Architectures de contrôle comportementales et réactives pour la coopération d'un groupe de robots mobiles

Lounis Adouane

▶ **To cite this version:**

Lounis Adouane. Architectures de contrôle comportementales et réactives pour la coopération d'un groupe de robots mobiles. Automatique / Robotique. Université de Franche-Comté, 2005. Français. NNT : . tel-00128160v1

HAL Id: tel-00128160

https://theses.hal.science/tel-00128160v1

Submitted on 30 Jan 2007 (v1), last revised 11 May 2010 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° ordre : 1071

Année 2005

THÈSE

préparée au

Laboratoire d'Automatique de Besançon (UMR CNRS 6596)

et présentée à

L'U.F.R. DES SCIENCES ET TECHNIQUES DE L'UNIVERSITÉ DE FRANCHE-COMTÉ

pour obtenir le

GRADE DE DOCTEUR D'UNIVERSITÉ

spécialité **Automatique**

Architectures de Contrôle Comportementales et Réactives pour la Coopération d'un Groupe de Robots Mobiles

par

Lounis Adouane

(DEA en Automatique et Informatique Appliquée)

Soutenue le 11 avril 2005 devant le jury composé de :

Rapporteurs	**Rachid Alami** (Directeur de recherche CNRS, LAAS–Toulouse III)
	René Zapata (Professeur, LIRMM-Montpellier II)
Examinateurs	**Alain Bourjault** (Professeur, ENSMM–Besançon)
	Alexis Drogoul (Professeur, LIP6–Paris VI)
(Président du jury)	**Wisama Khalil** (Professeur, IRCCYN–École Centrale de Nantes)
Directeur de thèse	**Nadine Lefort-Piat** (Professeur, ENSMM–Besançon)

Dédicaces

A ma mère Hayat, pour son amour et son courage
A mon père Larbi, pour m'avoir inculquer les valeurs auxquelles je tiens tant
A ma soeur Sabine et à mon frère Massinissa, vous êtes dans mon coeur
A mon ami, et non moins frère Djamel
A ma famille et aux gens que j'aime.

Remerciements

Ces travaux de thèse ont été réalisés au Laboratoire d'Automatique de Besançon LAB (UMR CNRS 6596 - ENSMM/UFC) au sein du groupe de recherche "Microrobotique et Micromécatronique". Ils n'auraient jamais été concrétisés sans l'intervention de personnes que je tiens à remercier ici.

En premier lieu, je tiens à exprimer ma profonde gratitude à Monsieur Alain Bourjault, Professeur à l'ENSMM et directeur du LAB pour m'avoir accueilli au sein de cette structure de recherche, et m'avoir permis d'enrichir ma réflexion sur différents aspects de la recherche, de ces moyens et de ces enjeux. Je le remercie aussi d'avoir accepté de juger ce travail.

Ces travaux de recherche n'ont pas pu être réalisés sans la confiance ô combien importante dont m'a fait preuve mon directeur de thèse Madame Nadine Le Fort-Piat, Professeur à l'ENSMM. Elle a su diriger cette thèse avec un dévouement et une gentillesse des plus appréciables. Je la remercie d'autant plus pour toutes les libertés d'action dont j'ai pu bénéficier tout au long du déroulement de mes travaux de thèse.

Mes remerciements vont aussi à Monsieur Rachid Alami, Directeur de Recherche CNRS au LAAS et à Monsieur René Zapata, Professeur a l'Université de Montpellier II, pour avoir accepté de rapporter sur cette thèse.

Je tiens aussi à exprimer mes remerciements et sentiments les plus respectueux à Monsieur Wisama Khalil, Professeur à l'Ecole Centrale de Nantes, pour m'avoir fait l'honneur de présider le jury de soutenance.

Je remercie sincèrement aussi Monsieur Alexis Drogoul, Professeur à l'Université de Paris VI, d'avoir accepté de juger ce travail.

Ma profonde gratitude va vers toutes les personnes qui ont su de part leur soutien et discussions, impulser en moi, des idées et des pistes qui ont permis d'influencer d'une manière ou une autre mes travaux. Parmi eux et au risque certain d'en oublier, je citerai Gilles Caprari, Noureddine Zerhouni, Nicolas Chaillet, Brigitte Morello, Yassine Haddab, Claude Jacquemard, Patrick Rougeot, Youssef Harrath, Bruno Barbier, Cédric Adda, Joël Agnus, André Janex (Dédé), Guillaume Laurent, Mickael Gauthier, Sounkalo Dembele. Merci encore une fois à toi Patrick et à toi Yassine pour votre aide inestimable durant la phase critique de la rédaction.

Je garde un souvenir agréable de ma vie quotidienne au sein du laboratoire, pour cela je tiens à vous exprimer à vous tous et sans exception mes vifs remerciements et amitié que j'espère inaltérable.

Table des matières

Glossaire et notations **xvii**

Introduction générale **1**

I Problématique de Recherche et État de l'Art **5**

1 La coopération d'entités autonomes **7**

1.1 L'entité élémentaire participant à la coopération 8

1.1.1 Qu'est-ce qu'un robot mobile ? 8

1.1.2 Importance des caractéristiques structurelles d'un robot (est-il capable de réaliser tout ce qu'on lui demande ?) 9

1.1.3 Parlons un peu d'autonomie 11

1.1.4 Le contrôle des robots mobiles 12

1.2 Et la coopération dans tout ça ! 13

1.2.1 La robotique collective 14

1.2.1.1 Qu'est-ce que la coopération de robots ? 16

1.2.1.2 Projets effectifs et tâches génériques pour la robotique collective 17

1.2.1.2.1 Transport et manipulation coopérative d'objets . 17

1.2.1.2.2 Mouvement en formation 19

1.2.1.2.3 Fourragement 19

1.2.1.3 Communiquer pour bien coopérer ! 21

1.2.1.3.1 Communication de haut niveau 21

1.2.1.3.2 Communication de bas niveau 23

1.2.1.3.3 Communication indirecte 23

1.2.2 Différentes approches méthodologiques pour faire coopérer un ensemble de robots 24

1.2.2.1 Approche des Sciences du Vivant – Éthologie des Animaux Sociaux . . . 25
1.2.2.1.1 Liens existants des sciences de l'ingénieurs vers l'éthologie . . . 26
1.2.2.1.2 Liens existants de l'éthologie vers les sciences de l'ingénieur . . . 27
1.2.2.2 Approche Informatique – Système Multi Agents . . . 29
1.2.2.3 Approche Automatique – Robotique Collective . . . 31
1.3 Conclusion . . . 34

2 Catégorisation des architectures de contrôle 37
2.1 Architecture de contrôle centralisée versus distribuée . . . 38
2.1.1 Architecture de contrôle centralisée . . . 38
2.1.2 Architecture de contrôle décentralisée-distribuée . . . 38
2.1.3 Faut-il centraliser ou distribuer le contrôle ? . . . 40
2.2 Architectures de contrôle cognitives versus réactives . . . 41
2.2.1 Les architectures de contrôle cognitives . . . 42
2.2.2 Les architectures de contrôle réactives . . . 44
2.2.2.1 Les architectures comportementales réactives . . . 45
2.2.2.2 Émergence de comportements . . . 46
2.2.3 Alors architectures cognitives ou bien réactives ? . . . 48
2.3 Conclusion . . . 50

3 Les architectures de contrôle comportementales 51
3.1 Mécanismes intervenants dans une architecture comportementale . . . 52
3.2 Sélection d'action . . . 54
3.2.1 Subsomption . . . 54
3.2.2 Événements discrets . . . 56
3.2.3 Dynamique . . . 57
3.2.4 EthoModélisation . . . 60
3.2.5 ALLIANCE . . . 63
3.3 Fusion d'actions . . . 66
3.3.1 Schémas moteurs . . . 66
3.3.2 Logique floue . . . 70
3.4 Conclusion . . . 73

II Contrôle d'un Groupe de Robots Minimalistes **75**

4 Processus de Sélection d'Action Hiérarchique "PSAH" **77**
4.1 Processus de Sélection d'Action Hiérarchique "PSAH" 78
4.1.1 Comportement élémentaire 78
4.1.2 Modèle cinématique 79
4.1.3 Fonctionnement du "PSAH" entre comportements 82
4.1.4 Mise en oeuvre du PSAH sur une tâche de navigation 84
4.2 La tâche générique de poussée d'objets coopérative 88
4.3 Architecture de contrôle proposée 89
4.3.1 Les comportements élémentaires 91
4.3.1.1 Comportement d'exploration 92
4.3.1.2 Comportement d'attraction à la boite 92
4.3.1.3 Comportement d'évitement d'obstacles 93
4.3.1.4 Comportement de repositionnement 93
4.3.1.5 Comportement de poussée de boite 96
4.3.1.6 Comportement d'alignement 96
4.3.1.7 Comportements altruistes 98
4.3.1.7.1 Comportement d'émission de signaux altruistes . 98
4.3.1.7.2 Comportement de réponse aux signaux altruistes 100
4.4 Étude en simulation de l'architecture de contrôle 100
4.4.1 Paramètres des simulations 100
4.4.1.1 Conditions initiales des simulations 102
4.4.2 Résultats obtenus et discussions 102
4.5 Conclusion 105

5 Processus de Sélection d'Action Hiérarchique et Hybride **107**
5.1 Processus de Sélection d'Action Hiérarchique et Hybride "PSAHH" 108
5.2 Le PSAHH appliqué à la tâche coopérative de poussée d'objets 112
5.2.1 Intérêt de l'utilisation des blocs de fusion 112
5.2.2 Étude en simulation de l'architecture de contrôle 115
5.2.2.1 Utilisation ou non des comportements altruistes 116
5.2.2.2 Comparaison qualitative entre le PSAH et le PSAHH . . 118
5.2.2.3 Utilisation du PSAHH pour contrôler un grand nombre de robots 120
5.3 Conclusion 121

6 Optimisation paramétrique de l'architecture de contrôle 123
6.1 La bio-inspiration pour optimiser les architectures de contrôle 124
6.1.1 Les réseaux de neurones artificiels 125
6.1.2 L'apprentissage par renforcement 126
6.1.3 Les algorithmes génétiques 128
6.1.3.1 Mécanismes utilisés par les algorithmes génétiques 129
6.1.3.2 Le codage 130
6.1.3.3 La population initiale 131
6.1.3.4 La fonction d'évaluation "*fitness*" 131
6.1.3.5 Les opérateurs génétiques 131
6.1.3.5.1 Sélection 132
6.1.3.5.2 Croisement 133
6.1.3.5.3 Mutation 134
6.1.3.6 Les algorithmes génétiques appliqués au contrôle 135
6.2 Optimisation paramétrique de l'architecture de contrôle proposée 137
6.2.1 Méthodologie proposée pour l'optimisation 137
6.2.1.1 L'environnement de test 140
6.2.1.2 Les opérateurs génétiques utilisés 141
6.2.1.2.1 Opérateur de sélection 141
6.2.1.2.2 Opérateur de croisement 141
6.2.1.2.3 Opérateur de mutation 141
6.2.1.3 Valeurs des paramètres de l'algorithme génétique 142
6.2.1.4 Résultats de l'optimisation paramétrique 142
6.2.2 Évaluation de la validité de l'optimisation 143
6.3 La bibliothèque pour algorithmes génétiques *BibAG* 144
6.4 Conclusion 146

III Environnement de Simulation et Plate-forme Expérimentale 149

7 Simulateur *MiRoCo* 151
7.1 Introduction 152
7.2 Les environnements de simulation multi-robots 154
7.3 Développement de *MiRoCo* 159
7.3.1 Conception *orientée objet* de *MiRoCo* 160
7.3.2 Classes et organisation des simulations sous *MiRoCo* 160
7.3.2.1 Agent robot 161

7.3.2.2 Agents boite, obstacle, cible . . . 162
7.3.2.3 Classe superviseur . . . 163
7.3.2.3.1 Moteur physique . . . 163
7.3.2.3.2 Modèle de déplacement de l'objet à pousser . . . 167
7.3.2.3.3 Modèle d'échange de signaux entre agents . . . 168
7.3.2.4 Classe Simulation . . . 171
7.3.2.5 Bibliothèques externes utilisées . . . 172
7.3.2.5.1 Environnement de programmation . . . 172
7.3.2.5.2 Affichage . . . 173
7.3.2.5.3 Détection de collisions . . . 173
7.3.2.6 Déroulement d'une simulation complète pour le cas de la *TCPO* . . . 174
7.4 Conclusion . . . 174

8 Phase expérimentale **175**
8.1 Plate-forme expérimentale . . . 176
8.1.1 L'environnement . . . 176
8.1.2 L'aspect vision . . . 176
8.1.3 Les mini-robots ALICE . . . 178
8.2 Implémentation effective des architectures de contrôle proposées . . . 180
8.2.1 Aspects matériels . . . 180
8.2.1.1 Dispositif pour la mesure d'angles . . . 182
8.2.1.2 Dispositif pour les comportements altruistes . . . 188
8.2.1.3 Dispositif pour la détection de l'objet à pousser . . . 190
8.2.2 Aspects logiciels . . . 191
8.2.3 Expérimentations . . . 191
8.3 Conclusion . . . 191

Conclusion générale et perspectives **193**

Annexes **201**

A Principales définitions liées à un système multi-robots **201**

B Comportements élémentaires continus **205**

Bibliographie **209**

Table des figures

1.1 Capteurs, actionneurs, structures et intelligence interagissent constamment dans les systèmes vivants. 9
1.2 Robot mobile "Sojourner" utilisé pour la mission Pahtfinder de la NASA . 12
1.3 Schéma d'un asservissement conventionnel 13
1.4 Tâches nécessitant la coopération de plusieurs robots 14
1.5 Schéma d'asservissement d'un système multi-robots 16
1.6 Exemple de manipulation coopérative d'objet 18
1.7 Formations pour quatre robots mobiles (de gauche à droite : diamant, cale, ligne, colonne). Les points noirs représentant les obstacles (Balch & Arkin 1995*b*). 19
1.8 Tâche du tri d'objets 20
1.9 Scénario possible d'une communication de haut niveau 22
1.10 Exemple possible d'une communication de bas niveau 23
1.11 Méthodologies traitant de la coopération d'entités autonomes 24
1.12 Tâche coopérative de transport d'objets par un groupe de fourmis 26
1.13 Coopération de robots au sens automatique-robotique 32
1.14 Algorithme pour trouver la configuration optimale pour saisir un objet . . 33

2.1 Tâches coopératives contrôlées d'une manière centralisée 39
2.2 Degré d'intelligence exigé pour les robots de chaque école 42
2.3 Situation d'interblocage 43
2.4 Importance de la portée de la planification 43
2.5 Machine de Braitenberg 44
2.6 Décompositions possibles d'une architecture de contrôle (Brooks 1986). . . 46
2.7 Contrôle réactif versus cognitive 49

3.1 Mécanismes intervenants dans les architectures de contrôle comportementales 53

3.2 Arbre des différents mécanismes de coordination entre comportements . . 53
3.3 Processus d'interaction entre comportements 54
3.4 Types de hiérarchie appliqués au cas des architectures à base de subsomption 55
3.5 Sélection d'actions sous forme d'automates à états finis 56
3.6 Capteurs spatialement orthogonaux (Kube & Zhang 1997) 57
3.7 Sélection d'action dynamique (Maes 1989) 59
3.8 Un agent est constitué d'un ensemble de tâches, chaque tâche étant déclenchée par un stimulus particulier, qu'il soit interne ou externe. Une tâche est composée d'une séquence de primitives, représentées ici par des figures géométriques, qu'elle exécute lorsqu'elle est déclenchée (Drogoul 1993). 61
3.9 ALLIANCE, l'architecture de contrôle proposée par Lynne Parker 64
3.10 Relations existantes dans un schéma "perception-action" (Arkin 1998) . . 67
3.11 Les figures (a) et (b) représentent respectivement les champs de forces 2D et 3D, générés par les schémas moteurs : *aller-vers-l'objectif* et *éviter-un-obstacle-statique*. La figure (c) correspond à l'addition de schémas moteurs pour la navigation (atteindre l'objectif tout en évitant les obstacles) d'un robot mobile. 68
3.12 Contrôleur flou 71
3.13 Comportement d'*évitement d'obstacles* contrôlé par logique floue 72

4.1 Comportement élémentaire 79
4.2 Mini-robot ALICE 79
4.3 Variables caractéristiques d'un robot de type char dans le plan 80
4.4 Forme générale de l'espace atteignable "P_a" par le robot ALICE en un pas d'échantillonnage. 81
4.5 Exemple d'une décomposition de l'espace atteignable P_a du robot en six sous-espaces atteignables distincts $P_i|_{i=0..5}$. 82
4.6 Processus de Sélection d'Action Hiérarchique "PSAH" entre comportements 84
4.7 Architecture de contrôle simple en utilisant deux comportements élémentaires 85
4.8 Positions atteignables et positions refuges du comportement d'*évitement d'obstacles*. 85
4.9 Trajectoires du robot obtenues pour différents types d'environnement tests et pour différents Ω. 87
4.10 Tâche coopérative de poussée d'objets "TCPO" 89

4.11 Architecture de contrôle à base de PSAH appliquée à la tâche coopérative de poussée d'objets "TCPO". 90
4.12 Espace atteignable discret 91
4.13 Informations nécessaires pour exécuter le comportement de poussée d'objets 97
4.14 Comportements altruistes sous forme de signaux attractifs et répulsifs . . 99
4.15 Évolution du nombre d'itérations nécessaire pour réaliser la tâche coopérative de poussée d'objets en utilisant l'architecture de contrôle à base de PSAH. 101
4.16 Positions initiales aléatoires de 20 robots autour de la boite à pousser . . . 102
4.17 Gain en temps induit par les comportements altruistes 103

5.1 Bloc de fusion de commandes 108
5.2 Architecture de contrôle intégrant le PSAHH au cas de la TCPO 113
5.3 Effet observé lorsque le comportement d'*évitement d'obstacles* n'est pas activé quand les robots poussent la boite. 114
5.4 Influence de la valeur du gain "g_{32}" attribuée au comportement d'*évitement d'obstacles* pour le cas du bloc de fusion Σ_3. 115
5.5 Évolution du nombre d'itérations nécessaire pour réaliser la TCPO en utilisant l'architecture de contrôle à base de PSAHH. 117
5.6 Gain en temps acquis en utilisant les comportements altruistes pour l'exécution de la TCPO. 118
5.7 Amélioration du temps d'exécution de la TCPO en fonction de Nc 118
5.8 Évolution du nombre d'itérations pour réaliser la TCPO en fonction de N et de Nc. 120
5.9 Gain en temps acquis lors de l'utilisation de l'architecture de contrôle à base de PSAHH. 120
5.10 PSAHH appliqué à la TCPO pour le cas d'une coopération d'un grand nombre de robots. 121

6.1 Exemples de robots à pattes apprenant à marcher en utilisant un RNA . . 127
6.2 Cycle génétique 130
6.3 Croisement en un point de deux chromosomes 133
6.4 Trajectoires obtenues pour différents types de robots, et pour différents encombrement d'obstacles (les points noirs) dans l'environnement (Ram et al. 1994). 136
6.5 Méthodologie appliquée pour évaluer les chromosomes attribués à Σ_3 . . . 139

6.6 Configuration adoptée pour l'environnement de test "*setup*", afin d'évaluer les performances des chromosomes. 141
6.7 Évolution de l'optimisation paramétrique 143
6.8 Comparaison entre l'efficacité des chromosomes 145
6.9 Diagramme de classes de la bibliothèque "*BibAG*" 146
6.10 Interface existante entre la bibliothèque pour algorithmes génétiques "*BibAG*" et le simulateur "*MiRoCo*". 147

7.1 Modélisation d'un système multi-robots par une simulation multi-agents. Pour une modélisation précise, l'idée est de faire tendre $\varepsilon \longrightarrow 0$. 154
7.2 Illustrations de quelques simulateurs présentés dans la littérature 158
7.3 Simulateur *MiRoCo* "*Mini-Robotique Collective*" 159
7.4 Schéma UML représentant les principales classes intervenant dans le cadre des simulations sous *MiRoCo*. 161
7.5 Schéma UML de l'organisation des classes d'un mini-robot ALICE simulé 162
7.6 Capteurs simulés du mini-robot ALICE 162
7.7 Quelques situations conflictuelles induites par les déplacements des agents (Magnin 1996). 164
7.8 Modèle de déplacement des agents mobiles dans les simulations 166
7.9 Application du *moteur physique* implémenté dans *MiRoCo* pour un cas simple de deux agents en mouvement. 166
7.10 Modèle de déplacement de la boite 169
7.11 Émission de signaux à valeurs discrètes dans l'environnement 170
7.12 Signaux à décroissance continue 172
7.13 Angle de vue à partir de la position du robot numéro 1 173

8.1 Plate-forme expérimentale 176
8.2 Chaîne d'acquisition et de traitement vidéo utilisée 177
8.3 Outil de *tracking* utilisé durant les expérimentations 178
8.4 Mini-robots ALICE 179
8.5 Anneau de six capteurs de lumière 183
8.6 Plate-forme de tests pour l'anneau de capteurs 184
8.7 Carte électronique développée. 185
8.8 Étage capteurs ajouté à ALICE 186
8.9 Tensions relevées aux bornes des photorésistances de l'anneau de capteurs 186
8.10 Utilisation du nord magnétique terrestre comme repère absolu pour orienter les robots. 187

8.11 Tests effectués pour l'intégration de la boussole numérique à ALICE . . . 188
8.12 Séquentialité de l'utilisation des capteurs infrarouges de proximité 189
8.13 Type d'informations transmises entre robots 189
8.14 Détection du contact du robot avec l'objet à pousser 190
8.15 Expérimentation de l'architecture de contrôle avec 8 mini-robots poussant un objet cylindrique d'une position initiale vers la cible à atteindre. 192

Liste des tableaux

2.1 Caractéristiques et problématiques posées par un contrôle distribué 40
2.2 Contrôle réactif versus cognitif 49

4.1 Nombre d'itérations nécessaires pour que le robot atteigne son objectif . . 86
4.2 Stratégie adoptée pour attirer les robots vers la boite à pousser 93
4.3 Stratégie adoptée pour le comportement d'évitement d'obstacles 94
4.4 Amélioration en pourcentage du temps moyen nécessaire pour exécuter la TCPO. 104

5.1 Amélioration en pourcentage du temps moyen nécessaire pour réaliser la TCPO, quand l'architecture de contrôle à base de PSAHH utilise les comportements altruistes. 119
5.2 Amélioration en pourcentage du temps moyen nécessaire pour réaliser la TCPO, et ce entre l'architecture de contrôle à base de PSAH et celle à base de PSAHH. 122

Table des algorithmes

3.1 Schéma moteur pour le cas de l'*évitement d'un obstacle statique* 68
4.1 Coordination entre comportements élémentaires en utilisant le PSAH . . . 84
4.2 Comportement de *repositionnement* 95
4.3 Comportement de *poussée de boite* 96
4.4 Comportement d'*alignement* 97
5.1 Processus de sélection d'action hiérarchique et hybride "PSAHH" appliqué au bloc de fusion Σ_i. 109
7.1 *Moteur physique* implémenté dans *MiRoCo* 167
7.2 Méthode d'émission de signaux continus 171
7.3 Méthode de réception des signaux continus 172
7.4 Étapes nécessaires pour l'exécution de la TCPO sous *MiRoCo* 174
B.1 Comportement d'*évitement d'obstacles* (continu) 207
B.2 Comportement d'*attraction à la lumière* (continu) 208

Glossaire et notations

- B : Symbolise la boite,
- C : Symbolise la cible à atteindre,
- $\mathbb{C}$: Ensemble de commandes,
- $\mathbb{C}_i$: Ensemble de commandes caractérisant le comportement "i",
- $\mathbb{C}_d$: Ensemble discret de commandes caractérisant le comportement "i",
- c ou $\mathcal{C}$: Commande élémentaire,
- c_i ou $\mathcal{C}_i$: Commande élémentaire générée par le comportement"$_i$",
- $c(Rot, Tra)$: Commande qui consistera à faire faire au robot une rotation de Rot degrés et une commande de translation de Tra centimètres, qui vont s'exécuter en un mouvement couplé,
- $\mathbb{CR}_i$: Ensemble des commandes refuges qui sont susceptible de déplacer le robot vers $\mathbb{PR}_i$,
- $\mathcal{CR}_i$: Commande refuge élémentaire caractéristique du comportement "i",
- CIR_i : Capteur infrarouges avec l'identifiant "i",
- CS_i : Capteur de signaux avec l'identifiant "i",
- $\mathcal{F}$: Application de l'ensemble des stimuli $\mathbb{S}$ vers l'ensemble des commandes $\mathbb{C}$,
- $\mathcal{F}_i$: Application caractéristique du comportement "i",
- N : Nombre de robots présent dans l'environnement pour réaliser la tâche coopérative,
- N^* : Nombre optimal de robots pour réaliser la tâche coopérative,
- Nc : Nombre minimal de robots qu doivent coopérer pour réaliser la tâche coopérative,
- $NbSim$: Nombre de simulations effectuées pour valider chaque résultat statistique,
- PSAH : Processus de Sélection d'Action Hiérarchique,
- PSAHH : Processus de Sélection d'Action Hiérarchique et Hybride,
- P_a : Ensemble des positions atteignables par le robot en un pas d'échantillonnage,
- P_{a_i} : Ensemble des positions atteignables par le robot en un pas d'échantillonnage et ce en utilisant le comportement "i" ($P_{a\_i} \subset P_a$),

- P_i : Sous-ensemble des positions atteignables par le robot ($P_i \subset P_a$),
- $\mathbb{PR}$: Ensemble de positions refuges avec $\mathbb{PR} \subset P_a$,
- $\mathbb{PR}_i$: Ensemble de positions refuges caractéristiques du comportement "i",
- $\mathcal{PR}_i$: Position refuge élémentaire caractéristique du comportement "i",
- $\mathbb{S}$: Ensemble de stimuli,
- $\mathbb{S}_i$: Ensemble de stimuli caractérisant le comportement "i",
- s ou S : Stimulus élémentaire,
- s_i ou S_i : Stimulus élémentaire attaché au comportement élémentaire "i",
- TCPO : Tâche Coopérative de Poussée d'Objets.

Introduction générale

La thématique générale de recherche abordée dans cette thèse concerne le contrôle d'un groupe de robots mobiles autonomes et coopératifs, l'objectif étant de réaliser des tâches complexes dans un environnement dynamique. Les avantages liés à la coopération de robots mobiles sont multiples. Parmi eux, nous pouvons citer la fiabilité, la flexibilité, la robustesse et la rapidité accrue des systèmes contrôlés induites principalement par la redondance et les mécanismes de contrôle des robots.

Plus spécifiquement, l'objectif de nos travaux de thèse vise à contrôler des entités robotiques les plus *minimalistes* possibles et à utiliser les interactions locales entre robots pour produire une forme d'*intelligence collective* évoluée. La maîtrise de ces interactions passe inévitablement par la maîtrise des actions engendrées par les robots élémentaires. C'est donc dans le cadre de la conception d'une architecture de contrôle qui exhibe à la fois, des caractéristiques individuelles adaptées à des *systèmes multi-robots* hautement dynamiques, et des caractéristiques collectives qui favorisent des buts globaux, que s'inscrivent nos travaux de recherche.

Cette approche nécessite néanmoins de s'éloigner davantage des conceptions *centralisées* et *cognitives* du contrôle des systèmes multi-robots pour tendre vers des systèmes de contrôle exhibant des caractéristiques complètement *distribuées* et *réactives*. Dans ce contexte, les *architectures de contrôle comportementales*[1] investiguées durant nos travaux de thèse, nous ont conduit à proposer des architectures de contrôle appropriées au contrôle et à la coopération d'un groupe de robots minimalistes.

L'un des éléments primordiaux du fonctionnement de ces architectures comportementales, réside dans les mécanismes de coordination des actions engendrées par les différents comportements élémentaires. En effet, ceci constitue un domaine important de recherches qui a conduit au développement de différents *mécanismes de coordination* entre comportements. Parmi les plus représentatifs, nous pouvons citer la subsomption de Rodney Brooks (Brooks 1986) et les schémas moteurs de Ronald Arkin (Arkin 1989*b*).

[1] "*Behavior-based roboticicts argue that there is much that can be gained for robotics through the study of neurosciences, psychology, and ethology*" (Arkin 1998).

En se basant sur les points forts caractéristiques des mécanismes de coordinations proposés dans la littérature, nous avons proposé tout d'abord le PSAH "Processus de Sélection d'Action Hiérarchique" puis le PSAHH "PSAH Hybride" qui permettent de coordonner des comportements élémentaires d'une manière hiérarchique, flexible, et surtout intuitive afin de réaliser des tâches coopératives complexes.

L'organisation générale de ce manuscrit est décomposée en trois parties distinctes :

- **Partie I "Problématique de Recherche et État de l'Art" :** Cette partie est consacrée à l'introduction de la thématique générale de notre recherche et à la présentation de l'état de l'art concernant les architectures comportementales. Cette partie est constituée de trois chapitres. Le chapitre I nous immerge progressivement dans notre thématique de recherche et nous permet d'expliciter la pluridisciplinarité des approches régissant le contrôle d'un groupe de robots mobiles. Le chapitre II confronte les différentes classes d'architecture de contrôle : centralisée versus distribuée, cognitive versus réactive. Cette étude jumelée à quelques notions données au chapitre I nous a permis de définir un cahier des charges précisant les principales caractéristiques d'un contrôle approprié pour la coopération d'un groupe de plusieurs dizaines de robots minimalistes. Le chapitre III, quant à lui, se focalise davantage sur l'objet même de notre problématique de recherche, et ce au travers d'un état de l'art des différentes architectures de contrôle comportementales proposées dans la littérature.

- **Partie II "Contrôle d'un Groupe de Robots Minimalistes" :** Cette partie est constituée de trois chapitres, et met en exergue les mécanismes de contrôle et d'optimisation que nous proposons. Le chapitre IV décrit le Processus de Sélection d'Action Hiérarchique "PSAH" proposé pour coordonner l'activité d'un ensemble de comportements élémentaires. Le chapitre V expose l'adjonction de nouveaux mécanismes de coordination au mécanisme de base déjà exposé au chapitre IV donnant ainsi le "PSAH Hybride". Ce processus permet de rendre plus flexible et intuitive l'obtention d'architectures comportementales appropriées au contrôle et à la coopération d'un groupe de robots mobiles. Les deux mécanismes sont testés et comparés d'une manière intensive sur la base de l'exécution de tâches coopératives de poussée d'objets. Pour conclure cette partie nous proposons au chapitre VI, une méthode appropriée pour optimiser les paramètres prépondérants au fonctionnement du PSAHH.

- **Partie III "Environnement de Simulation et Plate-forme Expérimentale" :** Cette dernière partie est consacrée à la présentation détaillée de l'environnement de simulation et de la plate-forme expérimentale développés durant nos travaux de thèse. Le chapitre VII s'articule autour du simulateur *MiRoCo* que nous avons développé et utilisé pour simuler des environnements multi-robots avec une grande précision. Quant au chapitre VIII, il est dédié à l'étude et à l'implémentation des différentes possibilités de transposition matérielle des éléments de conception, de contrôle et d'analyse proposés et utilisés aux chapitres IV et V.

Première partie

Problématique de Recherche et État de l'Art

"Une question bien posée n'est déjà plus une question, puisqu'elle renferme tous les éléments de la réponse. Sans paradoxe, une question ne peut en tant que telle, être que mal posée".
Geroges Canguilhem.

Chapitre 1

La coopération d'entités autonomes

Résumé : *Ce premier chapitre se veut une introduction à la problématique de base qui consiste à faire coopérer un groupe de robots mobiles autonomes. Après avoir introduit brièvement les caractéristiques et les contraintes inhérentes au fonctionnement d'une seule entité robotique*[a]*, nous exposerons certains éléments qui ont conduit la robotique dite traditionnelle à s'intéresser davantage au fait de produire une forme d'intelligence collective au lieu de se contenter d'intelligences individuelles éparses. Une grande partie de ce chapitre sera consacré aux différentes approches disciplinaires gravitant autour des thématiques liées à la coopération d'entités autonomes. Ainsi les approches émanant des sciences du vivant, de l'informatique et bien évidemment de l'automatique seront mises en avant, montrant ainsi le caractère pluridisciplinaire des approches et des concepts développés dans la littérature, concernant la coopération d'entités autonomes d'une manière générale, et d'entités robotiques d'une manière particulière.*

[a] Unité indivisible qui est amenée dans le contexte d'un système multi-robots à interagir avec d'autres entités robotiques pour réaliser des tâches coopératives.

1.1 L'entité élémentaire participant à la coopération

Avant de parler de mécanismes et d'architectures de contrôle caractérisant un groupe de robots mobiles, nous allons nous intéresser tout d'abord aux capacités structurelles et cognitives propres à une unité robotique élémentaire. En effet, il est indéniable qu'il y a une corrélation importante entre capacités individuelles et capacités collectives d'un groupe de robots mobiles pour la réalisation des tâches coopératives désirées.

1.1.1 Qu'est-ce qu'un robot mobile ?

Le terme "robot" a été introduit en 1920 par l'écrivain tchèque Karel Capek dans sa pièce de théâtre RUR[1]. Ce terme, provenant du tchèque ROBOTA "travail forcé", désignait à l'origine une machine androïde capable de remplacer l'homme dans toutes ses tâches. Effectivement, les robots ont envahi progressivement différents secteurs d'activité. L'industrie est sans doute le secteur qui a le plus profité des avancées de la robotique. Le bras manipulateur par exemple, constitue la forme de robot la plus utilisée, ce qui lui a donc permis de bénéficier autant de la part de la communauté scientifique qu'industrielle, des plus grands efforts de développement et de perfectionnement (modélisation, identification, contrôle, etc.). Depuis leurs premières utilisations (début des années 60) jusqu'à nos jours, les bras manipulateurs avec leurs différents types de structure (série, parallèle, hybride), de dimensions et de spécificités (Khalil & Dombre 1999) se sont intégrés aux applications les plus diverses, tels que l'industrie de l'automobile, l'usinage à grande vitesse ou la micro-électronique.

Au début de la robotique, les robots étaient principalement solidaires d'un bâti statique, ce qui n'empêchaient pas bien évidemment, qu'ils aient un espace de travail étendu (fonction de leur envergure). Le besoin progressif de disposer de robots qui se déplacent, c'est-à-dire *mobiles*, s'est imposé par la suite comme une évidence à atteindre[2], car à une époque où la flexibilité et l'adaptabilité sont des maîtres mots, cette mobilité ne pouvait être que très souhaitable. D'ailleurs, cette mobilité trouve un grand écho dans différents domaines d'utilisation tels que l'agriculture, les travaux publics ou l'exploration spatiale.

Le besoin de cette mobilité des robots a permis aussi l'ouverture de vastes champs d'investigation pour la recherche et ce dans des disciplines aussi variées et distinctes que sont la mécanique, l'électronique, l'automatique, l'intelligence artificielle, etc.

Note : Dans ce qui suit, quand nous parlerons de "robot" nous sous-entendons "robot mobile". Si nous parlons d'un autre type de robot, alors nous le spécifierons dans le texte.

[1] *Rossum's Universal Robots*

[2] Le premier robot mobile au nom de Shakey, est apparu en 1967 au *Stanford Research Institute*.

1.1.2 Importance des caractéristiques structurelles d'un robot (est-il capable de réaliser tout ce qu'on lui demande ?)

Le rôle des caractéristiques structurelles d'un robot (i.e., sa morphologie, les caractéristiques qualitatives et quantitatives de ses capteurs et de ses actionneurs et leur disposition autour du robot) par rapport aux tâches qu'on lui demande d'exécuter, n'a que très peu été étudié dans la littérature. Pourtant, nous savons pertinemment que les êtres vivants évoluent et s'adaptent (intellectuellement et physiquement) (Darwin 1859) en fonction de leurs aspirations et de l'environnement dans lequel ils sont immergés. Il parait donc primordial que le corps soit disposé de telle sorte qu'il puisse réaliser ce qu'il est censé exécuter. La figure 1.1 montre les interactions bidirectionnelles existantes entre structure, capteurs/actionneurs et intelligence dans les systèmes vivants (Dittmar & Delhomme 2001).

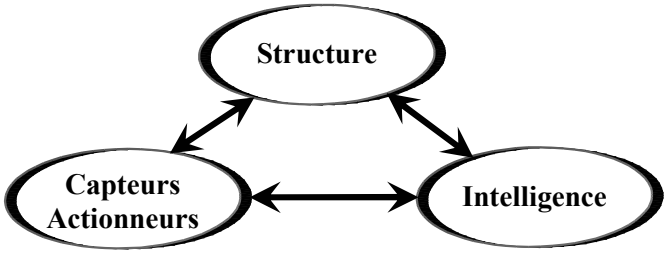


FIG. 1.1 – Capteurs, actionneurs, structures et intelligence interagissent constamment dans les systèmes vivants.

En effet, la plupart des travaux traitant du contrôle de robots mobiles, ne prête pas ou très peu d'importance aux relations existantes entre les caractéristiques intrinsèques du robot l'intelligence qu'il lui est imputée. Considérant ainsi que la seule forme d'adaptabilité du robot, émane de son architecture de contrôle, négligeant par ce fait le concept de "*The Ecological Balance*", c'est-à-dire la liaison entre le matériel et la notion d'adaptabilité. Rolf Pfeifer et Lukas Lichtensteiger dans respectivement (Pfeifer 2000) et (Lichtensteiger 2000) démontrent clairement l'influence de la morphologie par rapport aux capacités décisionnelles du robot. Pfeifer change entre autres les positions et le nombre de capteurs autour d'un robot mobile pour démontrer cette influence. Les yeux des insectes par exemple différent d'une variété à une autre et parfois cette différence est apparente entre le mâle et la femelle d'une même espèce. Ceci est justifié dans (Lichtensteiger 2000) par le fait que c'est la tâche à accomplir qui influence la morphologie de ces organes. Les travaux de Lichtensteiger utilisent l'adaptation de la distribution spatiale de facettes d'oeils artificiels (adaptation morphologique du matériel) afin d'ob-

tenir la meilleure précision possible concernant le moment d'impact d'un mobile avec un obstacle dans l'environnement. Les capteurs des êtres vivants par exemple, sont localisés aux endroits les plus pertinents pour avoir le maximum d'efficacité. Il est classiquement admis par exemple qu'il y a plus de capteurs sur la paume d'une main que sur la peau du thorax ou de l'abdomen réunis (Dittmar & Delhomme 2001).

Dans un autre registre, il existe des travaux faisant une analogie entre les comportements d'animaux sociaux et leur reconstitution à l'aide du contrôle d'un groupe de robots mobiles. Nous pouvons citer par exemple les travaux de Ronald Kube et d'Eric Bonabeau dans (Kube & Bonabeau 2000) qui utilisent un groupe de robots mobiles pour déplacer (pousser) des objets lourds à la manière des fourmis (Lioni 1999), ou bien les travaux de Christian Jost et de ses collèges (Jost et al. 2004) dans le cadre du projet LEURRE[3] qui ont implémenté sur des robots mobiles, des comportements analogues au déplacement et à l'agrégation des blattes, et ce dans le but ultérieur d'immerger ces robots mobiles dans une société de blattes afin d'étudier, voire modifier grâce à ces artefacts, autant leurs comportements individuels que collectifs.

Dans ce genre de travaux très intéressants, ne pas tenir compte des capacités sensorielles ou d'actionnement des robots par rapport à leurs analogues dans la nature, peut mener à des résultats peu précis. Il faut garder à l'esprit qu'aussi limitées soient les capacités cognitives et motrices des insectes en général, il est indéniable qu'ils ont tout de même des capacités sensorielles et d'actionnement importantes, d'autant plus qu'ils sont complètement adaptées à ce qu'ils réalisent depuis des milliers voire des millions d'années dans leur environnement. Martin Giurfa dans (Giurfa 2003) au travers d'une étude poussée sur l'activité neuronale de cerveaux d'abeilles[4], a attribué à celles-ci des capacités cognitives d'apprentissage et d'adaptation insoupçonnées.

