Ce comportement permet fort logiquement d'éviter tous les obstacles et les individus l'exhibant se révèlent en conséquence bien adapté.

Pour forcer les robots à se déplacer, on ajoute un terme donnant une indication de la surface parcourue par le robot. Le terrain est divisé en carrés de 30×30 cm et le nombre de case visitées par le robot est ajouté à la fonction d'adaptation 3 :

$$\mathrm{F} = \frac{1}{\mathrm{T}} \sum_{n} \sum_{t=0}^{\mathrm{T}} d_{t,t+1} - \sum_{i,j} w_i j^2 + \sum_{i} \mathrm{V}(i) \qquad (6)$$

où $d_{t,t+1} = \|\overrightarrow{p(t) - p(t-1)}\|$ et :

$$\mathrm{V}(i) = \begin{cases} 1 & \text{Si la case i a été visitée} \\ 0 & \text{sinon} \end{cases} \qquad (7)$$

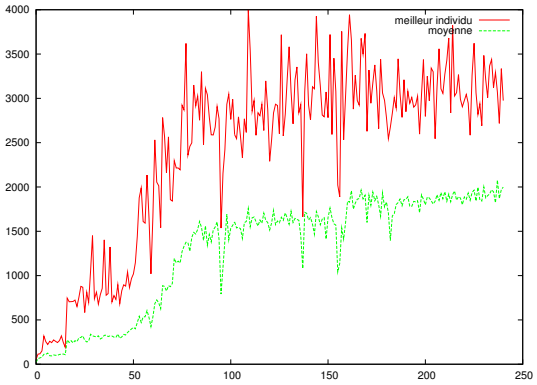

Figure 14: Évolution de la valeur de la fonction d'adaptation en fonction de la génération

3.4 Résultats

Évolution de la fonction d'adaptation La figure 14 montre l'évolution de la valeur de la fonction d'adaptation en fonction des générations. On observe trois paliers successifs, particulièrement visible sur la courbe représentant la valeur moyenne, mais aussi sur celle du meilleur individu. Le premier, correspondant à environ 10 générations, est lié à l'émergence de robots capables d'éviter les murs. L'adaptation passe alors d'un coup d'une valeur proche de 0 à une valeur de l'ordre de 30. Pendant environ 40 générations, les nouvelles solutions sont sont à peine meilleures. Puis, aux alentours de la 50-ème génération, des contrôleurs capables d'éviter les palmiers apparaissent et permettent ainsi une explosion de la valeur d'adaptation. Cette valeur ne croît ensuite que légèrement, n'améliorant que légèrement les capacité d'évitement.

Il faut de plus noter que comme nous l'avons vu précédemment (Section 2), il faut prendre garde à prendre en compte que à chaque génération les tests sont de plus en plus complexes. La faible augmentation des valeurs constatées cache par conséquent une amélioration légèrement plus importante de l'adaptation.

Cette évolution par paliers est étrangement proche de la *théorie des équilibres ponctués* [NE72], actuellement très discutée par les paléontologues. Opposée au *gradualisme*, cette théorie est basée sur l'interprétation des changements morphologiques survenus dans des collections de fossiles d'organismes marins. Stephen J. Gould et Eldredge ont constaté que des espèces fossiles variaient très peu morphologiquement au cours de leur existence qui du-

rait plusieurs millions d'années. Ensuite, une nouvelle espèce morphologiquement bien différenciée la supplantait subitement – en quelques dizaines de milliers d'années – et l'on n'observait pas de formes intermédiaires. Cette absence de formes intermédiaires est vraisemblablement due au faible effectif des populations intermédiaires qui n'ont pas laissées de traces fossiles.

Comportements observés Les figure 15 et 16 montrent deux exemples de trajectoires observées. Les robots sont effectivement capables d'éviter les palmiers et les murs dans la plupart des situations.

On peut légitimement se demander si les contrôleurs obtenus tirent parti des connexions récurrentes et des neurones cachées. Si répondre exhaustivement à cette question nécessiterait une étude plus poussée, on observe parmi les meilleurs individus et à partir d'environ 100 générations, des contrôleurs utilisant assez clairement une mémoire à court terme. Ils sont en effet capables, suite à la rencontre d'un obstacle, de maintenir une marche arrière pendant un temps variant entre les individus, puis de revenir sur un mode de marche avant. Il est très difficile d'estimer l'influence de la présence de neurones cachés.

