République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : Intelligence Artificielle

Sujet :

Elaboration d'un contrôleur adaptatif pour un robot mobile basé sur les réseaux immunitaires artificiels

Présenté par : CHABANE Yahia
DJIAR Sofiane

Promoteur : Dr S. OUKID-KHOUAS

Organisme d'accueil : Laboratoire de Recherche pour le Développement des Systèmes Informatisés (LRDSI).

Soutenue le: date soutenance, devant le jury composé de :

Nom. président du jury, grade, organisme	**Président**
Nom examinateur 1, grade, organisme	**Examinateur**
Nom examinateur 2, grade, organisme	**Examinateur**

- promotion 2005/2006-

Remerciements

Nous remercions avant tout Dieu Tout-puissant qui nous a donné la force, le courage et la volonté pour réaliser ce travail.

Nous tenons à remercier notre promotrice Mme S. OUKID-KHOUAS de nous avoir accepté durant cette année et pour son suivi, ses orientations et ses précieux conseils, et qui a su nous faire profiter de sa grande expérience.

Nous remercions Mr BEN CHERCHALI, du département Électronique, pour son aide et sa disponibilité.

Un grand merci à toutes nos familles pour leur présence, leur préoccupation et le souci qu'ils se sont fait pour nous, leurs encouragements et leur suivi, avec patience, du déroulement de notre projet.

Enfin, Nous remercions, de tout coeur, tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail.

Dédicace

Je dédie ce travail :

A ma mère : même si la vie n'a pas toujours été facile dans le foyer familial, tu as su parfois te sacrifier pour nous offrir une vie meilleure. Et grâce à toi, je profite aujourd'hui du meilleur de la vie. J'aimerais pouvoir te rendre tout l'amour et la dévotion que tu nous as offerts, mais une vie entière n'y suffirait pas. J'espère au moins que ce mémoire y contribuera en partie,

A mon père qui m'a soutenu durant toute ma carrière et m'a aidé à surmonter les moments difficiles,

A mes deux frères Mohamed et Djamel,

A toute ma famille,

A mon binôme Sofiane et à toute sa famille,

A la mémoire de notre amis Nasr-eddine,

A tous mes amis,

A tous les enseignants et étudiants du département Informatique de l'université de Blida.

C. Yahia

Dédicace

Je dédie ce travail :

A la mémoire de ma chère mère. Qui nous a quittées le 30-01-1998 à jamais pour un monde meilleur. Je lui dois ce travail et tout ce que je suis. Et je veux lui dire qu'on connaît l'arbre, de ses fruits.
A dieu nous appartenons, et à lui nous retournons.

A celui qui ma aidé à découvrir le 'savoir' le trésor inépuisable
A mon très cher père, c'est à toi que je dédie ce mémoire, pour te dire merci pour ton soutien et tes encouragements, merci pour ta douceur et ta compréhension.

A mes frères, sœurs, mes cousins et tous ceux qui me sont cher pour leur encouragement et aide.

A mon binôme Yahia et à toute sa famille,
A la mémoire de notre amis Nasr-eddine,
A tous mes amis, Hamza, Billel, Noureldine, Ibrahim ...

A tous les enseignants et étudiants du département Informatique de l'université de Blida.

D. Sofiane

Résumé

Aujourd'hui, le marché commercial de la robotique mobile est toujours relativement restreint, mais il existe de nombreuses perspectives de développement qui en feront probablement un domaine important dans le futur. Les applications des robots peuvent se trouver dans de nombreuses activités "ennuyeuses, salissantes ou dangereuses", mais également pour des applications ludiques ou de service, comme l'assistance aux personnes âgées ou handicapées.

De nos jours, une des tâches les plus difficiles en robotique est le problème de la navigation autonome, ou un robot, ou un ensemble de robots, doit accomplir certaines tâches sans intervention humaine.

Ces dernières années beaucoup d'attention a été concentrée sur l'approche comportementale de l'intelligence artificielle, qui a déjà démontré sa robustesse et flexibilité contre des environnements dynamiques. Notre travail consiste à réaliser un mécanisme d'arbitrage de comportement pour un robot mobile autonome afin de l'adapter à un environnement inconnu et ceci en utilisant les systèmes immunitaires artificiels et plus précisément la théorie des réseaux immunitaires présenté par N. Jerne.

Mots clés : robot mobile autonome, navigation réactif, contrôleur adaptatif, système immunitaire artificiel, réseau immunitaire.

Abstract

Today, the produce market of mobile robotics is always relatively restricted, but there are many development prospects which will probably make of it a significant field in the future. The applications of the robots can be in many "tedious, dirtying or dangerous" activities, but also for ludic applications or of service, like the assistance with the old or handicapped people.

Nowadays, one of the most difficult tasks in robotics is the problem of autonomous navigation, where a robot, or a whole of robots, must achieve some tasks without human intervention.

These last years much of attentions were concentrated on the behavioral approach of the artificial intelligence, which already showed his robustness and flexibility in dynamic environments. Our work consists in producing a arbitration mechanism of behavior for an autonomous mobile robot in order to adapt it to an unknown environment by using the artificial immune systems and more precisely the immunizing network analysis presented by N Jerne.

Key words : autonomous mobile robot, reactive navigation, adaptive controller, artificial immune system, immune network.

Sommaire

Chapitre III — Conception, Réalisation et Tests

Listes des figures

Listes des tableaux

Introduction générale

Introduction générale

Un agent autonome doit, à chaque instant, choisir quelle action effectuer parmi un ensemble d'actions possibles, en fonction de ses perceptions internes et externes, de son état interne et de ses objectifs.

L'Intelligence Artificielle classique s'attache généralement à simuler les facultés humaines telles que le raisonnement logique ou la compréhension du langage naturel, elle est basée sur la manipulation de connaissances symboliques, considérée comme nécessaire et suffisante pour obtenir des comportements intelligents. Certains de ces systèmes symboliques se sont avérés très performants pour un domaine d'expertise donné, mais complètement inefficaces pour des domaines connexes car le modèle de monde introduit par le programmeur n'est souvent pas adaptable à un environnement légèrement différent, par exemple, un agent champion d'échecs sera incapable de jouer aux dames. De nombreux travaux en IA classique portent en particulier sur la planification globale symbolique, c'est-à-dire sur la sélection d'une séquence d'actions à réaliser en fonction d'une unique représentation symbolique de l'environnement et d'un objectif global lui aussi unique. L'élaboration d'un plan est une opération coûteuse dans des environnements complexes, par exemple lorsqu'il faut calculer le chemin optimal à suivre pour traverser une pièce contenant un grand nombre d'obstacles. Lorsque la planification est le seul mécanisme de sélection de l'action, l'utilisation d'un plan n'est efficace que dans un environnement prévisible, il est évidemment impossible de prévoir tous les cas dans un environnement dynamique et partiellement connu. La planification traditionnelle n'est généralement pas interruptible, et ne prend donc pas en compte l'environnement courant.

Devant le manque de réactivité, de robustesse et d'adaptabilité des agents purement cognitifs, des chercheurs ont posé les bases de ce qu'ils ont appelé «la nouvelle IA» à partir de leurs expériences dans les domaines de la robotique autonome et de la vie artificielle, en s'inspirant de la biologie et de l'éthologie.

D'après R. Brooks, L. Steels ou encore P. Maes, il n'est pas indispensable d'être capable de raisonner sur une représentation abstraite du monde pour présenter un comportement intelligent, c'est-à-dire permettant de survivre au mieux dans son environnement. En couplant les capteurs et les effecteurs d'un agent situé, il est en effet possible de produire des comportements de survie correspondant à une certaine forme d'intelligence, la représentation symbolique n'est pas nécessaire dans ce cas, puisqu'il suffit à l'agent de savoir interpréter les informations en fonction de sa perception de l'environnement et du but poursuivi.

De nombreuses architectures ont été proposées et testées avec succès sur des robots mobiles, mais il a été rapidement démontré que ces approches limitaitent considérablement les capacités des agents, ces système sont robuste, mais manquent d'adaptabilité puisqu'il ne possède ni représentation dynamique de son environnement, ni capacité d'apprentissage ce qui a mené les chercheurs a ce diriger vers des architectures hybrides, qui permettent aux agents de réagir rapidement aux modifications de leur environnement tout en étant capables d'anticiper et de planifier quand cela est possible.

Devant ce manque d'adaptabilité plusieurs architectures basées sur l'approche réactive ont été couplées aux techniques d'apprentissage (réseaux de neurones, algorithmes génétiques et systèmes immunitaires artificiels...) afin que le comportement de l'agent soit intelligent, adaptatif, flexible et autorégulateur.

Le système immunitaire est un moyen de défense remarquable, qui existe chez de très nombreux organismes comme les insectes, les plantes ...etc. Mais dont la forme la plus évoluée est retrouvée chez les vertébrés supérieurs. Il comporte un ensemble de moyens (organes, tissus, cellules et molécules) permettant de répondre rapidement, de façon souvent spécifique et efficace contre les nombreux pathogènes auxquels nous sommes confrontés. Il montre beaucoup de caractéristiques cognitives intéressantes comme : la reconnaissance des formes moléculaires étrangères, l'apprentissage de nouvelles formes, la mise en mémoire de ces formes et l'adaptation à des environnements dynamiques. D'où l'apparition des systèmes immunitaires artificiels (AIS) qui ont démontré leur efficacité dans plusieurs domaines.

Notre objectif est la construction d'un système basé sur la théorie des réseaux immunitaires présenté par N. Jerne pour diriger et faire adapter un robot mobile dans un environnement inconnu en évitant des obstacles statiques et en atteignant un but.

Le mémoire est organisé en trois chapitres :

- *Chapitre I :* présente le problème de la navigation autonome des robots mobiles et les différentes architectures proposés pour sa solution.
- *Chapitre II :* présente les systèmes immunitaires artificiels, la théorie des réseaux immunitaires et justifie leur application à la navigation autonome des robots mobiles.
- *Chapitre III :* présente la démarche de développement de notre système et les résultats obtenus.

Pour plus de détail, nous présenterons la bibliographie et les Annexes :

- L'annexe A présente les robots mobiles et leurs applications.
- L'annexe B présente le robot Khepera.
- L'annexe C présente le simulateur du robot Khepera.

CHAPITRE

I

I. Navigation autonome des robots mobiles

1. Introduction

Un robot mobile (voir l'annexe A) est un système mécanique, électronique et informatique agissant physiquement sur son environnement en vue d'atteindre un objectif qui lui a été assigné. Cette machine est polyvalente et capable de s'adapter à certaines variations de ses conditions de fonctionnement. Elle est dotée de fonctions de perception, de décision et d'action. Ainsi, le robot devrait être capable d'effectuer des tâches diverses, de plusieurs manières, et accomplir correctement sa tâche, même s'il rencontre de nouvelles situations inattendues.

Nous nous intéressons dans ce mémoire à une sous famille des robots mobiles existants appelée *Les robots mobiles autonomes.* Un robot est dit autonome :

- S'il est capable de choisir ses actions pour atteindre son but.
- S'il est capable d'accomplir correctement sa tâche même s'il rencontre de nouvelles situations inattendues sans intervention humaine.

Dans ce chapitre nous présenterons le problème de la navigation dans la robotique mobile. On définira les différentes stratégies de navigation et on donnera les catégories des contrôleurs existants.

2. Quelques notions sur la navigation des robots

2.1. Définition

D'après Levitt et Lawton (1990), la navigation est définie comme le procédé permettant de répondre aux trois questions suivantes :

(i) "Où suis-je ?".

(ii) "Où sont les autres lieux par rapport à moi ?".

(iii) "Comment puis-je atteindre ces autres lieux depuis l'endroit où je me trouve ?".

Donc, la navigation est la science [Wiki 2006] et l'ensemble des techniques qui permettent de :

- connaître la position (coordonnées) d'un mobile par rapport à un système de référence, ou par rapport à un point fixe déterminé.
- calculer ou mesurer la route à suivre pour rejoindre un autre point de coordonnées connues en respectant un certain nombre de contraintes et de critères qui découlent de plusieurs facteurs, qui dépendent généralement des caractéristiques du robot, de l'environnement, et du type de tâche a exécuter.
- calculer toute autre information relative au déplacement de ce mobile (distance et durée, vitesse de déplacement, heure estimée d'arrivée, etc.)

2.2. Les stratégies de navigation

Les stratégies de navigation qui permettent à un robot mobile de se déplacer pour rejoindre un but sont extrêmement diverses, de même que les classifications qui peuvent en être faites. Nous reprenons ici une classification établie par Trullier et Meyer [Trul et Mey 1997], qui présente l'avantage de distinguer les stratégies sans modèles internes et les stratégies avec modèle interne. Cette classification comporte :

2.2.1. La navigation par carte

Elle s'appuie sur un modèle interne du monde, une carte, qui supporte une planification. Ce modèle interne mémorise la structure spatiale de l'environnement, indépendamment d'un but précis. Chacune des positions mémorisées dans ce modèle interne peut alors être utilisée comme but par le processus de planification dont le rôle est de calculer une route vers ce but. Le principe de la navigation par carte est basé sur la cartographie, la localisation et la planification. On peut citer deux sous classe :

- **Navigation par carte topologique :** Une carte topologique est un graphe, dans lequel chaque noeud correspond à un endroit caractéristique (carrefour entre couloirs, entrées dans les espaces ouverts ...), appelé *meeting point* dans plusieurs travaux, ou encore lieu. Chaque lieu devra être décrit par un ensemble de caractéristiques propres, qui permettront au robot de le reconnaître. Une arête liant deux noeuds signifie qu'il existe une commande référencée capteur que le robot peut exécuter afin de se déplacer entre les deux lieux correspondants. Cette représentation est particulièrement adaptée pour

décrire des réseaux de couloirs : les lieux correspondent aux carrefours, les arêtes aux couloirs qui les lient. Pour la génération d'une trajectoire entre deux lieux définis dans une carte topologique, il suffit de rechercher un chemin dans le graphe. Dans cette stratégie, le robot n'a nul besoin de connaître sa position précise par rapport à un repère du monde. Il doit seulement déterminer une localisation qualitative « *Je suis dans le lieu A* » et, au mieux, une estimation de sa position relative par rapport à un repère lié au lieu dans lequel il se trouve.

- **Navigation par carte métrique :** La navigation métrique exploite une représentation géométrique du monde, qui peut être donné par un utilisateur, ou construite par le robot lui-même. Elle représente une extension de la précédente car elle permet au robot de planifier des chemins au sein de zones inexplorées de son environnement. Elle mémorise pour cela les positions métriques relatives des différents lieux, en plus de la possibilité de passer de l'un à l'autre. Ces positions relatives permettent, par simple composition de vecteurs, de calculer une trajectoire allant d'un lieu à un autre, même si la possibilité de ce déplacement n'a pas été mémorisée sous forme d'un lien. L'équivalent mathématique de la carte métrique, est un espace vectoriel muni d'une distance, où chaque vecteur $\overrightarrow{AB}$ exprime le chemin entre un lieu A et un lieu B en tenant compte de la distance et de l'orientation. Les détours et raccourcis métriques peuvent alors être manipulés via les opérations vectorielles telles que la norme (estimation des distances), l'addition ou la soustraction vectorielle (construction de nouveaux chemins). Il n'est plus requis d'avoir emprunté un chemin pour pouvoir l'utiliser, et c'est bien là la différence fondamentale avec la carte topologique.

2.2.2. La navigation réactive

Les algorithmes de modélisation de l'environnement sont très coûteux en temps de calcul. Le robot en plus a souvent qu'une vue très partielle de son environnement. Ceci implique que le robot peut remettre en cause son plan, en raison de présence imprévue d'obstacles par exemple, entraînant une nouvelle modélisation coûteuse de l'environnement et une régénération de chemin. De plus, le délai entre l'apparition d'un nouvel obstacle, sa prise en compte dans le modèle et la régénération d'un nouveau plan pouvant être important, il existe un risque de collision pour le robot si une réponse immédiate est nécessaire. Pour cela

les roboticiens ont cherché à remplacer ou compléter la navigation à base de carte par des processus rapide qui se regroupent sous le terme de navigation réactive. La navigation réactive utilise des actions réflexes pour guider le robot et se différencient essentiellement par le type de perceptions utilisées pour déclencher ces actions. Les comportements de ce type restent essentiels dans les robots modernes car, du fait de leur simplicité, ils sont généralement exécutés très rapidement et ils permettent de réaliser des tâches de bas niveau, comme l'évitement des obstacles imprévus, essentielles à la sécurité d'un robot. On peut citer trois sous classe :

- **Navigation par approche d'un objet :** cette capacité de base permet de se diriger vers un objet visible depuis la position courante du robot. Elle est en général réalisée par une remontée sur la perception de l'objet, comme dans l'exemple célèbre des véhicules de Valentino Braitenberg qui utilisent deux capteurs de lumière pour atteindre ou fuir une source lumineuse. Cette stratégie utilise des actions réflexes, dans lesquelles chaque perception est directement associée à une action. C'est une stratégie locale, c'est-à-dire fonctionnelle uniquement dans la zone de l'environnement pour laquelle le but est visible.

- **Navigation par Guidage :** Cette capacité permet d'atteindre un but qui n'est pas un objet matériel directement visible, mais un point de l'espace caractérisé par la configuration spatiale d'un ensemble d'objets remarquables, ou amers, qui l'entourent ou qui en sont voisins. La stratégie de navigation, consiste alors à se diriger dans la direction qui permet de reproduire cette configuration. Le robot tente donc de mémoriser une "photo" de l'ensemble de ses perceptions lorsqu 'il se trouve dans une situation but et cherche à faire correspondre ses perceptions courantes avec la situation mémorisée. Cette stratégie de navigation requiert néanmoins la perception directe des amers caractéristiques de la situation but et dès lors qu'ils ne sont plus perçus, la stratégie échoue et ne peut donc être utilisée que dans un contexte local. Cette capacité semble utilisée par certains insectes, comme les abeilles, et a été utilisée sur divers robots. Cette stratégie utilise également des actions réflexes et réalise une navigation locale qui requiert que les amers caractérisant le but soient visibles. Enfin il est à noter que ce type de navigation ne requiert aucune modélisation spatiale. En effet, la mise

en correspondance d'une mémoire perceptive avec les perceptions courantes ne requiert en aucun cas un traitement spatial spécifique.

- **Navigation par association :** Cette capacité est la première capacité réalisant une navigation globale, c'est-à-dire qui permette de rejoindre un but depuis des positions pour lesquelles ce but ou les amers qui caractérisent son emplacement sont invisibles. Elle s'applique généralement à des environnements plus conséquents et repose sur la notion de lieu et requiert une représentation interne de l'environnement qui consiste à définir des lieux comme des zones de l'espace dans lesquelles les perceptions restent similaires, et à associer une action à effectuer à chacun de ces lieux. L'enchaînement des actions associées à chacun des lieux reconnus définit une route qui permet de rejoindre le but. Cependant, ce type de navigation, s'il est plus robuste que la navigation par guidage, ne permet pas encore l'établissement de réelle stratégie dans la mesure où les informations mémorisées dépendent directement du but que l'on souhaite rallier. Si le robot souhaite rallier un autre but, il doit posséder une mémorisation différente de l'environnement orientée vers ce nouveau but. Il n'existe pas encore de véritable représentation interne de l'environnement à ce niveau de navigation dans la mesure où les différents lieux mémorisés ne possèdent pas de relations internes. Etant donné un lieu *A*, un robot connaît la direction immédiate à prendre, mais il est incapable de prédire la prochaine direction avant de se trouver effectivement dans le nouveau lieu. Ces modèles permettent donc une autonomie plus importante mais sont limités à un but fixé. Une route qui permet de rejoindre un but ne pourra en effet pas être utilisée pour rejoindre un but différent. Changer de but entraînera l'apprentissage d'une nouvelle route, indépendante des routes permettant de rejoindre les autres buts.

3. Les architectures de contrôles de robots

Les architectures de contrôles peuvent être classées en trois grandes catégories que nous détaillerons par la suite : les contrôleurs hiérarchiques, les contrôleurs réactifs et les contrôleurs hybrides. Toutes ces architectures ne diffèrent pas forcement par les méthodes élémentaires employées mais plutôt par leur agencement et leurs relations.

3.1. Les contrôleurs hiérarchiques

Ces approches s'appuient sur les techniques de l'intelligence artificielle classique. Elles visent à reproduire le mode de raisonnement humain ou tout au moins une certaine vision du mode de raisonnement humain. Pour cela, ils sont appuyé sur un schéma d'intelligence artificielle conçu dans les années cinquante par Newell et Simon. Le traitement est décomposé en une série d'opérations successives décrites par la figure I.1.

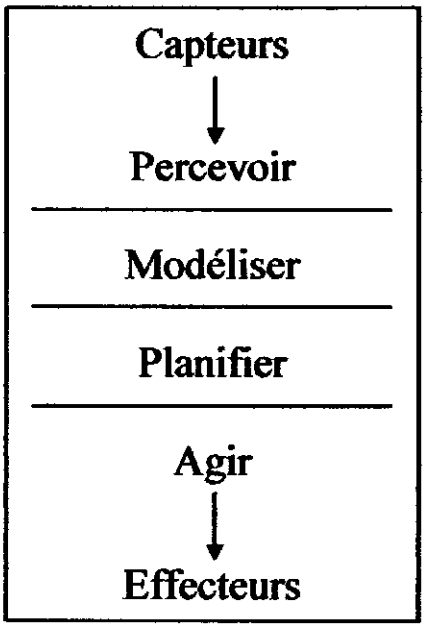


Figure I.1 : contrôleur hiérarchique

- La première opération consiste à traiter les données sensorielles qui fournissent au robot des informations sur son environnement.

- L'étape suivante consiste à construire ou à mettre à jour, à partir des données capteurs, un modèle du monde dans lequel évolue le robot. Ce modèle peut être par exemple une identification du type et de la position des obstacles que le robot doit éviter.
- Ce modèle est utilisé pour planifier une suite d'états conduisant le robot à effectuer la tâche recherchée.

- La dernière opération a pour but de calculer puis d'exécuter les actions permettant au robot de suivre le plan généré par l'étape précédente.

La partie la plus importante, et la plus gourmande en calcul, est celle qui concerne la modélisation et la planification qui peuvent prendre un temps assez long, et ceci notamment en environnement peu structuré. Le temps disponible pour l'action est par conséquent beaucoup plus court. L'intervalle de temps entre les phases de perception et d'action peut de plus être relativement long. On parle souvent pour cette approche de structure S.M.P.A. (pour Sense, Model, Plan, Act).

Dans une telle architecture, on essaie de construire un modèle de l'environnement le plus complet possible et ensuite de raisonner sur la (les) représentation(s) de celui-ci. On privilégie l'aspect cognitif dans le but de reproduire au mieux l'intelligence humaine. Ces architectures ont rapidement montré leurs limites et leur incapacité à fonctionner dans un environnement qui ne soit pas statique et simplifié à l'extrême.

Ses points forts sont :

- L'avantage essentiel de ces architectures réside dans la possibilité d'intégrer des raisonnements de haut niveau (niveau mission, planification) qui s'appuient sur des modèles assez complets (cartes par exemple) de l'environnement dans lequel évolue le robot. Cet aspect est, comme nous le verrons plus loin, absent des architectures purement réactives.

Ses points faibles sont :

- Leur principal inconvénient provient de leur faible vitesse de réaction. Elles sont en effet totalement incapables de prendre en compte des obstacles dynamiques ou non détectés lors de la modélisation. Ceci est dû à la lenteur des phases de modélisation et de planification qui ne peuvent généralement pas être exécutées en temps réel, même lorsqu'elles sont implantées sur de gros calculateurs extérieurs au robot.

- La construction du modèle sur lequel se base la planification n'est pas une tache aisée. Il est en pratique souvent difficile de relier les données sensorielles aux objets du monde réel. Ceci est dû en grande partie à l'imprécision de l'information délivrée par

les capteurs utilisés en robotique. R. A. Brooks résume ce problème en affirmant qu'il existe une grande différence entre capter et percevoir.

- Ces systèmes ne sont pas adaptatifs dans la mesure où il leur est difficile de réagir à des situations légèrement différentes de celles qu'ils connaissent. Cela exprime une lacune dans la possibilité de généraliser des situations possibles.

- Ces systèmes sont en général peu robustes de par leur modèle centralisé. Une défaillance d'un module peut provoquer le blocage de toute l'architecture.

L'intelligence artificielle traditionnelle s'est appuyée sur la volonté de reproduire des systèmes vivants, à travers des concepts plus ou moins valides issus d'observations macroscopiques empiriques, ne correspondant pas aux mécanismes réels régissant les êtres vivants.

L'utilisation de ces approches semble donc limitée aux robots évoluant dans des environnements statiques et dont la structure est fortement contrainte et connue a priori. Les limitations de cette approche ont conduit certains chercheurs à développer des systèmes réactifs.

3.2. Les contrôleurs réactives

3.2.1. Principe

Cette approche, radicalement différente de la précédente se distingue par l'abandon des phases de modélisation et de planification. Elle est née à partir de considérations issues de l'éthologie et plus particulièrement de l'entomologie. Nous savons aujourd'hui que de nombreux êtres vivants, dont en particulier les insectes, ne possèdent, en raison de leur taille que très peu de capacités de modélisation et de raisonnement. Ils n'en sont pas moins capables de mener à bien des tâches complexes. Leur comportement semble même du point de vue d'un observateur extérieur résulter d'une intelligence avancée, similaire par certains cotés à la notre. Cette constatation induit l'idée suivante : pourquoi ne pas abandonner le concept centré sur la modélisation de l'environnement et le raisonnement, au profit d'une architecture

totalement distribuée privilégiant un lien étroit entre perception et action. Arkin parle de modèle "action-oriented perception" précisant que les besoins perceptifs d'un agent doivent être basés sur ses nécessités en action. Cela s'oppose à une approche hiérarchique collectant des informations, puis travaillant à différents niveaux d'abstraction, sans vraiment prendre en considération les utilisations ultérieures qui en seront faites (on privilégie le modèle et sa mise à jour). Brooks va même plus loin qu' Arkin en affirmant qu'il n'y a pas de meilleur modèle de l'environnement que l'environnement lui-même.

En partant de cette constatation, des chercheurs ont tenté de s'inspirer de cette autre vision de l'intelligence en proposant une architecture de commande nouvelle basée sur la composition de plusieurs modules implémentant des comportements simples. Le comportement global du robot est alors le résultat de la composition des divers comportements élémentaires. Il émerge de l'interaction de ces derniers entre eux et avec l'environnement. Chacun des comportements de base, réalise un couplage étroit entre les capteurs et les effecteurs du robot. La simplicité du traitement réalisé, ainsi que la parallélisation des comportements, permettent une grande rapidité d'exécution.

Figure I.2 : contrôleur réactive

On parle généralement de systèmes de commande réactifs pour qualifier ce type de fonctionnement. On peut toutefois distinguer deux définitions assez différentes de la réactivité. L'informaticien parlera de système réactif pour désigner un système apte à réagir continûment à un environnement physique, à une vitesse déterminée par cet environnement. Dans le domaine des sciences cognitives, par contre, on dira d'un agent (i.e. robot) qu'il est réactif s'il ne possède pas de représentation explicite du monde. Ces deux définitions se recoupent toutefois en ce qui concerne les systèmes de commande réactifs, puisque ceux-ci réagissent directement aux stimuli provenant du monde extérieur en évitant la construction d'un modèle forcement entaché d'erreurs en environnement réel, ce qui autorise des vitesses d'exécution importantes. Nous pouvons également noter que si l'on s'en tient à la première définition, l'opposition entre systèmes réactifs et systèmes hiérarchiques n'est plus aussi nette. Les systèmes hiérarchiques peuvent en effet être considérés comme réactifs dans une

échelle de temps plus lente. Rien n'interdit de plus d'imaginer, la puissance des ordinateurs augmentant sans cesse, que des systèmes hiérarchiques beaucoup plus rapides pourront être réalisés dans le futur.

3.2.2. L'approche comportementale

En IA classique, les différents modules d'un système intelligent sont organisés verticalement, par grandes fonctions telles que la vision, la perception, la construction de la représentation du monde ou encore la planification. Les flux de données circulent selon un unique cycle «perception-décision-action», qui ne garantit pas la rapidité de réponse du système. L'approche «orientée-comportement» propose une organisation horizontale, dans laquelle chaque module correspond à un cycle perception-décision-action spécialisé dans la réalisation d'une fonctionnalité. Ce type d'architecture distribuée permet d'organiser des modules qui réagissent indépendamment les uns des autres à des aspects précis de l'environnement. Il ne repose pas sur un système centralisé de perception, chargé d'interpréter les signaux en provenance de l'environnement et de les traduire en données symboliques : chaque module perçoit uniquement les stimuli qui le concernent. De plus, les modules ne partagent pas de représentation centralisée de l'environnement ou de l'état de l'agent : chaque module manipule exclusivement ses données propres.

L. Steels définit un comportement d'un système (individu ou groupe) comme une «régularité» observée dans la dynamique d'interaction entre le système et son environnement. Le comportement est qualifié d'intelligent s'il permet au système de survivre au mieux dans son environnement. L'approche «orientée-comportement» tente de dégager des motifs représentatifs de l'interaction d'un agent avec son environnement et de les exprimer sous la forme de règles comportementales. Le problème de la perception et de l'action dans un environnement physique, ou tout au moins finement simulé, est important. Dans certains cas, le comportement peut émerger, c'est-à-dire être observé sans pour autant avoir été décrit explicitement sous forme de règles. L'adaptabilité et les capacités d'apprentissage sont souvent mises en exergue, car elles améliorent l'intelligence en permettant au système d'évoluer dans un environnement dynamique qu'il doit explorer.

Pour décrire les architectures basées sur les comportements, L. Steels distingue les comportements (définis précédemment), les systèmes comportementaux et les mécanismes comportementaux d'un agent. Les systèmes comportementaux sont des modules capables d'assurer une fonctionnalité particulière pour l'agent, comme l'évitement d'obstacle ou la recherche de nourriture. Pour être efficace, un système comportemental doit être adapté à l'environnement et aux objectifs de l'agent. Une fonctionnalité est réalisée par un ou plusieurs comportements, et un système comportemental peut contribuer à la réalisation de plusieurs fonctionnalités.

Un système comportemental est constitué d'un ensemble de mécanismes comportementaux. Un mécanisme comportemental est un principe ou une technique permettant de créer un comportement particulier, par exemple en couplant des capteurs et des effecteurs ou en construisant une carte cognitive. Un tel mécanisme est implémenté en utilisant des structures et des processus (capteurs, effecteurs, programmes, structures de données, *etc.*). Les mécanismes comportementaux doivent être les plus simples possibles, afin que ce soit les interactions entre ces mécanismes qui fassent émerger le comportement attendu. Pour résumer, un comportement est ce qui est observé, alors qu'un système comportemental est ce qui est implémenté dans l'agent par des mécanismes comportementaux de façon à faire émerger le comportement.

Enfin, les différents systèmes comportementaux d'un agent interagissent, directement ou indirectement, et sont généralement en compétition : pour obtenir un comportement global cohérent, il faut les coordonner. Le problème de la sélection de l'action est alors généralement celui du choix des actions à réaliser en fonction d'objectifs conflictuels. Il ne s'agit pas seulement de choisir une séquence d'actions nécessaires à la satisfaction d'un objectif prioritaire : il faut fréquemment faire des compromis, ou encore combiner des actions indépendantes. T. Tyrrell a réalisé un excellent travail de comparaison des mécanismes de sélection de l'action pour différentes architectures basées sur les comportements. Selon lui, une bonne architecture de sélection de l'action doit être capable de choisir un système comportemental possédant un certain nombre de caractéristiques, dont voici les principales :

- **motivé :** le système comportemental sélectionné correspond prioritairement à celui de plus forte motivation en fonction des objectifs et des ressources de l'agent. Par

exemple, la faim peut être provoquée par un niveau d'énergie faible ; la faim peut déclencher la recherche de nourriture, à condition qu'une activité prioritaire, telle que la fuite devant un prédateur, ne soit pas en cours.

- **interruptible :** en cas d'urgence, le système comportemental courant peut être interrompu au profit d'un autre. Par exemple, la recherche de nourriture pourra être reportée le temps d'échapper à un prédateur.
- **persistant :** s'il est pertinent, le système comportemental en cours doit continuer afin d'éviter l'oscillation entre plusieurs systèmes comportementaux éligibles. Par exemple, si l'agent est à la recherche de nourriture, il doit continuer même s'il a subitement autant soif que faim.
- **opportuniste :** si l'occasion se présente et si le système comportemental courant le permet, un système comportemental de moins forte motivation peut être sélectionné le temps de satisfaire un objectif secondaire. Par exemple, si l'agent passe à côté d'une source de boisson alors qu'il cherche de la nourriture sans être trop affamé, il pourra faire un détour pour boire puis reprendre sa recherche.
- **direct :** le système comportemental doit préférer les actions consommatoires aux actions appétitives, c'est-à-dire favoriser les actions qui mènent directement à la satisfaction d'un objectif (action finale) par rapport aux actions qui préparent une action consommatoire. Par exemple, l'agent a intérêt à manger ce qui est disponible là où il se trouve plutôt que d'aller vers une autre source de nourriture.
- **conciliant :** les actions choisies doivent correspondre de préférence à un compromis entre plusieurs solutions, de façon à satisfaire un maximum d'objectifs concurrents plutôt que de n'en satisfaire qu'un seul de façon optimale.

Pour conclure voici les avantages et les inconvénient des systèmes comportementale :

Ses points forts sont :

- Rapidité de réponse qui lui donne la capacité d'évoluer dans des environnements fortement dynamiques. La simplicité des comportements permet d'accroître encore cette vitesse par une réalisation sur circuit imprimé.

- Robustesse qui découle de la parallélisation et de l'indépendance relative des comportements qui permet au robot de continuer à fonctionner en cas de panne de l'un d'entre eux, ce qui est très important lorsque l'on a à faire à un robot autonome.

- L'aspect incrémental en niveau : il suffit de mettre en place un niveau de compétence puis d'ajouter les niveaux supérieurs sans avoir, en théorie, à remettre en question les niveaux inférieurs. Cet aspect est primordial pour la mise au point et la recherche de pannes. Cette approche permet également d'accroître la complexité du comportement global du robot sans pour autant accroître celle du contrôle global puisque celui-ci est complètement distribué.

- Sa simplicité implique un faible coût de production et des possibilités de miniaturisation importantes. Ceci ouvre de nouveaux domaines d'application. Brooks s'intéresse notamment à l'utilisation d'un grand nombre de petits robots dans le domaine spatial.

Ses points faibles sont :

- L'approche comportementale souffre d'une absence de capacité de modélisation et de raisonnement.

La naissance de l'approche comportementale a été en 1986 dans l'architecture de subsumption (SA) proposée par le chercheur Rodney Brooks qui est à la base de toutes les autres architectures.

3.2.3. L'architecture de subsumption

L'architecture de subsomption a été proposée par le chercheur Rodney Brooks pour la construction d'agents réactifs. Elle consiste à décomposer un agent en modules verticaux chacun produisant un comportement très limité. Chacun de ses comportements peut être vu comme une fonction qui prend en entrée des perceptions et produit des actions en sortie afin de réaliser une tâche précise. Un point important à noter est que ces tâches ne doivent

nécessiter pour leur résolution aucune représentation symbolique complexe, ni mécanisme de raisonnement sophistiqué.

Ceci permet de construire les couches incrémentalement en partant de la couche la plus primitive. Chaque couche dépend et communique peu avec les autres. Il s'agit plutôt d'une approche de conception montante (bottom-up) plutôt que descendante (top-down). Brooks propose une décomposition d'un système autonome basé sur les manifestations extérieures du système. Chaque couche doit provoquer un comportement observable. Chaque couche connecte ses propres senseurs aux actions et n'est pas dépendante d'aucune autre couche pour prendre des décisions. Les couches opèrent en parallèles. De nouvelles couches peuvent être rajoutées au système sans modification des couches déjà existantes.