Les capteurs des êtres vivants ne sont pas utilisés à l'unité mais par centaines, milliers voire millions (Dittmar & Delhomme 2001). Par conséquent comparer d'une manière générale un robot disposant uniquement de quelques capteurs[5] à un animal, peut être très lointain de la réalité des choses, et devra être traité avec précaution. Par conséquent, il faut garder à l'esprit qu'on ne peut pas demander à un robot qui ne dispose que de deux capteurs infrarouges, d'avoir les mêmes comportements qu'un autre robot qui dispose de quatre capteurs, moins encore par rapport à une fourmi qui dispose d'une multitude de capteurs répartis sur tout son corps. Donc, le tout est de trouver le bon compromis entre

[3] `http://leurre.ulb.ac.be`

[4] Cerveau qui contient tout de même plus de 960 000 neurones.

[5] Qui peuvent être très limités en précision ou même peu fiables pour récupérer les stimuli appropriés de leur environnement.

capacités sensorielle et d'actionnement pour réaliser les tâches voulues.

Cette section nous amène à nous poser cette question : pour la réalisation d'une tâche de contrôle quelconque par un robot mobile, est-ce au contrôle de s'adapter à la structure sensorielle et d'actionnement du robot, ou alors doit-on mener une réflexion conjointe entre contrôle et structure du robot ? D'une autre manière, quelle configuration de capteurs ou d'actionneurs serait la plus adaptée pour optimiser la réalisation de la tâche désirée ?

Pour différentes raisons, parmi elles, celle de la composante temps à investir, pour établir les meilleurs compromis entre structure et tâche à réaliser, nous avons décidé dans le cadre de nos travaux de thèse d'adopter une structure robotique mobile existante, en l'occurrence celle du mini-robot ALICE (Caprari et al. 2002), choisi pour son caractère minimaliste mais très opérationnel (cf. chapitre VIII). L'idée est d'intégrer à cette base existante, les éléments (capteurs, actionneurs) qui viendraient à manquer pour réaliser les tâches coopératives que nous nous avons choisi d'étudier. Sans réduire la généricité des architectures de contrôle proposées aux chapitres IV et V, il faut noter que celles-ci tiennent compte aussi des spécificités structurelles du mini-robot ALICE.

1.1.3 Parlons un peu d'autonomie

L'obtention de l'autonomie complète d'un robot mobile est conditionnée par l'adjonction de plusieurs types d'autonomie que nous répertorions ainsi : *énergétique*, *sensorielle*, et *décisionnelle*.

Concernant l'aspect énergétique, le robot mobile peut être :
- *énergétiquement autonome* : ceci est atteint, soit en embarquant sur le robot des modules de stockage d'énergie tels qu'une batterie ou des piles, soit en intégrant des modules de transformation d'énergie tels que des panneaux solaires,
- *énergétiquement non autonome* : dans ce cas, le robot s'alimente via des sources d'alimentation externes telles que des fils électriques, ou un faisceau laser orienté vers le robot.

L'autonomie énergétique est fortement souhaitée cependant elle n'est pas obligatoire pour toutes les tâches à accomplir par le robot. En fait, elle dépend essentiellement des technologies de production et de stockage de l'énergie nécessaire au robot.

Concernant l'autonomie décisionnelle, c'est celle qui pose le plus d'interrogations de fond, car elle conditionne l'approche qui doit être utilisée pour le contrôle du robot mo-

bile, et ceci autant sur des aspects conceptuels que méthodologiques. Il est à noter que l'autonomie décisionnelle est souvent liée à l'autonomie sensorielle du robot en question (cf. §2.1, page 38).

Un exemple illustrant les différents types d'autonomie, peut être donné au travers du robot "Sojourner" (cf. figure. 1.2) qui est le premier robot mobile à avoir roulé sur le sol martien[6]. Le Sojourner dispose entre autres d'une batterie et d'un panneau solaire qui lui assure une autonomie énergétique. Il possède aussi une caméra et des capteurs lasers pour pouvoir percevoir son environnement. Quant à son aspect décisionnel, Sojourner a été principalement téléopéré depuis la Terre. Sachant que la transmission radio des commandes mettent en moyenne 11 minutes pour parcourir les 192 millions de kilomètres séparant Mars de la Terre, Sojourner a été conçu pour fonctionner aussi d'une manière semi-autonome (Volpe et al. 1997) appelé *Supervised Autonomous Control*, c'est-à-dire que les ordres généraux (exemple atteindre un objectif) sont donnés au robot depuis la Terre, cependant celui-ci navigue d'une manière complètement autonome entre les roches du sol martien.

FIG. 1.2 – Robot mobile "Sojourner" utilisé pour la mission Pahtfinder de la NASA

1.1.4 Le contrôle des robots mobiles

Le contrôle des robots mobiles nécessite la juxtaposition de trois phases : la *perception*, la *décision*, et l'*action*. La perception construit un modèle même simple de l'environnement du robot, la décision utilise ce modèle pour générer des consignes de mouvement, et finalement l'action transforme ces consignes en commandes adéquates pour les effecteurs

[6]Plus précisément le 4 juillet 1997, la sonde Pathfinder se pose sur le sol martien, avec à son bord le robot Sojourner.

du robot. Un contrôle bien mené nécessite donc la maîtrise de ces trois phases.

Il est possible de faire l'analogie entre la *tâche* que doit réaliser un robot mobile et la représentation habituelle en automatique d'un système à asservir (cf. figure. 1.3). Dans ce cas, le bloc "Contrôleur" correspond au contrôleur du robot mobile, quant au bloc "Système" il correspond au robot immergé dans son environnement. Le robot va récupérer les informations émanant de son environnement via ses capteurs. Après un traitement approprié de ses perceptions au niveau de son contrôleur, une *commande* est produite afin de satisfaire au mieux la *consigne* qui est dictée entre autres par la tâche à réaliser. L'exécution de la commande par les effecteurs du robot permettra à ce dernier de changer son état ainsi que l'état du système global (modification de la *sortie*), ce qui referme la boucle de retour.

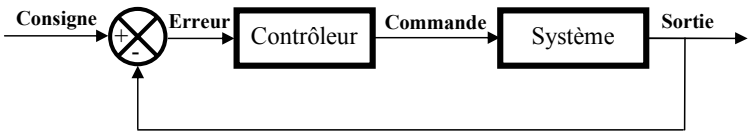


FIG. 1.3 – Schéma d'un asservissement conventionnel

Après avoir présenté succinctement les principales caractéristiques inhérentes à une entité robotique élémentaire (structure, autonomie, contrôle), nous passons à présent au grand champ d'investigation qui est celui de la robotique mobile collective. Ses mécanismes, et ses différents courants conceptuels et méthodologiques seront traités dans ce qui suit. Dans un souci de clarté par rapport au lecteur, nous donnons en Annexe A, quelques définitions de base du vocabulaire que nous sommes amenés à utiliser par la suite.

1.2 Et la coopération dans tout ça !

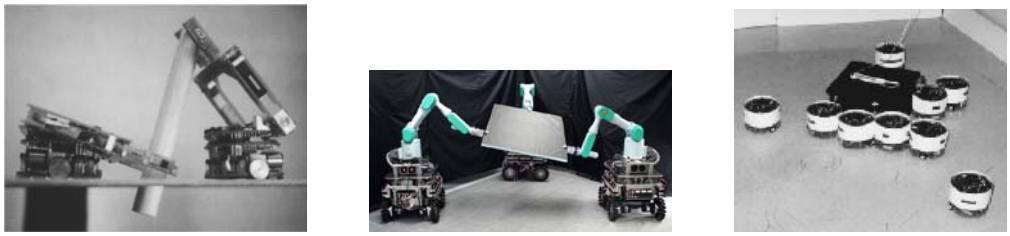
Nous allons nous focaliser à présent sur l'immersion d'un robot mobile dans un système multi-robots coopératif. Ceci implique entre autres, que le contrôle du robot en question, ne dépendra plus uniquement de ses propres perceptions et objectifs, mais devra tenir compte aussi d'un certain nombre de considérations liées à l'évolution globale du système multi-robots. Ceci ajoute bien entendu un niveau de complexité au contrôle du robot mobile.

1.2.1 La robotique collective

La robotique collective est synonyme d'existence d'un groupe de robots. Elle stipule non seulement la mise en place d'un contrôle individuel pour chaque robot, mais impose aussi l'utilisation de stratégies de contrôle appropriées afin que l'assemblage de toutes ces entités robotiques engendre des configurations cohérentes et efficaces pour la réalisation des tâches désirées.

Généralement, faire appel à un groupe de robots au lieu d'en utiliser qu'un seul est motivé par deux facteurs majeurs :

1 - soit la tâche à exécuter nécessite impérativement la coopération d'un nombre minimal de robots, i.e., que la tâche en question ne peut s'accomplir sans l'intervention simultanée d'un nombre critique *Nc* de robots. Dans (Martinoli & Mondada 1995), les auteurs font coopérer des robots munis de pinces (cf. figure. 1.4(a)) en faisant en sorte qu'ils puissent synchroniser leurs activités dans le but de retirer un long bâton d'un trou. Les travaux menés par Yasuhisa Hirata (Hirata et al. 2002) (cf. figure. 1.4(b)) permettent de contrôler un ensemble de bras manipulateurs montés chacun sur un robot mobile pour réaliser la tâche dite des *déménageurs* (i.e., soulever et déplacer un objet trop volumineux et/ou trop lourd par la coordination des activités d'un ensemble de robots). Ronald Kube dans (Kube & Bonabeau 2000) (cf. figure. 1.4(c)) utilise quant à lui plusieurs robots réactifs afin de pousser un objet trop lourd pour un seul robot. Dans cet exemple, la poussée simultanée d'au moins deux robots est nécessaire pour déplacer l'objet.



(a) (b) (c)

FIG. 1.4 – Tâches nécessitant la coopération de plusieurs robots

La coopération dans ce premier cas de figure exige donc impérativement, l'intervention d'un groupe de robots pour pallier le manque : d'effecteurs, de puissance, d'informations ou bien plus radicalement le manque d'intelligence embarquée dans chacune des entités robotiques.

2 - soit l'amélioration de certaines performances liées à l'exécution des tâches à réaliser, comme par exemple :
- la *rapidité* : on cherche à obtenir un niveau de performance élevé en tirant parti de la parallélisation des tâches. e.g., l'exploration parallèle d'un environnement inconnu par ensemble de robots mobiles, dans le but de faire soit une cartographie de l'environnement (Cohen 1996), (Sty 2001), soit la réalisation d'une tâche de fourragement (cf. §1.2.1.2.3, page 19). Ces deux tâches peuvent être réalisées par un seul robot, mais l'ajout d'autres robots va faire en sorte d'accélérer l'exécution des tâches en question,
- *robustesse* - *fiabilité* : les performances du contrôle peuvent être très peu affectées en cas de défaillance d'un agent par exemple (Parker 2000), (Mataric´ et al. 1995),
- *flexibilité* : possibilité d'exécuter les tâches désirées, de diverses manières. Ceci est induit principalement par la redondance des entités robotiques,
- *émergence* : l'idée ici est de produire une performance collective qualitativement supérieure à celle de l'addition des unités (cf. §2.2.2.2, page 46).

Cependant, le fait de contrôler/coordonner plusieurs robots au lieu de n'en contrôler qu'un seul, complexifie considérablement la commande du système, et ce vu l'augmentation :
- de la dynamique des interactions entre entités robotiques dans l'environnement. Ces interactions sont susceptibles, si elles sont mal maîtrisées, d'influer d'une manière néfaste sur l'évolution du système, car les robots peuvent se gêner, se bloquer, se désynchroniser, etc.
- du nombre de variables régissant l'évolution du système, dû directement à l'augmentation du nombre de systèmes (robots) dans l'environnement,
- de la complexité du contrôle inhérent déjà à un seul robot, qui doit non seulement agir en fonction des stimuli lui parvenant de l'environnement mais doit aussi adapter son comportement à ceux des autres individus. C'est-à-dire que le robot va essayer de tendre vers un équilibre viable voire optimal pour l'exécution de la tâche coopérative,
- des incertitudes perceptuelles des robots. Celles-ci sont dues principalement à la nature hautement dynamique qui peut être engendrée par le système multi-robots. Ces incertitudes peuvent complexifier aussi davantage le contrôle du système si elles sont effectives au niveau d'un grand nombre de capteurs.

Les points cités ci-dessus, sont parmi ceux qui font que le contrôle d'un système multi-robots, est plus délicat à mettre en place que le contrôle d'un seul robot élémentaire.

Pour rester dans le même cheminement d'idée quant à l'analogie faite entre un système à asservir et la réalisation d'une tâche par un robot mobile (cf. §1.1.4, page 12), nous étendons cette analogie pour le cas d'un système multi-robots (cf. figure. 1.5). Nous remarquons d'une part, que les robots partagent le même environnement et d'autre part, que les décisions (commandes) générées par chacun des contrôleurs, sont aussi influencées par les interactions avec les autres robots.

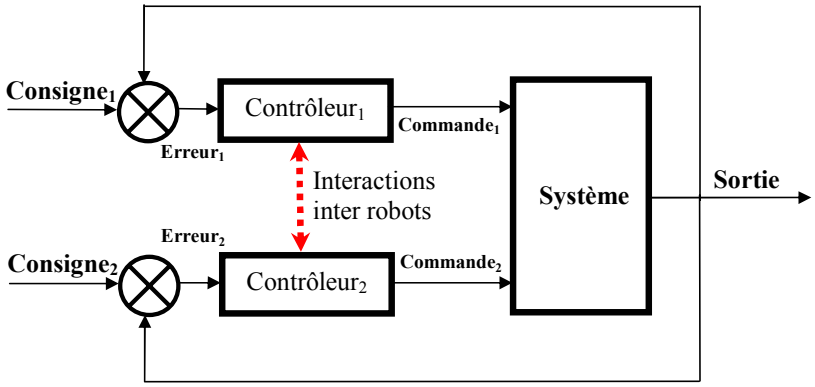


FIG. 1.5 – Schéma d'asservissement d'un système multi-robots

1.2.1.1 Qu'est-ce que la coopération de robots ?

Pour parler effectivement de robotique coopérative, les robots doivent impérativement partager et/ou intervenir simultanément sur des ressources communes (e.g., un objet à déplacer collectivement, une voiture à souder ou à peindre dans une chaîne de montage de voiture, etc.), ce qui signifie que, d'une part les robots doivent évoluer dans le même environnement, et d'autre part qu'ils sont appelés à coordonner leurs actions pour réaliser la tâche coopérative. Cette coordination peut être très élaborée, et donc obtenue via des mécanismes de très haut niveau, que l'on peut qualifier de *cognitifs*. Cette coordination peut aussi être complètement émergente des interactions induites pas les robots, et dans ce cas de figure l'architecture de contrôle est qualifiée de *réactive* (Brooks 1986). Ces deux approches (cognitive, réactive) de coordination des interactions entre robots seront traitées avec plus de détails en section 2.2, page 41.

1.2.1.2 Projets effectifs et tâches génériques pour la robotique collective

Les projets applicatifs potentiels et effectifs impliquant un ensemble de robots mobiles, sont nombreux et très prometteurs. Ceci est dû principalement aux nombreuses plus-values qualitatives et quantitatives qui peuvent être apportées par rapport aux approches traditionnelles existantes. Parmi les applications majeures qui tirent et qui pourraient tirer profit des recherches menées sur la coopération d'un groupe de robots autonomes, nous pouvons citer : les applications militaires telles que le maintien de patrouille en formation (Balch & Arkin 1995*b*), (Fraisse et al. 2004) ; l'exploration spatiale (Huntsberger et al. 2003) ; le transport coopératif (Alami et al. 1998*a*), (Ahmadabadi & Nakano 2001), etc.

Les tâches génériques en robotique collective, ont comme principal objectif de fournir des tâches de références "*benchmark*" pour évaluer les architectures de contrôle proposées dans la littérature. Ces évaluations se font sur plusieurs aspects :

- *quantitatif* et *qualitatif* : ceci concerne par exemple les tests d'amélioration de certaines métriques caractérisant les tâches à exécuter (Parker 2001),
- *méthodologique* : au sens que nous trouvons des approches de contrôle émanant de différentes disciplines comme l'automatique, l'informatique, ou les sciences du vivant (cf. section 1.2.2, page 24),
- *conceptuel* : nous pouvons alors confronter les avantages et les inconvénients d'un contrôle centralisé par rapport à un contrôle distribué (cf. §2.1, page 38), ou d'un contrôle cognitif par rapport à un contrôle réactif (cf. §2.2, page 41).

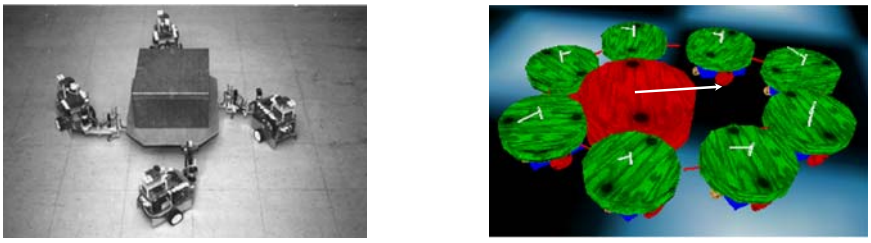
Ces tâches génériques sont censées exciter l'ensemble des modes de fonctionnement du système multi-robots. Par conséquent, la viabilité, la pertinence et les différentes caractéristiques qu'offre le contrôle utilisé sont ainsi systématiquement évaluées. Ces évaluations permettront ultérieurement d'appliquer l'architecture de contrôle proposée sur des applications effectives, totalement différentes des tâches génériques mais qui partageraient certains points caractéristiques avec l'application désirée.

Nous présentons dans ce qui suit, un aperçu de la multitude de tâches génériques, couramment traitées en robotique coopérative.

1.2.1.2.1 Transport et manipulation coopérative d'objets L'objectif dans ce type de tâches est de faire coopérer un groupe de robots mobiles pour soulever et/ou porter et/ou réorienter et/ou pousser des objets qu'un seul robot ne pourrait réaliser

tout seul. De très nombreux travaux traitent de ce type de tâches et cela vu son aspect pratique et potentiellement transposable à des applications effectives.

Stiwell dans (Stilwell & Bay 1993) utilise plusieurs robots porteurs qui se positionnent en dessous d'une palette chargée pour la déplacer. Les robots mobiles disposent d'un capteur de force qui leur sert d'information pour contrôler le déplacement de la palette d'une manière stable et distribuée. Dans le même contexte Ahmadabadi dans (Ahmadabadi & Nakano 2001) propose une architecture de contrôle distribuée nommée *constrain-move strategy*, et ce pour soulever, orienter, déplacer, déposer une charge (cf. figure. 1.6(a)) le long d'une trajectoire contrainte par la disposition des robots. Rus dans (Rus et al. 1995) utilise des robots pousseurs qui sont guidés par des scénarios appropriés pour orienter collectivement des objets volumineux dans l'environnement. L'approche proposée par Rus dépend d'une multitude d'informations capteurs et d'une communication de haut niveau entre les robots. Dans le projet européen *Swarm-Bots*[7] un groupe de robots (nommés s-bots) a la possibilité de s'auto-assembler pour former différents types de structures physiques. Parmi elles, on trouve une structure qui permet de pousser-tirer d'une manière coopérative des objets en les entourant complètement (cf. figure. 1.6(b)) afin de donner ainsi la possibilité à certains robots de pousser et à d'autres de tirer (Baldassarre et al. 2003). Kube dans (Kube & Bonabeau 2000) s'inspire des comportements des fourmis (e.g., déplacement des proies), pour contrôler un groupe de robots complètement réactifs, afin de réaliser une tâche de poussée d'objets. Il est à noter que Kube dans son architecture n'a prévu aucun mécanisme de communication entre robots ce qui rend le contrôle proposé d'autant plus réactif (cf. §2.2, page 41).



(a) (b)

FIG. 1.6 – Exemple de manipulation coopérative d'objet

Il existe dans la littérature de nombreux autres travaux traitant de tâches en relation avec le transport et la manipulation coopérative d'objets. Nous pouvons citer : (Ozaki et al. 1997) avec l'architecture ACTRESS[8], (Muñoz 2003) avec les mécanismes de coopéra-

[7] `http://www.swarm-bots.org`

[8] *ACTOR-based Robot and Equipments Synthetic System.*

tion située proposés, (Caloud et al. 1990), (Donald et al. 1994), (Mataric´ et al. 1995), (Yamada & Saito 2001), (Müller et al. 1998).

1.2.1.2.2 Mouvement en formation Naviguer en formation et en présence d'obstacles est la tâche générique qui consiste à simultanément :

- se déplacer vers l'objectif,
- éviter les obstacles statiques,
- éviter les autres robots,
- et maintenir la formation.

Balch dans (Balch & Arkin 1995*b*) propose de contrôler la navigation de quatre robots mobiles en formation dans un milieu contraint (présence d'obstacles) en utilisant des *schémas moteurs* spécifiques (cf. §3.3.1, page 66). Les configurations spatiales testées sont représentées en figure 1.7. Yamaguchi dans (Yamaguchi et al. 2001) traite cette tâche générique en distribuant complètement le contrôle sur les entités autonomes qui n'exploitent que des informations localisées pour établir leurs actions. Yamaguchi établit aussi un modèle mathématique pour démontrer la faisabilité et la stabilité de l'approche proposée.

Parmi les autres travaux qui traitent de la tâche de mouvement en formation, nous pouvons citer : (Chen & Luh 1994), (Wang 1991), (Premvuti & Yuta 1990).

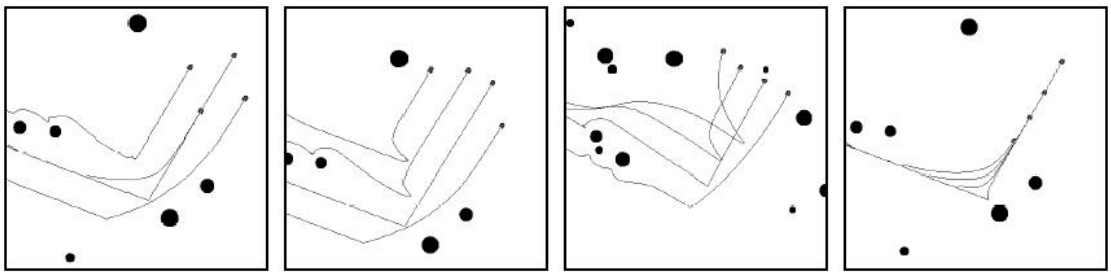


FIG. 1.7 – Formations pour quatre robots mobiles (de gauche à droite : diamant, cale, ligne, colonne). Les points noirs représentant les obstacles (Balch & Arkin 1995*b*).

1.2.1.2.3 Fourragement La tâche de fourragement caractérise à l'origine les sociétés d'insectes, comme celles des fourmis. Cette tâche consiste à rechercher la nourriture dans l'environnement soit d'une manière aléatoire, soit d'une manière dirigée (e.g., l'utilisation des phéromones pour guider les autres congénères vers la source de nourriture) et à la transporter par la suite au nid. Cette tâche a inspiré bon nombre de travaux en robotique, et cela bien évidemment en remplaçant : les fourmis par des robots, la nourriture

par des objets caractéristiques dans l'environnement, les phéromones par des communications directes entre robots, et le nid par une position identifiée dans l'environnement. Mataric´ dans (Mataric´ 2004) utilise un groupe homogène de sept robots mobiles communiquant entre-eux, pour trouver et collecter des palets distribués aléatoirement dans l'environnement. Balch dans (Balch & Arkin 1995*a*) a investigué l'importance de la communication pour réaliser différentes variantes du fourragement en l'occurrence : la mine[9], la consommation[10] et l'exploration[11]. Vaughan dans (Vaughan et al. 2000*a*, 2000*b*) a implémenté la tâche de fourragement avec quatre robots autonomes. Ces robots tiennent compte entre autres de leur odométrie pour diffuser (en utilisant une communication hertzienne) les positions des sites de récolte trouvés. L'auteur constate une certaine robustesse du système multi-robots, malgré les perturbations évidentes, dues aux erreurs d'odométrie.

Il est à noter que le fourragement est l'une des tâches génériques la plus étudiée dans la littérature. C'est ce qui justifie le nombre important de travaux traitant de cette tâche (Drogoul 1993), (Steels 1990), (Arkin 1992), (Simonin 2001), etc.

Il y a d'autres tâches génériques en robotique collective. Parmi elles nous pouvons citer : la tâche de tri d'objets "*sorting objects*" qui consiste à mettre les objets de même nature dans les mêmes tas (e.g., tri en fonction des couleurs des disques) (Melhuish 2001) (cf. figure. 1.8), l'exploration d'environnements inconnus dans le but de faire une cartographie de l'environnement (Cohen 1996), (Singh & Fujimura 1993).

FIG. 1.8 – Tâche du tri d'objets

[9] Consiste à trouver une ou plusieurs mine(s) et à rapporter le minerai à la base.
[10] Cette tâche a pour effet de consommer le minerai trouvé sur place.
[11] Consiste à parcourir la plus grande surface possible de l'environnement.

1.2.1.3 Communiquer pour bien coopérer !

La communication est la base de la résolution coopérative de problèmes (Parker 1998). Balch et Arkin dans (Balch & Arkin 1995*a*) ont testé et confirmé, l'importance de différents types de communication, entre robots, pour l'exécution de tâches coopératives telles que le fourragement.

Communiquer d'une manière générale peut être synonyme de partage d'informations, d'expression des états internes, de négociation pour trouver le meilleur compromis aux situations, de requête pour avoir des informations, etc. En fait, ce sont tous les mécanismes qui conduisent, s'ils sont bien traités, analysés et interprétés à ce que le groupe soit plus efficace. L'importance évidente de la communication nous amène donc à étudier plus en détails ces mécanismes dans le cadre plus précis d'un système multi-robots. On note que cette communication sert essentiellement à orienter les interactions entre robots vers des situations non conflictuelles (e.g., éviter les conflits spatiaux temporels (Leroy 1998), (Simonin 2001)), et/ou coopératives.

La communication entre agents est catégorisée dans ce qui suit, selon plusieurs critères tels que le degré de sophistication, le type d'informations échangées, les mécanismes de fonctionnement.

1.2.1.3.1 Communication de haut niveau On entend par communication de haut niveau, toute communication s'exerçant entre individus via des mécanismes de protocoles de évolués, tels que :

- l'établissement d'une connexion orientée entre deux entités avec des identifiants (émetteur$_{idx}$ / récepteur$_{idy}$),
- l'échange d'informations de haut niveau, comme par exemple :
  - l'état d'achèvement des sous-tâches entreprises par chaque robot, ceci est utilisé par exemple dans l'architecture de contrôle ALLIANCE (cf. §3.2.5, page 63) proposée par Parker (Parker 1998),
  - les plans d'actions propres à chaque robot, pour trouver le meilleur compromis des actions à entreprendre pour chaque robot (Robert 1996), (Aguilar 1997), (Leroy 1998),
  - la mise aux enchères des tâches qui doivent être exécutées, Gerkey dans (Gerkey & Mataric´ 2002*b*) définit le protocole nommé MURDOCH, qui réalise une allocation dynamique des tâches à effectuer par un groupe de robots mobiles. Le protocole qu'il propose est la mise aux enchères (ou avec une forme d'appel d'offre) des tâches qui doivent être exécutées "*auction-based task allocation*"

et laisse par la suite le robot le plus apte (le plus offrant) exécuter la tâche en question.

Il est à noter que les types de communication cités ci-dessus exigent l'échange de grands flux d'informations.

- l'existence de protocole de validation de la réception des messages,
- l'existence d'un tableau noir "*Blackboard*[12]" qui a comme effet de centraliser les communications entre agents. Le projet MARTHA[13] (Alami et al. 1998*a*) par exemple utilise une forme de tableau noir pour pouvoir coordonner l'activité de plusieurs dizaines de robots mobiles transportant des conteneurs dans des gares, des ports ou des aéroports. Dans ce système, un superviseur centralise les décisions et les communications globales du système multi-robots.

La figure 1.9 montre un scénario possible d'une communication de haut niveau. Il est à noter que la littérature regorge de travaux et de réalisations en matière de protocoles de communication que nous avons nommé ici haut niveau[14]. Ils ne sont pas tous appliqués en robotique mobile coopérative, mais demeurent très facilement transposables.

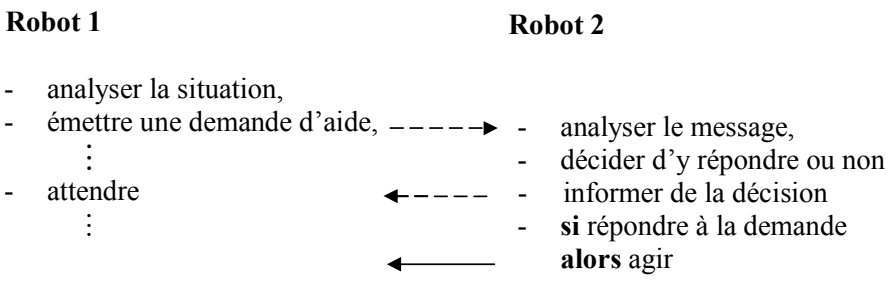


FIG. 1.9 – Scénario possible d'une communication de haut niveau

[12]On définit par *Blackboard* (Englemore & Morgan 1988), toute structure qui centralise les messages émis par les différents agents présents dans l'environnement et ce à la manière d'une base de données partagée (Hayes-Roth 1984), (Labidi & Lejouad 1993). L'objectif étant par la suite de pouvoir redistribuer les informations sur les agents qui en formulent la requête. L'idée de cette structure est de mieux maîtriser les flux de communication, et de pallier les problèmes de collisions éventuelles entre messages. Le tableau noir constitue donc une mémoire collective robuste et efficace pour les systèmes multi-agents.

[13]*Multiple Autonomous Robots for Transport and Handling Applications.*

[14]Le plus utilisé d'entre eux est sans doute le protocole TCP/IP (*Transmission Control Protocol/Internet Protocol*) utilisé par Internet.

1.2.1.3.2 Communication de bas niveau La première interrogation qui peut nous interpeller, est de savoir à partir de quel moment juge t'on qu'une communication est de bas niveau ? Effectivement, la limite n'est pas clairement établie entre une communication considérée comme de haut niveau et celle plutôt de bas niveau. Nous pouvons néanmoins affirmer que les communications de type bas niveau :

- s'établissent d'une manière très localisée autour de l'entité émettrice, et véhiculent des informations très simples,
- n'utilisent pas d'identifiant pour caractériser les émetteurs et les récepteurs de message. C'est-à-dire que les messages sont diffusés spontanément,
- n'utilisent pas de protocole particulier de négociation entre agents.

La figure 1.10 montre un mécanisme simple de communication bas niveau. Il est à noter que ce genre d'interaction entre entités, est généralement attribuée à des entités plutôt réactives, fonctionnant par stimulus-réponse, c'est-à-dire que le signal reçu par le robot 2 est considéré comme un stimulus directement exploitable sans qu'il y ait besoin d'un prétraitement particulier pour y répondre.

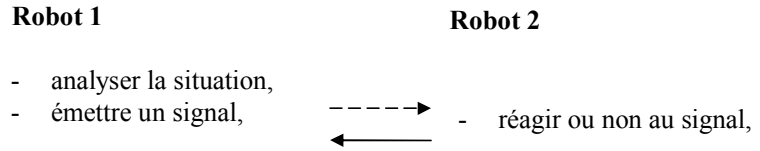


FIG. 1.10 – Exemple possible d'une communication de bas niveau

1.2.1.3.3 Communication indirecte Dans ce genre de communication c'est l'environnement des agents qui sert de médium de communication (Jung & Zelinsky 2000), (Hugues 2000), et cela à l'exemple des fourmis qui déposent des phéromones dans l'environnement pour guider leurs congénères vers les sources de nourriture. Cette communication indirecte peut être aussi illustrée par ce qui est appelé par Paul Grassé la *stigmergie*[15] (Grassé 1959).

[15]Du grec "stigma", piqûre, point, et "erglia" ou "ergia", action, efficacité, de "ergon", oeuvre. Désigne le fait, observé chez les Termites, que le résultat d'un travail exécuté par une collectivité d'individus, dirige à mesure ce travail, ce qui donne l'apparence d'une coordination planifiée. Il s'agit d'une oeuvre stimulante. Cette collaboration entre les fourmis a été décrite pour la première fois dans les années 50 par le biologiste français Pierre Paul Grassé (Grassé 1959). Il disait des insectes sociaux : "Ils ne dirigent pas leur travail, ils sont plutôt guidés par lui". À cette forme de stimulation, Pierre Paul Grassé a donné le nom de "stigmergie".

1.2.2 Différentes approches méthodologiques pour faire coopérer un ensemble de robots

Des problématiques voisines voire similaires sont parfois traitées par des communautés scientifiques différentes, cependant les approches et les outils utilisés pour aborder ces problèmes diffèrent d'une communauté scientifique à une autre. C'est le cas par exemple des investigations scientifiques traitant de la *coordination*, de l'*organisation*, de l'*analyse de fonctionnement*, du *contrôle*, de l'*auto-organisation* (Bonabeau & Theraulaz 1994), d'un ensemble d'entités autonomes, l'entité autonome signifiant ici autant, un agent logiciel, un être vivant, qu'un robot mobile.

Nous proposons, une décomposition en trois ensembles distincts d'approches traitant de l'étude d'un ensemble d'entités autonomes. Le premier ensemble concerne les approches orientées *sciences du vivant* (Éthologie des Animaux Sociaux), le deuxième ensemble concerne les approches émanant de l'*informatique* (Système Multi Agents), et le troisième ensemble correspond aux approches issues de l'*automatique* (Robotique Collective).

La richesse et la diversité de ces approches seront abordées dans ce qui suit. Il est à noter que les travaux des uns et des autres ne sont pas complètement indépendants. Au contraire, les différentes disciplines interagissent, se chevauchent, et se complètent sur bien des points, créant ainsi des ponts entre toutes ces disciplines, et des champs interdisciplinaires pour aborder la coopération entre entités autonomes (cf. figure. 1.11).

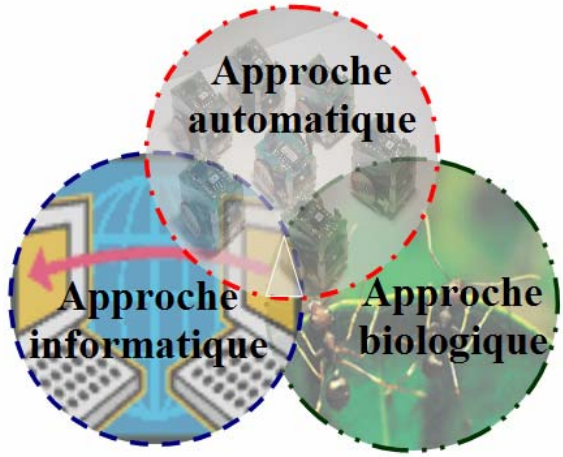


FIG. 1.11 – Méthodologies traitant de la coopération d'entités autonomes

1.2.2.1 Approche des Sciences du Vivant – Éthologie des Animaux Sociaux

La nature est un grand laboratoire qui innove, invente, teste, valide, améliore, et complexifie des systèmes vivants depuis des centaines de millions d'années (Darwin 1859). Les réalisations de la nature, plus spécifiquement celles liées au règne animal, sont par conséquent une réserve de solutions et une grande source d'inspiration pour les chercheurs[16]. D'ailleurs, d'innombrables réalisations atteintes par l'homme sont le fruit direct d'une bio-inspiration macroscopique[17] ou microscopique[18].

Depuis quelques décennies, cette bio-inspiration ne s'est plus contentée d'imiter les êtres vivants sur un plan purement matériel, mais elle s'est étendue à leurs aspects plus méthodologiques et organisationnels. Plus spécifiquement ce type d'inspiration, s'est focalisée sur la fascinante organisation des insectes sociaux tels que les fourmis, les termites, les guêpes ou bien les abeilles, pour contrôler un groupe d'agents coopératifs.

Notre vision centralisatrice du monde nous amène souvent à postuler l'existence par exemple d'une forme centralisée d'organisation, où une reine dirigerait fermement son petit royaume, ordonnant à ceux-ci de partir à la recherche de nourriture et à ceux-là de veiller sur le couvain larvaire, etc. Il est pourtant difficile d'imaginer un individu qui serait en possession de toute l'information nécessaire à la répartition des rôles et à la bonne marche de ces sociétés. Par conséquent, si les sociétés d'insectes sont dépourvues de superviseurs, comment peuvent-elles alors, coordonner l'ensemble de leurs activités, de manière aussi flexible, et présenter une telle plasticité dans l'ajustement de leurs performances pour réaliser des tâches collectives très abouties. Parmi ces tâches coopératives, nous citons par exemple : la construction de nids, le fourragement et le transport coopératif d'objets chez les fourmis (cf. figure. 1.12), la ventilation du nid chez les termites, la récolte du pollen chez les abeilles (Theraulaz & Bonabeau 1995).

De nos jours les éthologues[19] ne se contentent plus d'observer, classifier, rapporter des observations liées à des société d'insectes. Ils tentent à présent de domestiquer les mécanismes sous-jacents aux phénomènes observés, aux travers de modèles mathématiques, qui sont les plus représentatifs des phénomènes observés. Deneubourg dans (Deneubourg 1977) propose un modèle décrivant les étapes initiales de construction de nid chez les

[16] Encore faut-il savoir observer ces phénomènes et leur donner les bonnes hypothèses de fonctionnement !

[17] Telle que l'analogie entre le fonctionnement de l'oeil et celui d'un appareil photo.

[18] Tels que les mécanismes chimiques pour éviter le gel de certains liquides à des températures très basses qui est inspiré par des bactéries vivant à très basse température, ou l'analogie faite entre la structure interne des os et certaines constructions métalliques réalisées (e.g., la tour Eiffel).

[19] Le terme éthologie, vient du grec êthos, "moeurs" et logos, "discours, science", et a été proposé en 1854 par Geoffroy Saint-Hilaire, qui définit cette discipline comme "la science des comportements des espèces animales dans leur milieu naturel".

FIG. 1.12 – Tâche coopérative de transport d'objets par un groupe de fourmis

termites, Lioni dans (Lioni 1999) propose un modèle pour le cas du transport coopératif de proies chez les sociétés de fourmis. Beekman dans (Beekman et al. 2001) a modélisé avec une équation différentielle stochastique, l'activité de fourragement chez des fourmis qui utilisent un recrutement par pistes chimiques.

Les modélisations données par les éthologues sont généralement caractérisées par l'utilisation de *variables probabilistes* pour modéliser les différents états (activités) du système, et le passage d'un état à un autre se fait aussi en fonction de probabilités de transition entre états, qui peuvent être modifiées (renforcées ou atténuées) en fonction de l'occurrence de certains stimuli.

Les éthologues veulent effectivement faire émerger de leurs modèles (Bonabeau & Theraulaz 1994) qui décrivent le fonctionnement des sociétés d'insectes, des caractéristiques semblables aux phénomènes observés, c'est-à-dire une forme d'émergence de comportement de haut niveau à partir de comportements rudimentaires des entités constituant les sociétés d'insectes. Cette manière d'aborder la résolution de problèmes complexes, ouvre d'énormes potentialités pour beaucoup de disciplines liées aux sciences de l'ingénieur. Mais qu'en est il réellement, des connexions bidirectionnelles possibles entre ces travaux d'éthologues et les sciences de l'ingénieur ? Ces mêmes sciences, qui vont en général vers une volonté évidente de maîtriser les phénomènes, afin d'obtenir le minimum de risque de défaillance, et le maximum d'efficacité et de robustesse des phénomènes reproduits.

