APPRENTISSAGE DE COMPORTEMENTS REACTIFS POUR DES ENSEMBLES DE ROBOTS MOBILES

Philippe Lucidarme, Alain Liégeois
LIRMM (UMR 55060)
161, rue Ada, 34392 Montpellier Cedex 5
lucidarm@lirmm.fr

Résumé – *Ce papier décrit trois méthodes permettant de réaliser l'apprentissage de comportements réactifs. Ces comportements sensori-moteurs lient directement les capteurs aux actionneurs. Nous nous intéressons ici à une tâche d'évitement d'obstacles. Dans un premier temps, un algorithme évolutionniste distribué permet de réaliser un apprentissage collectif pour un ensemble de robots mobiles homogène. Ensuite deux méthodes, basées sur le recuit simulé sont présentées en vue de l'application vers un système multi-agents hétérogène. Tous les résultats présentés ont été expérimentés en simulation et sur une véritable plate-forme de robots mobiles.*

Mots clé – multi-agents, robots mobiles, apprentissage, algorithme évolutionniste, neuro-contrôleur, recuit simulé.

1 Introduction

Habituellement, les tâches complexes qui nécessitent la coopération de plusieurs robots mobiles, pour l'agriculture, l'exploration des planètes, ou les applications industrielles et domestiques, sont pratiquement programmées à l'avance. De plus, elles mettent en jeu un niveau très cognitif : connaissance d'une carte de l'environnement, capacité à se localiser et à localiser les autres agents, algorithmes de planification de trajectoires et de mouvements, etc. [1, 2 et 3]. Seuls des travaux sur l'émergence d'un comportement collectif intelligent à partir d'individus à capacités limitées traitent une forme d'apprentissage automatique de coopération [4, 5 et 6].
Ce papier va s'intéresser principalement à l'apprentissage de comportements réactifs; l'action appliquée sur les actionneurs est directement déduite de l'information perçue par les capteurs. Pour cela, nous recherchons des méthodes qui permettent à chaque robot de construire par lui-même ses règles de renforcement à partir de l'auto-évaluation de l'efficacité de son comportement sensori-moteur. L'apprentissage sera réalisé sur une tâche d'évitement d'obstacles. Trois méthodes ont été étudiées : une version distribuée des algorithmes évolutionnistes, et deux méthodes inspirées du recuit simulé. L'objectif final de ces travaux est d'apprendre à combiner ces comportements réactifs afin de faire émerger du système multi-agents des comportements complexes de plus haut niveau.

2 Hypothèses

2.1 Description de la tâche

Afin de comparer nos résultats avec des travaux antérieurs [7], nous avons choisit une tâche d'exploration de l'environnement avec évitement d'obstacles. Ce type de comportement présente plusieurs aspects intéressants pour une tâche d'apprentissage. Il est aisé d'estimer les performances du système et par conséquent, de le récompenser. L'agent doit apprendre à anticiper certaines actions pour obtenir des récompenses retardées plus importantes que la récompense instantanée. Les résultats présentés dans ce papier ont été simulés et implémentés sur une plate-forme composée de 4 robots mobiles.

2.2 Plate-forme expérimentale

Il existe des différences notables entre les systèmes simulés et réels. Afin de valider nos résultats dans des circonstances réelles, nous avons implémenté les méthodes proposées sur des robots mobiles. Les robots utilisés ont été conçus au LIRMM [8 et 9]. La Figure 1 montre Type 1. Il s'agit d'un robot mobile conçu pour l'approche multi-agents. Leur forme est un cylindre d'environ 13 cm de diamètre et 10 cm de hauteur. 16 émetteurs et 8 récepteurs infrarouges sont régulièrement espacés sur la périphérie. La disposition des capteurs est décrite sur la Figure 2. Ces capteurs sont modulés à 40KHz pour garantir une bonne insensibilité au bruit. La portée des communications infrarouges peut être ajustée. Elle est typiquement de 0,5 m. Les capteurs infrarouges sont utilisés à la fois

pour la détection d'obstacles et pour les communications entre robots. Le robot est équipé de deux roues différentielles motorisées. Chaque moteur est équipé d'un codeur magnétique (352 impulsions par tour de roue). Le codeur est utilisé pour l'asservissement en vitesse des roues et pour la mesure de performance. La vitesse maximum a été limitée à 30 cm/s par sécurité pour le matériel. Un PC embarqué (80486 DX avec horloge à 66 MHz) assure le traitement de l'information. Les commandes et informations issues des capteurs sont transmises via le BUS ISA (BUS PC 104).