R. Brooks définit, dans le cas d'un robot d'exploration, 8 niveaux de compétences qui sont dans l'ordre :

- Éviter les obstacles (s'arrêter avant de rentre en collision avec un obstacle immobile, s'écarter des obstacles qui s'approchent).
- Errer sans but (au hasard) sans rien toucher.
- Explorer le monde en observant à distance les emplacements qui semblent atteignables et s'y rendre.
- Construire une carte de l'environnement et des routes d'un lieu à l'autre
- Remarquer les changements dans l'environnement statique
- Raisonner sur le monde en termes d'objets identifiables et de réalisation de tâches relativement à certains objets
- Formuler et exécuter un plan qui implique un changement voulu de l'état du monde
- Raisonner sur les comportements des objets et modifier les plans en conséquence

On observe que chaque niveau de comportement inclut comme sous-ensemble le comportement des niveaux précédents. Par exemple, pour errer sans but sans heurter d'obstacles, il faut savoir éviter les obstacles. La couche de base doit être complètement implémentée et testée avant de passer à la couche supérieure. La couche de base donne un robot qui évite les obstacles s'approchant de lui s'il est immobile et qui s'arrête avant de frapper un obstacle. La deuxième couche donne un robot qui se déplace au hasard en évitant

les obstacles. Cette couche supprime la direction qui est prise par la couche qui fuit les obstacles. En fait, cette couche combine le déplacement au hasard avec l'évitement d'obstacles. La troisième couche subsume la deuxième couche en percevant le monde extérieur.

Une architecture de subsomption est basée sur les caractéristiques suivantes :

- une architecture de décomposition du comportement de l'agent en niveaux,
- les niveaux pouvant se déclencher simultanément, il y a une politique de priorité fixe entre les niveaux,
- chaque module est connecté en parallèle entre les entrées (capteurs) et les sorties (effecteurs) de l'agent,
- Les modules sont arrangés en couches dans lesquels les couches inférieures peuvent subsumer les fonctions des couches supérieures (i.e plus bas est la couche, plus grande est sa priorité),
- Les couches basses gouvernent les comportements de base et les couches supérieures ajoutent des comportements plus sophistiqués,
- Le comportement total d'un agent peut être changé en ajoutant une nouvelle couche au dessus des précédentes, sans changer les couches existantes.

Les interactions entre les modules sont fixes et s'effectuent par l'intermédiaire d'un rapport de dominance défini lors de la conception. Les modules effectuent leurs tâches en parallèles, mais si deux modules sont en conflits (i.e produisent des résultats contradictoires), alors seules les informations du module dominant sont prises en compte.

Le problème qu'il faut noter est que la priorité entre les modules de compétence est fixe, mais la priorité devra changer suivant l'environnement, et l'état du robot, ce qui a conduit à un couplage de cette architecture aux techniques d'étude qui ont été inspiré de la biologie.

3.2.4. Autres architectures réactives

L'approche comportementale a inspiré plusieurs travaux concernant aussi bien la réalisation des comportements réactifs, que la mise en place d'architectures globales permettant leur composition.

Une autre approche possible consiste à composer les commandes préconisées par plusieurs comportements. C'est notamment le principe des travaux de T. L. Anderson et M. Donath [AND 1990]. Chaque module de l'architecture qu'ils proposent représente un comportement réflexe simple ne possédant aucune capacité de mémorisation. La définition de tels comportements est donnée par D. McFarland : « *Un comportement réflexe est la forme la plus simple de réaction à une stimulation externe. Des stimuli tels qu'un changement soudain dans le niveau d'illumination, ou un contact à un endroit donné du corps, génèrent une réponse automatique, involontaire et stéréotypée* ». Ces modules génèrent des champs de potentiels dont la composition permet de calculer la direction à suivre par le robot. Un mécanisme permet de plus de sélectionner les comportements à prendre en compte en fonction de la tâche à accomplir ou de l'environnement actuel du robot.

D. W. Payton et J. K. Rosenblatt [PAY 1990] proposent d'utiliser un réseau de neurones artificiels pour fusionner les commandes préconisées par les différents comportements. Le système est cependant construit à la main de manière totalement empirique.

Certains auteurs ont également cherché à réaliser par apprentissage un système de sélection des différents comportements. Ces travaux font généralement intervenir l'apprentissage par renforcement. Dans le système proposé par M. Humphrys [HUM 1995] par exemple, chaque module comportemental apprend et met à jour une fonction donnant une évaluation de l'importance de l'action qu'il propose dans l'état courant. Le comportement présentant l'action de plus grande importance est sélectionné pour commander le robot. L. J. Lin [LIN 1993] utilise un système similaire dans lequel il construit par apprentissage une fonction donnant le comportement le plus adapté à l'état courant.

3.3. Les contrôleurs hybrides

Les approches hiérarchiques et réactives sont diamétralement opposées. Cependant, chacune présente des caractéristiques intéressantes. Pour cela, des chercheurs ont essayé de les combiner en mettant au point des architectures hybrides permettant notamment d'allier des capacités de raisonnement et de décision de haut niveau, s'appuyant sur des représentations abstraites des connaissances, avec des comportements réactifs garantissant robustesse et flexibilité.

La plupart des contrôleurs actuellement utilisés choisissent une solution intermédiaire entre ces deux approches sous la forme d'une architecture hybride. Cette notion est à distinguer de celle de système hybride en intelligence artificielle (ou système hybride neuro-symbolique) qui désigne un système alliant des composantes utilisant des techniques issues de l'intelligence artificielle symbolique classique et des réseaux de neurones artificiels.

L'architecture hybride se compose de deux niveaux. Le premier est chargé des tâches de navigation de haut niveau, telles que la localisation, la cartographie et la planification. Pour cela, il s'appuie sur un second niveau réactif qui est chargé d'exécuter les commandes avec le plus de précision possible et de gérer les éléments non modélisés de l'environnement tels que les obstacles inconnus ou dynamiques. L'action conjointe de ces deux niveaux permet de réagir rapidement face aux variations imprévues de l'environnement, tout en permettant la réalisation d'actions planifiées à plus long terme.

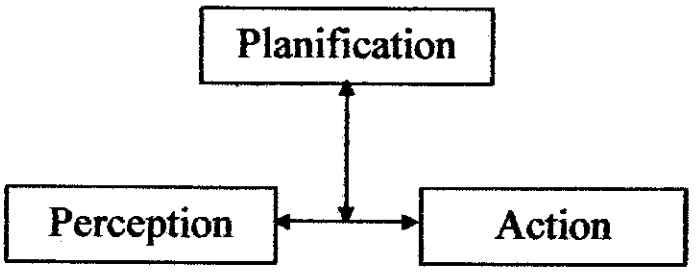


Figure I.3 : contrôleur hybride

Le bas niveau de ces architectures peut être réalisé sous forme de comportements, tels que ceux utilisés dans les architectures réactives. Ces comportements sont des boucles

sensorimotrices qui relient les actions aux perceptions avec une phase de décision très courte, qui assurent la réactivité. Dans le même temps, les informations sensorielles sont utilisées par le haut niveau dans une boucle sensorimotrice à une échelle de temps beaucoup plus longue. C'est la mise en parallèles de ces deux échelles de temps qui fait la force de ces architectures.

4. L'apprentissage en contrôle

Dans un premier temps, toute l'intelligence dans un agent, provenait de son concepteur. L'agent est alors lâché dans un environnement et fait du mieux qu'il peut selon la manière dont il a été programmé. Cette approche n'est pas nécessairement la meilleure, tant pour l'agent que pour le concepteur. En effet, le concepteur n'a qu'une connaissance incomplète de l'environnement dans lequel l'agent évolue, l'apprentissage étant alors une voie possible pour que l'agent acquièrt les informations dont il a besoin.

Les perceptions peuvent être utilisées non seulement pour agir mais aussi, pour améliorer les capacités d'action de l'agent dans le futur. L'apprentissage est le résultat de l'interaction entre l'agent et l'environnement, à travers l'observation de son propre processus de décision. Apprendre en contrôle signifie modifier le contrôleur de manière à augmenter ses performances telles qu'elles sont mesurées par des critères. Le domaine d'apprentissage est vaste, allant de la mémorisation triviale d'expériences à la modélisation de phénomènes. Un agent apprenant peut être divisé en quatre composants conceptuels représentés par la figure I.4. Voici une description de chacun d'eux :

- Le composant de contrôle est responsable de la sélection des actions. Il considère les données sensorielles et décide des actions.
- Le composant d'apprentissage est responsable de l'apport d'améliorations. Il se base sur la rétroaction (feedback), c'est-à-dire l'observation du processus de décision de l'agent, afin de déterminer les modifications à effectuer au niveau du composant de contrôle pour obtenir de meilleurs résultats futurs.
- Le composant d'évaluation des performances permet de spécifier au composant d'apprentissage, la qualité des actions de contrôle effectuées. L'évaluation est rendue possible par la définition d'un standard de performance.

- Le composant de génération de problème est responsable de la suggestion d'expériences nouvelles apportant de l'information. Il permet une exploration des différentes actions possibles qui ne sont pas optimales à court terme mais peuvent s'avérer intéressantes à longue échéance.

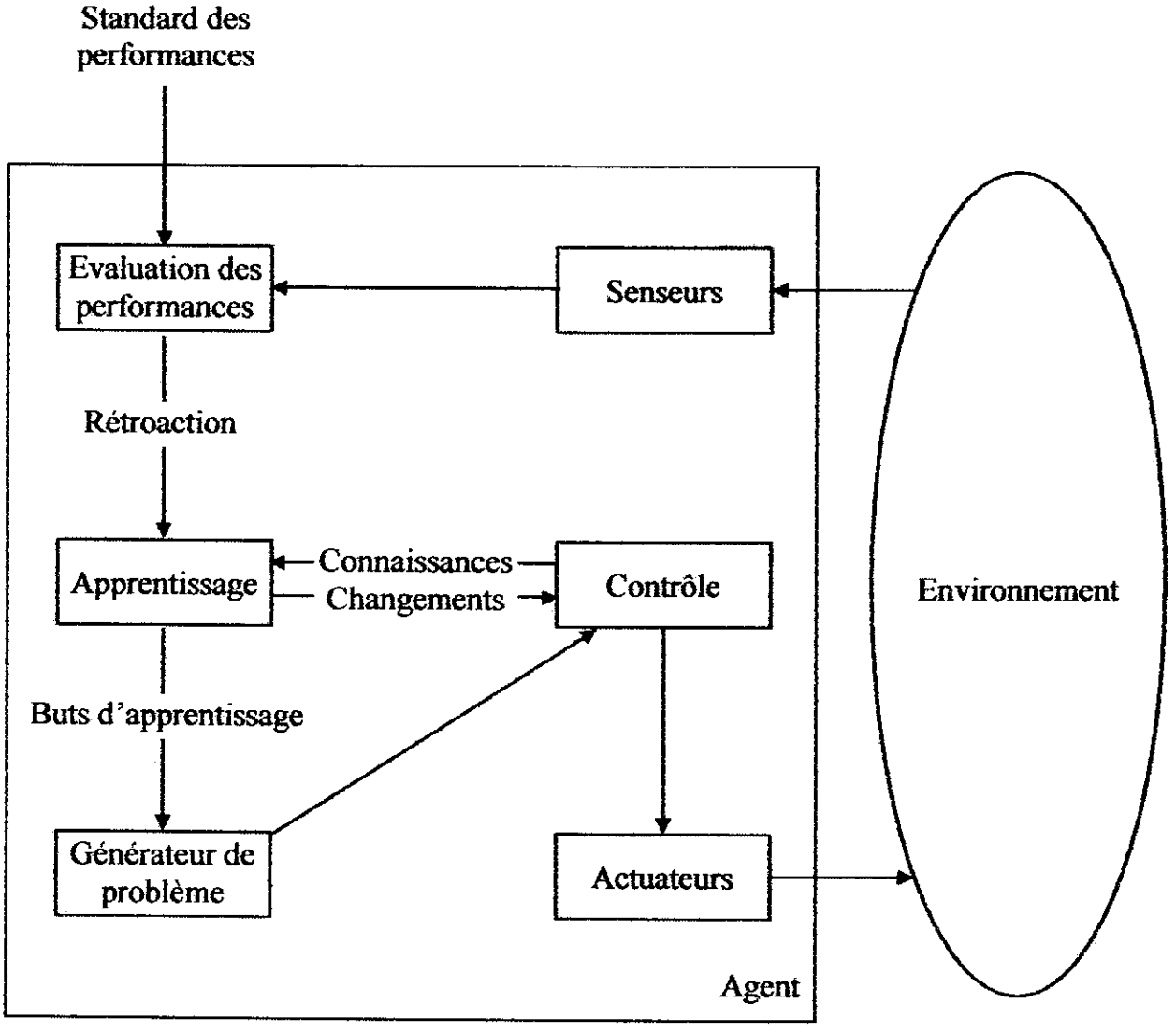


Figure I.4 : Modèle générique d'un agent apprenant.

Il existe trois grandes catégories de méthode d'apprentissage que sont l'apprentissage supervisé, l'apprentissage avec maître distant et l'apprentissage par renforcement. Elles se distinguent par le type de rétroaction disponible.

4.1. Les catégories d'apprentissage

Comme nous l'avons cité précédemment il existe trois grandes catégories :

- **L'apprentissage supervisé :** En apprentissage supervisé, le maître fournit soit l'action qui devrait être exécutée, soit un gradient sur l'erreur commise (au niveau de l'action). Dans les deux cas, le maître fournit au contrôleur une indication sur l'action qu'il devrait générer afin d'améliorer ses performances.

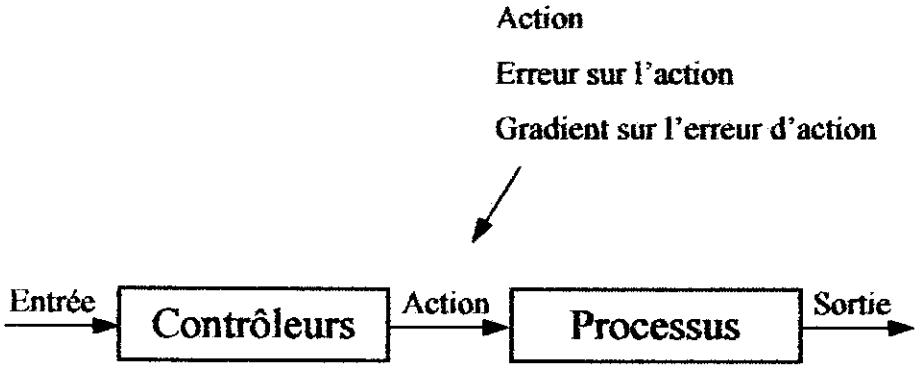


Figure I.5 : L'apprentissage supervisé

- **L'apprentissage par renforcement :** Par opposition, le maître en apprentissage renforcé a un rôle d'évaluateur et non pas d'instructeur. Il est en général appelé critique. Le rôle du critique est de fournir une mesure indiquant si l'action générée par le robot est appropriée ou non. Il laisse le contrôleur déterminer seul comment il doit modifier ses actions de manière à obtenir une meilleure évaluation dans le futur. Contrairement donc à l'apprentissage supervisé, le critique connaît les objectifs visés mais ne sait pas comment les atteindre.

Figure I.6 : L'apprentissage par renforcement

- **L'apprentissage avec un maître distant :** L'apprentissage par maître distant, comme l'on appelé Jordan et Rumelhart, est une troisième catégorie d'apprentissage située

entre l'apprentissage supervisé et l'apprentissage renforcé. Le maître est dit distant car les informations fournies pour l'apprentissage sont en relation avec la sortie du processus et pas directement avec les actions générées par le contrôleur. Ces actions sont de type "sortie", "erreur sur la sortie", ou "gradient de cette erreur". Comme dans le cas de l'apprentissage renforcé, le contrôleur doit découvrir seul l'action qu'il doit générer afin d'obtenir la sortie désirée. L'apprentissage renforcé peut alors être vu comme un cas particulier de l'apprentissage avec maître distant.

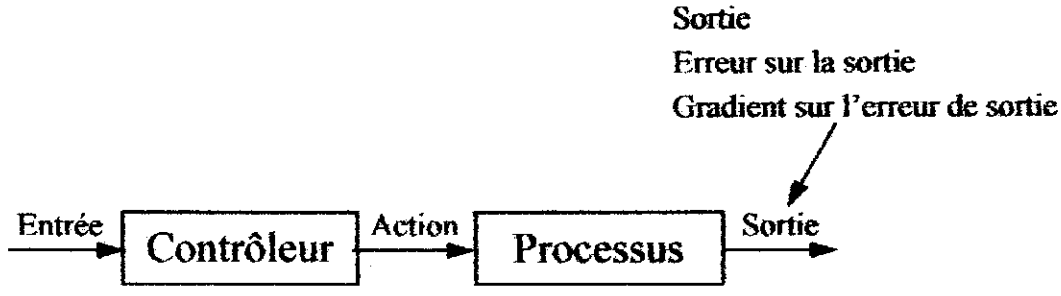


Figure I.7 : L'apprentissage avec un maître distant

4.2. Le contrôle adaptatif

Les recherches concernant la commande adaptative ont débuté vers le début des années 1950 et ont été motivées par le besoin de fournir une aide de haute performance pour le pilotage d'avions. Après une première période de relatif succès, les premiers résultats théoriques fondamentaux sont apparus vers 1960 et ont permis au domaine de réaliser de grandes avancées. De plus, les progrès rapides de la micro-électronique ont rendu possible la réalisation de tels contrôleurs de manière simple et peu coûteuse.

Un contrôleur adaptatif est un contrôleur de structure classique ou de structure plus complexe muni de coefficients ajustables dont l'ajustement à l'aide d'un algorithme convenable permet d'étendre le domaine de fonctionnement. De manière plus générale, on peut également définir un contrôleur adaptatif comme toute méthode de commande utilisant une mise à jour en temps réel du modèle mathématique du système à régler. Les différentes méthodes de commande adaptative se différencient par la manière choisie pour réaliser l'ajustement des coefficients. On distingue principalement :

- La commande adaptative à paramètres pré-programmés.

- La commande adaptative à modèle de référence.
- La commande adaptative avec identification du système à régler.

5. Conclusion

Nous avons vu dans ce chapitre les différentes architectures de contrôle pour la navigation autonome. Nous remarquons que dans les premiers contrôleurs, l'idée était de prévoir toutes les situations possibles, donc de fournir aux robots toute l'intelligence, mais il a était vérifié que cette approche était très limité, ce qui ramené les chercheurs de s'inspirer de la nature pour crée des contrôleurs qui peuvent s'adapter à l'environnement en appliquant les techniques d'apprentissage.

CHAPITRE
II

II. Systèmes immunitaires artificiels

1. Introduction

Dans le but de résoudre des problèmes complexes, des idées glanées à partir de mécanismes naturels ont été exploités pour développer des heuristiques inspirées de la nature. L'optimisation naturelle a été largement investiguée durant la dernière décennie ouvrant la voie à de nouveaux paradigmes de calcul comme les réseaux de neurones, les systèmes évolutionnaires ou encore les systèmes immunitaires artificiels. L'objectif ultime est de développer des systèmes qui ont la capacité d'apprendre de façon incrémentale tout en s'adaptant à leur environnement. L'immunologie artificielle est un paradigme récent qui tente de capturer des caractéristiques intéressantes des systèmes immunitaires naturels, comme la mémorisation, la reconnaissance de formes, l'apprentissage, et les capacités d'adaptation. Un système immunitaire artificiel a été défini comme un système adaptatif inspiré par le système immunitaire biologique pour la résolution de problèmes.

Un système immunitaire naturel (SIN) est un système complexe qui peut être analysé à différents niveaux : molécules, cellules et organes. Les interactions complexes entre les entités de chaque niveau résultent en un système complexe capable de protéger notre corps de n'importe quelle entité dangereuse appelée antigène. Les éléments de base des SIN sont les globules blancs ou lymphocytes. Pour pouvoir identifier les autres molécules, des lymphocytes particuliers appelés cellules B produisent des récepteurs appelés anticorps. La partie de l'anticorps responsable de la reconnaissance de l'antigène est appelée paratope. Le paratope se lie à une partie spécifique de l'antigène appelée épitope. La liaison entre un paratope et un épitope est d'autant plus forte que leurs formes sont complémentaires. La force de cette liaison est appelée affinité.

Les caractéristiques intéressantes des systèmes immunitaires ont encouragé leur adaptation au domaine informatique pour la résolution de problèmes du monde réel allants de la sécurité des réseaux jusqu'à la détection de changements en passant par le web mining. En particulier, l'investigation des capacités d'adaptation des systèmes immunitaires a conduit la création des modèles informatiques pour la navigation autonome des robots mobiles.

Dans ce chapitre nous présenterons le système immunitaire naturel en indiquant ses caractéristiques cognitives et en présentant la théorie des réseaux immunitaires de Jerne, puis on introduira les systèmes immunitaires artificiels en expliquant le passage vers l'artificiel.

On présentera par la suite le modèle de Farmer des réseaux immunitaires qui est le modèle le plus connu et le plus utilisé. Enfin on abordera l'application des réseaux immunitaires à la navigation autonome des robots mobiles.

2. Système immunitaire naturel

2.1. Principe du fonctionnement

Le système immunitaire protège l'organisme contre l'invasion de corps étrangers. Il est constitué de cellules différentes réparties dans tout l'organisme. Chaque catégorie de cellules a une fonction spécifique et se déplace dans l'organisme selon les besoins.

La réponse immunitaire fait intervenir deux types de mécanismes qui sont d'apparitions successives au cours de l'évolution des espèces et sont intimement connectés :

- L'immunité naturelle non spécifique, encore appelée innée ou naïve, repose sur une distinction globale du soi et du non-soi. C'est une réponse immédiate, non spécifique de l'agresseur et non adaptative.
- L'immunité acquise spécifique, également appelée immunité saisie ou adaptative, représente la partie du système immunitaire qui peut identifier spécifiquement et éliminer sélectivement le micro-organisme et les molécules étrangères elle se caractérise par une spécificité antigénique, diversité, et mémoire immunologique.

Les phagocytes (une partie du système inné) sont capables de détruire de nombreux antigènes (envahisseurs) sur le contact. Le système immunitaire adaptatif utilise des lymphocytes qui peuvent rapidement s'adapter, afin de détruire les antigènes qui sont entrés dans la circulation sanguine. Une différence importante entre ces deux systèmes est que les cellules adaptatives sont spécifiques aux antigènes et ont une plus grande capacité de mémoire que les cellules innées. Il existe deux types principaux de lymphocytes, à savoir les B-cellules et les T-cellules, qui jouent un rôle remarquable dans les deux immunités (humorale et cellulaire). Les B-cellules participent à l'immunité humorale et sécrètent des anticorps par la prolifération clonale tandis que les T-cellules participent à l'immunité communiquée par les cellules. Une classe des T-cellules, appelé les T-cellules tueur, détruit la cellule infectée toutes

les fois qu'elle identifie l'infection. L'autre classe qui déclenche l'expansion clonale et stimule ou supprime la formation d'anticorps s'appelle les T-cellules d'aide. Les lymphocytes flottent librement dans des noeuds de sang et de lymphe.

Quand un microbe pathogène étranger infectieux attaque le corps humain, le système immunitaire inné est lancé en première ligne de défense. L'immunité innée n'est pas dirigée vers des envahisseurs particuliers, plutôt contre tout microbe pathogène qui envahit le corps. Elle s'appelle l'immuno-réaction non spécifique. La cellule la plus importante dans l'immunité innée est un phagocyte, y compris le monocyte, le macrophage, etc... Un phagocyte intériorise et détruit les envahisseurs du corps humain. Ensuite le phagocyte devient un antigène présentant la cellule (APC). L'APC interprète l'annexe d'antigène et extrait les dispositifs, en traitant et en présentant les peptides antigéniques sur sa surface aux T-cellules et aux B-cellules. Ces peptides antigéniques sont un genre de molécule appelé MHC (complexe principal d'histocompatibilité) et peuvent distinguer le "soi" du "non-soi" (antigènes).

Les lymphocytes reconnaîtront l'antigène et réagissent. Les T-cellules d'aide relâchent les cytokines qui sont des signaux proliférative agissant sur les B-cellules. Les B-cellules deviennent stimulées et créent des anticorps quand elle identifie l'antigène. L'identification est réalisée par l'attache intercellulaire, qui est déterminée par la forme moléculaire et la charge électrostatique. Les anticorps peuvent être distribués dans tout le corps. Le paratope d'un anticorps peut se lier avec l'épitope d'un antigène (voir figure II.1) selon son affinité. L'affinité se rapporte au degré de liaison du récepteur des cellules avec l'antigène. Plus l'affinité est haute plus l'attache est fort et ainsi plus l'identification et la réponse immunisées sont meilleures. Les T-cellules d'aide fonctionnent, principalement, à côté des substances de sécrétion, connues sous le nom de cytokines. Ils constituent des puissants messagers chimiques. Les cytokines favorisent la croissance, le lancement et le règlement cellulaires. En outre, les cytokines peuvent également détruire des cellules cibles et des macrophages stimulés.

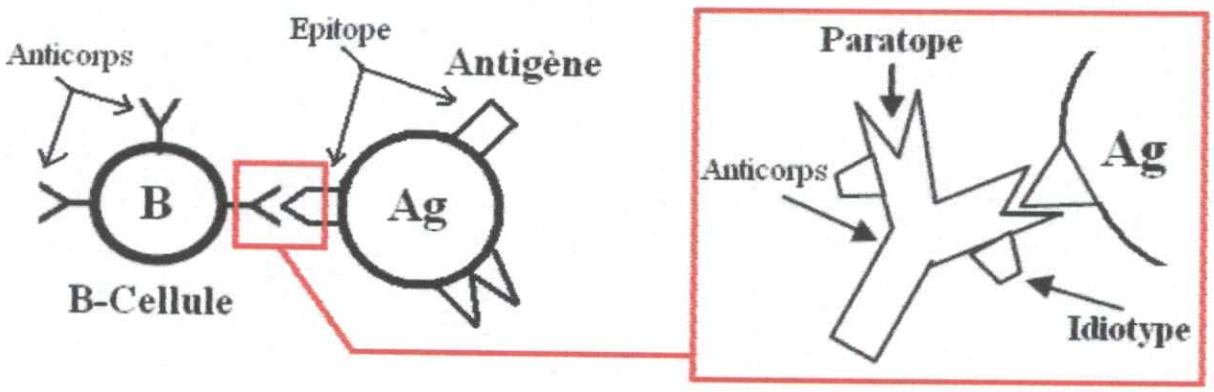


Figure II.1 : Interaction entre l'anticorps et l'antigène

Après identification réussite d'un antigène par une B-cellule avec une certaine affinité, la cellule est choisie pour le clonage. Il y'aura une production d'anticorps en un volume élevé puis leur libération de la surface des B-cellules pour faire face aux envahisseurs. La prolifération dans le cas des cellules immunisées est asexuelle, les cellules se divisent (il n'y aura aucun croisement). Un important mécanisme se produit lors du clonage des cellules, c'est *la maturation d'affinité* qui combine deux processus *la mutation* qui affecte le récepteur qui se lie avec l'antigène et *le choix* qui garantit la survie des fils qui présente le meilleur degré d'affinité à l'antigène. Donc la maturation d'affinité (voir figure II.2) a pour rôle l'amélioration de la réponse à l'antigène et la création des cellules de mémoires qui serviront à des futures attaques et ainsi ont peut dire que le système s'est adapté à l'attaque.

La figure II.2 illustre le mécanisme d'adaptation. Dans le cas (a) la B-Cellule n'a pas reconnu l'antigène et aucune réaction ne s'est produite. Dans le cas (b) la B-Cellule a reconnu l'antigène, elle sera affecté par la mutation qui changera la structure de son anticorps. Dans le cas (d) l'affinité entre l'anticorps de la B-Cellule et l'antigène n'a pas été améliorée mais dans le cas (c) la mutation a considérablement amélioré l'affinité et c'est pour ça que ces B-Cellule seront cloner afin de créer des cellules de mémoire pour améliorer la réponse à de futures attaques par le même antigène.

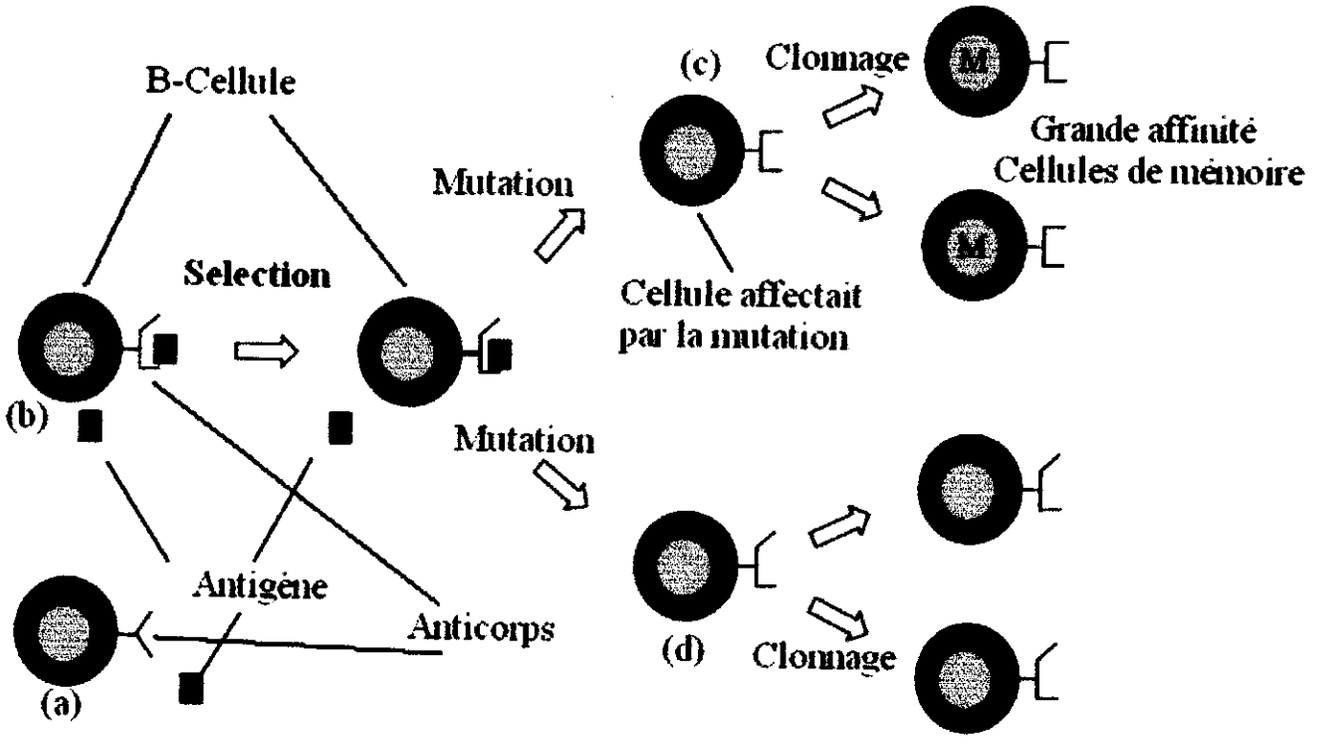


Figure II.2 : Prolifération et maturation d'affinité

2.2. Caractéristiques des systèmes immunitaires

Le rôle classiquement attribué à l'immunité est de protéger le soi de l'étranger. Le système immunitaire est donc censé se défendre contre les agressions extérieures grâce à des mécanismes qui permettent d'éliminer le non-soi. Or, pour établir un rôle défensif, le système immunitaire doit exhiber des propriétés qui sont typiquement cognitives : la reconnaissance des formes moléculaires étrangères, l'apprentissage de nouvelles formes, la mise en mémoire de ces formes. Ces trois types de processus montrent que le système est un réseau cognitif et pas seulement un mécanisme de défense car il se caractérise par :

- ***L'Identification :*** Le système immunitaire a la capacité de reconnaître, identifier et répondre à un vaste nombre de différents modèles d'antigène. En plus, le système immunitaire peut faire la différence entre les cellules de l'individu et les cellules étrangères.

- ***Extraction de dispositif*** : Par l'utilisation des antigènes de présentation des cellules (APC), le système immunitaire a la capacité d'extraire des dispositifs de l'antigène, avant d'être présentée à d'autres cellules immunisées, y compris les lymphocytes.

- ***Diversité :*** Il y a deux processus principaux impliqués dans la génération et l'entretien de la diversité dans le système immunitaire. Le premier est la génération des molécules de récepteur par la recombinaison des segments de gène des bibliothèques de gène. En recombinant des gènes d'un ensemble fini, le système immunitaire est capable de produire un nombre presque infini de types de récepteurs, de ce fait le système immunitaire est doté d'une grande assurance dans l'univers des antigènes. Le deuxième processus, qui assiste la diversité dans le système immunitaire, est connu en tant que hyper mutation somatique. Les cellules immunisées se reproduisent en réponse aux antigènes envahissants le corps. Pendant cette reproduction, elles sont soumises à un processus somatique de mutation avec des taux élevés qui permettent la création des modèles de molécules de récepteurs, de ce fait augmente la diversité des récepteurs immunisés.

- ***Étude :*** Le mécanisme de l'hyper mutation somatique suivi d'un mécanisme de sélection permet également au système immunitaire d'améliorer sa réponse à un microbe pathogène envahissant le corps, ce processus est nommé la maturation d'affinité. La maturation d'affinité garantit que le système immunitaire améliore sa tâche d'identifier les modèles d'antigène pour que la réponse sera plus rapide.

- ***Mémoire :*** Après une immuno-réaction à un antigène donné, quelques ensembles de cellules et molécules seront dotés d'une grande durée de vie afin de fournir des immuno-réactions plus rapides et plus puissantes à des futures infections par le même antigène ou à des antigènes semblables. Ce processus, connu comme maturation de l'immuno-réaction, permet l'entretien de ces cellules et molécules. C'est le principe principal des procédures de vaccination dans la médecine et l'immunothérapie. Un échantillon affaibli ou mort d'un antigène (par exemple, un virus) est inoculé dans un individu afin de favoriser une immuno-réaction (sans les symptômes de la maladie) afin de produire des cellules et des molécules de mémoire à cet antigène.

- ***Détection distribuée :*** Il y a une distribution inhérente dans le système immunitaire. Le contrôle n'est pas centralisé, chaque cellule immunisée est spécifiquement stimulée et répond aux nouveaux antigènes dans n'importe quel endroit.

- ***Autorégulation :*** La dynamique du système immunitaire est telle que la population de système immunitaire est commandée par des interactions locales mais pas par un point central de commande. Après qu'une maladie ait été combattue avec succès, le système immunitaire revient à son état d'équilibre normal, jusqu'à ce qu'il soit nécessaire de répondre à un autre antigène. La théorie immunitaire du réseau explique explicitement ce type de mécanisme autorégulateur.

- ***Métadynamique :*** Le système immunitaire crée constamment de nouvelles cellules et molécules, et élimine ceux qui sont trop vieux ou qui ne sont pas d'une grande utilité. Métadynamique est le nom donné à ces fonctions.

Ces caractéristiques intéressantes ont poussé un grand nombre de chercheurs à essayer de les reproduire, mais la construction d'un système informatique qui reproduit toutes ces caractéristiques est impossible et c'est pour cela les chercheurs essayent de reproduire un sous ensemble de cet énorme ensemble. Dans de le domaine de la navigation autonome des robots mobiles ils essayent de reproduire l'étude, la mémoire, l'autorégulation et la métadynamique.