1.2.2.1.1 Liens existants des sciences de l'ingénieurs vers l'éthologie

L'utilisation de systèmes artificiels constituent de nouveaux moyens d'expérimentation pour les éthologues. Ainsi les hypothèses et les théories ne sont plus testées sur des modèles mathématiques, ou comme c'est l'habitude, sur des populations biologiques[20],

[20] Certes réelles mais sont généralement peu maniables et se prêtant peu aux expériences sur de grandes échelles.

mais sur des artefacts qui implémentent des traits saillants, soigneusement sélectionnés sur des ordinateurs ou sur d'autres supports physiques, tels que des robots mobiles, permettant ainsi des raffinements importants et une reproductibilité rapide et exhaustive (Arkin et al. 2001), (Kube & Bonabeau 2000), (Jost et al. 2004).

1.2.2.1.2 Liens existants de l'éthologie vers les sciences de l'ingénieur

Les études faites par les éthologues décrivant le fonctionnement social des sociétés d'insectes, sont autant de pistes et de voies potentielles pour les sciences de l'ingénieur. En effet, ces études peuvent nous servir pour proposer et élaborer des systèmes capables de résoudre de manière robuste, distribuée et adaptative les problèmes que nous leur soumettant. C'est-à-dire une résolution des problèmes à la manière des sociétés d'insectes. Parmi ces points nous pouvons citer :

- l'***effet de masse*** et la ***fusion d'informations*** : ces deux points liés d'une manière générale mais à des niveaux différents, nous indiquent l'importance intrinsèque du facteur **nombre**, tel que le nombre critique d'individus coopérants pour réaliser une tâche (Beekman et al. 2001), ou le nombre de capteurs nécessaires à l'obtention d'informations cohérentes et robustes de l'environnement. Dittmar dans (Dittmar & Delhomme 2001) dénote le caractère massivement parallèle des capteurs utilisés par les êtres vivants pour percevoir leur environnement (e.g., les capteurs de température, de pression existants sur la paume d'une main). En plus de ce fait, les informations (stimuli) régissant les comportements des êtres vivants sont la résultante de la fusion d'informations de plusieurs capteurs distincts. C'est par exemple le cas pour les méthodes de navigation utilisées par les oiseaux migrateurs (Kreithen 1983), qui utilisent à la fois des amers naturels tels que les étoiles ou la lune, ainsi que le champ magnétique terrestre, pour avoir l'information la plus fiable par rapport à la direction de leur lieu de migration.
  Ceci nous amène donc à favoriser (dans la mesure du possible), l'utilisation d'un grand nombre d'entités autonomes même frustres plutôt que d'utiliser un nombre restreint d'individus disposant d'une structure plus évoluée. L'adjonction à ces structures robotiques de dispositifs sensoriels de masse (fusion d'informations de plusieurs capteurs, appartenant soit au même robot, soit à plusieurs robots) est aussi un des moyens sûrs, pour rendre l'exécution des tâches d'autant plus fiable.

- la ***communication bas niveau*** : les échanges d'informations sont l'un des principes de base des êtres vivants (Dittmar & Delhomme 2001). Ce type de communication bas niveau peut être constatée chez les grandes sociétés de fourmis (Naka-

mura & Kurumatani 1997). Celles-ci utilisent des mécanismes de recrutement entre congénères simples (Bonabeau et al. 1999), tels que les contacts antennaires (communication directe) ou via des pistes de phéromones (communication indirecte). Ces mécanismes, une fois mis en place, arrivent à faire partager d'une manière très efficace des informations essentielles pour la réalisation des tâches coopératives.
Ce point nous invite à faire communiquer le groupe de robots, via des informations de bas niveau, et ce afin d'éviter toute forme de confusion liée aux communications.

- l'***aléatoire*** : les sociétés d'insectes ont sans doute besoin d'un certain degré d'erreur pour pouvoir s'adapter aux nombreuses perturbations qui peuvent affecter leur fonctionnement (Ashby 1960), (Chauvin 1982). Ceci peut être observé à travers le rôle de l'aléatoire pour la découverte des sources de nourritures chez les fourmis fourrageuses (Bonabeau et al. 1999), ou pour trouver le chemin le plus court pour rejoindre le nid (Beckers et al. 1992). En plus de cette notion d'aléatoire, il apparaît clairement que l'erreur est un élément très récurrent chez les sociétés d'insectes, et qu'elle joue le rôle d'un bruit de fond, qui fait que les sociétés d'insectes ne se bloquent jamais sur des situations conflictuelles et s'adaptent ainsi plus facilement aux environnements incertains et aux tâches complexes.
  La composante aléatoire devrait donc donner au système une meilleure flexibilité et robustesse face aux imprévus. Cependant, elle ne doit pas le mener vers des configurations qui pourraient mettre en péril le système et/ou l'exécution des tâches. Alors comment peut-on introduire cette composante aléatoire dans le contrôle de robot, sans pour autant risquer des dysfonctionnements ? D'une autre manière, comment peut-on utiliser par exemple les bruits ou toute forme de perturbations (que nous avons l'habitude de vouloir absolument éliminer du système !) inhérentes à toute expérimentation, afin de rendre les architectures de contrôles dédiées à la coopération de robots mobiles plus robustes et flexibles ?

- la ***division du travail*** et l'***évolution de la socialité*** : comment est-ce que les sociétés d'insectes arrivent-elles à faire en sorte que chaque individu de la société ait un rôle bien ciblé dans l'organisation globale de celle-ci ? Quels sont les mécanismes responsables de la formation des hiérarchies (Gervet 1967) (e.g., l'existence des récolteuses, nourrices ou gardiennes chez les abeilles, qui initialement pourraient occuper n'importe quel rôle cité ci-dessus) ?
  La maîtrise des processus d'affectation et de réaffectation des rôles caractérisant les sociétés d'insectes, ne pourra que renforcer la réactivité des systèmes distri-

bués que nous sommes amenés à concevoir, pour s'auto-reconfigurer d'une manière flexible, rapide et efficace, dans le but de pallier les aléas de l'exécution des tâches coopératives.

La ***motivation de comportements*** (Lorenz 1984), la ***synchronisation d'activités***, l'***auto-organisation***, l'***auto-assemblage***, la ***stigmergie***, l'***imitation*** (mimétisme) sont autant de mécanismes étudiés par les éthologues qui pourraient trouver un terrain d'applications très fertile dans les sciences de l'ingénieur, et ce pour contrôler entre autres, des systèmes complexes d'une manière plus flexible et plus efficace.

1.2.2.2 Approche Informatique – Système Multi Agents

L'intérêt de l'informatique aux notions d'agents intelligents autonomes, a commencé avec les premiers travaux en Intelligence Artificielle "IA" (fin des années 50). Vers les années 70, ces même travaux ont donné naissance à l'Intelligence Artificielle Distribuée "IAD" (Lenat 1975), (Steels 1990), (Avouris & Gasser 1992) qui traitent pour leur part de l'intelligence produite par un ensemble d'agents coopératifs. De l'IAD a émergé par la suite la notion de Système Multi-Agents "SMA" qui résout des problèmes complexes en les décomposant en un ensemble d'agents autonomes. La spécificité des approches SMA par rapport à celles attribuée à l'IAD, est qu'elle impose beaucoup plus de contraintes (structurelles et décisionnelles) par rapport aux entités constituant le système (Labidi & Lejouad 1993).

Les travaux liés au contrôle de systèmes complexes en les décomposant en plusieurs sous-systèmes (agents), ont été abordés dès la fin des années 60 par Michie et Chambers (Michie & Chambers 1968). Les auteurs proposent de commander un pendule inversé par un apprentissage approprié par essai-erreur via la mise en coopération d'un ensemble d'agents (BOX) qui représentent chacun une partition de l'espace d'états atteignable par le système[21].

Drefus dans (Dreyfus 1992, pp.231-281) et Searle dans (Searle 1980) critiquent d'une manière générale le manque d'*incarnation* (*embodiment* (Brooks 1990), voir Annexe A) des programmes informatiques représentant entre autres les SMA. Ceci engendre selon eux, une incapacité de ces programmes (agents) à interagir de manière significative avec

[21] "*The BOXES learning algorithm partitions the state space into regions according to how each dimension of the space is discretised. Each box represents a region of the problem space. In the pole and cart problem, there are four dimensions, one for each state variable (position and velocity of the cart, angle and angular velocity of the pole)*".

leur environnement et surtout avec leurs semblables (les autres agents). Dans le cadre du contrôle d'entités autonomes situées, l'approche développée par les informaticiens favorise généralement plus l'aspect conceptuel du problème à celui de la prise en compte effective des contraintes physiques caractérisant les agents dans leur environnement. En effet, parmi les tâches coopératives génériques traitées en simulation par cette communauté telles le fourragement, le tri d'objets ou la tâche de poussée d'objets, nous notons le fait que leur exécution se fait généralement sans contraintes physiques explicites. De ce fait, le passage de l'aspect conceptuel vers l'implantation effective des solutions proposées, ne se fait pas toujours hélas sans encombre, car la physique de la manipulation (e.g., soulever, déplacer ou déposer des objets) ou la dynamique de mouvements des entités robotiques ne sont généralement pas modélisées d'une manière précise.

L'autre exemple, caractérisant à la fois, le monde de l'informatique et celui de la coopération d'agents, est celui du génie logiciel. Ce domaine produit des logiciels complexes en faisant coopérer un ensemble d'agents logiciels : classe, objets, ou toute autre forme de technologie basée sur une conception orientée objets[22]. Cette démarche objet qui tend donc à décomposer le système complexe en plusieurs modules (/ou objets) faiblement couplés, a permis d'envisager autrement la conception des logiciels, passant de la notion de programme à celle d'organisation (Erceau & Ferber 1991).

Les secteurs liés au génie logiciel ou aux réseaux de communication (Bonabeau et al. 1999) montrent des succès certains dans leurs applications les plus diverses, et cela malgré la complexité des systèmes traités. Il est à noter cependant que la nature de ces systèmes, tendent à spécifier **exactement** : les agents qu'ils manipulent (attributs, méthodes, protocoles...), les ressources partagées, les communications et la hiérarchie existantes entre agents. De plus, la nature des interactions entre agents est quasi déterministe. Tout ceci permet donc de réduire considérablement voire éliminer complètement tout type d'aléa quant aux scénarios prévus pour le fonctionnement de ces systèmes complexes.

Les travaux attribués aux informaticiens, comme nous pouvons le constater ci-dessus, traitent des agents sous plusieurs formes, qu'ils soient agents virtuels (plutôt orienté logiciel) ou bien situés dans un environnement (tel qu'un routeur de communication ou un robot mobile). Les apports méthodologiques et conceptuels potentiels qui peuvent

[22]Tels que : l'ActiveX ou l'OLE "*Object Linking and Embedding*" qui sont des technologies Microsoft qui permettent de créer et d'éditer des objets partageables par de plusieurs applications.

être apportés par les informaticiens (ou plus spécifiquement par la communauté SMA) pour faire coopérer un groupe de robots mobiles, sont donc certains.

1.2.2.3 Approche Automatique – Robotique Collective

Parler de contrôle en automatique est avant tout synonyme d'existence d'un modèle décrivant le système à contrôler[23]. Même si le système n'est pas trivial à modéliser, alors une estimation de ces paramètres est effectué pour avoir une représentation la plus proche possible du système à contrôler (Khalil & Dombre 1999). Le contrôle d'un robot mobile se fait par exemple en utilisant l'un des modèles qui le représente soit géométriquement, soit cinématiquement ou bien dynamiquement. Le choix d'un modèle se fait en fonction du type de contrôle et de la tâche qu'on veut lui faire exécuter (Laumond 2001), (Pruski 1996), (Tournassoud 1992). Ces modélisations ont pour avantage majeur de pouvoir synthétiser mathématiquement un contrôleur approprié pour le système automatique et ainsi adapter, améliorer le contrôle en fonction des performances désirées de l'asservissement[24](précision, rapidité, robustesse, etc.).

Ainsi, si nous restons dans le même ordre d'idée, faire coopérer un ensemble de systèmes robotiques, est par conséquent aussi synonyme d'existence de modèles régissant le système multi-robots[25]. Effectivement, c'est ce qui est appliqué généralement dans le cadre de la robotique coopérative, où chaque robot est modélisé de telle sorte à ce qu'il intervienne dans le modèle global du système multi-robots, et que les interactions entre les systèmes (robots, objet à manipuler, etc.) se fassent par une modélisation appropriée des couplages.

Khatib dans (Khatib et al. 1995) utilise une architecture décentralisée pour contrôler deux robots mobiles équipés de bras manipulateurs "*vehicle/arm*" (cf. figure. 1.13(a)), et ceci pour manipuler d'une manière coopérative des objets. Nous dénombrons deux types de coopération dans ce système robotique :

[23]Ceci bien évidemment en excluant les nouveaux types de contrôle émergents, basés par exemple sur de la logique floue, des réseaux de neurones ou de l'apprentissage par renforcement, qui n'ont pas besoin de modèle explicite du système à contrôler (cf. §6.1, page 124).

[24]Il est à noter aussi que contrôler un robot mobile au sens automatique-robotique du terme passe aussi par la prise en compte de modèles tels que ceux liés : aux frottements, aux bruits, à l'holonomie ou non des robots, aux couples maximums engendrés par les moteurs des roues, etc. C'est-à-dire tenir compte effectivement, des contraintes physiques et/ou structurelles des robots, ainsi que de leurs interactions avec leur environnement.

[25]Dans ce qui suit nous nous focaliserons sur la coopération de robots mobiles qui manipulent (i.e., déplacent, soulèvent, déposent) une ressource commune.

- la première consiste en la coordination des mouvements entre la plate-forme robotique mobile (pour les mouvements lents et peu précis) et le mouvement du bras manipulateur (pour les mouvements rapides et précis), pour avoir une coopération efficace entre ces deux structures,
- la seconde correspond à la coopération des deux "*vehicle/arm*" pour la manipulation commune d'objet. Khatib a considéré l'ensemble des structures robotiques en contact comme une seule structure redondante. Les forces internes du système (i.e., les forces connectant les effecteurs des bras manipulateurs avec l'objet à manipuler) sont modélisées comme des articulations virtuelles. Pour avoir un modèle cohérent des articulations virtuelles, il faut néanmoins considérer que le contact est rigide et s'effectue sans glissement, ce qui est très difficile à respecter. Afin de rectifier les erreurs inhérentes à la modélisation, Khatib a prévu une communication entre les robots.

(a) Coopération de deux PUMA 560 placés sur des plate-formes mobiles

(b) Coopération mixte entre des robots et un humain

FIG. 1.13 – Coopération de robots au sens automatique-robotique

Hirata dans (Hirata et al. 2000) propose un algorithme décentralisé de contrôle de deux robots holonomes, qui doivent déplacer un objet sans connaissance préalable ni de la position des autres robots, ni de la géométrie de l'objet à déplacer. Le déplacement de l'objet est effectué en utilisant un robot leader et un autre suiveur. Le leader connaissant à tout moment l'objectif à atteindre, il impulse alors une force à travers l'objet à déplacer pour désigner au robot suiveur la direction désirée. Il est à noter que les robots disposent de capteurs d'effort pour détecter le module et l'orientation de la force appliquée via l'objet. Ce principe a été développé par la suite dans (Hirata et al. 2002) pour réaliser une coopération mixte entre des robots et un humain qui impulse les forces et les moments nécessaires pour guider les déplacements de l'objet (cf. figure. 1.13(b)).

Sasaki dans (Sasaki et al. 1995) traite aussi le cas du transport (soulever-acheminer) d'un objet par plusieurs robots. L'objectif de ces travaux étaient de trouver la géométrie optimale des robots en dessous de l'objet à transporter, et ce afin de garantir à la fois, la maximisation de la stabilité de l'objet posé sur les robots et la minimisation de l'énergie fournie par les robots. La figure 1.14 donne une idée des différentes étapes nécessaires pour réaliser la tâche de soulever-acheminer selon Sasaki.

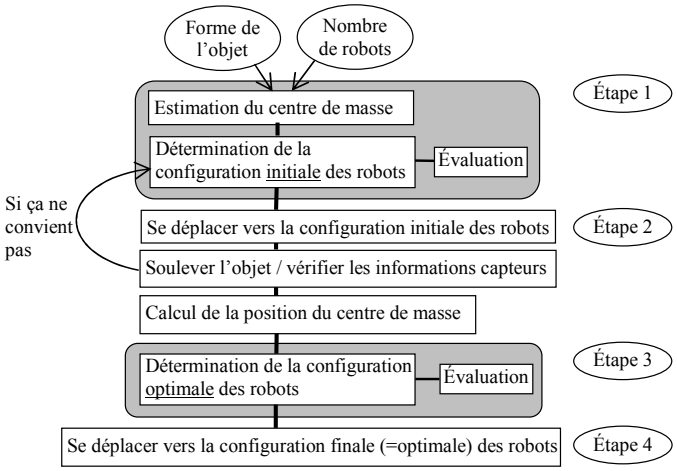


FIG. 1.14 – Algorithme pour trouver la configuration optimale pour saisir un objet

Stilwell dans (Stilwell & Bay 1993) utilise une architecture décentralisée avec un leader pour contrôler un ensemble de robots mobiles non holonomes. Les robots se positionnent en dessous d'une palette chargée pour la déplacer d'une manière stable. Chaque robot est modélisé comme une masse et le contact existant entre les robots et la palette est modélisé par un ressort ayant un amortissement omni-directionnel, ce qui donne une approximation du second ordre à la dynamique des forces transmises entre les robots et la palette à déplacer.

Ahmadabadi dans (Ahmadabadi & Nakano 2001) propose une architecture de contrôle distribuée nommée *constrain-move strategy* pour soulever, orienter, déplacer, déposer, une charge selon une trajectoire contrainte. Le scénario utilisé par Ahmadabadi est inspiré directement des lois de la mécanique, et il consiste à contraindre l'objet à déplacer par un ensemble de robots mobiles (disposant de bras manipulateurs planaires), afin que l'objet ne puisse être déplacé que selon une direction privilégiée. Une fois que ceci est vérifié, il suffit qu'un ou plusieurs robots applique une force dans le sens où l'objet n'est

pas contraint afin de le déplacer. L'ensemble des étapes nécessaires pour réaliser cette tâche est implémenté en utilisant une architecture à base de subsomption (Brooks 1986), où les robots communiquent entre-eux et connaissent à tout moment leur position et orientation par rapport à l'objet à déplacer.

Il est évident que l'approche prise par les roboticiens pour faire coopérer un ensemble d'entités robotiques, est celle qui permet le mieux de faire une analyse rigoureuse et constructive des solutions proposées. En effet, cette approche permet de quantifier les performances atteintes par le système, au sens automatique du terme : précision, rapidité, robustesse, stabilité, etc. Ceci est effectif vue la disponibilité des modèles régissant le système multi-robots. Cependant, ce qui est flagrant dans ce genre d'approches, c'est que le nombre de robots en coopération ne dépasse pas quelques individus (deux à quatre robots au maximum), car les modèles et les scénarios proposés deviendraient nettement plus complexes à gérer. Ce qui est notable aussi dans ce genre d'approche, c'est l'utilisation de modèles pour représenter les interactions entre systèmes (robot-robot, robot-objet). Par conséquent, la pertinence du contrôle est étroitement liée à la précision de ces modèles d'interaction. Cependant, ces modèles peuvent s'avérer très peu fiables, surtout si l'environnement est hautement dynamique, et que les robots doivent par exemple se reconfigurer plusieurs fois pour réaliser efficacement leurs manipulations.

Le principal verrou faisant donc que l'approche automatique-robotique ne soit pas en mesure dans l'immédiat de donner des solutions adaptées, viables et surtout exactes au contrôle coopératif d'un grand nombre de robots mobiles (plusieurs dizaines), est lié à la relative pauvreté des modèles existants définissants les interactions entre les robots et leur environnement.

1.3 Conclusion

Ce chapitre nous a permis d'exposer les principales spécificités des systèmes multi-robots, et ceci en partant des caractéristiques structurelles des robots élémentaires, jusqu'aux mécanismes induisant la coopération entre robots.

Ce chapitre a été aussi l'occasion de présenter les principales approches et méthodologies (sciences du vivant, informatique, automatique) qui partagent des problématiques communes à la coopération d'un ensemble d'entités élémentaires d'une manière générale, et d'un groupe de robots mobiles d'une manière plus spécifique. Ceci nous a donc permis de nous rendre compte des interconnexions possibles entre les différentes disciplines, et

de faire ressortir les points forts et points faibles inhérents à chacune d'elles, et ceci pour le cas précis de la coopération d'un groupe de robots mobiles.

L'idée à présent est de pourvoir trouver l'équilibre entre les apports (approches, méthodes, outils, etc.) de ces différentes disciplines, pour pouvoir proposer le contrôle le plus viable et approprié pour faire coopérer un groupe de robots mobiles.

Chapitre 2

Catégorisation des architectures de contrôle

Résumé : *Ce deuxième chapitre a pour objectif de présenter les différentes classes d'architectures de contrôle pour robots mobiles développées dans la littérature. En l'occurrence, nous détaillerons les spécificités des architectures centralisées par rapport aux architectures distribuées et celles des architectures cognitives par rapport aux architectures réactives. Ce chapitre traitera aussi les principes de bases des architectures comportementales ainsi que la notion d'émergence de comportements propres aux architectures réactives. L'étude de ces différentes architectures va nous permettre de fixer notre cahier des charges à suivre pour l'élaboration d'un contrôle adapté à la coopération d'un groupe de robots mobiles minimalistes.*

2.1 Architecture de contrôle centralisée versus distribuée

L'un des points essentiels à fixer avant l'élaboration d'une architecture de contrôle pour un groupe de robots mobiles, concerne le choix de centraliser le contrôle du système multi-robots, ou de le distribuer sur les entités robotiques (Cao et al. 1997).

2.1.1 Architecture de contrôle centralisée

Une architecture de contrôle est dite centralisée, quand une partie ou la totalité, des boucles sensorielles et/ou décisionnelles de chaque entité robotique est délocalisée par rapport à sa structure physique, et ce pour être gérée au niveau d'une unité centrale, appelée aussi *superviseur* (Jones & Snyder 2001), ou planificateur central (Noreils 1993) (Causse & Pampagnin 1995). Une architecture de contrôle centralisée peut être imagée par un chef d'orchestre (le superviseur) qui dirige ses musiciens (les robots mobiles).

Les règles des compétitions des robots footballeurs établies par la FIRA[1] prévoient, pour certaines catégories de compétition telle que celles nommées MiroSot[2], un contrôle centralisé des robots footballeurs (cf. figure. 2.1(a)). Chaque équipe dispose d'un ordinateur qui possède une vision globale de la scène et qui établit en conséquence une commande appropriée pour chaque robot de son équipe, et qu'il transmet via une communication hertzienne (Huang et al. 2001). Brian Gerkey dans (Gerkey & Mataric´ 2002*a*) utilise quant à lui une méthode de contrôle pseudo-centralisée qui permet une coopération efficace entre robots mobiles pour pousser un objet volumineux. Ce système nommé pousseur-observateur "*pusher-watcher*" (cf. figure. 2.1(b)), permet à un robot observateur (*watcher*) qui joue aussi le rôle du superviseur[3] de guider les actions des robots pousseurs (*pushers*) qui jouent ici principalement les rôles d'effecteurs. Il est à noter que la communication entre le robot observateur et les robots pousseurs se fait via un protocole de communication haut niveau (cf. §1.2.1.3.1, page 21).

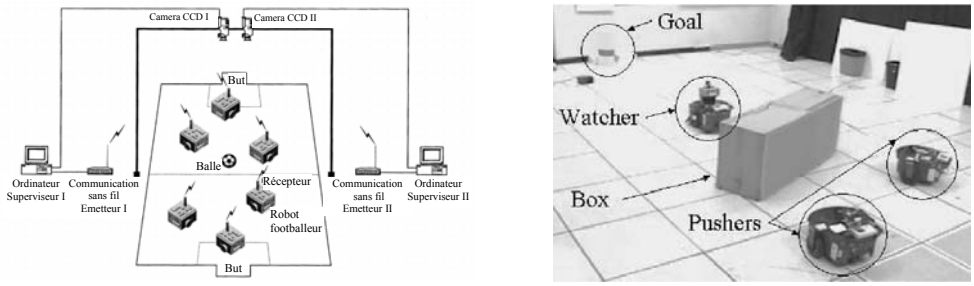
2.1.2 Architecture de contrôle décentralisée-distribuée

Devant la complexité grandissante des systèmes à automatiser, l'idée de distribuer le contrôle sur plusieurs entités coopérantes dans l'objectif de réaliser une tâche principale est très attrayante. La distribution du contrôle permet, si elle est concluante, de briser la complexité inhérente au système, et conduit ainsi à une mise en oeuvre du contrôle beaucoup plus simple et surtout plus près du modèle du système. En effet, l'obtention

[1] *Federation of International Robot-soccer Association*, `http://fira.net`

[2] *Micro Robot World Cup Soccer Tournament.*

[3] Vu qu'il a en sa possession toutes les informations nécessaires pour guider la tâche de pousser d'objet.



(a) Robots footballeurs (b) Pousseur-observateur

FIG. 2.1 – Tâches coopératives contrôlées d'une manière centralisée

d'un modèle pour un système complexe est toujours une entreprise très fastidieuse et trop peu fiable (Bonabeau & Theraulaz 1994, chapitre. 8) (Goldberg & Mataric´ 1999), (Sugawara & Watanabe 2002), (Lerman et al. 2001).

Contrairement à une architecture centralisée, une architecture de contrôle distribuée, consiste à faire en sorte que chaque robot mobile ait une autonomie sensorielle (capteurs, récepteurs) et décisionnelle (unité de calcul et de traitement de l'information) totale (Brooks 1986), (Arkin 1992), (Kube & Bonabeau 2000). Ceci constitue sans nulle doute un grand challenge technologique et conceptuel. En effet, les autonomies souhaitées requièrent à la fois des technologies sensorielles et calculatoires appropriées, pour qu'elles puissent être embarquées aisément sur les robots mobiles, mais elles requièrent plus spécifiquement des approches méthodologiques et conceptuelles adaptées à ce type d'architectures de contrôle, et ce afin de pallier entre autres :

- le manque d'informations globales sur l'environnement. Effectivement le fait que le robot ne dispose que d'informations locales, peut mener :
  - à de *grands coûts calculatoires* pour la coordination et la résolution des conflits. En effet, à cause de leurs connaissances locales et donc partielles du problème, les robots peuvent être amenés à se concerter, négocier ou s'échanger leurs plans d'actions (Leroy 1998) afin de prendre des décisions cohérentes et éviter ainsi, par exemple, des conflits d'accès à des ressources,
  - à la *sous-optimalité des décisions*, car la distribution et la coordination des tâches résultent de prises de décision locales. Par conséquent, ces décisions peuvent être sous-optimales par rapport à des critères globaux.
- le manque de puissance de calcul embarquée sur chacune des entités autonomes,
- la non-existence d'un superviseur qui coordonne l'ensemble des entités, vu qu'un superviseur est censé disposer d'une vision globale du déroulement des tâches en

cours d'exécution.

La distribution du contrôle sur l'ensemble des entités robotiques peut aussi être dictée tout simplement par des aspects économiques (coût plus faible en distribuant le contrôle), ou bien par le fait que la configuration de l'environnement rende difficile voire impossible l'existence d'un superviseur pour contrôler l'ensemble du groupe de robots, car celui ci pourrait ne pas disposer d'une vue d'ensemble du système multi-robots (e.g., de la conquête spatiale : exploration de planètes lointaines). Le tableau 2.1 résume les principales caractéristiques des architectures de contrôle distribuées et les contraintes qui en résultent.

Sachant que ... !	Comment obtenir ... ?
- la connaissance et le contrôle sont distribués,	- la coordination des tâches (évitement de conflits),
- les agents ont des connaissances partielles sur leurs activités respectives et sur l'environnement,	- la parallélisation des tâches,
- certaines tâches nécessitent plusieurs agents pour être réalisées,	- la non redondance des tâches,
- les tâches changent de façon imprévisible, en fonction des aléas d'exécution ou des requêtes d'un opérateur,	- la robustesse du groupe aux pannes,
- certaines tâches sont réalisées de façon concurrente.	- le partage des ressources, - la non persistance des conflits (résolution des conflits).

TAB. 2.1 – Caractéristiques et problématiques posées par un contrôle distribué

2.1.3 Faut-il centraliser ou distribuer le contrôle ?

Il est difficile de se prononcer sur la supériorité de l'une ou de l'autre de ces deux types d'approches, en l'occurrence entre celle qui supervise complètement le contrôle des robots mobiles et celle qui distribue le contrôle sur des robots épars.

La conception centralisée du contrôle est sans doute celle qui est la plus aisée à imaginer pour nous les hommes, ceci est sans doute due à un héritage (poids culturel) lié à

notre conception qu'il y a toujours une hiérarchie fonctionnelle des choses, *ordonnateur-exécutant*. Il est à noter que l'ordonnateur est **censé** connaître beaucoup plus que l'exécutant qui ne détient pour sa part que les informations qui lui serviront pour exécuter les ordres. D'ailleurs, c'est ce modèle qui est le plus souvent rencontré dans l'industrie, et dans les organisations sociales et administratives. Les stratégies de contrôle proposées par ces architectures centralisées impliquent donc une connaissance plus au moins globale de l'évolution des différents éléments du système, et à partir de toutes ces informations, la commande la plus appropriée est calculée puis appliquée au système. Un contrôle centralisé considère par conséquent le groupe de robots comme un "super-robot", dont l'espace d'état correspond au produit des espaces d'états de tous les robots le constituant. Cette analogie n'est pas sans conséquence, car elle implique que l'espace d'état du système multi-robots peut être très grand (en fonction du nombre de robots), ce qui induit que le contrôle centralisé de la moindre tâche devient rapidement inextricable. Les architectures centralisées peuvent aussi être non utilisables du fait de verrous technologiques, comme l'exemple de l'exécution en temps réel d'une tâche coopérative demandant l'intervention d'un grand nombre d'entités robotiques et dont les vitesses d'évolution seraient trop importantes. Ceci entraînerait inéluctablement des goulots d'étranglement au niveau des communications reliant le superviseur aux entités robotiques.

Une architecture de contrôle centralisée est bien souvent synonyme d'approche *Top-Down* alors qu'une approche distribuée est synonyme d'approche *Bottom-Up*. On peut plus facilement imaginer l'élaboration d'une architecture de contrôle avec une complexité ascendante maîtrisée en utilisant une approche *Bottom-Up* plutôt qu'avec une approche *Top-Down* qui risque de ne pas être compatible lors de l'ajout d'autres contraintes (Brooks 1990).

Il est à noter qu'il existe aussi dans la littérature des architectures hybrides jumelant à la fois des aspects centralisés et des aspects distribués (Fukuda et al. 2000), le contrôle centralisé est appliqué pour donner les stratégies et les ordres généraux des tâches à exécuter, et le contrôle distribué prend la main pour la navigation et les actions locales.

2.2 Architectures de contrôle cognitives versus réactives

Après avoir présenté les deux principales approches de contrôle pour un groupe de robots mobiles, en l'occurrence les architectures centralisées versus les architectures distribuées, nous nous focalisons à présent sur les principes de fonctionnement des architectures distribuées, car c'est l'approche que nous avons adoptée pour le contrôle de

nos robots mobiles (cf. chapitres IV et V). Néanmoins, il reste à préciser le degré de sophistication des robots que nous allons utiliser. La littérature dénote deux grands écoles (/ou courants), le cognitif et le réactif. Ces deux écoles s'opposent principalement sur le degré d'intelligence à embarquer sur chacune des entités constituant le système multi-robots (Bonabeau & Theraulaz 1994, p.163). Pendant que l'école cognitive prêche l'utilisation d'entités robotiques coopérants d'une manière très élaborée et évoluée (ce qui implique des entités élémentaires très sophistiquées), l'école réactive prône plutôt l'utilisation d'entités robotiques les plus frustres possibles (cf. figure. 2.2).

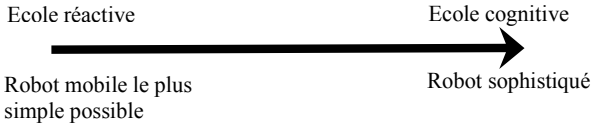


FIG. 2.2 – Degré d'intelligence exigé pour les robots de chaque école

Dans les sections qui suivent, les caractéristiques conceptuelles de chacune de ces deux écoles sont abordées et traitées.

2.2.1 Les architectures de contrôle cognitives

L'élaboration d'architectures de contrôle cognitives intègrent plusieurs caractéristiques et principes de haut niveau. En effet, le contrôle d'un robot cognitif peut inclure des notions d'*intentionnalité*, de *rationalité*, d'*engagement*, d'*adaptabilité*, de *savoir-faire*, de *croyances*, de *planification*, de *communication haut niveau*, de *représentation symbolique de l'univers*, de *reconstruction du monde* (Brooks 1990), (Alami et al. 1998*b*), (Labidi & Lejouad 1993), (Fiorino 1999). Tous ces éléments font que l'entité robotique doit continuellement traiter un grand flux d'informations qui lui parvient, soit via ses propres capteurs, soit via les autres entités autonomes (communication de haut niveau). Ceci peut donc entraîner une grande lourdeur de fonctionnement de ces entités robotiques.

La gestion des situations de conflits de ressources et de conflits spatiaux-temporels sont des exemples intéressants pour illustrer certains processus intervenants dans les stratégies de contrôle cognitives. La figure 2.3 montre un cas simple d'un agent R1 occupant la ressource S2 et attendant que l'agent R2 libère la ressource S0. Or si de même l'agent R2 attend la libération de S2 par R1, un interblocage apparaît. L'un des agents peut alors devenir "coordonnateur" (Qutub 1998) et proposer une solution (e.g., R2 en S1, R1 en S0 et R2 en S2).

La figure 2.4(c) montre un autre cas d'interblocage plus complexe : les objectifs à atteindre par les robots sont donnés par la figure 2.4(a). On y remarque que les robots

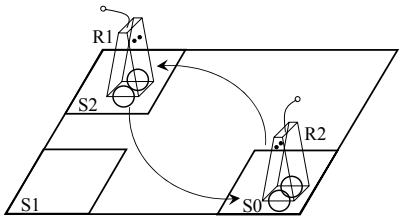


FIG. 2.3 – Situation d'interblocage

planifient individuellement leur trajectoire, ce qui implique une connaissance plus au moins précise de l'environnement qui les entoure (obstacle, objectif à atteindre, etc.). Le fait que les robots aient planifié leur trajectoire avec une petite portée temporelle (cf. figure. 2.4(b)) entraîne donc une situation d'interblocage (cf. figure. 2.4(c)). On remarque que l'interblocage a pu être évité (cf. figure. 2.4(e)) en augmentant d'une part la portée de planification propre à chaque robot (cf. figure. 2.4(d)), et d'autre part en communicant les *plans d'actions* (Leroy 1998), (Aguilar 1997) (correspondants ici aux trajectoires planifiées) propres à chaque robot. Ceci permet donc de trouver le meilleur compromis, afin de synchroniser et coordonner leurs mouvements.

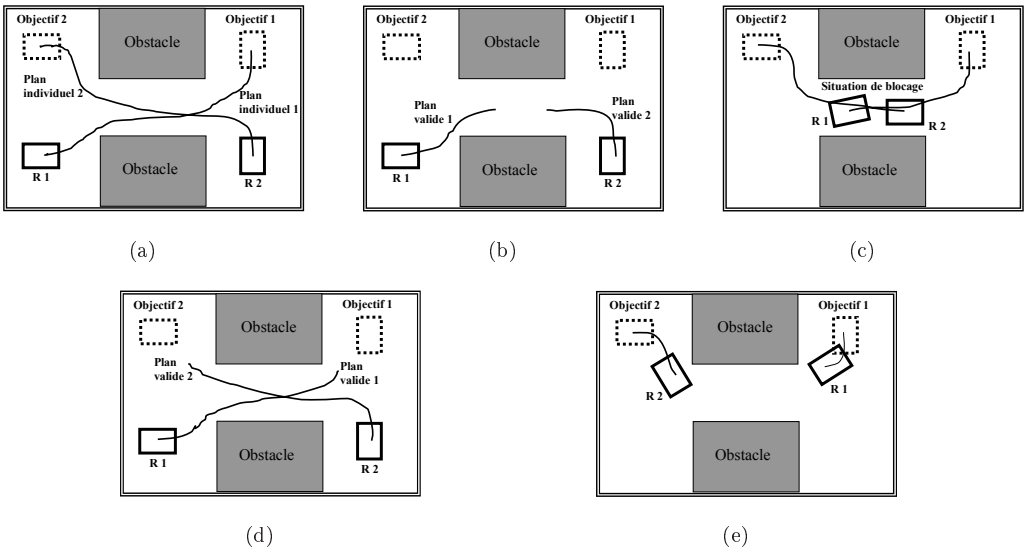


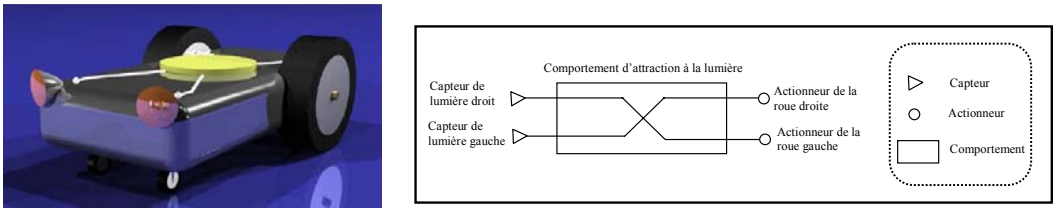
FIG. 2.4 – Importance de la portée de la planification

2.2.2 Les architectures de contrôle réactives

Le contrôle réactif d'un robot mobile ne veut pas dire forcément le dénuder complètement de tout dispositif sensoriel ou d'actionnement sophistiqué. Cependant, ce qui est important à respecter, c'est le lien plus au moins direct "*direct-mapping*" entre les informations capteurs et les primitives d'actionnement, et cela contrairement à une architecture de contrôle cognitive, qui tend à établir avant tout, une correspondance entre le monde externe réel et ça représentation symbolique interne au robot, afin d'établir les actions à entreprendre (Albus 1991).

Bien évidemment les architectures de contrôle réactives n'utilisent ni de planification de tâches ou de trajectoires, ni de reconstruction de l'environnement, ni même de communications haut niveau, c'est-à-dire tous les outils de base d'une architecture de contrôle cognitive.

La machine de Braitenberg (cf. figure. 2.5(a)) (Braitenberg 1984) est une illustration très significative de ce qui est une machine purement réactive. Le *mapping* direct entre les informations capteurs et les actions des moteurs se fait via une relation de *stimulus-réponse*. La figure 2.5(b) montre les liens existants, entre les stimuli parvenant des capteurs et les actions des roues droite et gauche du robot mobile, et ce afin qu'il puisse se diriger vers la source de lumière. La figure 2.5(b) montre, que dans le cas où c'est le capteur gauche qui reçoit le plus de lumière alors la roue droite aura une vitesse supérieur à celle de gauche. Ceci induit finalement le fait que le robot se dirige vers la source de lumière (en l'occurrence vers la gauche).



(a) (b)

FIG. 2.5 – Machine de Braitenberg

Le véhicule de Braitenberg exhibe plusieurs autres comportements, soit en combinant différemment les connexions existantes entre les capteurs et les effecteurs, soit en ajoutant à la structure de base des connexions inhibitrices ou excitatrices qui relient les capteurs aux effecteurs. Les comportements qui résultent de ce type de structure n'ont pas été programmés explicitement comme pourrait le croire un observateur extérieur, mais sont bien le résultat de propriétés *émergentes* (cf. §2.2.2.2) de l'architecture de contrôle.