Le tableau suivant résume les différents comportements observés chez les meilleurs individus pour chaque pallier :

Génération	Comportement
$0-10$	mouvement en ligne droite pas de tentative d'évitements
$11-50$	évitement des murs
$51-100$	évitement des palmiers

Les indications générations sont évidemment purement indicatives, le nombre de générations correspondant à chaque palier variant légèrement selon les expériences.

4 Conclusion

Dans une première expérience, nous avons utilisé un réseau de neurones très simple, inspiré de ceux suggérés par Braitenberg que nous avons optimisé à l'aide d'un algorithme génétique afin d'obtenir un contrôleur pour un robot mobile capable d'éviter les murs sur le terrain de la Coupe de France de Robotique. La fonction d'évaluation utilisée est très simple et ne donne aucune indication sur la manière de résoudre le problème. Ainsi, les meilleurs contrôleurs sont ceux qui permettent aux robots de parcourir le maximum de distance sans toucher d'obstacles. Il est apparu nécessaire d'ajouter à la fonction d'adaptation un terme de régularisation. Les opérateurs génétiques utilisés incluent la roulette biaisée et le croisement barycentrique.

Les résultats observés sont conformes à nos attentes. Les robots utilisent des lignes droites jusqu'à ce qu'ils soit proches d'un mur, ils amorcent ensuite une manoeuvre d'évitement.

La deuxième expérience prend place sur le même terrain sur lequel on ajoute des poteaux dont l'évitement nécessite l'exploitation d'une information ponctuelle. Le réseau de neurones utilisé est apparenté à un Continuous Time Recurrent Neural Network, comportant de nombreuses connexions récurrentes et des neurones cachés. De nouveaux capteurs de distance sont aussi ajoutés.

Une cinquantaine de générations est suffisante pour obtenir des contrôleurs performants. La fonction d'adaptation utilisée correspond à celle utilisée dans l'expérience précédente à laquelle on ajoute un terme mesurant le nombre de cases traversées par le robot.

L'étape suivant cette étude est la validation des contrôleurs évolués dans le simulateur sur le robot réel. Les capteurs du robot ayant chacun leurs défauts, les moteurs ne réagissant pas exactement comme ceux simulés, comment obtenir un contrôleur performant sur le robot réel ? Plusieurs études ont été faite dans ce sens [Jak98, MLN95]. [Jak98] suggère l'ajout d'un bruit uniforme sur les capteurs et actionneurs lors de la période d'évolution. [MLN95] modélisent les capteurs particuliers de leur robot via un ensemble de mesures avant de les incorporer dans leur simulateur.

Une autre approche est de «terminer» l'évolution commencée dans le simulateur sur le robot réel. Dans ce cas, en partant de la population résultant d'environ 200 générations sur le simulateur, on peut imaginer que quelques générations sont suffisantes pour obtenir un contrôleur viable. Il faut cependant noter que l'évaluation de la performance des contrôleurs sur le robot réel nécessite l'emploi d'un dispositif d'évaluation externe, par exemple une caméra enregistrant les déplacements du robot.

Une autre approche envisageable est d'intercaler entre les moteurs et le contrôleur un perceptron multicouche (PMC) dont on réglerait les poids en utilisant une rétro-propagation standard. Disposant d'un nombre important d'exemples correspondant à des couples ordre-réponse, le PMC devrait être capable de modéliser un comportement réaliste du moteur que l'on pourrait ensuite incorporer dans le simulateur. Le défaut évident de cette approche est l'impossibilité de modéliser l'accélération.

