RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'enseignement supérieur et de la recherche scientifique

Université SAAD DAHLEB

Mémoire de projet de fin d'études

pour l'obtention du diplôme

d'ingénieur d'état en informatique option: Intelligence Artificielle

Approche de navigation collective basée sur l'Apprentissage par Renforcement d'un groupe de robots mobiles autonomes

Encadré par:

M^elle O.Azouaoui

Jugé par:

M^me Oukid (présidente)

M^me Mokhtari (Membre)

M H.Hantabli (Membre)

Réalisé par:

Bensalem Redha

Chérifi Abdrezak

Promotion 2004

Ce mémoire a été réalisé en collaboration avec le Centre de Développement des Technologies Avancées (C.D.T.A.), Haouch Oukil, Baba-Hassen.

Remerciements

Nous remercions avant le bon Dieu qui nous a aidé à réaliser ce modeste travail.

Nous remercions notre promotrice M^elle^ O. Azouaoui de nous avoir accepté comme stagiaires durant toute cette année de préparation de projet de fin d'études et pour son aide, sa patience, sa disponibilité, et sa compréhension.

Nous remercions tous les enseignants de la faculté des sciences de BLIDA et surtout nos enseignants du département informatique.

Nous remercions les membres du jury pour nous avoir fait l'honneur de juger notre travail.

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 tous nos amis pour leur inquiétude et leur soutien, et tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail.

Table des Matières

Chapitre II : Apprentissage par Renforcement

Chapitre III : Formation des figures géométriques

Chapitre IV : Approche de navigation collective basée sur l'Apprentissage par renforcement d'un groupe de robots mobiles autonomes

Introduction Générale

Introduction Générale

Le terme « robot » est né de la plume du tchèque Karel Capek en 1921 dans sa pièce de théâtre Rossum's Universal Robots [MAT, 00]. Le mot robot a été introduit dans la langue courante en l'adaptant du mot tchèque : robota (travail forcé). En 1940 Westinghouse Electric Corp crée deux premiers robots fonctionnant avec des moteurs électriques. Elektra pouvait danser et compter jusqu'à 10, pendant que le chien Sparko pouvait marcher, se tenir sur ses pattes arrières et japper [COD, 02]. Un peu plus tard en 1968, l'institut de recherche de Stanford construit le premier robot mobile capable de voir, 'Shakey'. Il était équipé d'une caméra de télévision et d'un détecteur de distance. 'Shakey' est le premier robot capable de penser et de réagir aux changements dans son environnement [COD, 02].

Depuis quelques années, on peut constater une augmentation sensible de l'intérêt porté à la robotique mobile, aussi bien dans le domaine de la recherche que dans celui de l'industrie. Plus précisément, plusieurs groupes de laboratoires et d'industriels se sont associés en vue de démontrer qu'il est possible de mettre au point des moyens de transport automatisés [MAT, 00].

Dans ce cadre applicatif, les objectifs visés sont très différents de ceux des premiers temps de la robotique mobile. En effet, les premiers robots mobiles étaient très maniables, et ne pouvaient se déplacer que dans un environnement adapté à leurs besoins, parfaitement connu et immuable. Les techniques de programmation des robots les plus répandues sont basées sur le tout-programmé (c'est-à-dire que l'ensemble des situations envisageables a été programmé). Mais pour des missions complexes d'exploration ou de sauvetage, ces méthodes ne permettent pas de résoudre tous les problèmes, l'ensemble des cas ne peut être envisagé et programmé. Les robots mobiles actuels doivent être capables de se mouvoir dans un environnement plus complexe. Pour que ces robots mobiles puissent réaliser des tâches données dans des environnements partiellement connus, nous allons adopter une approche par apprentissage qui est l'apprentissage par renforcement.

En conséquence, les robots mobiles actuels doivent posséder des capacités de perception évoluées leur permettant de modéliser l'environnement qui les entoure, mais aussi des capacités de décision afin de choisir le comportement le mieux adapté à l'accomplissement d'une mission préalablement fixée, en fonction des capacités de déplacement du robot et aussi de l'état de l'environnement.

Dans certaines situations, un seul robot mobile est incapable de réaliser une tâche complexe, quelque soit son degré d'autonomie, requérant impérativement la participation de plusieurs robots mobiles. Ces robots mobiles doivent travailler en groupe pour effectuer toutes sortes de tâches. Par exemple, un groupe de robots peut être utilisé pour surveiller des bâtiments, explorer des espaces dangereux ou inconnus, transporter de gros objets, etc.

Le but global de notre projet est d'étudier les comportements des robots mobiles autonomes en groupe pour réaliser une navigation coordonnée dans des environnements inconnus pour une tâche donnée.

Nous avons tout d'abord étudié la navigation individuelle et en groupe des robots mobiles autonomes. Nous avons par la suite développé une approche d'évolution en groupe de robots mobiles, basée sur un apprentissage par renforcement, qui a pour objectif de maximiser la performance du système en renforçant les meilleures actions, et finalement, une phase de simulation est présentée afin de valider l'approche proposée.

Ce mémoire est organisé en 5 chapitres :

Le Premier Chapitre aborde divers aspects de la robotique mobile, sur leur architecture, leur coopération, et sur les approches de navigation telles que les approches locales, les approches globales et les approches mixtes.

Le Deuxième Chapitre traite de l'apprentissage par renforcement. Il présente ainsi les méthodes élémentaires de résolution telles que la programmation dynamique, la méthode de Monté Carlo, etc.

Le Troisième Chapitre introduit la navigation en groupe, la coopération et la coordination. Le groupe de robots doit se déplacer en formation, c'est-à-dire que les robots doivent se déplacer selon des configurations géométriques telles que le cercle, polygone, ou ligne. Ces formes géométriques sont réalisées à l'aide des algorithmes qui sont présentés dans ce même chapitre.

Dans **le chapitre quatre**, nous détaillons l'approche de navigation basée sur l'apprentissage par renforcement que nous avons proposé, et nous présentons l'implémentation de notre étude et les solutions apportées aux problèmes rencontrés.

Le Cinquième et dernier Chapitre introduit les résultats obtenus en simulation.

CHAPITRE

I

I Introduction à la robotique mobile autonome

I.1 Introduction

Le domaine de la robotique mobile est relativement récent et a connu une évolution rapide. Un grand effort a été réalisé pour doter les robots d'une intelligence s'appuyant sur des processus de raisonnement. Les robots sont alors capables de réaliser des tâches de plus en plus complexes, comme les robots géologues envoyés sur Mars [LEM, 04]. Un robot mobile autonome peut avoir par exemple un rôle d'exploitation ou d'intervention, etc.

Il est préférable d'utiliser un groupe de robots mobiles pour accomplir une tâche globale comme par exemple transporter un grand objet ou pour coopérer pour se localiser et construire une carte (construire un modèle d'environnement), etc. L'avantage du travail collectif sur le travail individuel n'est plus à démontrer, travailler en groupe diminue le temps nécessaire pour effectuer une tâche dès que celle-ci est parallélisable ou encore permet d'exécuter des tâches impossibles à réaliser par un individu seul [LUC, 03].

Ce chapitre inclut des généralités et des définitions sur la robotique mobile autonome et leur navigation, et les différentes approches de planification, et ensuite un aperçu sur la robotique collective, ainsi que les différentes méthodes de coopération sont présentées.

I.2 Robot mobile autonome

I.2.1 Définition

Un robot est une machine physique qui agit sur son environnement pour atteindre un objectif. Cette définition est incomplète car il existe des machines automatiques antérieures ou modernes à l'existence de robot qui entrent dans cette définition, par exemple les machines-outils [COI, 01]. Pour une définition complète il faut que l'on ajoute deux propriétés :

- La versatilité.
-L'auto-adaptativité.

La versatilité : le robot devrait être capable d'effectuer des tâches diverses de plusieurs manières [COI, 01].

L'auto-adaptativité : le robot doit pouvoir accomplir correctement sa tâche même s'il rencontre de nouvelles situations inattendues. La plupart des robots industriels ne possèdent pas cette propriété car ne possédant pas de système de perception [COI, 01].

Dans la robotique mobile, les tâches peuvent être par exemple des tâches de manutention (robot de transport hospitalier) ou de nettoyage. La robotique mobile peut également avoir un rôle d'exploration et d'intervention en milieu hostile à l'homme (exploration sous marine, véhicule planétaire, etc) [COI, 01].

Nous nous intéressons au cours de cette étude à une sous famille des robots mobiles existants appelée *les robots mobiles autonomes*. Un robot mobile 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 imprévues sans intervention humaine [COI, 01].

I.3 Navigation des robots mobiles autonomes

I.3.1 Définition

Le but de la navigation n'est pas seulement de permettre au robot d'éviter les obstacles, mais d'amener également le robot dans une zone particulière tout en évitant tous types d'obstacles. Elle utilise des cheminements et des transformations de l'information entre les capteurs et les effecteurs [KHI, 97].

I.3.2 Étapes de navigation

La navigation distingue clairement les étapes qu'on résume dans le schéma de la **Figure I.1**.

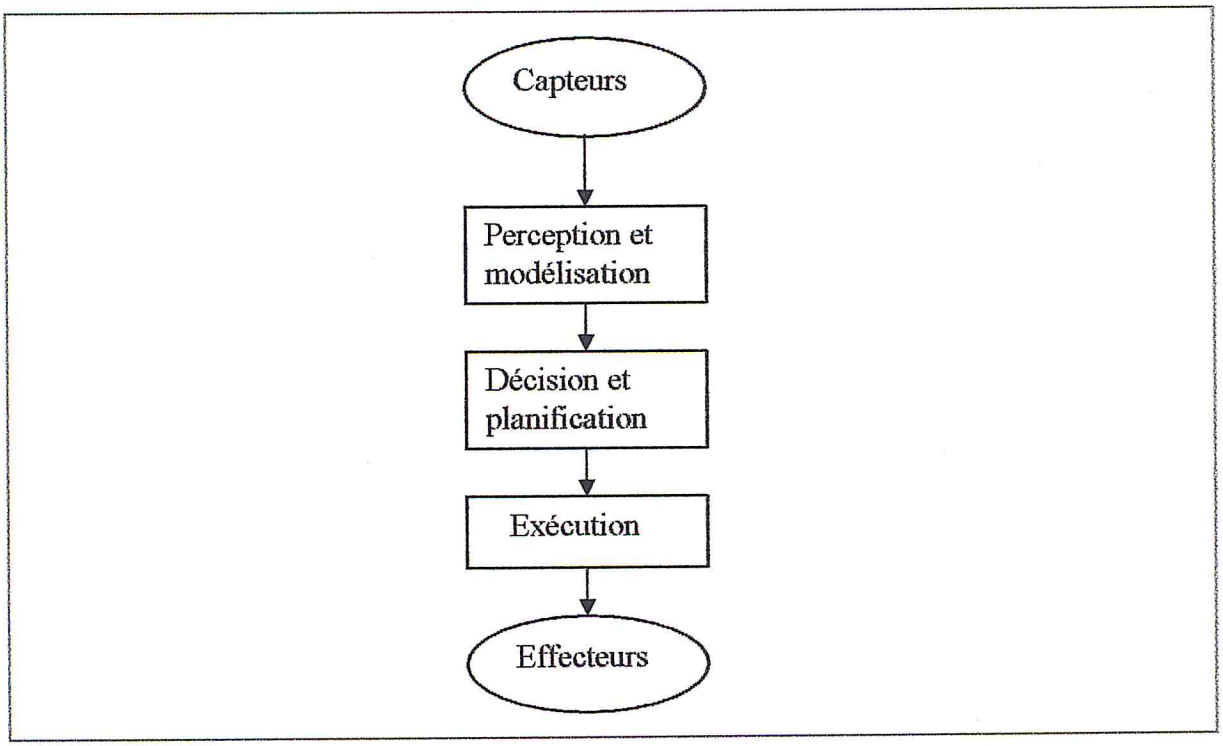


Figure I.1 : Etapes de navigation.

I.3.2.1 Perception et modélisation

La perception permet de détecter l'environnement proche et éloigné du robot. Elle est nécessaire pour la sécurité du robot, pour la modélisation de l'environnement et pour sa localisation dans cet environnement [PRU, 98].

L'étape de modélisation a pour but de maintenir un modèle d'environnement. Ce modèle intègre les données situées dans l'environnement.

I.3.2.2 Décision et planification

Il est nécessaire de planifier la suite d'actions à entreprendre afin de résoudre le but qui lui est fixé. Le robot doit donc effectuer une décision du chemin à suivre tout en évitant tous type d'obstacles.

I.3.2.3 Exécution

Le robot dispose d'actionneurs (moteurs) permettant son déplacement. Le rôle de ces actionneurs est d'exécuter les commandes correspondant aux décisions.

I.4 Capteurs

Les capteurs sont en fait des outils qui transforment une grandeur physique (grandeur à mesurer) en une grandeur électrique (signal de mesure correspondant à la grandeur mesurée) adaptée au traitement de l'information.

Les capteurs qui équipent un robot mobile doivent assurer la perception de l'environnement (capteurs extéroceptifs) et la localisation dans cet environnement.

La localisation permet de se positionner par rapport à une position initiale ou par rapport à l'environnement connu d'avance. Les capteurs de localisation sont appelés capteurs proprioceptifs.

I.4.1 Capteurs proprioceptifs

Les capteurs proprioceptifs sont destinés à donner les grandeurs réelles de la position, de la vitesse et de l'accélération du robot. Parmi les capteurs proprioceptifs les plus connus est l'**odomètre**. Ce capteur est utilisé dans l'estimation de la position actuelle du robot dans l'environnement. L'odométrie est un outil de mesure intégrant les rotations élémentaires des roues du robot pour en déduire son mouvement [PRU, 98].

I.4.2 Capteurs extéroceptifs

L'obtention d'un état de l'environnement s'exécute selon deux types de principe :

- L'environnement émet une énergie propre détectable par des capteurs appelés capteurs passifs.

- Ou des capteurs émettent de l'énergie vers la cible à télémètre et d'en analyser le retour. Ces capteurs sont appelés capteurs actifs [PRU, 98].

I.4.2.1 Capteurs passifs

Les capteurs passifs sont des capteurs qui détectent l'énergie émise par l'environnement. Dans la plupart des cas, se sont des cameras permettant d'obtenir des informations bidirectionnelles, la connaissance de la position du robot assure la détermination de la profondeur. L'énergie émise dans ce cas est la lumière. Il existe d'autres capteurs passifs qui détectent la chaleur ou le son [PRU, 98].

I.4.2.2 Capteurs actifs

Les capteurs actifs ont pour point commun l'émission de l'énergie vers la cible et d'en analyser le retour. Il existe une grande diversité de capteurs actifs (Capteurs lasers, ultrason, infrarouge, etc.).

Capteur laser : son fonctionnement est très simple. Le capteur envoie un signal et initialise un compteur (t=0) et arrête le compteur à l'arrivée du retour du signal et ensuite utilise la vitesse et le temps pour calculer la distance [ABI, 03].

Capteur infrarouge : c'est un capteur qui mesure l'angle d'incidence de la lumière réfléchie par un objet (obstacle, mur, etc.) (Voir **Figure I.2**) [ABI, 03].

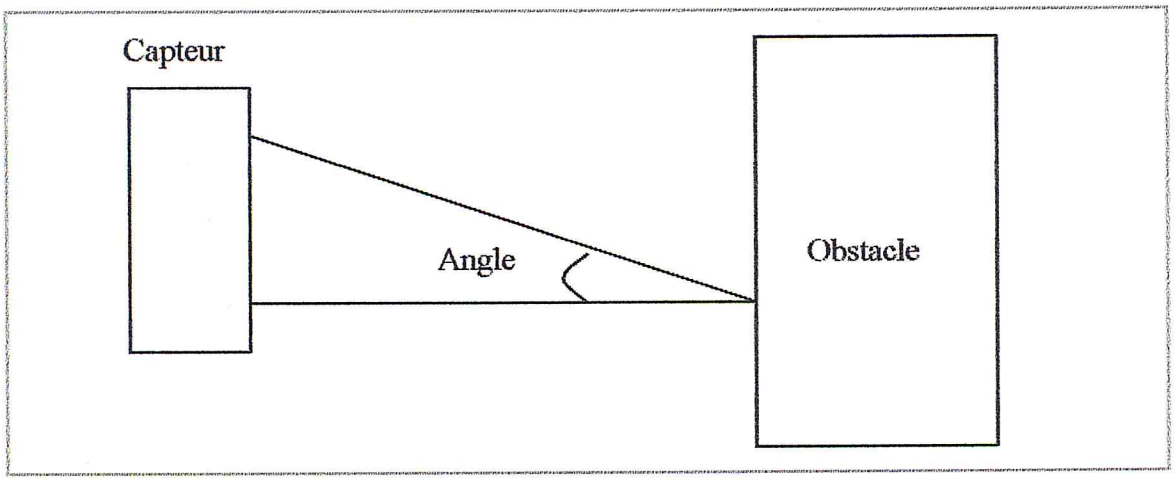


Figure I.2 : Capteur infrarouge.

I.5 Approches de planification et de navigation

La planification consiste à déterminer pour un robot *R* évoluant dans un environnement W une séquence continue de configurations reliant les configurations initiales et finales considérées, tout en respectant un certain nombre de contraintes et de critères. Une configuration est définie par m-uplet de paramètres indépendants de position /orientation caractérisant le robot dans son environnement.

Il existe un grand nombre de méthodes de planification qui sont regroupées en trois approches :

-Approche globale.
-Approche locale.
-Approche mixte.

I.5.1 Approche globale

Les méthodes globales sont appliquées dans un environnement entièrement connu. Elles consistent à capturer la connexité de l'espace libre dans un graphe [KHI, 97]. A titre d'exemple on peut citer deux méthodes :

- Méthode par décomposition cellulaire.
- Méthode par squelette.

I.5.1.1 Méthode par décomposition cellulaire

Cette méthode consiste à partitionner l'espace de configuration libre en un ensemble de régions (cellules) puis construire le graphe de connectivité dont les nœuds correspondent aux différentes régions et les arcs aux relations d'adjacence entre elles (voir **Figure I.3** ci-dessous) [KHI, 97].

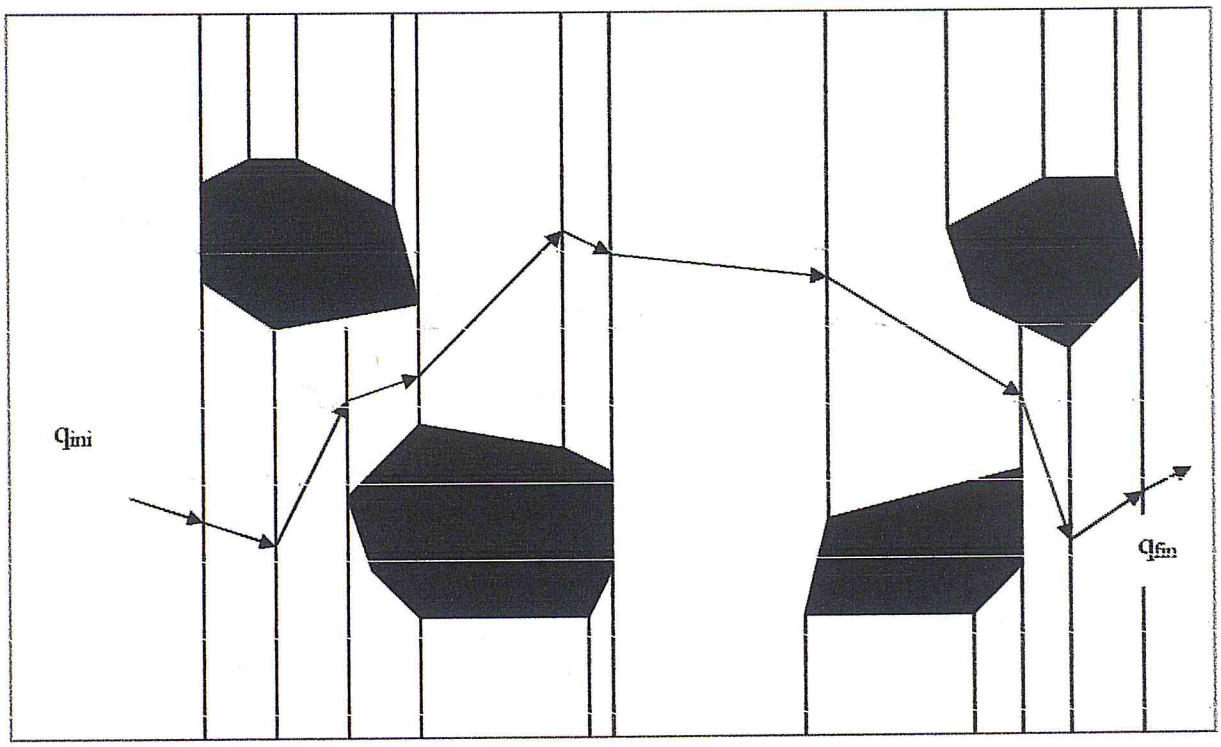


Figure I.3 : Chemin libre : connexion de q_{ini} à q_{fin} via les points milieux des arêtes communes de 2 cellules adjacentes.

I.5.1.2 Méthode par squelette

Un squelette est une concaténation de portions de courbes. Il est structuré en un graphe dont les arêtes sont ces portions de courbes et les noeuds leurs extrémités.

Il existe plusieurs types de squelettes entre autres : les graphes de visibilités et les 'freeways' :

1. *Graphe de visibilité* : consiste à construire un graphe dont les nœuds sont les sommets des obstacles et les arêtes sont les segments reliant deux sommets [KHI, 97].

2. *Les 'freeways'* : Consiste à diviser l'espace libre en un ensemble de couloirs ou cônes (*'freeways'*). Les arêtes du graphe sont les'freeways', et les noeuds sont leurs extrémités [KHI, 97].

I.5.2 Approche locale

Le principe de l'approche locale consiste à déterminer les mouvements du robot mobile en ne considérant qu'une représentation locale de l'environnement et à percevoir la planification de mouvement comme un problème d'optimisation [KHI, 97]. On peut citer deux méthodes :

- Méthode des champs de potentiel.
- Méthode par apprentissage.

I.5.2.1 Méthode des champs de potentiel

Cette méthode est largement utilisée et consiste à assimiler le robot à une particule contrainte à se déplacer dans un champ de potentiel induit par les obstacles et le but à atteindre. Le but attire et les obstacles repoussent (voir **Figure I.4** ci-dessous) [KHI, 97].

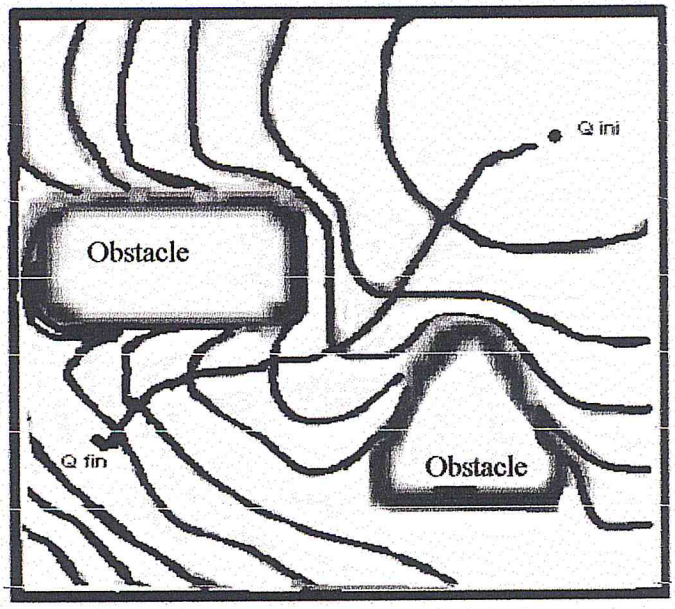


Figure I.4: Principe du champ de potentiel.

I.5.2.2 Méthodes par apprentissage

Il s'agit ici de collecter de nouvelles connaissances, à partir d'un ensemble de techniques inspirées des méthodes essai-erreur, et vise à déterminer une loi de commande pour un système autonome dans un environnement inconnu. Les techniques d'apprentissage en robotique sont directement inspirées de l'étude du vivant et peuvent être regroupées selon les trois exemples ci-dessous [LUC, 03] :

- L'apprentissage par l'exemple,
- Méthodes évolutionnistes,
- L'apprentissage par renforcement.

I.5.2.2.1 Apprentissage par exemple

Dans l'apprentissage par l'exemple, on peut citer les réseaux de neurones. Comme son nom l'indique, cette technique est directement inspirée de l'étude du cerveau. L'objectif

consiste à montrer une série d'exemples, chaque exemple étant associé à un stimulus d'entrée. Après cette phase d'apprentissage, si le réseau est stimulé avec une configuration d'entrées, alors il saura automatiquement de quel exemple il s'agit. Les réseaux de neurones trouvent de nombreuses applications dans de nombreux domaines, notamment en classification de données ou en reconnaissance vocale. Les méthodes d'apprentissage par l'exemple sont supervisées, elles nécessitent des exemples pour réaliser l'apprentissage et elles sont difficilement applicables dans les cas d'auto-apprentissage et d'adaptation où aucun exemple de comportement n'est connu [LUC, 03].

I.5.2.2.2 Méthodes évolutionnistes

Les algorithmes évolutionnistes sont inspirés de l'évolution darwinienne. Le principe général consiste à attribuer à chaque individu une séquence génétique correspondant à un comportement. Régulièrement, les individus sont rassemblés pour pouvoir effectuer des croisements et des mutations sur leurs chaînes chromosomiques. L'apprentissage est donc découpé en générations successives et les performances de l'ensemble de la population sont ainsi accrues au fil de l'évolution [LUC, 03].

I.5.2.2.3 Apprentissage par renforcement

Cette technique consiste à déterminer quelles sont les meilleures actions à réaliser. L'agent obtient des informations de l'état de l'environnement (perceptions) et agit également sur l'environnement puis reçoit une estimation de sa performance : la récompense [LUC, 03]. Nous allons présenter en détails cette techniques d'apprentissage dans le deuxième chapitre.

I.5.3 Approche mixte

L'intérêt de cette approche est de profiter des avantages des deux approches précédentes, à savoir :

- Temps de calcul faible pour l'approche locale.
- Chemin généré optimal pour l'approche globale.

Le principe de cette approche est de générer un chemin optimal, en utilisant une méthode globale, donnant l'allure générale de la trajectoire à suivre. Ensuite, le robot exécutera cette trajectoire, à l'aide d'un générateur local de trajectoire et de son système de perception, tout en évitant de rentrer en collision avec des obstacles imprévus au moment de la modélisation (changements éventuels de l'environnement, obstacles mobiles) [KHI, 97].