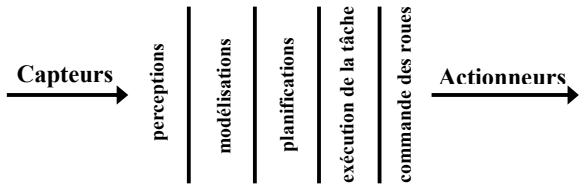
Les caractéristiques émergentes mentionnées ci-dessus sont attribuées au fonctionnement d'un seul robot contrôlé d'une manière réactive. De nombreuses propriétés émergentes toutes aussi inattendues, peuvent aussi être la résultante d'interactions d'un grand nombre de robots réactifs. En effet, les architectures de contrôle réactives se prêtent bien au contrôle d'un grand nombre de robots mobiles. Chaque individu dans ce type d'architecture ne possède que des informations très localisées sur son environnement et utilise une stratégie de stimuli-réponses pour agir sur son environnement immédiat (à l'image des comportements sociaux chez les sociétés d'insectes). Ceci implique par conséquent un contrôle du groupe de robots complètement distribué. Ce type de stratégie de coopération transfère par conséquent la complexité du contrôle, de l'entité élémentaire (à la base simple) vers le groupe (au travers de sa grande dynamique d'interaction, ses stratégies de coopération, ses mécanismes de gestion des conflits, etc.) faisant émerger ainsi une forme d'*intelligence de groupe* (Bonabeau et al. 1999).

2.2.2.1 Les architectures comportementales réactives

Il existe chez les animaux des *comportements* innés que l'on peut qualifier d'atomiques (ou primaires) dans le sens où ils sont difficilement réductibles à d'autres comportements directement observables. De manière générale, tous les actes moteurs (coordination d'un ensemble d'activités musculaires) des animaux sont compris dans cette catégorie. Ces comportements constituent les briques de base avec lesquelles des comportements de plus haut niveau peuvent être construits et décrits (Anderson & Donath 1990).

Pour l'élaboration d'architectures de contrôle réactives de plus en plus complexes, Brooks dans (Brooks 1986) propose de décomposer le comportement global du robot, non plus en blocs fonctionnels, comme il est de tradition de le faire (cf. figure. 2.6(a)), mais en un ensemble d'entités élémentaires appelées *comportements* (cf. figure. 2.6(b)). Cette nouvelle décomposition a comme principaux objectifs d'avoir la constructibilité "*buildability*" aisée de l'architecture de contrôle ainsi qu'une testabilité "*testability*" facile des comportements élémentaires (Brooks 1986). Ceci est rendu possible en isolant les comportements individuellement afin de les tester et de les adapter jusqu'à ce qu'ils soient les plus appropriés à ce quoi ils sont dédiés (e.g., la manipulation d'objets, l'évitement d'obstacles, l'attraction à la lumière, etc.). Cette construction ascendante "*Micro-Macro*" ou "*Bottom-Up*" des architectures de contrôle, est très attrayante, car elle permet d'obtenir des comportements de plus en plus complexes, simplement en adjoignant à l'architecture de base d'autres comportements élémentaires. Néanmoins, le fait qu'il y ait plusieurs comportements (fonctionnant généralement en parallèle) qui cohabitent dans une même

structure de contrôle, impose l'adoption de mécanismes appropriés de coordination entre comportements (cf. chapitre III).



(a) Décomposition verticale

(b) Décomposition horizontale

FIG. 2.6 – Décompositions possibles d'une architecture de contrôle (Brooks 1986).

2.2.2.2 Émergence de comportements

L'émergence est une notion, énormément décrite et controversée dans la littérature (Arkin 1998, p.105). Elle est étudié autant en biologie, en physique, que dans le cadre des SMA réactifs. Dans le cadre de la robotique, un comportement est dit émergent s'il n'a pas été explicitement programmé, ce qui signifie qu'un observateur, et même des fois le concepteur, n'est pas en mesure de décrire le lien causal entre l'effet global observé et les interactions locales au sein de l'architecture de contrôle.

L'émergence est donc le résultat d'interactions entre deux ou plusieurs agents (comportements, molécules, robots, insectes, etc.) qui fait apparaître des fonctionnalités inattendues (trop complexes pour être prédites aisément). En effet, selon la démarche holistique[4], la fonctionnalité accomplie par un groupe d'éléments n'est pas forcément la simple somme des fonctionnalités de ces composants, mais la résultante de l'action commune de ces individus interdépendants. Il est à noter que cette résultante peut dépasser la somme des actions individuelles (Forrest 1991).

[4]L'holisme (du grec holos : entier) désigne l'attitude épistémologique consistant à considérer qu'un système complexe est une entité possédant des caractéristiques liées à sa totalité, des propriétés qui ne sont pas réductibles et déductibles de celles de ses éléments.

Brooks décrit dans (Brooks 1987) comment il a pu faire émerger un comportement relativement évolué de "suivi de murs" en faisant intervenir deux comportements antagonistes d'attraction et de répulsion vis-à-vis des murs. Sugihara dans (Sugihara & Suzuki 1990) montre qu'on peut obtenir différentes configurations spatiales (cercles, carrés, etc.) de robots, uniquement en les dotant de règles d'actions locales simples. Les travaux de Reynolds (Reynolds 1987) ont démontré qu'une animation d'un groupe d'agents en formation (e.g., des oiseaux), peut être très réaliste, en utilisant des règles simples d'évolution de chaque entité par rapport à son voisinage immédiat.

Il y a dans la nature aussi, plusieurs phénomènes auxquels on associe volontiers le terme d'émergence de comportements. Cullen par exemple dans (Cullen & Shaw 1965) constate que la densité d'un banc de poissons en mouvement est approximativement égale dans tout le plan du regroupement, et ce comme si chacun des poissons avait une sphère autour de lui qui devait être respectée par les autres poissons. Les *lucioles* sont un autre exemple qui peut nous interpeller, car elles arrivent à synchroniser leur période d'émission des signaux lumineux, en utilisant uniquement les interactions locales existantes entre elles, c'est-à-dire que chaque luciole est influencée uniquement par les signaux émis par les lucioles voisines (Ávila et al. 2003).

Après avoir succinctement défini ce qu'est l'émergence de comportements et donné quelques exemples liés à certains types de ses manifestations, des questions essentielles par rapport à cette notion d'émergence restent néanmoins posées :

- comment peut-on avoir la certitude de la provoquer ?
- comment la constater et quantifier ses effets ?

Dans le cadre des systèmes multi-robots, il est évident que la difficulté d'obtention d'une modélisation précise de la dynamique d'interaction induite par le groupe de robots, rend très difficile voire impossible de certifier l'émergence de comportements plus complexes à partir des actions individuelles des robots. Néanmoins, le fait de développer d'une part, des comportements élémentaires robustes et d'une grande précision (ce qui ne veut pas dire rigide et sans notion d'aléatoire) et d'autre part, de pouvoir maîtriser les interactions possibles entre comportements élémentaires ne peut que nous rapprocher davantage de la maîtrise de ces comportements émergents.

Pour constater l'apparition de phénomènes émergents on procède généralement, en observant les effets macroscopiques induits par les robots (Balch & Arkin 1995*b*), tels que l'apparition de motifs ou de configurations "*patterns*" spatiales caractéristiques des tâches à exécuter, e.g., le tri sélective de disques (Melhuish 2001, chapitre 7).

Les approches cognitives ou réactives du contrôle des systèmes multi-robots ont d'énormes lacunes par rapport aux méthodes et outils qui peuvent leur permettre d'évaluer *quantitativement* : la robustesse, la tolérance aux erreurs, la précision, la flexibilité, l'adaptabilité (Parker 2000, 2001) des architectures de contrôle. Cette quantification du degré d'efficacité reste donc une notion floue et difficile à exprimer et des efforts importants doivent y être consacrés, pour ne plus avoir à manipuler des boites noires[5].

Cette quantification est encore plus difficile à obtenir dans le cas de robots purement réactifs. Pour pallier cette lacune, on procède bien souvent à une étude *qualitative* de l'architecture de contrôle. Les travaux de (Balch & Arkin 1995*b*) ou ceux de (Adouane & LeFort-Piat 2004*c*) par exemple, se basent sur l'étude d'un grand nombre de simulations, pour quantifier statistiquement l'effet induit par les communications inter-robots sur le déroulement des tâches coopératives étudiées. Ainsi les études statistiques[6] sont un moyen qualitatif efficace pour se rendre compte de l'émergence ou non de comportements collectifs satisfaisants, et permettent aussi de juger des performances des architectures de contrôle appliquées aux cas des systèmes multi-robots.

2.2.3 Alors architectures cognitives ou bien réactives ?

Le choix d'une approche plutôt qu'une autre dépend essentiellement de l'application traitée et du cahier des charges à suivre. Il est évident que de nos jours, c'est l'école cognitive qui a engendré les réalisations les plus abouties en robotique coopérative. C'est sans doute dues aux contraintes relativement moins sévères des cahiers des charges imposés aux architectures cognitives par rapport à celles plutôt réactives[7]. C'est dû d'autre part aussi à la relative nouveauté de l'investigation des domaines liés à la coopération de robots mobiles réactifs, et ainsi qu'à la relative complexité de synthèse et d'analyse des contrôles liés aux systèmes réactifs.

Le tableau 2.2 et la figure 2.7 (Arkin 1998) résume les principales caractéristiques liés aux deux modes de contrôle.

Un contrôle cognitif doit donc pouvoir disposer d'énormément d'informations, sinon de la totalité des informations du système, pour pouvoir coordonner efficacement les actions du système multi-robots. Contrairement à ceci, un contrôle réactif n'a aucune-

[5] "C'est un peu comme si l'on voulait comprendre le fonctionnement d'un distributeur automatique de tickets sans connaître d'autres éléments que le type de pièces que l'on peut y insérer et le type de tickets que l'on peut en obtenir" (Lorenz 1984, p.355)

[6] A condition qu'elles soient réellement représentatives des différents modes de fonctionnement du système.

[7] En effet, nous pouvons utiliser plusieurs mécanismes de haut niveau pour le cas des architectures cognitives, alors qu'ils ne sont pas tolérés pour le cas des architectures réactives.

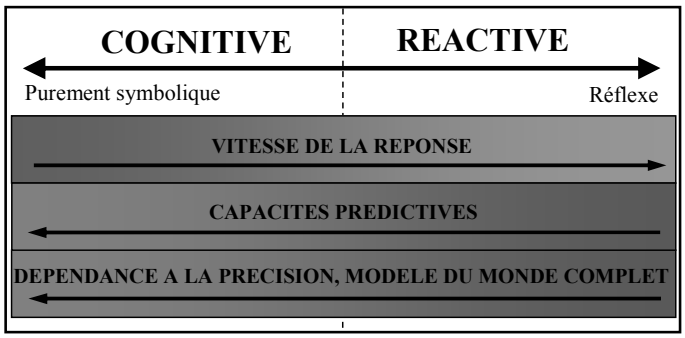


FIG. 2.7 – Contrôle réactif versus cognitive

Systèmes d'agents cognitifs	Systèmes d'agents réactifs
Représentation explicite de l'environnement,	Pas de représentation explicite,
Demande une puissance de calcul importante,	Travaille en temps réel,
Agent généralement contrôlé d'une manière complexe,	Fonctionne par stimulus-réponse,
Petit nombre d'agents,	Se prête aux systèmes avec un grand nombre d'agents,
Architecture centralisée ou distribuée,	Architecture distribuée uniquement,
Peut tenir compte de son passé.	Pas ou peu de mémoire de son historique.

TAB. 2.2 – Contrôle réactif versus cognitif

ment besoin d'une représentation explicite des interactions existantes dans le système multi-robots pour produire un contrôle en conséquence. Le contrôle réactif procède pour sa part, via des mécanismes et des objectifs spécifiques à atteindre, tels que le fait de vouloir provoquer certaines situations (aller vers une source attractive, communiquer via des protocoles bas-niveau, etc.) ou le fait de vouloir éviter certaines configurations (blocage, collisions, etc.). Ces mécanismes simples permettent d'appréhender le contrôle d'un groupe de robots d'une manière complètement différente des types de contrôle que nous avons l'habitude de rencontrer en automatique.

La combinaison de l'approche cognitive et de l'approche réactive peut s'avérer être une solution intéressante pour bénéficier des avantages liés à chacune des approches, tels que l'aspect temps réel pour l'approche réactive et l'aspect planification pour l'approche

cognitive. Cette approche hybride pose néanmoins le problème de savoir à quel moment c'est la fonctionnalité réactive qui prend la main et à quel moment c'est la fonctionnalité cognitive. Cette problématique est investiguée par exemple dans (Firby 1987), (Gat 1998), (Arkin 1989*a*) pour le cas de systèmes mono-robot.

2.3 Conclusion

Les chapitres I et II nous ont permis de présenter les principales approches, mécanismes, méthodes, caractéristiques, contraintes liés au contrôle d'un groupe de robots mobiles. Nous pouvons établir donc à présent en fonction de tous ces éléments et des contraintes que nous nous sommes donnés, le cahier des charges que nous avons suivi aux chapitres IV et V, pour élaborer des architectures de contrôle appropriées pour le contrôle d'un groupe de robots.

Nos objectifs correspondent à contrôler un *grand* nombre de robots (plusieurs dizaines) *minimalistes* autant d'un point de vue sensoriel que calculatoire, et ceci d'une manière *distribuée* et *coopérative* afin de réaliser des tâches complexes. Ces éléments nous ont amené à prospecter plus avant les approches *comportementales réactives* à base de *stimuli-réponse*, afin de proposer des architectures de contrôle respectant notre cahier des charges.

Le chapitre qui suit nous donnera un panel assez large des différents mécanismes et spécificités liés aux architectures de contrôle comportementales proposées dans la littérature.

Chapitre 3

Les architectures de contrôle comportementales

Résumé : *Ce chapitre est consacré à l'état de l'art des différentes architectures de contrôle comportementales proposées dans la littérature. Nous nous focaliserons principalement sur les mécanismes et sur les spécificités propres à chaque type d'architecture de contrôle. Une attention toute particulière sera accordée aux mécanismes de coordination entre comportements proposés par chaque architecture de contrôle.*

3.1 Mécanismes intervenants dans une architecture comportementale

Avant d'arriver à faire coopérer efficacement un groupe de robots mobiles, il est impératif de pouvoir contrôler au préalable le fonctionnement de l'entité élémentaire constituant ce groupe, c'est-à-dire pouvoir appliquer une architecture de contrôle appropriée sur l'entité robotique mobile. L'objectif de toute architecture de contrôle, est d'arriver à atteindre et/ou à maintenir des états (configurations) désirés du système. Dans le cadre des architectures de contrôle comportementales[1], ceci passe inéluctablement par la maîtrise simultanée de plusieurs niveaux d'action et de prise de décision (cf. figure. 3.1). Nous pouvons les résumer ainsi :

Le niveau 1 dans la figure 3.1 correspond à tous les éléments intervenants pour la gestion et l'exécution matérielle des commandes envoyées aux roues du robot, ceci a donc rapport avec ses actionneurs, son inertie, ou d'une manière générale avec la dynamique effective du robot mobile. Ce niveau est donc responsable entre autres de faire en sorte que la cadence de réception et le type des commandes produites par les comportements soient en adéquation avec la dynamique réalisable par le robot.

Le niveau 2 concerne le fonctionnement interne de chaque comportement élémentaire, c'est-à-dire qu'il correspond à la méthodologie adoptée pour créer la *relation* entre les stimuli perçus par le comportement et les commandes qu'il génère (cf. §4.1.1, page 78).

Le niveau 3 correspond à la manière de coordonner les actions proposées par la multitude de comportements appartenant à la même architecture de contrôle. La figure 3.2 montre un arbre représentatif des différents mécanismes de coordination entre comportements proposés dans la littérature. Ces mécanismes se décomposent en deux branches principales :

- sélection d'action : dans cette branche, la commande à envoyer aux actionneurs à chaque pas d'échantillonnage correspond à la commande générée par un *seul* comportement élémentaire. Ce comportement est choisi d'une manière hiérarchique, dynamique, etc. (cf. §3.2, page 54),
- fusion d'actions : comme son nom l'indique cette branche correspond aux mécanismes qui vont fusionner deux commandes ou plus (émanant de comportements distincts) pour aboutir à une seule commande, qui correspond à un certain consensus entre les comportements, e.g., les schémas moteurs, le contrôle par logique floue, etc. (cf. §3.3, page 66).

[1] Architectures de contrôle obtenues par agrégation de plusieurs comportements élémentaires, qui cohabitent donc dans une même structure.

Le niveau 4 correspond quant à lui aux mécanismes de coordination d'un ensemble d'agrégat de comportements, chaque agrégat représentant une tâche complexe à exécuter. Ce niveau 4 correspond plus au cadre de la vie artificielle, où une créature (un robot) doit vivre (réaliser plusieurs tâches complexes) et interagir dans un environnement complexe.

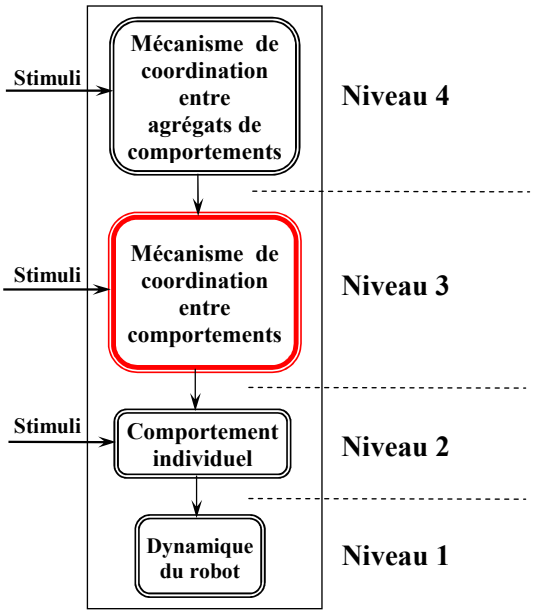


FIG. 3.1 – Mécanismes intervenants dans les architectures de contrôle comportementales

Les sections qui vont suivre, présentent un panel représentatif des différentes architectures de contrôle comportementales proposées dans la littérature. Nous focaliserons plus particulièrement sur les mécanismes de coordination entre comportements proposés (cf. figure 3.1, niveau 3 ; et figure 3.2).

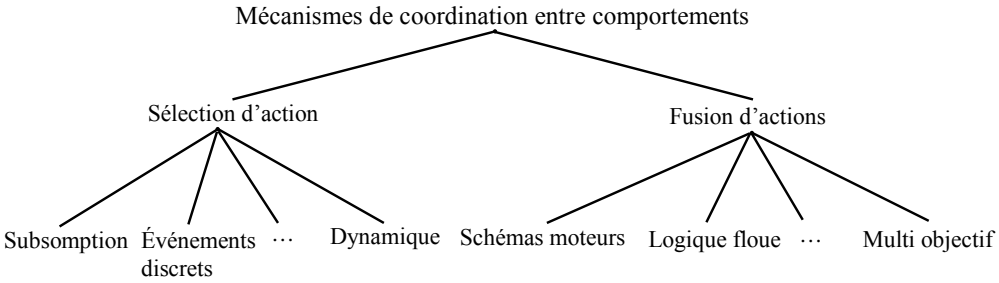


FIG. 3.2 – Arbre des différents mécanismes de coordination entre comportements

3.2 Sélection d'action

Le point commun entre toutes les architectures de contrôle appartenant à cette branche (cf. figure. 3.2) peut se résumer ainsi : parmi la multitude de comportements (fonctionnant en parallèle) constituant l'architecture de contrôle, il y a, à chaque pas d'échantillonnage, qu'un seul comportement qui va contrôler les actionneurs du robot. Ceci implique donc l'existence d'un **mécanisme de sélection d'action** approprié, pour désigner lequel des comportements va prendre effectivement la main à chaque pas d'échantillonnage.

3.2.1 Subsomption

Rodney Brooks (Brooks 1986) est le premier à proposer et à utiliser des mécanismes de subsomption[2] dans le cadre d'architectures de contrôle pour robots mobiles. L'une des principales caractéristiques liées à la subsomption, correspond à l'existence d'une hiérarchie entre les comportements (cf. figure. 3.4(a)), au sens qu'un comportement de rang supérieur (haut niveau) peut influencer les actions des comportements de rangs inférieurs (bas niveaux). Brooks prévoit pour gérer cette forme de hiérarchie entre comportements, un mécanisme de communication liant les comportements de rang supérieur à ceux de rangs inférieurs (cf. figure. 3.3). Cette communication donne la possibilité aux comportements de rangs supérieur d'agir sur les comportements de rangs inférieurs en :

- *supprimant* ou en modifiant leurs entrées,
- ou bien en *inhibant* complètement leurs sorties.

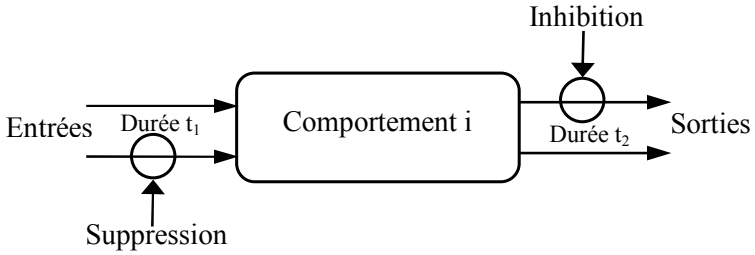


FIG. 3.3 – Processus d'interaction entre comportements

Le comportement de plus haut niveau a donc la possibilité d'examiner les données des comportements de plus bas niveau, et il peut par la suite, supprimer ce flot de données,

[2]"Subsumer" vient du latin subsumere, de sub et sumere "prendre". qui veut dire, penser (un objet individuel) comme compris dans un ensemble (un individu dans une espèce, une espèce dans un genre). *Définition du Petit Robert.*

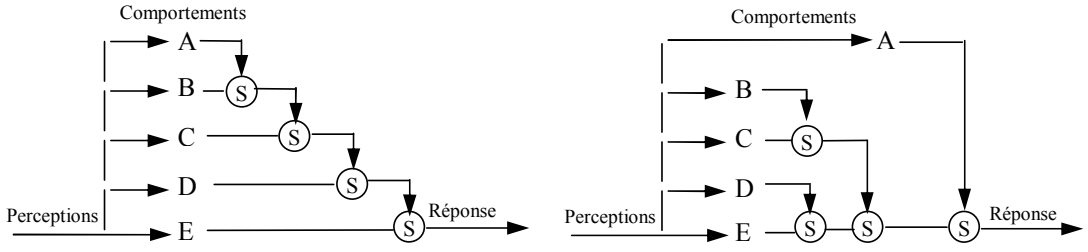
pour par exemple le remplacer par un autre. Ce comportement de plus haut niveau peut aussi agir sur le comportement de bas niveau en inhibant complètement ces sorties (commandes).

Ce qui est notable par rapport aux deux mécanismes d'interaction entre comportements proposés par Brooks, c'est qu'ils sont asynchrones et supposés volatiles, i.e., qu'un message envoyé à un comportement donné, peut être perdu si le message n'a pas pu être lu dans un délai (Durée $t_{i|i=1,2}$ sur la figure 3.3) caractérisant chaque envoi de message. Ceci donne par conséquent une caractéristique pseudo-aléatoire au fonctionnement de l'architecture de contrôle proposée par Brooks.

Commentaires

Les architectures de contrôle utilisant une hiérarchie de comportements à l'image de l'architecture de subsomption, sont très répandues dans la littérature. Nous pouvons citer par exmple celles proposées dans (Mataric´ 1992), (Ferrell 1995), (Adouane & LeFort-Piat 2003*a*), etc.

Parmi les points qui peuvent être reprochés aux architectures de contrôle à base uniquement de subsomption, c'est leur caractère figé, au sens où il y a au préalable une hiérarchie entre comportements qui est fixée une fois pour toute, et qui ne peut pas par exemple être modifiée en cours de fonctionnement. Pour briser cette forme de rigidité de la structure de contrôle à base de subsomption, Connell dans (Connell 1990), propose un ordre hiérarchique partiel (cf. figure. 3.4(b)), plutôt qu'un ordre total comme cela est défini dans Brooks (cf. figure. 3.4(a)). Il est ainsi possible dans ce modèle d'avoir des comportements de même niveau hiérarchique, et donc des comportements concurrents. Cependant, la gestion des conflits entre ces comportements, reste extrêmement délicate à effectuer.



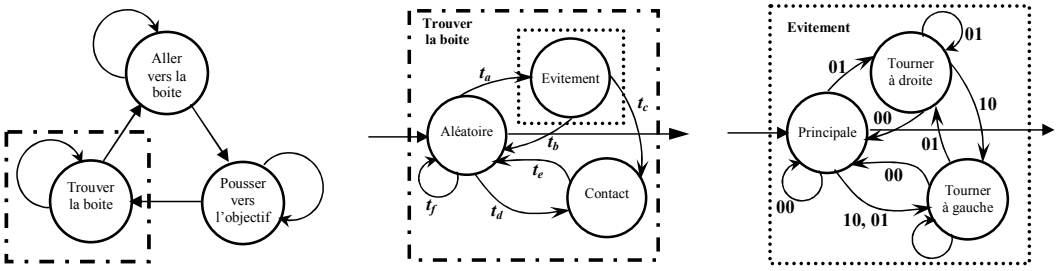
(a) hiérarchie totale (Brooks 1986) (b) Hiérarchie partielle (Connell 1990)

FIG. 3.4 – Types de hiérarchie appliqués au cas des architectures à base de subsomption

3.2.2 Événements discrets

La sélection d'action dans le cas d'un traitement par événements discrets (Kosecká & Bajcsy 1993), se fait par transition séquentielle entre états (Arkin & MacKenzie 1984). La représentation la plus courante de ces mécanismes, utilisent les automates à états finis, pour décrire les états et leurs transitions (e.g., voir la figure 3.5 pour le cas de la modélisation d'une tâche de poussée de boite). Les états dans cette modélisation, représentent l'exécution des actions et/ou des comportements, et les événements responsables des transition entre états, sont assurés grâce aux perceptions (Kube 1997, 96).

Nous allons illustrer ce mécanisme de sélection d'action, basé sur les événements discrets, via la présentation des travaux de Ronald Kube (Kube & Zhang 1997). Kube a proposé une architecture de contrôle basée sur des comportements réactifs, pour réaliser la tâche coopérative de poussée de boite (cf. figure. 3.5(a)) par un groupe de robots mobiles. Son architecture de contrôle consiste à décomposer la tâche globale à réaliser en plusieurs états (cf. figure. 3.5(a)). Chaque état est à son tour décomposé en plusieurs autres états (cf. figure. 3.5(b)), et l'opération est répétée encore une fois pour aboutir à la figure 3.5(c).



(a) Tâche de poussée de boite (b) Comportement de Trouver la boite (c) Comportement d'Évitement

FIG. 3.5 – Sélection d'actions sous forme d'automates à états finis

Les automates à états finis ont donc servi à Kube pour modéliser l'architecture de contrôle qu'il propose. La spécificité des travaux de Kube, est le fait que les transitions existantes entre états, sont toutes de type booléen (cf. figure. 3.5(c)). Ces transitions sont nommées "*perceptual cues*".

Kube propose deux assertions pour pouvoir manipuler effectivement des *perceptual cues* de type booléen, nous pouvons les résumer ainsi :

– l'*orthogonalité* des stimuli, c'est-à-dire que les capteurs sont :

- de nature complètement différentes (e.g., un capteur de température et un capteur de contact),

- ou bien de même nature, mais leurs fenêtres d'observation sur l'environnement (champ perceptuel) ne se chevauchent pas, ou bien que ces capteurs utilisent des seuils différents pour caractériser les stimuli (cf. figure. 3.6).

– l'*additivité* des stimuli, i.e., que les stimuli booléens émanants des capteurs, sont concaténés pour obtenir des vecteurs de type booléen.

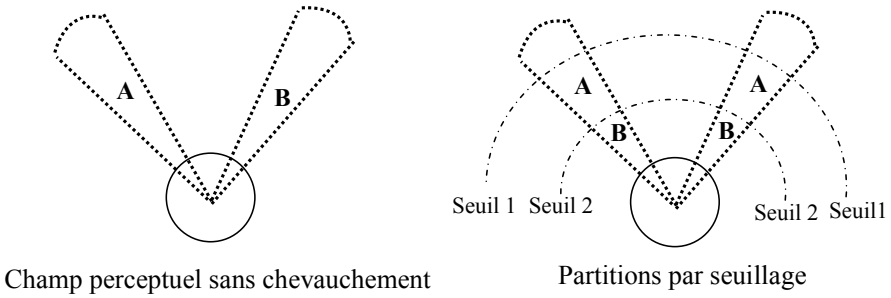


FIG. 3.6 – Capteurs spatialement orthogonaux (Kube & Zhang 1997)

Commentaires

Les motivations de Ronald Kube pour l'utilisation de ce qu'il a appelé *the perceptual cues* (i.e., la concaténation de stimuli booléens orthogonaux) sont d'une part, de manipuler des informations booléennes, ce qui permet de réduire considérablement la phase de prétraitement des informations capteurs, et d'autre part, d'éviter les conflits de transitions d'un état à un autre. Ces deux points donnent par conséquent un aspect plus réactif à l'architecture de contrôle.

Une interrogation persiste néanmoins, par rapport aux *perceptual cues* énoncées par Kube. Elle se rapporte à la pertinence d'utiliser uniquement des informations booléennes, pour décrire complètement l'état dans lequel se trouve un robot mobile, surtout si celui-ci, se trouve immergé dans un environnement complexe, et avec une très grande dynamique d'interaction. D'ailleurs Kube dans (Kube & Zhang 1994) utilise plusieurs mécanismes pour éviter des situations de blocage entre robots induites à notre sens principalement par la prise en compte que d'informations booléennes.

3.2.3 Sélection d'action dynamique

Pattie Maes dans (Maes 1989) introduit une nouvelle vision des mécanismes de sélection d'action dans une architecture de contrôle comportementale. Contrairement à l'architecture de subsomption (Brooks 1986) qui définit un ordre hiérarchique fixe entre

comportements, l'architecture proposée dans (Maes 1989) n'établit aucun ordre hiérarchique entre comportements, laissant des liens activateurs/inhibiteurs existants entre comportements, produirent une forme de *sélection d'action dynamique*.

Le modèle de sélection d'action dynamique proposé dans (Maes 1989) et développé dans (Maes 1991*a*, 91b, 91c), est intéressant dans la mesure où il est basé sur une forme de compétition entre comportements, et ce via l'intermédiaire de niveaux d'activation différentiels. Ces niveaux d'activation résultent de la conjugaison de facteurs exogènes (stimuli de l'environnement) et de facteurs endogènes représentés par des motivations[3].

L'autre aspect caractéristique de l'architecture de contrôle de Pattie Maes, est l'organisation des comportements sous forme d'un réseau, ce qui implique l'existence de liens spécifiques entre comportements. Au travers de ces liens, les comportements s'influencent mutuellement. Ils ont la possibilité entre autres d'augmenter et/ou de diminuer les *facteurs d'activation* propres aux autres comportements.

Un comportement devient *exécutable* à l'instant t, uniquement lorsque l'ensemble de ses préconditions devient vrai. Un module exécutable peut alors être sélectionné parmi l'ensemble de tous les comportements exécutables à l'instant t. Néanmoins, l'*unique* comportement qui va être actif à l'instant t est celui qui :

- a son *niveau d'activation* qui dépasse le seuil I_α (seuil attribué au comportement au préalable),
- a aussi le niveau d'activation maximum parmi tous les comportements exécutables à l'instant t (cf. figure. 3.7(b)).

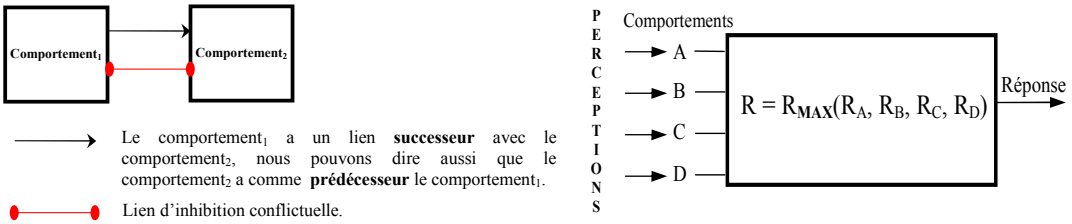
Les facteurs influençant le fonctionnement du réseau de comportements sont les suivants :

- la valeur du *taux d'augmentation* du *niveau d'activation* durant chaque période d'échantillonnage, et ce dans le but de moduler la vitesse du comportement à dépasser son seuil I_α,
- le taux de propagation émis par les comportements *actifs*, propagation vers l'avant "*forward spreading*" (utilisation des liens *successeurs* pour influencer les autres comportements (cf. figure. 3.7(a))),
- le taux de propagation émis par les comportements non actifs (non exécutables), propagation vers l'arrière "*backward spreading*" (utilisation des liens *prédécesseurs* (cf. figure. 3.7(a))),

[3]Ces motivations modulent le *taux d'augmentation* et/ou de diminution du *niveau d'activation* propre à chaque comportement élémentaire.

– le taux relatif à l'influence des stimuli de l'environnement et du but à atteindre.

Les mécanismes de sélection d'action dynamique utilisent donc le réseau de connexions entre comportements pour permettre aux comportements de s'activer et/ou de s'inhiber mutuellement, c'est-à-dire respectivement augmenter (utilisation des liens successeurs et prédécesseurs) et/ou diminuer (utilisation des liens d'inhibitions conflictuelles) les *niveaux d'activation* propres à chaque comportement (cf. figure. 3.7(a)).



(a) Liens existants entre comportements : lien successeur/prédécesseur, et lien inhibiteur

(b) Sélection du comportement qui a le plus grand taux d'activation

FIG. 3.7 – Sélection d'action dynamique (Maes 1989)

Commentaires

L'architecture proposée par Pattie Maes dénote l'existence de configurations où le réseau de comportements converge vers des cycles oscillants, ainsi que des configurations où aucun comportement ne peut être actif "*deadlocks*" (cas où les activations respectives des comportements s'auto-annulent). Drogoul dans (Drogoul 1993) prévoit pour éviter cette configuration, l'existence d'une tâche défaut qui est activée si aucun autre comportement n'est actif.

L'autre aspect certainement plus difficile à résoudre avec cette architecture, est la détermination, efficace et pourquoi pas optimale, de la multitude de paramètres régissant le bon fonctionnement du réseau de comportements. Plus spécifiquement : les taux d'influence des comportements entre-eux, les seuils d'activation, ainsi que leur vitesse d'accroissement ou de diminution en fonction des motivations cumulées durant le fonctionnement de l'architecture de contrôle.

A défaut d'avoir une méthode systématique pour déterminer les paramètres cités ci-dessus, ce genre d'architecture de contrôle s'attache à constater l'apparition ou non de l'émergence du comportement global désiré, et ce à partir des interactions parallèles et dynamiques inter-comportements. Ceci fait un lien intéressant avec les notions d'émergence de comportements globaux en robotique coopérative, néanmoins dans ce cadre,

l'émergence est induite principalement par les interactions existantes entre robots et non pas via les interactions entre comportements. Cette émergence globale peut être le fruit d'émergence de comportements locaux au niveau des robots, cependant, le fait d'avoir l'émergence de comportements locaux au niveau des individus n'implique pas directement l'émergence du comportement global désiré du groupe de robots. Par conséquent, l'utilisation de l'approche de Pattie Maes dans le cadre de la robotique coopérative peut constituer une difficulté supplémentaire à surmonter pour tendre vers l'émergence de comportements coopératifs. En effet, si les interactions entre robots ne sont pas facilement maîtrisables, alors que dire si on ajoute à ceci, un comportement pouvant être chaotique de l'individu élémentaire.

Le caractère peu prédictible de l'entité artificielle contrôlée avec l'architecture de Pattie Maes, augmente considérablement la difficulté d'exploiter cette architecture pour le cas où les tâches à exécuter exigent une grande rigueur de déplacement et/ou de coordination. D'autre part, de nombreuses tâches génériques en robotique mobile exigent une séquentialité entre les comportements, ce qui n'est pas systématiquement obtenue avec l'approche de Pattie Maes.

3.2.4 EthoModélisation

Drogoul dans (Drogoul 1993, chapitre 3) s'est intéressé entre autres à la manière dont les éthologues tentent de formaliser les déclenchements ou activations des comportements chez les animaux, d'où le nom d'EthoModélisation "*EthoModelling*" donné au système de contrôle proposé.

L'architecture de contrôle proposée (cf. figure. 3.8) postule que le comportement d'un animal peut être caractérisé par un ensemble de *tâches indépendantes*, où chacune est composée d'une séquence de comportements élémentaires moteurs appelés *primitives*. Chacune de ces tâches est exclusive et son déclenchement provient d'une stimulation externe ou interne, caractérisée par des *stimuli* à *force* variable, qui se combine à une *motivation* préexistante. Cette motivation est exprimée sous le double aspect d'un *seuil inhibiteur*[4] et d'un *poids* qui reflète l'expérience antérieure de l'animal pour l'exécution de la tâche en question. Ce *poids* pouvant varier positivement ou négativement, permet de représenter les phénomènes de *renforcement* ou *d'habituation*[5] observés chez les animaux.

[4]Le seuil inhibiteur décrit dans (Drogoul 1993), n'est que l'inverse de la motivation de l'agent. Un seuil élevé représentera une motivation faible, d'où son nom de seuil inhibiteur.

[5]Le renforcement est un processus qui conduit l'animal à être de plus en réceptif à un stimulus donné, i.e., à activer de plus en plus vite et avec une intensité croissante le comportement. L'habituation est exactement l'inverse.

Pour modéliser l'intensité avec laquelle une tâche est déclenchée, Drogoul utilise aussi un *niveau d'activation* affecté à chacune des tâches, qui résulte de la multiplication de la *force du stimulus* déclencheur et du *poids* de la tâche. Il est recalculé à chaque excitation de la tâche en question, et sert ainsi par comparaison avec les *niveaux d'activation* des autres tâches, à les discriminer.

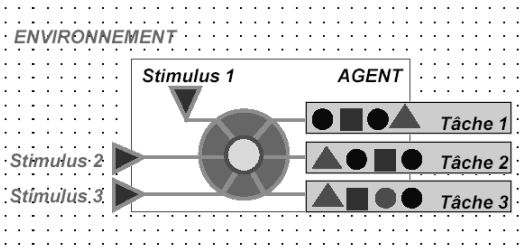


FIG. 3.8 – Un agent est constitué d'un ensemble de tâches, chaque tâche étant déclenchée par un stimulus particulier, qu'il soit interne ou externe. Une tâche est composée d'une séquence de primitives, représentées ici par des figures géométriques, qu'elle exécute lorsqu'elle est déclenchée (Drogoul 1993).

Les mécanismes dynamiques de déclenchement des tâches préconisés par Drogoul se composent de trois étapes : (1) recherche des tâches déclenchables, (2) comparaison avec les tâches en cours, (3) commutation éventuelle.

Les tâches *activables* sont les tâches excitées dont le *niveau d'activation* est supérieur au *seuil inhibiteur*. Enfin, le mécanisme de sélection d'action consiste à extraire de cet ensemble, les tâches considérées comme *déclenchables*, c'est-à-dire toutes les tâches *activables* dont le niveau d'activation est supérieur au *niveau d'activité* de la tâche en cours d'exécution.

Les mécanismes affectant le déclenchement des tâches sont paramétrés en fonction de données biologiques fournies par des éthologues. La définition d'une échelle de temps appropriée, revête une valeur de première ordre dans l'architecture préconisée par Drogoul, car c'est elle qui conditionne en grande partie le déclenchement des tâches. En effet, Drogoul affecte un temps idéal-théorique (constant) pour l'exécution de chaque tâche[6]. Ce temps influence directement la mise à jour du *poids* affecté à la tâche donc à son *niveau d'activation*. Plus spécifiquement ce *poids* est remis à jour en fonction du fait que la tâche est arrivée à terme ou est interrompue. Les mécanismes proposés en relation avec le paramètre temps, sont les suivants :

[6]Ce temps servira d'indicateur implicite de l'état interne de l'agent.