2.3. Théorie des réseaux immunitaires

Le système immunitaire d'un individu donné dans son état de sollicitation minimum, exprime à l'état latent l'ensemble de ses capacités et maintient un certain état d'équilibre. Cet état d'équilibre se trouve perturbé par une sollicitation immunogène qui stimule une fraction du répertoire latent qui provoque la multiplication et différenciation des cellules effectrices et des cellules régulatrices. Cet état d'excitation n'est que provisoire et progressivement un nouvel état d'équilibre est trouvé enrichi de la mémoire de la sollicitation passée. La fonction du réseau de régulation interne est double : le maintien permanent du répertoire et le contrôle de l'expression de ce répertoire.

Comment s'établit le répertoire ?

Par des mutations somatiques et des recombinaisons. On pourrait se demander pourquoi les possibilités de reconnaissances sont elles si grandes alors que le nombre d'antigènes qu'aura à affronter l'organisme est si peu nombreux ? Quel est l'intérêt d'un aussi vaste potentiel de réponses immunitaires ? → Théorie de Jerne

En 1970, Jerne avançait l'idée que les organes lymphoïdes primaires éliminent les anticorps qui reconnaissent le soi et sélectionnent ceux qui reconnaissent le non soi. Cette première réflexion n'exploite pas l'idiotypie mais exprime déjà une idée qui sera l'un des éléments fondamentaux de la théorie du réseau : le système fonctionnel se différencierait sous la pression sélective des cellules du soi, avant toute stimulation clonale par les antigènes.

La théorie du réseau idiotypique (1974) a pour base des faits expérimentaux établis : qu'il existe dans l'organisme des anticorps capables de reconnaître le soi. Ces anticorps peuvent certes, à des concentrations plus importantes engendrer des maladies auto-immunes mais pas à leur niveau de circulation normale. Mais ce qui est important est que les anticorps qui sont censés effectuer la discrimination soi/ non soi sont eux-mêmes des éléments du soi. Il doit donc exister des anticorps d'anticorps (anticorps anti idiotypiques). Les interactions entre les anticorps signifient que les anticorps circulants et ceux affichés à la surfaces des cellules ne sont pas indépendants mais qu'ils sont étroitement connectés : c'est une organisation en réseau. On se rendait ainsi compte que le système pouvait présenter une dynamique propre.

Imaginons qu'un antigène (E) entre dans l'organisme. On doit prendre en compte que les anticorps s'attachent aux idiotopes des anticorps anti E . Ces anticorps seront eux mêmes reconnus par d'autres anticorps et ainsi de suite. L'antigène pourra entrer dans le réseau dans la mesure où circule déjà un anticorps de profil moléculaire semblable au sien : une "image interne". L'antigène ne devient qu'une petite perturbation dans le réseau qui a une activité continue. Le système immunitaire fondamentalement, ne discrimine pas le soi du non soi (mais sens et non-sens), le réseau ne peut être perturbé que par de nouveaux antigènes ressemblant à ce qu'il contient déjà. Donc tout antigène qui perturbe le réseau immunitaire est par définition un antigène de l'intérieur et ne fera que changer la dynamique du réseau.

L'idée de Jerne est montrée schématiquement dans la figure II.3 L'idiotype Id1 de l'anticorps 1 stimule les B-Cellules 2, par le paratope P2. Du point de vue de l'anticorps 2, l'idiotype Id1 de l'anticorps 1 est considéré comme antigène. En conséquence, l'anticorps 2

supprime les B-Cellules 1. D'autre part, l'anticorps 3 stimule l'anticorps 1 puisque l'idiotype Id3 fonctionne comme antigène par rapport à l'anticorps 1. De cette façon, les chaînes de stimulation et de suppression des anticorps forment un réseau à grande échelle qui fonctionne comme système de reconnaissance d'individu et de non individu. Par conséquent, le système immunitaire peut être prévu pour fournir de nouvelles approches parallèles de traitement distribué.

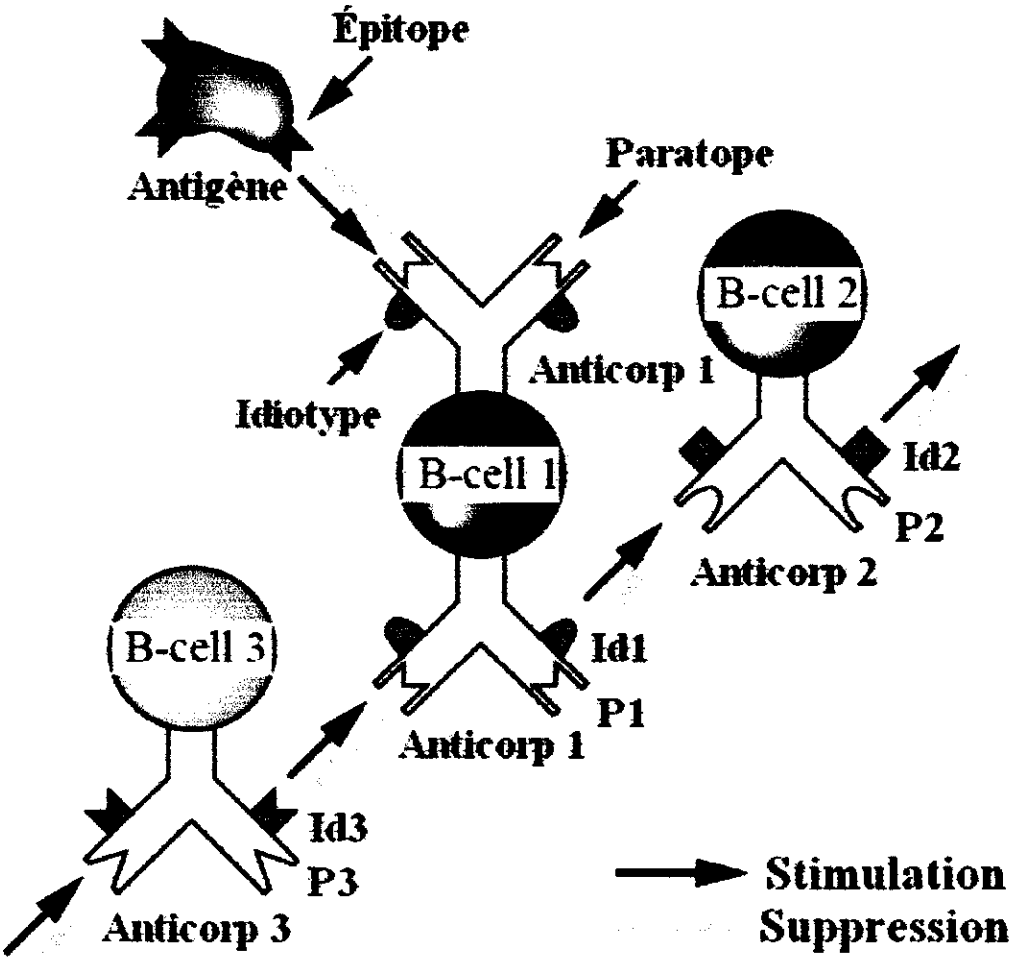


Figure II.3 : Hypothèse du réseau idiotypique de Jerne

3. Systèmes immunitaires artificiels

3.1 Passage du naturel vers l'artificiel

L'établissement de la zone des systèmes immunitaires artificiels (AIS) a été difficile pour un certain nombre de raisons. D'abord, le nombre de personnes actives dans la zone de

recherches est encore petit. Deuxièmement, les chercheurs ont trouvé des difficultés a identifier la différence entre un AIS et un travail en immunologie théorique. Troisièmement, les domaines d'application des systèmes immunitaires artificiels sont très larges. En conclusion, seulement très récemment que le premier manuel proposant un cadre général pour concevoir l'AIS a été édité.

Pour concevoir des solutions basées sur les systèmes immunitaires artificiels il faut suivre les étapes suivantes :

- ***Description du problème*** : Identifier tous les éléments qui feront partie de l'AIS. Ceci inclut des variables, des constantes, des agents, des fonctions, et des paramètres nécessaires pour décrire et résoudre le problème.
- ***Choix d'un certain principe immunitaire a utiliser*** : Plusieurs principes, modèles et théories immunitaires peuvent être employés dans différents contextes pour développer les outils de calcul pour la résolution de problèmes.
- ***Modéliser le système immunitaire artificiel*** : Ceci concerne des aspects, tels que décider quels composants immunisés vont être utilisés, comment représenter (créer les modèles abstraits de ces composants), et l'application des principes immunitaires (algorithmes) qui contrôleront le comportement du système.
- ***Traduire de l'AIS vers le problème réel*** : Après la résolution du problème, il est parfois nécessaire d'interpréter (ou décoder) les résultats présentés par le système immunitaire artificiel dans le domaine réel du problème.

Ce qui peut être résumé par quatre étapes :

- Extraction des idées et des métaphores ;
- Définition d'une représentation pour les composants de l'AIS ;
- Définition d'un ensemble de fonctions qui déterminent les interactions des éléments de l'AIS entre eux et avec l'environnement ;
- Définition des algorithmes immunitaires pour régir la dynamique de l'AIS.

3.2. Les algorithmes des AIS

Afin de créer des approches inspireés du système immunitaire, des algorithmes immunitaires ont été proposé. Ils peuvent être classifié suivant la figure II.4. Nous

présenteront brièvement les algorithmes du choix négatif et de la sélection clonale et nous détaillerons par la suite l'algorithme du réseau immunitaire car notre travail se base sur cet algorithme, dans le cas continu.

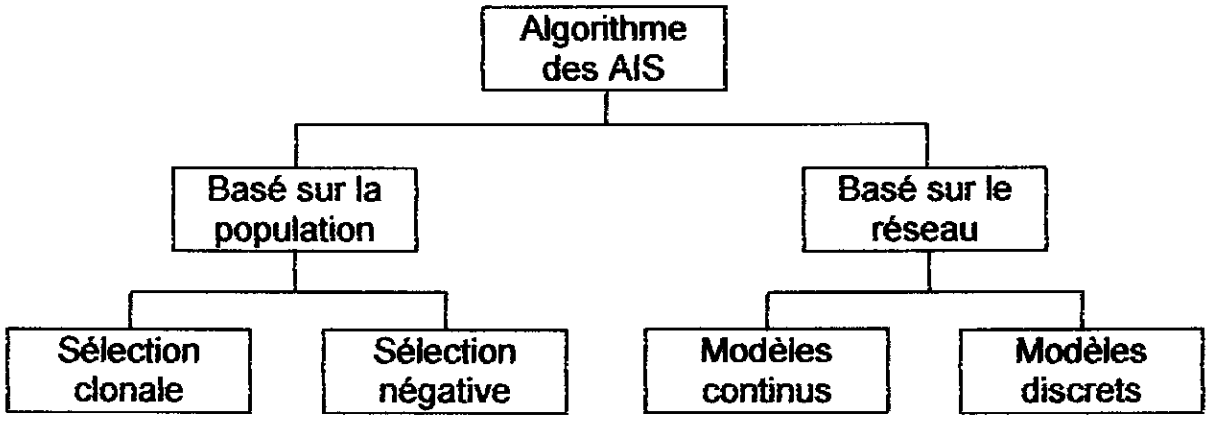


Figure II.4 : Classification des algorithmes de l'AIS

3.2.1. Algorithme de la sélection clonale

Quand les anticorps d'une B-cellule se lient avec un antigène, la B-cellule devient activer et commence a proliférer. Les nouvelles B-cellules qui sont produites sont des copies exactes de la B-cellule parente, mais elles subissent le mécanisme d'hyper mutation somatique pour produire des anticorps qui sont spécifiques à l'antigène envahissant. Le principe de la sélection clonale est le terme employé pour décrire les propriétés de base d'une immuno-réaction adaptative à un stimulus antigénique. Elle établit l'idée que seulement les cellules capables d'identifier un stimulus antigénique prolifèreront et subiront un choix qui ne garde que les clones qui ont les plus grands degrés d'affinité.

Ce genre d'algorithme peut être utilisé pour des problèmes d'optimisations, de clustering, ou de reconnaissance des formes.

L'algorithme de la sélection clonal est :

1. Produire un ensemble de solutions (répertoire d'anticorps) de N candidat qui sont défini par le problème à étudier ;
2. Choisir les n_1 cellules qui ont la plus grande affinité à l'antigène ;
3. Copier (produire des copies identiques de) ces cellules choisies. Le nombre de copies est proportionnel aux affinités : plus l'affinité est haute, plus le nombre de clone est grand;
4. Changer la structure des cellules choisies (hyper mutation). Le taux de changement est proportionnelle à leurs affinités : plus l'affinité est haute, plus le taux de Changement est petit ;
5. Sélectionner les n_2 cellules (du résultat de l'étape 4) qui ont la plus grande affinité à l'antigène pour composer le nouveau répertoire ;
6. Remplacer quelques cellules qui possèdent des valeurs d'affinité faible par les nouvelles cellules ;
7. Répéter les étapes 2 à 6 jusqu'à ce qu'un critère d'arrêt donné soit rencontré.

Algorithme général de la sélection clonal

3.2.2. Algorithme de la sélection négative

La sélection négative est le processus qui permet de distinguer le soi du non soi, elle a été appliquée à des problèmes de détections d'anomalies, son algorithme a la forme suivante :

1. Produire aléatoirement des anticorps et placez les dans un ensemble P ;
2. Déterminer l'affinité de touts les anticorps dans P avec toutes les cellules de l'individu S ;
3. Si l'affinité d'un anticorps *de* P avec au moins une cellule de S est supérieur ou égal à un seuil donné ε, alors l'anticorps identific l'individu et doit être éliminée; si non l'anticorps est accepté.

Algorithme général de la sélection négative

3.2.3. Algorithme du réseau immunitaire

La théorie immunitaire de réseau semble très attrayante pour n'importe quel chercheur sur l'intelligence artificielle. Car premièrement, elle présente un système dynamique capable d'exécuter des interactions entre ses propres constituants et entre ses constituants et l'environnement externe. Deuxièmement, elle offre les possibilités d'ajuster la structure du système (réseau) et les paramètres du système sur l'environnement. Son algorithme général est :

1. **Initialisation** : initialiser un réseau de cellules immunisées ;
2. **Boucle de population :** pour chaque antigène :
   - **2.1. Identification antigénique :** calculez les affinité des anticorps face à l'antigène ;
   - **2.2. Interactions du réseau :** calculez les affinité entre tous les paires d'anticorps ;
   - **2.3. Métadynamique :** ajoutez des nouveaux anticorps aux réseaux et supprimez qui sont inutile (le choix est basé sur un certain critère) ;
   - **2.4. Niveau de stimulation :** évaluez le niveau de stimulation de chaque cellule du réseau tenant compte des résultats des étapes précédentes ;
   - **2.5. Dynamique de réseau :** mettre à jour la structure et les paramètres du réseau selon le niveau de stimulation de différentes cellules ;
3. **Cycle :** répéter l'étape 2 jusqu'à ce qu'un critère donné d'arrêt soit rencontré ;

Algorithme général du réseau immunitaire

3.3. L'apprentissage dans les réseaux immunitaires

Il a été proposé que les réseaux immunitaires peuvent être considérés comme des systèmes cognitifs [Varela et al 1988] et offrant des possibilités d'apprentissage. Cette proposition a été justifiée par quatre raisons :

(i) ils peuvent identifier des formes moléculaires ;

(ii) ils se rappellent des rencontres ;

(iii) ils définisent les frontières des molécules de l'individu, et

(iv) ils peuvent créer des molécules qui peuvent identifier des molécules étrangères inconnues.

Ce travail suggère que le réseau immunitaire soit capable de produire des modèles dynamiques de l'activité du réseau et qu'il y a un mécanisme de fonctionnement autorégulateur qui aide à maintenir cette structure de réseau. Ces modèles du réseau immunitaire sont caractérisés par le changement des concentrations des B-cellules qui subi la sélection clonale à chaque réponse contre un antigène. Les auteurs emploient le terme "métadynamique" pour ce processus, qui signifie essentiellement la production et la mort continuelles des cellules et des molécules immunisées. Une grande variété de nouvelles B-cellules sera produite, mais elles ne seront pas toute une addition utile au système immunitaire et beaucoup n'entreront jamais dans la dynamique du système immunitaire (l'interaction entre les B-cellules dans le réseau) et mourront par la suite. Les auteurs ont produit un modèle simple en utilisant ces idées et ont constaté qu'il y a des oscillations dans plusieurs des variables dans leur système, en particulier le nombre de B-cellules qui sont produites. Ils ont remarqué qu'il y avait souvent une production rapide des B-cellules, suivie d'une descente pointu en nombre. Les auteurs ont observé qu'un certain noyau et une structure stable de réseau émerge avec le temps. Cette structure émerge en raison d'une organisation individuelle topologique dans le réseau, le résultat est que le réseau a pu enregistrer l'histoire de la rencontre avec des antigènes. Par conséquent, les auteurs ont conclu que le réseau immunitaire est un excellent système pour l'apprentissage de nouvelles situations et peut supporter une mémoire des rencontres précédentes par l'utilisation de l'assortiment des modèles complexes et la propriété de l'organisation individuelle du réseau.

En 1990 Bersini et Varela ont soutenu les idées présentées ci-dessus. Ils ont mis en application le modèle et ont suggéré que les mécanismes de la mémoire immunisée,

l'adaptabilité et la capacité d'exécution distribué pourraient être très utiles à la résolution des problèmes, en particulier la commande adaptative. À la suite de leur travail, en 1994, Bersini et Valera ont exploré la dynamique et la métadynamique du réseau immunitaire. Ils concluent que la métadynamique du système immunitaire permet de préservée les caractéristiques dans le temps, en permettent toujours de s'adapter à de nouvelles situations. Les simulations d'un réseau immunitaire ont confirmé ceci.

Bersini et Varela [Bersini et al 1990] ont proposé des principes généraux qui peuvent être extraits à partir des réseaux immunitaires pour être appliqués à la création des contrôleurs dans le domaine de la commande adaptative, mais ils espèrent trouver d'autres champs d'application.

4. Modélisation des réseaux immunitaires

La théorie du réseau immunitaire proposée par Jerne (1974), a supposé que le système immunitaire a un comportement dynamique même en l'absence des antigènes. Cette proposition était différente des théories de la sélection clonale et de la sélection négative, car elle a supposé que les B-cellules étaient capables de s'identifier. Ceci doterait le système immunitaire d'un réseau de communication entre les anticorps des B-cellules.

Plusieurs immunologistes étaient intéressés à créer des modèles des réseaux immunitaires afin de présenter de nouvelles manières pour expliquer comment les systèmes immunitaires fonctionnent. Une fois que les chercheurs de l'intelligence artificiel découvre ces travaux, l'intérêt a été grand pour appliquer ces nouveaux modèles pour résoudre des problèmes de différents domaines. Les premiers modèles de réseau ont été principalement basés sur des ensembles d'équations dirigeant les variations des tailles dans les populations d'anticorps et les autres cellules. Ces travaux sont classés parmi les modèles continus de réseau. Ils ont été largement répandus par la communauté d'AIS dans les applications telles que la robotique, l'optimisation et le contrôle.

Les réseaux immunitaires ont également servi d'inspiration pour le développement des modèles de réseau pour d'autres applications et principalement pour l'analyse des données. Ces modèles sont classés parmi les modèles discrets de réseau, car ils ne sont pas basés sur des équations, mais sur des procédures itératives d'adaptation.

4.1. Le modèle de Farmer

Dans leur modèle Farmer et le autres ont fait beaucoup de simplification afin de créer un modèle simple qui présente les propriétés caractéristiques essentielle du réseau. Ils ont ignoré des éléments important dans le système immunitaire tels que les T-cellules et les macrophages pour se concentrer sur les éléments qui sont essentiels dans les réseaux immunitaires.

La première simplification qu'ils ont faite est qu'ils ont supposé que les anticorps et les antigènes sont composés de deux types d'acides aminés donc les structures d'anticorps et d'antigènes sont représentées par des chaînes binaires. Farmer a simplifié encore en supposant que chaque antigène et chaque type d'anticorps a seulement un seul épitope.

L'existence d'une réaction entre deux différents anticorps ou entre un anticorps et un antigène est déterminé par le calcul de l'affinité entre les deux chaînes binaires. Puisque deux molécules n'ont pas besoin d'être exactement complémentaires pour réagir l'une sur l'autre, les deux chaînes binaires peuvent s'assortir en n'importe quel alignement possible. Par exemple prenant une chaîne de longueur l_e qui représente un épitope, et une autre chaîne de longueur l_p qui représente un paratope. L'affinité entre ces deux chaînes est égale a :

$$m_{ij} = \sum_{K} G(\sum_{N} e_i(n+k) \wedge p_j(n) - s+1)$$

Tel que :

- $e_i(n)$ représente la valeur du nième bit de la chaîne du i ème épitope ;
- $p_j(n)$ représente la valeur du nième bit de la chaîne du j ème paratope ;
- $\wedge$ représente l'opération ou exclusive ;
- s représente le seuil de l'assortiment. Une valeur au dessus d'elle deux anticorps peuvent réagir ;
- G(x) représente une fonction qui mesure la force d'une réaction possible entre l'épitope et le paratope pour un alignement donné. Elle est égale à :

  x si x >0

  0 si non

Quand une B-cellule identifie un épitope d'un antigène, elle se divise et produit des anticorps libres. Les résultats de ce processus est qu'il y'aura des anticorps libres et des B-cellules avec le même type d'anticorps attaché à leur surface. Pour la simplification, Farmer a gardé que la concentration x_i d'un anticorps i et a ignoré les B-cellules. Pour modéliser la dynamique du réseau Farmer a utilisé l'équation :

$$x_i` = c\left[\sum_{j=1}^{N} m_{ji}\, x_i\, x_j - k_1 \sum_{j=1}^{N} m_{ij}\, x_i\, x_j + \sum_{j=1}^{n} m_{ji}\, x_i\, y_j\right] - k_2\, x_i$$

Tel que :

- x_i représente la concentration de l'anticorps i ;
- c est une constante ;
- m_{ij} l'affinité entre le paratope i et l'épitope j ;
- k_1 est une constante d'inégalité entre la stimulation et la suppression ;
- k_2 représente le coefficient de la mort normale de l'anticorps ;
- N représente le nombre d'anticorps ;
- n représente le nombre d'antigènes ;

Dans cette équation la variation de la concentration des anticorps est de la forme :

Taux de variation de population = *stimulation de réseau* − *suppression de réseau* + *afflux de nouveaux éléments* − *la morts des éléments non stimulés*

Une caractéristique essentielle du modèle de Farmer est que les listes des anticorps et des antigènes sont dynamiques, donc de nouveaux anticorps peuvent être ajouté et même des anciens anticorps peuvent être supprimé (N et n peuvent changer). Ils fixent une valeur minimum pour toutes les concentrations, s'ils trouvent une concentration qui descend au dessous de la valeur fixer ils suppriment l'anticorps. Ceci simule la mort de la dernière cellule d'un type donné d'anticorps.

Farmer utilise des algorithmes génétiques pour créer des nouveaux anticorps afin de les ajouter au système, en prenant comme population les chaînes binaires des anticorps.

5. L'application des réseaux immunitaires aux robots mobiles

Les réseaux immunitaires ont été utilisés pour résoudre le problème de la navigation autonome pour atteindre un but en évitant les obstacles. Ils consistent à donner au robot les moyens d'effectuer ses déplacements de manière autonome, c'est-à-dire sans faire appel à l'homme. Plusieurs travaux ont été mené dans ce domaine par : Ishiguro 1998, Watanabe 1999, Michelan & Von Zuben 2002, Hart 2003 et Vargas 2003.

L'idée de base est que le système sera composé de deux parties :

- Un ensemble de comportements de base (par exemple : avancez, tournez à gauche, etc.) qui forme les modules de compétence.
- Un mécanisme d'arbitrage de ces modules.

Les modules de compétence sont définis comme anticorps. L'interaction entre ces modules de compétence est représentée par la stimulation et la suppression entre les anticorps. L'état actuel où se trouve le robot est défini comme un antigène, qui est composé par un ensemble d'épitope qui présenteront une information particulière (par exemple : obstacle à gauche, but à droite, etc.). Le robot contiendra donc un réseaux d'anticorps qui sera perturbé par l'information collecté par les capteurs qui joucra le rôle d'antigène et il exécutera une immuno-réaction pour choisir l'action à exécuter. Ce choix est basé sur les concentrations des anticorps, l'anticorps dominant (qui va être choisi) sera celui qui possédera la plus grande valeur de concentration.

L'algorithme général qui réalise l'arbitrage du comportement est :

Algorithme mécanisme d'arbitrage

Début

Tant que (vrai)

Faire

- Collection des informations à partir des capteurs ;
- Construction des épitopes de l'antigène ;
- Calcul des affinités entre l'antigène et tous les anticorps ;
- Calcul des nouvelles concentrations des anticorps ;
- Choix de l'anticorps dominant et exécution de son action ;
- Exécutez la fonction d'ajustement du réseau qui modifiera l'affinité de l'anticorps choisit ;
- Exécution de la fonction de métadynamique qui modifiera les anticorps (élimine ou crée) ;

Fait

Fin

6. Conclusion

Nous avons vu dans ce chapitre que les systèmes immunitaires possèdent un grand nombre de caractéristiques cognitives comme la mémoire, la reconnaissance des nouvelles formes, l'adaptabilité et le traitement distribué de l'information donc on peut dire que les systèmes immunitaires forment un excellent outil pour la création des contrôleurs adaptatifs, particulièrement pour la navigation réactive de robots.

CHAPITRE

III

II. Conception, Réalisation et Tests

1. Introduction

On est bien d'accord qu'aujourd'hui, pour construire un système informatique, il est difficile de s'attaquer directement à l'implémentation, mais au contraire, il faut d'abord modéliser le système.

La modélisation consiste à créer une représentation simplifiée d'un problème : le modèle. Grâce au modèle il est possible de représenter simplement un problème, un concept et le simuler. La modélisation comporte deux composantes :

- L'analyse, c'est-à-dire l'étude du problème.
- La conception, soit la mise au point d'une solution au problème.

Le modèle constitue ainsi une représentation possible du système pour un point de vue donné.

Dans ce chapitre nous présenterons les étapes les plus importantes dans la conception et la réalisation de notre système puis nous montrerons les résultats des tests qu'on a faits en utilisant le simulateur WSU Khepera Robot Simulator v7.2 développé par Steven Perreta et John C. Gallagher.

2. Conception et réalisation du système

2.1. Les différents éléments nécessaires

Le but de ce projet est de construire un contrôleur réactif, qui fait la sélection d'action pour la navigation autonome d'un robot mobile. Afin de réaliser cet objectif, on a définit les différents éléments qu'on aura besoin pour cette réalisation. Ces éléments se résument par :

- **Un ensemble d'actions :** ce sont les comportements de base (par exemple : avancez, tournez à droite, tournez à gauche, . . . etc.) qui forme les modules de compétences et qui sont les candidats du mécanisme de sélection.

- **Une fonction de filtrage :** elle prend comme paramètre les valeurs numériques retourné par les capteurs. Elle analyse ces valeurs et sélectionne à partir de l'ensemble

d'actions, un sous ensemble d'actions qui peuvent être exécutées par le robot en respectant les contraintes d'évitement d'obstacles.

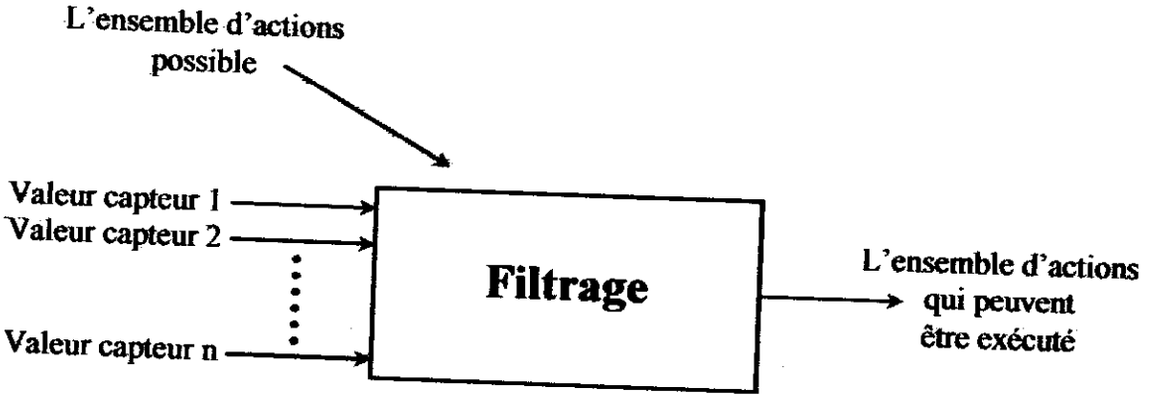


Figure III.1 : Diagramme de la fonction du filtrage

- **Une fonction d'exécution d'action :** elle assure la transformation de l'action qui a été choisi en un ensemble de commande pour le robot. Elle prend en argument le type de l'action a exécuter, et retourne les valeurs des vitesses et le temps nécessaire à l'exécution de cette action.

Figure III.2 : Diagramme de la fonction d'exécution des actions

- **Un mécanisme de sélection :** à partir du sous ensemble d'actions déterminé par la fonction de filtrage, il sélectionne l'action dominante qui va être envoyé à la fonction d'exécution d'action. Ce choix est basé sur un certain ensemble de paramètres qui seront modifiés par un mécanisme d'adaptation afin d'assurer un comportement intelligent et autorégulateur.

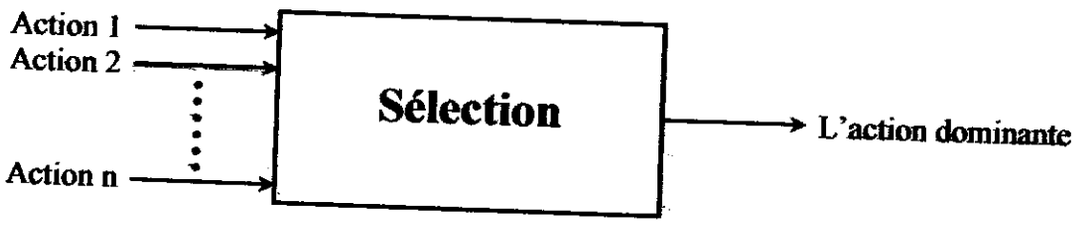


Figure III.3 : Mécanisme de sélection

- **Un mécanisme d'adaptation :** Pour plus d'utilité, l'introduction d'un mécanisme d'adaptation est fortement indispensable. Le mécanisme d'adaptation est habituellement classifié selon deux types : ajustement et innovation.

  L'ajustement est une adaptation réalisée en changeant les paramètres du système.

  L'innovation est une adaptation réalisée par les changements topologiques du système.

  Donc pour réaliser un contrôleur adaptatif, on doit faire un mécanisme d'adaptation, qui emploie au moins un type d'adaptation (l'ajustement ou l'innovation).

2.2. Le choix du principe immunitaire

Notre système se base sur la théorie de Jerne. Notre choix est inspiré des principes des réseaux immunitaires car :

- premièrement elle présente un système dynamique capable d'exécuter des interactions entre ses propre constituants et entre ses constituants et l'environnement externe.
- Deuxièmement, elle offre les possibilités d'ajuster la structure du système (réseau) et les paramètres du système sur l'environnement.

Un réseau immunitaire se définit par :

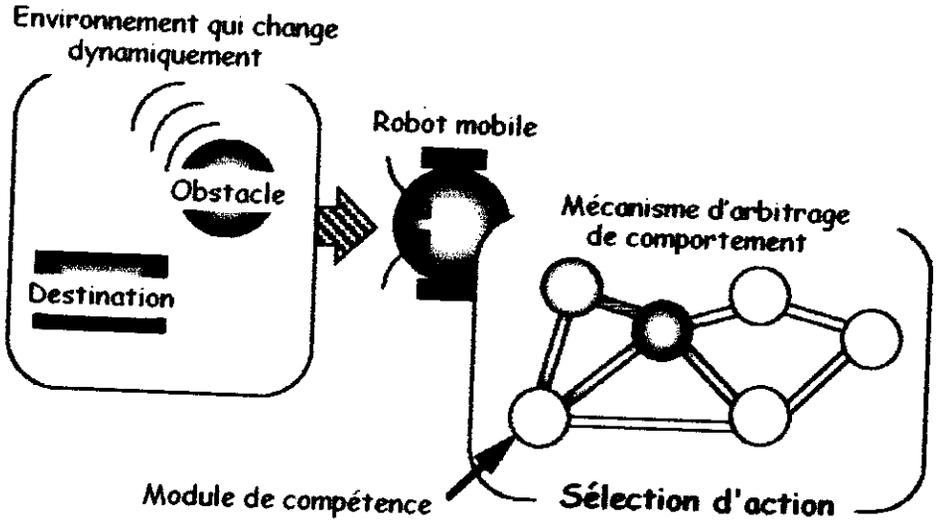
- **Sa structure :** décrit les types d'interactions et les relations entre ses composants cellulaires.

- **Sa dynamique :** décrit la variation par rapport au temps des concentrations et des affinités des cellules composants le réseau.

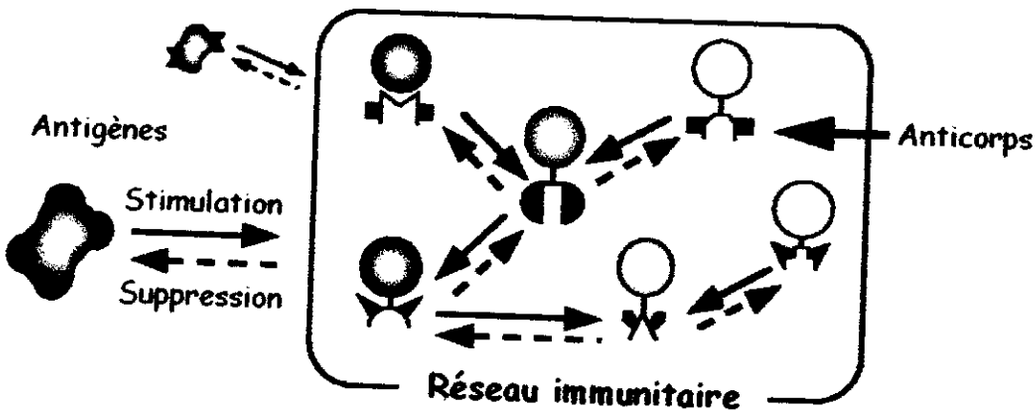
- **Sa métadynamyque :** décrit la production continu des anticorps et la mort des cellules non stimulé ou par des réactions d'individu.

Le robot correspond donc à une organisation d'anticorps (qui représente les actions) qui sera perturbé par l'information collectée par les capteurs qui jouera le rôle d'antigène et il exécutera une immuno-réaction pour choisir l'action a exécuter. Ce choix est guidé par les concentrations des anticorps. L'anticorps dominant (qui va être exécuter) sera celui qui possédera la plus grande valeur de concentration.

Dans la figure III.4 la partie (a) représente l'architecture globale d'un contrôleur réactif et la partie (b) représente l'architecture globale d'un réseau immunitaire. On peut remarquer qu'il y a une grande similitude entre les deux architectures.



(a) Contrôleur réactif pour des robots mobiles autonomes



(b) Architecture d'un réseau immunitaire

Figure III.4 : similitude entre un contrôleur réactif et un réseau immunitaire

2.3. Environnement de travail

Nous avons utilisé le robot Khepera dans notre projet. Les tests ont été développés avec le simulateur *WSU Khepera Robot Simulator v7.2* développé par Steven Perreta et John C. Gallagher à l'université de Wright State. Ce logiciel freeware peut être téléchargé à l'adresse : *http://gozer.cs.wright.edu/classes/ceg499/sim/WSU_Sim_v7.2.zip*. Il a pour but de tester visuellement des algorithmes de commande pour les robots. Il permet aussi de piloter à distance un robot réel.