Un robot mobile seul, peut être incapable de réaliser certaines tâches ou des tâches impossibles de façon individuelle. Donc, il faut ajouter d'autres robots mobiles pour réaliser ces tâches en utilisant les mécanismes de coopération et auto-organisation. Nous allons voir ces mécanismes dans les paragraphes qui suivent.

I.6 Robotique collective

La robotique collective est maintenant envisagée pour un bon nombre d'applications dans des domaines tels que la défense, l'exploration planétaire, l'agriculture, les usines nucléaires, l'exploration sous-marine, la délivrance (libération) dans les environnements hostiles. Les systèmes multi-robots présentent beaucoup d'avantages comme la robustesse, la résistance aux perturbations inattendues [LUC, 01]. On trouve dans la robotique collective les différents moyens permettent la mise en place d'un groupe de robots. Nous pouvons citer :

- L'auto-organisation,
- La coopération.

I.6.1 Auto-organisation

I.6.1.1 Définition

En 1954 Farley et Clark du laboratoire Lincoln sortent la notion de l'auto-organisation [MAL, 99]. L'auto-organisation est le fait de passer d'un état stable à un autre de façon autonome, et elle permet à une organisation (robots) de s'adapter elle-même aux situations dynamiquement changeantes [MAL, 99].

L'auto-organisation est un moyen pour parvenir à gérer la dynamique de l'environnement de manière autonome.

I.6.1.2 Mécanismes d'auto-organisation

Les mécanismes d'auto-organisation sont classés en deux catégories :

Les mécanismes d'auto-organisation logique qui modifient, placent et multiplient la répartition des connaissances (ex., les compétences, les liens d'interaction) [MAL, 99].

Les mécanismes d'auto-organisation physique qui placent et multiplient la répartition géographique des robots mobiles [MAL, 99].

D'une manière générale, le placement permet d'augmenter les temps de réponse et les coûts de communication en rapprochant l'information ou les robots mobiles vers les demandeurs (en limitant le nombre d'intermédiaires) ; la duplication permet quant à elle d'augmenter le parallélisme et la robustesse du système, mais nécessite en contrepartie des coûts de coordination et de maintien de la cohérence plus importants. Dans la plupart des cas, un système ne pourra pas garantir toutes ces propriétés à la fois, mais pourra, par auto-organisation, adopter un comportement qui sera contextuellement le « meilleur » (par rapport aux critères d'évaluation choisis) [MAL, 99].

I.6.2 Coopération

L'intérêt et la pertinence des systèmes coopératifs sont évidents. Il est donc plus facile de résoudre un problème unique et complexe en le divisant en plusieurs sous problèmes élémentaires. Il existe également de nombreux autres aspects positifs : la répartition de tâches sur de nombreux robots permet de réduire les coûts de conception, de réalisation et même de maintenance. De même, la notion de répartition permet d'accroître la robustesse et la fiabilité des systèmes par la redondance et le parallélisme des processus engagés. Tous ces atouts sont à l'origine de l'attrait et de l'étude de la coopération des robots mobiles.

I.6.2.1 Méthodes de Coopération

I.6.2.1.1 Regroupement et multiplication

Le regroupement des robots mobiles est une forme de coopération (elle assure aux animaux une meilleure sécurité, plus de chaleur et de partage de nourriture). L'agrégation de nombreux robots simplifie la navigation (volée d'oiseaux ou bancs de poissons). Un robot mobile ou un petit groupe de robots mobiles décide de la direction à prendre et tous les autres n'ont qu'à le suivre [SIM, 01].

Le regroupement (comme montré sur la **Figure I.5)** consiste tout simplement pour un groupe de robots mobiles à se rapprocher physiquement, donc construire un bloc ou un réseau de communication entre robots.

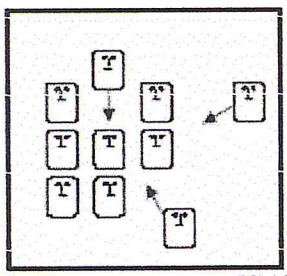


Figure I.5 : Regroupement [SAN, 03].

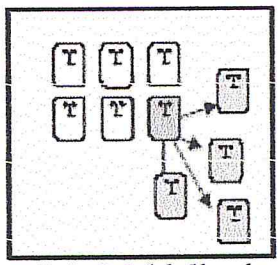


Figure I.6 : Multiplication [SAN, 03].

La multiplication des robots mobiles (la redondance des compétences), assure une grande fiabilité et robustesse au groupe. En contrepartie, tout regroupement peut dans certains cas critiques créer des situations d'encombrement et de conflits [SIM, 01].

La multiplication est l'augmentation quantitative des robots dans un système donné et présente des avantages considérables, tels que la performance et fiabilité (voir **Figure I.6)**.

I.6.2.1.2 Spécialisation

C'est un processus qui mène les robots mobiles graduellement à la spécialisation dans certaines de leurs tâches. Elle permet aux robots mobiles une adaptation dynamique aux conditions et aux besoins de son environnement (généralement dans les systèmes réactifs) [SIM, 01].

La spécialisation permet à des robots mobiles de devenir de plus en plus adaptés à leur tâche. Mais, il est difficile pour un robot mobile d'être spécialisé dans toutes les tâches. La **Figure I.7** est un exemple de cinq robots, tel que, chaque robot effectue une tâche précise (a, b ou c).

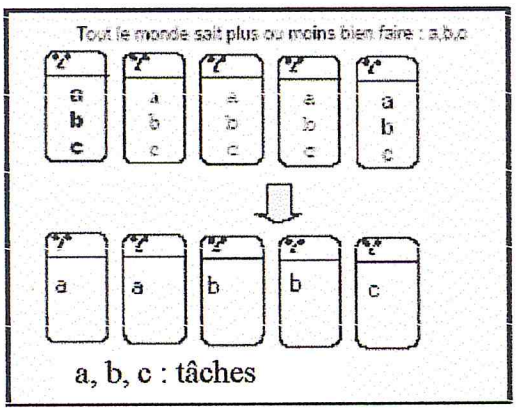


Figure I.7 : Spécialisation [SAN, 03].

I.6.2.1.3 Arbitrage et négociation

Ce sont deux moyens de contrôler les conflits entre les robots (voir **Figure I.8** ci-dessous). L'arbitrage établit des règles sur le comportement des robots qui ont comme conséquences de limiter les conflits [SIM, 01].

La négociation permet aux robots mobiles de résoudre des conflits. Ceux-ci s'engagent dans une vraie "discussion", exigeant un niveau élevé des systèmes de communication, afin d'arriver à une solution ou à un compromis [SIM, 01].

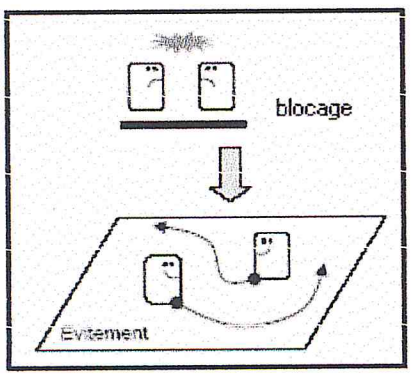


Figure I.8 : Arbitrage et négociation [SAN, 03].

I.6.2.1.4 Répartition des tâches, des informations et des ressources

Il s'agit des processus collaboratifs permettant aux robots de distribuer les tâches, les informations et les ressources afin d'effectuer un objectif commun (voir **Figure I.9**). Cette distribution peut être faite avec des systèmes délibératifs par des mécanismes d'approvisionnement. Dans les systèmes réactifs, cette répartition se fait par les moyens de l'environnement, et mène à la spécialisation des robots et à leur répartition géographique [SIM, 01].

La collaboration correspond à un ensemble de processus collaboratifs qui permettent aux robots mobiles de répartir des tâches, des informations et des ressources dans le but de réaliser un objectif commun [SIM, 01].

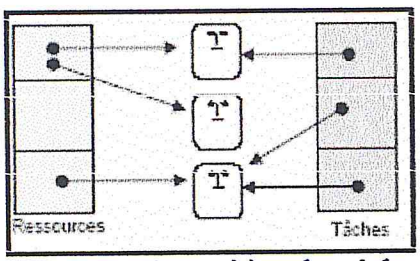


Figure I.9 : Répartition des tâches et des ressources [SAN, 03].

I.6.2.1.5 Coordination d'actions

d'après J.Ferber : "Dans le cadre de la coopération, elle peut être définie comme l'articulation des actions individuelles accomplies par chacun des robots de manière à ce que l'ensemble aboutisse à un tout cohérent et performant" [SIM, 01].

Il s'agit de coordonner ses actions avec les autres agents. Ce sont toutes les tâches ou actions qui ne sont pas directement productives mais qui assurent l'accomplissement de celles qui le sont. On distingue quatre formes de coordination d'actions [SIM, 01] :

- Coordination par **synchronisation :** recherche d'une succession cohérente d'actions non parallélisables. Elle est très présente dans les systèmes automatiques industriels et les systèmes d'exploitation répartis.

I.7 Conclusion

Un robot mobile ne saurait être autonome s'il est incapable de planifier ses mouvements, et réaliser une tâche spécifiée. Mais dans certaines situations (tâches impossibles), ce robot mobile seul est incapable de réaliser sa tâche, quelque soit son degré d'autonomie. Il faut alors penser à ajouter d'autres robots mobiles. Ces robots mobiles doivent effectuer cette tâche de façon collective et coopérative. En effet, la collectivité permet aux ensembles des robots mobiles de réaliser une tâche commune le plus rapidement possible. Mais la coopération permet aux robots mobiles d'effectuer une tâche commune où chacun a sa tâche, par coordination ou spécialisation ou regroupement ou communication, etc. Finalement la coopération entre les robots mobiles, permet de résoudre des problèmes qu'un seul robot ne pourrait résoudre.

Dans le chapitre suivant, on va présenter un type d'apprentissage qui est très utilisé dans la robotique mobile : c'est l'apprentissage par renforcement. Il est basé sur trois fonctions principales qui sont la fonction d'évaluation, la fonction de mise à jour et la fonction de renforcement.

II.1 Introduction

L'apprentissage consiste à acquérir ou à modifier une représentation de l'environnement. Ce processus permet à un animal d'utiliser son expérience passée pour assimiler l'organisation de son environnement et les conséquences de ses propres actions, et pour s'y accommoder. Il contribue donc à l'autorégulation et à l'adaptation des comportements.

Les différentes méthodes d'apprentissage automatique peuvent être classées selon le type de ***professeur*** qu'elles considèrent.

Deux principales classes se distinguent :

L'apprentissage supervisé

L'enseignant (superviseur) fournit des *exemples* au système, il dit la bonne action à prendre dans une situation *s* [ZUC, 00].

Exemple: dans la situation s_i il faut prendre l'action a_j.

L'apprentissage non-supervisé

Il n'y a pas d'enseignant ou de critique pour contrôler le processus d'apprentissage, et pas d'exemples spécifiques de ce qui doit être appris [ZUC, 00].

Le troisième type d'apprentissage, intermédiaire :

L'apprentissage par renforcement (AR)

L'apprentissage par renforcement consiste à rechercher l'action optimale à prendre dans chacun des états du système, en exploitant pour cela un modèle de ce système.

L'apprentissage par renforcement peut être utilisé quand l'environnement retourne, en réponse aux actions d'un système, un signal qui permet au système de juger si ses actions sont appropriées ou non [COU, 02].

CHAPITRE

II

II Apprentissage par renforcement

- Coordination par **planification :** nécessite soit un agent (robot mobile) coordinateur qui planifie les actions de chaque agent, soit l'échange et la construction des plans partiels dans le but de converger vers un plan global.
- Coordination **réactive :** aucune planification, ce sont des robots réactifs qui s'auto-organisent à travers leur interaction avec l'environnement.
- Coordination par **réglementation :** règle de comportement visant à éviter les conflits.

I.6.2.1.6 Communication

Le robot doit pouvoir échanger des informations plus ou moins complexes avec d'autres robots.

La première méthode permettant de faire communiquer des robots mobiles consiste à centraliser tous les échanges de messages. Ce type de protocole existe toujours dans divers système multi-robots. Il y a un robot superviseur qui centralise les décisions et les communications. Il existe aussi des systèmes distribués utilisant un système de communication centralisé. Mais ce type d'approche supervisée est inadapté aux systèmes distribués pour deux raisons principales. Afin de conserver les propriétés de robustesse et d'autonomie de ces systèmes, il est préférable de ne jamais centraliser des informations essentielles sur un système unique. De plus, il faut constamment garantir que chaque robot puisse rester en contact avec la station ou le robot superviseur, ce qui n'est pas toujours possible. La deuxième méthode consiste à décentraliser les échanges des messages, il n'y a pas un robot superviseur qui centralise les décisions et les communications [SIM, 01].

I.6.2.2 Intelligence collective

Inspirée de l'observation du comportement des fourmis ou des abeilles, cette discipline postule qu'il est possible d'arriver à des comportements globaux intelligents en utilisant des comportements individuels simples et mécaniques. Bien qu'il puisse sembler paradoxal à certains de pouvoir obtenir de pareils résultats sans une organisation centrale, de nombreux systèmes composés d'une multitude d'éléments très simples, ne communiquant que localement, peuvent s'auto-organiser pour résoudre des problèmes complexes et difficiles [MON, 00].

Cette méthode d'apprentissage est de plus en plus populaire et paraît fortement adaptée au problème de l'apprentissage d'un comportement par un robot mobile.

Notation du chapitre

S : L'ensemble des états.

A : L'ensemble des actions.

s : Un état.

a : Une action.

r : Le signal de renforcement.

E_π: L'espérance en suivant la politique π.

$0 \leq \gamma \leq 1$: Le taux de diminution des renforcements.

$0 \leq \alpha \leq 1$: Le taux d'apprentissage.

II.2 Principe de l'apprentissage par renforcement

Le cadre applicatif de l'apprentissage par renforcement se compose de trois éléments. Il faut un environnement sur lequel on puisse agir par le biais d'actions et qui fournit, en réponse à ces actions, un signal de renforcement et des informations sur l'état de l'environnement. L'agent apprenant utilise alors une politique d'action qui, en fonction de l'état de l'environnement, produit les actions à exécuter. Enfin, le processus d'apprentissage modifie cette politique d'action en fonction du signal de renforcement [BUF, 99].

L'apprentissage par renforcement a pour objectif de maximiser la performance du système en renforçant les meilleures actions. L'agent obtient des informations de l'état de l'environnement (nous les appellerons les perceptions) et agit également sur l'environnement puis reçoit une estimation de sa performance : la récompense. Dans les algorithmes d'apprentissage par renforcement, cette récompense peut être immédiate ou retardée. Le schéma général est représenté sur la **Figure II.1**. L'agent (robot) connaît son état (s_t) dans l'environnement, il a la possibilité d'agir sur ce dernier (a_t) et il reçoit une récompense (r_t) [LUC, 03].

L'objectif de l'apprentissage par renforcement est d'associer à chaque état du système une action qui permet de maximiser la récompense.

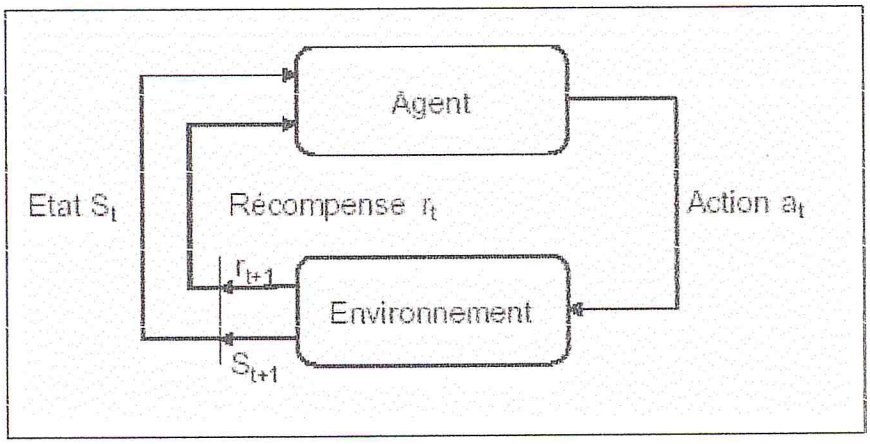


Figure II.1: Système d'apprentissage par renforcement.

II.3 Caractéristique clefs de l'apprentissage par renforcement

- L'**agent** (robot) ou le **système** apprend lui même l'action nécessaire pour arriver à la cible.
- Recherche de meilleures actions basée sur **Essai-Erreur**.
- Possibilité d'une **récompense retardée**, sacrifier des **gains à court terme** pour des **gains plus grands à long terme**. La **récompense r** ressemble à notre joie (si r est grande) et notre douleur (si elle est petite) dans l'immédiat tandis que les valeurs de la fonction de valeurs V correspondent à un jugement plus raffiné et une vue à long terme de comment heureux ou malheureux nous sommes quand notre environnement sera dans un état particulier [SUT, 98].
- Besoin d'**explorer** et d'**exploiter** :

Défis : compromis d'**exploration/exploitation** pour augmenter la récompense, l'agent (robot) doit préférer des actions qui ont été essayées dans le passé. Mais pour découvrir de telles actions il doit essayer des actions qui n'ont pas encore étaient sélectionnées [LIT, 96]. L'agent doit **exploiter** ce qui est déjà connu pour obtenir une récompense, mais il doit aussi **explorer** pour sélectionner les meilleures actions dans le futur.

Dilemme : ni exploitation ni exploration ne peut être poursuivie exclusivement sans échouer dans la tâche. L'agent (robot) doit essayer une variété d'actions et favoriser progressivement celles qui apparaissent les meilleures.

Sur le plan stochastique, chaque action doit être essayée plusieurs fois pour estimer efficacement ses récompenses espérées [SUT, 98].

- Considère le problème global d'un agent (robot) qui agit avec un environnement incertain. Ceci est en contraste avec beaucoup d'approches (apprentissage supervisé) qui s'adressent à des sous-problèmes sans s'adresser à la question comment ils pourraient correspondre à un problème plus large [SUT, 98].

II.4 Eléments de l'apprentissage par renforcement

II.4.1 Politique

En Psychologie, une politique est un ensemble de règles ou associations stimulus-réponse. Mais en intelligence artificielle, la politique correspond au comportement de l'agent à un instant donné. C'est une **association** (mapping), à partir des états de l'environnement mesurés par l'agent aux actions à prendre quand l'environnement se trouve dans ces états. Autrement dit, c'est une fonction simple « lookup table », ou une fonction complexe nécessitant des calculs intensifs tel qu'un processus de recherche [SUT, 98].

II.4.2 Renforcement

C'est une fonction qui définit une association, en général stochastique, à partir des états de l'environnement mesurés par l'agent ou à partir des couples état-action [SUT, 98].

II.4.2.1 Récompense (r)

La récompense est une valeur immédiate qui définit les caractéristiques du problème vues par l'agent. Tel qu'elle est, la récompense doit être nécessairement fixée. Elle peut, cependant, être utilisée comme une base pour modifier la politique [SUT, 98].

Exemple : si une action sélectionnée par une politique est suivie par une récompense faible alors cette politique peut être modifiée pour sélectionner une autre politique quand l'agent se trouve dans la même situation dans le futur.

II.4.2.2 Pénalité (-r)

C'est une valeur négative immédiate qui définit les caractéristiques du problème vues par l'agent.

II.4.3 Fonction de valeur (V)

Tandis qu'une fonction r indique ce qui est bien dans l'immédiat, une fonction valeur spécifie ce qui est bien dans le futur.

Une valeur V de l'état est la quantité totale des renforcements accumulés par l'agent dans le futur en partant de cet état. Malheureusement, il est plus difficile de déterminer V que de déterminer r. le renforcement r est, en principe, donné directement par l'environnement tandis que V doit être **estimée** et **ré-estimée** à partir des séquences d'observations (séquences d'états de l'environnement) que l'agent fait sur la totalité de sa vie (ou totalité du processus) [SUT, 98].

II.4.4 Modèle de l'environnement

Ce modèle est l'imitation du comportement de l'environnement. Par exemple, étant donné un couple état-action, un modèle permet de prédire le couple état-action futur [SUT, 98], [CON, 02]. Il sert principalement dans les tâches de planification, c'est-à-dire qu'il offre

le moyen de décider de la prochaine action à entreprendre en fonction des situations possibles à venir sans avoir besoin de les tester effectivement.

Tous les agents (robots) n'utilisent pas la notion de modèle de l'environnement dans le cadre d'un problème de type AR. Les méthodes qui ne font jamais appel au modèle sont dites *modèle indépendant* (model-free) [CON, 02]. Elles sont, en général, très simples et, de façon surprenante, sont quand même capables de découvrir un comportement optimal.

II.5 Méthodes de résolution

Il existe trois classes fondamentales de méthodes pour résoudre le problème d'apprentissage par renforcement et qui sont:

Méthodes de Programmation Dynamique (PD)

Bien développées mathématiquement, mais nécessitent un modèle complet et précis de l'environnement.

Méthodes de Monte Carlo (MC)

Ne nécessitent pas de modèle et sont conceptuellement très simples.

Méthodes de Différence Temporelle (DT)

Ne nécessitent pas de modèle et sont complètement incrémentales, mais sont plus complexes à analyser.

Toutes ces méthodes permettent de résoudre le problème de l'apprentissage par renforcement incluant les récompenses retardées.

Ces méthodes se différencient par rapport à leurs efficacités et la vitesse de convergence.

II.5.1 Programmation Dynamique (PD)

Le terme « Programmation Dynamique » se rapporte à une collection d'algorithmes qui peuvent être utilisés pour calculer des politiques optimales étant donné un modèle parfait de l'environnement tel qu'un Processus de Décision de Markov (Markov Decision Process (MDP)) [BUF, 02].

Processus de Décision de Markov

Nous désignons l'information reçue par le robot à partir de l'environnement. L'état peut inclure des "sensations" immédiates comme les mesures sensorielles, des "sensations" de haut niveau, et des structures construites dans le temps à partir des séquences de sensations, par exemple : nous regardons un objet, puis on tourne notre regard ailleurs, et nous pensons dans notre esprit que cet objet est toujours là. Idéalement, un état devrait résumer les sensations dans le passé pour retenir toute l'information "essentielle", i.e., il devrait avoir la **Propriété de Markov** qu'est énoncée comme suit [SUT, 98] :

$$\Pr\{s_{t+1}=s', r_{t+1}=r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1} \ldots r_1, s_0, a_0\} = \Pr\{s_{t+1}=s', r_{t+1}=r \mid s_t, a_t\} \quad \textbf{(II.1)}$$

(et pour tout s', r et les historiques $s_t, a_t, r_t, s_{t-1}, a_{t-1} \ldots r_1, s_0, a_0$). Où Pr : est une probabilité.

Si une tâche de l'AR a la propriété de Markov, elle est alors essentiellement un Processus de Décision de Markov (PDM). Si les ensembles d'états et des actions sont finis, alors le processus est fini. Et pour définir un MDP fini, nous devons donner:

- Une dynamique du système à un pas définie par les probabilités de transition :

  $\mathbf{P}^{a}_{ss'} = \Pr\{s_{t+1}=s' \mid s_t=s, a_t=a\} \ \forall s, s' \in S, a \in A.$

- La récompense immédiate espérée :

  $\mathbf{R}^{a}_{ss'} = E\{r_{t+1} \mid s_t=s, a_t=a, s_{t+1}=s'\} \ \forall s, s' \in S, a \in A.$

Les algorithmes classiques de PD ont une utilité limitée dans l'AR à la fois à cause de :

1. leur hypothèse d'un modèle parfait.
2. leur coût élevé de calcul.

L'idée principale de PD et de AR en général est de calculer une **fonction valeur** pour organiser et structurer la recherche pour des politiques optimales. Une politique optimale peut être trouvée si nous calculons des valeurs optimales V* ou Q* dans l'équation de Bellman [SUT, 98].

II.5.1.1 Evaluation de la politique [CON, 02]

Pour une politique donnée π, calculer la fonction de la valeur d'état V^{π}.

Appelons :

fonction valeur d'état pour une politique donnée π :

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\} \qquad \textbf{(II.2)}$$

L'équation de Bellman pour V^{π} est donnée par :

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'}\left[R^{a}_{ss'} + \gamma V^{\pi}(s')\right] \quad \text{, pour tout s} \in \text{S} \qquad \textbf{(II.3)}$$

Où :

π(s, a) : La probabilité que l'action a soit choisie dans l'état s par la politique π.

s' : L'état futur.

L'équation **(II.2)** effectue tout simplement la moyenne sur toutes les possibilités, en pondérant chaque possibilité par la probabilité d'occurrence. Elle signifie que la valeur de l'état de départ s doit être égale à la valeur réduite espérée de l'état futur s' [SUT, 98].

II.5.1.2 Amélioration de la politique

Supposons que nous avons calculé V^{π} pour une politique déterministe π. Pour un état donné *s*, l'utilité de l'action *a* dans l'état *s* est [SUT, 98] :

$$Q^{\pi}(s,a) = E_{\pi}\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \middle| s_t = s, a_t = a\right\} = \sum_{s'} P^{a}_{ss'}\left[R^{a}_{ss'} + \gamma V^{\pi}(s')\right] \quad \textbf{(II.4)}$$

Il est préférable de choisir l'action *a* dans l'état *s* si : $Q^{\pi}(s, a) > V^{\pi}(s)$ [SUT, 98].

II.5.1.3 Algorithme d'itération de politique [CON, 02]

Initialisation arbitraire de π

Faire

Calcul de la fonction valeur avec π

$$V_{\pi}(s) = \max_{a} \sum_{s'} P^{a}_{ss'}\left[R^{a}_{ss'} + \gamma V^{\pi}(s')\right]$$

Amélioration de la politique à chaque état

$$\pi'(s) = \arg\max_{a} \sum_{s'} P^{a}_{ss'}\left[R^{a}_{ss'} + \gamma V^{\pi}(s')\right]$$

$\pi' = \pi$

Jusqu'à ce qu'aucune amélioration ne soit possible

Cet algorithme garantit la convergence vers une politique optimale.

II.5.2 Méthode de Monte Carlo (MC)

Les méthodes de PD supposent que l'on connaisse l'environnement à travers des probabilités de transition d'état $\mathbf{P}^{a}_{ss'}$ et de renforcement $\mathbf{R}^{a}_{ss'}$ [MIC, 02]. Le principe des méthodes MC est d'estimer directement la fonction valeur (valeur d'utilité V(s)) à partir d'expériences. Les expériences pouvant être réelles, on parlera d'un apprentissage ``on-line", ou bien simulé - bien sûr dans ce cas on aura besoin d'un modèle de l'environnement. Les approches MC sont alors des méthodes basées sur une moyenne de valeurs de retour obtenues par expériences. On ne s'intéresse ici qu'à des situations où les expériences se terminent (on est donc certain de récupérer une évaluation en retour) [COR, 02].