- le *renforcement* : dans ce cas de figure, la tâche exécutée est arrivée à terme, l'idée alors est de renforcer sa probabilité à se réexécuter dans le futur. Ceci passe donc par une augmentation du *poids* affecté à la tâche en question,
- l'*habituation* : dans ce cas de figure, la tâche est interrompue avant de terminer son exécution (ce qui indique d'une certaine manière, la non pertinence de l'exécution de cette tâche), donc l'idée ici est de réduire la probabilité qu'à cette tâche pour se réexécuter dans le futur. Ceci passe donc par une diminution du *poids* affecté à la tâche en question.

Le mécanisme nommé *motivation* proposé par Drogoul n'est quant à lui pas directement corrélé avec le paramètre temps. Ce mécanisme est lié à la probabilité qu'aurait une tâche d'être déclenchée indépendamment de la valeur courante de son *seuil inhibiteur*.

Par conséquent, l'analyse de la dynamique de gestion des paramètres de tâches présentée ici, dénote l'existence d'ensembles de rétroactions positives et négatives, qui vont influer sur le comportement de l'agent de manière significative. Les rétroactions de *renforcement* sont considérées comme des rétroactions intervenant sur le long terme, alors que les rétroactions sur la valeur du *seuil inhibiteur* sont par opposition des rétroactions de court terme, qui vont refléter les modifications dynamiques de l'environnement et de l'activité de l'agent.

Commentaires

Le modèle de sélection d'action proposé par Drogoul nommé "EthModélisation" démontre avec brio son adéquation avec les paradigme initiés par les éthologues sur le fonctionnement des sociétés de fourmis (projet MANTA[7] (Drogoul 1993, partie 2)). Ce modèle a permis de tester entre autres les hypothèses concernant l'émergence de structures sociales, telles que la division du travail, la sociogénèse, les relations de dominance et de hiérarchie au sein d'un type particulier de colonies de fourmis (*Ectatomma ruidum*). Cependant, l'applicabilité effective de ces mécanismes de sélection d'action en robotique mobile collective, pourrait s'avérer difficile à obtenir et ceci vu le relatif haut niveau d'abstraction des tâches coopératives réalisées par Drogoul en simulation (e.g., le fourragement, le tri collectif, coopération de robots dockers).

Le connaissance du temps qui s'écoule ainsi que des temps références pour caractériser l'exécution des tâches par les entités autonomes, peut être considéré comme une assertion forte dans le cas d'un modèle purement réactif des entités robotiques.

Les mêmes commentaires par rapport aux mécanismes de sélection d'action dynamique, proposés par Pattie Maes conviennent aussi ici. Nous insistons néanmoins, sur

[7] `http://www-poleia.lip6.fr/~drogoul/projects/manta/index.html`

la difficulté méthodologique que pourrait avoir un concepteur, à fixer la multitude de paramètres régissant le déclenchement des tâches.

3.2.5 ALLIANCE

Lynne Parker dans (Parker 1994, 1998) propose une architecture de contrôle comportementale nommée ALLIANCE dédiée au contrôle distribué d'un groupe de robots mobiles *hétérogènes* (structurellement et cognitivement), qui doivent coopérer pour la réalisation d'une tâche globale. Celle-ci étant considérée, comme un ensemble de sous-tâches indépendantes qui disposerait d'un ordre d'exécution fixe.

L'architecture ALLIANCE (cf. figure. 3.9) choisit à chaque instant l'action appropriée à appliquer, et ce en tenant compte à la fois : des exigences de la tâche à accomplir, de l'activité des autres robots, des conditions courantes de l'environnement, et finalement de l'état interne du robot. Plus spécifiquement l'état interne du robot est caractérisé par les blocs de "comportements motivationnels" (cf. figure. 3.9). ALLIANCE met en place deux types de motivation internes :

- *l'impatience* : qui indique, l'impatience que pourrait avoir un robot à remplacer délibérément un autre robot qui exécuterait maladroitement une sous-tâche,
- *le consentement* "*acquiescence*" : c'est-à-dire consentir à abandonner l'exécution d'une sous-tâche, quand le robot estime que la progression attendue de la sous-tâche en cours de réalisation, n'est pas atteinte.

Les blocs de motivation utilisés dans ALLIANCE sont les éléments qui lui permettent d'être tolérante entre autres : aux erreurs "*fault tolerance*", aux pannes inopinées, à l'ajout ou à l'élimination d'un robot lors de l'exécution de la tâche coopérative. Ces blocs de motivation reçoivent leurs stimuli :

- des capteurs,
- des communications échangées entre robots,
- des autres blocs de motivation internes au robot.

La démarche de formalisation des mécanismes internes d'activation/inhibition entre comportements initiée par Pattie Maes (Maes 1989) a été poursuivie par Lynne Parker (Parker 1994) pour proposer une formulation plus appropriée pour le cas, d'une part de la coopération d'un groupe de robots mobiles, et d'autre part pour l'activation/inhibition d'un ensemble de comportements (cf. figure. 3.9).

La formulation correspondante par exemple à la *motivation globale* propre à chaque *ensemble de comportements*, est donnée par ce qui suit : chaque robot "r_i" disposera d'une

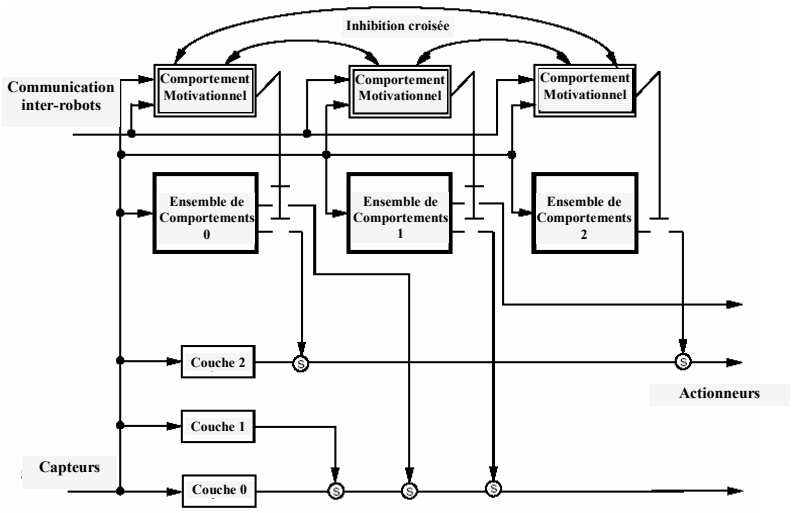


FIG. 3.9 – ALLIANCE, l'architecture de contrôle proposée par Lynne Parker

motivation pour chaque ensemble de comportements a_{ij} qui est calculé comme suit : $m_{ij}(0) = 0$, et

$$m_{ij}(t) = [m_{ij}(t-1) + impatience_{ij}(t)] \times sensory\_feedback_{ij}(t) \times activity\_suppression_{ij}(t) \times impatience\_reset_{ij}(t) \times acquiescence_{ij}(t).$$

Avec $impatience_{ij}(t)$ est la fonction décrivant l'évolution du taux d'*impatience* du robot. Les autres variables sont de type booléen telles que : $sensory\_feedback_{ij}(t)$ décrit si les préconditions nécessaires pour l'activation de l'ensemble de comportements sont vérifiées ou non ; $activity\_suppression_{ij}(t)$ décrit s'il existe ou non un ensemble de comportements a_{ik} avec k $\neq$ j, qui soit actif à l'instant t ; $impatience\_reset_{ij}(t)$ est mise à 0 si un autre robot réalise efficacement la tâche qu'il est en train d'attendre, cette variable est à 1 sinon, $acquiescence_{ij}(t)$ détermine quand le robot doit abandonner ou non la tâche en cours d'exécution.

Par conséquent la motivation correspondante à l'ensemble de comportements a_{ij} va continuer à augmenter à moins que : les informations capteurs indiquent le contraire, un autre comportement (qui est en compétition) est actif, un autre robot prend la relève pour l'exécution de la tâche, ou bien que le robot abandonne la tâche. Dès que la valeur de la motivation globale dépasse un seuil fixé au préalable, celle-ci activera l'ensemble de comportements a_{ij} dont elle dépend directement (cf. figure. 3.9), inhibant par la même occasion les autres ensembles de comportements via des liens inhibiteurs. Le robot émet alors un signal à tous les autres robots pour indiquer que la tâche a_{ij} est active.

ALLIANCE a fait l'objet de plusieurs tests autant en simulation qu'en expérimentation, et ceci pour plusieurs tâches génériques de coopération de robots mobiles (tâche coopérative de poussée d'objets, nettoyage coopératif, etc.), une liste exhaustive est donnée dans (Parker 2001).

Commentaires

ALLIANCE trouve son inspiration pour une grande part, dans les comportements sociaux des êtres humains, où l'on dénote des comportements relatifs aux motivations à l'*impatience* et/ou au *consentement*[8]. L'implémentation de ce type de motivations dans ALLIANCE exige néanmoins des capacités cognitives de haut niveau pour les robots mobiles. Parmi ces capacités nous pouvons citer celles qui permettent au robot de s'apercevoir de l'évolution de la tâche qu'il est en train de réaliser, le robot est même amené à se rendre compte de l'évolution des sous-tâches exécutées par les autres robots[9]. Ce genre d'informations nécessaires au fonctionnement d'ALLIANCE sont données par exemple dans (Parker 1998) via un superviseur externe, dans le but de réaliser la tâche coopérative de nettoyage aléatoire "*hazardous waste cleanup*".

[8]L'analogie des blocs de motivation décrites dans ALLIANCE et les relations pouvant exister entre humains, peut être décrite par le cas d'une personne qui est en train de réaliser une tâche et qui deviendrait de moins en moins motivée pour la réaliser, conséquence par exemple de sa maladresse (*consentement* à abandonner la tâche). Si par le fruit du hasard une autre personne qui se sentirait compétente est présente dans l'environnement, alors cette personne deviendrait de plus en plus *impatiente* de prendre en main la réalisation de la tâche en question.

[9]Cette information est récupérée soit d'une manière implicite via les capteurs propres à chaque robot qui observe l'exécution des sous-tâches, soit via une communication explicite comme par exemple une communication délibérée du robot exécutant la sous-tâche, qui est en principe en mesure de rendre compte de l'évolution de la sous-tâche en question.

3.3 Fusion d'actions

La fusion d'actions, comme son nom l'indique, correspond à la branche des mécanismes de coordination entre comportements (cf. figure. 3.2, page 53), qui fusionne des actions (commandes) émanant de plusieurs comportements élémentaires, pour déterminer la commande à appliquer sur le robot. Cette commande correspond donc à un certain consensus entre les comportements qui contribuent ainsi simultanément au contrôle du robot.

En effet, ce genre de mécanismes de coordination entre comportements, fait coopérer les comportements au lieu de les mettre en compétition, comme c'est le cas dans les mécanismes de sélection d'action (cf. §3.2, page 54).

Nous présentons dans ce qui suit, les mécanismes de fusion d'actions correspondant aux schémas moteurs et aux contrôleurs à base de logique floue. Il y a néanmoins plusieurs autres mécanismes de fusion d'actions proposés dans la littérature. Parmi eux nous citons ceux proposés dans (Pirjanian 1999), (Rosenblatt & Thorpe 1995), (Simonin 2001), etc.

3.3.1 Schémas moteurs

Les schémas décrivant l'interaction entre les perceptions et les actions, tirent d'une manière générale leur origine des théories liées au fonctionnement du cerveau "*brain theory*" (Head & Holmes 1911). Arbib (Arbib 1981) est l'un des premiers à avoir appliquer ces théories à la robotique.

Les schémas décrits dans (Arkin 1989*b*) représentent des agents qui sont instanciées dans le but, soit de représenter une perception, c'est-à-dire le cas d'un "*schéma perceptuel*", ou bien dans le but de contrôler les actions d'un robot "*schéma moteur*". La figure 3.10 montre la relation existante entre les schémas perceptuels et les schémas moteurs. Les sous-schémas perceptuels présents dans la figure, montrent le cas où au sein même d'un schéma perceptuel, on isole certaines informations pour une exploitation appropriée. Il est à noter qu'on instancie autant de *schémas perceptuels*, qu'il y a d'éléments à observer dans l'environnement. En effet, contrairement à l'architecture de subsomption où les comportements sont hiérarchiquement organisés, l'approche schémas fonctionne sous forme de réseau dynamique de schémas perceptuels et/ou moteurs, qui s'adaptent en fonction des perceptions et des objectifs courants du robot.

Les principes liés aux champs de potentiels[10] sont les briques de base de l'utilisation des schémas perceptuels et/ou moteurs (Arkin 1989*b*). Khatib dans (Khatib 1980, 86) est

[10]Largement utilisés pour entre autres les lois : de la gravitation universelle, du magnétisme, et de l'éléctrostatique, où l'intensité du champ est inversement proportionnelle au carré de la distance.

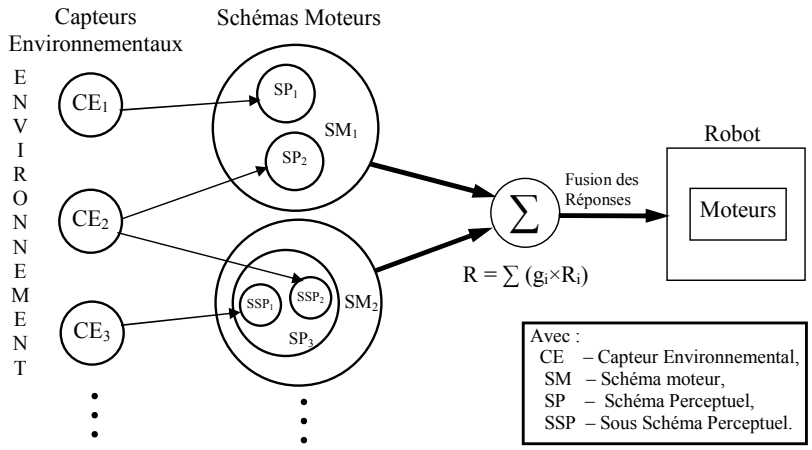


FIG. 3.10 – Relations existantes dans un schéma "perception-action" (Arkin 1998)

l'un des premiers à utiliser le concept des champs de potentiel pour proposer une architecture de contrôle temps-réel adaptée pour l'évitement d'obstacles autant pour des robots manipulateurs que pour des robots mobiles. Khatib dit pour expliquer la philosophie inhérente à l'application des champs de potentiels en robotique "*Le bras manipulateur se déplace dans un champ de forces. La position à atteindre est un point attractif pour l'effecteur et les obstacles sont des surfaces répulsives pour le bras manipulateur*".

Il est possible de croire que les *schémas moteurs* ne sont qu'une application directe des champs de potentiels. En réalité, différents éléments sont ajoutés dans les *schémas moteurs* pour contrôler l'évolution du robot. La démarche utilisée pour un contrôle fondé sur l'utilisation des *schémas moteurs* (Arkin 1989*b*) est la suivante :

1. chaque *schéma moteur* calcule une sortie de commande sous forme vectorielle, qui est déduite des mécanismes propres aux champs de potentiels. L'algorithme 3.1 montre un exemple simple d'une formulation mathématique du schéma moteur "*éviter-un-obstacle-statique*" (cf. figure. 3.11(b)), tels que V_{module} et $V_{direction}$ correspondant respectivement au module et à l'orientation de la réponse (e.g., une commande en vitesse) résultante du *schéma moteur*. La figure 3.11 montre la représentation en deux et trois dimensions de quelques *schémas moteurs*,
2. comme l'évolution du robot n'est généralement pas déduite que d'un seul *schéma moteur*, mais d'un réseau dynamique de *schémas moteurs*, chaque sortie "R_i" d'un *schéma moteur* est multipliée par un gain g_i, puis additionnée aux autres sorties (cf. figure. 3.10),

Algorithme 3.1 Schéma moteur pour le cas de l'*évitement d'un obstacle statique*

$$V_{module} = \begin{cases} 0 & pour \quad d > S \\ \dfrac{S-d}{S-R} \times G & pour \quad R < d \leqslant S \\ \infty & pour \quad d \leqslant R \end{cases}$$

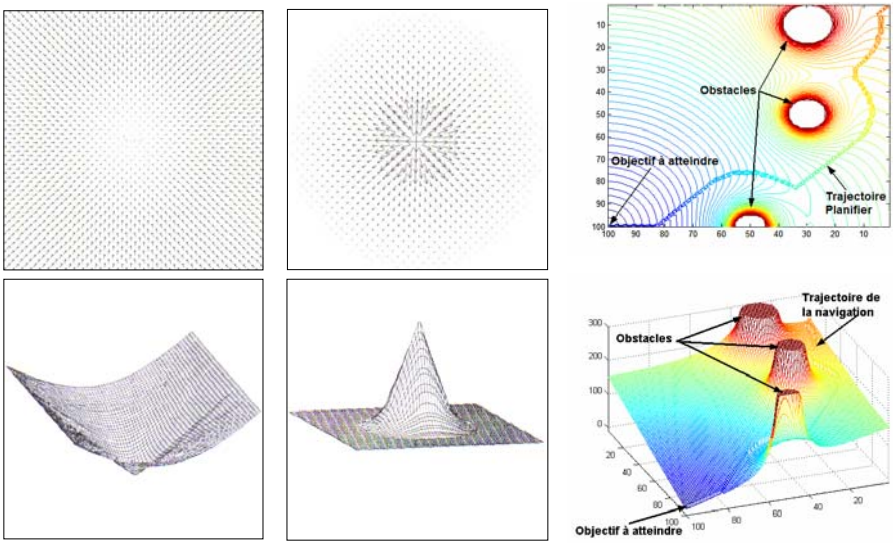
Avec :

- d distance séparant le robot du centre de l'obstacle,
- S sphère d'influence (radiale au centre de l'obstacle),
- R rayon de l'obstacle,
- G gain.

Et

$V_{direction}$ longe le centre de l'obstacle vers le centre du robot, et dans le sens allant de l'obstacle vers le robot.

3. une normalisation du vecteur de commande finale est nécessaire, pour ne pas avoir des aberrations par rapport aux vitesses instantanées que doit prendre le robot à



(a) (b) (c)

FIG. 3.11 – Les figures (a) et (b) représentent respectivement les champs de forces 2D et 3D, générés par les schémas moteurs : *aller-vers-l'objectif* et *éviter-un-obstacle-statique*. La figure (c) correspond à l'addition de schémas moteurs pour la navigation (atteindre l'objectif tout en évitant les obstacles) d'un robot mobile.

des points déterminés de l'espace (Arkin 1989*b*).

Le contrôle d'un robot en utilisant les *schémas moteurs* n'implique pas forcément le calcule des commandes pour chaque point de l'environnement, mais juste un calcul à la configuration courante du robot. Ce qui permet ainsi de garder l'aspect réactif et donc temps-réel du contrôle.

Commentaires

L'approche proposée par Ronald Arkin (Arkin 1989*b*) est très attrayante du fait qu'elle est très intuitive (elle transforme directement des informations capteurs en actions à exécuter par le robot), simple d'utilisation et démontre une efficacité remarquable. Ce qui est aussi notable est le fait que le traitement effectué par chaque *schéma moteur* est totalement indépendant des autres schémas, ce qui permet une distribution efficace des tâches, dans un système parallèle.

Cependant, la réalisation de tâches complexes par un assemblage de plusieurs schémas moteurs est très difficile à obtenir, ce qui rend laborieuse toute prévision du comportement global du robot. L'approche *schéma moteur* est aussi très dépendante des *schémas perceptuels*, la rendant ainsi plus sensible aux incertitudes des capteurs. Afin de permettre de contrôler finement les actions des robots en utilisant les *schémas moteurs*, il est nécessaire d'une part, d'avoir une connaissance précise des positions à atteindre ou à éviter (objectif à atteindre, obstacles) et d'autre part, une connaissance exacte de la forme géométrique des obstacles[11].

L'utilisation des *schémas moteurs* dénote l'existence d'autres lacunes émanant directement des problèmes inhérents aux champs de potentiels (Koren & Borenstein 1991), nous pouvons citer par exemple l'existence :

- de situations pièges "*trap situation*" caractérisées par des minima locaux (e.g., les obstacles en forme de U),
- de positions dans le plan où la résultante des forces induites par les différents *schémas moteurs* est nulle (équilibre des forces), ceci implique que la vitesse du robot à cette position doit être nulle (robot à l'arrêt !),
- d'oscillation du mouvement du robot autour d'une trajectoire déterminée, telle que la traversée d'un couloir de largeur réduite. Dès que le robot s'écarte du centre du

[11] Il est à noter, que les formes convexes sont celles qui sont les plus utilisées dans la littérature pour représenter les obstacles.

couloir, il va subir des oscillations qui seront d'autant plus grandes que sa vitesse est importante et que le couloir est moins large,
- d'oscillations du mouvement du robot entre deux points de l'environnement, ce qui est d'ailleurs le plus critique, car le robot bouclera indéfiniment autour de configurations déterminées.

Plusieurs solutions sont fournies dans les travaux entre autres de (Koren & Borenstein 1991), (Krogh & Thorpe 1986), (Borenstein & Koren 1989, 91), (Khatib 1996), pour réduire voire éliminer les problèmes cités ci-dessus. Parmi elles, nous pouvons citer :
- prévoir une mémoire à court terme pour éviter les oscillations persistantes entre deux points (Balch & Arkin 1993),
- prévoir des communications locales entre les *schémas moteurs* pour trouver un compromis entre les valeurs des gains affectées à chaque *schéma moteur*,
- injecter du bruit pour le calcul de la somme des champs de forces (Arkin 1987), ce qui veut dire ajouter un *schéma moteur stochastique*, pour éviter entre autres l'équilibre des forces,
- réduire la vitesse du robot pour éviter les oscillations importantes qui peuvent affecter le robot,
- définir pour l'architecture de contrôle, un paramètre qui fixe un temps limite pour l'exécution d'une tâche déterminée "*the hard real-time deadline*", i.e., que dépassant ce temps caractéristique, le robot va par exemple re-planifier sa trajectoire.

3.3.2 Logique floue

C'est à Lotfi Zadeh (Zadeh 1965) que revient le mérite de faire la jonction en 1965, entre les logiques multivaluées des années 30 et les travaux déjà existants sur les concepts vagues, et ceci grâce à la notion d'ensembles flous et aux logiques associées.

L'un des principaux objectifs de la logique floue, est de traduire la notion de *nuance* existante entre les objets, les actions ou les concepts qui nous entourent. En effet, contrairement à la logique booléenne qui arrive à discriminer uniquement entre deux états (vrai ou faux, "1" ou "0", froid ou chaud, etc.), la logique floue donne une autre perspective, où une variable est caractérisée par un degré d'appartenance à un ensemble flou (fonction d'appartenance) donné. Par exemple, on ne se contentera plus de dire qu'une couleur est noire ou blanche, mais on pourra la caractériser par sa nuance de gris.

L'application de la logique floue qui nous intéresse, concerne plutôt le contrôle des robots mobiles, plus spécifiquement l'utilisation de la logique floue pour introduire cette notion de fusion d'actions dans une architecture de contrôle comportementale. La figure

3.12 montre les différentes parties composant un contrôleur flou. Par mesure de clarté, nous donnons succinctement leur définition :

- Fuzzification : chargée de convertir les grandeurs physiques (réelles) en variables comprises par le contrôleur flou (nommées variables linguistiques),
- Base de connaissance : généralement produite par un expert, elle contient entre autres, les caractéristique (forme, position, etc.) des fonctions d'appartenance des entrées et des sorties du contrôleur flou (cf. figure. 3.13),
- Règles floues : elles établissent les relations entre les entrées et les sorties sous forme de règles "**Si** Entrée_1 **Alors** Sortie_2" (cf. figure. 3.13). Il est possible d'adjoindre à ces règles d'autres opérateurs (e.g., OU et ET) pour éventuellement permettre de manipuler plusieurs entrées et/ou sorties simultanément (Bühler 1994),
- Défuzzification : consiste à convertir la partie floue de l'inférence en une grandeur de commande.

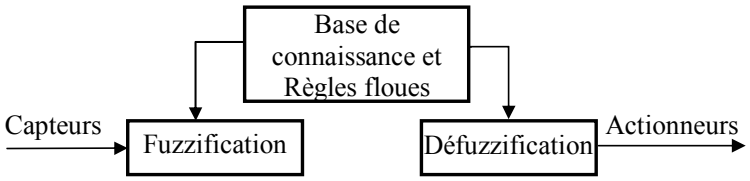


FIG. 3.12 – Contrôleur flou

La figure 3.13 donne un cas simple d'implémentation du comportement d'*évitement d'obstacles* en utilisant un contrôleur flou. Le contrôleur traduit les informations capteurs en variables linguistiques (obstacle à gauche, près, loin, etc.) "c'est la fuzzification". Cette entrée va activer une règle floue qui donnera une sortie floue. Cette sortie sera par la suite transformée en une commande exploitable par les actionneurs (commande d'angle) "c'est la défuzzification".

La fusion d'actions d'une multitude de comportements, dans le cadre d'un contrôleur flou, est obtenue quasi systématiquement via l'utilisation des mécanismes d'inférence et de défuzzification caractérisant les contrôleurs flous (Bühler 1994). Pour illustrer ceci, il suffirait par exemple de remplacer dans la figure 3.13 les fonctions d'appartenance des ensembles de sortie, par d'autres fonctions d'appartenance qui décriraient l'activation de différents comportements. Les chevauchements entre fonctions d'appartenances décriraient alors les intervalles où il y aurait éventuellement une contribution mutuelle entre deux comportements distincts (Yen & Pfluger 1995), (Arrúe et al. 1997), (Saffiotti et al. 1993).

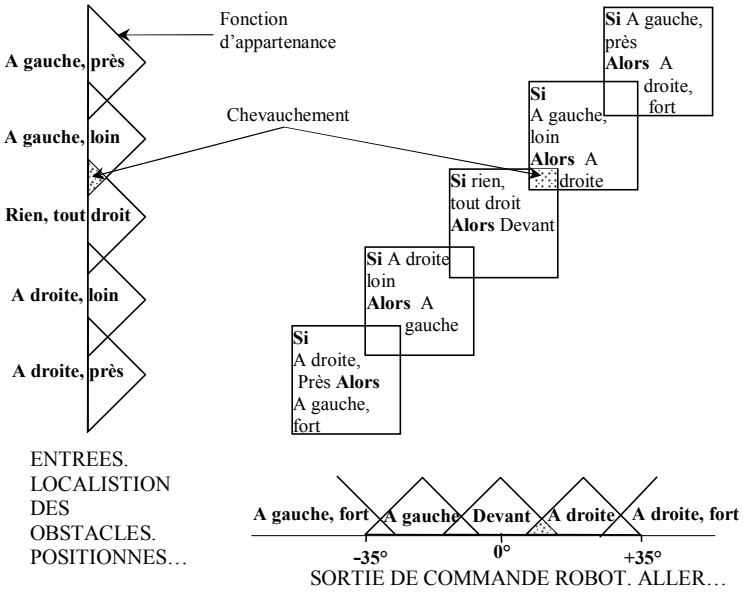


FIG. 3.13 – Comportement d'*évitement d'obstacles* contrôlé par logique floue

Commentaires

La définition précise des règles floues, ainsi que des caractéristiques (formes et position) des fonctions d'appartenance, sont parmi les contraintes majeures d'une utilisation efficace d'un contrôleur flou. Leur définition peut s'avérer donc comme une tâche fastidieuse à mettre en oeuvre, et peut provoquer, si ces fonctions sont mal définies, des conflits importants entre comportements (Pirjanian & Mataric´ 1999). Cependant, plusieurs techniques d'apprentissage sont utilisées dans la littérature, pour essayer de réduire voire éliminer les contraintes citées ci-dessus. Parmi ces techniques d'apprentissage nous pouvons citer : les algorithmes génétiques (Barberá & Skarmeta 2002) ou les techniques neuro-floues (Godjevac 1997).

3.4 Conclusion

Après cette description représentative des différents types d'architectures de contrôle comportementales proposés dans la littérature, nous sommes persuadés que le développement ascendant "*Bottom-up*" de ces architectures de contrôle est un moyen prometteur pour briser la complexité inhérente aux systèmes multi-robots. Néanmoins, contrôler ce genre de systèmes exige à notre sens de garder une maîtrise absolue de ses mécanismes de contrôle. En effet, le fait d'engendrer une complexité trop importante de ces mécanismes[12], ne nous ferait hélas que nous ramener vers le départ du cycle de conception.

Nous allons passer à présent à la partie II du rapport qui contient :

- les chapitres IV et V où sont décrites d'une manière détaillée les deux architectures de contrôle comportementales que nous avons proposées pour le contrôle d'un groupe de robots mobiles minimalistes,
- le chapitre VI où est présentée la méthodologie que nous avons proposée et appliquée pour optimiser, en utilisant les algorithmes génétiques, l'architecture de contrôle décrite au chapitre V.

[12] Au sens que l'on n'arrive plus à établir la causalité existante entre les paramètres de contrôle et les actions engendrées par l'architecture de contrôle.

Deuxième partie

Contrôle d'un Groupe de Robots Minimalistes

"Il faudrait faire en sorte que tout soit le plus simple possible, mais pas plus simple".
Albert Einstein.

"Pascal le disait déjà, je ne peux comprendre un tout que si je connais particulièrement les parties, mais je ne peux comprendre les parties que si je connais le tout".
Edgar Morin.

Chapitre 4

Processus de Sélection d'Action Hiérarchique "PSAH"

Résumé : *L'objectif visé du contrôle que nous voulons mettre en place correspond à commander un groupe de plusieurs dizaines de robots mobiles minimalistes, qui doivent coopérer pour réaliser des tâches complexes. Dans ce contexte, l'approche bio-inspirée émanant des processus rudimentaires d'organisation des sociétés d'insectes s'est distinguée plus particulièrement des autres approches possibles. Ce choix impose néanmoins des contraintes importantes par rapport, d'une part à la structure et à l'intelligence propre de chaque robot mobile, et d'autre part à la méthodologie à adopter pour la mise en oeuvre de l'architecture de contrôle. En effet, c'est dans le but d'obtenir des architectures de contrôle complètement distribuées*[a] *et réactives, que nous proposons dans ce chapitre, un mécanisme de sélection d'action intuitif de bas niveau nommé PSAH "Processus de Sélection d'Action Hiérarchique".*

[a] Que ce soit au niveau du robot lui même (architecture comportementale) qu'au niveau du groupe de robots qui ne reçoit pas d'ordres, d'un superviseur.

4.1 Processus de Sélection d'Action Hiérarchique "PSAH"

Le développement, la maîtrise et la cohérence du fonctionnement d'une architecture de contrôle comportementale, passe inéluctablement par la maîtrise du flux de commandes généré par la multitude de comportements cohabitant dans la même structure de contrôle (cf. chapitre III). En d'autres termes, il faut déterminer parmi l'ensemble des commandes[1] générées par les comportements élémentaires à chaque pas d'échantillonnage, celles qui vont contribuer effectivement aux actions du robot. Les interactions existantes entre comportements doivent être par conséquent complètement maîtrisées et ceci pourquoi pas à l'aide de mécanismes d'une simplicité affligeante ! En effet, c'est la maîtrise des relations entre comportements qui pourra nous permettre d'obtenir des comportements de plus en plus complexes (via l'adjonction de nouveaux comportements) sans toutefois perdre la flexibilité, la robustesse et la prédictibilité de l'architecture de contrôle proposée. Les architectures de contrôle proposées, par exemple dans (Maes 1991*c*), ou dans (Parker 1998), nous donnent un aperçu de certains mécanismes de coordination entre comportements qui peuvent provoquer des situations bloquantes, cycliques et même imprédictibles des robots mobiles. Cela est dû essentiellement à la non-maîtrise des interactions existantes entre comportements élémentaires.

Avant d'aborder le Processus de Sélection d'Action Hiérarchique "PSAH", que nous proposons dans ce chapitre pour coordonner l'activité d'un ensemble de comportements élémentaires, nous donnons au préalable, quelques définitions liées au contrôle d'un robot mobile.

4.1.1 Comportement élémentaire

Nous caractérisons le fonctionnement d'un comportement élémentaire (cf. figure. 4.1) par une application $\mathcal{F}$ de l'ensemble des stimuli $\mathbb{S}$ (qui sont observables par le comportement élémentaire) vers l'ensemble des commandes $\mathbb{C}$ (qui peuvent être générées par le comportement élémentaire).

$$\begin{aligned} \mathcal{F}: \quad & \mathbb{S} \longrightarrow \mathbb{C} \\ & s \longmapsto c = \mathcal{F}(s) \end{aligned} \tag{4.1}$$

Avec :
- $s \in \mathbb{S}$ est un stimulus élémentaire,
- $c \in \mathbb{C}$ est une commande élémentaire.

[1] Générées par des comportements élémentaires ou par la fusion d'un ensemble de comportements.

Il est à noter que les deux ensembles $\mathbb{S}$ et $\mathbb{C}$ peuvent contenir, soit un nombre fini (discret), soit un nombre infini (continu variant dans un intervalle borné) d'éléments. Ceci dépendra à la fois des capteurs et des actionneurs utilisés par le robot, ainsi que de la stratégie de contrôle adoptée.

FIG. 4.1 – Comportement élémentaire

4.1.2 Modèle cinématique

La structure robotique mobile que nous utilisons autant en simulation qu'en manipulation, est celle du mini-robot ALICE (Caprari et al. 2002) (cf. figure. 4.2). Le robot ALICE est non holonome (Laumond 2001) de type char (robot différentiel). Son déplacement est donc contrôlé en agissant sur les vitesses de rotation de ses deux roues motrices. La stabilité du robot est assurée par un élément analogue à une roue folle (voir chapitre VIII pour plus de détails).

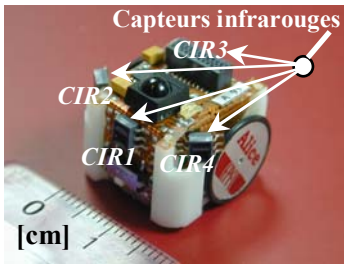


FIG. 4.2 – Mini-robot ALICE

Le modèle cinématique d'un robot mobile de type char, est le suivant :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \frac{r}{2} \begin{bmatrix} \cos(\varphi) & \cos(\varphi) \\ \sin(\varphi) & \sin(\varphi) \\ \frac{1}{l} & -\frac{1}{l} \end{bmatrix} \begin{bmatrix} V_d \\ V_g \end{bmatrix} \tag{4.2}$$

Avec :

- $\dot{x}, \dot{y}$ les composantes en x et en y de la vitesse linéaire instantanée $\overrightarrow{V}$ du robot, prise au milieu de l'axe de ses roues motrices (cf. figure. 4.3),
- $\dot{\varphi} = \omega$ la vitesse angulaire instantanée du robot,
- φ l'orientation du robot dans le repère absolu $(\boldsymbol{x},\boldsymbol{y})$,

– V_d, V_g les vitesses angulaires respectives des roues droite et gauche,
– r rayon des roues,
– $2\times l$ distance entre les roues.

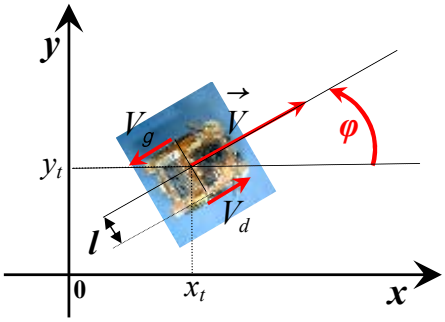


FIG. 4.3 – Variables caractéristiques d'un robot de type char dans le plan

Comme l'indique le modèle cinématique (cf. équation. 4.2), le contrôle du robot mobile se fait par l'intermédiaire des vitesses angulaires de ses roues droite et gauche, c'est-à-dire respectivement V_d et V_g. Dans l'objectif d'avoir des commandes qui soient plus explicites et interprétables que celles formulées avec V_d et V_g, nous avons choisi de contrôler le robot dans l'espace opérationnel, ce qui se traduit par une juxtaposition d'une commande en vitesse de translation V et une en vitesse de rotation ω. C'est-à-dire qu'à partir des vitesses de translation et de rotation désirées, nous devons déterminer V_d et V_g correspondantes (cf. équation. 4.3).

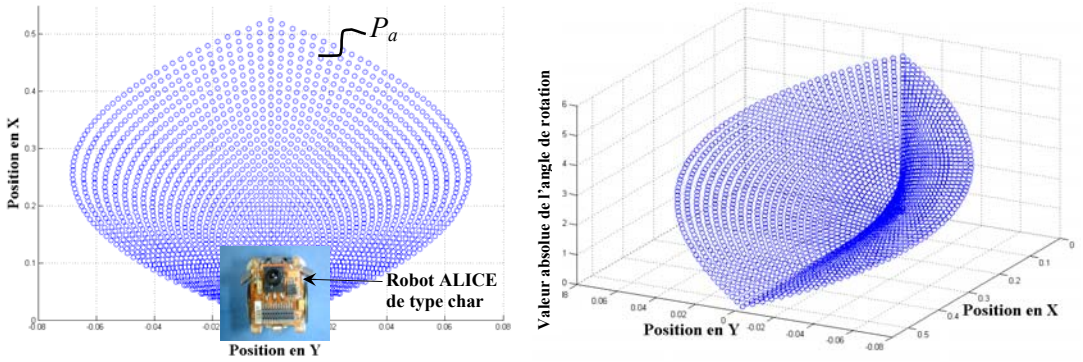
$$\begin{bmatrix} V_d \\ V_g \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & l \\ l & -l \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix} \tag{4.3}$$

L'élément de commande $c \in \mathbb{C}$ (cf. équation. 4.1) généré à chaque pas d'échantillonnage par un comportement élémentaire, doit avoir bien évidemment ses valeurs de V et ω bornées, et ce vu les contraintes structurelles inhérentes aux actionneurs. Nous allons nous intéresser au cas le plus général, où les comportements élémentaires génèrent des commandes $c \in \mathbb{C}$ continues et bornées, ce qui implique par voie de transitivité que V et ω le sont aussi, c'est-à-dire que :

$$\begin{cases} V \in [V_{Min},\ V_{Max}] \\ \text{et} \\ \omega \in [\omega_{Min},\ \omega_{Max}] \end{cases} \tag{4.4}$$

Nous adopterons dans ce qui suit la notation suivante : chaque commande $c_i \in \mathbb{C}$ générée par le comportement "$_i$" sera représentée par $c_i(Rot,\ Tra)$ et correspondra à commander le robot pour qu'il fasse une rotation de Rot [°] et une translation de Tra [cm], commandes qui vont s'exécuter en un mouvement couplé[2].

Pour que l'application $\mathcal{F}(s)$ (cf. équation. 4.1) soit en adéquation avec les contraintes structurelles du robot (e.g., couple moteur borné, non-holonomie du robot), il faut que, $\forall\ c \in \mathbb{C}$ alors c doit être réalisable par la structure matérielle du robot, c'est-à-dire que le robot peut atteindre la configuration (position et orientation) désirée en un temps suffisamment petit qu'il puisse être considéré comme infinitésimal. La figure 4.4 donne la forme générale des positions et orientations *atteignables*[3] "P_a" par le robot en un pas d'échantillonnage. Il est à noter, d'une part que nous considérons dans cette représentation que le robot ne peut qu'avancer et d'autre part, que le robot ALICE représenté dans la figure 4.4 est là uniquement pour donner l'orientation initiale du robot, et en aucun cas, pour montrer exactement les échelles de déplacements accessibles par le robot en un pas d'échantillonnage.