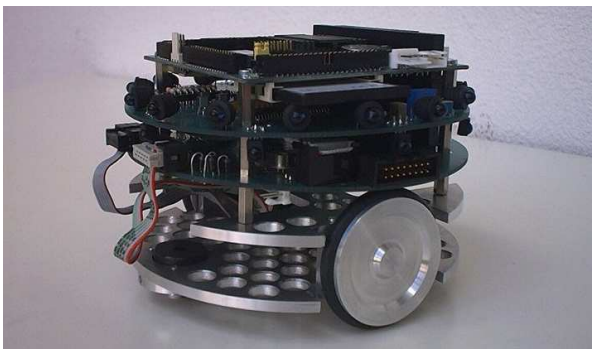

Figure 1. Le robot mobile Type 1

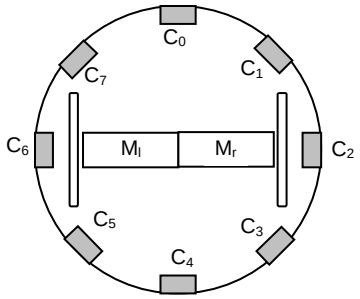

Figure 2. Disposition des capteurs et actionneurs

2.3 Estimation de la performance

Toutes les méthodes d'apprentissage nécessitent de connaître une estimation de la performance, aussi appelée récompense. Il est important de choisir judicieusement cette fonction. Dans les travaux antérieurs [10], l'auteur réalise l'apprentissage d'un comportement d'évitement d'obstacles. La récompense utilisée se divise en 3 termes :

- maximiser la vitesse du robot,
- minimiser la rotation du robot,
- minimiser le nombre de collisions.

Une telle récompense peut se montrer contraignante. En effet les deux derniers termes sont inclus dans le premier : comme les robots considérés sont à roues différentielles, il est impossible de maximiser à la fois la vitesse moyenne et la vitesse de rotation du robot. De la même façon, chaque collision réduit nécessairement la vitesse moyenne de l'agent. Dans [11], l'auteur explique "Pour notre application, la fonction de renforcement choisie permet de délivrer un signal +0.5 lorsque le robot choisi une bonne action (récompense) et –0.5 pour une mauvaise (punition)". L'intérêt de l'apprentissage est de permettre à l'agent de trouver par lui même la stratégie optimale. Ce type de récompense contraint l'agent à apprendre une stratégie proche de celle imaginée par l'opérateur. Pour ces travaux, nous avons choisi de maximiser la vitesse moyenne signée du robot. La récompense instantanée est donc la moyenne des vitesses de rotation des roues du robot.

3 Algorithme évolutionniste distribué

Dans un premier temps, nous nous sommes intéressés à l'apprentissage de comportements réactifs pour des systèmes homogènes (tous les agents sont identiques et disposent des mêmes capacités de perceptions et d'actions). Les algorithmes évolutionnistes semblaient parfaitement disposés pour ce type d'apprentissage : nous disposons d'une population d'individus, ceux ci peuvent chacun estimer leur propre performance, et il peuvent communiquer. Nous avons donc proposé une version distribuée des algorithmes évolutionnistes permettant aux agents de réaliser les croisements deux à deux.

3.1 Description et codage chromosomique d'un individu

Les entrées du système sensori-moteur sont délivrées par les émetteurs et récepteurs infrarouges entourant le robot. Elles correspondent aux 5 états donnés sur le Tableau 1.