Le terme ``Monte Carlo" est habituellement utilisé pour désigner n'importe quelle méthode d'estimation. Ce terme distingue toute méthode basée sur une moyenne des retours d'une expérience complète (par opposition aux méthodes utilisant des retours partiels) [COR, 02]. Dans les méthodes MC, après un épisode (expérience), on met à jour les $V_{\pi}(s)$ en fonction de la récompense obtenue.

Algorithme de Monte-Carlo [BUF, 02] :

Entrée : $\pi \leftarrow$ politique à évaluer
Entrée : V $\leftarrow$ fonction d'utilité arbitraire
Entrée : Retour(s) $\leftarrow$ listes vides
tant que vrai **faire**
Générer un épisode en suivant π
Pour tout *s* parcouru pendant l'épisode **faire**
r $\leftarrow$ récompense suivant la première occurrence de *s*
Ajouter *r* en queue de Retour(*s*)
V(*s*) $\leftarrow$ moyenne (Retour(*s*))
fin pour
fin tant que

II.5.3 Différence Temporelle (DT)

Richard Sutton et Andrew Barto [SUT, 98] ont proposé une classe d'algorithmes incrémentaux pour l'apprentissage par renforcement. Ces algorithmes, qu'ils appellent DT (Différence Temporelle), permettent de corriger des prédictions passées en fonction de prédictions futures. Il s'agit de méthodes combinant les idées de l'approche MC et de l'approche PD. Comme dans les méthodes MC, les méthodes DT peuvent apprendre directement à partir des expériences brutes sans notion de modèle de l'environnement dynamique. Comme pour les méthodes à base de PD, elles mettent à jour leurs estimations à partir d'autres estimations apprises, sans avoir à attendre la fin de l'expérience en cours. Les relations entre ces trois approches sont l'un des thèmes récurrents de l'apprentissage par renforcement [COR, 02].

Le principe général des méthodes DT est pour une politique π donnée :

V^{π} : espérance de gain depuis l'état courant.

Q^{π} : espérance de gain depuis l'état courant en précisant l'action choisie.

1. évaluer/prédire V^{π} ou Q^{π} pour une politique π donnée.
2. mettre à jour la politique en fonction de ces grandeurs.

Nous allons voir quelques algorithmes de cette classe.

II.5.3.1 Q-learning (TD+contrôle sans politique)

L'algorithme Q-Learning proposé par Watkins en 1989, est certainement l'algorithme d'apprentissage par renforcement le plus utilisé [TOU, 99]. Trois fonctions principales participent au Q-learning (voir **Figure II.2**), une fonction d'évaluation, une fonction de renforcement et une fonction de mise à jour.

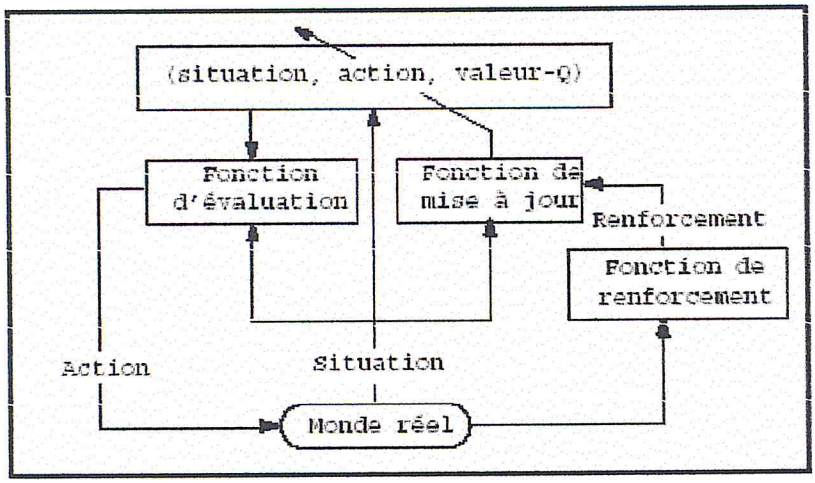


Figure II.2 : Modèle général pour les algorithmes d'apprentissage par renforcement, ici le Q-learning [TOU, 99].

II.5.3.1.1 Fonction d'évaluation

Les valeurs Q sont stockées dans une table à deux dimensions : situation et action. La fonction d'évaluation parcourt, pour la situation présente, les valeurs de Q associées aux actions et sélectionne l'action de plus grande utilité [TOU, 99].

II.5.3.1.2 Fonction de renforcement

Elle Correspond à chaque état (ou couple état-action) de l'environement un scalaire, la récompense, indiquant la valeur intrinsèque de l'adéquation de l'état. Le seul objectif d'un agent dans un problème de renforcement est de maximiser la somme des récompenses reçues. La fonction de renforcement doit être fixée à priori pour aider l'agent à remplir sa tâche. Cependant, elle peut servir de support à l'agent pour changer de politique [COR, 02].

II.5.3.1.3 Fonction de mise à jour

L'algorithme Q-Learning est basé sur l'estimation de la fonction $Q(s_t, a_t)$ définie par:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma.max\,[Q(s_{t+1}, a)] - Q(s_t, a_t)] \qquad \text{(II.5)}$$

$0<\alpha, \gamma<1$

Algorithme Q-learning [BUF, 02] :

Initialiser $Q(s, a)$ arbitrairement
pour tout épisode **faire**
Initialiser s
pour tout pas de l'épisode jusqu'à s terminal **faire**
Choisir a pour s en utilisant la politique dérivée de Q
Effectuer l'action a ; observer r et s'
$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} [Q(s', a')] - Q(s, a)]$
$s \leftarrow s'$
fin pour
fin pour

II.5.3.1.4 Choix des actions

Il y a deux stratégies possibles pour choisir l'action a_t [JOD, 03] :

1. **e-greedy** (où $e \in [0,1]$) dont les étapes essentielles sont les suivantes :

(a) tirer uniformément un nombre p au hasard dans l'intervalle $[0,1]$;
(b) si $p < e$, tirer uniformément une action a_t au hasard dans A (exploration) ;
(c) si $p >= e$, choisir l'action $a_t = \text{argmax } Q(s_t, a)$, $a \in A$ (exploitation).

2. **Q-Learning boltzmannien**, choisir a_t selon une distribution de Boltzmann :

$$P(a_t) = e^{Q(s, a_t)/T} \cdot \left[\sum_{a \in A} e^{Q(s, a)/T} \right]^{-1} \qquad \textbf{(II.6)}$$

Où T est la température de la distribution : quand T est élevée, la distribution est presque uniforme ; quand $T \rightarrow 0$, on se rapproche du 0-greedy. En pratique, on fait décroître la température, ce qui permet de moduler exploration et exploitation sans distinguer explicitement ces deux phases.

II.5.3.2 Méthode TD (λ)

Cette méthode est caractérisée par : Au lieu de réduire les différences entre les évaluations de deux états adjacents, on réduit les différences entre évaluations distantes dans le temps de *n* pas. Elle a un spectre de solutions entre TD (0) et Monte-Carlo (TD (1)), tel que λ varie entre 0 et 1. On prend en compte les estimations jusqu'à *n* pas dans le futur, et on modifie les estimations jusqu'à *n* pas dans le passé en tenant compte d'une *trace d'éligibilité* (mémoire à déclin des états visités) [SUT, 98].

Une méthode particulière de TD (λ) est présentée, c'est la méthode TD (0). Cette méthode effectue la mise à jour des valeurs d'utilité en ne regardant qu'un seul pas en avant. Au cours de tout épisode (dans chaque itération), on met à jour les V^{π} (s) en fonction de la récompense obtenue [BUF, 02].

Algorithme TD (0) [BUF, 02] :

Entrée : $\pi \leftarrow$ politique à évaluer
Entrée : V ← fonction d'utilité arbitraire
pour tout épisode **faire**
Initialiser *s*
pour tout pas de l'épisode jusqu'à *s* terminal **faire**
$a \leftarrow$ action donnée par π pour *s*
Choisir action *a* ; observer la récompense *r*, et l'état suivant *s'*
$V(s) \leftarrow V(s) + \alpha[r + \gamma.V(s') - V(s)]$
$s \leftarrow s'$
fin pour
fin pour

II.5.3.3 Sarsa (TD et contrôle avec politique)

Le principe de l'algorithme **Sarsa** est : à chaque choix d'action dans l'état courant s_t, l'agent (robot) suit approximativement la politique courante π. Après observation du nouvel état courant s_{t+1}, et du renforcement reçu r_{t+1}, il met à jour la valeur d'utilité de la situation rencontrée et est alors prêt à choisir l'action suivante [BUF, 02].

Sarsa est une méthode *sur politique* car l'agent (robot) s'inspire à chaque itération de la politique courante π pour le choix des ses actions. Elle estime la fonction d'utilité $Q^{\pi}(s, a)$ pour la politique courante et pour toutes les paires (état-action), et conduit alors la mise à jour suivante :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Cette mise à jour est effectuée après chaque transition partant d'un état s_t non terminal. Si s_{t+1} est un état terminal, alors $Q(s_{t+1}, a_{t+1})$ est défini comme égal à zéro [BUF, 02]. De cette formule, on déduit facilement un algorithme qui estime Q^{π} continuellement, faisant donc évoluer π au fur et à mesure.

Algorithme Sarsa [BUF, 02] :

Initialiser $Q(s, a)$ arbitrairement
pour tout épisode **faire**
    Initialiser s
    Choisir a pour s en utilisant la politique dérivée de Q
    pour tout pas de l'épisode jusqu'à s terminal **faire**
    Choisir a' pour s' en utilisant la politique dérivée de Q
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
    $s \leftarrow s' ; a \leftarrow a'$
    fin pour
fin pour

II.6 Conclusion

L'apprentissage par renforcement concerne le choix de la meilleure action dans une situation courante. Généralement la structure de l'environnement est supposée inconnue, et l'agent (robot) doit apprendre à partir de son interaction avec le monde. En particulier, aucun professeur ne lui dit quelle action est la meilleure à prendre dans une situation donnée, et seul le signal de renforcement assez pauvre (un scalaire) comme information liée à ses décisions passées est retourné par l'environnement.

Nous avons vu que l'AR est basé sur la fonction valeur, et diverses méthodes à savoir PD, MC et DT ont été proposées pour apprendre cette fonction valeur. L'agent (robot) cherche donc à apprendre la valeur de chaque état ou chaque (état-action). Il sélectionne alors l'action ayant la valeur maximale. Cependant, si la fonction valeur estimée est exacte, cette méthode conduit à la politique optimale. Mais pour la plupart des problèmes du monde réel, il est impossible de représenter les fonctions valeur exactement.

Dans le chapitre suivant, nous allons voir des mécanismes permettant à un groupe de robots mobiles de former des figures géométrique (cercle, cercle rempli, polygone,...). Le groupe de robots doit se déplacer en formation pour réaliser une tâche précise (transporter de gros objets, l'exploration de surface spatiale ou terrestre, ...).

CHAPITRE

III

III Formation des figures géométriques

III.1 Introduction

Tels les humains qui vivent en société, les robots mobiles seront appelés à travailler en groupe pour effectuer toutes sortes de tâches. Par exemple, un groupe de robots peut être utile pour manipuler en équipe ou transporter de gros objets. D'autres applications intéressantes sont l'exploration spatiale, terrestre ou sous marine, la surveillance, le déminage, la recherche et le sauvetage ou encore la cartographie d'environnements [BEA, 04].

La majorité de ces applications demande un déplacement organisé pour assurer une couverture adéquate de l'environnement d'opération. De ce fait, le groupe de robots doit se déplacer en formation, c'est-à-dire que les robots doivent se déplacer selon des configurations géométriques variables de par leurs positions respectives dans le groupe. Pour y arriver, les problèmes à résoudre sont :

- Comment initialiser et établir une formation ?
- Comment maintenir les robots en formation ?
- Comment éviter des obstacles ?
- Est-ce qu'il est possible de réaliser n'importe quel type de formation ?

Dans ce chapitre, nous présentons les algorithmes de formation de figures géométriques afin d'obtenir des groupes de robots pouvant réaliser une tâche donnée.

III.2 Formation de figures géométriques

Des algorithmes, qui permettent à un groupe de robots mobiles autonomes de former des figures géométriques, sont présentés dans ce paragraphe.

III.2.1 Formation d'un cercle

Un algorithme pour la formation d'un cercle avec un diamètre donné D a été proposé par Suzuki et Yamashita [ALB, 96]. Chaque robot est équipé de capteurs pour la

localisation des autres robots (en distance et en angle par rapport à lui même) c'est-à-dire chaque robot sait où les autres robots se trouvent.

L'algorithme permet aux robots de former un cercle itérativement. A chaque itération, un robot R doit effectuer les trois étapes suivantes :

Étape 1. Déterminer le robot le plus loin R_1 et le robot le plus proche R_2.

Étape 2. Calculer la distance d de sa position actuelle au point milieu $\mathbf{P}_M$ entre R_1 et R_2.

Étape 3. Déplacer avec une distance a =
$$\begin{cases} Min\ \{d\text{-}r,\ v\} \text{ vers } \mathbf{P}_M & \text{si } (d\text{-}r) >= 0. \\ \text{Ou} & \\ Min\ \{r\text{-}d,\ v\} \text{ loin de } \mathbf{P}_M & \text{si } (d\text{-}r) < 0. \end{cases}$$

Où v est la distance maximum qu'un robot peut réaliser à la fois, r est le rayon désiré du cercle à former (voir **Figure III.1**) [ALB, 96].

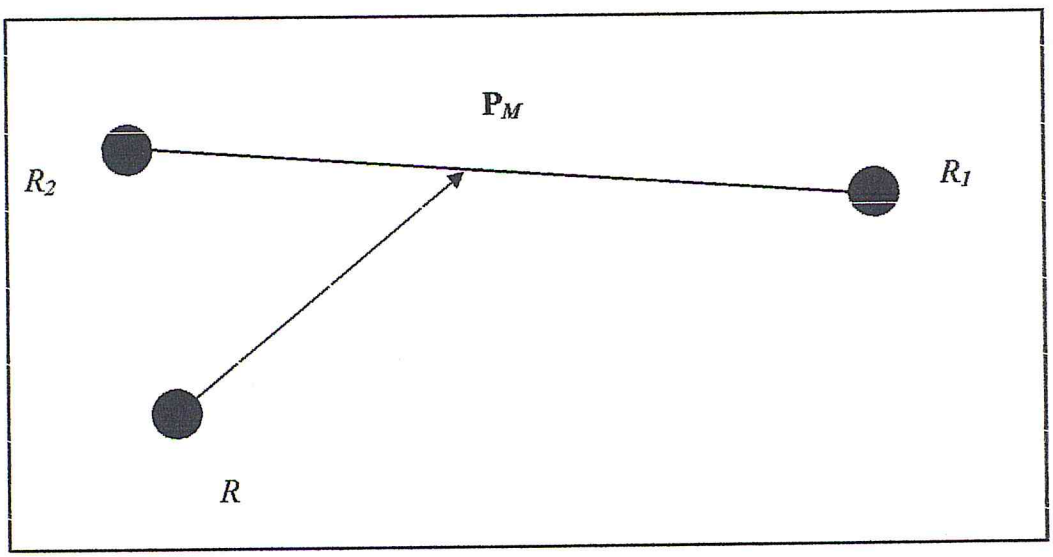


Figure III.1 : Représentation schématique de l'algorithme cercle.

L'inconvénient de cet algorithme est que le rayon du cercle final est toujours plus petit que le rayon désiré (20 pouces par rapport à 24 pouces d'après [ALB, 96]), parce que le point P_M ne correspond pas à l'origine du cercle.

Un autre algorithme de cercle a été proposé par Suzuki et Sugihara [SUZ, 95]. Cet algorithme donne une bonne approximation du cercle final.

Soit R un robot mobile, et soient R_1 (le robot le plus loin) et R_2 (le robot le plus proche). L'algorithme est itératif, à chaque itération, on détermine R_1 et R_2.

R se déplace comme suit :

- **Étape 1 :** Le robot R se déplace vers R_1, si $d > D$.
- **Étape 2 :** Le robot R s'éloigne *de* R_1, si $d < (D - \xi)$.
- **Étape 3 :** Le robot R s'éloigne *de* R_2, si $(D - \xi) \leq d \leq D$.

Où ξ est une petite constante positive, D est le diamètre du cercle à former, et d la distance entre R_1 et R_2..

Les deux étapes 1 et 2 assurent que la distance entre n'importe quel robot mobile et le robot le plus éloigné est D approximativement, et l'étape 3 du cercle a l'effet de distribuer les robots mobiles uniformément (voir **Figure III.2**) [SUZ, 95].

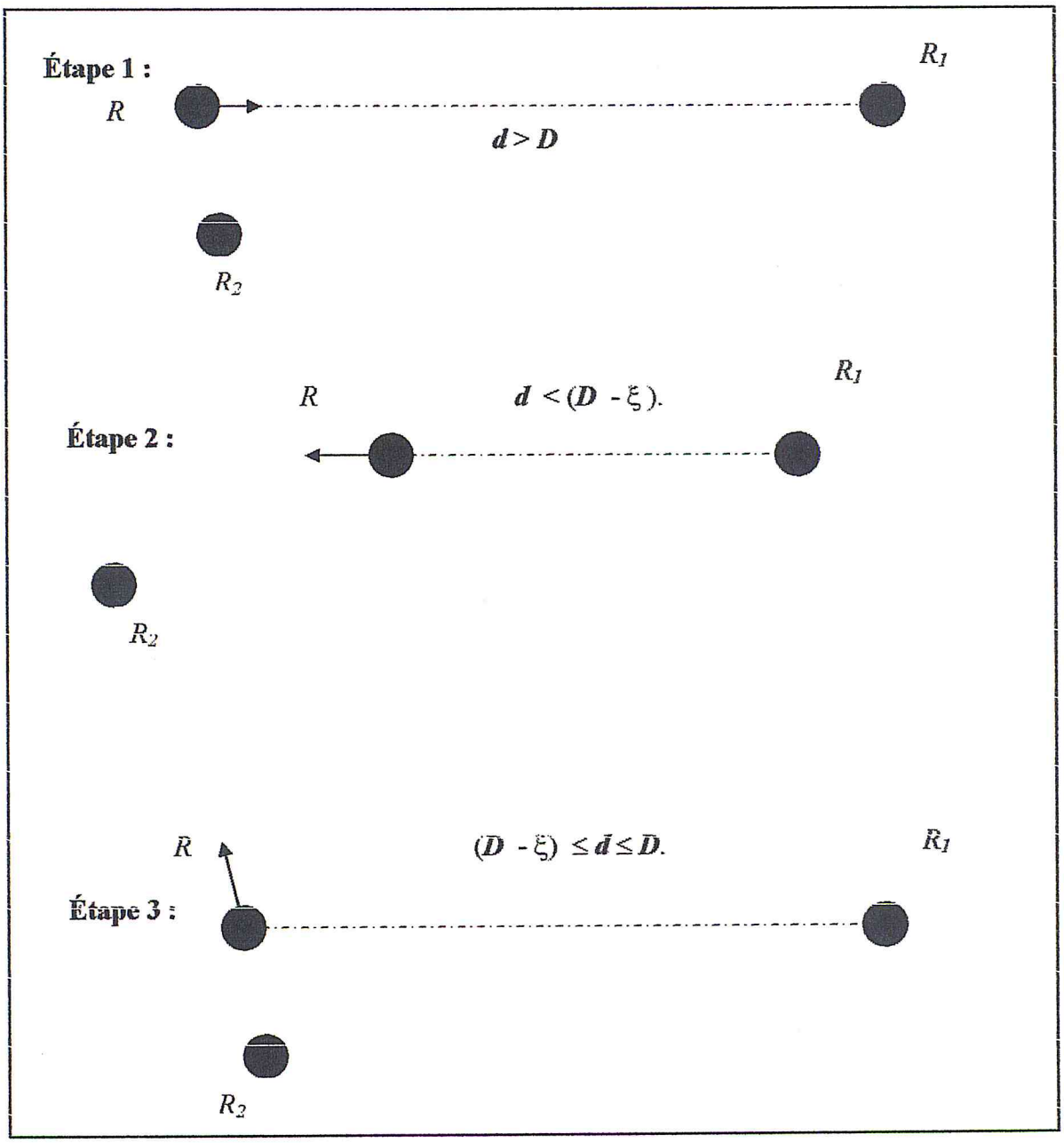


Figure III.2 : Différentes étapes de l'algorithme cercle proposé par Suzuki et Sugihara.

III.2.2 Formation d'un cercle rempli (Filicircle)

L'algorithme de cercle rempli consiste à distribuer Les robots de façon uniforme à l'intérieur d'un cercle.

L'algorithme de cercle rempli est itératif. A chaque itération, le robot R détermine le robot le plus éloigné R_1 et le robot le plus proche R_2, où d est la distance entre R et R_1, et D est le diamètre du cercle rempli à former [SUZ, 95].

- **Étape 1 :**

  Le robot R se déplace vers R_1, si $d > D$.

- **Étape 2 :**

  Le robot R s'éloigne de R_2, si $d <= D$.

La première étape assure que la distance entre deux robots est inférieure à D. La deuxième étape distribue les robots mobiles uniformément sur le lieu géométrique.

III.2.3 Formation d'un polygone simple

Premièrement, on laisse les robots exécuter l'algorithme de cercle jusqu'à la distribution approximative d'un cercle, pour que chaque robot puisse reconnaître son voisin de gauche et de droite immédiatement, notés l(R) et r(R) respectivement [SUZ, 95]. l(R) et r(R) sont déterminés une fois pour toute, chaque robot R considère le plus proche comme l(R), et r(R) est le plus proche robot dans la moitié du plan ne contenant pas l(R) (voir **Figure III.3**).

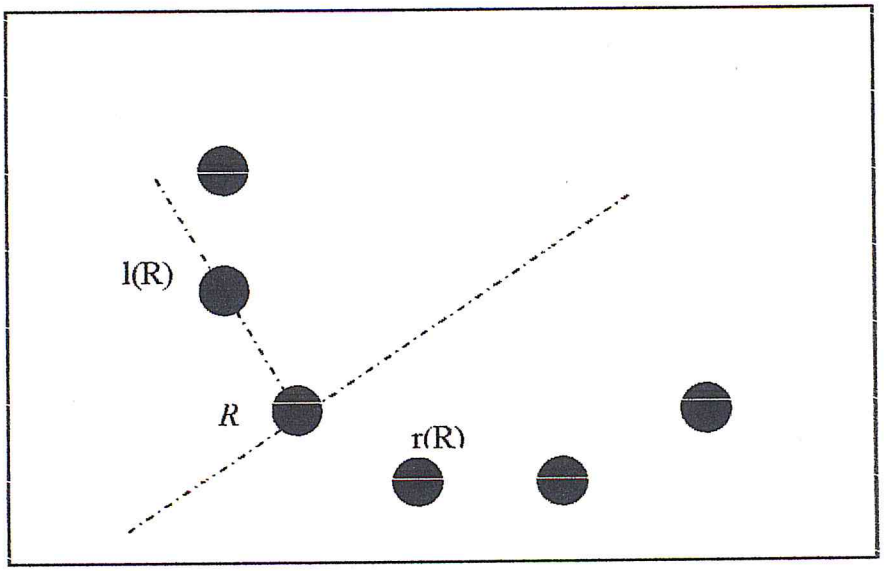


Figure III.3 : Choix de r(R) et l(R).

Quand l (R) et r(R) sont déterminés par chaque robot R, l'utilisateur sélectionne **n** robots qui seront les sommets du polygone à former. Ensuite, on exécute les deux étapes suivantes :

1. déplacer les robots manuellement vers leurs positions finales désirées (sommets du polygone).
2. permettre aux robots d'exécuter l'algorithme *contraction* défini comme suit :

Le robot *R*, connaissant à n'importe quel moment l(R) et r(R), se déplace vers le milieu du segment formé par l(R) et r(R) (voir **Figure III.4**).

Les robots qui sont les sommets du polygone n'exécutent pas l'algorithme *contraction*, et les autres robots sont distribués uniformément sur les **n** cotés du polygone [SUZ, 95].

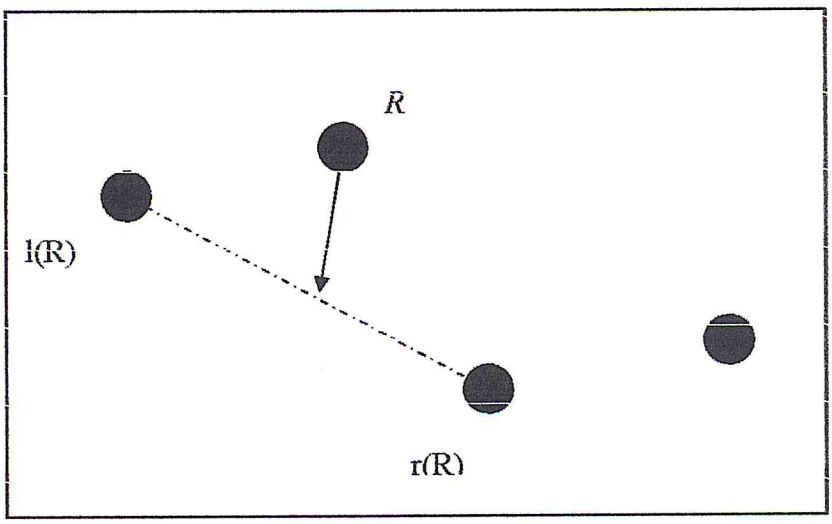


Figure III.4 : Algorithme *contraction*.

III.2.4 Formation d'un segment de ligne

La formation d'un segment de ligne peut être vue comme un cas particulier d'un polygone avec seulement deux sommets [SUZ, 95].

Premièrement, l'utilisateur sélectionne deux robots qui seront les deux extrémités de la ligne à former, ensuite on exécute les deux étapes suivantes :

1. Déplacer les deux robots manuellement vers leurs positions finales désirées (les deux extrémités de la ligne).
2. laisser les autres robots exécuter l'algorithme *polygone*.

Remarque : Les deux robots sélectionnés doivent être voisins dans le cercle formé.

Un autre algorithme de ligne a été proposé par Suzuki et Yamashita [ALB, 96]. Cet algorithme est itératif. A chaque itération un robot R fait ce qui suit :

- Étape 1. Déterminer le robot le plus loin R_1 et le robot le plus proche R_2.
- Étape 2. Calculer la distance d de sa position actuelle au point p qui est la projection perpendiculaire de lui-même à la ligne passant par R_1 et R_2.
- Étape 3. Déplacer avec une distance $a = min\ \{d, v\}$ vers le point p, où v est la distance maximum que le robot peut réaliser à la fois (voir **Figure III.5**).