(a) Positions atteignables P_a

(b) Orientations atteignables par ALICE en fonction de ses positions atteignables

FIG. 4.4 – Forme générale de l'espace atteignable "P_a" par le robot ALICE en un pas d'échantillonnage.

[2]Il est à noter que si le robot utilisé ne dispose que d'une faible inertie (c'est le cas du mini-robot ALICE), alors nous pouvons dire que les commandes c envoyées aux actionneurs, sont réalisées sans aucun retard (à condition que c respecte bien évidemment les équations 4.3 et 4.4).

[3]Ceci est nommé dans (Laumond 2001) "domaine d'accessibilité".

4.1.3 Fonctionnement du Processus de Sélection d'Action Hiérarchique entre comportements

La viabilité ou non des tâches exécutées par un robot mobile dans un espace de travail quelconque, dépend principalement des configurations successives qu'il va occuper au cours du temps[4]. Ces configurations successives caractérisent complètement le mouvement du robot. Dans le cadre du contrôle d'un bras manipulateur par exemple, on conçoit le mouvement presque exclusivement comme un moyen d'atteindre un but, d'effectuer une tâche. Le mouvement n'a pas de statut "en soi" en dehors du contexte de la tâche. Si le mouvement peut être engendré et justifié en tant que résolution de tâche, la notion de tâche elle-même peut réciproquement s'enrichir de l'étude du mouvement "en soi".

La *cohabitation* existante entre comportements au sein d'une architecture de contrôle comportementale, ne peut être considérée comme optimale, que si elle est en mesure de faire adopter au robot les configurations successives les plus favorables pour l'exécution des tâches désirées. Pour la gestion de la coordination entre comportements, nous proposons une heuristique qui fonde son raisonnement sur l'espace des positions atteignables P_a par le robot. Pour illustrer nos propos, nous décomposons, par exemple, l'espace atteignable par le robot en six sous-espaces distincts nommés $P_i|_{i=0..5}$ (cf. figure. 4.5).

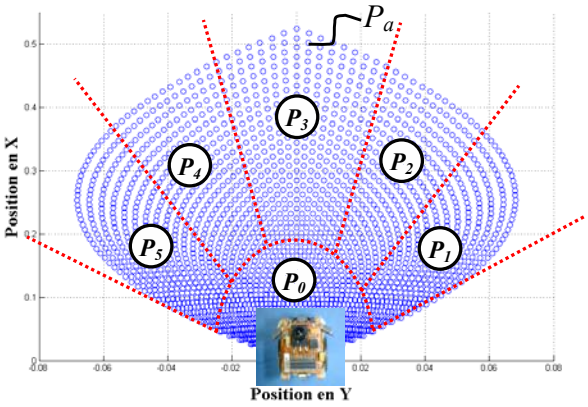


FIG. 4.5 – Exemple d'une décomposition de l'espace atteignable P_a du robot en six sous-espaces atteignables distincts $P_i|_{i=0..5}$.

[4]Ces configurations dans le cas d'un robot réactif, dépendent essentiellement de ses perceptions, de ses objectifs et des contraintes liées à la dynamique de l'environnement dans lequel il est immergé.

Chaque comportement$_i$ génère des commandes $c_i \in \mathbb{C}_i$ qui vont faire déplacer le robot vers des configurations incluses dans P_a. Parmi l'ensemble des commandes $\mathbb{C}_i$ qui peuvent être générées par le comportement$_i$, il existe un sous-ensemble de commandes que nous nommons "**Commandes Refuges**" $\mathbb{CR}_i$ qui vont déplacer le robot vers des positions spécifiques pour le comportement$_i$, nommées "**Positions Refuges**" $\mathbb{PR}_i$. Ces $\mathbb{PR}_i$ seront caractérisées par un ou plusieurs sous-ensembles de positions atteignables incluses dans P_a (cf. figure. 4.5).

Les $\mathbb{CR}_i$, qui vont mener le robot à l'instant $t + \delta t$ vers les $\mathbb{PR}_i$, sont caractéristiques pour le comportement$_i$, au sens qu'elles sont pour différentes raisons, peu pertinentes voire indésirables pour le bon fonctionnement du comportement en question, et donc de l'architecture de contrôle vue dans sa globalité. Ainsi, chaque comportement élémentaire doit pouvoir émettre une *réserve* sur l'application de ses commandes, et ceci en discriminant ses commandes pertinentes (fiables) de celles plutôt refuges (peu fiables, sans utilité). Cette discrimination des commandes au niveau des comportements élémentaires, se fait principalement en fonction de leur spécialisation (e.g., l'évitement d'obstacles, l'attraction vers un objectif, etc. (cf. §4.1.4)), et de la définition de l'application $\mathcal{F}$ qui les décrit.

L'idée de base du Processus de Sélection d'Action Hiérarchique "PSAH" que nous proposons, part du principe d'une *hiérarchie décisionnelle* entre comportements. Ce processus est un peu à l'image de l'architecture de subsomption décrite dans (Brooks 1986) (cf. §3.2.1, page 54) où un comportement de haut niveau peut inhiber un comportement de plus bas niveau. Nous proposons au travers du PSAH de réaliser cette hiérarchie en utilisant les notions de Commandes Refuges "$\mathbb{CR}$" et de Positions Refuges "$\mathbb{PR}$" décrites ci-dessus. La figure 4.6 donne le principe élémentaire de coordination entre comportements en utilisant le PSAH. Un comportement de haut niveau (e.g., le comportement$_1$ sur la figure 4.6) laisse la main aux comportements de plus bas niveau, uniquement lorsqu'il génère une commande $c \in \mathbb{CR}_1$ (cf. figure. 4.6). Le comportement$_1$ autorise alors les autres comportements de plus bas niveau (e.g., le comportement$_2$ sur la figure 4.6) à prendre la main pour éventuellement appliquer leur commande.

L'algorithme 4.1 donne un aperçu de la simplicité de mise en oeuvre de la coordination entre comportements en utilisant le PSAH. Celui-ci est appliqué bien évidemment une fois définie au préalable les commandes refuges propres à chaque comportement élémentaire.

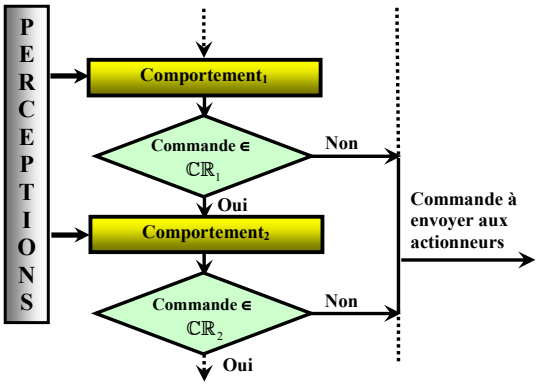


FIG. 4.6 – Processus de Sélection d'Action Hiérarchique "PSAH" entre comportements

Algorithme 4.1 Coordination entre comportements élémentaires en utilisant le PSAH

Répéter
$\mathcal{C} = \mathcal{C}_{Comportement_1}$; //Récupération de la commande générée par le $Comportement_1$
Si $\mathcal{C} \notin \mathbb{CR}_1$ **Alors**
GOTO EnvoiCommande ; //C'est une étiquette
Fin Si
$\mathcal{C} = \mathcal{C}_{Comportement_2}$;
Si $\mathcal{C} \notin \mathbb{CR}_2$ **Alors**
GOTO EnvoiCommande ;
Fin Si
⋮
$\mathcal{C} = \mathcal{C}_{Comportement_n}$; //C'est le comportement de plus bas niveau
EnvoiCommande : Appliquer la commande $\mathcal{C}$ sur les actionneurs du robot.
Jusqu'à Fin expérimentation

4.1.4 Mise en oeuvre du PSAH sur une tâche de navigation

Nous allons illustrer le fonctionnement du PSAH proposé, via un exemple simple de coordination entre le comportement d'*évitement d'obstacles* et celui d'*attraction à la lumière* (cf. figure. 4.7). L'objectif en coordonnant ces deux comportements élémentaires est de faire tendre le robot le plus rapidement vers une source de lumière tout en évitant les obstacles "*tâche de navigation en présence d'obstacles*". Il est à noter que les deux comportements cités ci-dessus peuvent générer dans de nombreuses configurations, des commandes antagonistes telles qu'aller à droite versus aller à gauche.

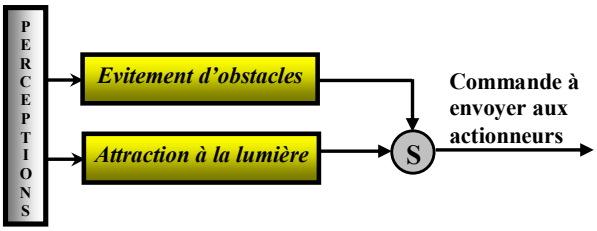


FIG. 4.7 – Architecture de contrôle simple en utilisant deux comportements élémentaires

Avant d'aller plus loin dans le détail concernant l'application effective du PSAH au cas de la tâche de navigation en présence d'obstacles, nous notons que les deux applications $\mathcal{F}_{EvitementObstacles}$ et $\mathcal{F}_{AttractionLumière}$ sont de nature continue[5] et correspondent respectivement aux comportements d'*évitement d'obstacles* et d'*attraction à la lumière* qui sont décrits en Annexe B.

La figure 4.8 montre la forme générale de l'étendue des positions atteignables[6] $P_{a\_EvitementObstacles}$ et l'ensemble des positions refuges $\mathbb{PR}_{EvitementObstacles}$ associés au comportement d'*évitement d'obstacles*. Nous pouvons remarquer que l'étendue de ces positions refuges dépend exclusivement de l'angle d'ouverture Ω.

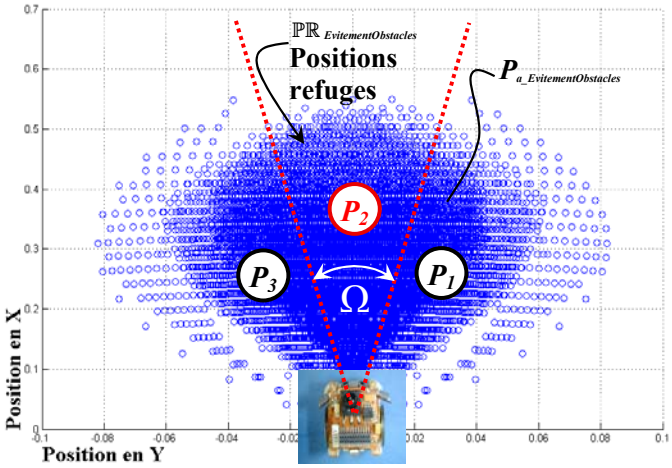


FIG. 4.8 – Positions atteignables "$P_{a\_EvitementObstacles}$" et positions refuges "$\mathbb{PR}_{EvitementObstacles}$" du comportement d'*évitement d'obstacles*.

[5] C'est-à-dire que ces applications créent une relation des stimuli continus vers des commandes continues (cf. §4.1.1, page 78).

[6] Ces positions sont obtenues en discrétisant l'espace des perceptions possibles des capteurs infrarouges du robot ALICE, et en appliquant par la suite l'application $\mathcal{F}_{EvitementObstacles}$ (voir Annexe B) qui lie les perceptions aux commandes.

La justification du positionnement des $\mathbb{PR}_{EvitementObstacles}$ par rapport à $P_{a\_Evitement\ Obstacles}$, suit pour le cas du comportement d'*évitement d'obstacles* le postulat suivant :

> L'objectif de fonctionnement d'un comportement d'évitement d'obstacles, est de virer plus ou moins à gauche ou à droite en fonction de l'imminence ou non de collisions du robot avec des obstacles. Autrement dit, si le comportement d'évitement d'obstacles donne en sortie une commande qui aura comme effet de dévier que **légèrement** à droite ou à gauche le robot (représenté par l'ouverture de l'angle Ω dans la figure 4.8), alors cela signifie :
> - soit la non-imminence d'une collision avec des obstacles. Les positions refuges $\mathbb{PR}_{EvitementObstacles}$ dans ce cas de figure, représenteront une forme de tolérance à la présence d'obstacles aux alentours du robot,
> - soit l'ambiguïté du comportement à donner une commande cohérente, due par exemple, à une disposition particulière des obstacles autour du robot.

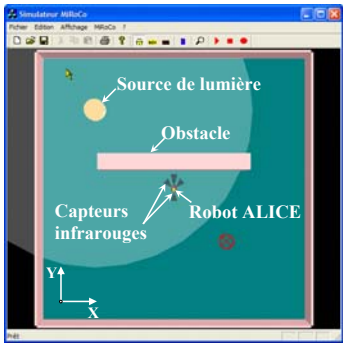
Dans les deux cas de figure cités ci-dessus, le comportement d'*évitement d'obstacles*, générera une commande refuge et laissera la main au comportement d'*attraction à la lumière* (cf. figure. 4.7 et 4.6).

L'objectif dans ce qui suit est de constater l'incidence que peut avoir la disposition des positions refuges propres au comportement d'*évitement d'obstacles* sur la réalisation de la tâche de navigation en présence d'obstacles. Pour cela, nous faisons varier l'angle Ω (cf. figure. 4.8), et relevons les trajectoires (cf. figure. 4.9) et les temps nécessaires (cf. tableau. 4.1) à la réalisation de la tâche pour différents environnements de tests.

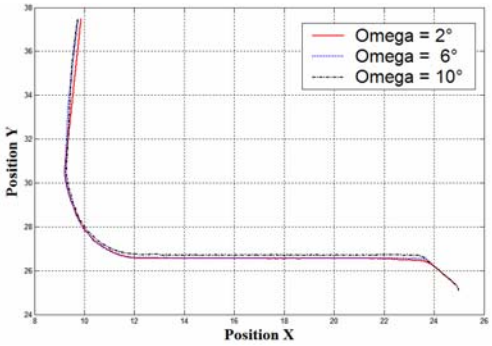
	$\Omega = 2^\circ$	$\Omega = 6^\circ$	$\Omega = 10^\circ$
Environnement 1	352	353	368
Environnement 2	484	484	567
Environnement 3	695	699	772

TAB. 4.1 – Nombre d'itérations nécessaires pour que le robot atteigne son objectif

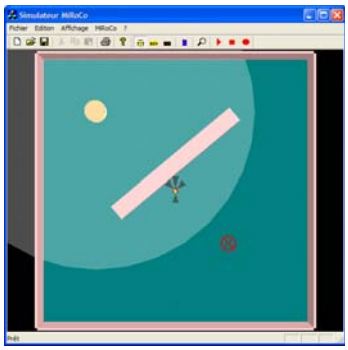
Les résultats obtenus sont certainement étroitement liés à différents paramètres, comme la configuration des obstacles dans l'environnement (position, dimension, forme, nombre) ou la manière dont sont implémentés les comportements élémentaires. Néanmoins, ce qui est notable et à priori confirmé par les simulations, correspond au fait que la tâche d'attraction à la lumière en présence d'obstacles s'effectue d'autant plus rapidement que Ω est petit (cf. tableau. 4.1). Si on fait tendre Ω vers zéro, cela correspond à définir une tolérance aux obstacles quasi nulle et a pour effet d'éloigner davantage les



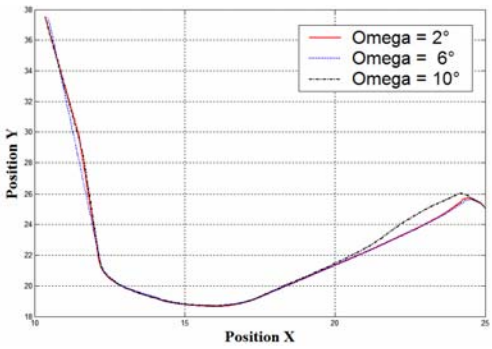
(a) Environnement 1



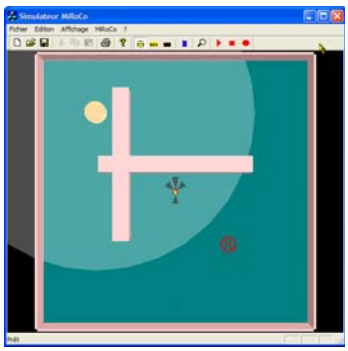
(b) Trajectoires du robot pour différents Ω



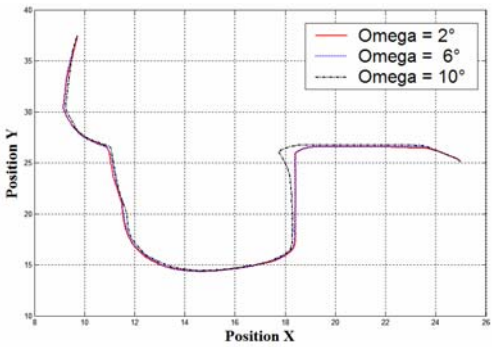
(c) Environnement 2



(d) Trajectoires du robot pour différents Ω



(e) Environnement 3



(f) Trajectoires du robot pour différents Ω

FIG. 4.9 – Trajectoires du robot obtenues pour différents types d'environnement tests et pour différents Ω.

trajectoires du robot des obstacles présents dans l'environnement (cf. figure. 4.9).

Nous avons donc pu réaliser grâce à l'utilisation du PSAH, une coordination entre deux comportements pouvant être antagonistes d'une manière complètement *hiérarchique*

et *intuitive*, et ceci sans avoir recours, ni à une planification de trajectoire, ni même à des processus cognitifs de haut niveau (cf. §2.2.1, page 42). Afin de vérifier plus avant l'adéquation du PSAH, nous allons l'appliquer dans ce qui suit, sur une architecture de contrôle comportementale plus complexe.

4.2 La tâche générique de poussée d'objets coopérative

Parmi la multitude de tâches génériques rencontrées dans la littérature (cf. §1.2.1.2, page 17) nous avons focalisé nos efforts sur la tâche coopérative de poussée d'objets, qui va servir de base d'essais aux différentes possibilités liées aux architectures de contrôle que nous proposons. L'objectif général de cette tâche est de pouvoir déplacer d'une manière coopérative un objet d'une position initiale vers une destination finale. L'objet à pousser, est choisi de telle sorte que son poids et/ou ses dimensions soient largement plus élevés que ceux d'un robot élémentaire. La coopération dans ce cas de figure est donc impérative.

Généralement, le mouvement d'un objet poussé par une multitude de forces de contacts reste difficilement prédictible (Lynch 1999), et ce pour différents problèmes complexes liés à la manipulation d'objets sur une surface plane. Les travaux de Matthew Mason (Mason 1986) et Kevin Lynch (Lynch 1992) montrent clairement les difficultés liées à la réalisation de la tâche de poussée d'objets par une multitude de manipulateurs mobiles (Lynch & Mason 1996). Ce qui peut paraître évident pour nous, telle que la prédiction du mouvement d'un objet poussé, ne l'est pas forcément dans la réalité, surtout si plusieurs forces de contacts sont exercées simultanément sur cet objet. En réalité, le mouvement de l'objet poussé dépend à la fois de la mécanique des frottements (secs et visqueux)/glissement de l'objet sur la surface de déplacement, de l'emplacement du centre de frottement instantané de l'objet, de la géométrie et des coefficients de frottements des contacts pousseurs-objet. Tous les points cités ci-dessus montrent clairement les difficultés qui doivent être surmontées pour contrôler la poussée coopérative d'objets, en utilisant un modèle prédictif de l'évolution de l'objet poussé en fonction des forces exercées.

L'architecture de contrôle que nous proposons pour l'exécution de la Tâche Coopérative de Poussée d'Objets "TCPO", n'utilise ni modèle physique, ni modèle géométrique liés aux objets à déplacer. Les robots mobiles, que nous utilisons sont considérés comme complètement réactifs, sans aucun modèle du monde qui les entoure et ceci à l'image des individus constituant les sociétés d'insectes (Bonabeau et al. 1999). La réalisation de la TCPO est rendue encore plus difficile dans nos travaux du fait de l'interaction d'un

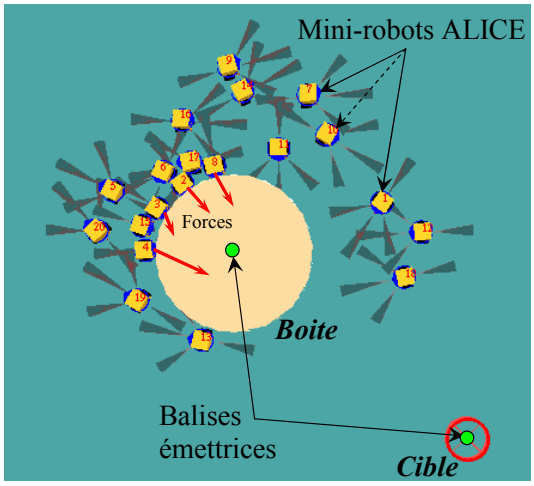


FIG. 4.10 – Tâche coopérative de poussée d'objets "TCPO"

grand nombre de robots dans un milieu très confiné (alentours immédiats de l'objet à pousser), et de l'utilisation de structures robotiques très simples. Effectivement, le mini-robot ALICE est très limité, que ce soit du point de vue structurel que décisionnel (cf. chapitre VIII pour plus de détails).

Nous pouvons résumer la TCPO (cf. figure. 4.10) telle que nous nous sommes attachés à la traiter dans nos travaux de recherche, comme suit : nous disposons d'un groupe de "N" mini-robots devant pousser une boite symbolisée par "B" vers une zone déterminée appelée cible symbolisée par "C". Il est à noter que la boite à pousser comme la cible à atteindre, disposent de balises émettrices caractéristiques qui permettent aux robots de les détecter et de les distinguer dans l'environnement. Le déplacement de la boite exige la coopération d'un nombre critique "Nc" de robots (c'est-à-dire que N doit être supérieur ou égal à Nc pour que la boite puisse se déplacer).

4.3 Architecture de contrôle proposée

L'architecture de contrôle que nous appliquons à la TCPO (cf. figure. 4.11) est composée d'un certain nombre de comportements génériques et d'autres spécifiques à la tâche à réaliser. Les détails du fonctionnement et de l'implémentation des comportements élémentaires sont décrits dans les paragraphes suivants.

Dans l'objectif d'avoir des robots les plus réactifs possibles (cf. §2.2.2, page 44), nous avons défini, d'une part des comportements élémentaires fonctionnant uniquement

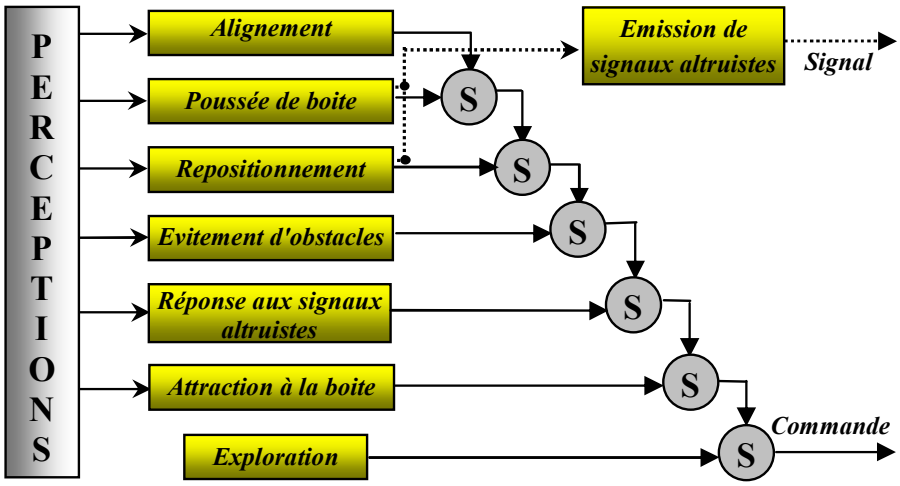


FIG. 4.11 – Architecture de contrôle à base de PSAH appliquée à la tâche coopérative de poussée d'objets "TCPO".

sur des principes de stimulus-réponse, et d'autre part, nous utilisons le PSAH en mode *discret*, ce qui le rend amplement plus simple à utiliser pour des robots minimalistes.

Pour cela nous avons développé des comportements élémentaires, qui disposent que d'un nombre *discret* de sorties de commande. La totalité de ces commandes élémentaires $\mathcal{C}$ appartient à l'ensemble des commandes discrètes $\mathbb{C}_d$ (qui rappelons le, sont constituées d'une commande en rotation et d'une commande en translation). Elles sont au nombre de sept (cf. figure. 4.12) :

1. *Aller Devant* $\boldsymbol{AD} = \mathcal{C}(0.0^\circ, \mathit{Dist}\text{ cm})$,
2. *Aller En Arrière* $\boldsymbol{AEA} = \mathcal{C}(0.0^\circ, \text{-}\mathit{Dist}\text{ cm})$,
3. *Aller A Droite* $\boldsymbol{AAD} = \mathcal{C}(\text{-}\mathit{Angle}^\circ, \mathit{Dist}\text{ cm})$,
4. *Tourner A Droite* $\boldsymbol{TAD} = \mathcal{C}(\text{-}\mathit{Angle}^\circ, 0.0\text{ cm})$,
5. *Aller A Gauche* $\boldsymbol{AAG} = \mathcal{C}(+\mathit{Angle}^\circ, \mathit{Dist}\text{ cm})$,
6. *Tourner A Gauche* $\boldsymbol{TAG} = \mathcal{C}(+\mathit{Angle}^\circ, 0.0\text{ cm})$,
7. *Rester Sur Place* $\boldsymbol{RSP} = \mathcal{C}(0.0^\circ, 0.0\text{ cm})$.

Avec : "*Angle*" et "*Dist*" des constantes positives.

Le PSAH sera appliqué exactement comme cela est décrit en section 4.1.3, à l'exception, qu'au lieu de caractériser chaque comportement "i" par un ensemble de positions refuges $\mathbb{PR}_i$ (cf. figure 4.8, pour l'exemple du comportement d'*évitement d'obstacles*

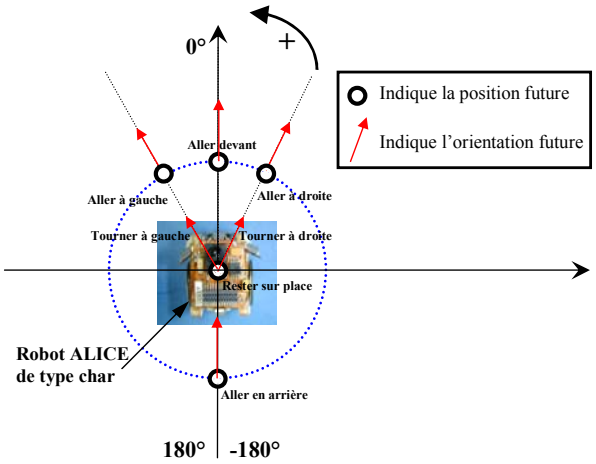


FIG. 4.12 – Espace atteignable discret

continu), le comportement "i" ne disposera, par mesure de simplicité, que d'une seule position refuge $\mathcal{PR}_i$. Cette position sera atteinte aussi en appliquant une seule commande refuge nommée $\mathcal{CR}_i$.

4.3.1 Les comportements élémentaires

Les comportements élémentaires sont les briques de base des architectures comportementales. C'est pour cette raison qu'il faut accorder le maximum d'attention à leurs phases de développement et de test. La construction ascendante *Bottom-Up* de l'architecture de contrôle (cf. figure. 4.11) permet d'isoler un ou plusieurs comportements élémentaires afin de tester et de vérifier leur robustesse et efficacité à réaliser une primitive ou une séquence de comportements bien déterminée. L'étape finale consiste bien évidemment, à assembler la multitude des comportements élémentaires développés, dans une architecture de contrôle unique, qui soit à la fois efficace et cohérente pour réaliser la tâche désirée.

Pour certains comportements élémentaires, il est nécessaire, parfois indispensable de prendre en compte les caractéristiques structurelles du robot mobile utilisé (l'hôte) (Pfeifer 2000), (Lichtensteiger 2000) (cf. §1.1.2, page 9). Le mini-robot ALICE (cf. figure. 4.2) a des dimensions de (2cm×2cm×2cm) et dispose de quatre capteurs infrarouges de proximité *CIR1*, *CIR2*, *CIR3*, *CIR4* positionnés respectivement à 0°, 45°, 180°, -45° par rapport à la face avant du robot. Ces capteurs détectent les obstacles à une distance maximale de 4cm, ce qui restreint le robot à des informations très localisées.

Nous notons que chaque robot, dispose de possibilités sensorielles qui lui permettent de mesurer les angles $\theta_i|_{i=1,2}$, qu'il fait avec deux balises actives distinctes dans l'environnement. Plus spécifiquement, pour le cas de la TCPO, ce sont les balises qui sont placées respectivement sur l'objet à pousser et sur la cible à atteindre (cf. figure. 4.10). Les angles θ_1 et θ_2 mesurés sont compris entre]-180°, 180°] (cf. figure. 4.12 et 4.13). Le détail du dispositif matériel préconisé et de son implémentation pour la mesure d'angle seront traités au chapitre VIII.

Pour ce qui est des simulations faites avec l'architecture de contrôle proposée, nous considérons que la mesure de cet angle est soumise à une incertitude pouvant aller jusqu'à $\pm 15°$ de l'angle réel que fait le robot avec la ou les balises (i.e., $\theta_i = \theta_{i_{Reel}} \pm 15°$). Cette forme d'incertitude est similaire à celle liée à l'orientation des insectes ou à certains animaux, par rapport à des positions caractéristiques dans l'environnement (nid, source de nourriture, lieu de migration, étoiles dans le ciel, etc.) (Cornetz 1911), (Wehner 1987).

Dans ce qui suit, nous allons expliquer le détail de tous les comportements élémentaires génériques et dédiés implémentés pour le cas de la TCPO.

4.3.1.1 Comportement d'exploration

Ce comportement donne en sortie les commandes appartenant à l'ensemble des commandes discrètes $\mathbb{C}_d$, et ceci en fonction de coefficients (prédéterminés) constants de probabilité d'occurrence de chaque commande. Les probabilités d'occurrence des commandes sont choisies comme suit :

- 0.70 pour la commande ***AD***,
- 0.15 pour chacune des commandes de ***AAD*** et ***AAG***,
- 0.00 pour les autres commandes $\in \mathbb{C}_d$.

Vu les valeurs choisies des probabilités d'occurrence de chaque commande, nous avons donc opté pour que le robot ait une tendance nette à aller devant lui, tout en ayant une légère probabilité d'aller à droite ou à gauche et ce pour une meilleure exploration de l'environnement.

4.3.1.2 Comportement d'attraction à la boite

Ce comportement consiste à attirer le robot vers la boite à pousser qui est dotée d'une balise émettrice. Les capteurs infrarouges *CIR1*, *CIR2*, *CIR4* du robot sont utilisés en simple réception et les commandes envoyées aux moteurs dépendent des intensités des signaux mesurés par les trois capteurs infrarouges placés à l'avant du robot (cf. tableau.

4.2). On procède en mode tout ou rien en affectant la valeur "1" aux capteurs dont l'intensité mesurée est la plus grande et "0" pour les autres.

Il est à noter que le comportement qui consiste à remonter un champ de gradient est très largement observé dans la nature, comme le cas des bactéries qui remontent une piste chimique (Bray 1992), (Marken & Powers 1989), ou des fourmis qui sont attirées par les sources de phéromones (Calenbuhr & Deneubourg 1989).

CIR1	*CIR2*	*CIR4*	***Commande***
0	0	0	$\boldsymbol{AD}$
0	0	1	$\boldsymbol{AAG}$
0	1	0	$\boldsymbol{AAD}$
0	1	1	$\boldsymbol{RSP} \equiv \mathcal{CR}$
1	0	0	$\boldsymbol{AD}$
1	0	1	$\boldsymbol{AAG}$
1	1	0	$\boldsymbol{AAD}$
1	1	1	$\boldsymbol{RSP} \equiv \mathcal{CR}$

TAB. 4.2 – Stratégie adoptée pour attirer les robots vers la boite à pousser

4.3.1.3 Comportement d'évitement d'obstacles

Ce comportement utilise les quatre capteurs infrarouges toujours en mode booléen ("1" : détection d'obstacle et "0" : pas de détection). Ce comportement consiste donc à éviter les obstacles selon une stratégie pré-établie, donnant les commandes à envoyer aux roues en fonction des seize stimuli possibles détectables par l'ensemble des capteurs infrarouges (cf. tableau. 4.3).

4.3.1.4 Comportement de repositionnement

Le but de ce comportement est de repositionner convenablement le robot autour de la boite, pour qu'il soit rapidement, dans une meilleure configuration pour pousser la boite vers la cible. Ce comportement est aussi un moyen de débloquer le robot de situations stagnantes.

Des comportements de déblocage de situations stagnantes, sont aussi observés chez les fourmis qui tentent de déplacer de la nourriture trop lourde pour elles. Les observations des éthologues ont relevé qu'à la rencontre d'un aliment trop lourd par une fourmi, celle-ci commence initialement par l'agripper avec ses mâchoires et essaye après de le tirer en

CIR1	*CIR2*	*CIR3*	*CIR4*	***Commande***
0	0	0	0	$\boldsymbol{AD} \equiv \mathcal{CR}$
0	0	0	1	$\boldsymbol{AAD}$
0	0	1	0	$\boldsymbol{AD} \equiv \mathcal{CR}$
0	0	1	1	$\boldsymbol{AAD}$
0	1	0	0	$\boldsymbol{AAG}$
0	1	0	1	$\boldsymbol{AEA}$
0	1	1	0	$\boldsymbol{AAG}$
0	1	1	1	$\boldsymbol{AD} \equiv \mathcal{CR}$
1	0	0	0	$\boldsymbol{AAD}$
1	0	0	1	$\boldsymbol{AAD}$
1	0	1	0	$\boldsymbol{AAG}$
1	0	1	1	$\boldsymbol{AAD}$
1	1	0	0	$\boldsymbol{AAG}$
1	1	0	1	$\boldsymbol{AEA}$
1	1	1	0	$\boldsymbol{AAG}$
1	1	1	1	$\boldsymbol{AAD}$

TAB. 4.3 – Stratégie adoptée pour le comportement d'évitement d'obstacles

appliquant des réalignements par rapport au sens de déplacement probable de l'aliment. Après un certain temps d'essai, la fourmi procède à un repositionnement radical autour de l'aliment, ici aussi la fourmi y consacre un certain temps, et puis passe en cas d'échec à la phase de recrutement d'autres congénères pour l'aider à déplacer l'aliment vers le nid. Il est à noter que les temps consacrés à chaque phase d'évitement de la stagnation de l'aliment à transporter, dépend de la taille de l'aliment : plus il est lourd, moins long est le temps consacré à ces phases d'empêchement de la stagnation. Ceci montre que la fourmi dispose de moyens proprioceptifs qui lui indiquent la progression de la tâche de transport (Lioni 1999). Ronald Kube dans (Kube & Zhang 1994) adopte les mécanismes cités ci-dessus pour empêcher la stagnation du déplacement de l'objet à pousser par un groupe de robots mobiles. Il définit alors des comportements spécifiques nommés "*stagnation recovery*" qui consistent à enclencher le comportement de réalignement et/ou de repositionnement et/ou de marche arrière "*Back-off*" en utilisant des comptes à rebours, qui donnent une indication sur le temps passé par le robot à la même position (détection des situations de stagnation).

Le comportement de *repositionnement*, que nous proposons ici, est paramétré en fonction de l'angle θ que fait le robot avec la cible C (cf. figure. 4.13). Il génère s'il est le plus prioritaire à l'instant t, une séquence de commandes (Adouane & LeFort-Piat 2003*a*) résumée dans l'algorithme 4.2.

Algorithme 4.2 Comportement de *repositionnement*

Requiert : θ_{Max} = Const ; //Avec Const une constante positive inférieure à 90°
Si (($\theta \geq \theta_{Max}$) et (robot est au voisinage immédiat de l'objet à pousser)) **Alors**

//Calcul des paramètres de repositionnement du robot
$\alpha = \boldsymbol{f}(\theta)$; //Calcul du degré de réorientation en degré [°]
$\boldsymbol{d} = \boldsymbol{g}\ (\theta)$; //Calcul du degré de translation vers l'avant en [cm]

//La réorientation
Si $\theta \geq 0$ **Alors**
Tant que Réorientation de α° n'est pas effective **Faire**
Appliquer la commande $\boldsymbol{TAD}$;
Fin Tant que
Sinon
Tant que Réorientation de α° n'est pas effective **Faire**
Appliquer la commande $\boldsymbol{TAG}$;
Fin Tant que
Fin Si

//La translation
Tant que Translation de $\boldsymbol{d}$ cm n'est pas effective **Faire**
Appliquer la commande $\boldsymbol{AD}$;
Fin Tant que

Sinon
Appliquer la commande $\boldsymbol{RSP} \equiv \mathcal{CR}$;
Fin Si

Le comportement de *repositionnement* va donc commander le robot pour faire tout d'abord :

1. une rotation d'un angle α [°], avec :

$$\alpha = \boldsymbol{f}(\theta). \tag{4.5}$$

2. puis le robot avance d'une distance $\boldsymbol{d}$ [cm], avec :

$$\boldsymbol{d} = \boldsymbol{g}(\theta). \tag{4.6}$$

Où :

- $\boldsymbol{f}$ et $\boldsymbol{g}$ sont des fonctions linéaires de l'angle θ,
- $|\alpha|$ et $\boldsymbol{d}$ sont d'autant plus grands que l'angle $|\theta|$ est grand,
- la direction de la rotation dépend du signe de l'angle θ, telle que :

> **Si** $(\theta \geq 0)$ **Alors** appliquer la commande $\boldsymbol{TAD}$
> **Sinon** appliquer la commande $\boldsymbol{TAG}$.

Il est à noter que les paramètres concrètement choisis pour les fonctions linéaires $\boldsymbol{f}$ et $\boldsymbol{g}$ sont donnés en section 4.4.1, et que la constante $\theta_{\boldsymbol{Max}}$, est aussi utilisée pour le fonctionnement du comportement de *poussée de boite* et celui de *réponse aux signaux altruistes*.

4.3.1.5 Comportement de poussée de boite

Le but de ce comportement est de pousser la boite B de façon à la rapprocher le plus rapidement de la cible C. Après la mesure des angles θ_1 et θ_2 que fait le robot respectivement avec B et C (cf. figure. 4.13), ce comportement génère ses commandes en fonction de la règle élémentaire suivante :

Algorithme 4.3 Comportement de *poussée de boite*

Requiert : $\theta_{\boldsymbol{Max}} =$ Const ;
  Si $(((|\theta_1|$ et $|\theta_2|) \leq \theta_{\boldsymbol{Max}})$ et (le robot est en contact direct avec la boite)) **Alors**
    Appliquer la commande $\boldsymbol{AD}$
  Sinon
    Appliquer la commande $\boldsymbol{RSP} \equiv \mathcal{CR}$.
  Fin Si

Il est à noter, que l'augmentation de la valeur de $\theta_{\boldsymbol{Max}}$ dans l'architecture de contrôle proposée, entraîne la persistance du robot à pousser la boite dans toutes les directions (donc pas forcement vers la cible) (cf. figure. 4.13). Ceci implique aussi des repositionnements moins fréquents du robot.

4.3.1.6 Comportement d'alignement

Ce comportement consiste à s'assurer que le robot est aligné correctement avec la boite avant de permettre l'activation du comportement de *poussée de boite* ou celui du

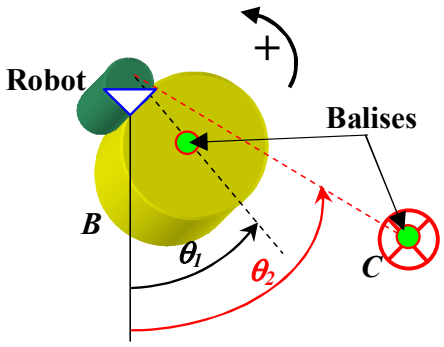


FIG. 4.13 – Informations nécessaires pour exécuter le comportement de poussée d'objets

repositionnement (cf. figure. 4.11). Ces deux comportements ont tous deux besoin de mesurer des orientations relatives par rapport à des points caractéristiques dans l'environnement (boite, cible), et ceci ne peut se faire que lorsque le robot est correctement aligné avec la boite (cette boite sert par conséquent de repère relatif).