Etat 1	Pas d'obstacle
Etat 2	Obstacle à gauche
Etat 3	Obstacle à droite
Etat 4	Obstacle en face
Etat 5	Blocage

Tableau 1. Les différents états du système

A l'autre extrémité (la sortie du système de contrôle-commande) l'individu a les comportements élémentaires donnés sur le Tableau 2.

Comportement 1	Avancer tout droit
Comportement 2	Tourner à droite
Comportement 3	Tourner à gauche
Comportement 4	Reculer

Tableau 2. L'ensemble des réactions

Le chromosome d'un robot est la concaténation (une chaîne) de N mots dans un sous-espace de l'espace binaire {0,1}M. N est le nombre d'entrées, et M le nombre de sorties. Bien entendu ici, un seul bit vaut 1 dans chaque mot. Un exemple est donné sur le Tableau 3.

0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0

Tableau 3. Un exemple de chaîne chromosomique

La population de robots va évoluer en suivant l'évolution des chromosomes sous l'action des opérateurs génétiques.

3.2 L'algorithme d'apprentissage

Initialisation. Au début, les chaînes sont remplies de mots choisis aléatoirement.
L'indice de performance individuel. Comme nous recherchons ici un apprentissage non supervisé, chaque robot doit s'auto-évaluer, mais il n'est pas conscient de la performance collective. Chaque individu numéro i calcule sa performance d'après (1):

$$R_{N(i)}(i) = (1-\alpha(i))R_{N(i-1)}(i) + \alpha(i)F_{N(i)}(i) \quad (1)$$

où

- $N(i)$ est le nombre de périodes élémentaires de la chaîne sensori-motrice depuis le début de l'auto-évaluation,
- $R_{N(i)}(i)$ est la récompense estimée à cette date,
- $F_{N(i)}(i)$ est la récompense instantanée à cette date.

Ici, la récompense instantanée est la distance parcourue par unité de temps sensori-moteur sans reculer ni être bloqué. Ainsi, la récompense intégrée est la distance moyenne parcourue depuis l'initialisation ou depuis le dernier croisement ou la dernière mutation.
Croisement. La plupart des chercheurs supposent une communication globale entre les individus, ce qui est parfois impossible et de toute façon prend du temps et nécessite un protocole compliqué et inutile puisque seulement deux individus peuvent être sélectionnés en même temps. Nous avons préféré une communication simple et locale. Ainsi, seuls deux robots qui se rencontrent se communiquent leur code génétique et leur indice de performance. Ils procèdent au croisement à condition qu'un temps suffisamment long se soit écoulé depuis la dernière mutation ou le dernier croisement pour chacun, afin que les valeurs de performances soient fiables. On démontre qu'après plusieurs croisements, la performance moyenne n'a pas diminuée. Toutefois les mutations sont indispensables pour garantir la convergence vers un optimum absolu.
Mutation. La stratégie suivante est adoptée : le robot n'a pas muté récemment et sa performance est faible. Nous faisons varier la probabilité de mutation en fonction de la performance : des fonctions sigmoïdes et en escalier sont essayées.

3.3 Résultats

La distance de communication est celle de nos robots mobiles expérimentaux. L'intervalle minimum de temps entre deux mutations ou croisements est de l'ordre de cinquante pas élémentaires, ce qui laisse le temps à deux robots qui viennent de se croiser de s'éloigner suffisamment pour ne pas recommencer juste après.