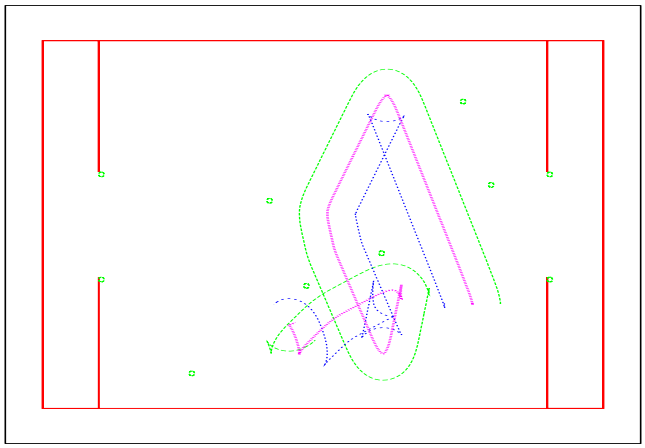

Figure 15: Trajectoires de chaque roue correspondant au meilleur individu de la génération 150. Après être parti du coin droit du terrain, le robot a d'abord évité un mur, puis un palmier et a réussit à se glisser entre deux autres palmiers. Il enclenche ensuite une manoeuvre d'évitement du mur mais rencontre un nouveau un palmier, qui le contraint à tourner à nouveau.

References

[Bra84] Valentino Braitenberg. *Vehicles*. MIT Press, Cambridge MA, 1984, 1984.

[Bro86] Rodney A. Brooks. A robust layered control system for a mobile robot. In *IEEE Journal of Robotics and Automation*, volume RA-2 (1), pages 14–23, April 1986.

[Bro91] Rodney A. Brooks. Intelligence without representation. Number 47 in Artificial Intelligence, pages 139–159. 1991.

[CHH92] D. Cliff, I. Harvey, and P. Husbands. Incremental evolution of neural network architectures for adaptive behaviour. Technical Report Cognitive Science Research Paper CSRP256, Brighton BN1 9QH, England, UK, 1992.

[FU99] D. Floreano and J. Urzelai. Evolution of neural controllers with adaptive synapses and compact genetic encoding. In *Proc. of EUCAL*, pages 183–194, 1999.

[Hay99] Simon Haykin. *Neural Networks, a comprehensive foundation*. Prentice Hall International Editions, 1999.

[Jak98] N. Jakobi. Minimal simulations for evolutionary robotics, 1998.

[Mey95] J.-A. Meyer. The animat approach to cognitive science. In H. Roitblat and J.-A. Meyer, editors, *Comparative Approaches to Cognitive Science*. The MIT Press, 1995.

[Mey98] Jean-Arcady Meyer. Evolutionary approaches to neural control in mobile robots. In *Proc. of the IEEE Int. Conf. on Systems, Man and Cybernetics*, page ?, Piscataway, NJ, 1998. IEEE Press.

[MF95] Francesco Mondada and Dario Floreano. Evolution and mobile autonomous robotics. In *Towards Evolvable Hardware*, pages 221–249, 1995.

[MLN95] Orazio Miglino, Henrik Hautop Lund, and Stefano Nolfi. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4) :417–434, 1995.

[NE72] S.J. Gould N. Eldredge. Punctuated equilibria : An alternative to phyletic gradualism. *Models of Paleobiology*, 1972.

[NFMM94] Stefano Nolfi, Dario Floreano, Orazio Miglino, and Francesco Mondada. How to evolve autonomous robots : Different approaches in evolutionary robotics. pages 190–197, 1994.

[Yao99] Yao. Evolving artificial neural networks. *PIEEE : Proceedings of the IEEE*, 87, 1999.

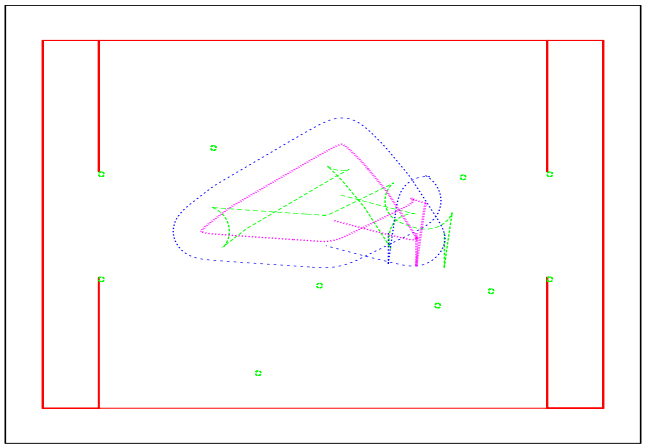

Figure 16: Trajectoires de chaque roue correspondant au meilleur individu de la génération 210 ; Ce robot présente l'intéressante particularité de reculer pendant environ une seconde lorsqu'il rencontre un obstacle. On peut observer qu'il a ainsi réussi à éviter tous les palmiers qu'il a rencontré.