2.3.1. Le robot Khepera

Khepera (voir l'annexe B) est un robot miniature de forme circulaire destiné à la recherche, développé au laboratoire de micro informatique (LAMI) de l'école Polytechnique de Lausanne (voir Figure III.5). D'un diamètre de 55mm, d'une hauteur de 30mm, Khepera pèse 86g. Il a été développé afin de tester les algorithmes de planification de trajectoire, d'évitement d'obstacles, de prétraitement des données de capteurs et de contrôle de comportement en robotique dans le monde réel.

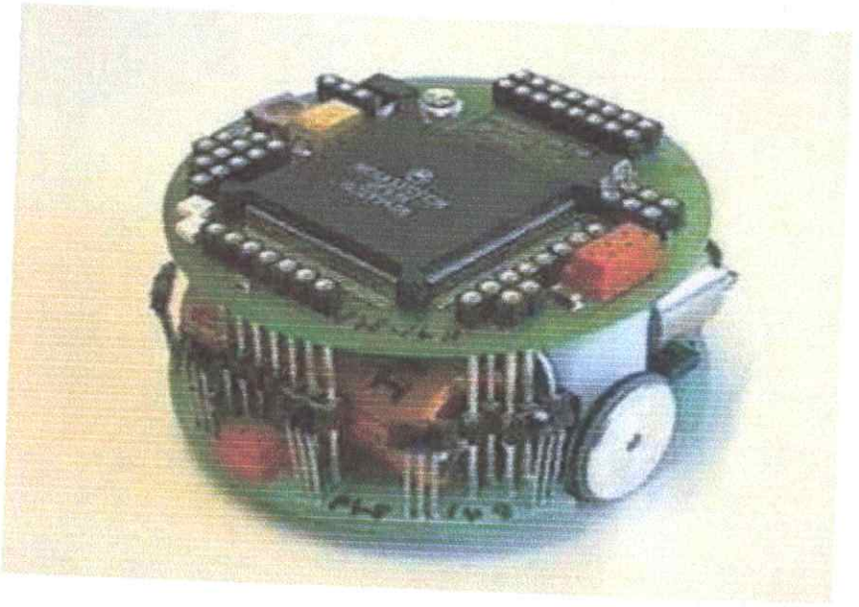


Figure III.5 : Le robot Khepera

Khepera est constitué de deux roues et 8 capteurs infrarouges (IR). Ces derniers permettent la détection des obstacles (émission/réception IR) et des sources lumineuses (réception IR). Six capteurs sont placés à l'avant du robot et les deux autres à l'arrière (voir Figure III.6). Pour la mesure de distance, les capteurs renvoient des valeurs numériques comprises entre 0 (lorsque le robot est à une distance supérieure à 5cm de l'obstacle) et 1023 (lorsque le robot est à une distance inférieure à 2cm de l'obstacle). Pour la mesure d'intensité lumineuse, l'intervalle est de 50 (intensité forte) à 520 (intensité nulle).

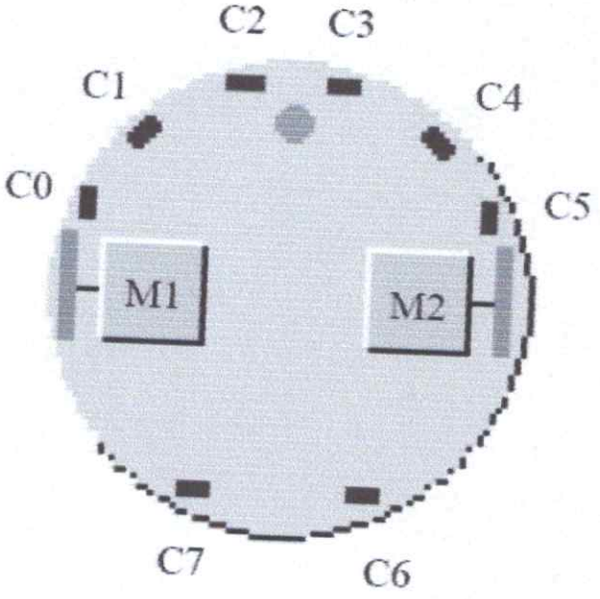


Figure III.6 : Emplacement des capteurs, des moteurs et des roues du robot Khepera

La commande des moteurs entraînant les roues est donnée par des valeurs comprises dans un intervalle entre -10 (tout vitesse vers l'arrière) et +10 (tout vitesse vers l'avant). Les deux moteurs sont indépendants, ce qui permet de faire des rotations par l'intermédiaire d'une commande utilisant deux valeurs opposées (exemple pour +5,-5 il réalise une rotation en vitesse moyenne dans le sens horaire).

2.3.2. Le simulateur de Khepera

Le *WSU Khepera Robot Simulator v7.2* (voir l'annexe C) a été développé par Steven Perreta et John C. Gallagher à l'université de Wright State avec le langage Java. Le simulateur est doté d'une interface graphique conviviale (voir figure III.7) qui permet de créer des environnements personnalisés et d'y déplacer le robot très facilement. L'environnement simulé est équivalent à un espace plan de 1m sur 1m.

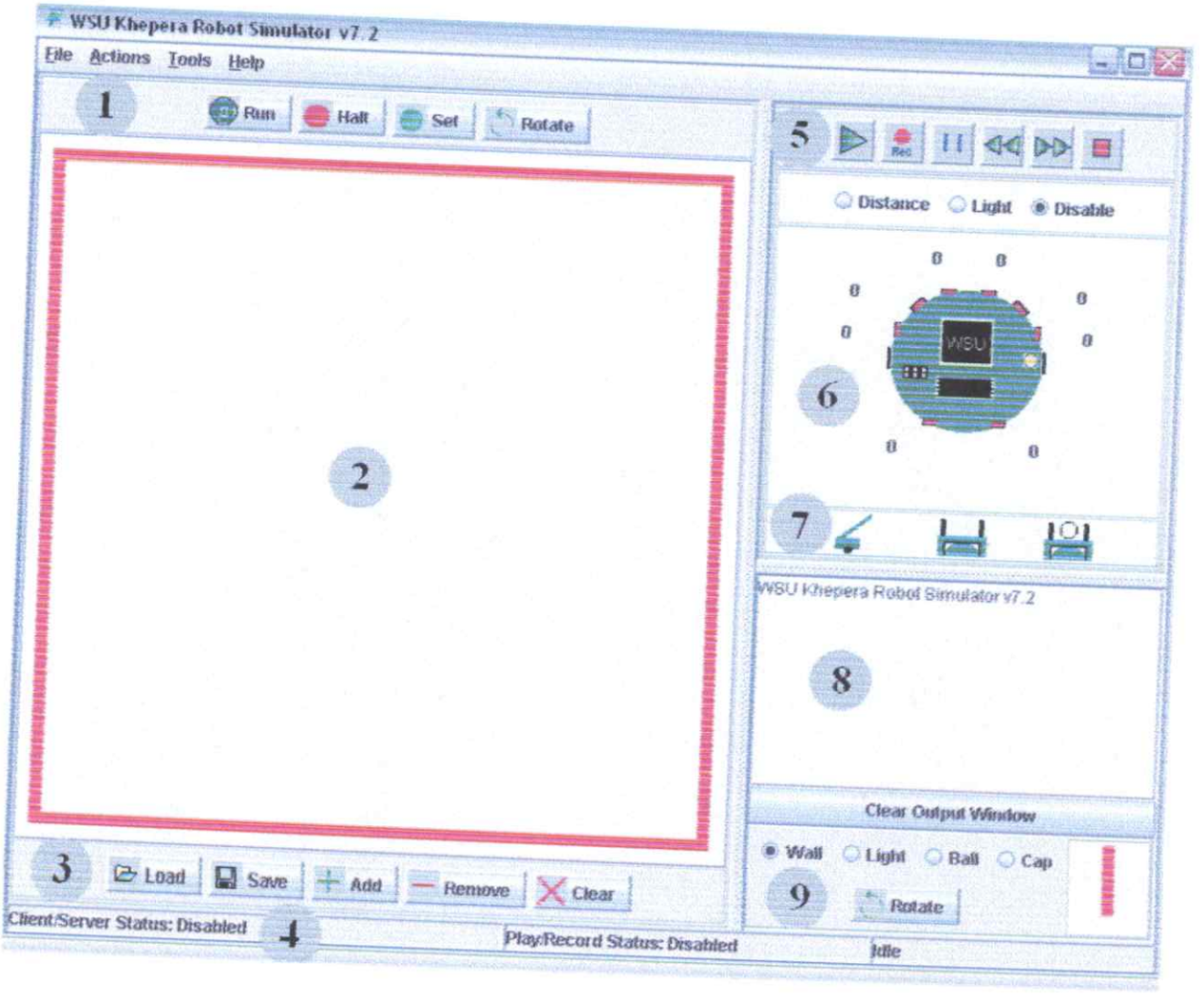


Figure III.7 : L'interface du simulateur

1) Commande du robot, 2) L'environnement du robot, 3) Commande de l'environnement, 4) Barre de statut, 5) Record/Playback, 6) Affichage des capteurs, 7) Bras du robot, 8) Sortie pour le contrôleur, 9) Choix d'objets.

Toutes les informations sur l'état du robot (par exemple valeurs des capteurs de distance, valeurs de la commande passée aux moteurs) sont facilement accessibles par des fonctions fournies par le simulateur. L'annexe C présente une description des fonctions disponibles et de leurs utilisations.

2.4. Modélisation

2.4.1. Les composants immunitaires utilisé

Le système immunitaire naturel est un système complexe qui possède un grand nombre de cellules différentes. Ses éléments de base sont les phagocytes et les lymphocytes qui sont constitué par deux types, les B-cellules et les T-cellules. Le système immunitaire adaptatif utilise des lymphocytes qui peuvent rapidement s'adapter.

Les cellules les plus utilisé dans l'immunité adaptative sont les B-cellules. Elles produisent des anticorps qui peuvent rapidement changez leur structure suivant les antigène qui envahissent le corps. Pour des raisons de simplification nous utiliserons que des B-cellules et plus précisément que des anticorps qui seront caractérisé par leurs concentrations et leurs forces d'affinités par rapport à d'autre anticorps.

2.4.1.1. Les anticorps

Les anticorps sont des récepteurs produits par les B-cellules, ils sont responsables de la reconnaissance des antigènes. La force de la liaison entre un anticorps et un antigène est définie par l'affinité qui se rapporte au degré de la liaison entre les deux récepteurs. Elle est d'autant plus forte que leurs forces sont complémentaires.

Dans notre projet, les anticorps représente les différentes actions (modules de compétence) que peut exécuter le robot. Ils se caractérisent par :

- une concentration qui est une valeur comprise entre 0 et 1, elle indique un rapport du nombre de cet anticorps dans le réseau.

- le type de l'action qui est une valeur naturelle qui indique le numéro de l'action. Les actions sont définies par un angle qui est un multiple de 15. L'angle compris entre deux actions successive est égale à 15, par exemple le numéro de l'action qui permet au robot d'avancer est égale à 90/15 = 6. Donc on aura 360/15 = 24 anticorps (actions) différents.

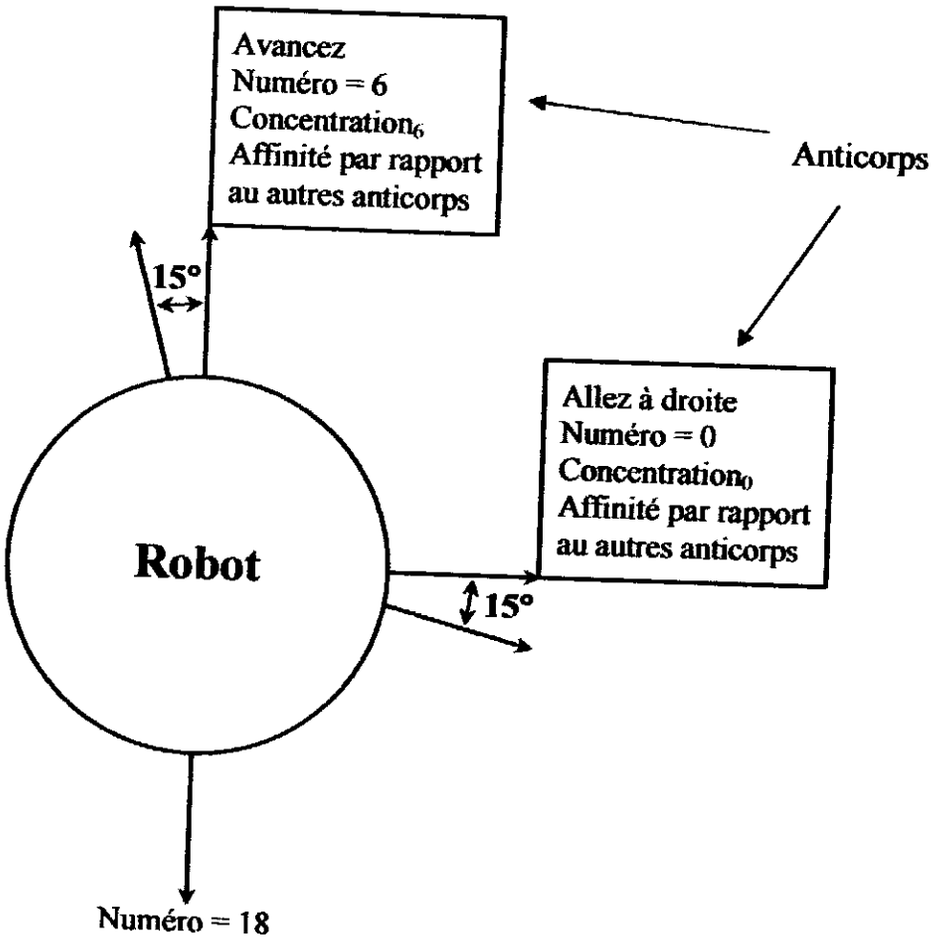


Figure III.8 : Représentation des anticorps

2.4.1.2. Les antigènes

Les antigènes sont les molécules étrangères infectieuses qui attaquent le corps. Dans notre projet, l'antigène représente l'état actuel de l'environnement du robot, cet état est obtenu en collectant les informations fournies à partir des capteurs du robot.

L'antigène possède six valeurs naturelles retourné par les capteurs pour chaque anticorps, il les obtient à partir des six premiers capteurs du robot et en faisant tourner le robot suivant le numéro de l'anticorps pour passer aux six valeurs suivantes.

<table>
<tr><td>Capteur 1
Capteur 2
Capteur 3
Capteur 4
Capteur 5
Capteur 6</td><td>Capteur 1
Capteur 2
Capteur 3
Capteur 4
Capteur 5
Capteur 6</td><td>Capteur 1
Capteur 2
Capteur 3
Capteur 4
Capteur 5
Capteur 6</td><td>Capteur 1
Capteur 2
Capteur 3
Capteur 4
Capteur 5
Capteur 6</td></tr>
<tr><td>Anticorps 1</td><td>Anticorps 2</td><td>Anticorps 3</td><td>Anticorps 4</td></tr>
<tr><td>Capteur 1
Capteur 2
Capteur 3
Capteur 4
Capteur 5
Capteur 6</td><td>Capteur 1
Capteur 2
Capteur 3
Capteur 4
Capteur 5
Capteur 6</td><td>Capteur 1
Capteur 2
Capteur 3
Capteur 4
Capteur 5
Capteur 6</td><td>Capteur 1
Capteur 2
Capteur 3
Capteur 4
Capteur 5
Capteur 6</td></tr>
<tr><td>Anticorps 5</td><td>Anticorps 6</td><td>Anticorps 7</td><td>Anticorps 8</td></tr>
<tr><td colspan="2">X</td><td colspan="2">Y</td></tr>
</table>

Figure III.9 : Représentation des antigènes

2.4.2. Les différentes étapes du contrôleur

Notre contrôleur réalise la sélection d'action pour un robot mobile autonome, il est basé sur l'approche comportementale, qui réalise un couplage direct entre les capteurs et les moteurs, le fonctionnement du contrôleur peut être décrit par cinq étapes :

a) Collection des données
b) Filtrage des actions
c) Choix de l'action dominante
d) Exécution de l'action dominante
e) Application du mécanisme d'adaptation

Ces cinq étapes seront détaillées par la suite.

2.4.2.1. Collections des données

La collection des données est une étape très importante, elle permet de percevoir l'environnement du robot et de positionner les obstacles dans cet environnement pour les éviter. Le robot Khepera possède huit capteurs mais nous utiliserons que ces six premiers capteurs pour une haute précision. La réalisation de l'étape de collection suit les étapes suivantes :

- L'initialisation de la direction du robot pour qu'il soit dirigé vers l'anticorps 0, c-a-d vers le nord, et cela en lisant sa direction et le faisant tourner jusqu'il atteint la direction de l'anticorps 0. Par exemple si le robot avait une direction qui est égale à 135°, pour initialiser sa direction, on affecte au moteur de la roue gauche une valeur positive +4 et au moteur de la roue droite une valeur positive -4 et on exécute cette commande pendant 0.8s.

- Le positionnement de la direction du robot suivant le numéro du premier anticorps de l'ensemble des anticorps possible, et cela en calculant la différence entre le numéro de l'anticorps actuel et le numéro de l'anticorps cible, donc on obtiendra l'angle entre les deux directions. Enfin on fait tourner le robot de la même manière de l'étape de l'initialisation en changeant le temps de l'exécution de la commande pour qu'il corresponde à l'angle désiré.

- La lecture des six valeurs retournées par les six premiers capteurs, et cela en appelant une fonction du simulateur ("int getDistanceValue(int *sensorID*)" de la classe "RobotController" tel que *sensorID* représente le numéro du capteur).

- La répétition des deux dernières étapes jusqu'à la lecture de toutes les données pour tout l'ensemble des anticorps.

- La lecture de la position actuelle du robot.

2.4.2.2. Filtrage des actions

L'ensemble des actions possible est constitué par huit anticorps. Pour exécuter une action elle doit respecter la contrainte d'évitement d'obstacle, et c'est pour cela qu'on filtre cet ensemble avant de choisir l'action dominante.

Pour savoir si une action peut être exécuté, on utilise les données collectées par l'étape précédente. Les capteurs du robot sont des capteurs infrarouges, il se caractérise par un angle et une distance de détection. La figure III.10 illustre le champ de vision des six premiers capteurs du robot.

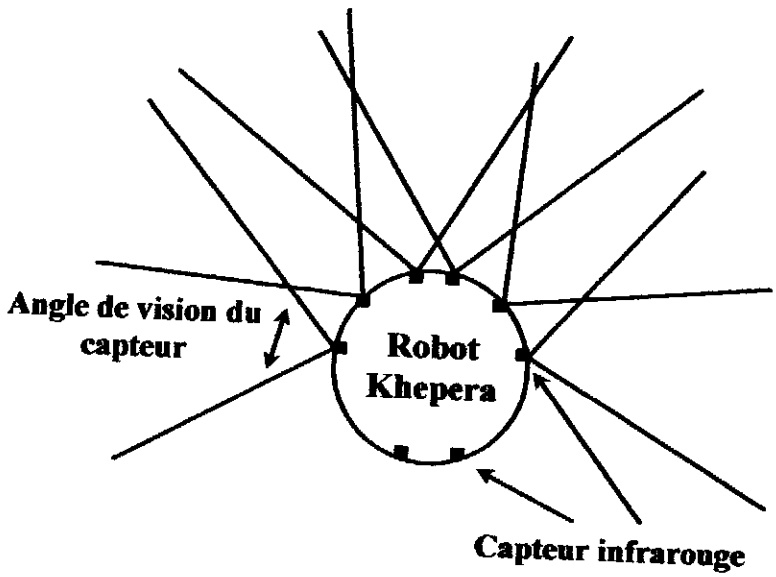
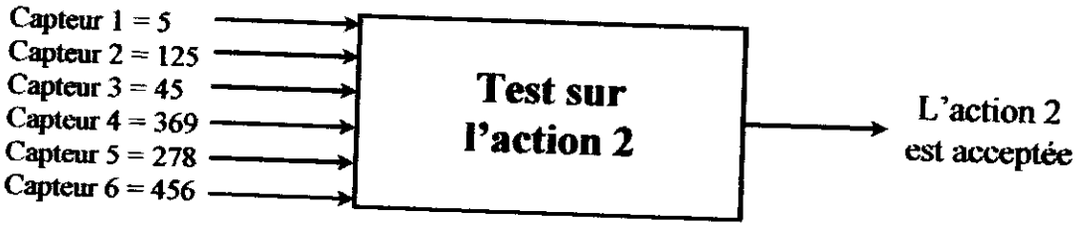


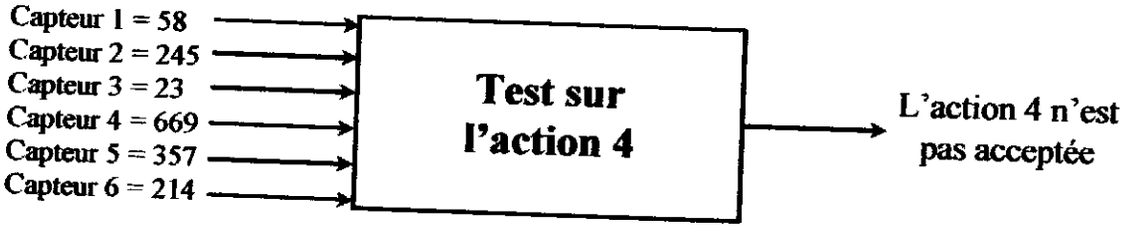
Figure III.10 : Champ de vision du Khepera

Les valeurs retournées par les capteurs sont des entiers numériques compris entre 0 (lorsque le robot est à une distance supérieure à 5cm de l'obstacle) et 1023 (lorsque le robot est à une distance inférieure à 2cm de l'obstacle). Les actions qu'on a réalisées font avancer le robot de 2cm, donc les obstacles doivent se situer à une distance supérieure à 2cm. Pour filtrer l'ensemble des actions possible on teste pour chaque anticorps les six valeurs des capteurs, si elles sont tous inférieur à la valeur 600 (pour assurer que la distance entre le robot et l'obstacle le plus proche soit supérieur à 2cm) l'action fera objet d'une candidature au mécanisme de sélection.

Dans la figure III.11, deux actions sont présentées à la fonction de filtrage, l'action 2 et l'action 4. Dans la partie (a), les valeurs des capteurs collectés pour l'action 2 sont tous inférieur à 600, donc l'action 2 vérifie les contraintes et fera l'objet d'une candidature au mécanisme de sélection, mais dans la partie (b), on remarque que la valeur du capteur 4 est supérieur à 600 donc l'action 4 ne sera pas candidate dans le mécanisme de sélection et elle ne pourra pas s'exécuter à l'état courant du robot.



(a) l'action 2 vérifie les conditions



(b) l'action 4 ne vérifie pas les conditions

Figure III.11 : Mécanisme de filtrage des actions

2.4.2.3. Choix de l'action dominante

Pour chaque état, on doit choisir une action qui fait passer le robot d'un état à un autre état, ce choix est fait après la construction de l'antigène. En présentant l'antigène au réseau il y aura une immuno-réaction qui perturbera le réseau et il y aura u changement de concentration des anticorps. Le choix de l'action se base sur les nouvelles concentrations des anticorps, l'action dominante sera l'action de l'anticorps qui possède la plus grand valeur de concentration donc le choix est dirigé par la dynamique du réseau qui décrit la variation par rapport au temps des concentrations des cellules du réseau, elle est liée aux interactions entre les anticorps, et les réponses contre les antigènes, elle a la forme suivante :

Taux de variation de population = stimulation de réseau – suppression de réseau + afflux de nouveaux éléments – la morts des éléments non stimulés

La sélection de l'action passe par deux étapes :

a) **Etape 1 :** Le calcul de l'affinité entre l'antigène et tous les anticorps du réseau. L'affinité entre un anticorps et un antigène est égale à :

- **0** Si l'action de l'anticorps ne peut pas être exécuter (présence d'un obstacle)
- **(1+cosθ)/2** Si non, tel que θ est l'angle entre l'orientation de l'action et l'orientation du but a atteindre

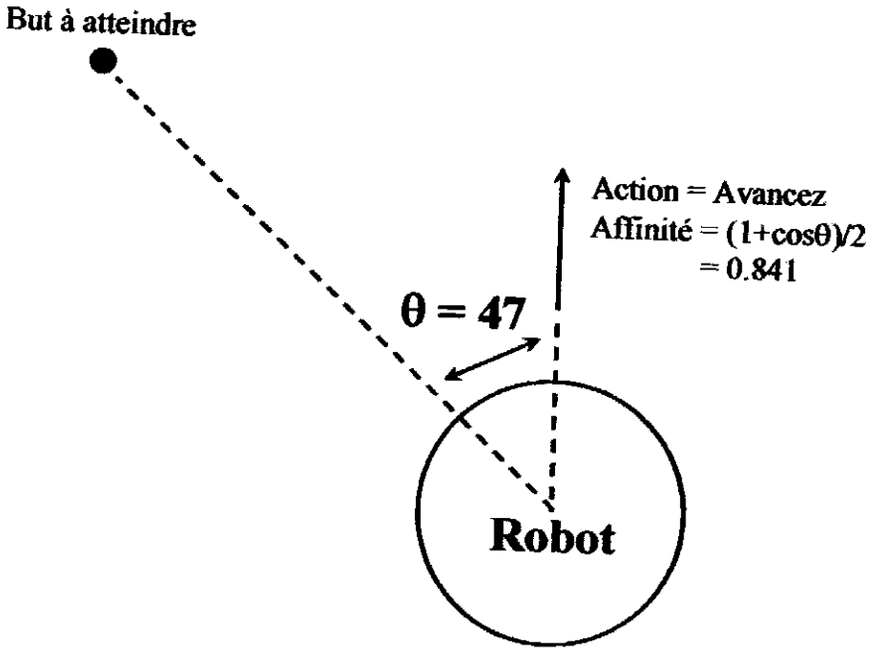


Figure III.12 : Calcul d'affinité entre un antigène et un anticorps

b) **Etape 2 :** Le calcul des concentrations de tous les anticorps. On a utilisé l'équation de Farmer et al [FAR 1986] pour calculer les concentrations des anticorps :

$$dA_i(t)/dt = [\sum_{j=1}^{N} m_{ji}\, a_j(t) - \sum_{k=1}^{N} m_{ik}\, a_k(t) + m_i - k_i\,]\, a_i(t) \qquad (1)$$

$$a_i(t+1) = 1/1+\exp(0.5-A_i(t)) \qquad (2)$$

Tel que :

- a_i : concentration de l'anticorps i
- N : nombre d'anticorps
- m_i : l'affinité entre l'antigène et l'anticorps i
- m_{ij} : l'affinité entre le i éme et le j éme anticorps
- k_i : coefficient de la mort normale de l'anticorps

L'équation 2 normalise la valeur de concentration des anticorps pour que la grande valeur possible soit égale à 1.

Dans la figure III.13, un antigène attaque le corps, donc le réseau immunitaire sera perturbé par cet antigène. Il exécutera une immuno-reaction qui sélectionne l'anticorps qui a la plus grande valeur d'affinité (dans cet exemple l'anticorps 1) afin d'exécuté son action qui sera la réponse à cet attaque.

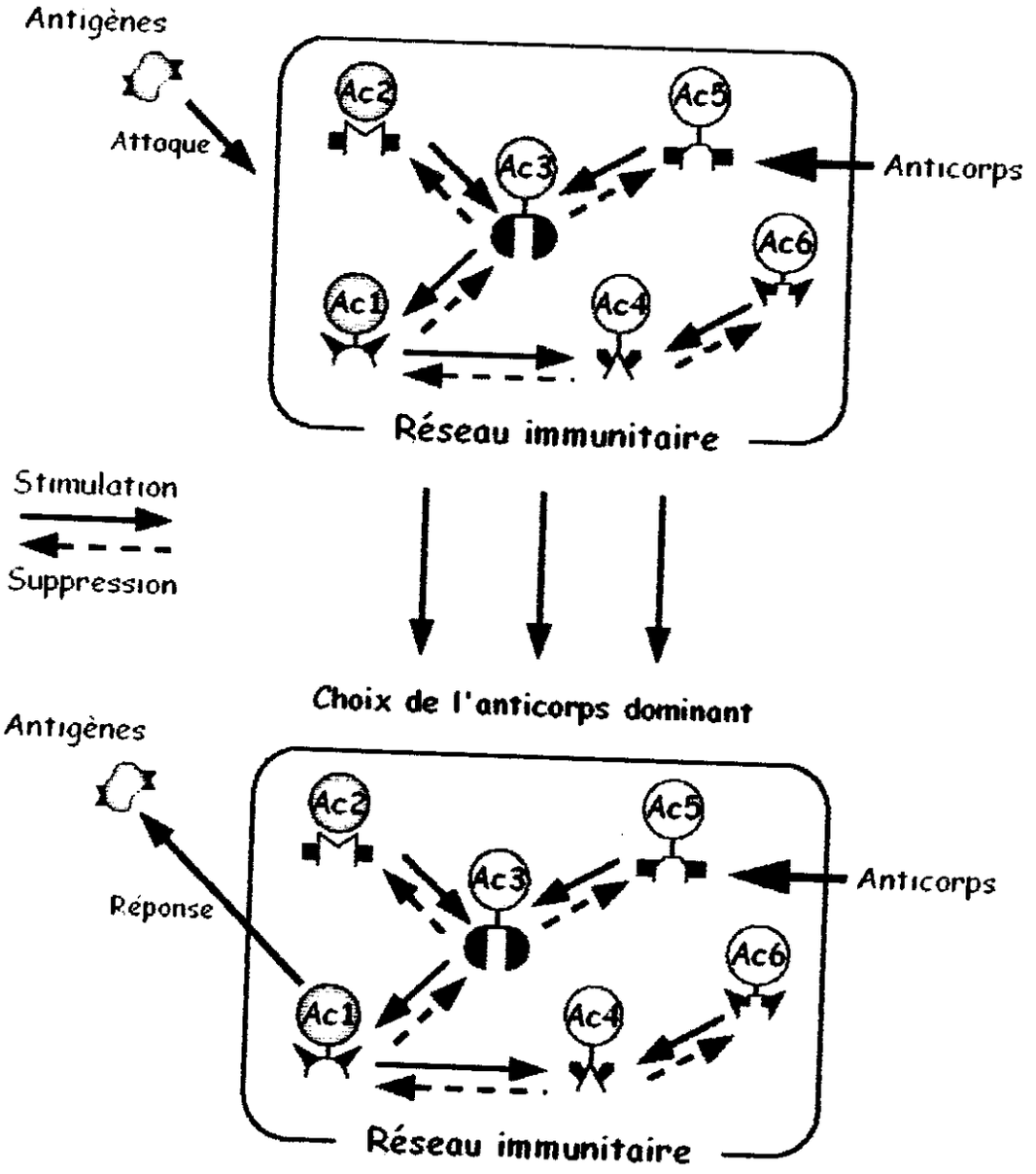


Figure III.13 : Mécanisme de sélection d'action

2.4.2.4. Exécution de l'action dominante

L'exécution de l'action est une étape très simple, elle est constituée par deux sous étapes qui sont :

- Le positionnement de la direction du robot suivant le numéro de l'action à exécuter, et cela en calculant la différence entre la direction courante du robot et l'angle de l'action à exécuter.

- L'exécution d'une commande qui fait avancer le robot d'une distance de 2cm, et cela en affectant une valeur positive qui est égale à 8 pour les moteurs des deux roues du robot (gauche et droite) et en exécutant cette action pendant une durée qui est égale à 0.05s.

2.4.2.5. Mécanisme d'adaptation

Pour assurer un comportement intelligent, adaptatif, flexible et autorégulateur il faut ajouter un mécanisme d'adaptation. Le mécanisme d'adaptation est habituellement classifié selon deux types : ajustement et innovation.

L'ajustement est une adaptation réalisée en changeant les paramètres du système. Pour les systèmes immunitaires artificiels, l'ajustement est réalisée en modifiant des formes moléculaires des anticorps (c.-à-d. l'affinité entre les anticorps).

L'innovation est une adaptation réalisée par les changements topologiques du système. Pour les systèmes immunitaires artificiels, l'innovation est réalisée par la fonction de métadynamique (c.-à-d. en supprimont et en créent des anticorps).

Dans notre système on a introduit le mécanisme d'ajustement. Il est réalisé par l'attribution des récompenses pour l'anticorps dominant afin de modifier l'affinité entre les anticorps.

2.4.2.5.1. L'ajustement

L'ajustement est réalisé par un opérateur de mutation, qui change la structure des anticorps, donc il modifie l'affinité entre les anticorps.

Dans notre système l'ajustement est assuré par des signaux de renfort qui attribuent des récompenses à l'anticorps dominant. Ces signaux sont de quatre types :

- Les signaux envoyés après chaque action
- Les signaux envoyés après chaque cinq actions (pour les cinq derniers anticorps dominants)

- Les signaux envoyés après chaque vingt actions (pour les vingt derniers anticorps dominants)
- Les signaux envoyés si le robot retourne à un point déjà visité (pour tous les anticorps dominants qui ont causé le retour du robot)

Les signaux de renfort sont positifs si l'état du robot a été amélioré et sont négatifs dans le cas contraire.

Quand une récompense positive est attribuée à un anticorps k l'affinité m_{ik} entre tous anticorps i et l'anticorps k est augmenter suivant la formule :

$$m_{ik} = m_{ik} + coeff^* m_{ik}/\delta$$

Tel que :

- δ : cœfficient de l'augmentation
- $coeff$: poids de la récompense

Pour normaliser les valeurs des affinités afin qu'elle soit comprises entre 1 et 0 on utilisera la formule :

$$m_{ik} = 1/1+\exp(0.5- m_{ik})$$

Quand une récompense négative est attribuée à un anticorps k l'affinité m_{ki} entre l'anticorps k et tous anticorps i est augmenté suivant la formule :

$$m_{ki} = m_{ki} + coeff^* m_{ki}/\delta$$

Ce mécanisme d'adaptation sera activé jusqu'à l'obtention d'une structure stable du réseau donc le robot continuera son apprentissage jusqu'à l'obtention d'une mémoire qui sera représenté par la structure du réseau et les différentes interactions entre les anticorps (les affinités).

Pour conclure cette partie, on résume le fonctionnement du contrôleur considéré par l'algorithme suivant :

Algorithme contrôleur réactif

Début

Créé un objet antigène ;

Créé un vecteur de 8 objets anticorps ;

Initialisé les actions des objets anticorps ;

Tant que (vrai)

Faire

Collectez les données ;

Pour i = 0 **à** 8

Faire

Calculez l'affinité de l'anticorps i ;

Calculez la concentration de l'anticorps i ;

Fait

Sélectionnez l'anticorps dominant ;

Exécuter l'action de l'anticorps dominant ;

Mémorisez l'action exécuter ;

Si (but) **Alors**

Stop ;

Si non

Attribuez les récompenses ;

Si (la structure du réseau est stable) **Alors**

Arrêtez l'attribution des récompenses ;

Fait

Fin

3. Tests

Dans cette partie, nous présenterons les résultats de nos tests. On rappelle, une autre fois que nous avons utilisé le simulateur WSU Khepera Robot Simulator v7.2. Nous avons opté pour ce choix car le simulateur est développé en Java et son code source est publié.

Les tests qu'on a fait peuvent être classé, en trois groupes : les tests sur des environnements simples, les tests sur des environnements complexes sans pièges et les tests sur des environnements qui présentent des situations pièges. Notre contrôleur a donné de bons résultats dans les deux premiers types d'environnements, mais il a montré ses limites dans le troisième type d'environnements. Dans ce qui suit, on va commenter les résultats obtenus.

3.1. Les tests dans les environnements simples

La figure III.14 représente l'environnement du test. Le problème était de faire une navigation entre deux points jusqu'à l'amélioration de la trajectoire du robot.