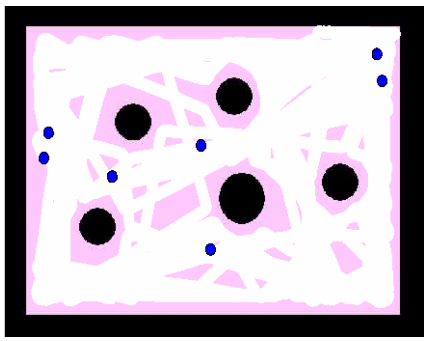

Figure 3. Surface explorée à la fin de l'apprentissage

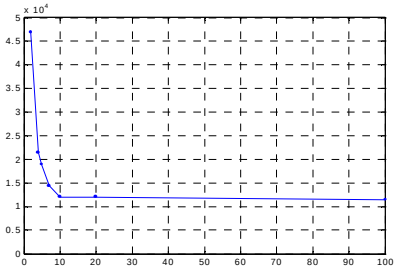

Figure 4. Temps de convergence en fonction du nombre de robots

La Figure 3 montre le résultat d'une exploration (la surface en blanc) à la fin d'un apprentissage en simulation (tous les robots ont la bonne association perception-action). Les petits cercles sont les robots, et les gros sont les obstacles. Nous avons étudié la durée de l'apprentissage en fonction du nombre de robots (Figure 4). Elle décroît jusqu'à 10 robots puis ne change plus. On n'a donc pas intérêt à multiplier le nombre d'individus. Dans l'application étudiée ici, la convergence est atteinte en quelques minutes avec les robots mobiles. Toutefois, la stratégie est décomposée en un nombre d'états et d'actions finis. Pour obtenir des comportements plus fins, et des réactions proportionnelles aux actions, il serait intéressant de

considérer le même problème dans le domaine continu. Il faudrait alors coder des valeurs réelles dans la chaîne chromosomique. Ces valeurs correspondraient aux paramètres du contrôleur. Les temps d'apprentissage augmentent avec la complexité du problème, et par conséquent avec la longueur de la chaîne. Utiliser les algorithmes génétiques pour apprendre des comportements dans le domaine continu risque d'augmenter les temps d'apprentissage de façon conséquente. De plus, la méthode proposée est difficilement applicable dans le cas de systèmes hétérogènes: si les agents n'ont pas la même structure, la séquence chromosomique n'aura pas le même sens d'un agent à l'autre, et les croisements seront impossibles. Il n'est pas envisageable d'utiliser un apprentissage collectif dans le cas des systèmes hétérogènes, nous nous sommes donc intéressés à l'apprentissage individuel de comportements réactifs dans le domaine continu.

4 Recuit simulé

L'objectif est de réaliser l'apprentissage de comportements réactifs dans le domaine continu. Les agents ne disposent pas de représentations complexes ou de carte de l'environnement, ni de leur structure interne. Les agents doivent être capables de modifier leur contrôleur en cas de panne ou de changement dans l'environnement. L'idée est de ne pas nécessairement rechercher le maximum global afin de privilégier la rapidité de convergence et l'adaptabilité.

4.1 Contrôleur neuronal

Le contrôleur de l'agent est un réseau de neurones représenté sur la Figure 5. Ce réseau ne dispose d'aucune couche cachée. Ainsi le comportement appris est une combinaison linéaire de chacune des entrées. Le réseau est alimenté en entrée par chacun des 8 capteurs (C_0 à C_7) plus une constante. Les sorties du réseau actionnent directement les moteurs du robot (M_l et M_r). Le réseau comporte deux perceptrons et 18 poids synaptiques correspondant chacun à une liaison entre une entrée et un perceptron.

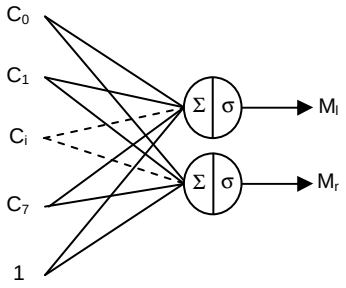

Figure 5. Le contrôleur neuronal

4.2 L'algorithme

Initialisation

$R_{max} \leftarrow 0$
Initialiser chaque poids (w_j) avec une petite valeur.
$T° \leftarrow F_t(0)$

Boucle principale

```
Tant que (T° > petite valeur)
{
Appliquer la stratégie courante et
mesurer sa performance R_N(i).
Si (R_max<R_N(i))
{
     R_max=R_N(i)
     Pour chaque poids: W_max(j) ← w_j
     }
     T° ← F_t(cycle)
}
Pour chaque poids (w_i), choisir
aléatoirement une nouvelle valeur
centrée sur W_max(i) avec une répartition
proportionnelle à T°.
}
```