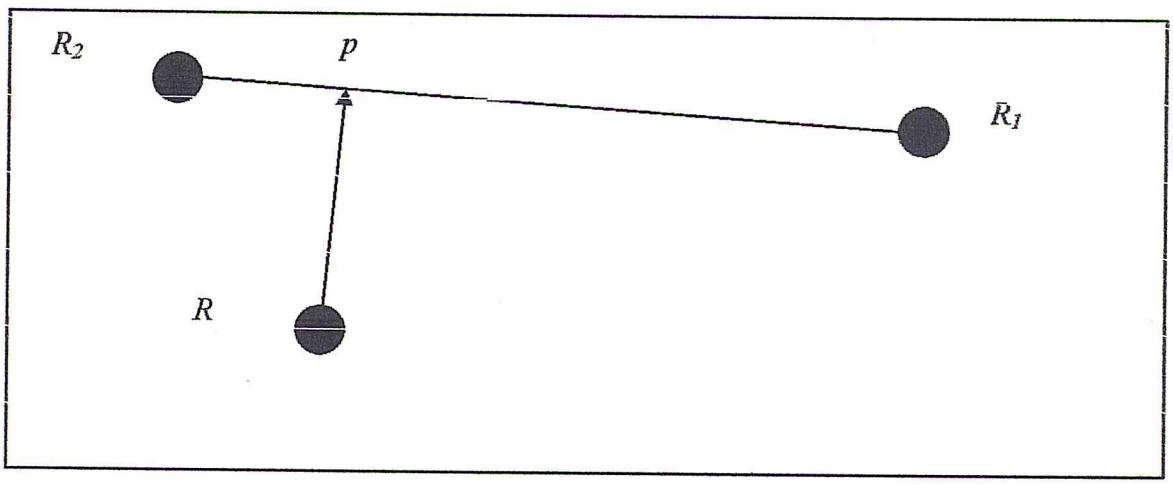


Figure III.5 : Représentation schématique de l'algorithme ligne proposé par Suzuki et Yamashita [ALB, 96].

III.3 Évitement de collision

Les algorithmes précédents ne prennent pas le problème de collision entre les robots en considération, une stratégie simple d'évitement de collision a été proposée par Suzuki et Sugihara, la stratégie est la suivante :

Si un robot détecte un autre robot tout près (20 centimètres par exemple) sur la direction de son chemin, alors il fait un écart vers la gauche, à condition qu'il trouve avec succès une direction qui soit nette pour n'importe quel robot. Si l'action tourner à gauche est impossible, le robot ne se déplace pas jusqu'à ce que son chemin devienne dégagé ou bien une dérivation à gauche devienne possible [SUZ, 95] (voir **Figure III.6**).

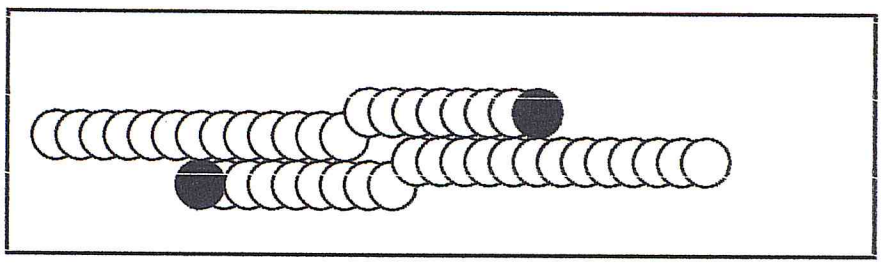


Figure III.6 : Exemple d'évitement de collision.

III.4 Conclusion

Dans les formations géométriques, l'intelligence des robots se retrouve dans leur capacité à éviter les obstacles et à établir une formation de manière autonome. Ceci correspond à l'objectif de la tâche à réaliser. De plus, la stratégie d'évitement de collision proposée par Suzuki et Sugihara, ne donne pas toujours le bon résultat, les robots peuvent entrer dans des situations imprévues, la solution proposée a été d'avoir recours à des techniques d'apprentissage. En effet, les robots vont devoir apprendre en premier à éviter les obstacles à s'éviter mutuellement avant de pouvoir former une figure géométrique et se déplacer par la suite en maintenant cette formation. Ceci nous incite à poser deux questions : comment contrôler le groupe de robots mobiles en formation, comment se fait l'apprentissage et l'adaptation du comportement d'un groupe de robots mobiles autonomes. Le chapitre suivant donne les réponses à ces questions.

CHAPITRE

IV

IV Approche de navigation collective basée sur l'Apprentissage par Renforcement d'un groupe de robots mobiles autonomes

IV.1 Introduction

La programmation des robots mobiles peut être un processus très long. Les algorithmes de l'apprentissage par renforcement sont considérés comme une très intéressante solution pour les systèmes non-linéaires ou les systèmes pour lesquels il est difficile de faire un modèle mathématique. Un robot mobile appartient aux systèmes de cette sorte.

D'autre part, lorsque des robots évoluent en milieu hostile, il n'est pas toujours possible de les téléopérer, et les méthodes traditionnelles de planification sont souvent gourmandes en temps de calcul. Leurs facultés d'apprentissage doivent donc être de plus en plus importantes. L'objectif est de faire apprendre à des robots une tâche collective sans l'intervention d'un opérateur (la robotique autonome). Donc, l'apprentissage peut être utilisé quasiment à tous les niveaux de conception du robot [FUJ, 98]. Particulièrement, il est presque toujours possible, au lieu de programmer une fonction, de la faire apprendre au robot. De même, quand il est en phase d'exploitation, un robot a toujours la possibilité d'apprendre à mieux faire.

La tâche que nous nous sommes assignés consiste à concevoir un système de navigation basé sur l'apprentissage par renforcement, permettant à un groupe de robots mobiles de se déplacer d'une position initiale à une position finale tout en évitant les obstacles dans un environnement inconnu. Et d'autre part de permettre à ces mêmes robots de former une figure géométrique (cercle, cercle rempli, polygone, ligne), et de se déplacer tout en maîntenant cette formation [AZO, 00].

IV. 2 Architecture du robot mobile

L'architecture générale des robots mobiles est illustrée dans la **Figure IV.1**. Les modules principaux constituant cette architecture sont :

- Module 1 : Système de perception.
- Module 2 : Processus d'apprentissage par renforcement (AR).
- Module 3 : Processus de formation des figures géométriques.
- Module 4 : Processus de formation des figures géométriques avec l'apprentissage par renforcement.
- Module 5 : Processus de navigation en formation.

La navigation d'un ou plusieurs robots mobiles appelée respectivement navigation individuelle et navigation en groupe utilise le processus de l'apprentissage par renforcement. Par contre, le déplacement en formation (cercle, cercle rempli,...) des robots mobiles utilise le processus de navigation en formation.

Nous allons voir plus en détail chacun de ces modules dans les paragraphes suivants.

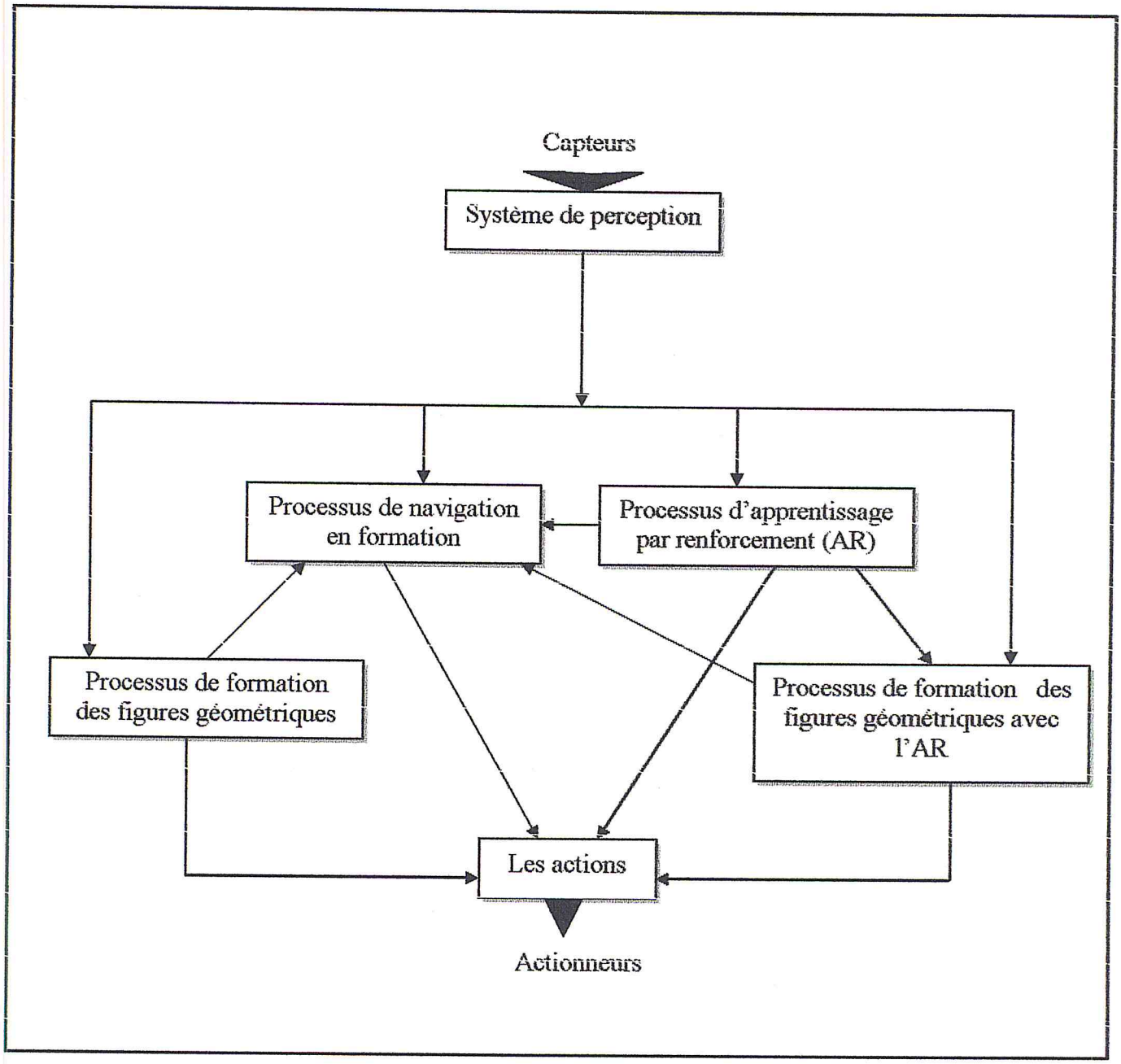


Figure IV.1 : Architecture du robot mobile autonome.

IV.2.1 Système de perception

Le système de perception permet à un robot mobile de découvrir son environement d'évolution au fur et à mesure. Il existe actuellement plusieurs type de dispositifs de perception, entre autres : les capteurs tactiles, les capteurs à ultrason, les capteurs lasers et les cameras [PRU, 98]. Puisque l'approche adoptée dans ce travail est locale, le robot doit être équipée d'un dispositif lui permettant de percevoir son environnement. Le robot doit faire un diagnostic de sa situation par rapport à ce qui l'entoure. Ceci n'étant possible que par un moyen de perception pouvant donner un maximum d'informations sur son environement. Nous avons utilisé des capteurs infrarouges, et nous avons choisi ce dispositif car il permet au robot d'une part de détecter d'autres robots en mouvement en s'échangeant des informations et d'autre part de reconnaître des objets fixes afin de les éviter. Le choix de ce dispositif est également guidé par sa simplicité d'utilisation.

IV.2.1.1 Capteurs infrarouges

Ils donnent une information de distance entre le robot et l'obstacle le plus proche dans la direction pointée par l'infrarouge. Ils sont également utilisés pour la communication entre robots (voir le fonctionnement du système LOCISS). Ces capteurs utilisent le spectre de lumière infrarouge (invisible pour les humains). Ils mesurent l'angle d'incidence de la lumière réfléchie par un objet (obstacle) qui est d'environ 15 degrés [LYO, 03].

Le robot est équipé de 8 capteurs infrarouges, et la disposition de ces capteurs sur le robot est donnée dans la **Figure IV.2** ci-dessous.

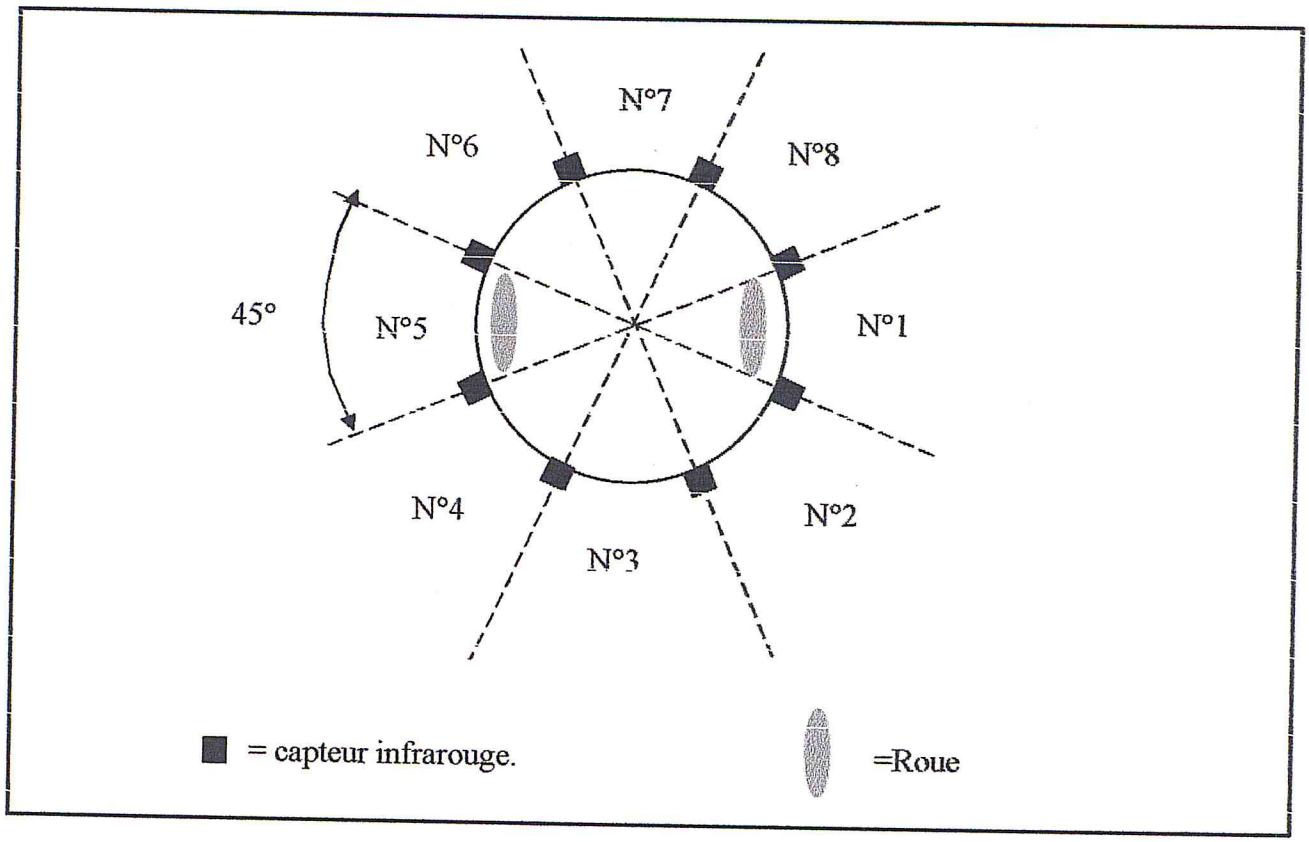


Figure IV.2 : Distribution utilisée des capteurs sur le robot.

IV.2.1.2 Système LOCISS (LOcally Communicable Infrared Sensory System)

L'évitement de collision est l'une des fonctions les plus importantes dans un système multi-robots mobiles pour réaliser des tâches cooperatives. Si un robot rencontre des obstacles ou d'autres robots mobiles sur son chemin , il peut anisi les éviter.

Il y a plusieurs méthodes de planification de chemin pour l'évitement d'obstacle. Elles sont principalement basées sur la distance entre le robot et l'obstacle. Bien que ce genre de méthodes soit efficace pour l'évitement de collision d'objets statiques, il est difficile de les

2. L'exécution du mouvement.
3. La réception du renforcement ***r***.
4. la mise à jour de la table des valeurs.

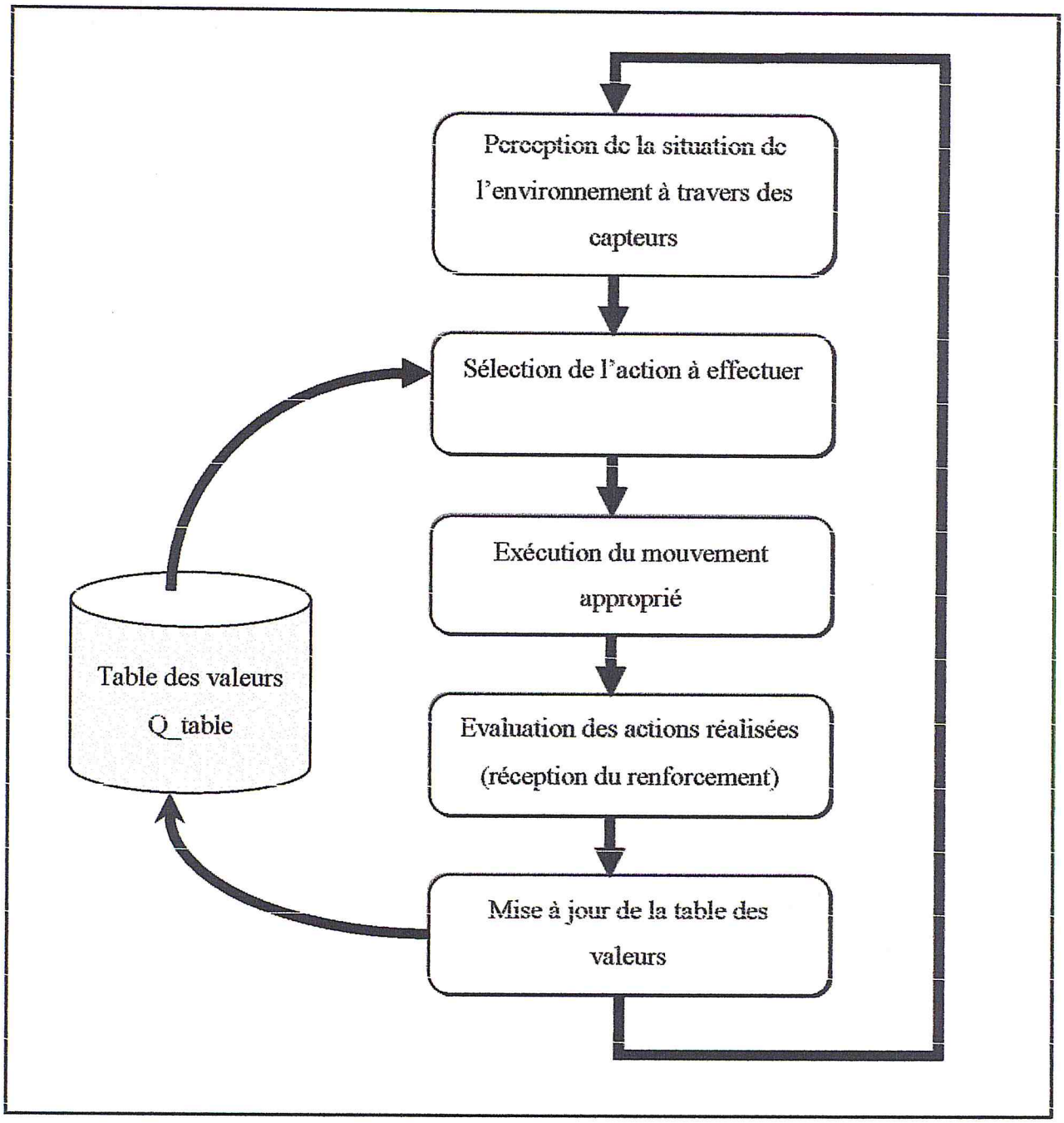


Figure VI.3 : Schéma du processus d'apprentissage par renforcement.

IV.2.2.1 Perception de l'environnement

Le robot possède 8 capteurs infrarouges. Chaque capteur donne une valeur, et on a codé la valeur du capteur sur 1 bit (0 ou 1). La valeur 1 indique qu'il y a un obstacle, la valeur 0 indique le contraire. Le robot fait correspondre à chaque configuration des états des capteurs une situation *s* de l'environnement. 256 situations sont donc possibles (voir exemple de la **Figure IV.4**).

Exemple

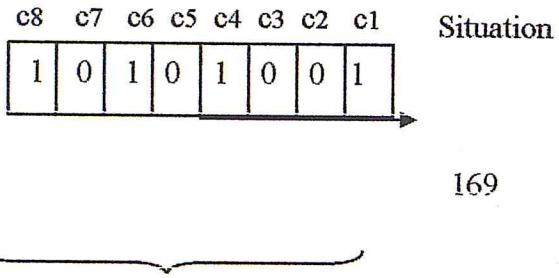


Figure IV.4 : Exemple de calcul d'une situation du robot mobile.

IV.2.2.2 Sélection de l'action

Etant dans la situation *s*, il faut choisir une action à émettre parmi les actions possibles de *A* (ensemble d'actions). On peut déterminer la meilleure action selon la stratégie d'exploitation : argMax $_{a\in A}$ *Q[s] [a]* [JOD, 03]. Selon cette stratégie, La fonction d'évaluation (argMax) parcourt, pour la situation présente, les valeurs de *Q* associées aux actions et sélectionne l'action de plus grande utilité. Au début, lorsque la table ne contient pas encore suffisamment de données, une composante aléatoire (Exploration) est ajoutée de façon à ne pas restreindre les actions éligibles au petit nombre des actions déjà essayées. Au fur et à mesure que la table se remplit, cette composante aléatoire est réduite afin de permettre l'exploitation des informations reçues et d'obtenir une bonne performance. On a utilisé la stratégie **e-greedy** pour choisir une action, tel que, $e \in [0, 1]$:

appliquer à l'évitement de collision d'objets mobiles, l'utilisation de LOCISS permet de pallier ce problème [SUZ, 95a].

Ce dispositif utilise les rayons infrarouges pour fournir au robot la manière de découvrir et d'identifier d'autres robots. Grâce à LOCISS, chaque robot peut reconnaître et distinguer entre les robots et les obstacles en communiquant son identificateur (*id*) et les informations de mouvement telle que la direction et la vitesse de déplacement. Ces informations sont utilisées pour éviter la collision entre les robots et les obstacles dans l'environnement [ARA, 97],[FUJ, 98].

IV.2.2 Processus d'apprentissage par renforcement (AR)

Dans ce paragraphe, nous présentons le processus d'apprentissage par renforcement. Celui-ci permet à un robot (navigation individuelle) ou à plusieurs robots (navigation en groupe) de se déplacer sans collision dans un environnement supposé non connu. Le robot apprend donc au fur à mesure à éviter les obstacles quels qu'ils soient.

L'Apprentissage par Renforcement (AR) est une méthode d'apprendre pour des agents, et elle est souvent utilisée pour les robots mobiles. L'AR suppose que le monde peut être décrit par un ensemble d'états S, et que le robot peut prendre un nombre fixe d'actions A. Le temps est divisé en étapes discrètes. À chaque étape (itération), le robot observe l'état du monde (qui pourrait inclure l'état interne du robot) et choisit une action. Après la prise de la mesure, l'environnement renvoie une récompense r, et le robot observe le nouvel état du monde, s_{t+1}. Le but de l'AR est de prendre ces expériences (s_t ; a_t ; r ; s_{t+1}), et d'améliorer le comportement états-actions associé à ces expériences, selon un algorithme particulier (dans notre cas le Q_learning).

Le processus de navigation basé sur l'apprentissage par renforcement se décompose en cinq étapes comme le montre la **Figure IV.3**, à savoir [AZO, 01] :

1. La perception de l'environnement.

1. prendre argMax $Q[s_t]$ $[a]$ avec une probabilité e-1.
2. prendre une action au hasard avec une probabilité e.

On fait varier alors e(e=e-0.01) au cours des épisodes, où t est le temps.

Les comportements de base

Nous utilisons quatre comportements qui sont :

Aller vers la cible : Le robot se dirige d'un pas vers la cible ignorant les objets autour de lui.
Aller tout droit : Le robot fait un pas sans changer son orientation.
Tourner à droite : Le robot effectue un mouvement à droite par rapport à son orientation.
Tourner à gauche : Le robot effectue un mouvement à gauche par rapport à son orientation.

L'ensemble des actions est : A– {***Aller vers la cible***, ***Tourner à droite***, ***Tourner à gauche***, ***Aller Tout droit***}.

IV.2.2.3 Exécution de l'action

Quand on sélectionne une action $a_t \in A$, à un instant t (initialement, $t - 0$), le robot sauvegarde sa position et exécute l'action convenable. Cette action peut amener le robot vers une collision. Dans le cas d'une collision le robot annule l'action qu'il vient d'exécuter en revenant à la position précédente. Après cette étape, le robot tente une nouvelle action a'.

IV.2.2.4 Fonction de renforcement

La fonction de renforcement fournit pour chaque nouvelle situation un scalaire r (évaluation) c'est à dire après l'exécution d'une action. Cette action a des conséquences à court terme et à plus long terme selon le facteur γ (voir chapitre II). Plus γ s'approche de 1, plus le robot prend en compte les conséquences à long terme de ces actions. Il est important

de remarquer que, bien que le retour de la fonction de renforcement estime l'intérêt de la situation présente, ce retour est utilisé pour définir l'utilité d'effectuer l'action précédente dans la situation précédente. Cette évaluation est une valeur habituellement ternaire (+2 : bon, -4 : mauvais, 0 : sinon). Cependant, il n'est pas toujours possible de connaître l'intérêt d'une situation atteinte par rapport à l'action souhaitée. Dans ce cas, le retour fournit par la fonction de renforcement est nulle.

Dans notre cas nous définissons la fonction de renforcement comme suit :

Si (il y a eu une collision)
Le renforcement ***r*** prend une valeur négative ***n***.
Sinon
Si (le robot se rapproche de sa cible)
Le renforcement ***r*** prend une valeur positive ***p***.
Sinon (c'est à dire le robot s'éloigne de sa cible)
Le renforcement ***r*** prend une valeur négative m.