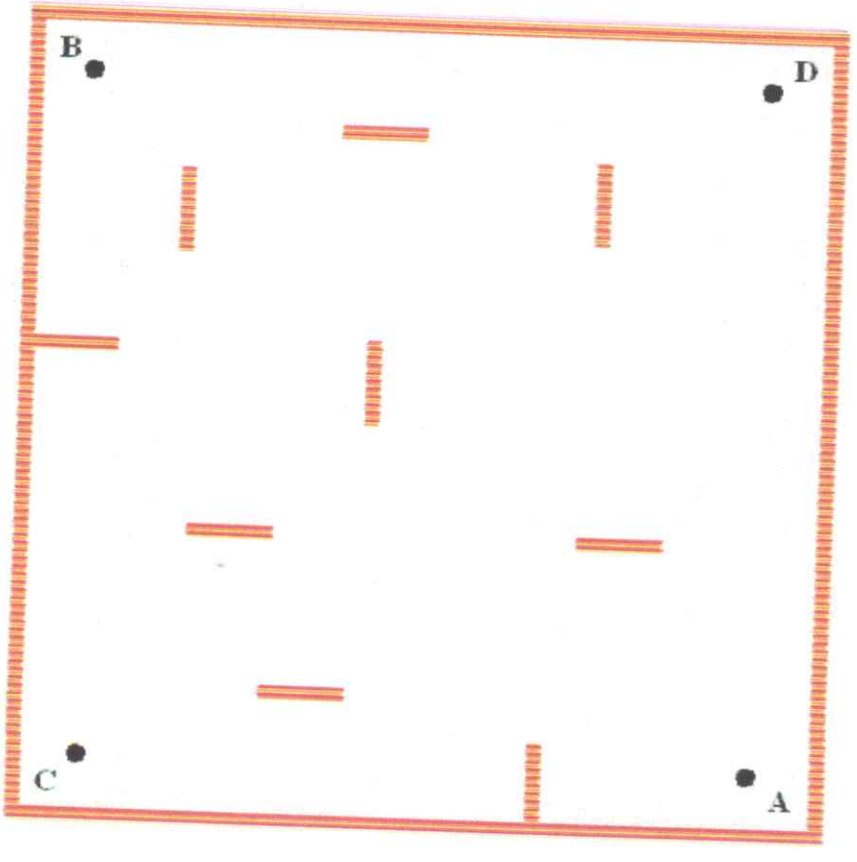
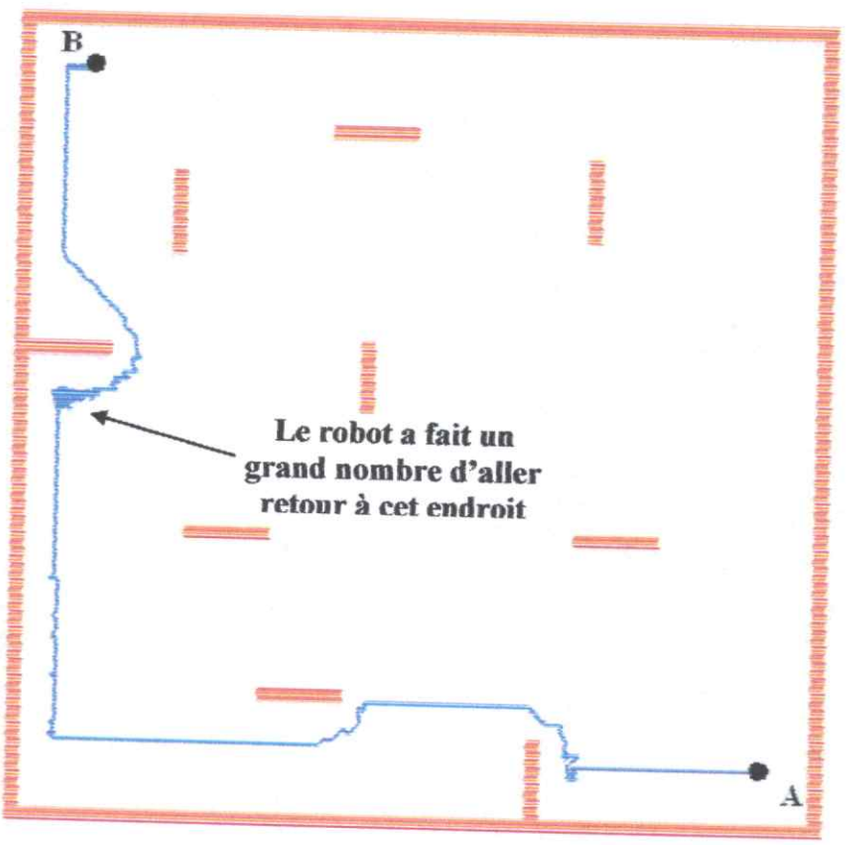


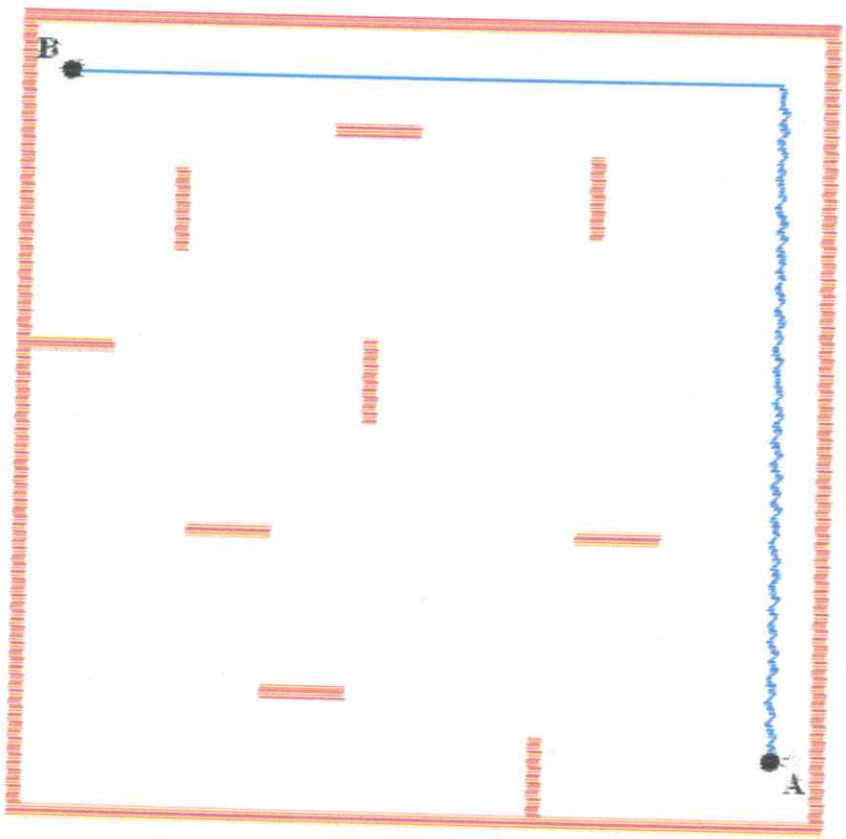
Figure III.14 : Premier environnement de test.

Comme représenté dans la figure, on a quatre buts (*A*, *B*, *C* et *D*), la navigation se fera selon deux cas : le premier cas entre *A* et *B* et le deuxième cas entre *C* et *D*.

Cas 1 : Dans ce cas la navigation du robot sera entre *A* et *B*. La trajectoire du robot à la première itération est représentée par la figure III.15.



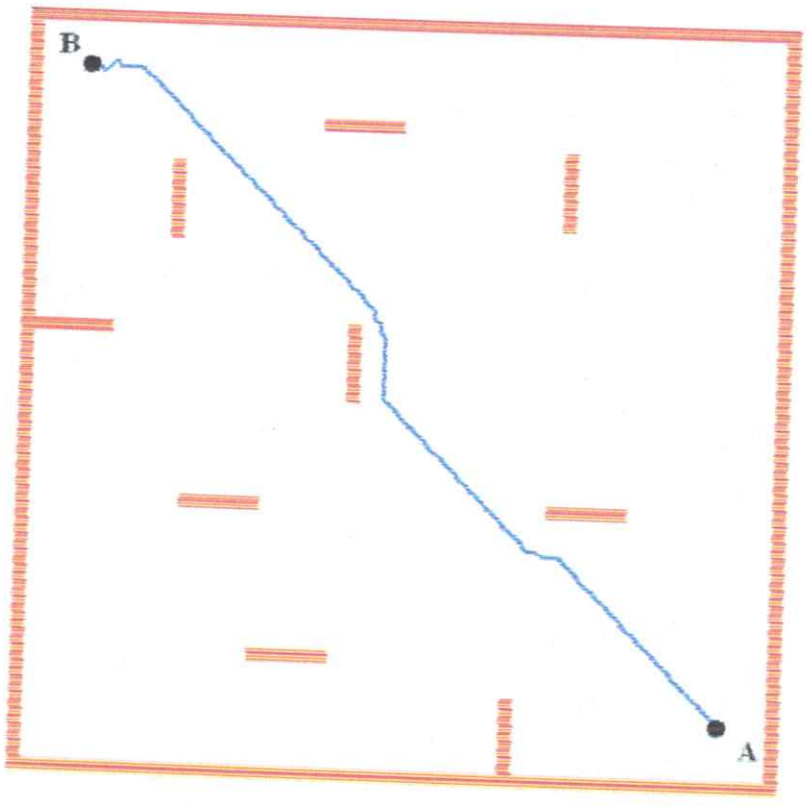
(a) La navigation se fait de *A* vers *B*.



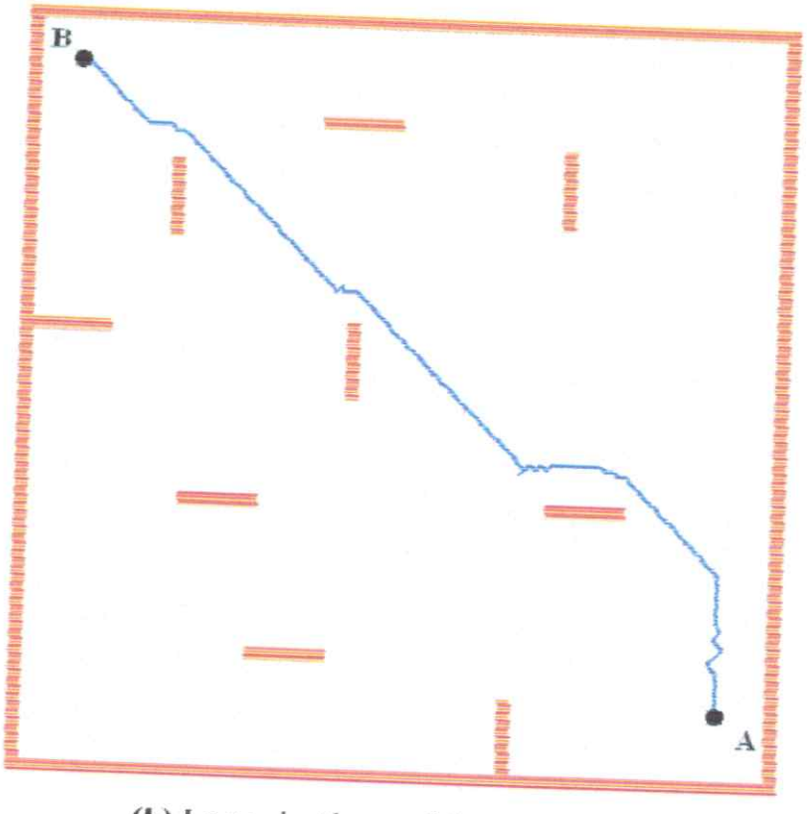
(b) La navigation se fait de *B* vers *A*.

Figure III.15 : Trajectoire du robot dans la première itération.

On remarque que dans la première itération la trajectoire du robot était mauvaise et la durée était assez importante, le robot n'avait pas un comportement intelligent. Ce comportement a était amélioré par la suite, la figure III.16 représente la trajectoire du robot après la phase d'apprentissage.



(a) La navigation se fait de *A* vers *B*.



(b) La navigation se fait de *B* vers *A*.

Figure III.16 : Trajectoire du robot après la phase d'apprentissage.

On remarque que la trajectoire du robot a été largement améliorée, on peut dire que le robot s'est adapté à cet environnement. Pour confirmer ces résultats la figure III.17 présente une courbe de la performance du robot durant ce test.

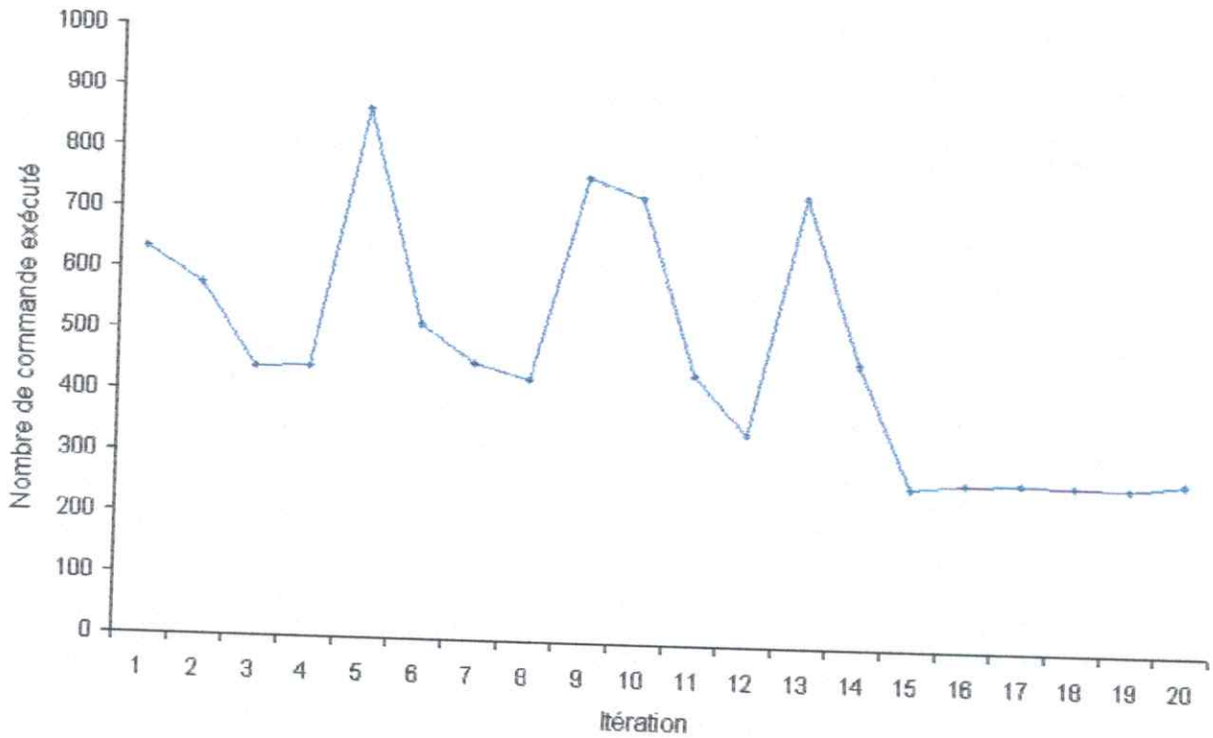


Figure III.17 : Performance du robot durant le test.

Les deux tableaux III.1 et III.3 représentent les valeurs des affinités entre les anticorps après la phase d'apprentissage. Dans ces tableaux, les anticorps des lignes stimulent les anticorps des colonnes, et les anticorps des colonnes suppriment les anticorps des lignes, par exemple la valeur situé à la deuxième ligne de la première colonne représente l'affinité de la stimulation de l'anticorps 0 par l'anticorps 1(ou suppression de l'anticorps 1 par l'anticorps 0).

Tableau III.1 : Les affinités entre les anticorps après la phase d'apprentissage (de *A* vers *B*)

Sm / Sp	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7
AC0	0.0	0.0547	0.0537	0.0537	0.0537	0.0555	0.0601	0.9585
AC1	0.0623	0.0	0.0524	0.0524	0.0524	0.0541	0.0587	0.9350
AC2	0.0623	0.0533	0.0	0.0524	0.0524	0.0541	0.0587	0.9350
AC3	0.0623	0.0533	0.0524	0.0	0.0524	0.0541	0.0587	0.9350
AC4	0.0623	0.0533	0.0524	0.0524	0.0	0.0541	0.0587	0.9350
AC5	0.0623	0.0533	0.0524	0.0524	0.0524	0.0	0.0587	0.9350
AC6	0.0623	0.0533	0.0524	0.0524	0.0524	0.0541	0.0	0.9350
AC7	0.0779	0.0667	0.0655	0.0655	0.0656	0.0677	0.0734	0.0

(Sm : stimulation, Sp : suppression)

Tableau III.2 : Correspondance entre les anticorps et les actions

Anticorps	Action
AC0 : anticorps 0	↑
AC1 : anticorps 1	↗
AC2 : anticorps 2	→
AC3 : anticorps 3	↘
AC4 : anticorps 4	↓
AC5 : anticorps 5	↙
AC6 : anticorps 6	←
AC7 : anticorps 7	↖

Le tableau III.1 correspond à la trajectoire de *A* vers *B* on remarque qu'après l'apprentissage le réseau est construit et sa structure devient stable. On a l'anticorps 7 qui est stimulé par tous autres anticorps donc l'action correspondante aura la plus grande priorité. La structure du réseau se déduit du tableau des affinités, en comparons les valeurs des affinités entre les anticorps. On prend par exemple deux anticorps *x* et *y* :

- si la valeur de l'affinité de la stimulation de *x* par *y* est nettement supérieur à la valeur de l'affinité de la suppression de *x* par *y* alors *y* stimule *x* et *x* supprime *y*.
- si la valeur de l'affinité de la stimulation de *x* par *y* est proche de la valeur de l'affinité de la suppression de *x* par *y* alors il n'y a pas d'interaction entre *x* et *y*.

La structure dans notre cas peut être illustrée par la figure suivante :

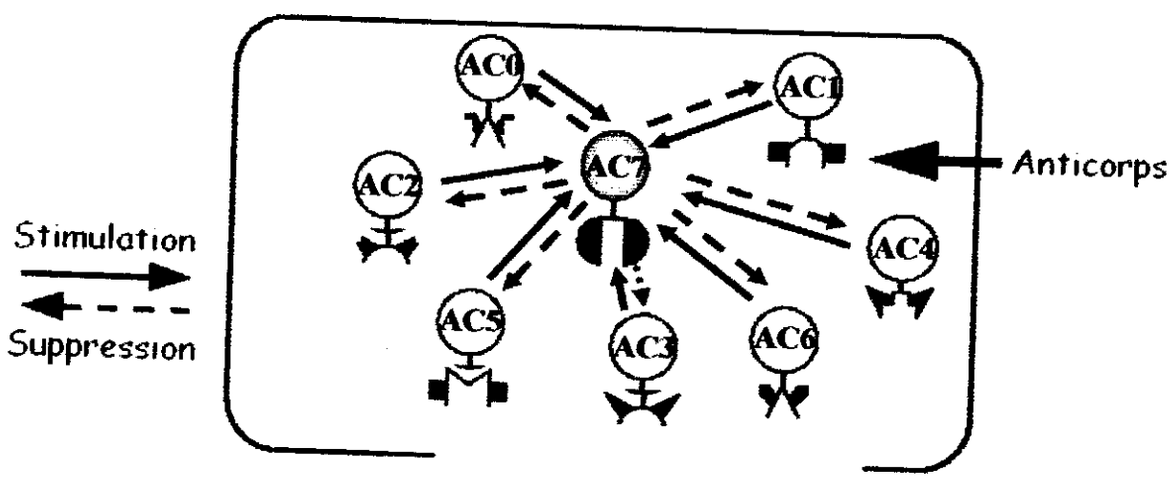


Figure III.18 : Structure de réseau après la phase d'apprentissage (de *A* vers *B*).

Tableau III.3 : Les affinités entre les anticorps après la phase d'apprentissage (de *B* vers *A*)

Sm / Sp	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7
AC0	0.0	0.0551	0.0669	0.7549	0.0661	0.0538	0.0524	0.0524
AC1	0.0579	0.0	0.0739	0.8338	0.0730	0.0595	0.0579	0.0579
AC2	0.0524	0.0551	0.0	0.7549	0.0661	0.0538	0.0524	0.0524
AC3	0.0639	0.0673	0.0816	0.0	0.0807	0.0657	0.0640	0.0639
AC4	0.0524	0.0551	0.0669	0.7549	0.0	0.0538	0.0524	0.0524
AC5	0.0579	0.0609	0.0739	0.8338	0.0730	0.0	0.0579	0.0579
AC6	0.0524	0.0551	0.0669	0.7551	0.0661	0.0538	0.0	0.0524
AC7	0.0524	0.0551	0.0669	0.7549	0.0661	0.0538	0.0524	0.0

(Sm : stimulation, Sp : suppression)

Le tableau III.3 correspond à la trajectoire de *B* vers *A* on remarque qu'après l'apprentissage le réseau est construit et sa structure devient stable. On a l'anticorps 3 qui est stimulé par tous autres anticorps donc l'action correspondante aura la plus grande priorité. La structure du réseau se déduit du tableau des affinités, en comparons les valeurs des affinités entre les anticorps. On prend par exemple deux anticorps *x* et *y* :

- si la valeur de l'affinité de la stimulation de *x* par *y* est nettement supérieur à la valeur de l'affinité de la suppression de *x* par *y* alors *y* stimule *x* et *x* supprime *y*.
- si la valeur de l'affinité de la stimulation de *x* par *y* est proche de la valeur de l'affinité de la suppression de *x* par *y* alors il n'y a pas d'interaction entre *x* et *y*.

La structure dans notre cas peut être illustrée par la figure suivante :

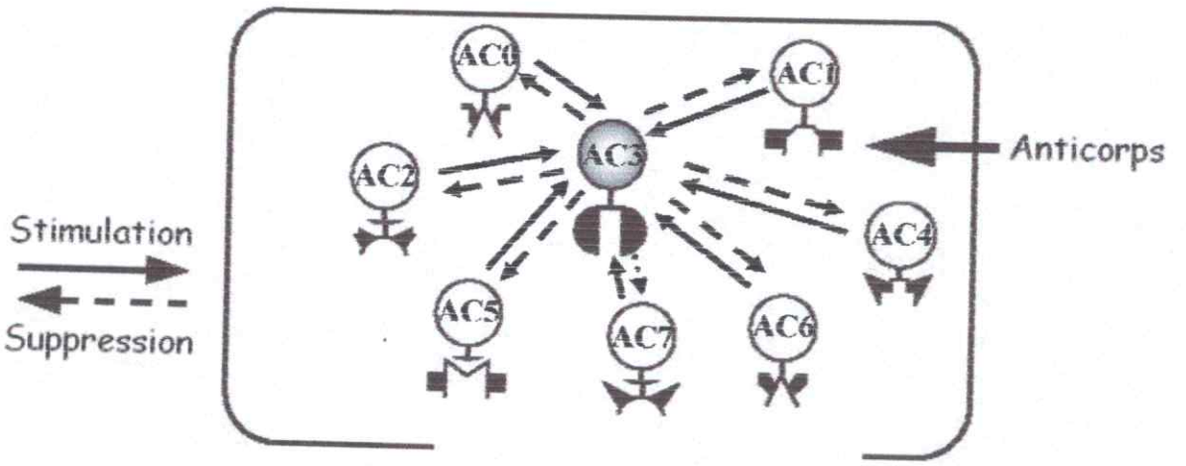
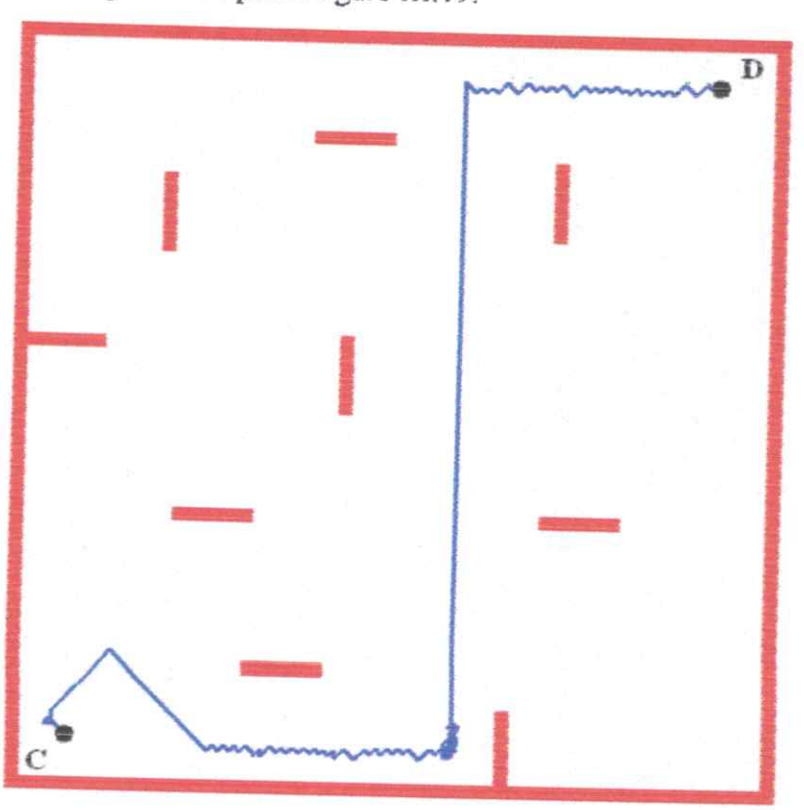


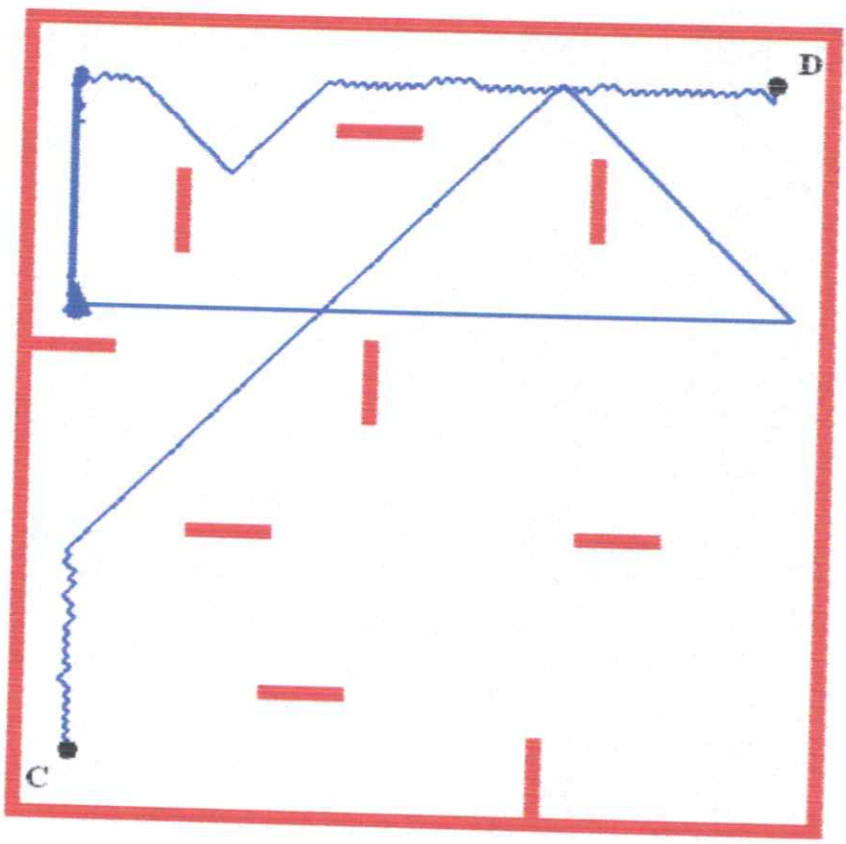
Figure III.19 : Structure de réseau après la phase d'apprentissage (de *B* vers *A*).

Donc le contrôleur a réglé les interactions entre les composants par rapport à la structure de l'environnement et a crée une mémoire qui se symbolise par les affinités entre les anticorps.

Cas 2 : Dans ce cas la navigation du robot sera entre *C* et *D*. La trajectoire du robot à la première itération est représentée par la figure III.19.



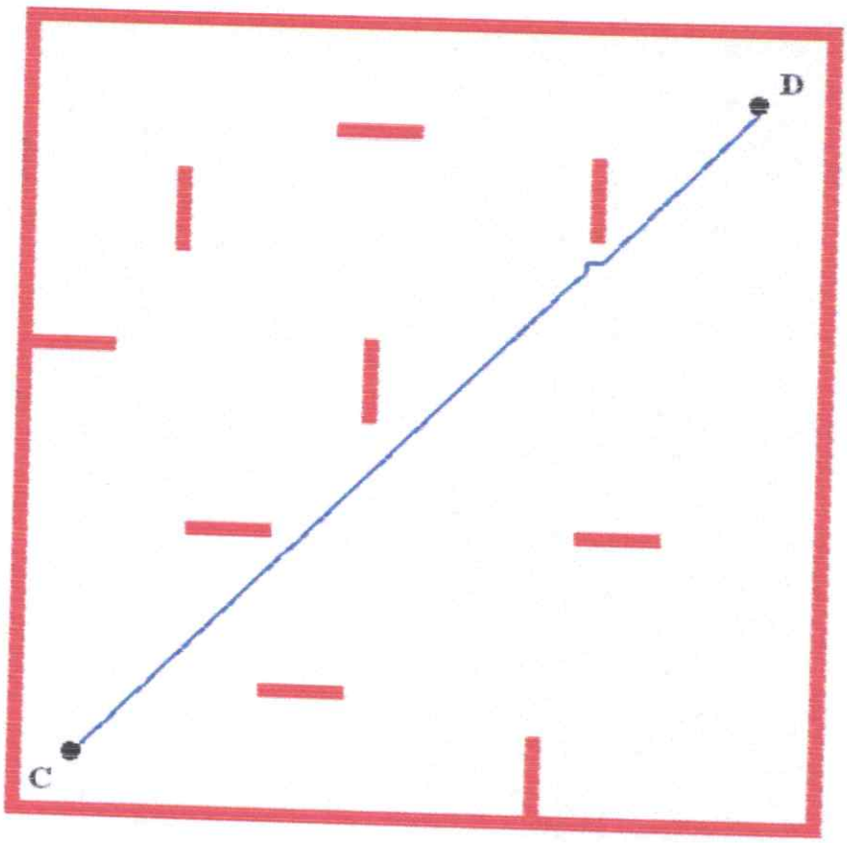
(a) La navigation se fait de *C* vers *D*.



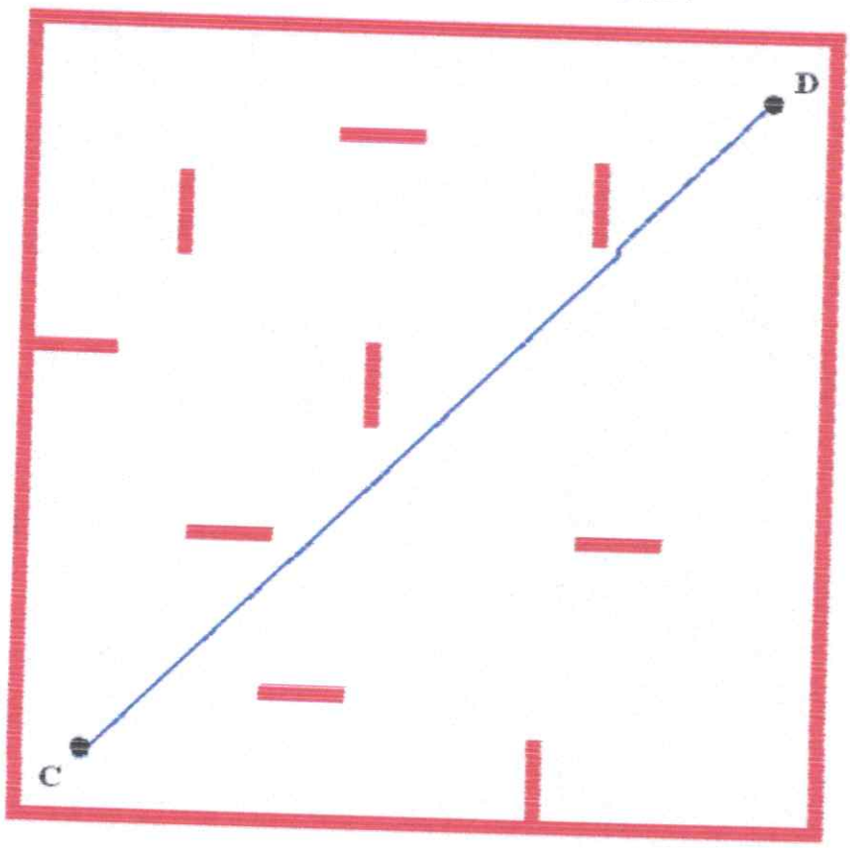
(b) La navigation se fait de *D* vers *C*.

Figure III.20 : Trajectoire du robot dans la première itération.

On remarque que dans la première itération la trajectoire du robot était mauvaise et la durée était assez importante, le robot n'avait pas un comportement intelligent. Ce comportement a était amélioré par la suite, la figure III.21 représente la trajectoire du robot après la phase d'apprentissage.



(a) La navigation se fait de *C* vers *D*.



(b) La navigation se fait de *D* vers *C*.

Figure III.21 : Trajectoire du robot après la phase d'apprentissage.

On remarque que la trajectoire du robot a été largement améliorée, on peut dire que le robot s'est adapté à cet environnement. Pour confirmer ces résultats la figure III.22 présente une courbe de la performance du robot durant ce test.

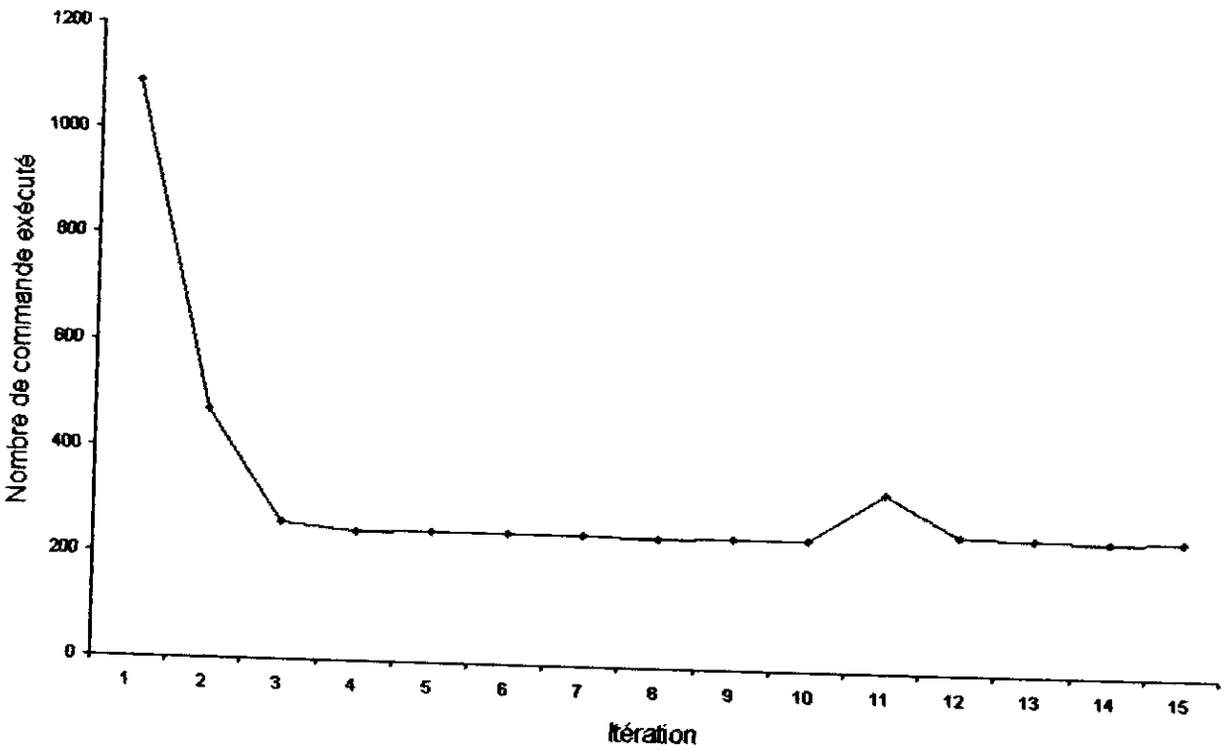


Figure III.22 : Performance du robot durant le test.