Les commandes générées par le comportement d'alignement ont comme but d'asservir l'angle que fait le robot avec la boite, afin qu'il soit compris dans l'intervalle [-η°, η°], tel que η est une constante positive $\ll 90^\circ$. Le fonctionnement du comportement d'*alignement* est résumé dans l'algorithme 4.4.

Algorithme 4.4 Comportement d'*alignement*

Requiert : η = Const ;
  Si (($|\theta| \geq \eta$) et (le robot est en contact direct avec la boite)) **Alors**
    Si $\theta \geq 0$ **Alors**
      Appliquer la commande $\boldsymbol{TAG}$;
    Sinon
      Appliquer la commande $\boldsymbol{TAD}$;
    Fin Si
  Sinon
    Appliquer la commande $\boldsymbol{AD} \equiv \mathcal{CR}$;
  Fin Si

Nous notons que plus η est grand plus le temps d'activation possible du comportement d'*alignement* est faible, donc le robot peut éventuellement pousser la boite plus longtemps. Cependant plus η est grand, plus la pertinence des angles θ_1 et θ_2 mesurés par le robot pour le fonctionnement des comportements de *poussée de boite*, de *repositionne-*

ment et d'*émission des signaux altruistes* est moindre, vu qu'ils ne sont pas calculés en ayant la boite à pousser en point de mire.

4.3.1.7 Comportements altruistes

The altruism as it is described in the nature can don several shapes (Godzinska 1986). According to (Gadagkar 1994) for example, the altruism dons more the sense of the sacrifice whereas Keller (Keller 1997) defines it more as a shape of indirect profit.

L'altruisme tel qu'il est décrit dans la nature peut revêtir plusieurs formes (Godzinska 1986). Selon (Gadagkar 1994) par exemple, l'altruisme revête plus le sens du sacrifice[7] alors que Keller (Keller 1997) le définit plus comme une forme indirecte de bénéfice.

Nous définissons l'altruisme dans le cadre de notre application, comme le fait d'engendrer un effet ou une action dans le but d'aider son prochain, et ceci sans gain apparent immédiat sur l'entité qui engendre cet effet. La notion d'altruisme introduite dans l'architecture de contrôle que nous proposons, se résume dans l'utilisation du comportement d'*émission de signaux altruistes* et celui de *réponse aux signaux altruistes* (cf. figure. 4.11, page 90). Le premier génère des signaux altruistes et le second exploite les signaux altruistes générés par les autres entités robotiques.

4.3.1.7.1 Comportement d'émission de signaux altruistes Ce comportement est activé, quand les comportements de *poussée de boite* ou de *repositionnement* sont activés. Ce comportement purement altruiste consiste à émettre des signaux altruistes d'attraction ou de répulsion, paramétrés en fonction de l'angle θ que fait le robot avec la cible (cf. figure. 4.14), tels que :

1. **Si** $(|\theta| < \theta_{Max})$ **Alors** émettre un signal attractif.
   Sinon émettre un signal répulsif.
2. l'étendue (le rayon) du champ d'émission E et l'intensité I du signal émis, sont modulés en fonction de l'angle mesuré θ que le robot fait avec la cible C. Cette modulation est d'autant plus grande que :
   - $|\theta|$ tend vers 0° dans le cas d'un signal attractif, indiquant par ce fait que le robot est dans une position d'autant plus idéale pour pousser la boite,
   - $|\theta|$ tend vers 180° dans le cas d'un signal répulsif, indiquant par ce fait que le robot est positionné de plus en plus à l'opposé d'où il devrait être pour pousser convenablement la boite vers la cible.

[7] Comme l'exemple au nom barbare de trophallaxie, qui consiste pour une fourmi à alimenter un autre individu en déglutissant une partie de son propre repas.

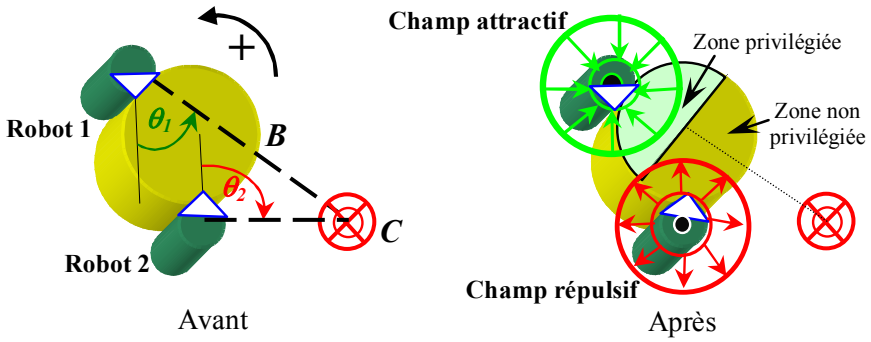


FIG. 4.14 – Comportements altruistes sous forme de signaux attractifs et répulsifs

Les caractéristiques des signaux émis par le comportement d'*émission de signaux altruistes* se résument ainsi :

$$\begin{cases} \begin{cases} E = \left(1 - \frac{|\theta|}{\theta_{Max}}\right) E_{Max} \\ I = \left(1 - \frac{|\theta|}{\theta_{Max}}\right) I_{Max} \end{cases} & \textbf{Si } |\theta| \leq \theta_{\textbf{Max}}, \text{ i.e., cas d'un signal attractif,} \\ \begin{cases} E = \left(\frac{|\theta| - \theta_{Max}}{180 - \theta_{Max}}\right) E_{Max} \\ I = -\left(\frac{|\theta| - \theta_{Max}}{180 - \theta_{Max}}\right) I_{Max} \end{cases} & \textbf{Si } |\theta| > \theta_{\textbf{Max}}, \text{ i.e., cas d'un signal répulsif.} \end{cases} \tag{4.7}$$

Avec :

- $\theta_{Max} \neq 0^{\circ}$ et à 180°,
- E_{Max}, I_{Max} des constantes définissant respectivement le rayon et l'intensité maximaux que peut avoir le signal altruiste émis par le robot. Il est à noter que les signaux attractifs et répulsifs sont discriminés ici via leur signe ("+" pour attractif et "-" pour répulsif).

Vu le caractère très élémentaire des signaux émis par le robot, nous pouvons donc affirmer, sans trop d'ambiguïté, que cette communication, peut être catégorisée dans les types de communication bas niveau (cf. §1.2.1.3.2, page 23). Ce type de communication caractérise principalement, les interactions existantes dans les sociétés d'insectes, comme c'est le cas par exemple des sociétés de fourmis, où les communications existantes entre individus, passent par le biais d'émission de phéromones, de signaux sonores par stridulations, etc. Ces signaux élémentaires modulés peuvent servir entre autres à attirer, guider, reconnaître les individus ou déclencher une alarme (Passera 1984), (Chauvin 1989).

Les principaux objectifs de l'utilisation des signaux altruistes introduits dans l'architecture de contrôle (cf. figure. 4.11), consistent à attirer plus rapidement les robots vers la zone privilégiée (zone dans laquelle le robot est susceptible de pousser convenablement

la boite vers la cible) et à l'opposé, de les éloigner rapidement de la zone non privilégiée. La figure 4.14 montre grossièrement ces deux zones[8].

Le degré de conformité des résultats attendus avec ceux de la réalité d'un système hautement dynamique sera vu en section 4.4.

4.3.1.7.2 Comportement de réponse aux signaux altruistes Ce comportement consiste à utiliser les capteurs infrarouges situés à l'avant du mini-robot ALICE (cf. figure. 4.2) (c'est-à-dire *CIR1*, *CIR2*, *CIR4*), pour attirer le robot vers les signaux attractifs de plus grande intensité I (le capteur correspondant sera mis à '1') et d'éloigner le robot des signaux les plus répulsifs (le capteur correspondant mis à '0'). La stratégie d'attraction/répulsion est résumée aussi dans le tableau 4.2.

4.4 Étude en simulation de l'architecture de contrôle

Pour évaluer d'une part la pertinence de l'architecture de contrôle proposée[9] pour le cas de la tâche coopérative de poussée d'objets, et d'autre part l'importance que peuvent avoir les comportements altruistes sur la réalisation de cette tâche, nous avons effectué un très grand nombre de simulations (plusieurs milliers) en utilisant le simulateur ***MiRoCo***[10] (cf. chapitre VII). Les résultats ainsi obtenus nous ont permis d'axer notre analyse sur une étude statistique (cf. §2.2.2.2, page 46).

4.4.1 Paramètres des simulations

Les paramètres intrinsèques aux comportements élémentaires de l'architecture de contrôle ont été fixés comme suit :

- $\theta_{Max} = 50^{\circ}$, angle conditionnant les comportements de *poussée de boite*, de *repositionnement* et d'*émission de signaux altruistes*,
- les fonctions $\boldsymbol{f}$ (caractérisant la réorientation α en degré du robot) et $\boldsymbol{g}$ (caractérisant la translation de $\boldsymbol{d}$ cm du robot) affectées au comportement de *repositionnement* sont fixées comme suit :

$$\begin{cases} \alpha = \boldsymbol{f}(\theta) = -\boldsymbol{sign}(\theta)\left(\dfrac{|\theta|}{20} + 70\right) \\ \boldsymbol{d} = \boldsymbol{g}(\theta) = \dfrac{|\theta|}{45} \end{cases}.$$

[8] Lesquelles sont discriminées, uniquement en utilisant la valeur de l'angle θ mesurée.

[9] En l'occurrence les comportements élémentaires, jumelés avec le PSAH pour coordonner les comportements entre eux.

[10] Qui nous a permis de simuler d'une manière précise la dynamique d'évolution du système multi-robots.

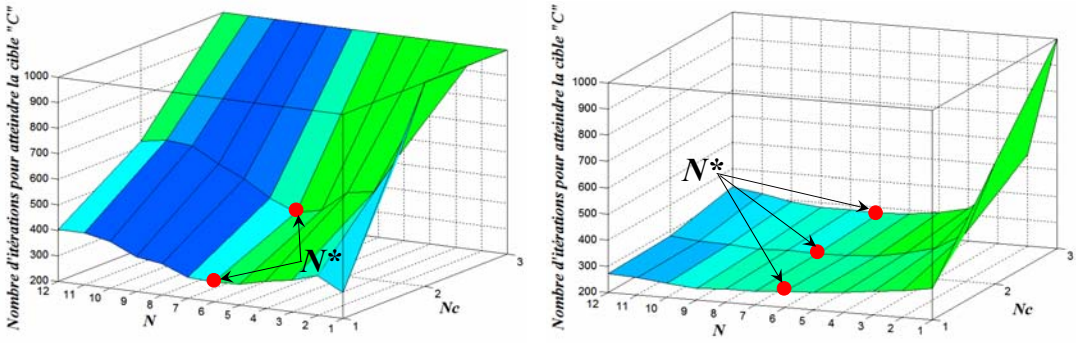
Avec ***sign***(θ) la fonction représentative du signe algébrique de θ,

- $E_{Max} = 2$ cm, champ d'émission (rayon) maximum des signaux émis par les robots sous l'impulsion du comportement d'*émission de signaux altruistes*. En sachant que les dimensions du mini-robot ALICE ne sont que de (2cm×2cm×2cm), ceci montre le caractère très localisé des signaux émis par les robots,
- $\eta = 15^{\circ}$, angle caractérisant le comportement d'*alignement* du robot avec l'objet à déplacer.

Le critère choisi pour évaluer les performances de l'architecture de contrôle proposée, est celui de l'évolution du temps (nombre d'itérations) d'exécution de la TCPO, et ceci en fonction :

- du nombre N de robots participant à la TCPO,
- du nombre minimal Nc de robots qui doivent pousser la boite en même temps,
- l'utilisation ou non des comportements altruistes.

Nous avons effectué deux séries de simulations avec des contraintes Nc allant de 1 à 3 et N allant de Nc à (Nc+11). Dans la première série de simulations (cf. figure. 4.15(a)) est testée tout d'abord l'architecture de contrôle sans les comportements altruistes et dans la seconde sont adjoints les comportements altruistes (cf. figure. 4.15(b)).



(a) Contrôle sans les comportements altruistes (b) Contrôle avec les comportements altruistes

FIG. 4.15 – Évolution du nombre d'itérations nécessaire pour réaliser la tâche coopérative de poussée d'objets en utilisant l'architecture de contrôle à base de PSAH.

Pour chaque Nc et N, nous effectuons une série de $NbSim = 60$ simulations avec des conditions initiales différentes. Nous calculons ainsi pour chaque ensemble de simulations avec Nc et N constants, le temps moyen d'exécution de la TCPO. Nous obtenons les deux figures (cf. figure. 4.15(a)) et (cf. figure. 4.15(b)) qui représentent l'évolution du temps

d'exécution de la TCPO en fonction de N et de Nc. Le temps d'exécution maximum des simulations est fixé à 1000 itérations, et une exécution d'une TCPO est considérée comme réalisée à partir du moment où la distance séparant la boite de la cible est inférieure à une distance fixée au préalable.

4.4.1.1 Conditions initiales des simulations

Les différentes conditions initiales de simulation, consistent à disposer *aléatoirement* les positions initiales des robots autour d'un cercle centré sur la boite (cf. figure. 4.16). L'objectif principal de positionner les robots autour d'un cercle, est de focaliser notre analyse sur la maîtrise de la dynamique d'interaction existante entre robots dans le milieu confiné qui entoure directement la boite, et non sur d'autres phénomènes tels que le temps nécessaire aux robots pour trouver la boite.

Le nombre important de simulations effectuées correspond à une volonté de conférer aux simulations des conditions initiales variables, et assez représentatives des différentes configurations possibles des robots autour de l'objet à pousser. Ceci permet de donner plus de crédit aux résultats statistiques, car représentatifs d'un grand nombre de mode de fonctionnement du système multi-robots.

Il est à noter que pour mener à bien la comparaison des résultats entre les deux séries de simulations (simulations avec et sans comportements altruistes), nous faisons en sorte que les conditions initiales prises pour la première série soient identiques à la seconde.

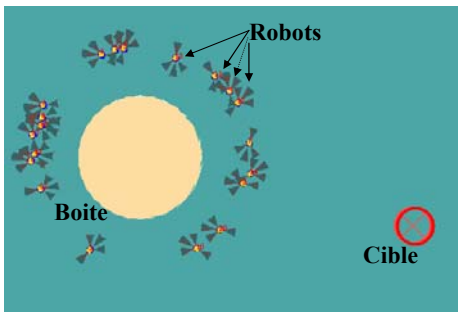


FIG. 4.16 – Positions initiales aléatoires de 20 robots autour de la boite à pousser

4.4.2 Résultats obtenus et discussions

Les simulations effectuées nous permettent d'observer un certain nombre de tendances. Parmi elles, celles communes aux deux figures 4.15(a) et 4.15(b) qui montrent que le temps nécessaire pour l'exécution de la TCPO augmente quand Nc augmente

(mais d'une manière nettement moins importante pour le cas de l'utilisation des comportements altruistes (cf. figure. 4.15(b))), et diminue généralement quand N augmente (ceci est vérifié jusqu'à atteinte d'un nombre critique N^* de robots).

Effectivement la figure 4.15(a) montre l'existence d'un nombre *optimal* N^* de robots pour pousser collectivement la boite. N^* est considéré comme optimal, car même si le nombre de robots participants à la TCPO est supérieur à cet optimum, le temps d'exécution de la TCPO ne sera que supérieur ou égal au temps d'exécution de la TCPO pour le nombre N^* de robots (économiquement parlant, N^* est optimal). L'existence de N^* s'explique par le fait que dépassant un certain nombre de robots participant à la TCPO, les robots vont plus se gêner que coopérer efficacement pour réaliser la TCPO.

Pour ce qui est de la figure 4.15(b), elle montre aussi l'existence d'un nombre optimal de robots N^* pour pousser la boite mais contrairement aux simulations sans l'intervention des comportements altruistes, l'augmentation du nombre de robots dans les simulations n'influe pas ou très peu, sur la variation du temps d'exécution de la TCPO.

La figure 4.17 montre la différence du nombre d'itérations entre les simulations effectuées avec et sans les comportements altruistes. Nous observons clairement le gain en temps acquis pour la réalisation de la TCPO avec l'intervention des comportements altruistes. Nous remarquons aussi que ce gain est d'autant plus grand que Nc et/ou N sont grands. Ceci nous montre d'une certaine manière que l'utilisation d'une communication de très bas niveau (signaux attractifs/répulsifs) permet d'une part, une meilleure coordination entre robots pour réaliser la tâche coopérative, et d'autre part permet une meilleure maîtrise des interactions entre les robots.

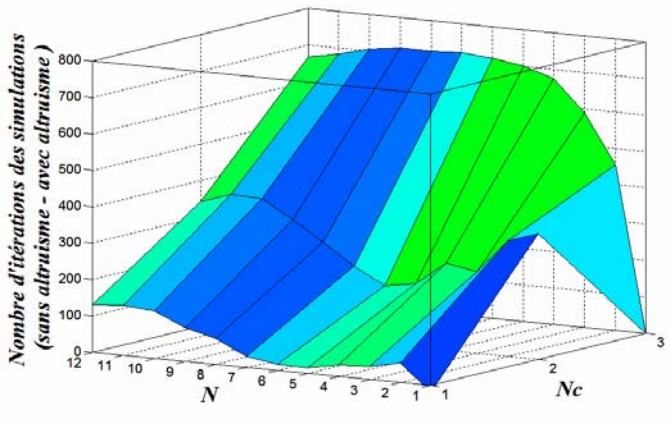


FIG. 4.17 – Gain en temps induit par les comportements altruistes

Le tableau 4.4 donne plus d'indication sur l'étendue de cette amélioration qui est exprimée en pourcentage. La référence prise en temps "100%" correspond au temps qu'il faut aux robots pour exécuter la TCPO en utilisant l'architecture de contrôle sans comportements altruistes (par exemple pour le cas où $Nc = 1$ et $N = 12$, le nombre d'itérations correspondant aux simulations sans et avec comportements altruistes, valent respectivement à 404.10 et 272.10 itérations, ce qui correspond à une amélioration du temps d'exécution égale à 32,68%). Ce tableau exprime par conséquent l'amélioration en pourcentage du temps nécessaire pour réaliser la TCPO en fonction de N et Nc. L'amélioration moyenne des simulations est de l'ordre de 41.73%.

Il est à noter qu'une modification des paramètres propres aux comportements élémentaires (détaillés dans les sections précédentes), ou la modification de paramètres plus

N \ Nc	1	2	3
Nc	-5.91	34.45	0
$Nc + 1$	15.52	45.38	45.35
$Nc + 2$	11.19	41.28	58.90
$Nc + 3$	10.13	45.17	67.54
$Nc + 4$	5.75	39.40	70.08
$Nc + 5$	5.89	38.07	71.38
$Nc + 6$	10.88	44.52	72.02
$Nc + 7$	22.12	50.03	71.75
$Nc + 8$	27.28	55.17	71.73
$Nc + 9$	33.91	58.01	70.54
$Nc + 10$	34.09	57.72	68.36
$Nc + 11$	32.68	55.00	66.72
Temps moyen par Nc	16.96	47.02	61.20
Temps moyen	41.73 %		

TAB. 4.4 – Amélioration en pourcentage du temps moyen nécessaire pour exécuter la TCPO.

intrinsèques à la TCPO (telles que la dimension et/ou la forme de la boite à pousser[11]) entraînera inévitablement une modification de la dynamique d'interaction entre robots et surtout modifiera les performances inhérentes à l'exécution de la TCPO. Cependant, la tendance globale à l'amélioration des temps d'exécution de la TCPO, quand l'architecture de contrôle utilise les comportements altruistes, reste toujours effective.

4.5 Conclusion

Nous avons proposé dans ce chapitre, le Processus de Sélection d'Action Hiérarchique "PSAH" qui correspond à un mécanisme flexible et intuitif pour coordonner d'une manière hiérarchique l'activité d'un ensemble de comportements élémentaires. Le PSAH a été testé initialement sur la tâche de navigation en présence d'obstacles et ce en utilisant des comportements continus. L'application du PSAH sur la Tâche Coopérative de Poussée d'Objets "TCPO" nous a permis par la suite d'évaluer son adéquation pour contrôler un groupe de robots mobiles d'une manière complètement distribuée et réactive. Les comportements élémentaires développés pour le cas de la TCPO sont d'un caractère très basique et ceci afin de garder l'aspect réactif de l'architecture de contrôle. La coopération entre robots induite par l'architecture de contrôle, s'est faite d'une manière complètement implicite et ceci en utilisant les comportements altruistes.

Dans le chapitre qui va suivre, d'autres mécanismes vont être intégrés au PSAH pour rendre plus flexible et précise la coordination des actions d'un groupe de comportements élémentaires.

[11] La dimension de la boite, conditionne par exemple la surface offerte par celle-ci pour permettre aux robots d'interagir directement avec elle pour la pousser. Dans le cas où cette dimension est trop petite alors ceci entraînera une augmentation considérable des interférences entre robots au voisinage immédiat de l'objet à pousser, ceci diminuera aussi les possibilités qu'ont les robots à synchroniser leur actions pour pousser.

Chapitre 5

Processus de Sélection d'Action Hiérarchique et Hybride "PSAHH"

Résumé : *Nous avons adjoint au processus de sélection d'action hiérarchique "PSAH" de base proposé au chapitre IV, un processus de* ***fusion d'actions*** *approprié, conduisant ainsi à un processus de sélection d'action hiérarchique et hybride "PSAHH". Ce nouveau mécanisme de sélection d'action permet de produire des architectures de contrôle plus flexibles et plus adaptées aux systèmes multi-robots hautement dynamiques. En plus de permettre l'organisation hiérarchique des comportements élémentaires, ce processus permet aussi de tenir compte de l'intervention de plusieurs comportements simultanément pour déterminer les actions du robot. La tâche coopérative de poussée d'objets nous servira aussi dans ce chapitre pour tester le PSAHH proposé et évaluer la pertinence de l'ajout du processus de fusion d'actions au PSAH de base.*

5.1 Processus de Sélection d'Action Hiérarchique et Hybride "PSAHH"

Dans bien des cas, une hiérarchie figée de comportements spécialisés au sein d'une architecture de contrôle comportementale, n'arrive pas à satisfaire amplement ou avec assez de précision certains objectifs liés aux tâches à exécuter. Nous proposons dans ce chapitre, d'adjoindre au processus de sélection d'action hiérarchique "PSAH" de base, un processus de fusion d'actions conduisant ainsi au Processus de Sélection d'Action Hiérarchique et Hybride "PSAHH".

L'intégration du processus de fusion d'actions au PSAH déjà existant, se fait en utilisant des blocs de fusion de commandes (cf. figure. 5.1). A chaque commande "$\mathcal{C}_{j=1..n}(Rot_j, Tra_j)$" en entrée d'un bloc de fusion Σ_i, est associée un gain respectivement $g_{ij}|_{j=1..n}$ qui représente le poids affecté aux commandes $\mathcal{C}_{j|j=1..n}$ générées par le comportement $Comportement_j|_{j=1..n}$. Les contraintes imposées sur ces gains doivent respecter l'équation 5.1.

$$\sum_{j=1..n} g_{ij} = 1 \quad \text{tel que : } g_{ij}\,|_{j=1..n} \in]0\ 1[\tag{5.1}$$

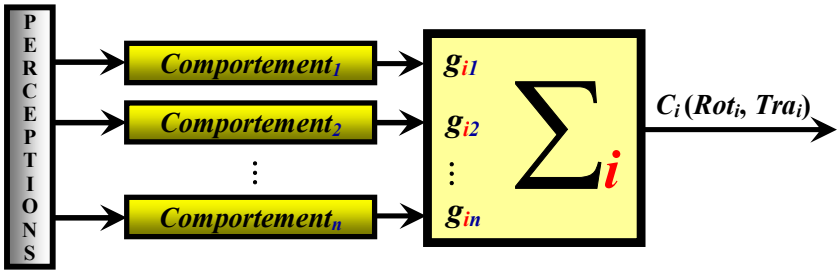


FIG. 5.1 – Bloc de fusion de commandes

La sortie du bloc de fusion Σ_i est une commande "$\mathcal{C}_i(Rot_i, Tra_i)$", qui est donnée par le processus de fusion que nous proposons (cf. algorithme. 5.1). Il est à noter que le $Comportement_1$ représentera le comportement de plus haut niveau hiérarchique par rapport au bloc de fusion Σ_i (cf. figure. 5.1).

Algorithme 5.1 Processus de sélection d'action hiérarchique et hybride "PSAHH" appliqué au bloc de fusion Σ_i.

SORTIE : La commande $\mathcal{C}_i$;

Si *Comportement*$_1$ génère une *commande refuge* $\mathcal{CR}_1 \in \mathbb{CR}_1$ **Alors**
  $\mathcal{C}_i = \mathcal{CR}_1$;
Sinon

//Mécanisme d'adaptation des gains $g_{ij}|_{j=1..n}$//
SommeGainsEliminer $= 0$;
Pour $j = 2$ à n **Faire**
  Si $\mathcal{C}_j = \mathcal{CR}_j$ **Alors**
    SommeGainsEliminer = SommeGainsEliminer + g_{ij} ;
    $g_{ij} = 0$;
  Fin Si
Fin Pour
//Affectation éventuelle des nouveaux gains
Si SommeGainsEliminer $\neq 0$ **Alors**
  Pour $j = 1$ à n **Faire**

$$g_{ij} = \left(\frac{1}{1 - \text{SommeGainsEliminer}} \right) \times g_{ij} \tag{5.2}$$

  Fin Pour
Fin Si

//Les composantes Rot_i et Tra_i de $\mathcal{C}_i$ sont obtenues comme suit :

$$Rot_i = \angle \left(\sum_{j=1..n} (\boldsymbol{g}_{ij} \cdot \vec{v}_{ij}) \right) \tag{5.3}$$

//Avec $\vec{v}_{ij}$ des vecteurs unitaires tels que : $\angle \vec{v}_{ij}$ (angle $\vec{v}_{ij}$) = "Rot_j"

$$Tra_i = \sum_{j=1..n} (\boldsymbol{g}_{ij} \cdot Tra_j) \tag{5.4}$$

Fin Si

Retourner $\mathcal{C}_i(Rot_i,\ Tra_i)$;

Le mécanisme de fusion d'actions proposé, repose donc sur une hiérarchie partielle des comportements au sein des blocs de fusion Σ_i. En effet, dans chaque bloc de fusion, un comportement est hiérarchiquement supérieur aux autres et conditionne le fait que

Σ_i génère ou pas une commande refuge "$\mathcal{CR}_1$". Cette commande refuge permettra, si elle est générée, de ne pas inhiber les comportements et/ou blocs de fusion de plus bas niveau (cf. section 5.2 pour un exemple applicatif).

L'implémentation du PSAHH décrit dans l'algorithme 5.1 correspond donc à générer des commandes $\mathcal{C}_i(Rot_i,\ Tra_i)$ obtenues via le protocole suivant :

Si le $Comportement_1$ génère une *commande refuge* $\mathcal{CR}_1 \in \mathbb{CR}_1$ **Alors** le bloc de fusion Σ_i générera à son tour une commande refuge, c'est-à-dire $\mathcal{C}_i \equiv \mathcal{CR}_1$.

Sinon le calcul des composantes de la commande $\mathcal{C}_i$ suit les étapes suivantes :

Nous vérifions tout d'abord, que les autres comportements $Comportement_i|_{i=2..n}$ (cf. figure. 5.1) ne génèrent pas une commande refuge, auquel cas cette commande n'est pas prise en compte pour le calcul de la commande finale $\mathcal{C}_i$. Ceci se fait via le *mécanisme d'adaptation des gains*. L'adaptation des gains est effective uniquement si au moins un des comportements $Comportement_i|_{i=2..n}$ génère une commande refuge, ce qui se traduit dans l'algorithme 5.1 par "SommeGainsEliminer $\neq 0$". Le but de cette réaffectation des gains est, d'une part de ne pas tenir compte des commandes refuges générées éventuellement par certains comportements, et d'autre part de redistribuer de façon uniforme les gains en tenant compte à la fois de l'équation 5.1, et des poids initiaux des commandes. Prenons un exemple : nous supposons le cas du bloc de fusion Σ_1 qui dispose de trois entrées, correspondantes aux commandes générées par les comportements $Comportement_j|_{j=1..3}$. Nous affectons aux commandes $\mathcal{C}_j|_{j=1..3}$ les gains $g_{11} = 0.700$, $g_{12} = 0.200$, $g_{13} = 0.100$ respectivement. Supposons qu'à l'instant "t" qu'il n'y ait que le comportement $Comportement_2$ qui gènére une commande refuge, alors ceci induit en utilisant le *mécanisme d'adaptation des gains*, une redistribution des gains qui donne : $g_{11} = 0.875$, $g_{12} = 0.000$, $g_{13} = 0.125$.

Les équations 5.3 et 5.4 donnent finalement les composantes "Rot_i" et "Tra_i" de la commande $\mathcal{C}_i$ qui va être générée par le bloc de fusion Σ_i.

Le mécanisme de fusion d'actions, qui fait partie du processus global de sélection d'action hiérarchique et hybride PSAHH proposé ci-dessus, s'inspire partiellement des schémas moteurs proposés par Arkin (Arkin 1989*b*) (cf. §3.3.1, page 66). Les principales analogies et différences sont résumées dans ce qui suit :

- les blocs de fusion utilisés par les schémas moteurs (cf. figure. 3.10, page 67) n'attribuent aucune forme de hiérarchie entre les comportements (schémas moteurs) qui sont à leurs entrées, ce qui n'est pas le cas pour le PSAHH qui définit une hiérarchie partielle entre comportements,
- les schémas moteurs ne prévoient aucun mécanisme pour coordonner l'activité d'un ensemble de blocs de fusion, mécanisme qui est complètement défini par le PSAHH, et ce via l'utilisation de la notion de *commande refuge* (cf. algorithme. 5.1),
- les schémas moteurs utilisent une représentation vectorielle des commandes qu'ils génèrent. Pour ce qui est de notre représentation des commandes, elle consiste en une décomposition des commandes en deux composantes distinctes "*Rot*" et "*Tra*" qui donnent à notre sens une vision plus intuitive des actions à réalisés par le robot (cf. §4.1.2, page 79). Cette représentation peut se traduire par exemple directement par une commande de rotation pure (sans translation) du robot $\mathcal{C}(Angle^{\circ}, 0 \text{ cm})$, alors qu'en représentation purement vectorielle, celle-ci peut s'avérer ambiguë, vu le module nul du vecteur représentatif de la commande[1],
- comme analogie affirmée entre les deux processus de fusion d'actions, nous notons l'attribution d'un gain g_i propre aux commandes de chaque $Comportement_i$. Cependant les gains attribués pour le cas de la fusion d'actions dans le PSAHH sont d'une part contraints par l'équation 5.1, et d'autre part ils peuvent être adaptés en ligne via le *mécanisme d'adaptation des gains* proposé,
- l'obtention de la commande finale des blocs de fusion utilisés avec les schémas moteurs passe :
  - initialement par la multiplication des gains $g_i|_{i=1..n}$ par les vecteurs de commande $\mathcal{C}_i|_{i=1..n}$ générés par les comportements élémentaires (les schémas moteurs), et ceci pour produire une modulation des vecteurs de commande initiaux,
  - après cela, une sommation vectorielle des vecteurs de commande modulés est faite, puis une normalisation est réalisée pour aboutir à la commande finale du bloc de fusion.

  Pour ce qui est de l'obtention des commandes finales des blocs de fusion adjoints au PSAHH, elle se fait via le protocole mentionné dans l'algorithme 5.1. C'est-à-dire en cas de fusion d'actions effective, elle se fait en utilisant les équations 5.3 et 5.4,

[1] Il est à noter, qu'une commande de rotation pure, peut être représentée vectoriellement, mais à condition, soit de représenter le module du vecteur comme étant égal à ε tel que $\varepsilon \to 0$, ou soit d'appliquer un post-traitement sur le vecteur de commande afin qu'il soit exécutable par les actionneurs.

qui utilisent respectivement une *sommation vectorielle* et une *sommation scalaire* pour obtenir les deux composantes de la commande finale.
Il est à noter que dans le cas du PSAHH, nous n'effectuons aucune normalisation des commandes générées par les blocs de fusion, car la validité des commandes générées par les blocs de fusion est déjà garantie par les contraintes pré-imposées sur les gains $g_i|_{i=1..n}$ (cf. équation. 5.1).

5.2 Le PSAHH appliqué à la tâche coopérative de poussée d'objets

L'application du PSAHH au cas de la TCPO (cf. figure. 5.2) s'est faite tout naturellement par rapport à l'architecture de contrôle antérieure basée uniquement sur le PSAH (cf. figure. 4.11, page 90). En effet, ceci a été facilité par :

- l'utilisation des mêmes comportements élémentaires proposés au chapitre IV (cf. §4.3.1, page 91),
- la conservation dans l'ensemble, de la même hiérarchie des comportements excepté pour le comportement d'*attraction à la boite* qui devient hiérarchiquement supérieur aux comportements d'*évitement d'obstacles* et de *réponse aux signaux altruistes*[2].

L'utilisation des blocs de fusion $\Sigma_i|_{i=1..3}$ dans l'architecture de contrôle (cf. figure. 5.2), permet une contribution effective des comportements d'*évitement d'obstacles* et de *réponse aux signaux altruistes* au niveau de toutes les phases distinctives de la TCPO, en l'occurrence celle de l'*attraction à la boite*, du *repositionnement* et celle finalement de la *poussée de boite*. Le fonctionnement des trois blocs de fusions $\Sigma_1, \Sigma_2, \Sigma_3$ suit le PSAHH détaillé dans l'algorithme 5.1. Il est à noter cependant que le comportement de plus grand ordre hiérarchique affecté à chaque bloc de fusion $\Sigma_i|_{i=1..3}$ est respectivement, le comportement de *poussée de boite*, celui du *repositionnement*, et celui de l'*attraction à la boite*.

5.2.1 Intérêt de l'utilisation des blocs de fusion

L'implémentation de l'architecture de contrôle pour le cas de la TCPO avec une coordination entre comportements basée uniquement sur le PSAH (Adouane & LeFort-Piat

[2]En revanche, les commandes générées par les comportements d'*évitement d'obstacles* ou de *réponse aux signaux altruistes*, pourront être affectées d'un plus grand poids au sein du bloc de fusion Σ_3 (cf. figure. 5.2).

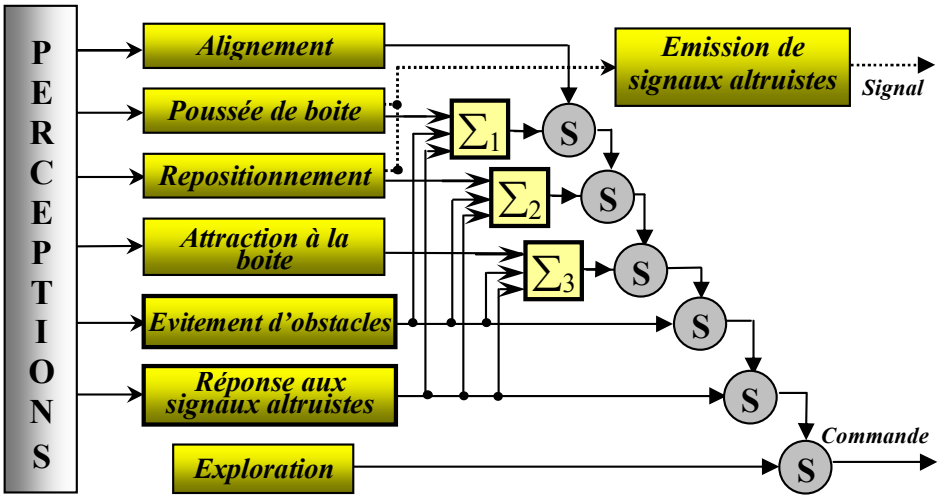


FIG. 5.2 – Architecture de contrôle intégrant le PSAHH au cas de la TCPO

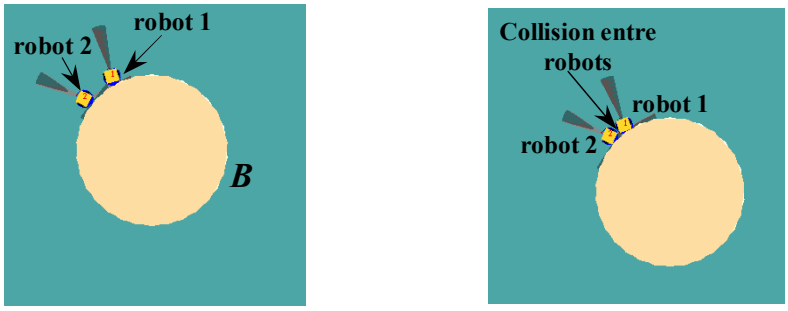
2004*a*) nous a amené à observer au cours des simulations, l'existence d'effets microscopiques[3] indésirables. Ram par exemple dans (Ram et al. 1994) se base sur des considérations visuelles (forme des trajectoires prises par le robot pour atteindre l'objectif) pour donner une note sur la qualité de l'apprentissage paramétrique effectué. Il faut néanmoins faire attention, à la causalité existante entre les situations microscopiques et celles macroscopiques, qui sont généralement le résultat d'un ensemble de micro-situations. Une situation considérée au niveau macroscopique comme une situation de collaboration, peut parfaitement résulter au niveau microscopique de situations locales de compétition et/ou d'encombrement (cas de la TCPO). Inversement, des situations peuvent être considérées au niveau local comme des situations de coopération (ni compétition, ni encombrement), alors que celles-ci peuvent être indésirables pour la réalisation de tâches coopératives nécessitant des interactions très étroites entre les robots.

Les effets microscopiques indésirables observés, sont directement liés au grand nombre de robots en interaction aux alentours immédiats de la boite à pousser. Ainsi pour éviter ces effets, nous avons introduit les blocs de fusion d'actions $\Sigma_1, \Sigma_2, \Sigma_3$ avec les motivations suivantes :

1. Pour le cas du bloc de fusion "Σ_1", nous avons observé que lorsque les robots poussent la boite en même temps, leurs trajectoires vont inévitablement se croi-

[3]Par effets microscopiques, nous sous-entendons, toutes les interactions locales entre robots, qui mènent le système multi-robots vers des configurations (observables et/ou quantifiables) qu'on peut juger comme favorables ou non à l'évolution des tâches en cours.

ser (cf. figure. 5.3), ce qui entraîne des collisions inopinées des robots. Cet effet se manifeste du fait que les robots tendent toujours à pousser la boite en direction de son centre (intervention du comportement d'*alignement*). Pour éviter ces collisions entre robots, l'idée est de garder actif le comportement d'*évitement d'obstacles* quand celui de *poussée de boite* est actif. La contribution du comportement d'*évitement d'obstacles* via le gain g_{12} qui lui est attribué, permet donc de maintenir toujours une distance minimale (fonction du gain g_{12}) entre les robots quand ils sont en train de pousser la boite. La contribution du comportement de *réponse aux signaux altruistes*[4] consiste à faire en sorte que le robot qui est en train de pousser la boite, soit en même temps attiré vers la zone privilégiée (cf. figure. 4.14, page 99) pour pousser la boite. Cette zone correspond aux positions sur la surface de la boite, qui permettent au robot d'observer la boite et la cible avec des angles qui tendent vers zéro (cf. figure. 4.13, page 97). En effet, les robots qui sont plus près de ces points, émettent des signaux attractifs avec une intensité plus grande que ceux éloignés par rapport à ces points (cf. §4.3.1.7.1, page 98).