Où : m<n.

IV.2.2.5 Mise à jour de la table des valeurs

Après l'émission de l'action a dans la situation s_t, le système reçoit un signal de renforcement r. La mise à jour de la table des valeurs ($Q(s_t, a)$) est réalisée de la manière suivante :

$$Q(s_t, a) = Q(s_t, a) + \alpha (r + \gamma . argMax_{a' \in A} Q(s_{t+1}, a') - Q(s_t, a)) \quad \text{IV.1}$$

Où : α (taux d'apprentissage) et γ (facteur de diminution des renforcements) et sont des constantes positives ($\in [0, 1]$).

Le dernier terme de cette expression est une estimation de l'erreur réalisée entre la prédiction de la valeur d'utilité attendue : $r + \gamma.argMax_{a' \in A} Q(s_{t+1}, a')$ et la valeur courante $Q(s_t, a)$. On considère que la valeur suivante est la valeur maximale possible. L'estimation de l'utilité de réaliser a' dans la situation s_{t+1} est transmise à l'estimation liée à réaliser l'action a dans la situation s_t.

La table des valeurs $Q[s][a]$ (voir **Tableau IV.1**) est implémentée par un tableau à deux dimensions avec 256 lignes correspondant aux différentes situations $\{s_0, s_1, \ldots, s_{255}\}$ et quatre colonnes correspondant aux actions $\{a_0$: ***Aller vers la cible***, a_1: ***Tourner à droite***, a_2: ***Tourner à gauche***, a_3: ***Aller Tout droit***$\}$

Tableau IV.1: Représentation de la table des valeurs.

$Q(s, a)$	a_0	a_1	a_2	a_3
s_0		-1.5	1	2.5
s_1	0.5			3
s_2	-1.2			
s_{254}	4	2		-3.2
s_{255}	1		1.2	2.5

La table des valeurs est représentée par un tableau à deux dimensions des couples de situations-actions. Les lignes correspondent aux situations et les colonnes aux actions. Les cases non vides de cette table correspondent aux couples déjà essayés. Les valeurs de $Q(s, a)$ sont calculées grâce à l'application de l'équation de mise à jour **IV.1**.

Nous avons présenté les différentes fonctions pour le processus de l'apprentissage par renforcement selon l'algorithme Q_learning. L'algorithme Q_learning correspondant, avec une table commune des valeurs pour tous les robots, est le suivant:

Algorithme Q_learning

- Initialiser la table $Q[s]\ [a]$ arbitrairement.
- **Répéter**
  - $t \leftarrow 0$ //temps ou pas d'itération.
  - Initialiser la situation initiale : s_t
  - **Répéter** // Effectuer un épisode.
    - Sélectionner l'action à émettre dans la situation s_t : a_t
    - émettre cette action.
    - observer le signal de renforcement r et la nouvelle situation s_{t+1}.
    - mettre à jour $Q[s]\ [a]$:

$$Q[s_t]\ [a_t] \leftarrow Q[s_t]\ [a_t] + \alpha[r+\gamma.argMax_{a'} Q\ [s_{t+1}]\ [a'] - Q[s_t]\ [a_t]].$$

    - $t \leftarrow t+1$
  - **Jusqu'à** ce que s_t soit une situation terminale.
- **Jusqu'à** condition d'arrêt remplie.

Pour des tables des valeurs différentes, c'est-à-dire chaque robot à sa propre table des valeurs, on utilise le même algorithme Q_learning précédent en changeant seulement l'équation de mise à jours qui est comme suit :

$$Q\ [i]\ [s_t]\ [a_t] \leftarrow Q\ [i]\ [s_t]\ [a_t] + \alpha[r+\gamma.argMax_{a'} Q[i]\ [s_{t+1}]\ [a'] - Q[i]\ [s_t]\ [a_t]].$$

Où : $i=1 \ldots n$, tel que n est le nombre de robots.

IV.2.3 Processus de formation des figures géométriques

Parmi toutes les formations géométriques possibles, nous avons utilisé dans le cadre de la navigation collective quatre formations géométriques (Cercle, Cercle rempli, Polygone, ligne), et nous avons choisi les algorithmes qui nous donnent une bonne approximation de la formation finale.

Nous avons utilisé une stratégie simple d'évitement de collision qui a été proposée par Suzuki et Sugihara [SUZ, 95]. Si un robot détecte un autre robot tout près sur la direction de son chemin, alors il fait un écart vers la gauche, à condition qu'il trouve avec succès une direction qui soit nette pour n'importe quel robot. Si l'action tourner à gauche est impossible, le robot ne se déplace pas jusqu'à ce que son chemin devienne dégagé ou bien une dérivation à gauche devienne possible.

IV.2.3.1 Formation d'un cercle

L'algorithme de cercle qu'on a utilisé a été proposé par Suzuki et Sugihara [SUZ, 95]. Cet algorithme donne une bonne approximation du cercle final. On rappelle le principe de cet algorithme qui est le suivant :

Soit R un robot mobile, et soient R_1 et R_2 deux autres robots dont les positions sont respectivement la plus proche et la plus lointaine de celle du robot R.
R se déplace comme suit [SUZ, 95]:

- **Étape 1 :** Le robot R se déplace vers R_1, si $d > D$.
- **Étape 2 :** Le robot R s'éloigne de R_1, si $d < (D - \xi)$.
- **Étape 3 :** Le robot R s'éloigne de R_2, si $(D - \xi) \leq d \leq D$.

Où ξ est une petite constante positive, D est le diamètre du cercle à former, et d la distance entre R_1 et R_2.

Arrêt de la formation d'un cercle

L'algorithme n'à aucun mécanisme pour terminer son exécution, une certaine intervention humaine est nécessaire pour le faire stopper ce qui est le cas dans Suzuki et Sugihara [SUZ, 95]. Dans notre cas, il n'y a pas d'intervention humaine mais une méthode de vérification de l'approximation du cercle est ajoutée. Pour chaque robot R, on calcule d qui est la distance entre le robot R et le robot le plus loin R_1, si $(D - \varepsilon) \leq d \leq D+\varepsilon$, où $0 < \varepsilon < \xi$, alors l'algorithme s'arrête.

IV.2.3 .2 Formation d'un cercle rempli

L'algorithme du cercle rempli a été proposé également par Suzuki et Sugihara [SUZ, 95], comme l'algorithme de cercle. A chaque itération, le robot R détermine le robot le plus éloigné R_1 et le robot le plus proche R_2, et R se déplace comme suit :

- **Étape 1 :** Le robot R se déplace vers R_1, si $d > D$.
- **Étape 2 :** Le robot R s'éloigne *de* R_2, si $d <= D$.

Arrêt de la formation d'un cercle rempli

Pour faire arrêter l'algorithme, on a ajouté une méthode de vérification de l'approximation du cercle rempli. Pour chaque robot R, on calcule d qui est la distance entre le robot R et le robot le plus loin R_1, si $(d < D)$ alors l'algorithme s'arrête.

IV.2.3.3 Formation d'un polygone

L'algorithme polygone est réalisé en commandant aux robots d'effectuer l'algorithme de cercle jusqu'à la distribution approximative d'un cercle, pour que chaque robot puisse reconnaître son voisin de gauche et de droite, notés l(R) et r(R) respectivement [SUZ, 95].

L'utilisateur sélectionne ensuite **n** robots qui seront les sommets du polygone à former, puis les deux étapes suivantes sont exécutées [SUZ, 95]:

3. déplacer les robots manuellement vers leurs positions finales désirées (sommets du polygone).
4. permettre aux robots d'exécuter l'algorithme Contraction dont le principe est :

R se déplace vers le milieu du segment formé par l (R) et r(R) en temps réel.

Arrêt de la formation d'un polygone

Pour faire arrêter l'algorithme, on a ajouté une méthode de vérification de l'approximation du polygone. Pour chaque robot R, on calcule P_M point milieu du segment

IV.2.4.1 Formation d'un cercle

IV.2.4.1.1 Comportements de base pour l'algorithme cercle

La base des valeurs dans l'algorithme d'une approximation de cercle est implémentée par un tableau à deux dimensions avec 768 lignes correspondant aux différentes situations $\{s_0, s_1, s_2, \ldots s_{767}\}$ et six colonnes pour les actions :

a_0 : approcher (R s'approche de R_1)

a_1 : éloigner R_1 (R s'éloigne de R_1)

a_2 : éloigner R_2 (R s'éloigne de R_2)

a_3 : tourner à gauche

a_4 : tourner à droite

a_5 : arrêter

En effet, Le nombre de situations dans cet algorithme dépend des états des capteurs et dépend également de l'algorithme de formation lui-même. D'après la figure IV.6 ci-dessous, si on utilise 256 situations (l'information des capteurs qui renseigne sur l'existence ou non d'obstacles), le robot *R* dans les trois cas est dans la même situation, donc le robot *R* doit apprendre à exécuter une seule action dans cette situation (dans les trois cas), mais d'après l'algorithme de cercle le robot *R* doit s'approcher de R_1 dans le premier cas, s'éloigner de R_1 dans le deuxième cas et s'éloigner de R_2 dans le troisième cas. Pour cette raison, on a ajouté les trois cas de l'algorithme de cercle, le nombre de situation devient alors 256 X 3=768 situations. La fonction qui donne le numéro de situation est la suivante.

Si (capteur (1) −1) alors s−1

Si (capteur (2) =1) alors s=s+2

Si (capteur (3) =1) alors s=s+ 4

Si (capteur (4) =1) alors s=s+8

Si (capteur (5) =1) alors s=s+16

Si (capteur (6) =1) alors s=s+32

Si (capteur (7) −1) alors s−s+64

Si (capteur (8) =1) alors s=s+128

Si (d>D) alors s=s sinon si (d<D- ξ) alors s=s+256 sinon s=s+512

formé par l(R) et r(R). Si la distance entre R et $P_M < \varepsilon$, où ε est une petite constante positive, alors l'algorithme s'arrête.

IV.2.3.4 Formation d'un segment de ligne

La formation d'un segment de ligne est un cas particulier d'un polygone avec seulement deux sommets. Premièrement l'utilisateur sélectionne deux robots qui seront les deux extrémités de la ligne à former et qui doivent être voisins dans le cercle formé, ensuite laisser les autres robots exécuter l'algorithme contraction [SUZ, 95].

La stratégie d'évitement de collision proposée par Suzuki et Sugihara [SUZ, 95] ne garantit pas de résoudre le problème de collision dans toutes les situations possibles. C'est pourquoi nous avons proposé l'introduction de l'apprentissage par renforcement pour que les robots disposent d'un maximum d'autonomie, et deviennent capables de s'adapter à des environnements inconnus et des situations imprévues.

IV.2.4 Processus de formation des figures géométriques avec apprentissage

Dans les algorithmes de formation des figures géométriques sans apprentissage, les robots peuvent entrer dans des situations imprévues, les résultats obtenus peuvent être médiocres. Face à une telle situation, nous avons proposé l'introduction de l'apprentissage par renforcement dans le processus des figures géométriques sans apprentissage.

Nous avons alors utilisé l'algorithme d'apprentissage par renforcement "Le Q-learning" afin de pallier ce problème. Nous allons voir plus en détail les comportements de base et les fonctions de renforcement pour la formation d'un cercle et d'un cercle rempli dans les paragraphes qui suivent.

IV.2.4.1.2 Fonction de renforcement pour l'algorithme cercle

La fonction de renforcement est donnée ci-dessous.

Si (il y a eu collision)

Alors Le renforcement **r** prend une valeur négative, n_1

Sinon

Si ($d > D$)

Alors { **Si** (R se rapproche de R_1)

Alors r prend une valeur positive, p

Sinon

r prend une valeur négative, n_2

Si ($d < (D - \xi)$)

Alors { **Si** (R s'éloigne de R_1)

Alors r prend une valeur positive, p

Sinon

r prend une valeur négative, n_2

Si ($(D - \xi) \leq d \leq D$)

Alors { **Si** (R s'éloigne de R_2)

Alors r prend une valeur positive, p

Sinon

r prend une valeur négative, n_2

Où : $n_1 < n_2$.

Où capteur (i) donne deux valeurs, la valeur 1 indique qu'il y a un obstacle, la valeur 0 indique le contraire et s est le numéro de situation.

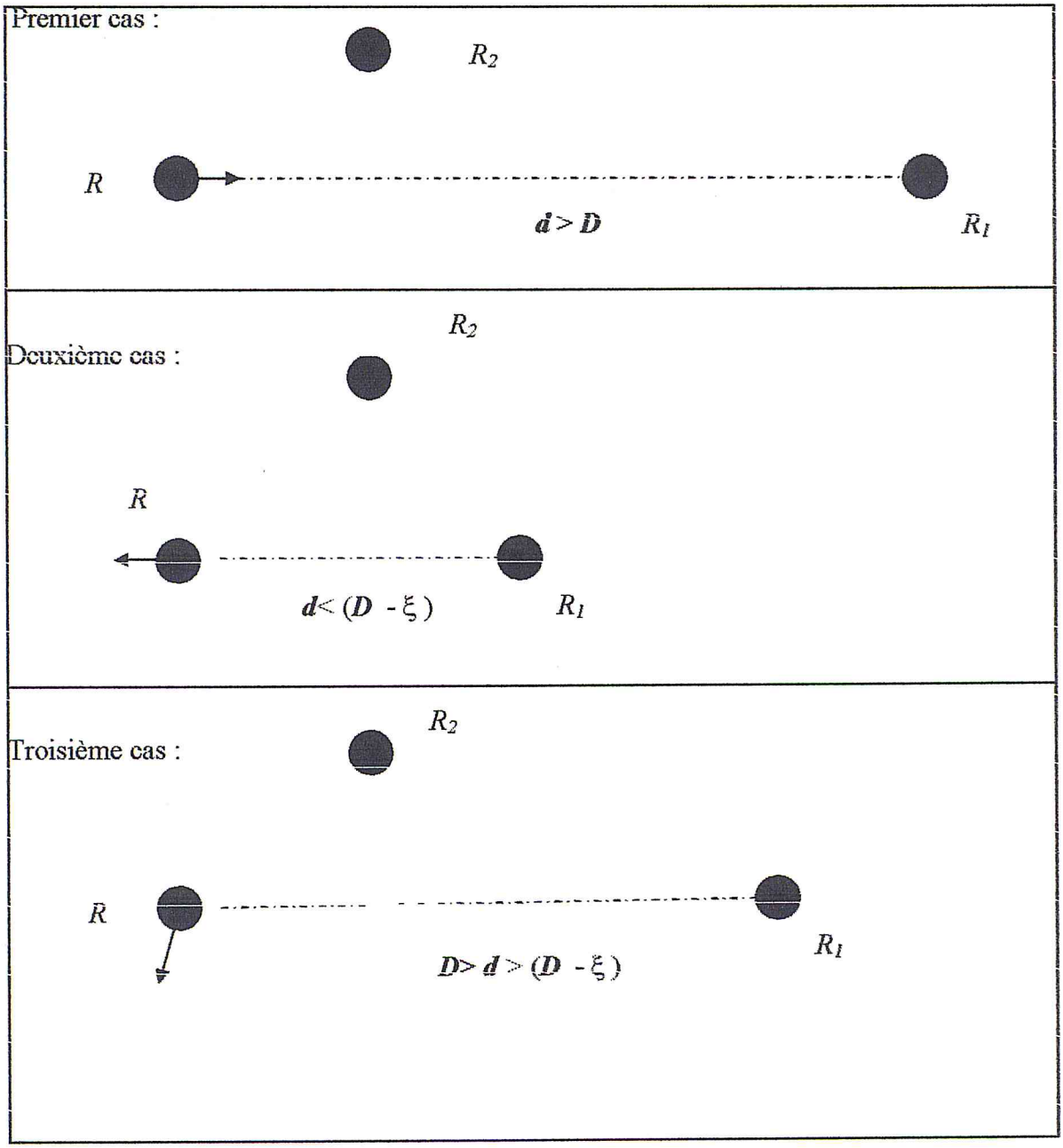


Figure IV.5 : Différentes étapes de l'algorithme cercle.

IV.2.4.2 Formation d'un cercle rempli

IV.2.4.2 .1 Comportements de base pour l'algorithme cercle rempli

La base des valeurs dans l'algorithme d'une approximation de cercle rempli est implémentée par un tableau à deux dimensions avec 521 lignes correspondant aux différentes situations $\{s_0, s_1, s_2, \ldots s_{511}\}$ et six colonnes pour les actions :

a_0 : approcher (R s'approche de R_1)

a_1 : éloigner R_2 (R s'éloigne de R_2)

a_3 : tourner à gauche

a_4 : tourner à droite

a_5 : arrêter

La fonction qui donne le numéro de situation est la suivante.

Si (capteur (1) –1) alors s–1

Si (capteur (2) =1) alors s=s+2

Si (capteur (3) =1) alors s=s+ 4

Si (capteur (4) =1) alors s=s+8

Si (capteur (5) =1) alors s=s+16

Si (capteur (6) =1) alors s=s+32

Si (capteur (7) –1) alors s–s+64

Si (capteur (8) =1) alors s=s+128

Si (d>D) alors s=s sinon si (d<D- ξ) alors s=s+256

IV.2.4.2.2 Fonction de renforcement pour l'algorithme de cercle rempli

Si (il y a eu collision)

Alors Le renforcement **r** prend une valeur négative, n_1

Sinon

Si ($d > D$)

Alors
- **Si** (**R** se rapproche de $\overline{R_1}$)
  - **Alors** **r** prend une valeur positive, p
  - **Sinon** **r** prend une valeur négative, n_2

Si ($d \leq D$)

Alors
- **Si** (**R** s'éloigne de R_2)
  - **Alors** **r** prend une valeur positive, p
  - **Sinon** **r** prend une valeur négative, n_2

Où : $n_1 < n_2$.

IV.2.5 Processus de navigation en formation

Pour qu'un groupe de robots mobiles soient capables d'acquérir un comportement coopératif, ils doivent:

- Rester au sein du groupe.
- Eviter des obstacles en groupe.
- Se déplacer en groupe.

Certaines tâches nécessitent la mobilisation d'un grand nombre de robots mobiles autonomes. On peut prendre comme exemple le transport d'un objet d'un point à un autre dans le milieu industriel, où l'objet peut avoir plusieurs formes. Les robots sont alors amenés à former des figures correspondant à l'objet à transporter et ensuite à se déplacer vers le but désiré.

La navigation en formation consiste pour chaque robot (du groupe) à effectuer une opération coopérative qui permet à ces robots de former une figure géométrique. Ils utilisent les techniques de formation des figures géométriques présentées ci-dessus. Après ce processus de formation, les robots doivent se déplacer dans un environnement inconnu en utilisant le processus de navigation basé sur l'apprentissage par renforcement.

La **Figure IV.6** montre le synoptique du système de navigation en groupe basé sur l'apprentissage par renforcement.

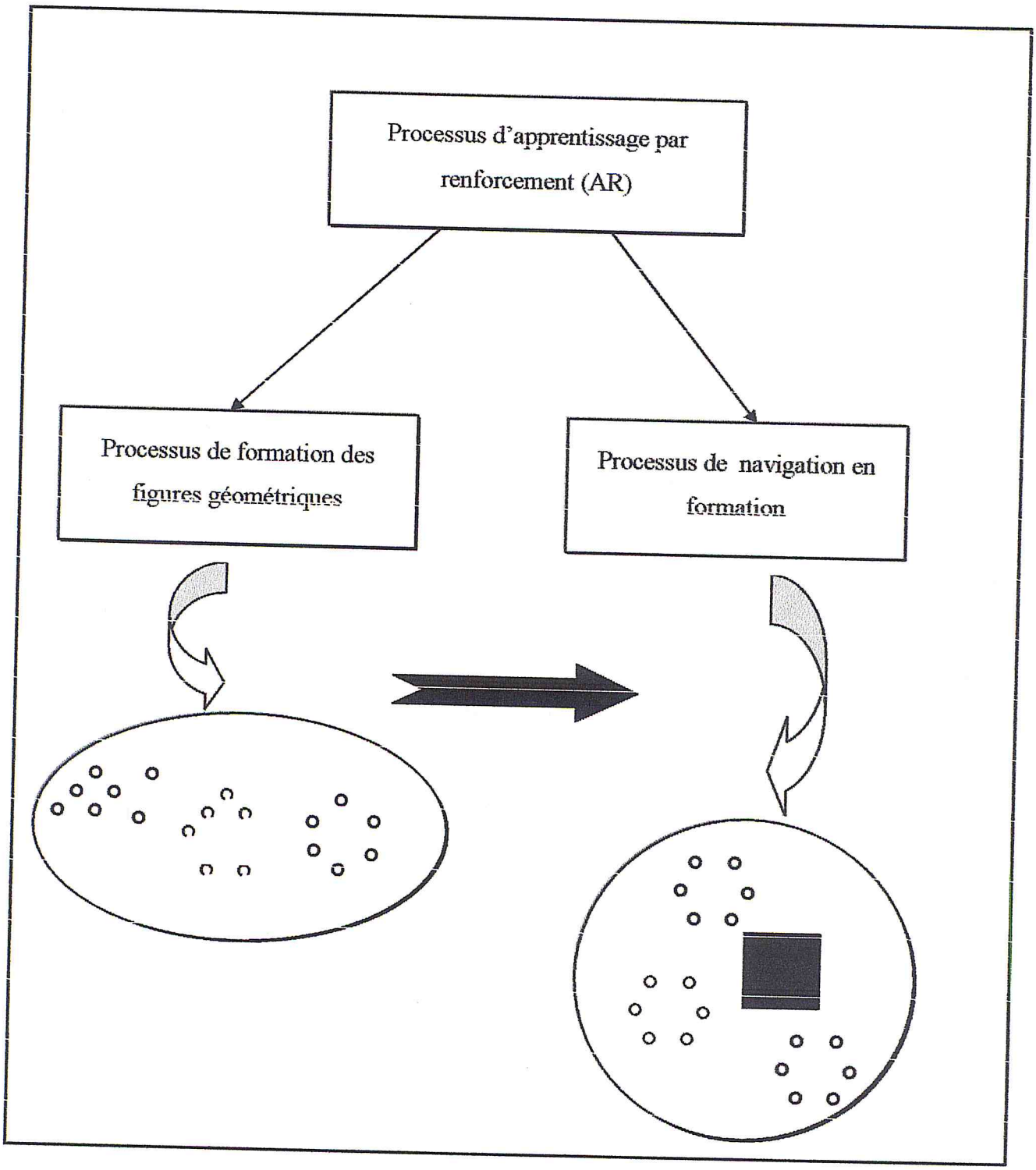


Figure IV.6 : Synoptique du système du navigation en formation basé sur l'apprentissage par renforcement.

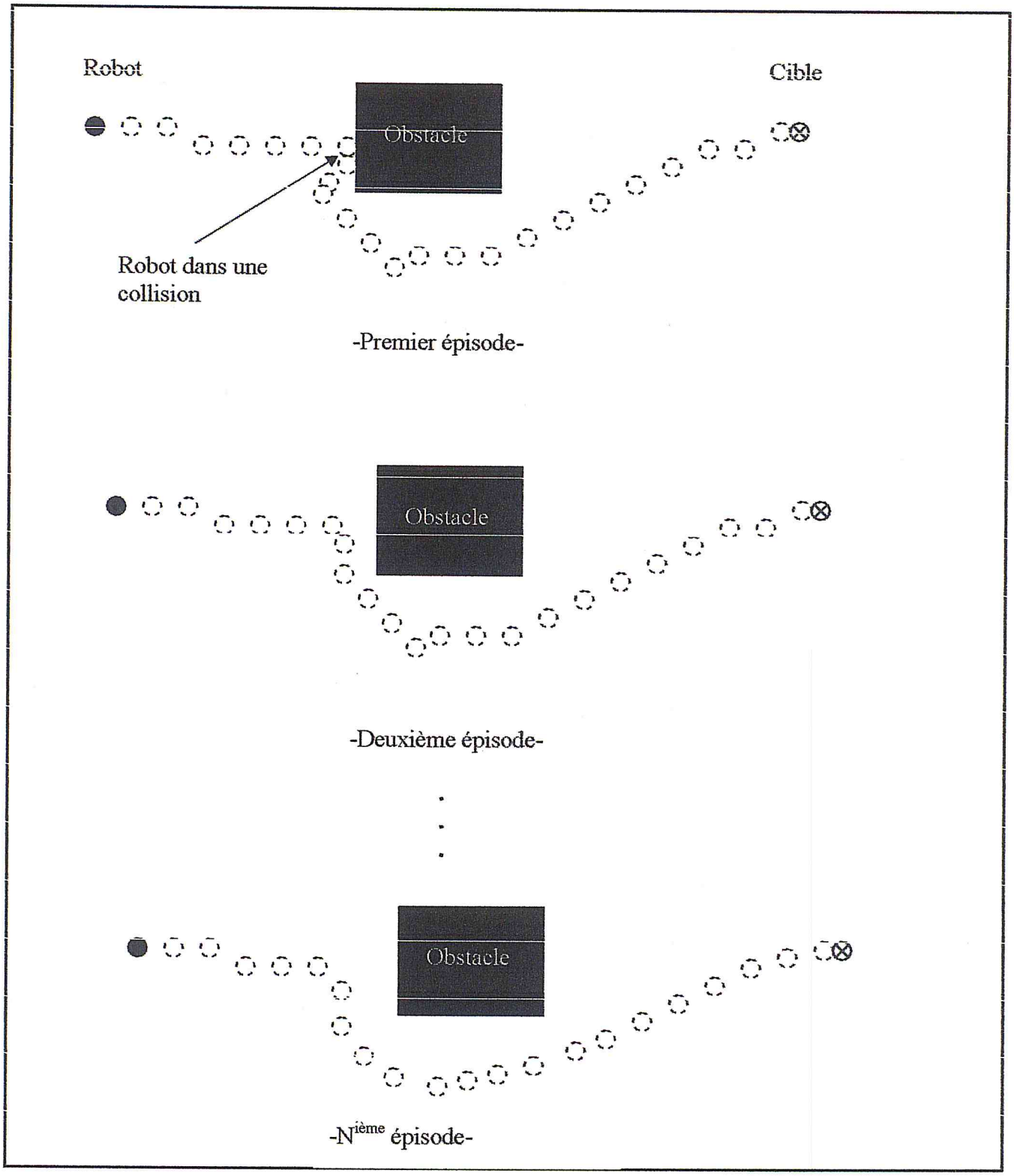


Figure IV.7 : Principe du système épisodique.