Les deux tableaux III.4 et III.5 représentent les valeurs des affinités entre les anticorps après la phase d'apprentissage. Dans ces tableaux, les anticorps des lignes stimulent les anticorps des colonnes, et les anticorps des colonnes suppriment les anticorps des lignes, par exemple la valeur situé à la deuxième ligne de la première colonne représente l'affinité de la stimulation de l'anticorps 0 par l'anticorps 1(ou suppression de l'anticorps 1 par l'anticorps 0).

Tableau III.4 : Les affinités entre les anticorps après la phase d'apprentissage (de *C* vers *D*)

Sm / Sp	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7
AC0	0.0	0.9972	0.0723	0.0696	0.0696	0.0696	0.0696	0.0700
AC1	0.4466	0.0	0.4547	0.4379	0.4379	0.4379	0.4379	0.4405
AC2	0.0684	0.9608	0.0	0.0671	0.0671	0.0671	0.0671	0.0675
AC3	0.0684	0.9608	0.0696	0.0	0.0671	0.0671	0.0671	0.0675
AC4	0.0684	0.9608	0.0696	0.0671	0.0	0.0671	0.0671	0.0675
AC5	0.0684	0.9608	0.0696	0.0671	0.0671	0.0	0.0671	0.0675
AC6	0.0684	0.9608	0.0696	0.0671	0.0671	0.0671	0.0	0.0675
AC7	0.0684	0.9608	0.0696	0.0671	0.0671	0.0671	0.0671	0.0

(Sm : stimulation, Sp : suppression)

Le tableau III.4 correspond à la trajectoire de *C* vers *D* on remarque qu'après l'apprentissage le réseau est construit et sa structure devient stable. On a l'anticorps 1 qui est stimulé par tous autres anticorps donc l'action correspondante aura la plus grande priorité. La structure du réseau se déduit du tableau des affinités, en comparons les valeurs des affinités entre les anticorps. On prend par exemple deux anticorps *x* et *y* :

- si la valeur de l'affinité de la stimulation de *x* par *y* est nettement supérieur à la valeur de l'affinité de la suppression de *x* par *y* alors *y* stimule *x* et *x* supprime *y*.
- si la valeur de l'affinité de la stimulation de *x* par *y* est proche de la valeur de l'affinité de la suppression de *x* par *y* alors il n'y a pas d'interaction entre *x* et *y*.

La structure dans notre cas peut être illustrée par la figure suivante :

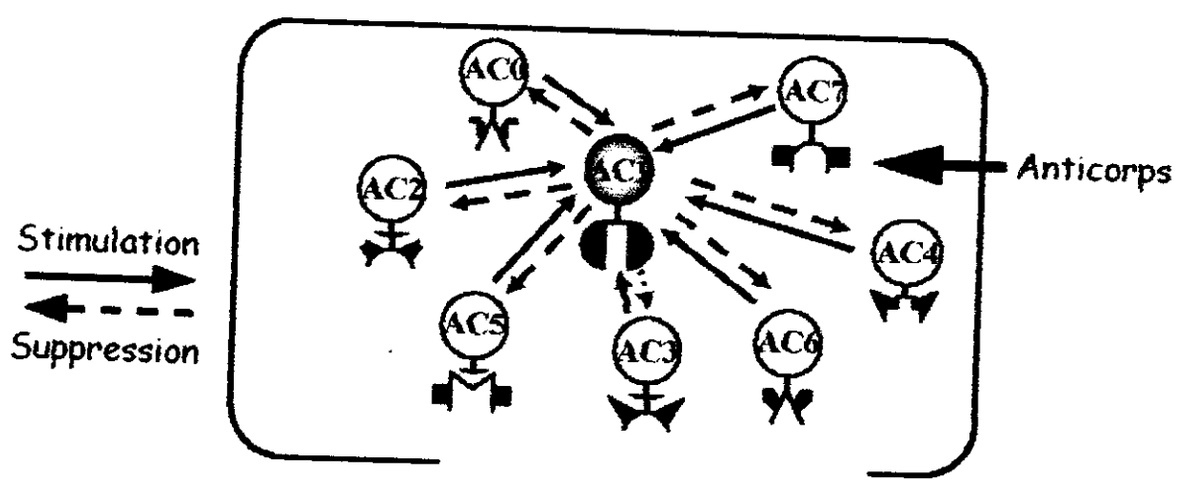


Figure III.23 : Structure de réseau après la phase d'apprentissage (de *C* vers *D*).

Tableau III.5 : Les affinités entre les anticorps après la phase d'apprentissage (de *D* vers *C*)

Sm / Sp	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7
AC0	0.0	0.0671	0.0671	0.0671	0.0694	0.9765	0.0686	0.0672
AC1	0.0671	0.0	0.0671	0.0671	0.0694	0.9765	0.0686	0.0672
AC2	0.0671	0.0671	0.0	0.0671	0.0694	0.9765	0.0686	0.0672
AC3	0.0671	0.0671	0.0671	0.0	0.0694	0.9765	0.0686	0.0672
AC4	0.0671	0.0671	0.0671	0.0671	0.0	0.9765	0.0686	0.0672
AC5	0.2473	0.2473	0.2473	0.2473	0.2561	0.0	0.2531	0.2476
AC6	0.0671	0.0671	0.0671	0.0671	0.0694	0.9765	0.0	0.0672
AC7	0.0671	0.0671	0.0671	0.0671	0.0694	0.9765	0.0686	0.0

(Sm : stimulation, Sp : suppression)

Le tableau III.5 correspond à la trajectoire de *D* vers *C* on remarque qu'après l'apprentissage le réseau est construit et sa structure devient stable. On a l'anticorps 5 qui est stimulé par tous autres anticorps donc l'action correspondante aura la plus grande priorité. La structure du réseau se déduit du tableau des affinités, en comparons les valeurs des affinités entre les anticorps. On prend par exemple deux anticorps *x* et *y* :

- si la valeur de l'affinité de la stimulation de *x* par *y* est nettement supérieur à la valeur de l'affinité de la suppression de *x* par *y* alors *y* stimule *x* et *x* supprime *y*.
- si la valeur de l'affinité de la stimulation de *x* par *y* est proche de la valeur de l'affinité de la suppression de *x* par *y* alors il n'y a pas d'interaction entre *x* et *y*.

La structure dans notre cas peut être illustrée par la figure suivante :

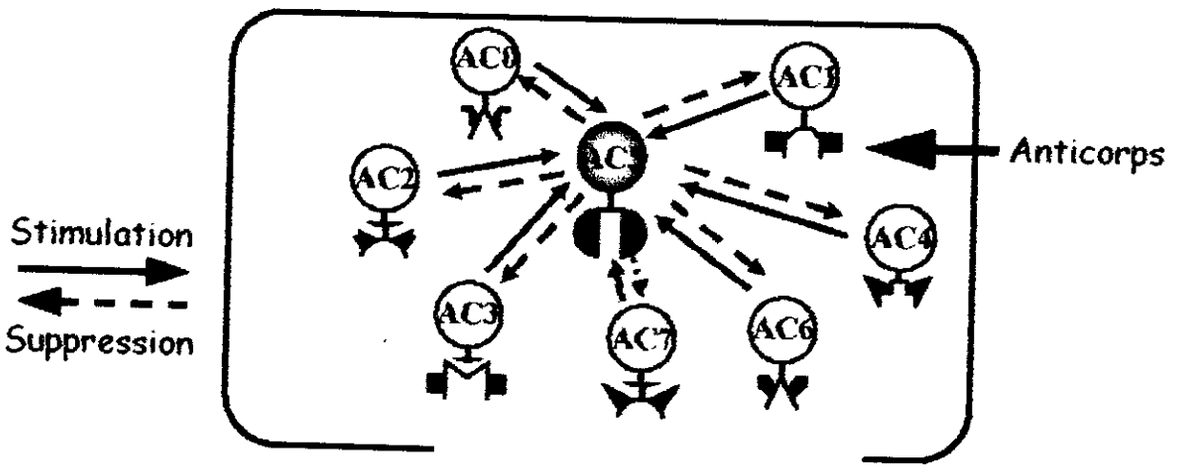


Figure III.24 : Structure de réseau après la phase d'apprentissage (de *D* vers *C*).

Donc le contrôleur a réglé les interactions entre les composants par rapport à la structure de l'environnement et a crée une mémoire qui se symbolise par les affinités entre les anticorps.

3.2. Les tests dans les environnements complexes sans pièges

La figure III.25 représente l'environnement du test. Le problème était de faire une navigation entre deux points jusqu'à l'amélioration de la trajectoire du robot.

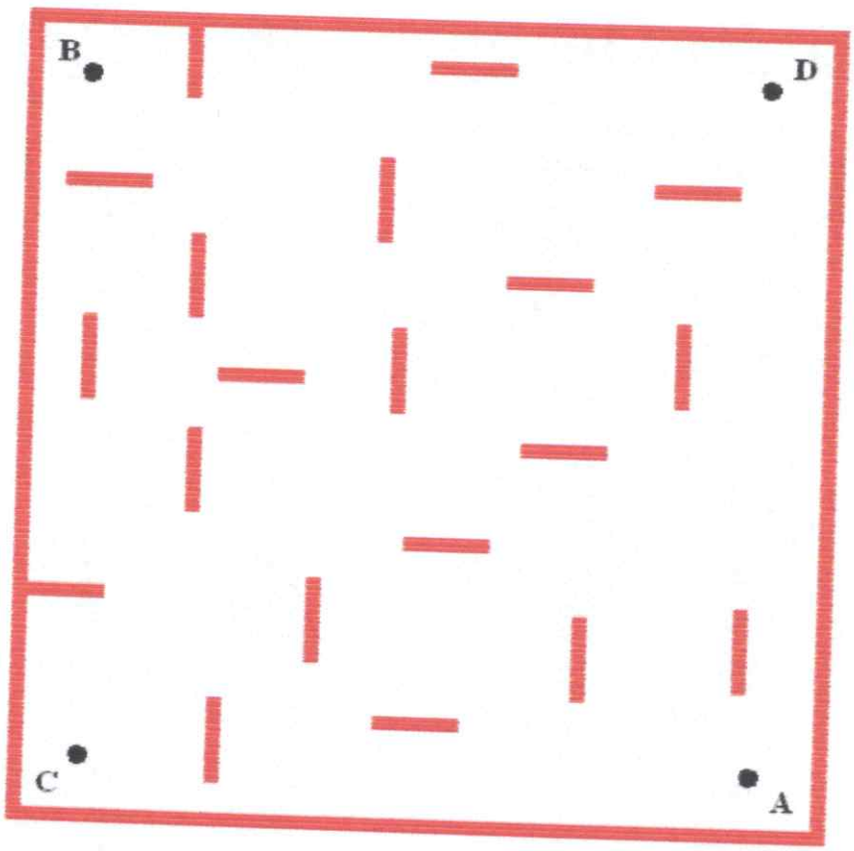
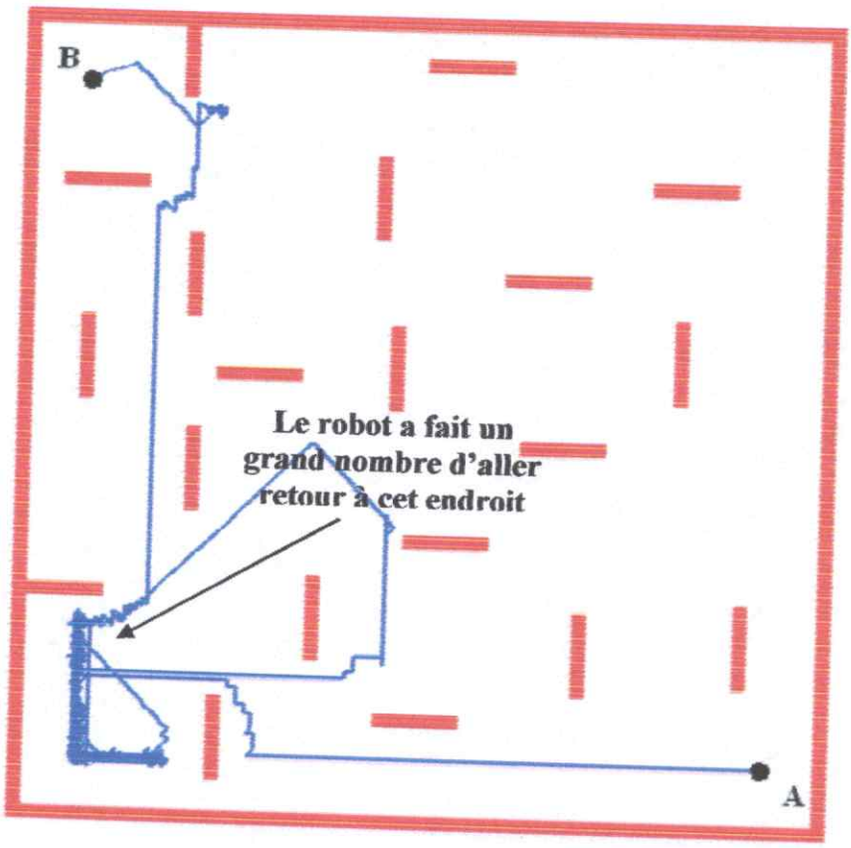


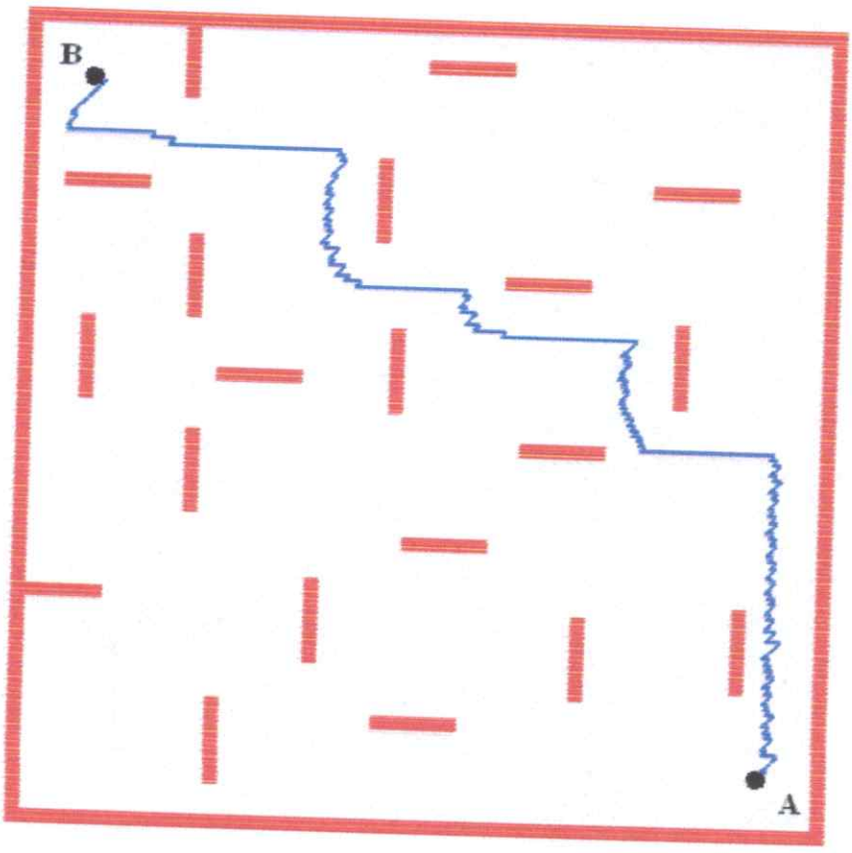
Figure III.25 : deuxième environnement de test.

Comme représenté dans la figure, on a quatre buts (*A*, *B*, *C* et *D*), la navigation se fera selon deux cas : le premier cas entre *A* et *B* et le deuxième cas entre *C* et *D*.

Cas 1 : Dans ce cas la navigation du robot sera entre *A* et *B*. La trajectoire du robot à la première itération est représentée par la figure III.26.



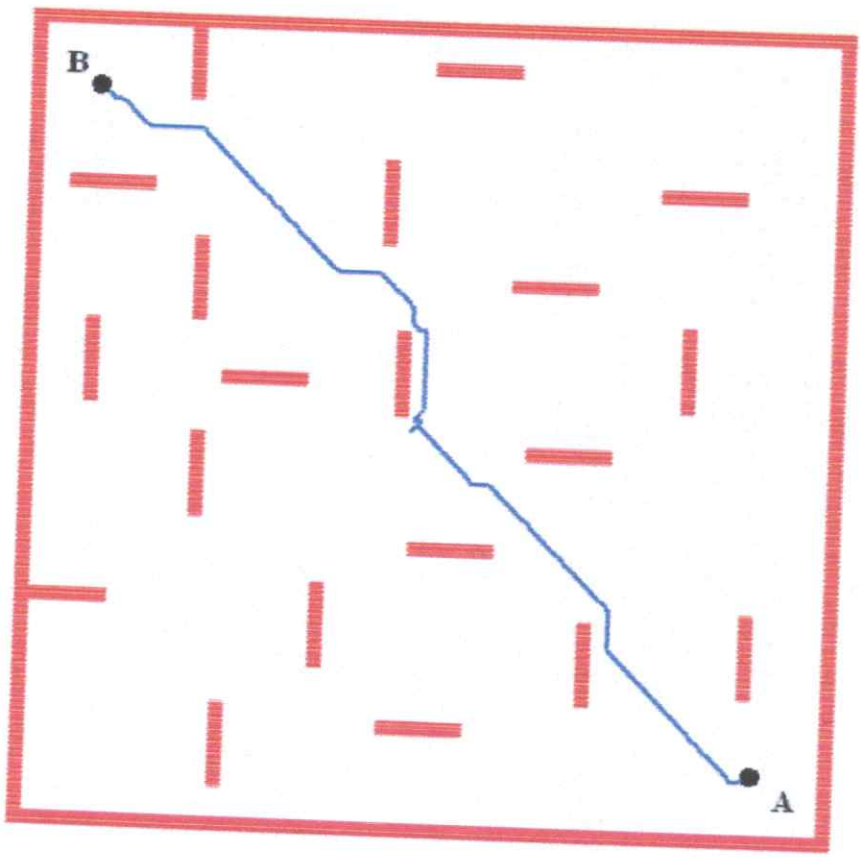
(a) La navigation se fait de *A* vers *B*.



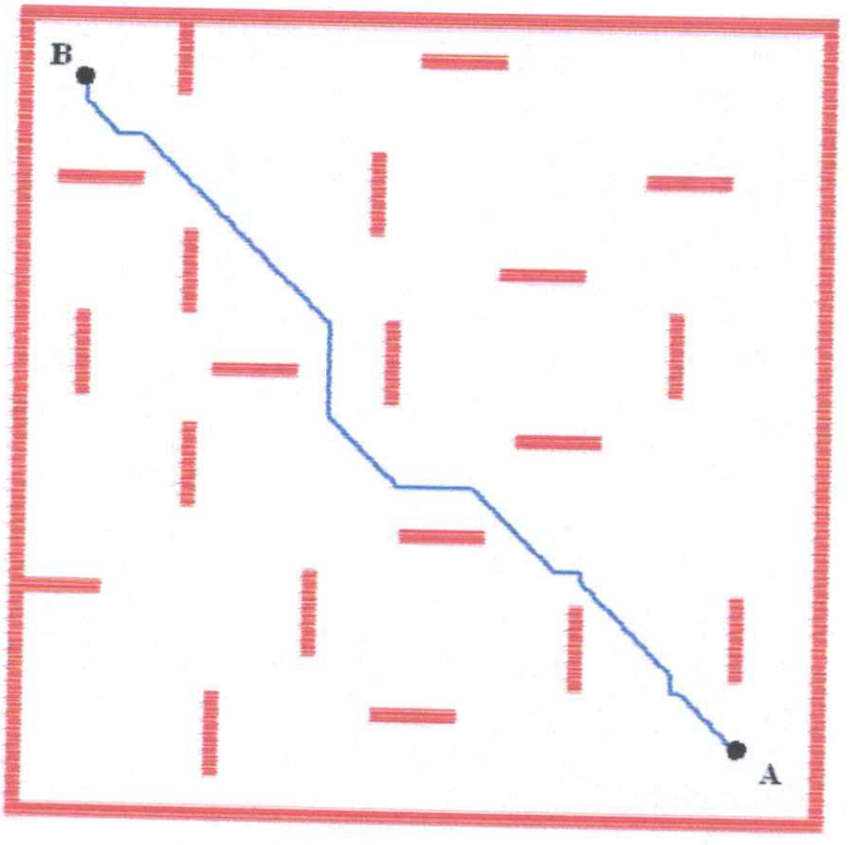
(b) La navigation se fait de *B* vers *A*.

Figure III.26 : Trajectoire du robot dans la première itération.

On remarque que dans la première itération la trajectoire du robot était mauvaise et la durée était assez importante, le robot n'avait pas un comportement intelligent. Ce comportement a était amélioré par la suite, la figure III.27 représente la trajectoire du robot après la phase d'apprentissage.



(a) La navigation se fait de *A* vers *B*.



(b) La navigation se fait de *B* vers *A*.

Figure III.27 : Trajectoire du robot après la phase d'apprentissage.

On remarque que la trajectoire du robot a été largement améliorée, on peut dire que le robot s'est adapté à cet environnement. Pour confirmer ces résultats la figure III.28 présente une courbe de la performance du robot durant ce test.

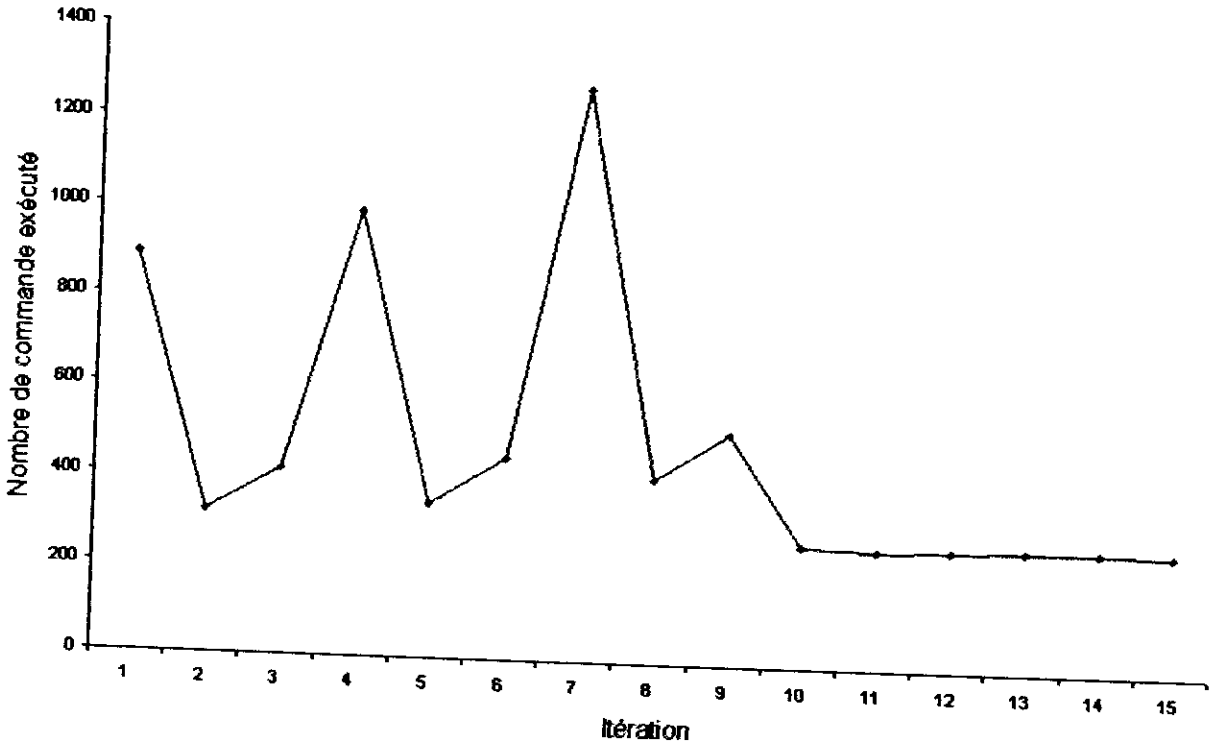


Figure III.28 : Performance du robot durant le test.

Les deux tableaux III.6 et III.7 représentent les valeurs des affinités entre les anticorps après la phase d'apprentissage. Dans ces tableaux, les anticorps des lignes stimulent les anticorps des colonnes, et les anticorps des colonnes suppriment les anticorps des lignes, par exemple la valeur situé à la deuxième ligne de la première colonne représente l'affinité de la stimulation de l'anticorps 0 par l'anticorps 1(ou suppression de l'anticorps 1 par l'anticorps 0).

Tableau III.6 : Les affinités entre les anticorps après la phase d'apprentissage (de *A* vers *B*)

Sm / Sp	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7
AC0	0.0	0.0587	0.0563	0.0563	0.0567	0.0609	0.0606	0.8221
AC1	0.0789	0.0	0.0652	0.0652	0.0657	0.0705	0.0702	0.9517
AC2	0.0649	0.0559	0.0	0.0536	0.0540	0.0580	0.0577	0.7830
AC3	0.0649	0.0559	0.0536	0.0	0.0540	0.0580	0.0577	0.7830
AC4	0.0684	0.0588	0.0565	0.0565	0.0	0.0610	0.0608	0.8242
AC5	0.0789	0.0679	0.0652	0.0652	0.0657	0.0	0.0702	0.9517
AC6	0.0682	0.0587	0.0563	0.0563	0.0567	0.0609	0.0	0.8221
AC7	0.1225	0.1054	0.1012	0.1012	0.1019	0.1093	0.1089	0.0

(Sm : stimulation, Sp : suppression)

Le tableau III.6 correspond à la trajectoire de *A* vers *B* on remarque qu'après l'apprentissage le réseau est construit et sa structure devient stable. On a l'anticorps 7 qui est stimulé par tous autres anticorps donc l'action correspondante aura la plus grande priorité. La structure du réseau se déduit du tableau des affinités, en comparons les valeurs des affinités entre les anticorps. On prend par exemple deux anticorps *x* et *y* :

- si la valeur de l'affinité de la stimulation de *x* par *y* est nettement supérieur à la valeur de l'affinité de la suppression de *x* par *y* alors *y* stimule *x* et *x* supprime *y*.
- si la valeur de l'affinité de la stimulation de *x* par *y* est proche de la valeur de l'affinité de la suppression de *x* par *y* alors il n'y a pas d'interaction entre *x* et *y*.

La structure dans notre cas peut être illustrée par la figure suivante :

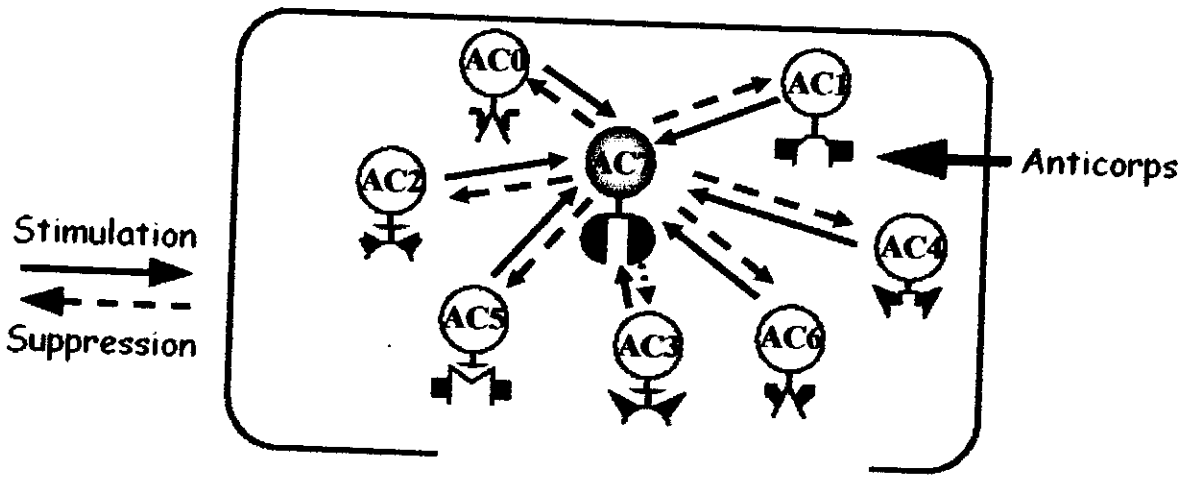


Figure III.29 : Structure de réseau après la phase d'apprentissage (de *A* vers *B*).

Tableau III.7 : Les affinités entre les anticorps après la phase d'apprentissage (de *B* vers *A*)

Sm / Sp	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7
AC0	0.0	0.0547	0.0652	0.8429	0.0650	0.0555	0.0536	0.0536
AC1	0.0536	0.0	0.0652	0.8429	0.0650	0.0555	0.0536	0.0536
AC2	0.0536	0.0547	0.0	0.8429	0.0650	0.0555	0.0536	0.0536
AC3	0.1116	0.1137	0.1357	0.0	0.1351	0.1155	0.1116	0.1116
AC4	0.0536	0.0547	0.0652	0.8429	0.0	0.0555	0.0536	0.0536
AC5	0.0536	0.0547	0.0652	0.8429	0.0650	0.0	0.0536	0.0536
AC6	0.0536	0.0547	0.0652	0.8429	0.0650	0.0555	0.0	0.0536
AC7	0.0536	0.0547	0.0652	0.8429	0.0650	0.0555	0.0536	0.0

(Sm : stimulation, Sp : suppression)

Le tableau III.7 correspond à la trajectoire de *B* vers *A* on remarque qu'après l'apprentissage le réseau est construit et sa structure devient stable. On a l'anticorps 3 qui est stimulé par tous autres anticorps donc l'action correspondante aura la plus grande priorité. La structure du réseau se déduit du tableau des affinités, en comparons les valeurs des affinités entre les anticorps. On prend par exemple deux anticorps *x* et *y* :

- si la valeur de l'affinité de la stimulation de *x* par *y* est nettement supérieur à la valeur de l'affinité de la suppression de *x* par *y* alors *y* stimule *x* et *x* supprime *y*.
- si la valeur de l'affinité de la stimulation de *x* par *y* est proche de la valeur de l'affinité de la suppression de *x* par *y* alors il n'y a pas d'interaction entre *x* et *y*.

La structure dans notre cas peut être illustrée par la figure suivante :

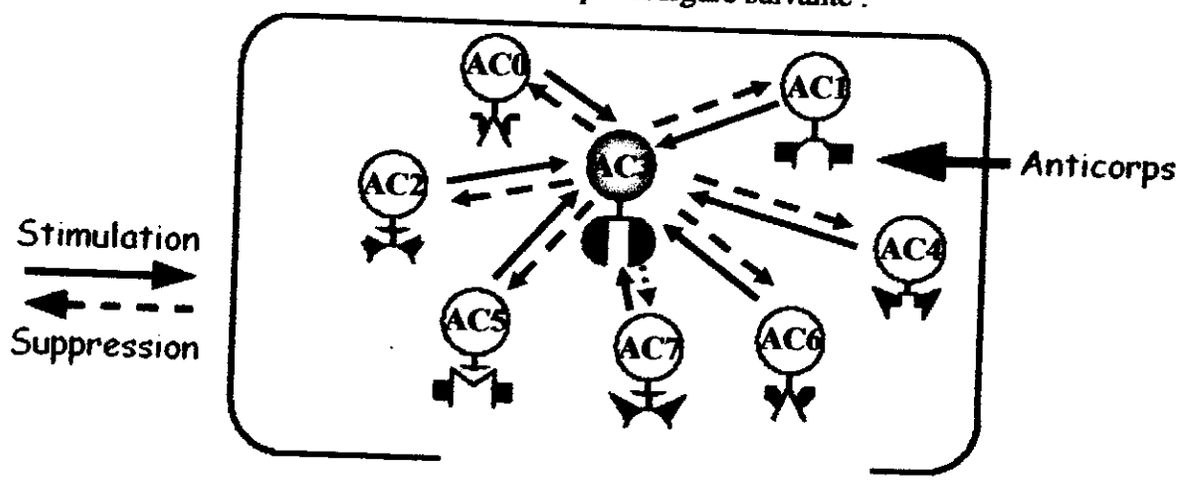
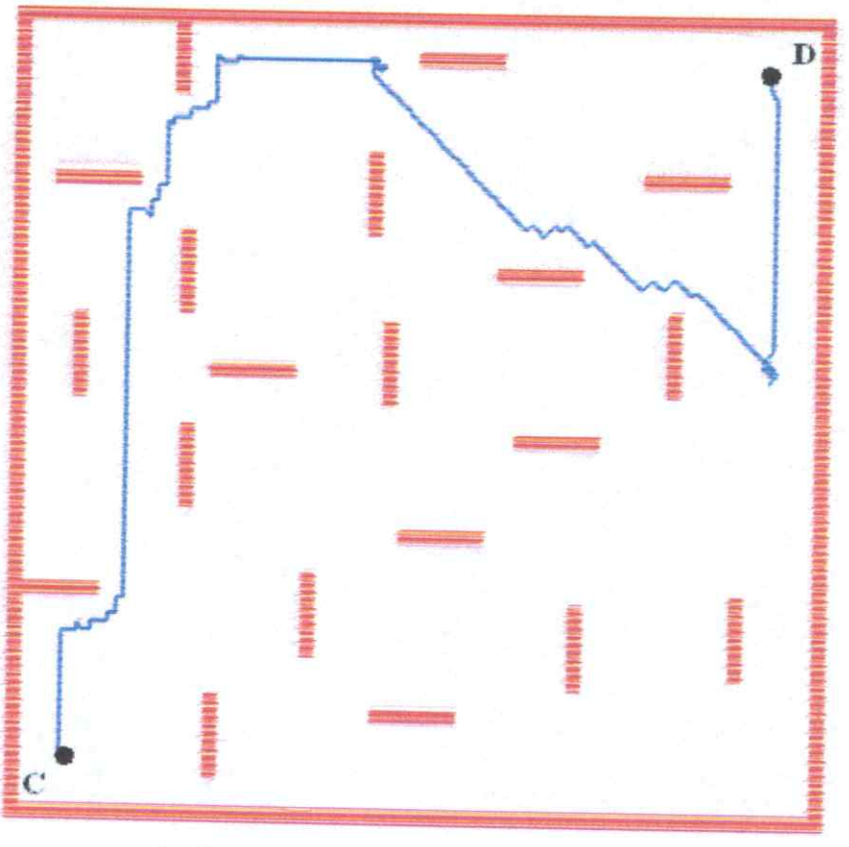


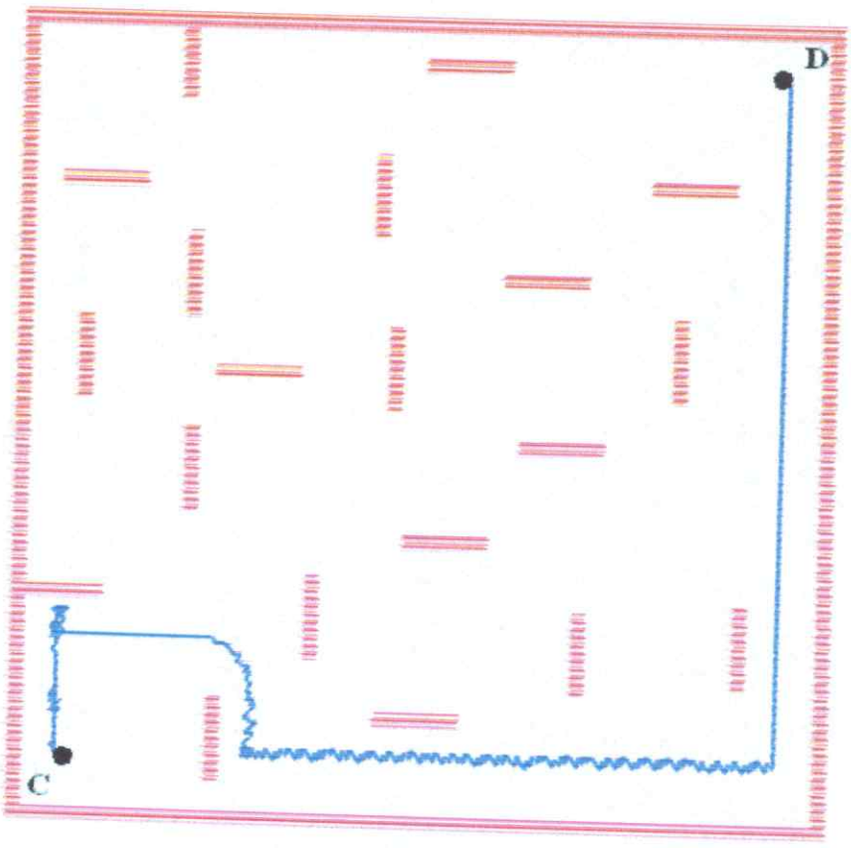
Figure III.30 : Structure de réseau après la phase d'apprentissage (de *B* vers *A*).