Figure 6. Algorithme utilisé pour entraîner le réseau

Dans un premier temps, nous avons réalisé l'étude en utilisant la version traditionnelle du recuit simulé. La Figure 6 présente l'algorithme. Afin d'éviter que l'agent ne reste bloqué après avoir heurté un obstacle pendant la phase d'apprentissage, un comportement pré-programmé a été ajouté afin de pouvoir continuer l'apprentissage. Quand une collision est détectée (les roues restent bloquées pendant un temps pré-défini), l'agent recule. Pendant cette manœuvre, la récompense est toujours calculée, de ce fait, entrer en collision avec un obstacle ne peut jamais être rentable.

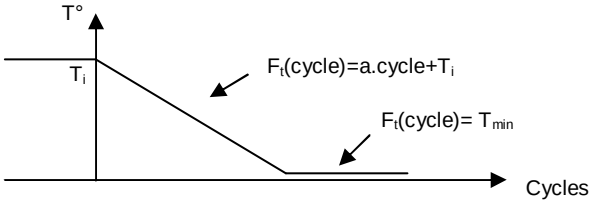

Figure 7. Evolution de la température en fonction du temps

La fonction de température utilisée est présentée sur la Figure 7. Nous avons arbitrairement choisi une fonction linéaire, toutefois d'autre fonctions seront expérimentées. Pour permettre à l'algorithme de glisser vers le maximum global, la température n'est jamais égale à 0. La valeur minimale T_{min} garantie de ne jamais figer les paramètres du système et de converger vers l'extremum, qu'il soit optimal ou global.

4.3 Résultats

4.3.1 Simulations

De nombreuses simulations ont été réalisées afin d'étudier l'influence des paramètres. La première constatation est que la méthode converge toujours vers la stratégie optimale à condition que la température décroisse lentement et que les temps d'évaluation soient suffisamment longs pour garantir une estimation correcte de la performance. La Figure 8 présente l'évolution de la meilleure stratégie connue au cours du temps. Dans un premier temps, le robot tourne sur place, garantissant ainsi de ne pas rencontrer d'obstacles. La trajectoire du robot décrit des cercles de plus en plus grands, pour finir par parcourir des trajectoires rectilignes tout en évitant les obstacles. Le robot dispose d'un capteur central C_0, lorsqu'un obstacle est détecté par celui, l'agent dispose de deux stratégies: tourner à droite ou à gauche. Il existe donc deux stratégies optimales dans notre système.

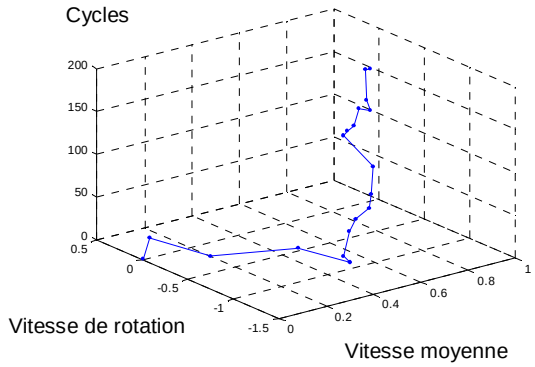

Figure 8. Evolution de la meilleure stratégie connue

4.3.2 Résultats expérimentaux

La Figure 9.a montre l'évolution des poids du réseau. La convergence est atteinte et la récompense est maximisée (Figure 9.b). Toutefois, l'analyse des résultats montre que le système ne converge pas toujours vers la solution optimale. Dans les conditions réelles, le système est perturbé par le bruit, et on constate que le même comportement peut donner des performances différentes selon la configuration initiale de l'agent. Pour contrer ce problème, il faudrait augmenter la durée d'une période d'évaluation afin de diminuer l'écart type autour de la valeur moyenne de la performance, mais cela augmenterait les temps d'apprentissage.