IV.4 Conclusion

Dans ce chapitre, nous avons présenté l'approche de navigation en groupe de robots mobiles basée sur l'apprentissage par renforcement dans un environnement inconnu. On a appliqué l'algorithme Q_learning (c'est l'algorithme d'apprentissage par renforcement le plus utilisé) à un groupe de robots mobiles. L'objectif de ce groupe est d'acquérir un comportement collectif et coopératif. Pour cela, nous avons conçu un système qui s'appuie sur le principe suivant:

- Les robots doivent savoir éviter les obstacles et s'éviter mutuellement (processus d'AR),
- Le groupe de robots doit apprendre à former une figure géométrique (l'exécution du processus de formation des figures géométriques avec l'AR),
- Le groupe de robots doit apprendre à naviguer en gardant la formation (processus de la navigation en formation). Cette phase se traduit par le choix d'un conducteur (leader) en premier lieu et ensuite ce conducteur guide l'ensemble des robots vers leur cible tout en maîntenant la figure formée.

Dans le chapitre suivant, nous allons évaluer notre méthode de navigation basée sur l'apprentissage par renforcement présentée dans ce chapitre.

IV.2.5.1 Choix du conducteur du groupe de robots mobiles (leader)

Le conducteur est le robot qui se trouve à la tête du groupe de robots mobiles lorsqu'ils sont en formation. Le conducteur mène la formation à l'endroit désiré [AZO, 00].

Le choix de la position du conducteur dépend du type de formation (cercle, cercle rempli, polygone, ligne). Dans notre cas, le conducteur est l'un des robots mobiles de la formation à un instant *t*. Le processus de contrôle de formation est centralisé (c'est-à-dire qu'un seul robot prend les décisions), dans notre cas, c'est le conducteur qui prend les décisions. Le choix du conducteur est basé sur une contrainte de distance entre robots-cible et robots-obstacle. Cette contrainte correspond à la distance minimale entre le robot et l'obstacle ou entre le robot et la cible. En d'autres termes, le robot se trouvant le plus proche d'un obstacle devient le conducteur. Grâce donc à ce critère, le groupe de robots arrive à naviguer en évitant les obstacles tout atteignant sa cible en gardant la figure réalisée.

IV.3 Système épisodique

Dans la navigation basée sur l'apprentissage par renforcement, le robot doit apprendre le maximum dans un épisode. Un épisode est le déroulement de la tâche du début jusqu'à la fin (navigation d'un robot dans notre cas). En d'autres termes, un épisode est une expérience complète commençant de la position initiale du robot jusqu'à la cible. Après un certain nombre d'épisodes, le robot acquiert un comportement optimal. Ce comportement permet au robot mobile de naviguer dans un environnement inconnu sans tomber dans une collision.

La **Figure IV.7** montre le principe du système épisodique pour un seul robot et un seul obstacle.

CHAPITRE

V

V Simulation, tests et résultats

V.1 Introduction

Afin d'évaluer notre méthode de navigation basée sur l'apprentissage par renforcement, présentée au chapitre précédent, on a implémenté cette méthode en utilisant un simulateur des robots mobiles *Mobotsim* sur un PC Intel Celeron 1.7 Mhz.

Le simulateur *Mobotsim* permet de visualiser les différentes situations prises par le robot mobile pendant son déplacement, et de programmer le contrôle de ce robot en utilisant le langage *Visual Basic*. Il permet également de simuler les erreurs de capteurs.

Dans ce chapitre, on présente les résultats de simulation concernant :

- La navigation individuelle et en groupe de robots mobiles basée sur l'apprentissage par renforcement.
- La formation des figures géométriques.
- La formation des figures géométriques par l'apprentissage par renforcement.
- La navigation d'un groupe de robots mobiles se déplaçant en formation.

L'interface et l'environnement de développement du simulateur *Mobotsim* est présentée dans la figure V.1.

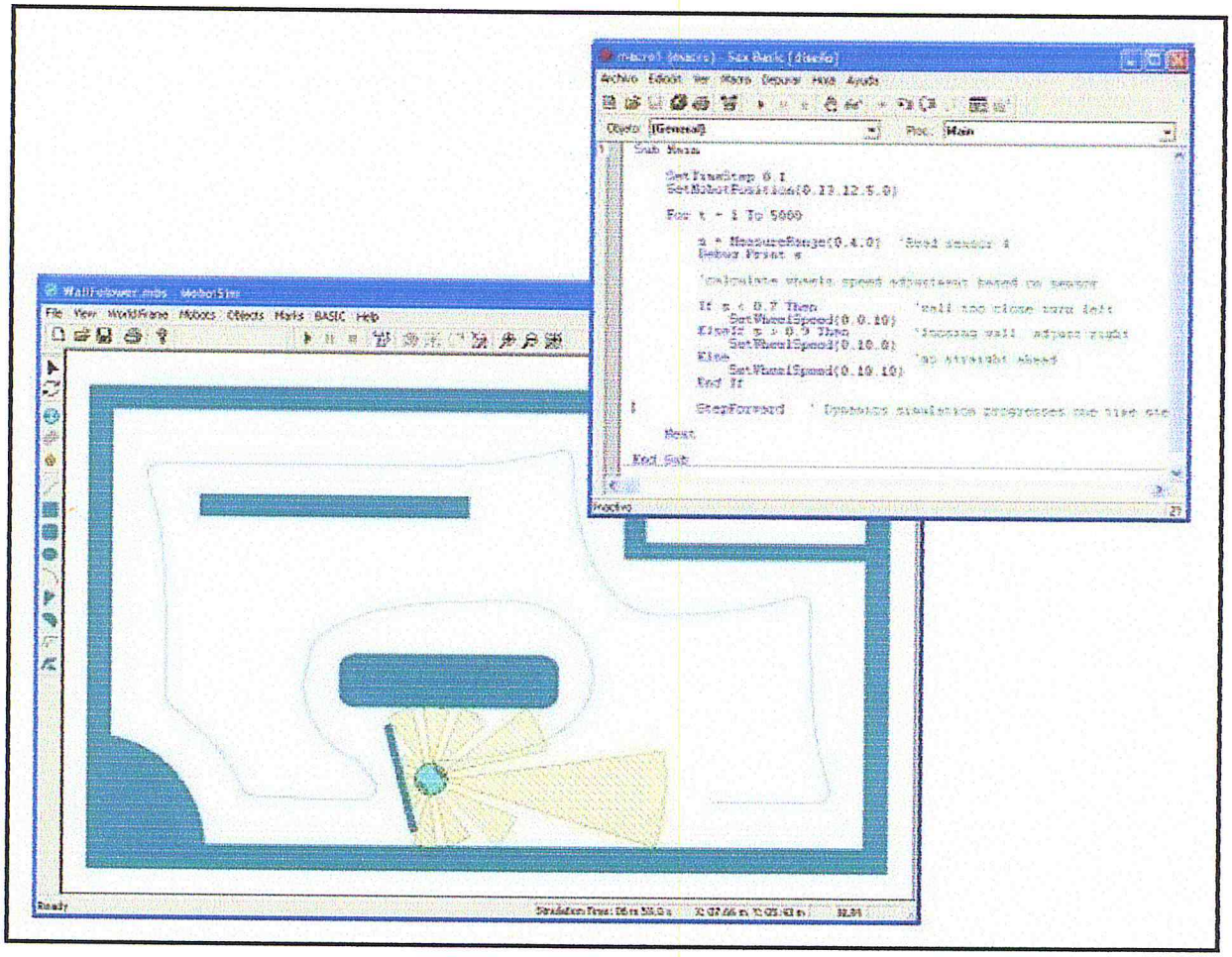


Figure V.1 : Interface et environnement de développement.

V.2 Navigation individuelle et en groupe basée sur l'apprentissage par renforcement de robots mobiles autonomes

Dans cette partie, nous présentons les résultats obtenus dans la phase de simulation. Ces résultats concernent la navigation individuelle en premier lieu et en second lieu la navigation en groupe de robots mobiles basée sur l'apprentissage par renforcement.

Le **Tableau V.1** représente les paramètres de l'algorithme Q_learning pour les expériences de cette partie.

Tableau V.1 : Paramètres de l'algorithme Q_learning.

Paramètres	valeur
ALPHA (α)	**0.5**
GAMMA (γ)	**0.5**
Renforcement cas collision	**-4**
Renforcement cas rapprochement de la cible	**2**
Renforcement cas éloignement de la cible	**-1**
Taux de l'exploration	**5%**

V.2.1 Navigation individuelle d'un robot mobile

Deux exemples sont présentés pour illustrer les résultats obtenus en appliquant l'approche développée dans le présent travail.

Exemple1

La **Figure V.2** montre un robot évoluant dans un environnement simple où un seul obstacle est présent. Le robot va apprendre grâce à l'approche proposée, à éviter cet obstacle. En effet, dans le premier et deuxième épisode le robot entre en collision avec l'obstacle, comme montrée sur la **Figure V.3**. Mais à partir du troisième épisode, le robot se déplace sans toucher l'obstacle. Il a donc appris à l'éviter, ce qui était prévisible.

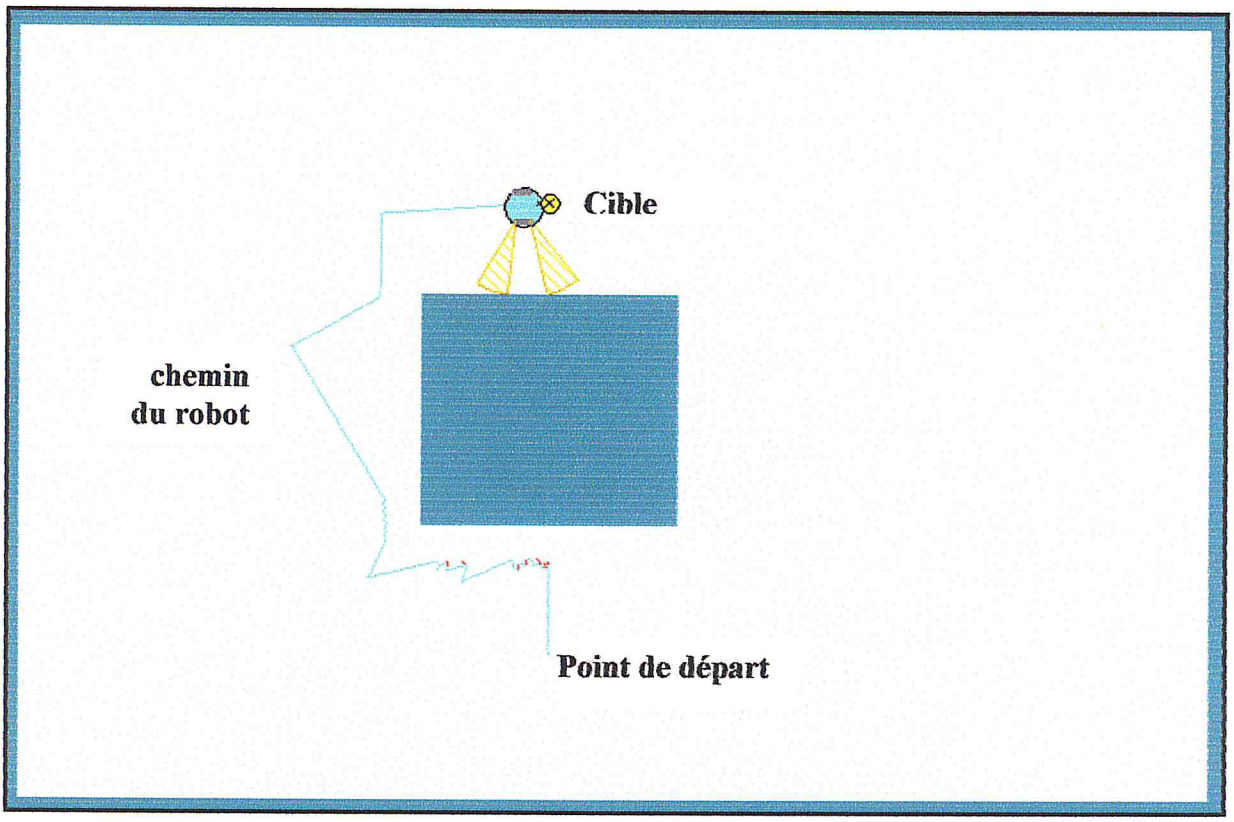


Figure V.2 : Evitement d'un seul obstacle.

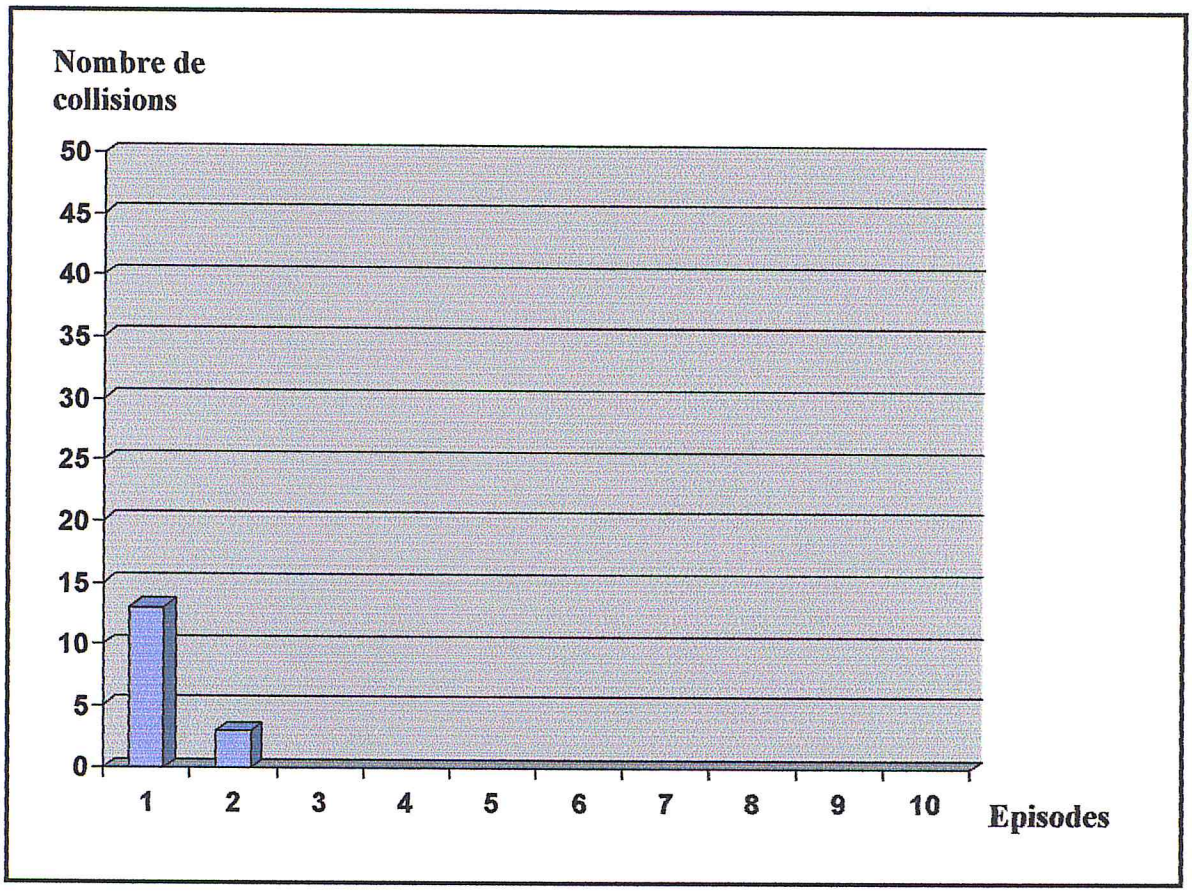


Figure V.3: Graphe de performance (Exemple1).

Exemple2

Dans ce deuxième exemple illustré sur la **Figure V.4**, le robot se déplace dans un environnement qui contient des obstacles de différentes formes. Ce robot va donc prendre plus de temps pour apprendre à éviter tous les obstacles qu'il trouve sur son chemin. En effet, il a fallu 4 épisodes (voir **Figure V.5**) avant que le robot ne puisse naviguer sans collision et atteindre sa cible avec succès.

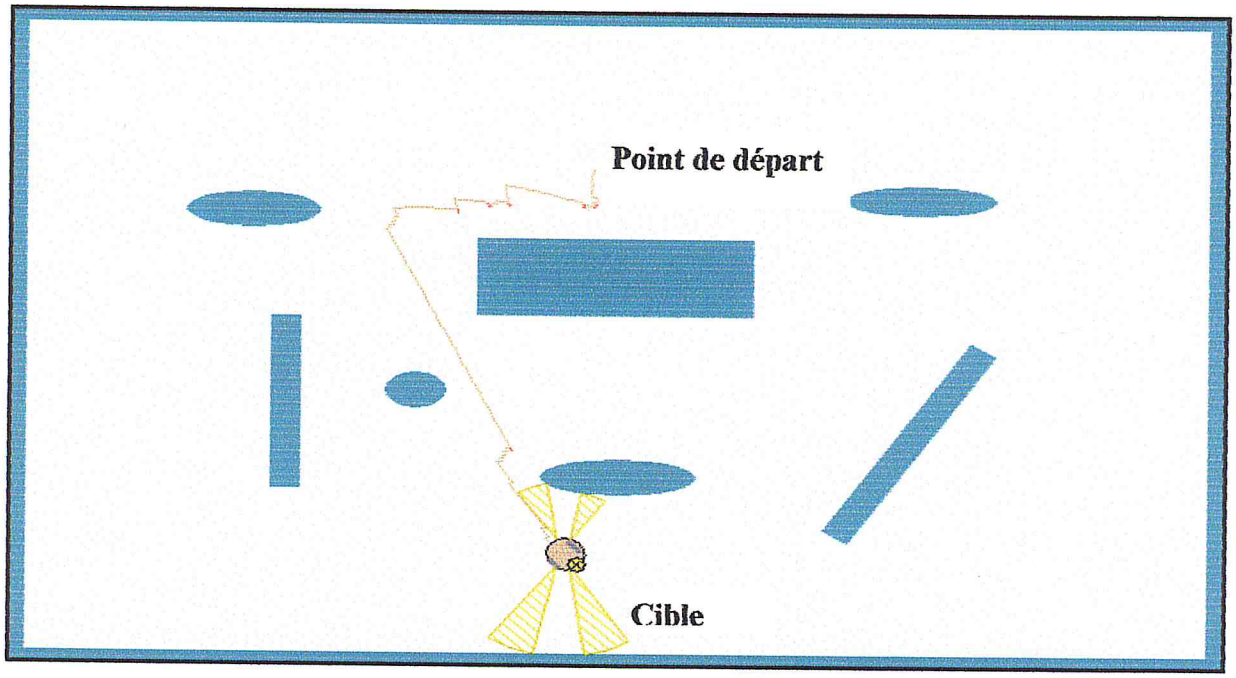


Figure V.4 : Evitement de plusieurs obstacles.

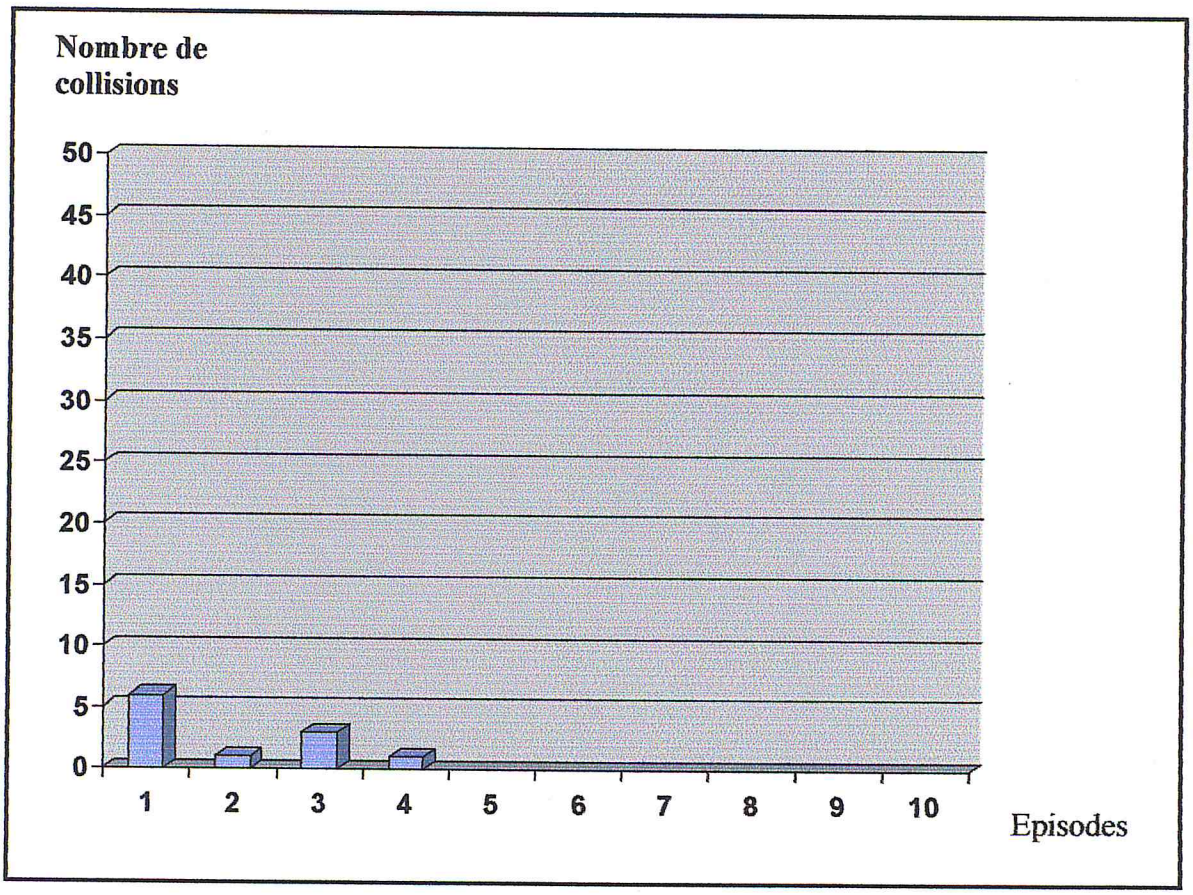


Figure V.5 : Graphe de performance (Exemple2).

V.2.2 Navigation d'un groupe de robots mobiles

Dans cette expérience (voir exemple3), nous utilisons trois robots mobiles, tel que chaque robot à sa cible et sa table de valeur (Q_valeur). Les résultats obtenus sont présentés dans le graphe de performance (Figure V.7).

Exemple 3

Dans cet exemple, on a trois robots mobiles se déplaçant dans un environnement inconnu (voir la figure V.6). Nous remarquons qu'à partir du cinquième épisode les trois robots arrivent à atteindre leurs cibles sans entrer en collision (voir graphe V.7).

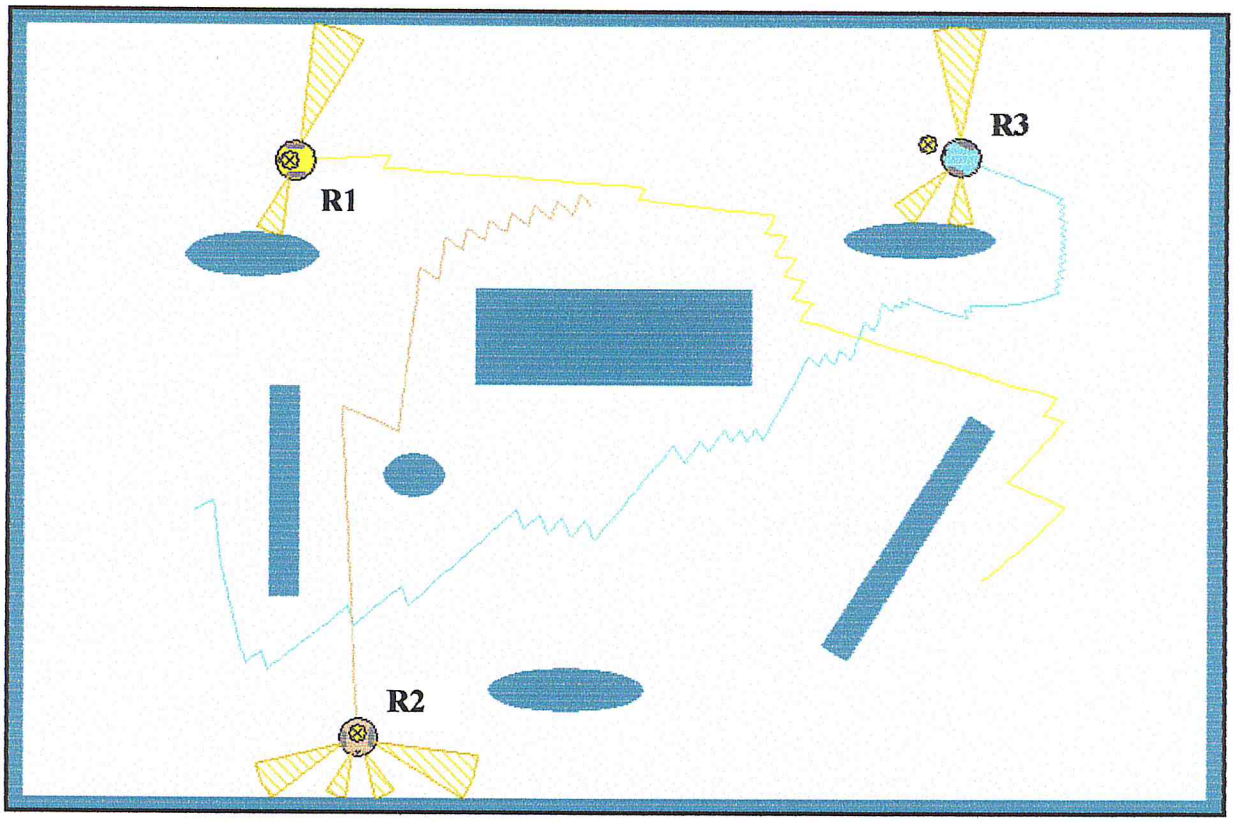


Figure V.6 : Navigation de trois robots mobiles.