Donc le contrôleur a réglé les interactions entre les composants par rapport à la structure de l'environnement et a crée une mémoire qui se symbolise par les affinités entre les anticorps.

Cas 2 : Dans ce cas la navigation du robot sera entre *C* et *D*. La trajectoire du robot à la première itération est représentée par la figure III.31.



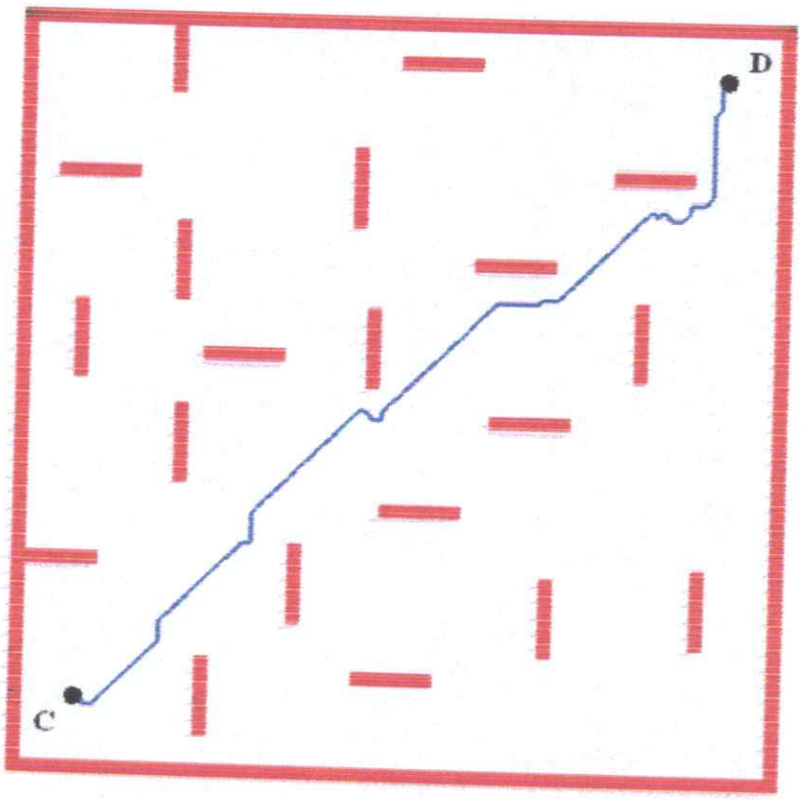
(a) La navigation se fait de *C* vers *D*.



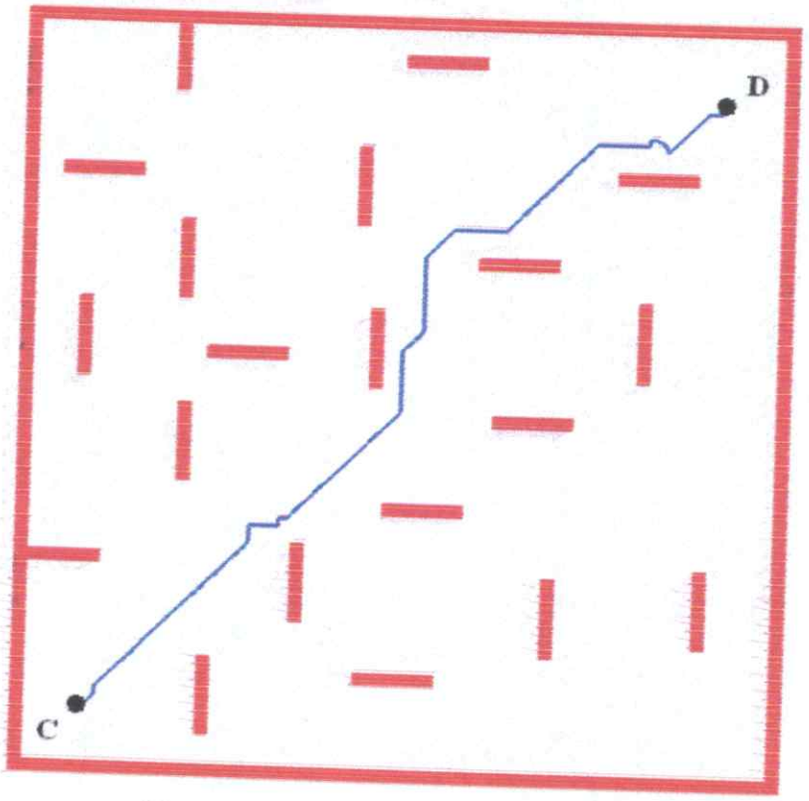
(b) La navigation se fait de *D* vers *C*.

Figure III.31 : Trajectoire du robot dans la première itération.

On remarque que dans la première itération la trajectoire du robot était mauvaise et la durée était assez importante, le robot n'avait pas un comportement intelligent. Ce comportement a était amélioré par la suite, la figure III.32 représente la trajectoire du robot après la phase d'apprentissage.



(a) La navigation se fait de *C* vers *D*.



(b) La navigation se fait de *D* vers *C*.

Figure III.32 : Trajectoire du robot après la phase d'apprentissage.

On remarque que la trajectoire du robot a été largement améliorée, on peut dire que le robot s'est adapté à cet environnement. Pour confirmer ces résultats la figure III.33 présente une courbe de la performance du robot durant ce test.

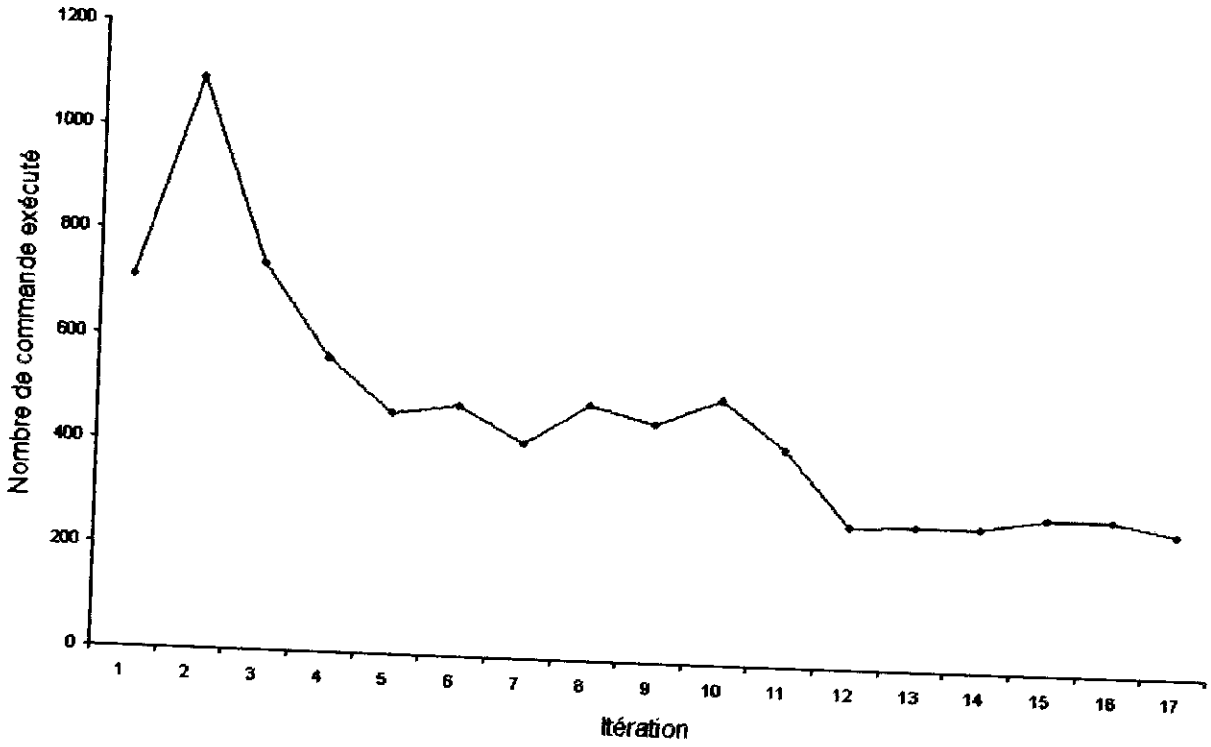


Figure III.33 : Performance du robot durant le test.

Les deux tableaux III.8 et III.9 représentent les valeurs des affinités entre les anticorps après la phase d'apprentissage. Dans ces tableaux, les anticorps des lignes stimulent les anticorps des colonnes, et les anticorps des colonnes suppriment les anticorps des lignes, par exemple la valeur situé à la deuxième ligne de la première colonne représente l'affinité de la stimulation de l'anticorps 0 par l'anticorps 1(ou suppression de l'anticorps 1 par l'anticorps 0).

Tableau III.8 : Les affinités entre les anticorps après la phase d'apprentissage (de *C* vers *D*)

Sm / Sp	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7
AC0	0.0	0.8896	0.1006	0.0874	0.0838	0.0838	0.0838	0.0856
AC1	0.2089	0.0	0.2197	0.1909	0.1831	0.1831	0.1831	0.1869
AC2	0.0957	0.8896	0.0	0.0874	0.0838	0.0838	0.0838	0.0856
AC3	0.0957	0.8896	0.1006	0.0	0.0838	0.0838	0.0838	0.0856
AC4	0.0957	0.8896	0.1006	0.0874	0.0	0.0838	0.0838	0.0856
AC5	0.0957	0.8896	0.1006	0.0874	0.0838	0.0	0.0838	0.0856
AC6	0.0957	0.8896	0.1006	0.0874	0.0838	0.0838	0.0	0.0856
AC7	0.0957	0.8896	0.1006	0.0874	0.0838	0.0838	0.0838	0.0

(Sm : stimulation, Sp : suppression)

Le tableau III.8 correspond à la trajectoire de *C* vers *D* on remarque qu'après l'apprentissage le réseau est construit et sa structure devient stable. On a l'anticorps 1 qui est stimulé par tous autres anticorps donc l'action correspondante aura la plus grande priorité, puis l'anticorps 2 qui est stimulé par les six autres anticorps. La structure du réseau se déduit du tableau des affinités, en comparons les valeurs des affinités entre les anticorps. On prend par exemple deux anticorps *x* et *y* :

- si la valeur de l'affinité de la stimulation de *x* par *y* est nettement supérieur à la valeur de l'affinité de la suppression de *x* par *y* alors *y* stimule *x* et *x* supprime *y*.
- si la valeur de l'affinité de la stimulation de *x* par *y* est proche de la valeur de l'affinité de la suppression de *x* par *y* alors il n'y a pas d'interaction entre *x* et *y*.

La structure dans notre cas peut être illustrée par la figure suivante :

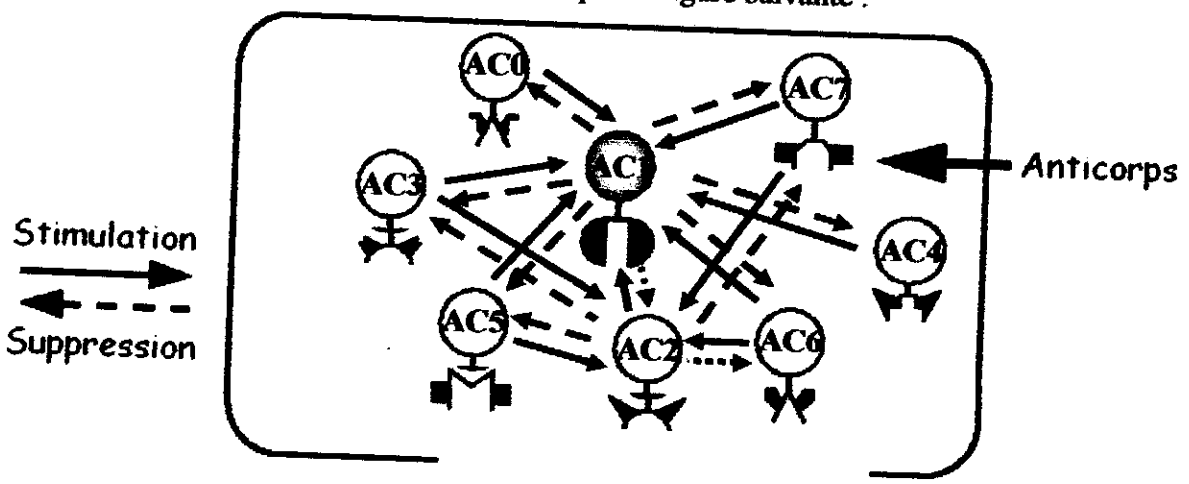


Figure III.34 : Structure de réseau après la phase d'apprentissage (de *C* vers *D*).

Tableau III.9 : Les affinités entre les anticorps après la phase d'apprentissage (de *D* vers *C*)

Sm / Sp	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7
AC0	0.0	0.0253	0.0253	0.0259	0.0336	0.2738	0.0345	0.0268
AC1	0.0141	0.0	0.0140	0.0143	0.0187	0.1520	0.0191	0.0149
AC2	0.0862	0.0859	0.0	0.0878	0.1142	0.9285	0.1170	0.0910
AC3	0.0308	0.0307	0.0307	0.0	0.0408	0.3318	0.0418	0.0325
AC4	0.0189	0.0188	0.0188	0.0192	0.0	0.2037	0.0256	0.0199
AC5	0.0901	0.0898	0.0899	0.0918	0.1194	0.0	0.1223	0.0951
AC6	0.0706	0.0704	0.0705	0.0720	0.0936	0.7610	0.0	0.0745
AC7	0.0279	0.0278	0.0278	0.0284	0.0370	0.3010	0.0379	0.0

(Sm : stimulation, Sp : suppression)

Le tableau III.9 correspond à la trajectoire de *D* vers *C* on remarque qu'après l'apprentissage le réseau est construit et sa structure devient stable. On a l'anticorps 5 qui est stimulé par tous autres anticorps donc l'action correspondante aura la plus grande priorité. On peut aussi remarqué que l'anticorps 2 stimule les anticorps 4 et 6. La structure du réseau se déduit du tableau des affinités, en comparons les valeurs des affinités entre les anticorps. On prend par exemple deux anticorps *x* et *y* :

- si la valeur de l'affinité de la stimulation de *x* par *y* est nettement supérieur à la valeur de l'affinité de la suppression de *x* par *y* alors *y* stimule *x* et *x* supprime *y*.
- si la valeur de l'affinité de la stimulation de *x* par *y* est proche de la valeur de l'affinité de la suppression de *x* par *y* alors il n'y a pas d'interaction entre *x* et *y*.

La structure dans notre cas peut être illustrée par la figure suivante :

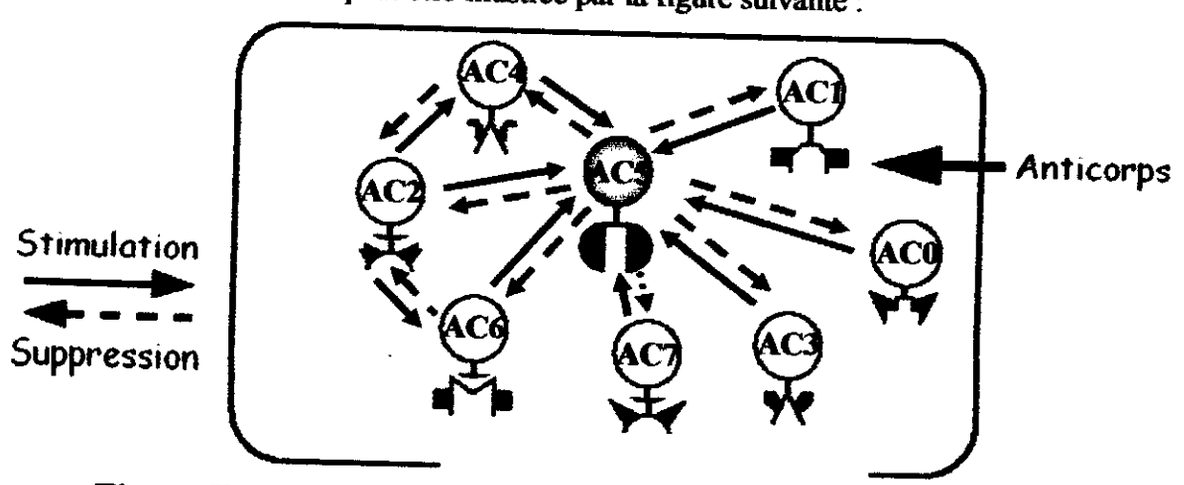


Figure III.35 : Structure de réseau après la phase d'apprentissage (de *D* vers *C*).

Donc le contrôleur a réglé les interactions entre les composants par rapport à la structure de l'environnement et a crée une mémoire qui se symbolise par les affinités entre les anticorps.

3.3. Les tests dans les environnements possédant des pièges

Dans ce type d'environnement les résultats des tests n'étaient pas positifs. La durée que fait le robot pour atteindre un objectif était très grande et dans plusieurs cas le robot se trouve emprisonné dans l'environnement.

Le mécanisme d'adaptation n'a pas abouti à des bons résultats dans ces environnements et le réseau d'anticorps n'a pas convergé vers une structure stable donc il faudra modifier la stratégie de l'adaptation.

La figure III.36 représente un exemple d'un environnement qui possède des situations pièges.

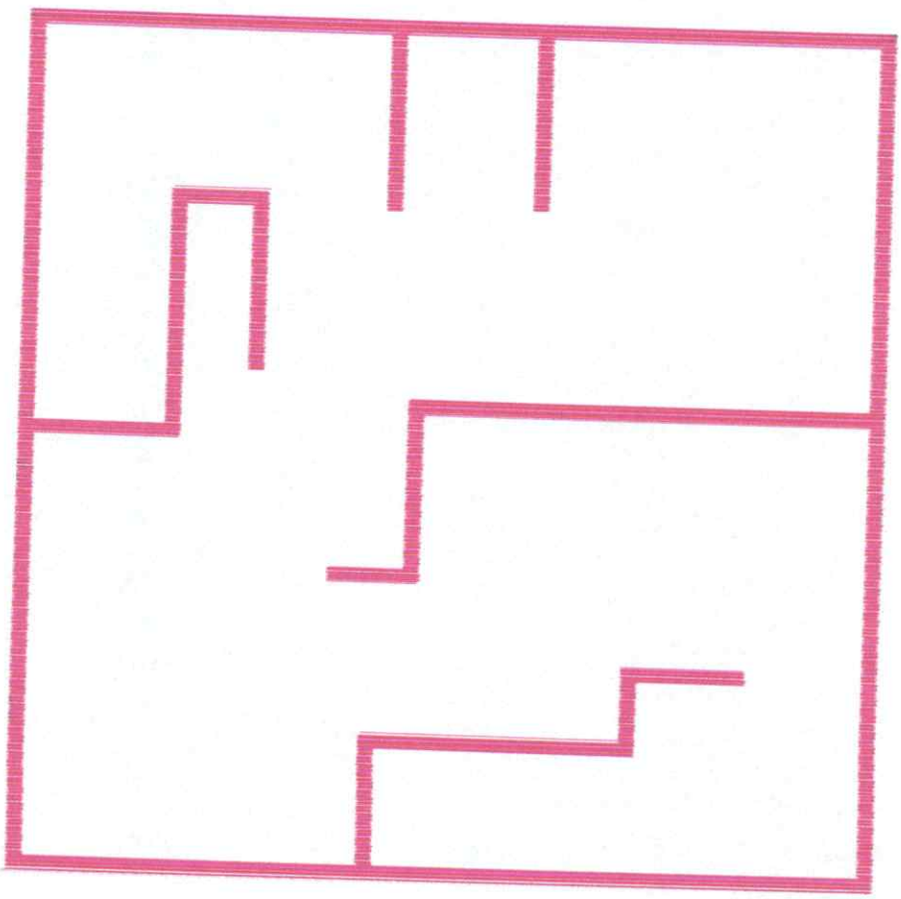


Figure III.36 : Exemple d'un environnement qui possède des pièges.

4. Conclusion

Nous avons présenté, au cours de ce chapitre, la mise en œuvre d'un contrôleur pour un robot mobile autonome, Nous avons commencé la démarche de développement par l'identification des différents besoins du contrôleur, en se basant sur ces besoins, nous avons défini l'architecture de notre système afin qu'il puisse répondre aux exigences, ensuite nous avons élaboré la partie implémentation de notre projet, enfin nous avons fait des tests. Nous pouvons dire que notre système réalise une adaptation du robot dans les environnements inconnus qui ne possèdent pas de pièges mais il est très limité face aux environnements qui possèdent des pièges.

Conclusion générale

Conclusion générale

Dans ce travail, notre objectif était de réaliser un mécanisme d'arbitrage de comportement pour un robot mobile autonome, ce mécanisme doit assurer l'adaptation du robot à des environnements inconnus. L'intérêt du projet est de permettre au robot une navigation autonome sans faire appel à l'homme.

L'autonomie des robots est une préoccupation internationale, cette autonomie est nécessaire pour l'application des robots à des activités dangereuses comme l'exploration des zones nucléaires. Par exemple dans l'exploration spatiale, les communications du robot vers la Terre se font pour l'instant, à très faible débit. Il faut savoir qu'une commande classique nécessite plusieurs heures de communication vers un robot situé sur Mars.

Dans ce mémoire, nous avons présenté les réseaux immunitaires, et la manière de profiter des caractéristiques de ces réseaux pour construire des contrôleurs adaptatifs car les réseaux immunitaires artificielles forment un excellent outil pour le contrôle adaptatif.

En première étape, on a construit un ensemble de comportement de base qui forme les modules de compétences, et on a construit aussi les règles qui permettent de choisir une action parmi l'ensemble possible, ces règles doivent assurer l'évitement d'obstacle qui est réalisé avec succès dans notre système.

En deuxième étape, on a construit un mécanisme d'adaptation, qui est réalisé par le changement d'affinités entre les anticorps du réseau, afin de créer un répertoire d'anticorps qui mémorise la structure de l'environnement. Les tests qu'on a faits ont prouvé que notre système était performant dans certain environnement et était très limites dans d'autre.

Dans notre travail, c'est nous qui avons choisi les modules de compétences du robot, il sera intéressant de créer un système, qui crée automatiquement les modules de

compétences, en offrent au robot la possibilité de la création et la suppression des modules sans intervention humaine.

Le mécanisme d'adaptation qu'on a fait est l'ajustement, qui réalise le changement des paramètres du système, il sera intéressant d'ajouter un autre mécanisme qui assure l'innovation (par l'application des algorithmes génétiques par exemple), cela peut être inspiré de la fonction métadynamique du système immunitaire naturel, qui créé des nouveaux anticorps et supprime d'autres anticorps. Cette solution peut probablement faire face aux limites de notre contrôleur.

Annexes

Annexe A

Les Robots mobiles

1. Définition

Un robot est une machine équipée de capacités de perception, de décision et d'action qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception qu'il en a. Ainsi, le robot devrait être capable d'effectuer des tâches diverses de plusieurs manières et accomplir correctement sa tâche même s'il rencontre de nouvelle situation inattendues.

Cette définition s'illustre par un schéma classique des interactions d'un robot avec son environnement (figure A.1).

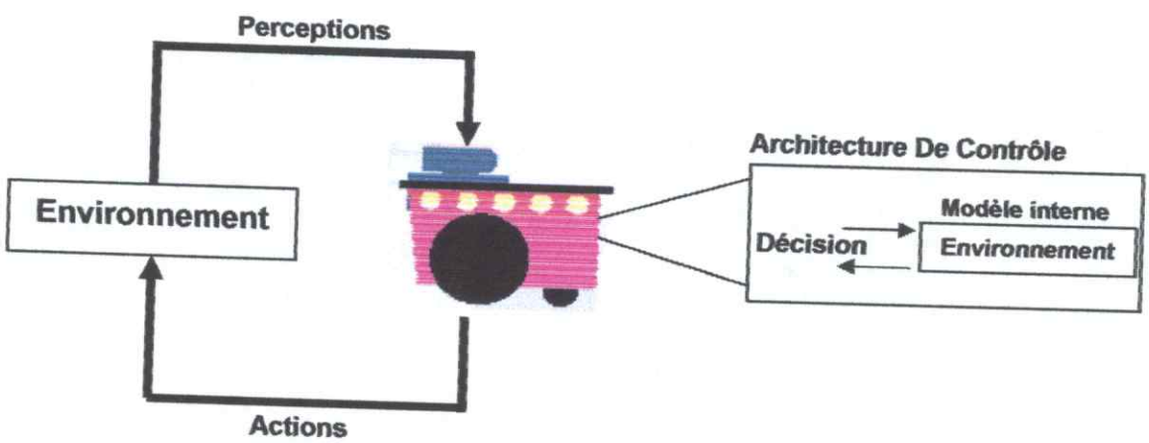


Figure A.1 : Schéma des interactions d'un robot avec son environnement. Le modèle interne de l'environnement est optionnel. Un robot peut fonctionner sans en utiliser.

2. Aperçu historique

Le terme de robot apparaît pour la première fois dans une pièce de Karel Capek en 1920 : Rossum's Universal Robots. Il vient du tchèque 'robota' (servitude) et présente une

vision des robots comme serviteurs dociles et efficaces pour réaliser les taches pénibles mais qui déjà vont se rebeller contre leurs créateurs.

La Tortue construite par GreyWalter dans les années 1950 (Figure A.2), est l'un des tout premiers robots mobiles autonomes. Grey Walter n'utilise que quelques composants analogiques, dont des tubes à vide, mais son robot est capable de se diriger vers une lumière qui marque un but, de s'arrêter face à des obstacles et de recharger ses batteries lorsqu'il arrive dans sa niche. Toutes ces fonctions sont réalisées dans un environnement entièrement préparé, mais restent des fonctions de base qui sont toujours sujets de recherche pour les rendre de plus en plus génériques.

Figure A.2 : La tortue de Grey Walter
(nommée "machina speculatrix" et surnommée Elsie).

Dans les années 60, les recherches en électronique vont conduire, avec l'apparition du transistor, à des robots plus complexes mais qui vont réaliser des tâches similaires. Ainsi le robot "Beast" (Figure A.3) de l'université John Hopkins est capable de se déplacer au centre des couloirs en utilisant des capteurs ultrason, de chercher des prises électriques (noires sur des murs blanc) en utilisant des photodiodes et de s'y recharger.

Les premiers liens entre la recherche en intelligence artificielle et la robotique apparaissent à Stanford en 1969 avec Shakey (Figure A.3). Ce robot utilise des télémètres à ultrason et une caméra et sert de plate-forme pour la recherche en intelligence artificielle, qui à l'époque travaille essentiellement sur des approches symboliques de la planification. La perception de l'environnement, qui à l'époque est considérée comme un problème séparé, voire secondaire, se révèle particulièrement complexe et conduit là aussi à de fortes

contraintes sur l'environnement. Ces développements se poursuivent avec le Stanford Cart dans les années 1980, avec notamment les premières utilisations de la stéréo-vision pour la détection d'obstacles et la modélisation de l'environnement.

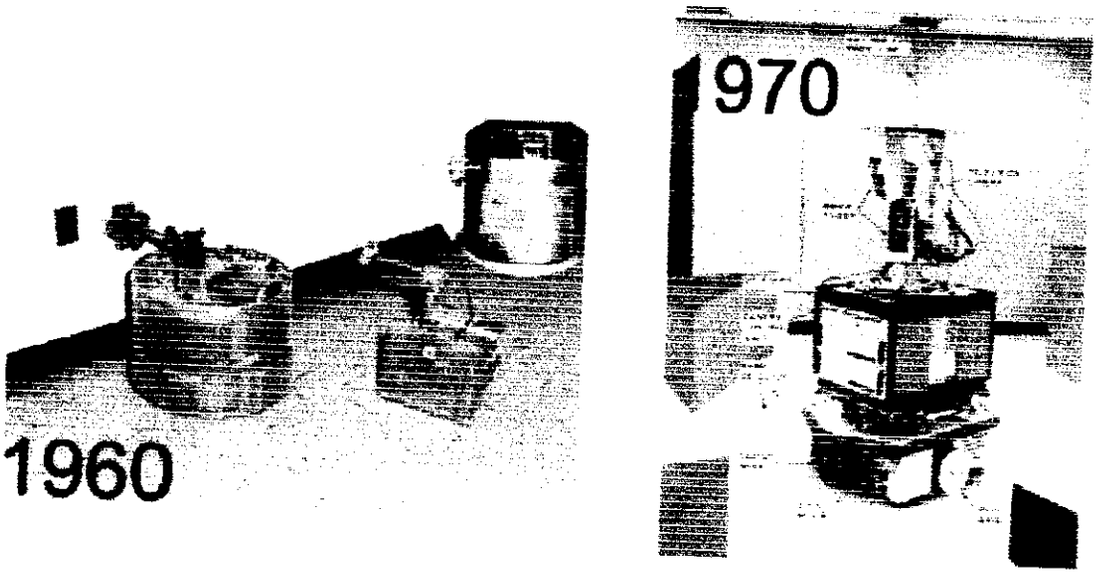


Figure A.3: A gauche : Robot "Beast" de l'université John Hopkins dans les années 1960. A droite : Le robot Shakey de Stanford en 1969 a été une plateforme de démonstration des recherches en intelligence artificielle.

Une étape importante est à signaler au début des années 1990 avec l'apparition de la robotique réactive, représentée notamment par Rodney Brooks. Cette nouvelle approche de la robotique, qui met la perception au centre de la problématique, a permis de passer de gros robots très lents à de petits robots (Figure A.4), beaucoup plus réactifs et adaptés à leur environnement. Ces robots n'utilisent pas ou peu de modélisation du monde, problématique qui s'est avérée être extrêmement complexe.

Ces développements ont continué depuis et l'arrivée sur le marché depuis les années 1990 de plates-formes intégrées a permis à de très nombreux laboratoires de travailler sur la robotique mobile et à conduit à une explosion de la diversité des thèmes de recherche. Ainsi, même si les problèmes de déplacement dans l'espace restent difficiles et cruciaux, des laboratoires ont pu par exemple travailler sur des approches multi-robot, la problématique de l'apprentissage ou sur les problèmes d'interactions entre les hommes et les robots.

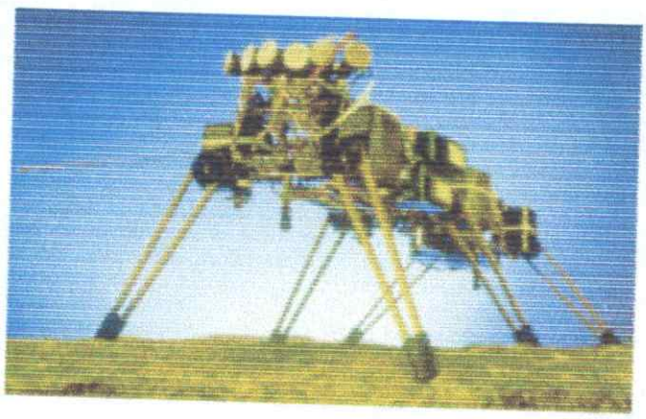


Figure A.4 : Genghis, développé par Rodney Brooks au MIT au début des années 1990.

3. Les sources d'information

La navigation repose sur deux types généraux d'informations différentes : les informations proprioceptives et les informations extéroceptives.

3.1 Informations proprioceptives

Les informations *proprioceptives* sont des informations internes au robot qui le renseignent, dans le cas de la navigation, sur son déplacement dans l'espace. Ces informations peuvent provenir de la mesure de la rotation de ses roues ou de la mesure de l'accélération grâce à une centrale inertielle. Un processus d'intégration permet alors, en accumulant ces informations au cours du temps, d'estimer la position relative de deux points par lesquels le robot est passé.

3.2 Informations extéroceptives

Les informations *extéroceptives* ou plus simplement les perceptions, sont des informations caractéristiques d'une position que le robot peut acquérir dans son environnement. Ces informations peuvent être de nature très variée. Par exemple, un robot peut mesurer la distance des obstacles avec des capteurs infrarouges ou utiliser une caméra.

4. Les applications des robots mobiles

Les applications des robots peuvent se trouver dans de nombreuses activités "ennuyeuses, salissantes ou dangereuses", mais également pour des applications ludiques ou de service, comme l'assistance aux personnes âgées ou handicapées. Parmi les domaines concernés, citons :

- La robotique de service (hôpital, bureaux)
- La robotique de loisir (robot 'compagnon')
- La robotique industrielle ou agricole (entrepôts, récolte de productions agricoles, mines)
- La robotique en environnement dangereux (spatial, industriel, militaire)

A cela, s'ajoute à l'heure actuelle des nombreuses plates-formes conçues essentiellement pour les laboratoires de recherche. La figure A.5 montre quelques exemples de robot réels.

Figure A.5 : Exemples de robots utilisés dans différentes applications.

Annexe B

Le Robot Khepera

1. Présentation

Le robot Khepera est un mini-robot conçu comme outil de recherches et d'enseignement dans le cadre d'un programme suisse de recherches. Il a été développé la première fois en 1992, par une équipe de recherche du laboratoire de microprocesseur et d'interface (LAMI) à l'institut de la technologie fédéral suisse Lausanne (EPFL). Il permet la confrontation au vrai monde des algorithmes développés dans la simulation pour l'exécution de trajectoire, l'action d'éviter d'obstacle, et le prétraitement d'information sensorielle. Le robot Khepera est maintenant largement répandu autour du monde comme plateforme pour différentes expériences et applications de la robotique.

Khepera peut être utilisé sur un bureau, ou connecté à une station de travail par un câble de liaison série du type RS232. Cette liaison permet de lire les valeurs des capteurs et d'envoyer des commandes au robot en temps réel. On peut aussi télécharger un programme de contrôle dans sa mémoire par la liaison série, puis de le déconnecter du câble série afin de pouvoir rester complètement autonome.