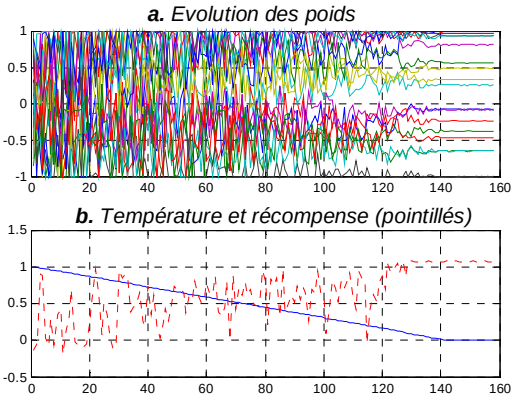

Figure 9. Résultats d'une expérimentation

5 Véritable recuit simulé

5.1 L'algorithme

Initialisation

$R_{max} \leftarrow 0$
Initialiser chaque poids (w_j) avec une petite valeur.
$T° \leftarrow F_t(0)$

Boucle principale

```
Tant que (vrai)
{
Appliquer la stratégie courante, et
mesurer la performance R_N(i).
Si (R_max<R_N(i))
{
      R_max=R_N(i)
      Pour chaque poids w_i : W_max(i) ← w_i
      }
      T° ← F_t(R_max)
      Diminuer R_max
Pour chaque poids (w_i), choisir
aléatoirement une nouvelle valeur
centrée sur W_max(i) avec une répartition
proportionnelle à T°.
}
```

Figure 10. Algorithme adaptatif utilisé pour entraîner le réseau

Avec la méthode présentée précédemment, aucune adaptation n'est envisageable, lorsque la température atteint T_{min}, l'apprentissage est verrouillé et le contrôleur n'évolue plus. Pour permettre au système de s'adapter aux pannes et aux changements, nous allons autoriser la remonté de température comme dans le véritable recuit simulé. Nous travaillons sans modèle de l'environnement, par conséquent, le seul moyen de détecter des changements est la récompense obtenue. Si un changement survient et que celui ci influence la performance de l'agent, alors la récompense va diminuer. Ici, la température est une fonction de la meilleure performance connue (R_{max}) : lorsque la performance est grande, la température sera faible et inversement. Le meilleur comportement

connu (R_{max}) est décrémenté à chaque cycle. De cette façon, si les performances du système sont constantes, R_{max} est régulièrement mis à jour. Si au contraire, les performances sont irrégulières, R_{max} va diminuer, autorisant la température à croître et le système à repartir dans une phase d'apprentissage. La Figure 10 décrit l'algorithme utilisé.

5.2 Résultats

5.2.1 Simulations

Les résultats de simulations et expérimentaux étant similaires, nous ne présenterons ici que les résultats obtenus avec le robot mobile.

5.2.2 Résultats expérimentaux

La Figure 11.a montre l'évolution des poids du système en fonction du temps. L'apprentissage peut être découpé en différentes phases : de 0 à 15 cycles, le système explore l'espace d'état de façon aléatoire jusqu'à trouver une solution acceptable. De 15 à 25 cycles, les poids convergent vers le maximum local. Ensuite la récompense étant maximale (Figure 11.b), la température décroît pour verrouiller le système autour de cette stratégie. Afin de tester les facultés d'adaptation du système, nous avons simulé une panne sur un des capteurs (C_1). Aux environs du 37ème cycle, on discerne une chute de la récompense (Figure 11.b.), qui a pour conséquence la remonté de la température. Le système compense la perte du capteur en renforçant l'influence des capteurs voisins (C_0 et C_2). D'autre pannes plus graves ont été testées (la perte de plusieurs capteurs). Si la panne est trop importante, l'agent ne peut obtenir de récompense suffisante pour garantir la stabilité du système, et il ne sort jamais de la phase d'apprentissage.