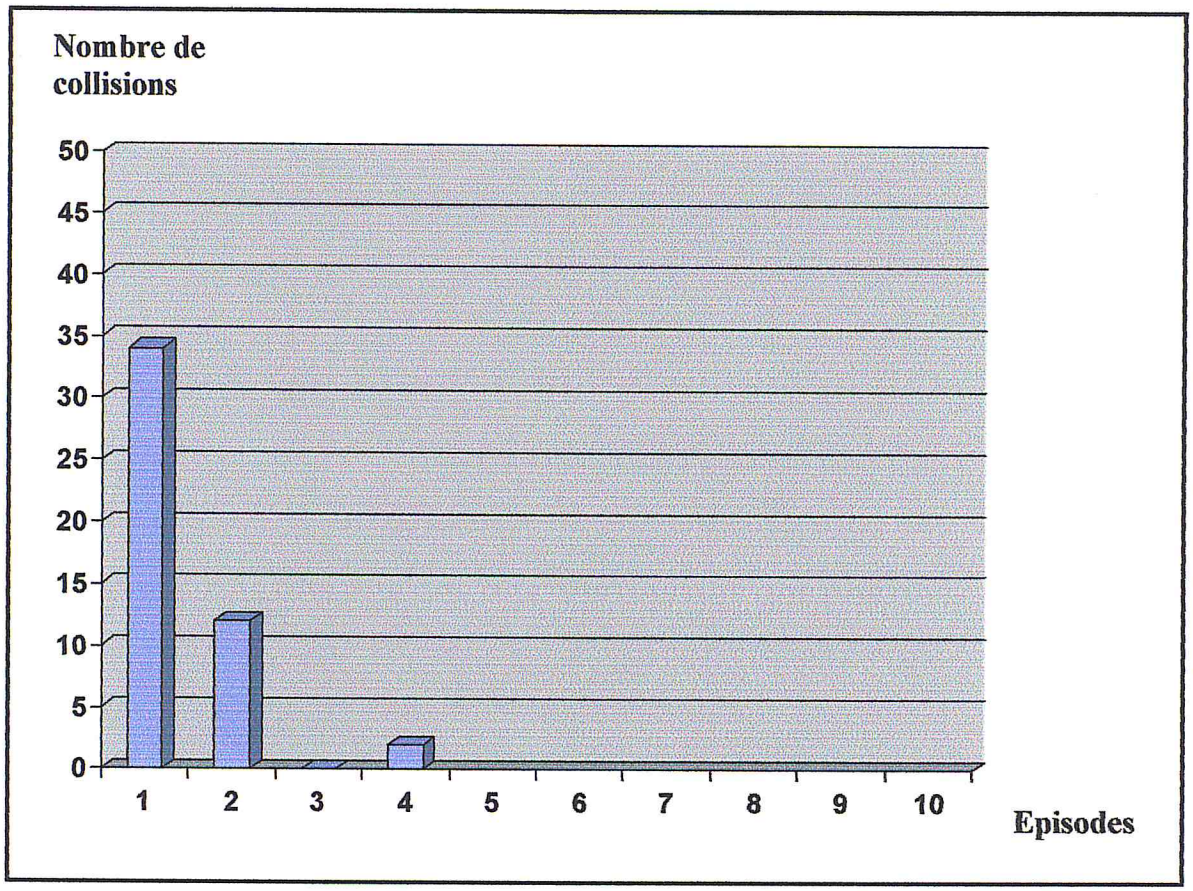


Figure V.7 : Graphe de performance (Exemple3).

V.3 Formation des figures géométriques

On rappelle que chaque robot peut connaître les positions des autres robots à l'aide de son système de perception.

Le nombre de robots reste un choix de l'utilisateur.

Les robots doivent former des figures géométriques à l'aide des algorithmes présentés au chapitre IV.

V.3.1 Formation d'une approximation d'un cercle

Pour l'exemple 4 de cette simulation, le tableau V.2 représente les paramètres de l'algorithme cercle.

Tableau V.2 : Paramètres de l'algorithme cercle.

Paramètres	valeur
ξ	**0.3**
Diamètre	**8**
Nombre de robots	**13**

Exemple 4

Cet exemple montre la simulation d'un groupe de 13 robots mobiles exécutant l'algorithme cercle. Les résultats de cette expérience sont donnés dans la Figure V.8.

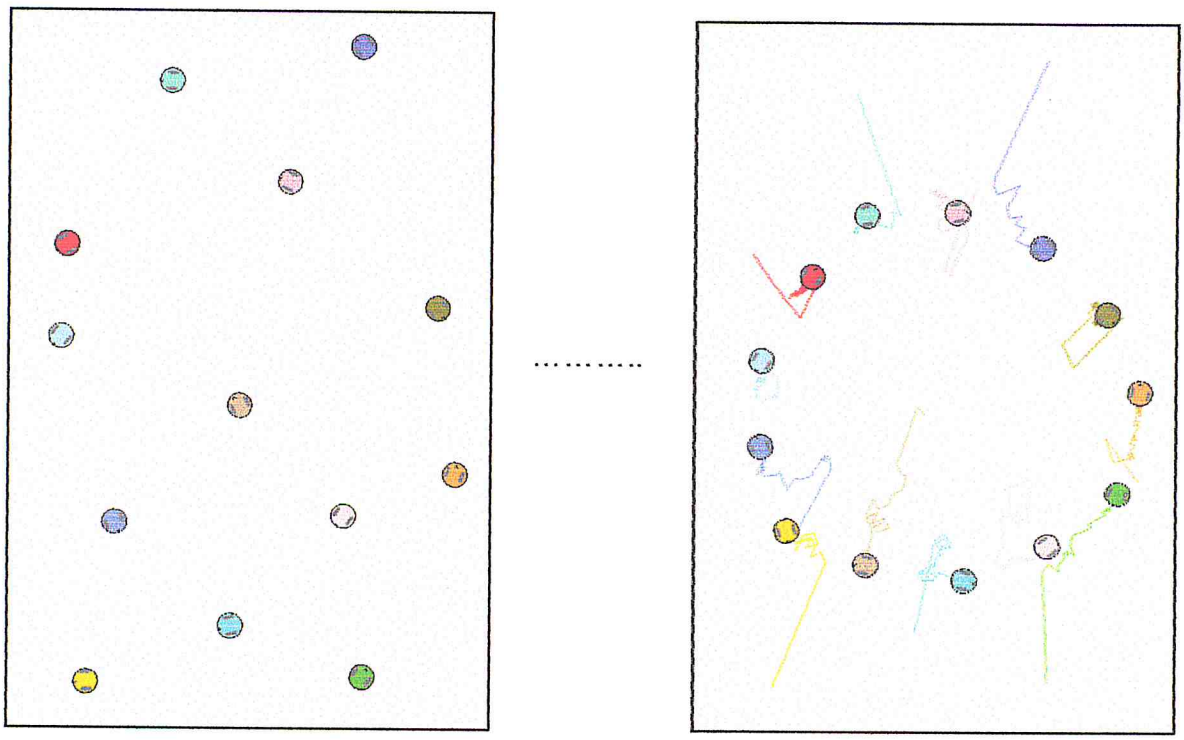


Figure V.8 : Étapes d'exécution de l'algorithme cercle.

V.3.2 Formation d'une approximation d'un cercle rempli

Pour l'exemple 5, le tableau V.3 représente les paramètres de l'algorithme cercle rempli.

Tableau V.3 : Paramètres de l'algorithme cercle rempli.

Paramètres	valeur
Diamètre	**6**
Nombre de robots	**12**

Exemple 5

Cet exemple montre la simulation d'un groupe de 12 robots mobiles exécutant l'algorithme cercle. Les résultats de cette expérience sont donnés dans la Figure V.9.

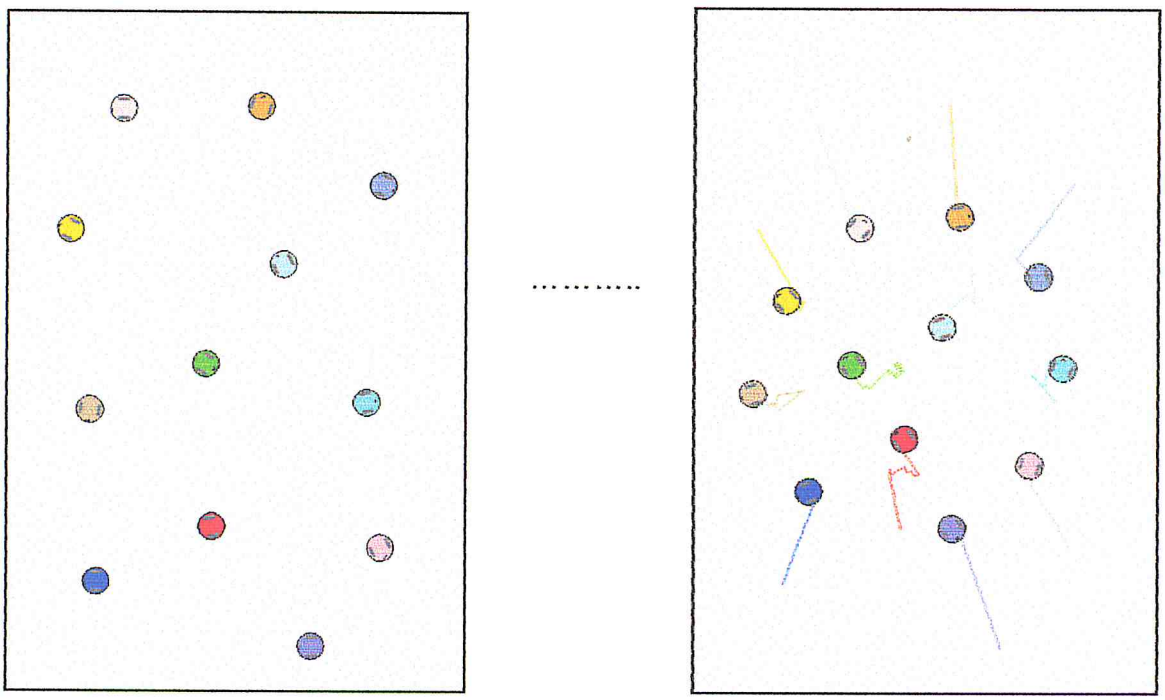


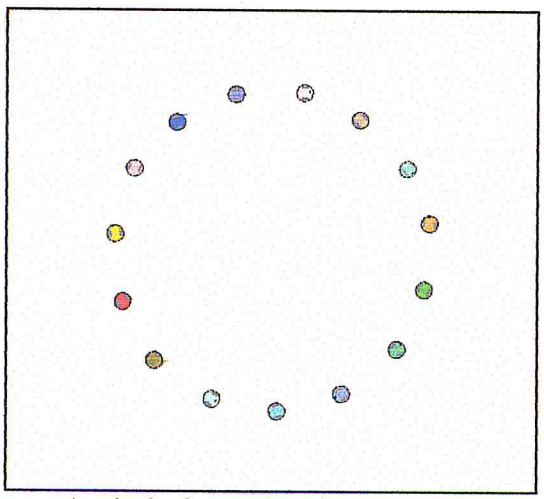
Figure V.9: Étapes d'exécution de l'algorithme cercle rempli.

V.3.3 Formation d'un polygone simple

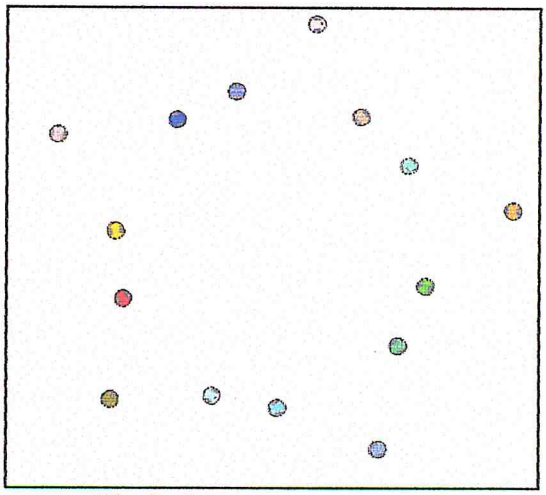
La formation d'un polygone nécessite d'abord l'exécution de l'algorithme cercle, ensuite on fait le choix du nombre de sommets ainsi que leurs positions et on exécute ensuite l'algorithme contraction présenté au chapitre III. Les résultats de cette expérience sont donnés dans la Figure V.10.

Exemple 6

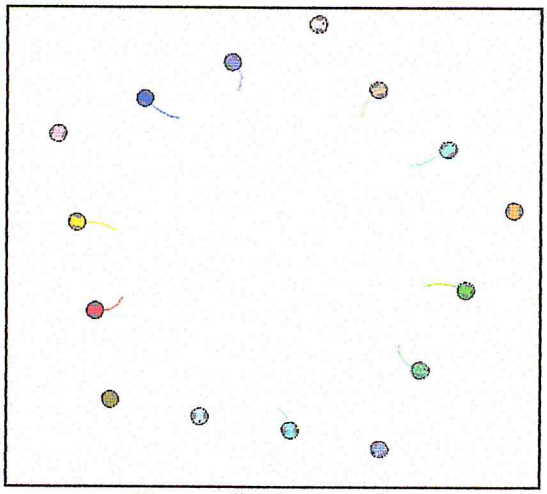
Cet exemple montre la simulation d'un groupe de 15 robots mobiles exécutant l'algorithme polygone.



Après la formation du cercle



Choix des sommets



Forme finale du polygone

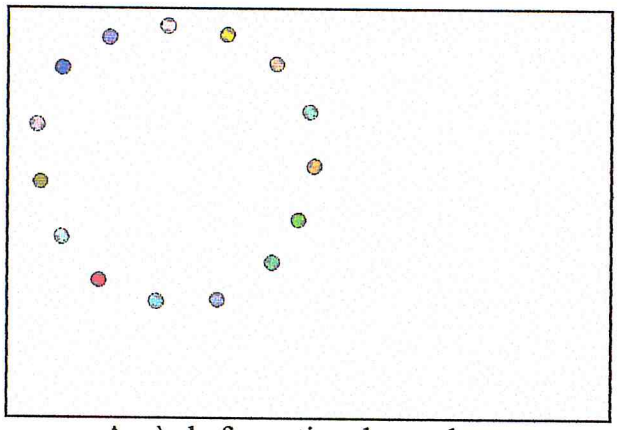
Figure V.10 : Étapes d'exécution de l'algorithme polygone.

V.3.4 Formation d'un segment de ligne

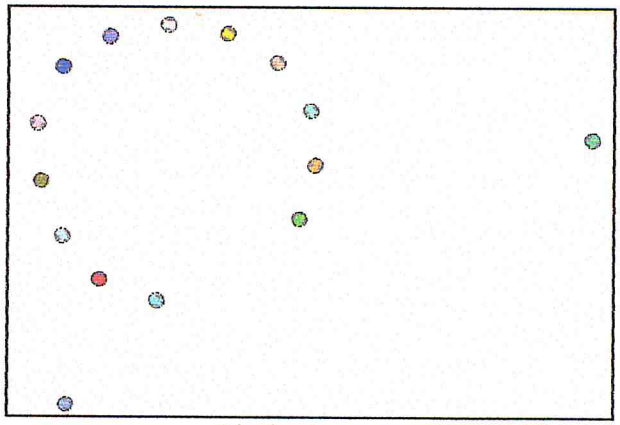
La formation d'un segment de ligne est un cas particulier d'un polygone avec seulement deux sommets. La formation d'un segment de ligne nécessite l'exécution de l'algorithme cercle, ensuite on sélectionne deux robots qui seront les deux extrémités de la ligne à former et qui doivent être voisins dans le cercle formé, et on exécute ensuite l'algorithme contraction présenté au chapitre III. Les résultats de cette expérience sont donnés dans la Figure V.11.

Exemple 7

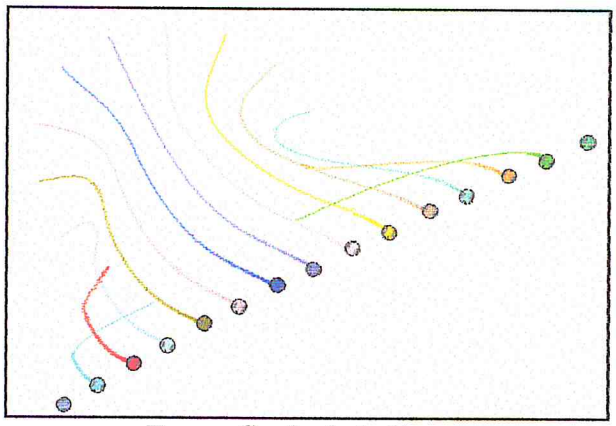
Cet exemple montre la simulation d'un groupe de 15 robots mobiles exécutant l'algorithme ligne.



Après la formation du cercle



Choix des sommets



Forme finale de la ligne

Figure V.11 : Étapes d'exécution de l'algorithme ligne.

Dans les quatre exemples 4, 5, 6 et 7, l'arrêt de la formation se fait automatiquement après que la condition de vérification de l'approximation de la formation soit satisfaite. Nous remarquons que les figures formées sont satisfaisantes.

V.4 Formation des figures géométriques avec l'apprentissage par renforcement

Dans cette expérience (voir figure 12 et 14), un groupe de robots mobiles réalise des figures géométriques en utilisant l'apprentissage par renforcement. Le tableau V.4 représente les paramètres de l'algorithme Q_learning pour les expériences de cette partie.

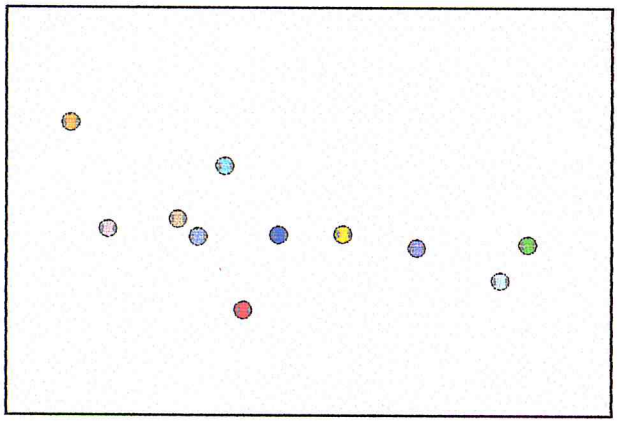
Tableau V.4 : Paramètres de l'algorithme Q_learning.

Paramètres	valeur
ALPHA (α)	**0.5**
GAMMA (γ)	**0.5**
Renforcement cas collision	**-5**
Renforcement négatif	**-5**
Renforcement positif	**+5**
Taux de l'exploration	**5%**

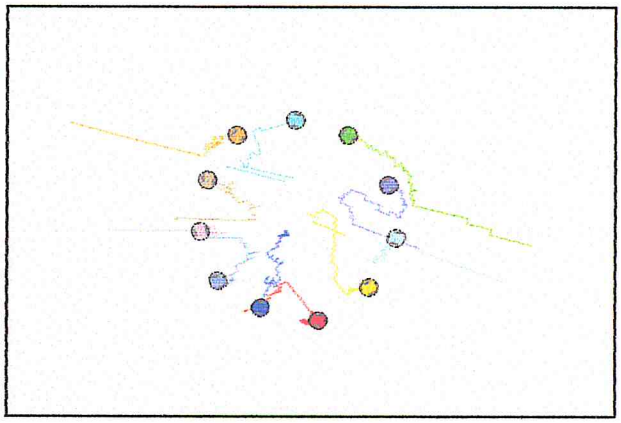
V.4.1 Formation d'un cercle avec l'AR

Dans cette expérience (voir exemple 8 illustré dans la figure V.12), nous utilisons 11 robots mobiles qui utilisent la même table des valeurs (Q_valeur). Les résultats obtenus sont présentés dans le graphe de performance (Figure V.13).

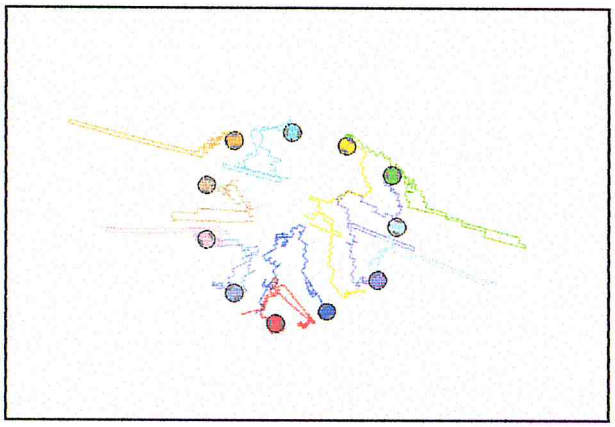
Exemple 8



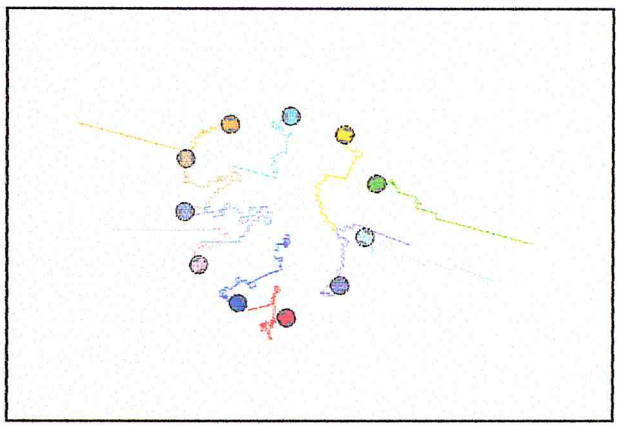
Distribution initiale



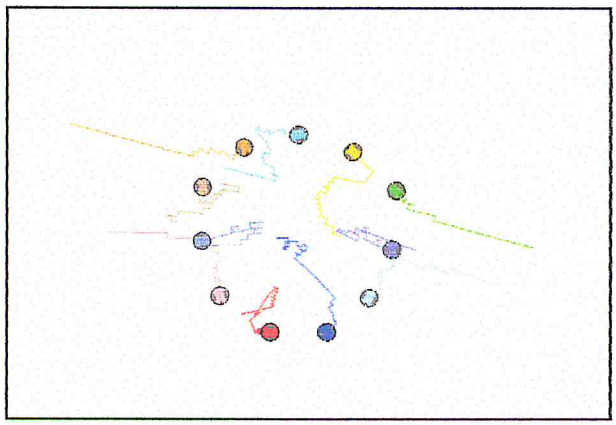
Episode 1



Episode 2



Episode 3



Episode 10

Figure V.12: Etapes de formation d'un cercle avec l'AR.

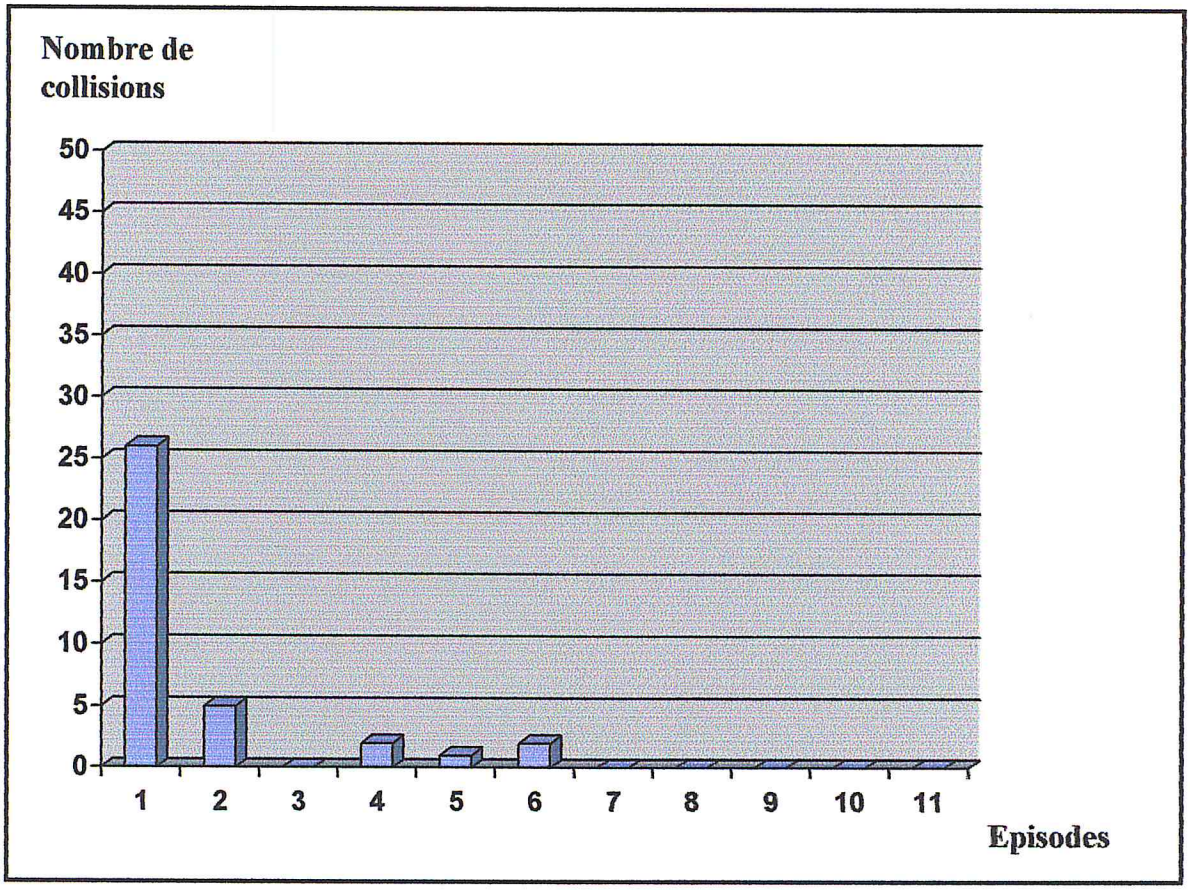
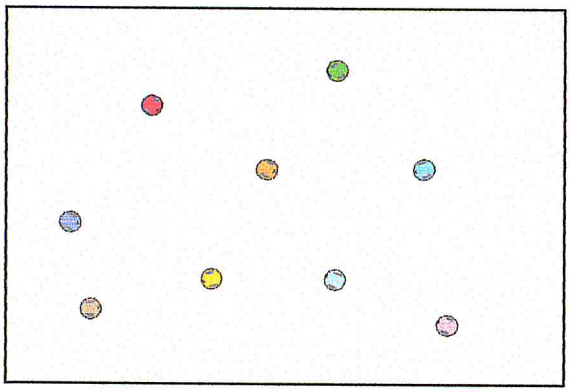


Figure V.13 : Graphe de performance (exemple 8).