Le Khepera (diamètre 55 mm, hauteur 30 mm, poids 80 g dans sa configuration de base) est montré sur la Figure B.1 et schématisé sur la Figure B.2. Bien que de dimension réduite, il dispose d'un processeur embarqué performant (M68331, 32 bits, cadencé à 16 MHz), associé à une EEPROM de 128 K octets et une RAM statique de 256 Koctets. Le BIOS (Basic I/Os System) est le système bas niveau embarqué dans le robot. Il offre des capacités multi-tâches et permet la gestion de plusieurs modules logiciels : acquisition et conversion des données sensorielles, asservissements des moteurs, contrôle de la communication entre différents modules du robot et l'extérieur.

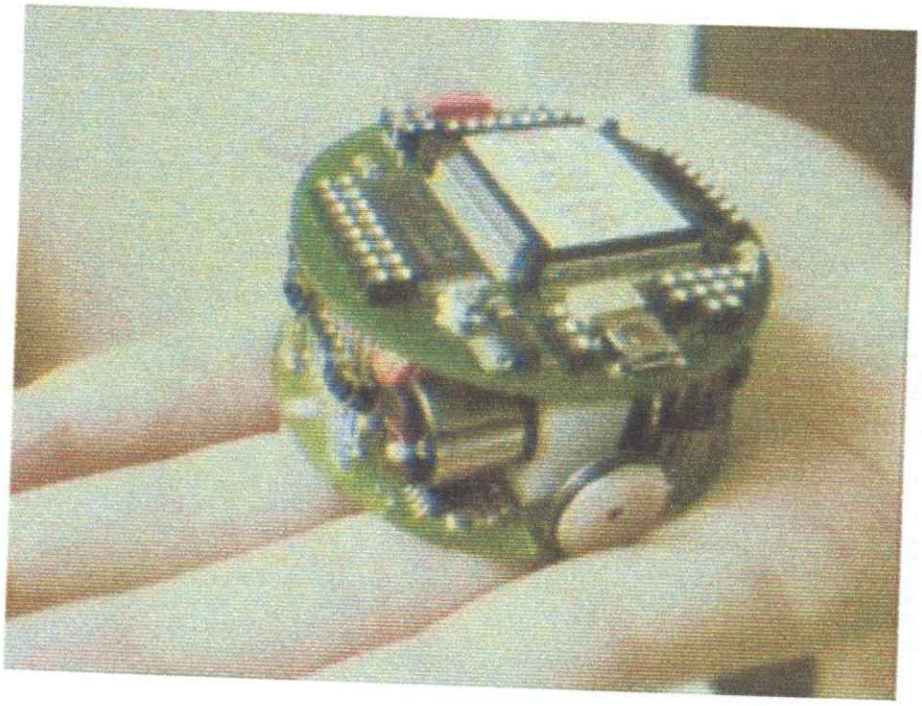


Figure B.1 : Le robot Khepera

Les robots Khepera se composent tous de deux modules : le module de bas comportant la partie motrice et la partie d'alimentation d'une part, le module de contrôle possédant le microcontrôleur et la mémoire d'autre part.

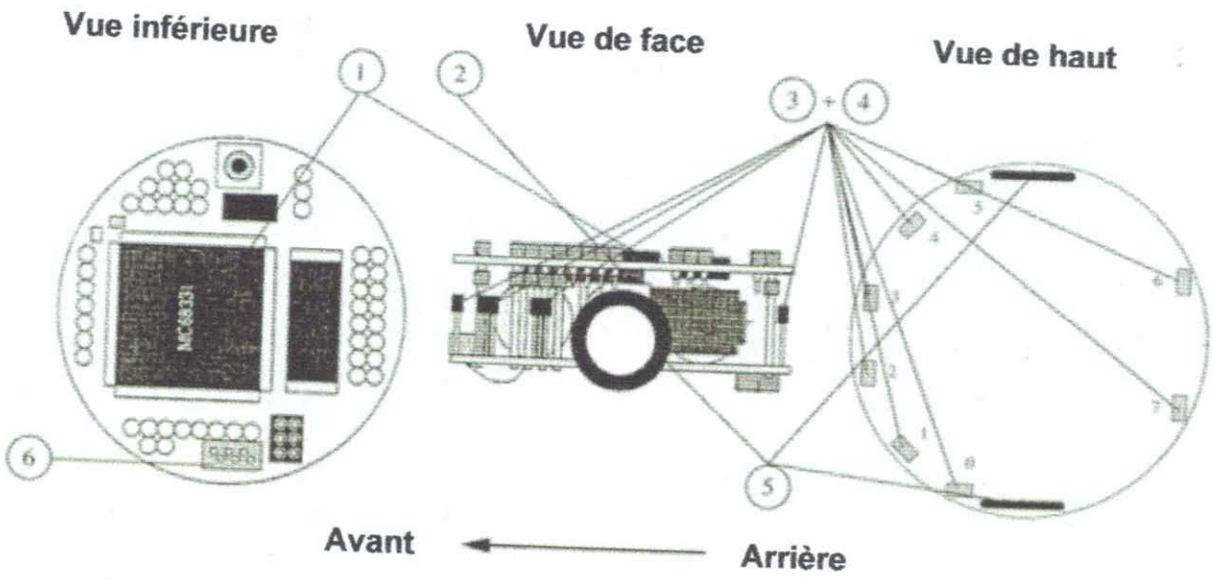


Figure B.2 : Schéma de robot Khepera

Le robot comprend les composants suivants :

1- Un microcontrôleur
2- Une mémoire effaçable et programmable (EPROM), qui permet de télécharger des programmes sur le robot.
3- Huit capteurs infrarouges, six dans l'avant et deux à l'arrière du robot. Chaque capteur infrarouge se compose d'un émetteur et d'un détecteur. L'émetteur envoie un faisceau infrarouge, qui pourrait être reflété par des obstacles. Le faisceau reflété est détecté par le détecteur. De cette façon le robot peut mesurer la distance.
4- huit capteurs de lumières, encore six dans l'avant et deux à l'arrière du robot. Ils sont utilisés pour la mesure de l'intensité lumineuse. Les valeurs sont proches de zéro lorsque l'intensité est forte.
5- deux moteurs actionnant les deux roues indépendamment.
6- une ligne périodique connecteur qui peut être utiliser pour raccorder le robot au port série de l'ordinateur.
7- quatre batteries rechargeables (non montrées dans la figure) pour permettre au robot de fonctionner sans raccordement à un ordinateur pendant environ 45 minutes.

2. Module de contrôle

2.1. Le microcontrôleur :

Le microcontrôleur du Khepera est du modèle MC68331 de Motorola, la puissance de calcul élevée de ce contrôleur fait du Khepera un robot bien adapté à la mise en oeuvre de problèmes de type : réseau de neurones artificiels, déplacements dans un milieu inconnu, ... etc. Le microcontrôleur est responsable du module de base et comporte :

- un microcontrôleur de 32 bits ;
- de la mémoire pour le système et pour l'utilisateur ;
- un bit d'extension ;
- un lien de commination synchrone microcontrôleur ;
- un lien série asynchrone pour la connexion à des machines hôtes ;

2.2. Mémoire :

Le Khepera dispose au totale de 768 Ko de mémoire répartis de la manière suivante :

- un circuit de 512 Ko de ROM, dont le champ d'adresse de $000000 à l'adresse $800000 ;
- deux circuit de 128 Ko de RAM, avec un champ d'adresse de $120000 à $140000 pour RAM1 et de $1A0000 à $1C0000 pour RAM2 ;

Pour les fonctions de base et les initialisations, les concepteurs ont réservé la moitié de RAM1, c'est-à-dire le champ d'adresses de $120000 à $12FFFF.

Pour le contrôle du robot, on trouve en ROM le code correspondant au fichier "ROM.ASM". Ce dernier contient plusieurs modules permettant, au moment du démarrage, d'effectuer les fonctions de base et les premières opérations, telle que :

- initialisation de la RAM ;
- configuration des différents périphériques ;
- exécution du module sélectionné par la configuration des cavaliers ;
- etc.

3. Module de base :

3.1. Généralité

Ce module regroupe les moteurs et les capteurs infrarouges. Chacune des roues est contrôlé par un moteur à courant continu. Les différents capteurs infrarouges présent sur le Khepera lui permettent de détecter des obstacles présents sur sa route afin de pouvoir, par exemple.

3.2. Moteurs a courant continu

Chacune des deux roues du Khepera est couplée à un moteur à courant continu. Le robot peut fonctionner en mode vitesse ou en mode position. Le premier donne une vitesse à

la roue qui ne sera plus modifiée jusqu'à ce qu'une nouvelle vitesse lui soit fournie. Le deuxième sert à déplacer le robot vers une position donnée par l'utilisateur.

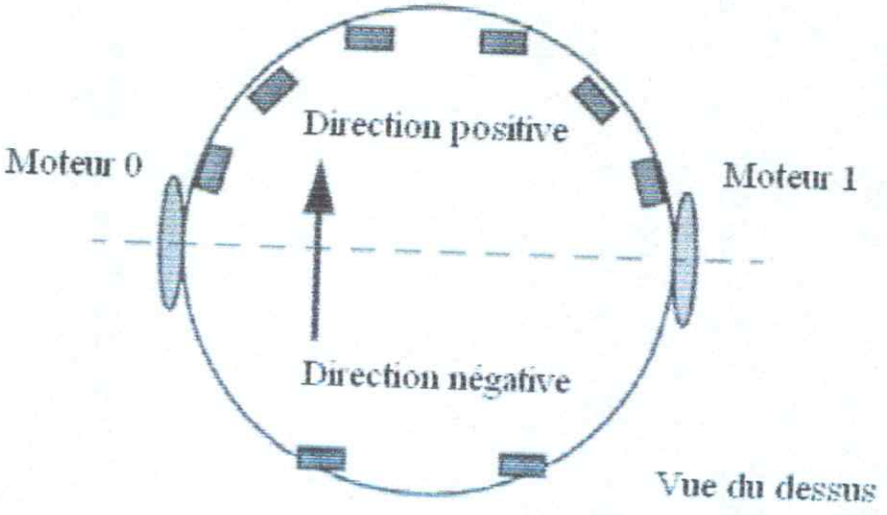


Figure B.3 : Les deux moteurs du Khepera

3.3. Les Capteurs infrarouges

Le robot Khepera utilise des ondes infrarouges pour se déplacer en évitant les obstacles. Pour cela, il se compose de huit capteurs infrarouges pour émette et recevoir ces ondes. Ils sont disposés et numérotés selon la figure B.4.

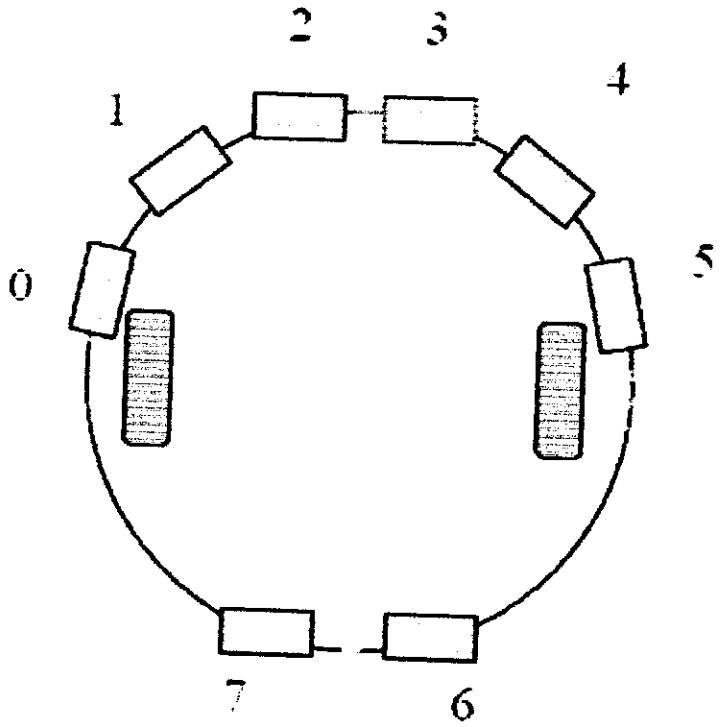


Figure B.4 : disposition et numérotations des capteurs infrarouges

Chacun de ces capteurs est en fait un émetteur/récepteur infrarouge qui peut fonctionner selon les deux modes suivantes :

- mode pulsé
- mode continu

a) **Mode pulsé :** Le mode pulsé consiste à envoyé un signal infrarouge qui, selon la présence ou l'absence d'un obstacle, sera réfléchi ou non. Si l'obstacle est assez près du capteur, le signal réfléchi et mesuré. On obtient ainsi une valeur représentative de la distance entre le capteur et l'obstacle. S'il n'y a pas d'objet ou si l'objet est trop loin du capteur, le signal n'est pas mesurable. La réflexion dépend du type d'objet utilisé, car des paramètres tel que la couleur et la forme d'objet influencent la mesure. La figure B.5 montre les performances d'un capteur face à l'objet.

b) **Mode continu :** Le mode continu permet de mesurer la lumière émise par source lumineuse. Les capteurs prennent comme référence la lumière à l'enclenchement. En suit, ils sont en mesure de détecter une variation de la lumière ambiante, d'identifier la présence d'une source lumineuse.

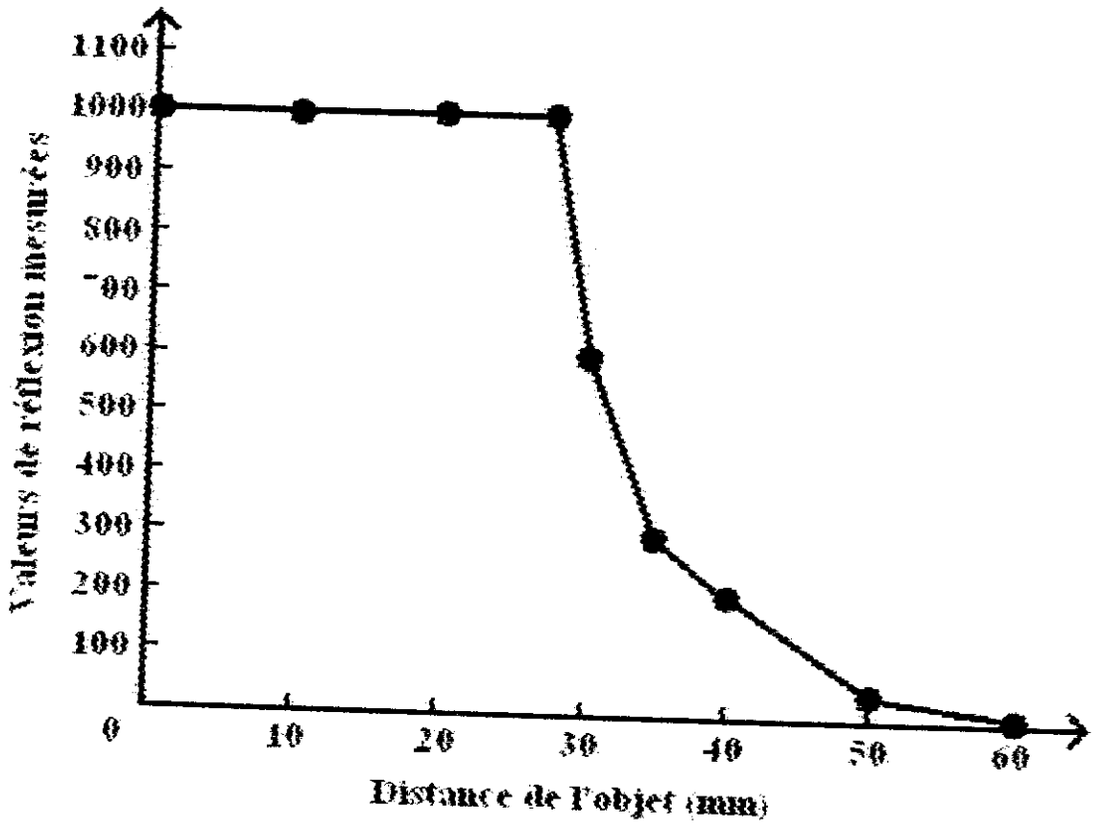


Figure B.5 : Mesure de la réflexion par rapport à la distance séparant le robot de l'obstacle.

Annexe C

WSU Khepera Robot Simulator

1. Présentation

WSU Khepera Robot Simulator v7.2 a été développé par John C. Gallagher et Steven Perretta à l'université Wright State, Dayton, Ohio. Avec le langage Java. Ce logiciel peut être téléchargé à l'adresse : *http://gozer.cs.wright.edu/classes/ceg499/sim/WSU_Sim_v7.2.zip.*

Le simulateur est doté d'une interface graphique conviviale (voir figureC.1) qui permet de créer des environnements personnalisés et d'y déplacer le robot très facilement.

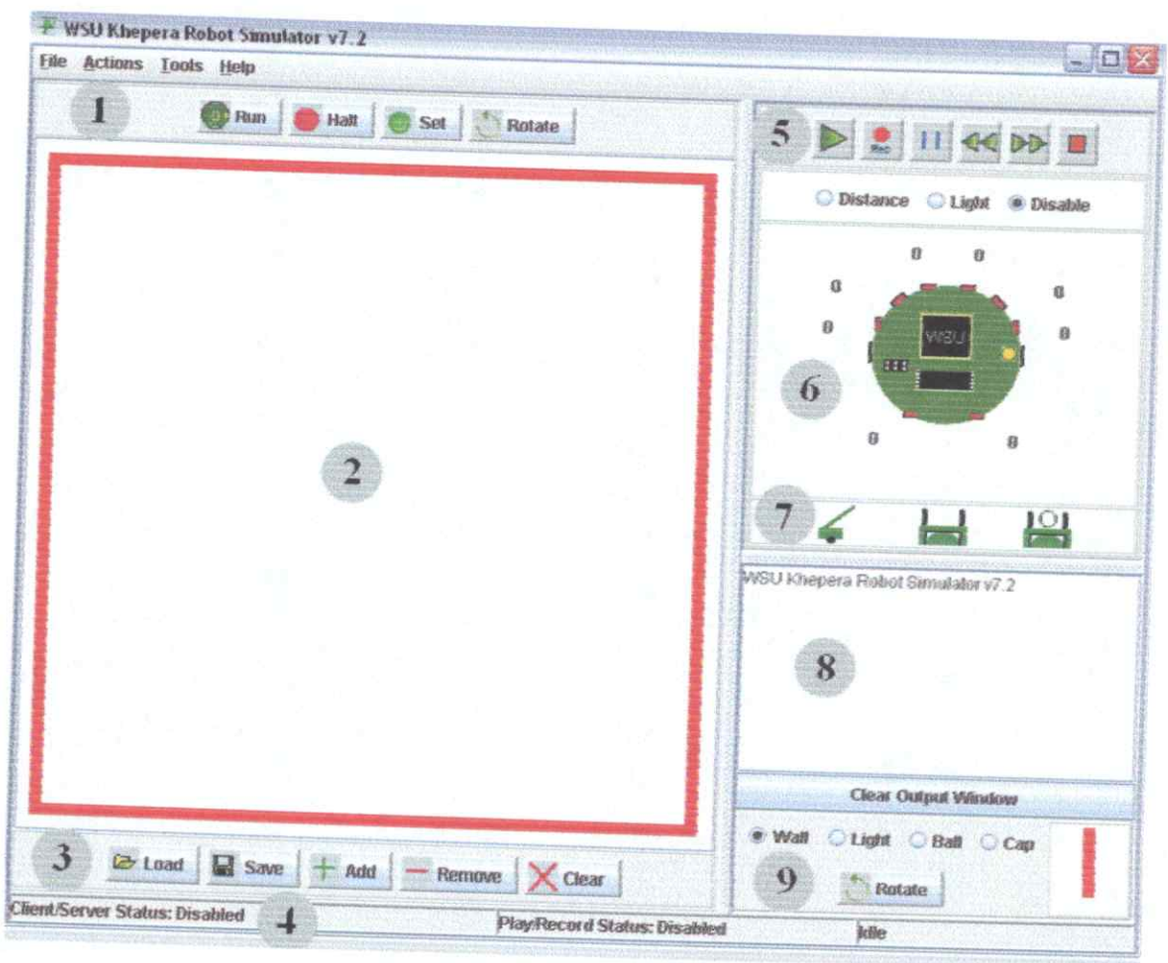


Figure C.1 : L'interface du simulateur, 1) Commande du robot, 2) L'environnement du robot, 3) Commande de l'environnement,

4) Barre de statut, 5) Record/Playback, 6) Affichage des capteurs,
7) Bras du robot, 8) Sortie pour le contrôleur, 9) Choix d'objets.

2. L'interface du simulateur

Les panneaux qui nous intéressent dans notre travail sont :

a). Panneau de commande du robot : il contient des boutons qui permettent de placer le robot et de commencer son exécution. Voici la fonction de chaque bouton :

- *Run* : pour choisir le contrôleur a exécuté parmi des contrôleurs existants.
- *Halt* : pour terminer l'exécution du contrôleur actuel.
- *Set* : pour placer manuellement le robot dans l'environnement.
- *Rotate* : pour tourner le robot par des incréments de 45°.

Figure C.2 : Panneau de commande du robot

b). Panneau de l'environnement du robot : il représente l'environnement simulé du robot c'est ici qu'on placera les obstacles.

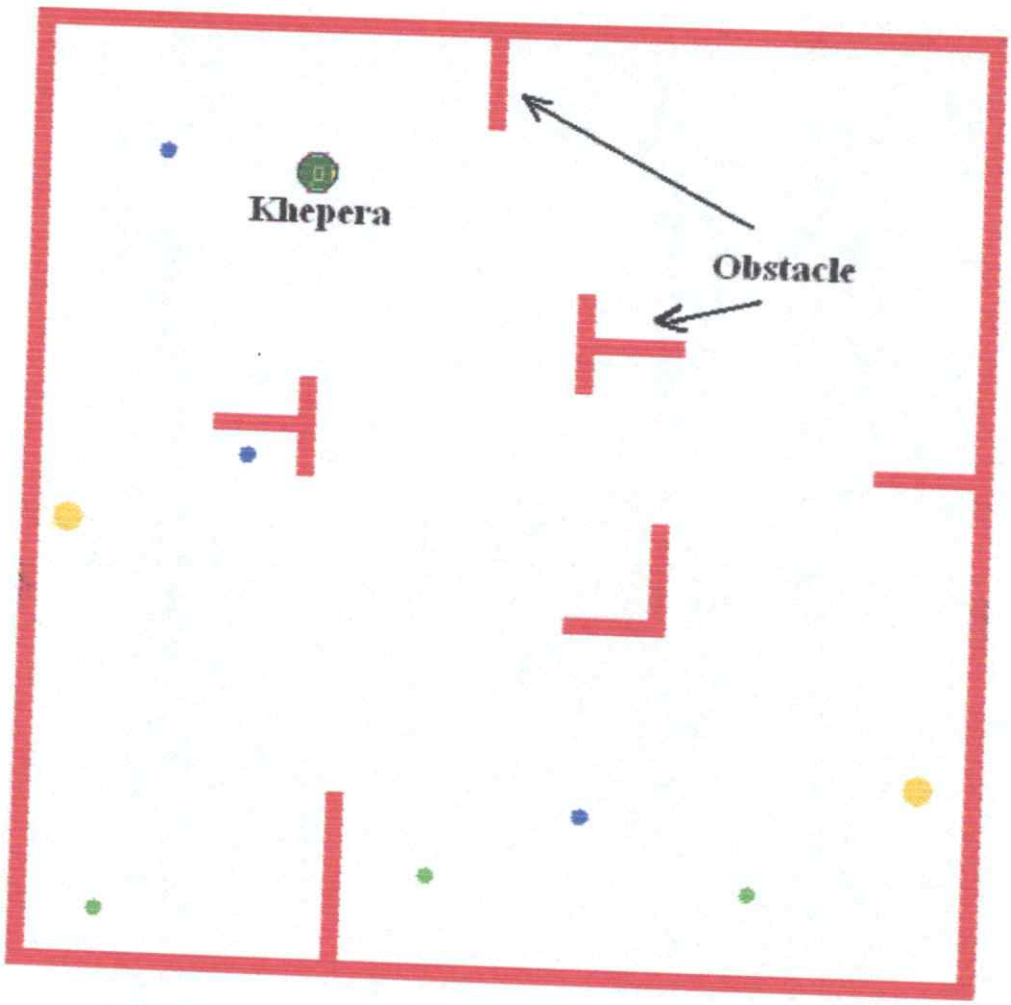


Figure C.3 : Panneau de l'environnement du robot

c). Panneau de commande de l'environnement : il contient des boutons qui permettent d'édité, de chargé, ou de sauvegardé l'environnement. Voici la fonction de chaque bouton :

- *Load* : pour charger à partir d'un fichier un environnement précédemment sauvegardé ;
- *Save* : pour sauvegarder un environnement crée ;
- *Add* : pour ajouter des objets à l'environnement ;
- *Remove* : pour supprimer des objets de l'environnement ;
- *Clear* : pour supprimer tous les objets de l'environnement ;

Figure C.4 : Panneau de commande l'environnement.

d). Panneau d'affichage des capteurs : il affiche la perception du robot pendant la simulation (lecture de distance ou de lumière).

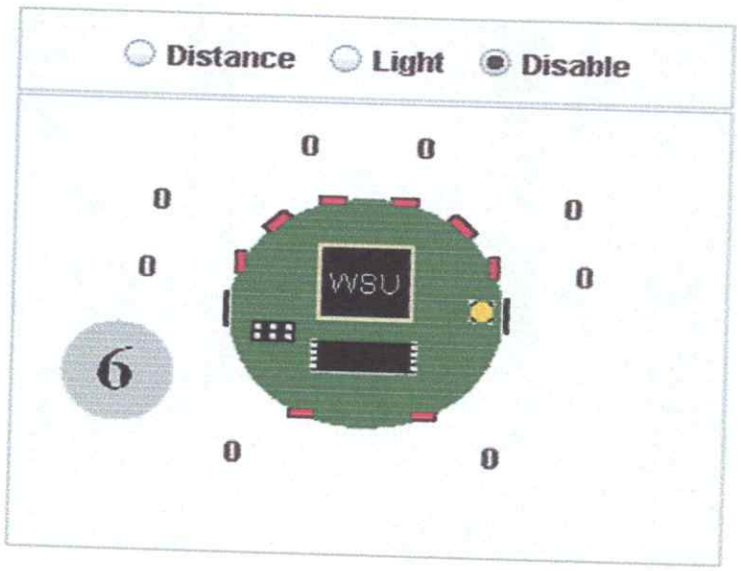


Figure C.5 : Panneau d'affichage des capteurs.

e). Panneau de choix d'objets : il permet de choisir des objets pour construire l'environnement du robot.

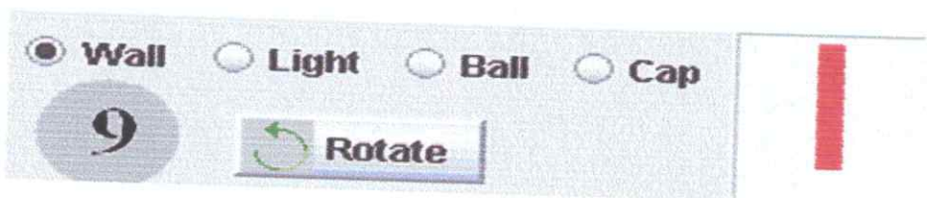


Figure C.6 : Panneau de choix d'objets.

3. L'écriture d'un contrôleur

Le fichier "Template.java" du répertoire "source" montre le format de base pour l'écriture d'un contrôleur et fourni la documentation approprié à chaque partie. Les règles à respecter dans l'écriture des contrôleurs sont :

- Le contrôleur doit importer la classe "edu.wsu.KhepheraSimulator.RobotController" ;
- La classe du contrôleur peut prendre un nom quelconque ;
- La classe du contrôleur doit hériter de la classe "RobotController" ;

- La classe du contrôleur doit définir les deux méthodes : "doWork( )" et "close( )" ;
- Le constructeur de la classe peut contenir n'importe quoi. Il est également un bon endroit pour appeler la méthode "setWaitTime( )" qui modifie la durée d'attente entre deux appels à la méthode "doWork( )" ;
- Ne jamais appeler la méthode "wait( )" pour les attentes mais il faudra utiliser la méthode "sleep( )" de la classe "RobotController" ;

Pour gérer son bon fonctionnement, le simulateur est composé par 34 différentes classes. Pour créer un contrôleur l'utilisateur doit utiliser les API du simulateur qui se présente par 4 classes qui sont :

La classe CurentRobotState : elle fournit une interface qui permet d'accédé au robot par une référence à cette classe, on peut obtenir des références aux capteurs et aux moteurs. Elle définit les méthodes suivantes :

- *Motor getMotorState( )* : retourne une référence sur l'objet moteur, elle est utilisé pour modifié la vitesse du moteur...etc.
- *Sensor[ ] getSensorValues( )* : retourne un vecteurs de huit références sur des objets capteurs.

La classe Sensor : elle définit un capteur infrarouge simple. La classe "Sensor" possède les méthodes suivantes :

- *int getDistValue( )* : retourne la valeur de la sensation actuelle de la distance, cette valeur est comprise entre 0 et 1023.
- *int getLightValue( )* : retourne la valeur de la sensation de la lumière, qui est comprise entre 50 et 510, les valeurs entre 500 et 510 correspondent à un manque complet de la lumière.

La classe Motor : définit une interface qui permet l'accès aux deux moteur du robot, chaque moteur à une vitesse et une position de roue, la vitesse est comprise entre -9 et +9. Cette classe définit les méthodes suivantes :

- *void setMotorSpeeds (int lSpeed, int rSpeed)* : modifie les vitesses des moteurs gauches et droits.

- *long getLeftPosition()* : retourne la position de la roue gauche.
- *long getRightPosition()* : retourne la position de la roue droite.

La classe MessagePasser : elle permet l'envoi des objets String à partir ou vers des instances de la classe Robot. Elle définit les méthodes suivantes :

- *void postMessage(String message)* : prend un argument de type String et l'insère dans le panneau de sortie du simulateur.
- *String getMessage()* : retourne le contenu du panneau d'entrée du simulateur.

Références bibliographiques

Bibliographies

[AND 1990] T. L. Anderson and M. Donath. Autonomous robots and emergent behaviour: A set of primitive behaviors for mobile robot control. In Proc. of the IEEE RSJ Int. Workshop on Intelligent Robots and Systems. volume 2, pages 723-730. Tsuchiura (JP), 1990.

[BER 1990] Bersini, H and Varela, F. (1990). Hints for adaptive problem solving gleaned from immune networks. Parallel Problem Solving from Nature, 1st Workshop. PPSW1. Dortmund, Germany. Pub. Springer-Verlag, pp. 343-354.

[CAS 1999] L.N. Castro, F.J. Von Zuben, "Artificial immune systems : part I – Basic theory and applications", Rapport technique, Décembre 1999.

[CAS 2002] L.N. Castro, "Immune Engineering: A Personal Account", Université de l'Etat de Campinas, Octobre 2002.

[CAS 2002] L.N. Castro, " Immune, Swarm, And Evolutionary Algorithms Part I: Basic Models", Université de l'Etat de Campinas, Novembre 2002.

[CAS 2003] L.N. Castro, J. I. Timmis, "Artificial Immune Systems as a Novel soft computing Paradigm", Université de l'Etat de Campinas, Juillet 2003.

[DEN 2004] A. Deneche, S. Meshoul, M. Batouche, "Une approche hybride pour la reconnaissance de formes en utilisant un Système Immunitaire Artificiel", Laboratoire LIRE, 2004.

[FAR 1986] J.D. Farmer, N.H. Packard, A.S. Perelson, "The immune system, adaptation, and machine learning", Physica 22D 187-204, 1986.

[FIL 2005] D. Filliat, "Robotique mobile", Cours ENSTA, Octobre 2005.

[GAL 2005] J.C. Gallagher, S. Perretta, "WSU Khepera Robot Simulator User's Manual",, université de Wright State, Dayton, Ohio, 24 Mars 2005.

[GAU 1999] E. Gauthier, "Utilisation des réseaux de neurones artificiels pour la commande d'un véhicule autonome", Thèse de doctorat, Institut national polytechnique de Grenoble, 1999.

[HAI 2003] B. Haibe-Kains, "Apprentissage d'une tache de contrôle pour un robot mobile en Lego mindstorms", Mémoire de licence, Université libre de Bruxelles, 2003.

[HAR 1995] I. Harvey, "The artificial evolution of adaptive behaviour", Thèse de Ph.D, University of Sussex, 1995.

[HUM 1995] M. Humphrys. W-learning: Competition among selfish Q-learners. Technical Report 362, University of Cambridge Computer Laboratory, 1995.

[ISH 1996] A. Ishiguro, Y. Watanabe, T. Kondo, Y. Shirai, Y. Uchikawa, "Immunoid: A Robot with a Decentralized Behavior Arbitration Mechanisms Based on the Immune System", Université de Nagoya Japan, 1996.

[ISH 1997] A. Ishiguro, Y. Watanabe, T. Kondo, Y. Shirai, Y. Uchikawa, "Emergent Construction of Artificial Immune Networks for Autonomous Mobile Robots", Université de Nagoya Japan, 1997.

[LIN 1993] L. J. Lin. Reinforcement learning for robots using neural networks. Technical Report CMU-CS-93-103, Carnegie Mellon University, 1993.

[LUH 2004] G. Luh, W. Liu, "Reactive Immune Network Based Mobile Robot Navigation", Université de Tatung Taiwan, 2004.

[MAR 2005] A. Marie, "An idiotypic immune network for mobile Robot control" Université de Nottingham, Septembre 2005.

[MAT 2005] N. Matur, S Karri , "Immune System Based Artificial Intelligence for a Mobile Robot", Texas A&M University-Kingsville, 4 février 2005.

[NAD 2002] R. Nadine, "Description de comportements d'agents autonomes évoluant dans des mondes virtuels", Thèse de doctorat, école nationale supérieure des télécommunication, France, 2002.

[OTE 2001] J. M. Otero, "Programmation réactive de robots Khepera ", Laboratoire d'informatique industrielle, Ecole d'ingénieurs de Genève, Décembre 2001.

[PAY 1990] D. W. Payton, J. K. Rosenblatt, and D. M. Keirsey. Plan guided reaction. IEEE Trans. Systems, Man and Cybernetics, 20(6): 1370-1382, 1990.

[REI 1994] P. Reignier, "Pilotage réactif d'un robot mobile étude du lien entre la perception et l'action", Thèse de doctorat, Institut national polytechnique de Grenoble, 1994.

[ROU 2000] N. Rougier, "Modèles de mémoires pour la navigation autonome", Thèse de doctorat, Université Henri Poincaré - Nancy 1, Octobre 2000.

[TIM 2001] J.I. Timmis, "Artificial immune systems : A novel data analysis technique inspired by the immune network theory", Thèse de Ph.D, Université de Wales, Septembre 2001.

[TIM 2004] J. Timmis, T. Knight, L.N. Castro, E. Hart, "An Overview of Artificial Immune Systems", 2004.

[TRU 1997] O. Trullier et J. A. Meyer. "Biomimetic navigation models and strategies in animats". AI Communications, 10 :79–92, 1997.

[VAR 1988] Varela, F, Coutinho, A, Dupire, B and Vaz, N. (1988). Cognitive Networks : Immune and Neural and Otherwise. Theoretical Immunology: Part Two, SFI Studies in the Sciences of Complexity, 2, pp.359-371.

[WAT 1998] Y. Watanabe, A. Ishiguro, Y. Shirai, Y. Uchikawa, "Emergent construction of behavior arbitration mechanism based on the immune system", Université de Nagoya Japan, 1998.

[WIK 2006] l'encyclopédie libre Wikipédia, [http://fr.wikipedia.org/wiki/Navigation].