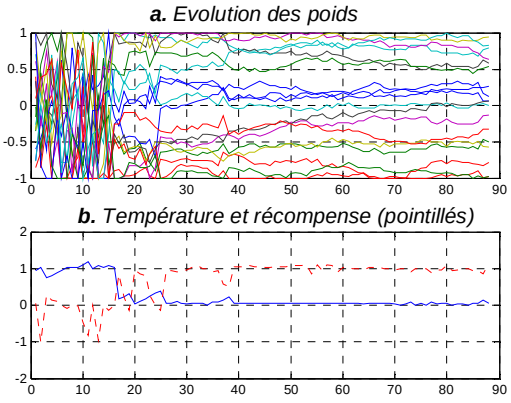

Figure 11. Résultats d'une expérimentation

6 Conclusion

Ces travaux se placent dans le contexte des systèmes multi-agents. Nous avons étudié trois méthodes d'apprentissage de comportements réactifs. La première méthode proposée, basée sur les algorithmes évolutionnistes, permet de réaliser l'apprentissage sur un système multi-robots, et de profiter du parallélisme du système sans les contraintes imposées par la supervision. Cette algorithme est difficilement applicable aux systèmes hétérogènes, et les temps de convergence croissent avec la taille de l'espace d'état. Deux méthodes basées sur le recuit simulé ont également été décrites. La première permet d'atteindre le maximum global dans un environnement faiblement perturbé et les temps de convergence sont connus par avance (dépendants de la fonction de température). En revanche, cette méthode n'est pas adaptative et ne permet pas de modifier le contrôleur si des changements ou des pannes surviennent. La dernière méthode, ne donne aucune garantie de trouver la solution optimale, mais les expérimentations montrent qu'une solution acceptable est rapidement trouvée et l'algorithme permet au système de s'adapter aux changements. Les futurs travaux vont s'orienter vers l'apprentissage de comportements réactifs (ralliement de cible, saisie d'objet, etc.). Nous pourrons ainsi disposer de bibliothèques de comportements et les séquencer afin de réaliser des tâches de plus haut niveau et plus complexes.

Bibliographie

[1] O. Pinchard, A. Liégeois and T. Emmanuel, 1995, A Genetic Algorithm for Outdoor Robot Path Planning, 4th Int. Conf. On Intelligent Robotic Systems, Karsruhe, IOS Press, p. 413-419.

[2] T. Balch and R. Arkin, 1995, Motor schema-based formation control of multiagent robot teams, Int. Conf. On Multiagent Systems, p. 10-16.

[3] S. Botelho et R. Alami, 1999, Multirobot Cooperation in the Martha Project, ICRA'99, p. 1234-1239.

[4] M. Mataric, 1997, Behavior-based Control Examples from Navigation, Learning, and Group Behavior, J. of Experimental and Theoretical A. I., 9(2-3), p. 323-336.

[5] L.E. Parker, 1998, ALLIANCE : An ARCHITECTURE for Fault-tolerant Multirobot Cooperation, IEEE Trans on Robotics and Automation, Vol. 14, N° 2, p. 220-240.

[6] O. Simonin and J. Ferber, 2000, Modeling Self Satisfaction and Altruism to Handle Action and Reactive Cooperation, SAB'00 Proceedings Supplement, p. 314-323.

[7] D. Floreano and F. Mondada, 1994, Automatic Creation of an Autonomous Agent : Genetic Evolution of a Neural-Network Driven Mobile Robot, SAB-3, Brighton, p. 421-430.

[8] P. Lucidarme, P. Rongier et A. Liégeois, 2001, Implementation and evaluation of a Reactive Multi-Robot System, AIM'01, Como, p. 165-170.

[9] P. Lucidarme, O. Simonin et A. Liégeois, 2002, Implementation and evaluation of a Satisfaction/Altruis-Based Architecture for Multi-Robot Systems, ICRA'02, Washington, May 2002.

[10] D. Floreano and J. Urzelai, 2000, Evolutionary Robots with On-line Self-Organization, Neural Networks, Vol . 13, p. 397-404.

[11] T. Madani, A. Benallegue et N.K. M'Sirdi, 2002, Apprentissage par renforcement pour la navigation d'un robot mobile dans des environnements inconnus, JJCR'16, Lyon.