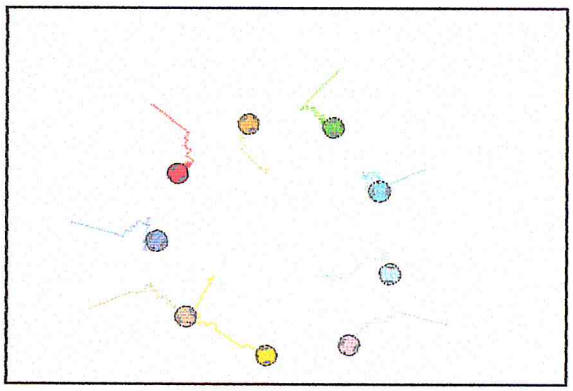
Dans cet exemple, les robots arrivent à former un cercle sans entrer en collision à partir du septième épisode (voir le graphe V.13). Après 5 épisodes sans collision, la connaissance des robots sera sauvegardée pour la réutiliser dans un autre environnement (par exemple on ajoute ou on enlève des robots ou des obstacles) sans passer par la phase d'exploration et sans faire la mise à jour de la table. En effet, le retrait d'un robot comme illustré dans la figure V.14, n'influence en rien l'apprentissage déjà réalisé. D'ailleurs en un seul épisode le cercle a été formé en utilisant la table des valeurs de l'exemple 8.

Exemple 9

Dans cet exemple, la simulation d'un groupe de 9 robots mobiles pour former un cercle est réalisée à l'aide d'une connaissance précédente (la connaissance acquis du test précédent).



Distribution initiale



Forme finale du cercle

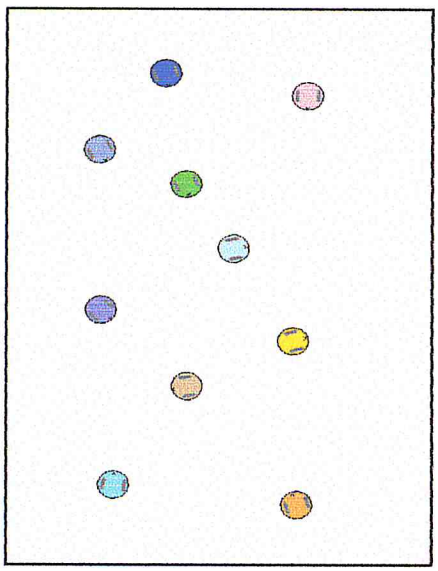
Figure V.14 : Formation d'un cercle à l'aide des connaissances du test précédent.

V.4.2 Formation d'un cercle rempli avec l'AR

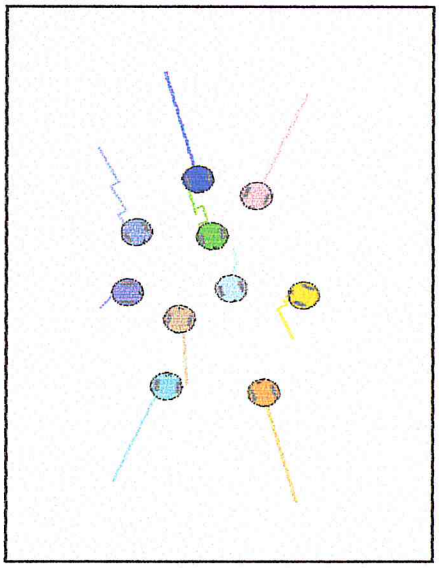
Dans cette expérience (voir exemple 10 illustré dans la figure V.15), nous utilisons 10 robots mobiles qui ont une table des valeurs (Q_valeur) commune. Les résultats obtenus sont présentés dans le graphe de performance (Figure V.16).

Exemple 10

Distribution initiale



Épisode 1



.
.
.

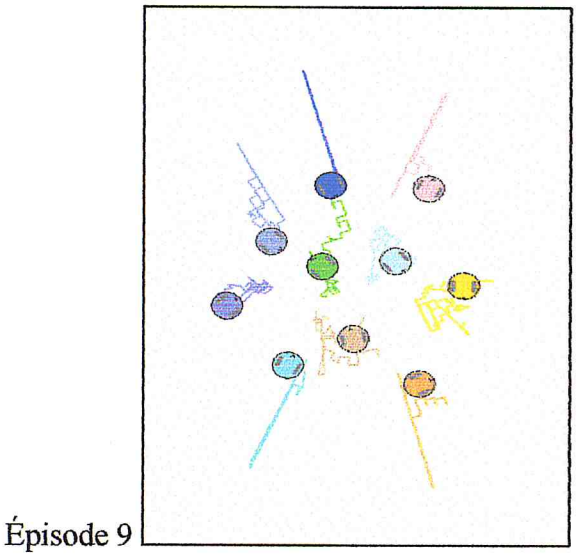


Figure V.15 : Etapes de formation d'un cercle rempli avec l'AR.

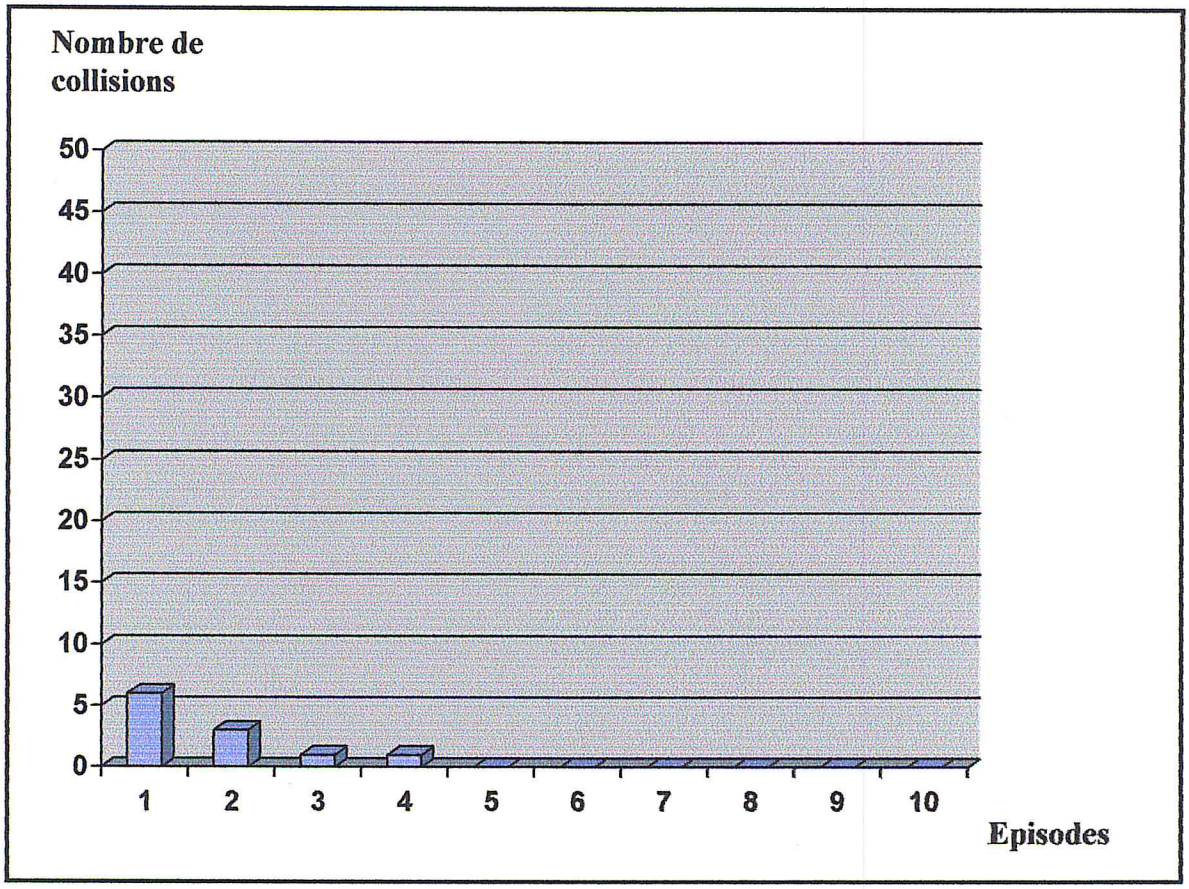


Figure V.16 : Graphe de performance (exemple 10).

V.5 Navigation en formation d'un groupe de robots mobiles

Dans cette expérience (voir Exemples 11 et 12), un groupe de robots mobiles se déplace en formation vers sa cible tout en évitant les obstacles. Ils gardent leur formation initiale du début jusqu'à atteindre leur cible. Les paramètres de l'algorithme Q_learning pour cette expérience sont les mêmes que ceux de la navigation individuelle et en groupe.

Exemple 11

Cet exemple (figure V.17) montre la simulation d'un groupe de 8 robots mobiles formant un *cercle* et se déplaçant dans un environnement inconnu. Le choix du conducteur (leader) est fait à chaque itération. A partir du troisième épisode comme le montre la figure V.18, le groupe de robots arrive à atteindre sa cible sans entrer en collision avec les obstacles.

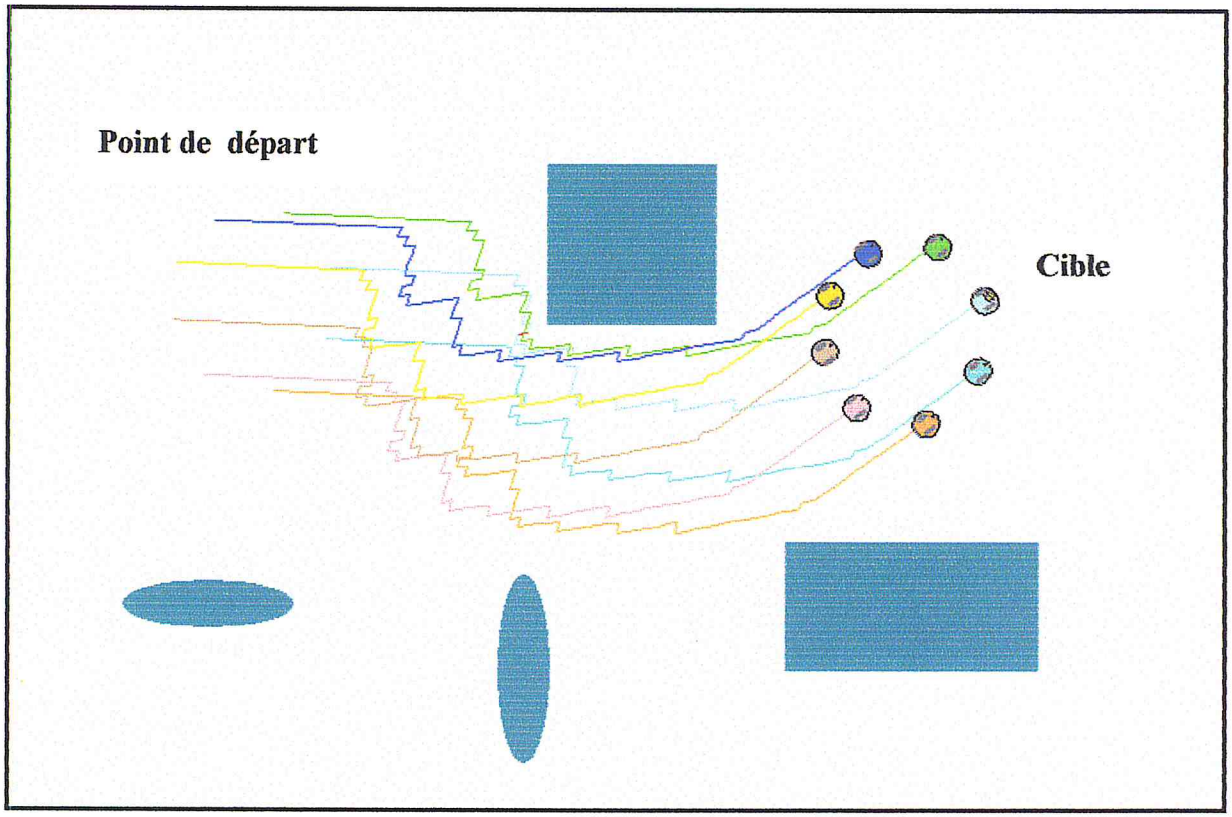


Figure V.17 : Navigation d'un groupe de robots mobiles en forme de *cercle*.

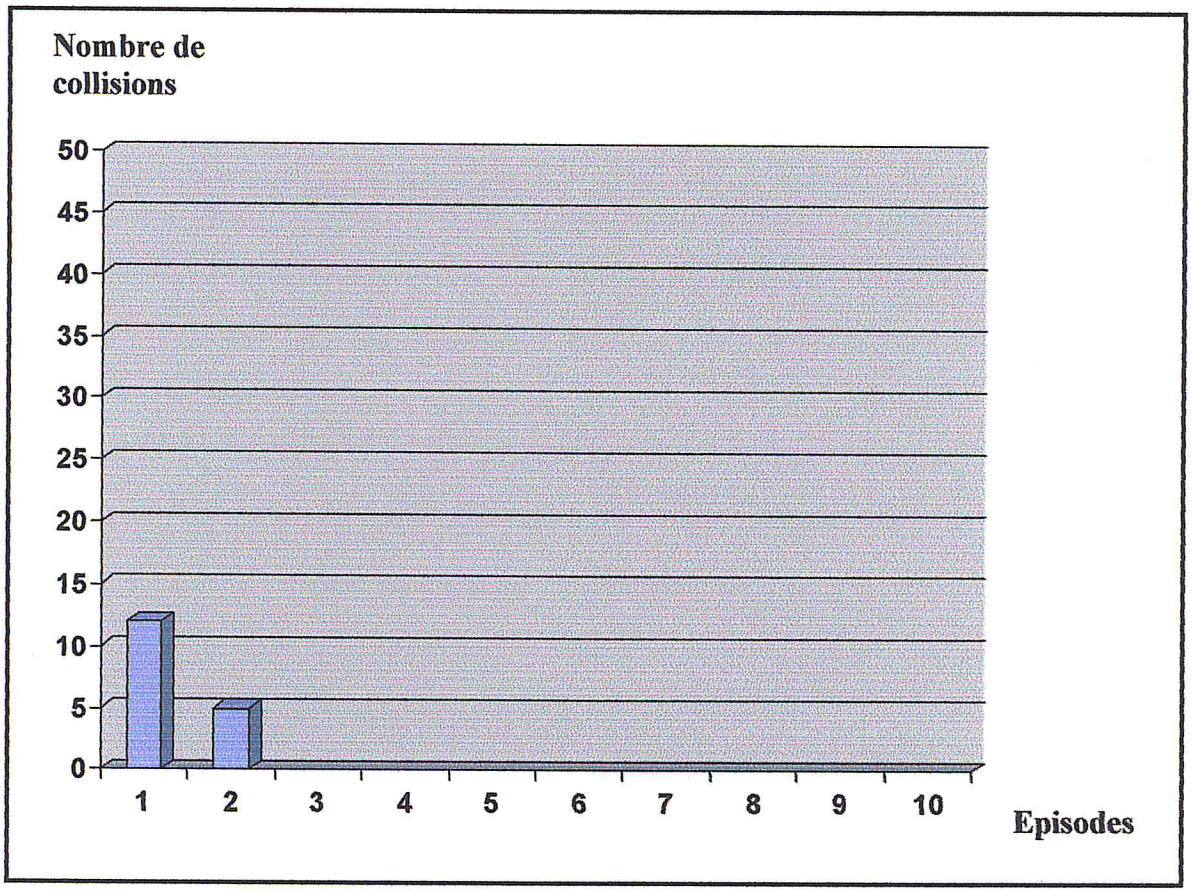


Figure V.18 : Graphe de performance (Exemple 11).

Exemple 12

Dans cet exemple, on prend un groupe de 10 robots mobiles en forme de *cercle rempli*. Cette formation se déplace dans l'environnement de la figure V.19. Ces robots apprennent à éviter les obstacles tout en gardant la formation après le quatrième épisode comme le montre le graphe de performance de la figure V.20.

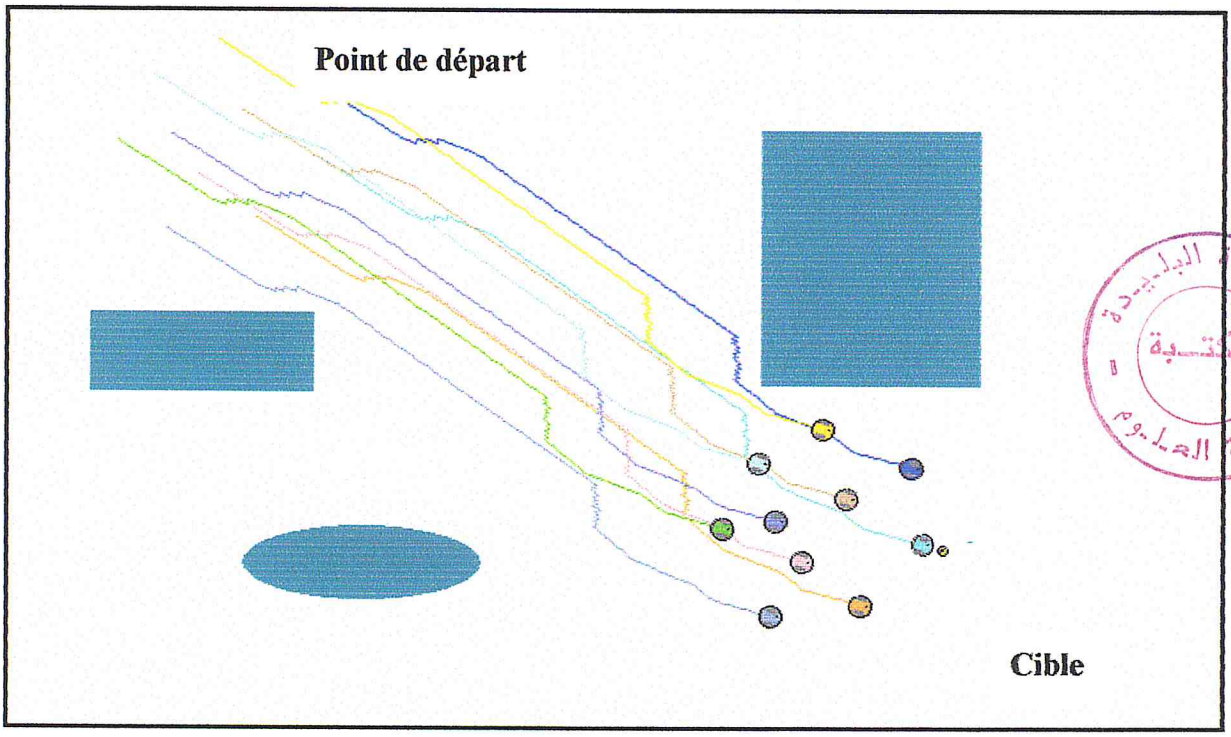


Figure V.19 : Navigation d'un groupe de robots mobiles en forme de *cercle rempli*.

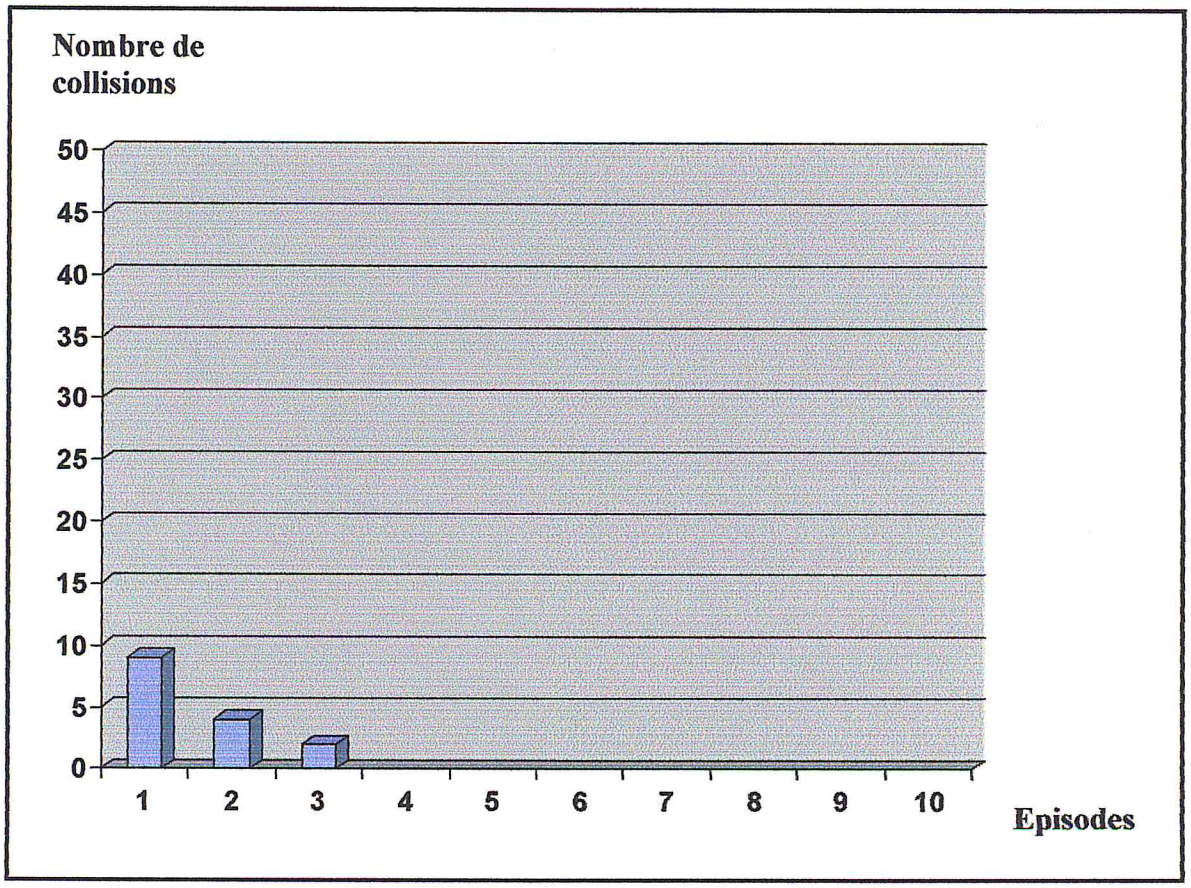


Figure V.20 : Graphe de performance (Exemple 12).

V.6 Conclusion

Les résultats de simulation ont montré que l'approche de navigation basée sur l'apprentissage par renforcement proposée réalise une navigation d'un ou de plusieurs robots dans un environnement inconnu et dynamique avec succès. D'autre part, les algorithmes de formation des figures géométriques permettent de réaliser une bonne coopération entre les robots. Ainsi, la simulation montre que la formation navigue sans entrer en collision avec les obstacles et de façon coopérative.

conclusion Générale

Conclusion Générale et Perspectives

Notre objectif est de simuler une navigation collective d'un groupe de robots mobiles basée sur l'apprentissage par renforcement dans un environnement inconnu. C'est une amélioration et continuation d'un travail qui a été réalisé dans un environnement partiellement connu. La simulation est faite par un simulateur des robots mobiles *Mobotsim*.

Dans ce mémoire, nous avons présenté des algorithmes de contrôle de robots mobiles Coopérants. Ces algorithmes permettent à des robots de fonctionner soit individuellement soit en coopération avec d'autres robots.

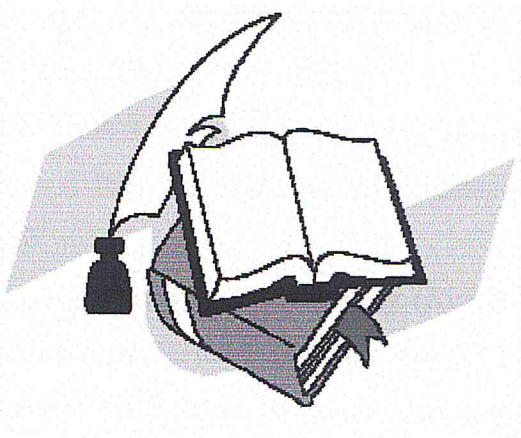
En première étape, on a implémenté des algorithmes qui permettent à un groupe de robots mobiles d'effectuer une tâche commune et assurer une bonne coopération entre eux. Cette tâche est la formation des figures géométriques tels que cercle, cercle rempli, polygone et ligne. Ainsi ces algorithmes permettent de réaliser la figure en question avec un petit ou un grand nombre de robots en jouant sur les paramètres utilisés (tel que le diamètre), mais il n'est pas toujours possible de réaliser n'importe quel type de formation. Cependant, les robots peuvent parfois être bloqués, c'est pour cela que nous avons introduit l'apprentissage par renforcement afin de pallier ce problème.

En seconde étape, après l'étape de la formation géométrique, la navigation en formation est présentée. L'objectif de l'approche développée est d'assurer à cette formation de naviguer dans un environnement inconnu tout en évitant les obstacles sur son chemin sans entrer en collision. Nous avons apporté une solution par l'utilisation de l'apprentissage par renforcement. La réalisation du déplacement de cette formation ne peut se faire si les robots ne peuvent se déplacer individuellement sans collision. C'est pour ça que nous avons développé une navigation individuelle et en groupe des robots mobiles afin de leurs permettre d'une part de réaliser la figure géométrique et d'autre part de faire naviguer cette formation avec succès dans un environnement non connu au préalable. En effet, l'approche développée assure à un groupe de robots de se déplacer en formation sans entrer en collision et de maintenir la forme de cette formation par l'utilisation de la technique du conducteur de groupe (leader). Mais, il n'est pas possible de changer la formation en cours de déplacement.

Nous avons par la suite simulé cette approche qui se traduit par la simulation de la navigation individuelle et en groupe basée sur l'apprentissage par renforcement, la formation des figures géométriques ainsi que la navigation en formation et nous avons obtenu de bons résultats dans différents environnements.

Enfin, à l'instar de tous projets de fin d'études, le travail présenté dans ce mémoire pourra être amélioré. Pour cela nous pensons qu'il serait très intéressant de reprendre ce travail, et ajouter quelques modifications sur la fonction de renforcement et les actions du robot pour lui permettre de trouver un meilleur ou le plus court chemin.

Références bibliographiques



[SAN, 03] : J. P. Sansonnet, « Introduction aux systèmes Multi agents », Présentation de cours 2003.

[SIM, 01] : O. Simonin, « Le modèle satisfaction-altruisme : coopération et résolution de conflits entre agents situés réactifs, application à la robotique », Thèse doctorat présentée à l'Université Montpellier II 2001.

[SUT, 98]: R. S. Sutton, G. B. Andrew, «Reinforcement Learning: An introduction », MIT Press, Cambidge, MA, 1998.

[SUZ, 95]: I. Suzuki et K. Sugihara, « Distributed algorithms for formation of geometric patterns with many mobile robots », Journal on Robotics Systems, Sept 1995.

[SUZ, 95a]: S. suzuki, H. Asama, « An Infra-Red Sensory System with Local Communication for Cooperation Multiple Mobile Robots », in Proc IROS 1995 IEEE.

[TOU, 99] : C. Touzet, «Apprentissage par renforcement », dans *Connexionnisme et Applications*, C. Jutten éditeur, 1999 chez Masson.

[ZUC, 00] : J. D. Zucker, « l'apprentissage par renforcement », Cours, LIP6-CNRS, 2000.