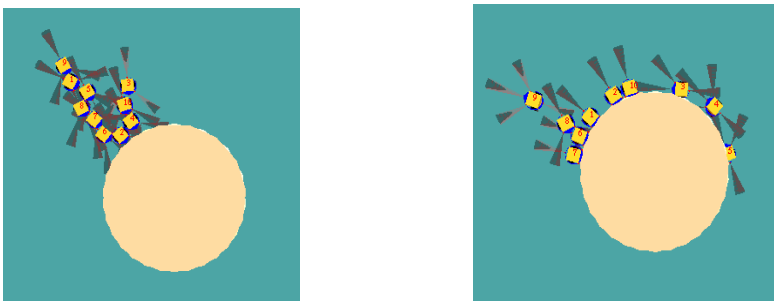
(a) Temps=T1 (b) Temps=T1+△T

FIG. 5.3 – Effet observé lorsque le comportement d'*évitement d'obstacles* n'est pas activé quand les robots poussent la boite.

2. Le bloc de fusion "Σ_2", fait en sorte que le *repositionnement* du robot tend plus rapidement vers la zone privilégiée (contribution des comportements altruistes) et ce en évitant les autres robots présents aux alentours immédiats de l'objet à pousser (contribution du comportement d'*évitement d'obstacles*).
3. Le bloc de fusion "Σ_3" est lié directement au comportement d'*attraction à la boite*. La principale contribution du comportement d'*évitement d'obstacles*, est d'empêcher que les robots soient les uns derrière les autres (cf. figure. 5.4(a))[5]. En effet, ces

[4] Cette contribution est effective dans le cas bien évidemment où le robot détecte des signaux altruistes.
[5] Ceci ce produit quand l'objet à pousser commence à bouger.

configurations empêchent un grand nombre de robots d'être en contact direct avec l'objet à pousser, et donc de contribuer à la réalisation de la TCPO. La contribution du comportement altruiste est toujours liée au fait d'attirer plus rapidement les robots vers la zone privilégiée.



(a) Avec un gain très petit (b) Avec un gain approprié

FIG. 5.4 – Influence de la valeur du gain "g_{32}" attribuée au comportement d'*évitement d'obstacles* pour le cas du bloc de fusion Σ_3.

L'architecture de contrôle à base de PSAHH proposée pour le cas de la TCPO, consiste donc à obtenir via les blocs de fusion $\Sigma_i|_{i=1..3}$ un équilibre entre :

- des comportements d'isolement (en l'occurrence celui de l'évitement d'obstacles et celui de l'éloignement par rapport aux signaux répulsifs) qui diminuent le risque d'interférences entre robots, qui ralentissent les déplacements des robots et par conséquent conduisent à l'augmentation du temps nécessaire pour l'exécution de la tâche,
- des comportements grégaires (en l'occurrence celui de l'attraction vers la boite et celui de l'attraction vers les signaux attractifs) qui structurent le groupe de robots.

5.2.2 Étude en simulation de l'architecture de contrôle

L'estimation des performances liées à l'application du PSAHH au cas de la TCPO, se fait ici comme au chapitre IV, par le biais d'une étude statistique sur un grand nombre de simulations. La critère retenu pour évaluer ces performances correspond aussi ici au temps nécessaire pour réaliser la TCPO.

Les paramètres de l'architecture de contrôle à base de PSAHH sont les suivants :

- les paramètres intrinsèques aux comportements élémentaires conservent les mêmes valeurs que pour l'étude statistique précédente, c'est-à-dire : $\theta_{Max} = 50^\circ$, $E_{Max} = 2$ cm, η=15°, $\alpha = \boldsymbol{f}(\theta) = -\boldsymbol{sign}(\theta)\,(|\theta|/20 + 70)$, $\boldsymbol{d} = \boldsymbol{g}(\theta) = |\theta|/45$,

– les gains $g_{ij}|_{i=1..3,j=1..3}$ adoptés pour les blocs de fusion Σ_1, Σ_2 et Σ_3, ont été obtenus de manière empirique de façon à éviter certaines observations visuelles mentionnées dans la section 5.2.1. Nous obtenons alors les gains suivants :
  – Σ_1 : g_{11}= 0.50, g_{12}= 0.40, g_{13}= 0.10
  – Σ_2 : g_{21}= 0.50, g_{22}= 0.45, g_{23}= 0.05
  – Σ_3 : g_{31}= 0.20, g_{32}= 0.16, g_{33}= 0.64
  Avec : g_{11}, g_{21}, g_{31} les gains des commandes affectés respectivement aux comportements de *poussée de boite*, *repositionnement* et d'*attraction à la boite* ; g_{12}, g_{22}, g_{32} correspondent au comportement d'*évitement d'obstacles* et g_{13}, g_{23}, g_{33} correspondent au comportement de *réponse aux signaux altruistes*.

  Il est à noter qu'au chapitre VI sont proposées, d'une part une méthodologie appropriée pour quantifier la pertinence des actions engendrées par les robots exécutant la TCPO, et d'autre part une optimisation paramétrique en utilisant les algorithmes génétiques, et ce afin de permettre l'obtention automatique des gains des blocs de fusion $\Sigma_i|_{i=1..3}$.

Les simulations présentées dans ce qui suit, portent principalement sur :

1. le test de l'importance des comportements altruistes sur l'architecture de contrôle basée sur le PSAHH et appliquée au cas de la TCPO,
2. la comparaison qualitative de l'efficacité d'une architecture de contrôle à base du PSAHH par rapport à celle à base du PSAH, appliquées toutes deux au cas de la TCPO,
3. le test de l'adéquation de l'architecture de contrôle basée sur le PSAHH, pour le cas de la coopération d'un grand nombre de robots (plusieurs dizaines) pour réaliser la TCPO.

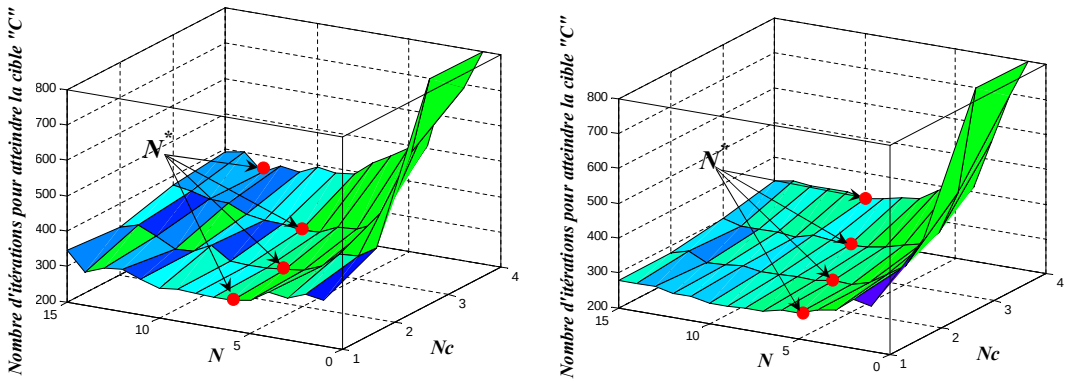
5.2.2.1 Utilisation ou non des comportements altruistes

Nous procédons exactement avec le même protocole expérimental utilisé au chapitre IV (cf. §4.4.1, page 100), qui a permis de tester l'importance des comportements altruistes pour l'exécution de la TCPO[6].

La figure 5.5(a), représente les simulations effectuées sans les comportements altruistes et la figure 5.5(b) celles avec les comportements altruistes. Chacune des deux

[6]L'application du PSAHH pour les deux séries de simulations s'effectue sans le *mécanisme d'adaptation des gains* (cf. algorithme. 5.1).

figures représente l'évolution du temps moyen (nombre d'itérations) de l'exécution de la TCPO en fonction du nombre de robots N et de la contrainte Nc (avec $Nc = 1..4$ et N allant de Nc jusqu'à (Nc+14)). Chaque point des figures 5.5(a) et 5.5(b) représente le nombre d'itérations moyen de $NbSim = 50$ simulations avec Nc et N constants et des positions initiales des robots différentes (cf. §4.4.1.1, page 102). Le nombre maximum d'itérations pour chaque simulation est fixé à 800.



(a) Contrôle sans les comportements altruistes (b) Contrôle avec les comportements altruistes

FIG. 5.5 – Évolution du nombre d'itérations nécessaire pour réaliser la TCPO en utilisant l'architecture de contrôle à base de PSAHH.

Les figures 5.5(a) et 5.5(b), montrent que le temps nécessaire pour l'exécution de la TCPO diminue quand le nombre de robots N augmente, cependant ce temps augmente quand Nc augmente. On note aussi l'existence d'un nombre optimal N^* de robots pour réaliser la TCPO, et ceci pour chaque Nc. Au dessus de N^*, on remarque pas ou peu d'amélioration par rapport au temps d'exécution de la TCPO. On note aussi que la figure 5.5(b) est plus lisse que la figure 5.5(a), ce qui montre d'une certaine manière, que l'utilisation d'une communication de très bas niveau (signaux attractifs/répulsifs) permet une meilleure coordination et une meilleure maîtrise des interactions entre robots.

La figure 5.6 représente la différence du nombre d'itérations entre les simulations effectuées sans les comportements altruistes, et ceux avec comportements altruistes, pour l'exécution de la TCPO. Nous remarquons d'une manière générale, une diminution du temps d'exécution quand l'architecture de contrôle utilise les comportements altruistes. Le tableau 5.1 donne une meilleure indication sur l'étendue de cette amélioration, qui est exprimée ici en pourcentage (la référence en temps 100% est le temps qu'il faut pour exécuter la TCPO avec l'architecture de contrôle sans les comportements altruistes). Ce tableau exprime par conséquent l'amélioration en pourcentage du temps nécessaire pour

réaliser la TCPO en fonction de N et Nc. L'amélioration moyenne des simulations est de l'ordre de 12.24%.

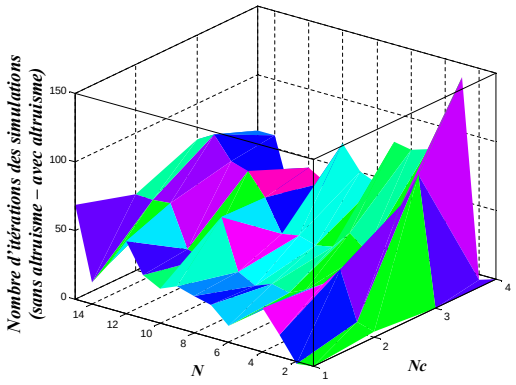


FIG. 5.6 – Gain en temps acquis en utilisant les comportements altruistes pour l'exécution de la TCPO.

La figure 5.7 donne la tendance de l'amélioration du temps nécessaire pour exécuter la TCPO en fonction de Nc. On remarque que cette amélioration est d'autant plus importante que la tâche nécessite la coopération effective de plusieurs robots qui poussent la boite en même temps.

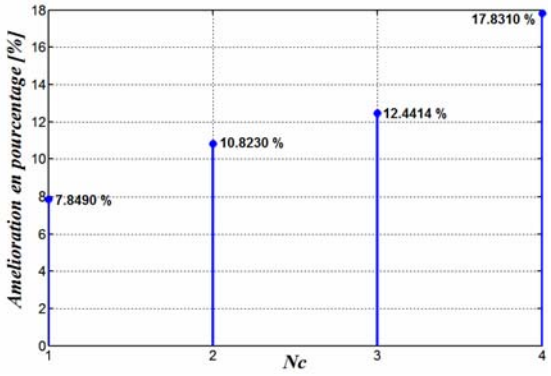


FIG. 5.7 – Amélioration du temps d'exécution de la TCPO en fonction de Nc

5.2.2.2 Comparaison qualitative entre le PSAH et le PSAHH

Nous allons vérifier dans ce qui suit, si les mécanismes complémentaires adjoints aux mécanismes de base du PSAH conduisent globalement aux améliorations attendues. A cette fin, nous effectuons une comparaison qualitative entre les deux processus de

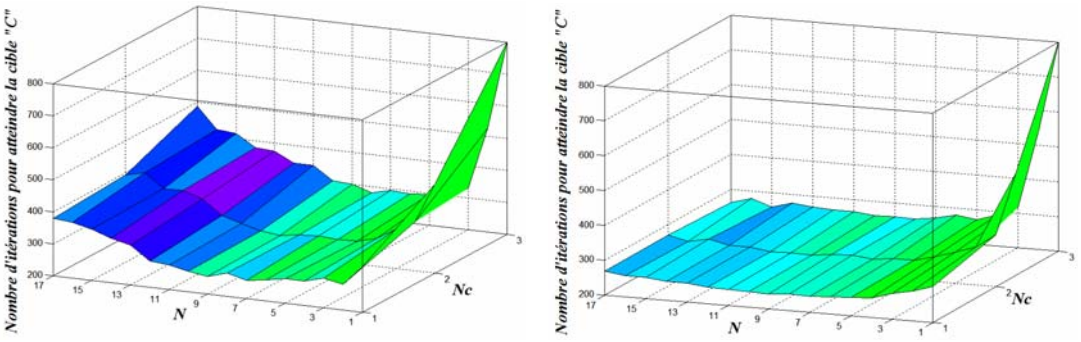
N \ Nc	1	2	3	4
Nc	0.00	1.56	0.00	0.00
$Nc + 1$	1.64	5.87	32.40	7.58
$Nc + 2$	9.39	19.41	31.71	39.52
$Nc + 3$	16.97	11.99	11.97	38.60
$Nc + 4$	12.88	12.08	5.65	26.97
$Nc + 5$	6.65	8.95	17.36	25.48
$Nc + 6$	8.42	7.25	4.59	25.34
$Nc + 7$	5.92	5.78	8.73	14.95
$Nc + 8$	11.58	9.64	1.82	15.68
$Nc + 9$	7.87	14.95	4.21	22.51
$Nc + 10$	11.85	7.39	8.98	3.16
$Nc + 11$	10.80	11.91	11.32	14.67
$Nc + 12$	2.54	11.63	18.64	5.34
$Nc + 13$	0.33	13.77	15.21	5.88
$Nc + 14$	10.89	20.16	14.02	21.79
Temps moyen par Nc	7.85	10.82	12.44	17.83
Temps moyen	12.24%			

TAB. 5.1 – Amélioration en pourcentage du temps moyen nécessaire pour réaliser la TCPO, quand l'architecture de contrôle à base de PSAHH utilise les comportements altruistes.

sélection d'action proposés. La comparaison se fonde sur les performances atteintes pour réaliser la TCPO.

Nous effectuons pour cela, deux séries de simulations avec les paramètres suivants : $Nc = 1..3$, N allant de Nc à Nc+16, et $NbSim = 25$. Les figures 5.8(a) et 5.8(b) donnent l'allure de l'évolution du nombre d'itérations nécessaires pour réaliser la TCPO en fonction de N et de Nc, et ceci respectivement pour le PSAH et le PSAHH.

La figure 5.9, montre la différence du nombre d'itérations entre les simulations effectuées avec le PSAH, et celles avec le PSAHH. Nous observons clairement le gain en temps acquis pour la réalisation de la TCPO en utilisant le PSAHH. On remarque aussi que ce gain est d'autant plus grand que Nc et/ou N sont grands. Le tableau 5.2 donne



(a) Simulations de l'architecture de contrôle avec le PSAH

(b) Simulations de l'architecture de contrôle avec le PSAHH

FIG. 5.8 – Évolution du nombre d'itérations pour réaliser la TCPO en fonction de N et de Nc.

une meilleure indication sur l'étendue de cette amélioration exprimée ici en pourcentage. L'amélioration moyenne des simulations est de l'ordre de 15.42%.

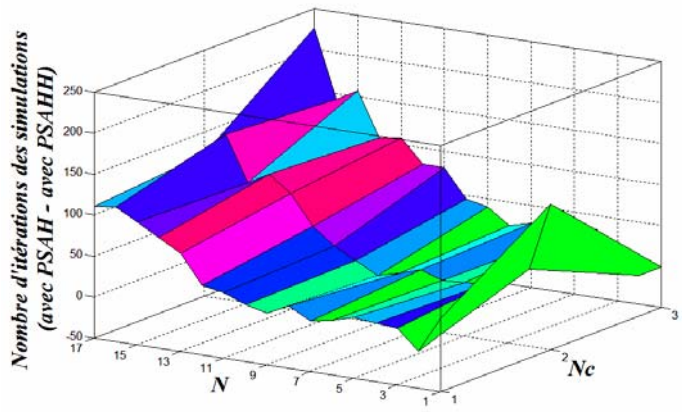


FIG. 5.9 – Gain en temps acquis lors de l'utilisation de l'architecture de contrôle à base de PSAHH.

5.2.2.3 Utilisation du PSAHH pour contrôler un grand nombre de robots

Afin de constater l'adéquation du PSAHH au cas de la coopération d'un grand nombre de robots (plusieurs dizaines), nous l'avons appliqué au cas de la TCPO qui voit la densité des robots autour de l'objet à pousser, augmenter jusqu'à atteindre 40 robots (cf. figure. 5.10).

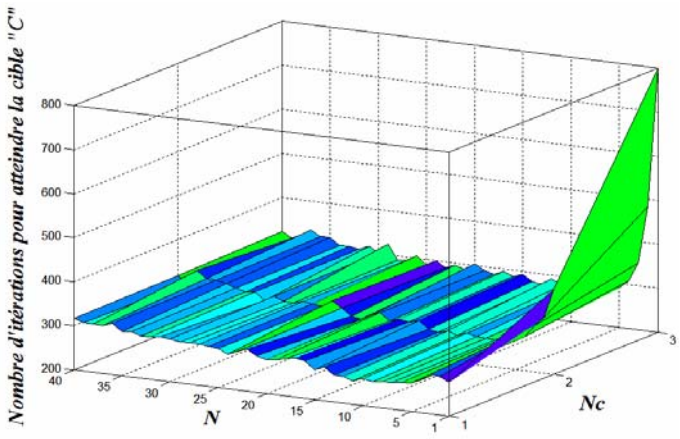


FIG. 5.10 – PSAHH appliqué à la TCPO pour le cas d'une coopération d'un grand nombre de robots.

Nous avons pour cela effectué un grand nombre de simulations, en faisant varier $Nc = 1..3$, N allant de Nc à $Nc+39$, et $NbSim = 25$. Nous constatons que le temps nécessaire pour réaliser la TCPO se stabilise autour d'une valeur fixe de temps, et cela même pour un grand nombre de robots intervenant sur l'objet à pousser. Ceci démontre l'adéquation de l'architecture de contrôle proposé pour une distribution du contrôle sur un grand nombre de robots. Les robots arrivent, par conséquent, à synchroniser leurs actions, éviter les situations conflictuelles, et ceci d'une manière complètement réactive.

5.3 Conclusion

Nous avons proposé dans ce chapitre, le Processus de Sélection d'Action Hiérarchique et Hybride "PSAHH" qui adjoint au PSAH de base (chapitre IV) des mécanismes de fusion d'actions spécifiques. Le PSAHH ainsi obtenu permet de coordonner l'activité d'un groupe de comportements élémentaires d'une manière encore plus flexible et intuitive que ce qui a été atteint avec le PSAH.

Effectivement, le PSAHH s'avère être particulièrement adapté pour une utilisation sur des architectures de contrôle comportementales réactives, dédiées au contrôle de systèmes multi-robots avec des dynamiques très importantes.

Cependant, l'élément principal qui peut rendre l'utilisation du PSAHH laborieuse, concerne la détermination des gains spécifiques des blocs de fusion (pour le cas de la TCPO, cela correspond à la détermination des gains propres aux blocs de fusion $\Sigma_1, \Sigma_2, \Sigma_3$). Pour cette raison, nous proposons au chapitre VI, une méthode automatique

de détermination de ces paramètres de gain et ce via une optimisation paramétrique par algorithme génétique.

N \ Nc	1	2	3
Nc	11.48	26.85	0.00
$Nc + 1$	-1.84	12.51	-2.85
$Nc + 2$	6.42	10.59	4.27
$Nc + 3$	7.71	5.99	10.33
$Nc + 4$	8.64	5.63	8.47
$Nc + 5$	6.23	4.97	4.54
$Nc + 6$	4.43	8.49	9.48
$Nc + 7$	9.41	5.25	7.70
$Nc + 8$	4.89	2.82	13.44
$Nc + 9$	7.38	8.15	14.96
$Nc + 10$	10.79	12.30	23.06
$Nc + 11$	12.31	18.37	23.26
$Nc + 12$	21.41	26.48	28.69
$Nc + 13$	25.09	30.23	27.51
$Nc + 14$	27.81	26.42	38.00
$Nc + 15$	30.42	37.19	33.86
$Nc + 16$	29.14	31.34	46.23
Temps moyen par Nc	13.04	16.09	17.12
Temps moyen	15.42 %		

TAB. 5.2 – Amélioration en pourcentage du temps moyen nécessaire pour réaliser la TCPO, et ce entre l'architecture de contrôle à base de PSAH et celle à base de PSAHH.

Chapitre 6

Optimisation paramétrique de l'architecture de contrôle

Résumé : *Après un bref survol d'un certain nombre de techniques d'apprentissage dites bio-inspirées, nous allons présenter dans ce chapitre une méthodologie d'apprentissage des paramètres intrinsèques au fonctionnement du PSAHH. Plus spécifiquement, nous allons optimiser les valeurs des gains des blocs de fusion de l'architecture de contrôle proposée au chapitre précédent. L'optimisation paramétrique est obtenue en utilisant une méthode évolutionniste à base d'algorithmes génétiques. Des opérateurs génétiques appropriés sont utilisés pour manipuler des chromosomes à valeurs réelles soumis à des contraintes émanant de l'architecture de contrôle.*

6.1 La bio-inspiration comme moyen d'optimiser les architectures de contrôle

L'objectif commun de toute technique d'optimisation est de trouver la solution du problème, qui a tendance à maximiser (ou à minimiser) un ou plusieurs critères caractérisant le problème posé. Dans ce qui suit sont présentées, de façon synthétique, différentes techniques qui ont comme points communs, le fait d'une part, d'être fortement inspirées de la nature au niveau de leurs mécanismes de fonctionnement et d'autre part, d'être appliquées au cas du contrôle de robots mobiles.

Nous présentons plus spécifiquement, les réseaux de neurones, l'apprentissage par renforcement et les algorithmes génétiques[1], qui sont largement utilisés dans la littérature pour l'obtention des paramètres optimaux ou sous-optimaux d'architectures de contrôle. Ces mêmes paramètres vont permettre au robot autonome de satisfaire avec un certain niveau d'exigence la réalisation d'une tâche précise. Effectivement, dans le cadre de l'optimisation des architectures de contrôle dédiées à l'exécution de tâches complexes, il est très difficile de parler d'un contrôleur optimal absolu, car cette affirmation exige de disposer au préalable d'un modèle exact de l'environnement d'évolution du robot ainsi que de la tâche complexe qu'il doit effectuer. Ces modèles ne sont pas évidents à obtenir surtout pour le cas des systèmes multi-robots à grande dynamique d'évolution. Les tâches complexes à exécuter sont aussi dans la plupart des cas, caractérisées par une multitude d'objectifs à satisfaire simultanément, ce qui rend cette notion d'optimalité du contrôle plus difficile à définir. Pirjanian dans (Pirjanian 1999) utilise la théorie de la prise de décision en tenant compte de plusieurs objectifs à la fois "*multiple objective decision making theory*" (Chankong & Haimes 1983), et ceci dans le but de gérer la fusion de comportements dans un contrôleur flou. Pirjanian parle alors de commande qu'il nomme "*good enough*", c'est-à-dire une commande qui va satisfaire avec un certain niveau d'exigence tel ou tel critère. En effet, nous pouvons aisément imaginer qu'une même commande, puisse être optimale pour le cas d'une configuration particulière du système robotique (e.g., une configuration précise des stimuli perçus), et s'avérer complètement inadaptée pour la même configuration de stimuli, mais pour une dynamique d'évolution différente (e.g., vitesses différentes des autres agents présents dans l'environnement, formes différentes des autres agents (obstacles, robots)).

Recourir donc aux techniques d'optimisation bio-inspirées n'est pas fortuit, même si elles sont loin d'être toujours optimales. Elles sont en revanche les plus génériques et se

[1] Les mécanismes de fonctionnement des algorithmes génétiques seront détaillés plus spécifiquement, car ils nous serviront de base introductive à l'optimisation paramétrique que nous proposons.

déclinent en une multitude de variantes, dont l'une pourra être plus adaptée que les autres pour un problème donné. La généricité de l'application de ces méthodes, leur permet aussi d'être appropriées pour le cas de systèmes hautement dynamiques et complexes sur lesquels on ne dispose pas de modèle exact de fonctionnement (Landau et al. 2002). Pour toutes ces raisons, l'apprentissage par évolution artificielle, est depuis quelques années, de plus en plus utilisé dans le cadre de la robotique ou plus particulièrement pour le cas des systèmes multi-robots.

6.1.1 Les réseaux de neurones artificiels

Les Réseaux de Neurones Artificiels "RNA" (Freeman & Skapura 1992) s'inspirent des capacités d'auto-organisation et d'adaptation des réseaux de neurones biologiques afin d'organiser ou contrôler des systèmes artificiels (e.g., robots, systèmes de diagnostic ou de classification, etc.). Les RNA à l'image des réseaux de neurones biologiques, sont des réseaux fortement connectés de neurones[2] (processeurs élémentaires) fonctionnant en parallèle. A chaque connexion, est associé un poids qui détermine l'influence réciproque des cellules connectées. Les poids des connexions sont modifiables et c'est cette plasticité qui donne lieu aux facultés d'adaptation et d'apprentissage[3] du RNA.

Les premiers travaux sur les RNA en tant qu'entités de traitement de l'information reviennent à Mc Culloch dès les années 40 (Culloch & Pitts 1943). Il a montré que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes (tout au moins au niveau théorique). Après cela une multitude de travaux sont apparus dans la littérature. Parmi les plus importants nous pouvons citer ceux de Rosenblatt sur le modèle du *perceptron*. En 1969, Minsky publia un ouvrage (Minsky & Papert 1969) dans lequel il met en exergue les limitations théoriques du *perceptron* et implicitement celles liées à tout modèle de RNA. Avec l'apparition de cet ouvrage, la communauté scientifique s'est peu à peu désintéressée des RNA. L'engouement des scientifiques pour les RNA n'est revenu que vers la fin des années 80 avec l'apparition des RNA multicouches et des algorithmes de rétropropagation du gradient (Rumelhart et al. 1986). Les RNA couramment utilisés actuellement sont généralement décomposés en trois types de couches, en l'occurrence, une couche d'entrée, une (ou plusieurs) couche(s) cachée(s), et finalement une couche de sortie.

[2]Unité de traitement de l'information dans le cerveau. e.g., ils sont entre 10^{10} à 10^{11} neurones au niveau du cerveau humain.

[3]L'apprentissage dans le contexte des RNA est une phase du développement durant laquelle le comportement du réseau est modifié, via une phase d'adaptation des poids des interconnexions neuronales (e.g. en utilisant les algorithmes de rétropropagation de l'erreur) et ceci jusqu'à l'obtention du comportement désiré.

Dans le cadre du contrôle, domaine d'utilisation des RNA qui nous intéresse plus particulièrement, les RNA ont en commun au minimum deux types caractéristiques de neurones. Les premiers neurones dits sensoriels, reçoivent les données directement des capteurs, alors que les seconds, dits moteurs, émettent des commandes vers les parties actives du robot, tels les moteurs des roues. Le système de commande d'un robot de type "char" avec quatre capteurs de proximité prendra la forme alors d'un réseau de neurones minimal, composé de quatre neurones sensoriels et de deux neurones moteurs. Chacun des neurones moteurs est relié à l'un des moteurs qui actionne l'une des roues du robot. Plus l'activité de ce neurone est élevée, plus la roue va tourner vite. De même, chaque neurone sensoriel verra son niveau d'activation plus élevé si celui-ci détecte un obstacle de plus près.

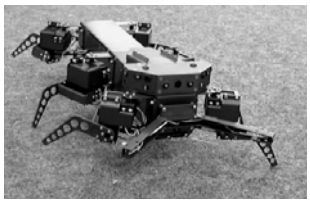
Les applications montrant les potentialités d'apprentissage des RNA sont nombreuses. L'exemple de la marche de créatures artificielles à pattes est un bon challenge d'application pour ce genre de contrôle. Les ingénieurs de la société "Sony" ont fait évoluer le système de commande de la locomotion du robot-chien "Aibo" (cf. figure. 6.1(a)) grâce à un RNA approprié. Ils ont pu obtenir ainsi, des marches au pas et au trot de façon automatique, plus performantes que des programmes préconçus et optimisés théoriquement pour Aibo. Le critère d'apprentissage consistait alors à obtenir une marche qui fasse avancer le plus loin possible le robot en un temps déterminé. L'optimisation du contrôleur de la marche est, dans ce cas, plus performante car elle intègre d'une manière naturelle les spécificités matérielles et les contraintes structurelles non modélisées par le modèle établi d'Aibo (Hornby et al. 1999). Un autre exemple d'apprentissage de la marche d'une entité artificielle, concerne le robot hexapode de l'équipe d'AnimatLab du LIP6[4] (cf. figure. 6.1(b)). Le RNA proposé prend la forme d'une hiérarchie de réseaux de neurones susceptibles de contrôler un ou plusieurs comportements exhibés par le robot hexapode. Des contrôleurs de marche tripode ont ainsi été obtenus, ainsi que des contrôleurs de comportements plus élaborés comme l'évitement d'obstacle et le suivi d'un gradient de lumière (Filliat et al. 1999).

6.1.2 L'apprentissage par renforcement

L'apprentissage par renforcement "AR" est l'une des approches qui s'inspire aussi de la nature pour permettre l'optimisation et la synthèse de contrôleurs appropriés pour des robots mobiles. Les algorithmes à base d'apprentissage par renforcement, tels qu'ils ont été introduits vers la fin des années 80 (Sutton 1988), utilisent en majorité deux éléments de base :

[4] `http://animatlab.lip6.fr/`

(a) Robot Aibo de Sony.



(b) Robot hexapode du LIP6.

FIG. 6.1 – Exemples de robots à pattes apprenant à marcher en utilisant un RNA

- le principe d'*essais-erreurs* qui caractérise la psychologie d'apprentissage des êtres vivants dans leur environnement[5],
- les outils liés au *contrôle optimal*, plus spécifiquement ceux liés à la *programmation dynamique* introduite par Richard Bellman (Bellman et al. 1959). Cette formulation du problème permet d'affirmer la convergence de l'optimisation si le système est Markovien.

Les principes généraux du déroulement d'un algorithme d'apprentissage par renforcement, se déclinent selon les trois étapes suivantes :

1. à chaque instant t, le robot est dans un état $s_t \in S$ (tel que S correspond à l'ensemble des états possibles du robot),
2. en fonction de l'état courant s_t, le robot sélectionne l'action à produire $a(t) \in A(s_t)$ (tel que $A(s_t)$ est l'ensemble des actions possibles à partir de l'état s_t),
3. un incrément de temps après, le robot se retrouve dans l'état s_{t+1}, et il peut alors recevoir lors de la transition de s_t à s_{t+1}, une récompense ou une punition numérique $r_{t+1} \in R$, (tel que R correspond à l'ensemble des récompenses (ou punitions) attribuables au robot).

Le but de l'AR est de réitérer les trois étapes citées ci-dessus jusqu'à obtenir la politique optimale Π^* qui va mener le robot d'un état initial s_0 vers un état final s_f en maximisant les récompenses ou en minimisant les punitions qu'il aura cumulées durant son évolution. L'un des points limitant l'utilisation exhaustive de l'AR à différents types de systèmes de contrôle, correspond à l'explosion combinatoire qui peut résulter de l'augmentation de l'espace d'état ou de commande du robot (appelée en anglais "*the curse of dimensionality*").

[5]La lecture de nombreux manuels sur la manière de conduire une voiture, ne remplacera jamais le fait de prendre une voiture et d'adapter nos actions en fonction de l'évolution de celle-ci.

Ceci n'a néanmoins pas empêché un foisonnement d'applications utilisant l'AR, pour apprendre les paramètres caractérisant le contrôle de robots mobiles. Lynne Parker par exemple au travers de son architecture de contrôle nommée L-ALLIANCE (pour *Learning*-ALLIANCE) (Parker 1997, 94), adapte en utilisant l'AR, les taux de motivation au *consentement* ou à l'*impatience* qui caractérisent l'activation des groupes de comportements de l'architecture de contrôle ALLIANCE (cf. §3.2.5, page 63). L-ALLIANCE intervient spécifiquement en adaptant les taux d'accroissement ou de diminution de ces motivations en fonction de l'efficacité du système multi-robots. Tucker Balch dans (Balch 1997) utilise l'AR pour apprendre en ligne la coordination d'un ensemble de schémas moteurs dédié au contrôle de robots footballeurs. Maja Mataric´ dans (Mataric´ 1994) décrit un ensemble de stratégies de renforcement, pour accélérer l'apprentissage d'un groupe de quatre robots autonomes, qui utilisent à la fois, une indication de l'avancement de la tâche en cours "*the progress estimator*" et la somme des récompenses "*reward-summing*" à affecter à chaque entité élémentaire.

L'apprentissage par renforcement soulève un problème majeur par rapport à la stratégie d'affectation des récompenses ou des punitions dans le cas des systèmes multi-robots coopératifs. Poj Tangamchit dans (Tangamchit et al. 2002) compare pour cela l'importance d'avoir des récompenses :

- globales (en fonction de l'activité totale du groupe) ou locales (en fonction des activités individuelles des robots) à affecter à chaque robot,
- différées dans le temps ou instantanées.

Tangamchit conclut, que l'utilisation de l'algorithme de Monte-Carlo avec des récompenses globales "*average-reward-based*" mène l'activité du système multi-robots vers une coopération plus effective, alors qu'utiliser le Q-Learning "*cumulative discounted reward*" (Sutton & Barto 1998) mène le système multi-robots vers des solutions sous-optimales.

Plusieurs travaux dans la littérature traitent la tâche coopérative de poussée d'objets en utilisant l'apprentissage par renforcement. Nous pouvons citer parmi eux les travaux de (Simsarian & Mataric´ 1995), (Ono & Fukuta 1996), (Mahadevan & Connell 1991)...

6.1.3 Les algorithmes génétiques

La théorie darwinienne (Darwin 1859) sur l'évolution par sélection naturelle, tend à expliquer l'évolution par un processus qui permet "la survie du plus apte". Ceci stipule d'une manière générale, que les individus de chaque espèce entrent inévitablement en compétition pour leur survie. Les survivants étant par définition, ceux qui donneront naissance à la génération suivante, car ils possèdent les caractéristiques les plus favorables pour leur survie. Ces caractéristiques seront transmises à leurs descendants par l'hérédité.

Chaque génération sera donc mieux adaptée que les précédentes à son environnement. Ce processus continu de variations est la source, pour Darwin, de l'évolution des espèces.

L'appellation générique d'algorithmes évolutionnistes désigne des systèmes calculatoires de résolution de problèmes, qui s'inspirent des mécanismes adaptatifs de l'évolution des espèces animales. Bien que simpliste d'un point de vue biologique, ces algorithmes sont suffisamment performants pour fournir des mécanismes de recherche adaptatifs robustes et puissants. La robotique évolutionniste plus spécifiquement, utilise plus de méthodes pour améliorer progressivement les techniques de contrôle, que de techniques d'optimisation déterministes. En effet, le manque de modèle d'évolution du système impose ce choix (Harvey et al. 1996).

Les Algorithmes Génétiques "AG" font partie de ces algorithmes évolutionnistes qui ont été initiés par les biologistes dès le début des années 50. Vers les années 70, Holland et ses collègues (Holland 1975) ont reformulé les AG de telles sortes qu'ils soient plus facilement exploitables par les sciences de l'ingénieur. Les AG sont basés sur le principe d'évolution d'une *population d'individus*. Par analogie avec la génétique, chaque *individu* de cette *population* est un *chromosome* et chaque caractéristique composant l'*individu* est un *gène*.

Les AG disposent de grandes potentialités pour l'optimisation paramétrique. Une optimisation paramétrique peut être vue comme un problème de recherche (fouille) dans un espace à plusieurs dimensions. Chaque dimension correspond à un paramètre à fixer, codé dans une petite section du chromosome.

Les aspects parallèles[6] et aléatoires biaisés[7], qui caractérisent le processus de recherche de la solution optimale par AG, permettent d'éviter les minima locaux qui pourraient caractériser le paysage de l'espace de recherche. Cependant, la généricité et la souplesse d'utilisation des AG, ont parfois pour conséquence de ne pas garantir la convergence vers la solution optimale. Ceci est d'autant plus vrai que le temps de calcul est limité, on obtient alors des solutions satisfaisantes mais pas toujours optimales.

Dans ce qui suit un ensemble de définitions et de mécanismes propres aux algorithmes génétiques est donné.

6.1.3.1 Mécanismes utilisés par les algorithmes génétiques

Un algorithme génétique fonctionne typiquement selon le cycle représenté sur la figure 6.2. Ce cycle est inspiré de la terminologie génétique. Lors de chaque cycle, une nouvelle génération de solutions du problème est produite. Avant d'exécuter ce cycle, une *popu-*

[6] Recherche de plusieurs solutions de façon simultanée.

[7] Utilisation des opérateurs génétiques, combinaison de solutions éloignées...

lation initiale de solutions admissibles doit être fournie. Chaque *individu-solution* de la population est *codé* tel un chromosome afin d'être manipulé par les opérateurs génétiques. L'étape suivante consiste à *évaluer* la qualité de chaque chromosome à l'aide de la fonction d'évaluation "*fitness*". En se basant sur la fitness des chromosomes, un mécanisme de *sélection* est appliqué sur la population afin de permettre de garder principalement les individus les plus adaptés pour être manipulés par les *opérateurs génétiques* de *croisement* et de *mutation*. Une fois que les opérateurs génétiques ont été appliqués sur la population "k", on obtient une nouvelle population "k+1" pour laquelle on peut calculer la *fitness moyenne* et la *fitness maximum*. Ces deux fitness correspondent respectivement à la moyenne des fitness de tous les individus constituant la population k+1 et à la fitness du chromosome le plus adapté au problème posé à la génération k+1. Les cycles génétiques s'arrêtent à la satisfaction d'un critère d'arrêt (représenté par la constante k_{Max} dans la figure 6.2). Chacun des éléments clés cités ci-dessus, est détaillé dans ce qui suit.

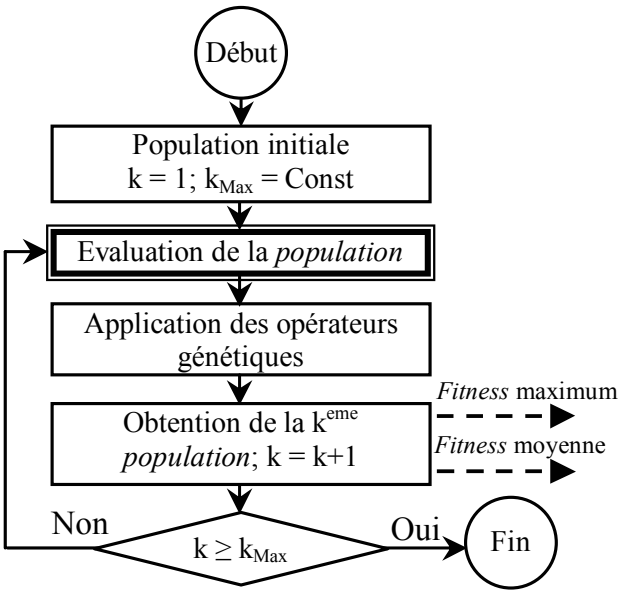


FIG. 6.2 – Cycle génétique

6.1.3.2 Le codage

Cette phase correspond à *représenter génétiquement* le problème, c'est-à-dire définir un codage approprié des solutions sous la forme d'un chromosome. Un codage classique des chromosomes consiste à représenter ses gènes par des variables binaires (0 ou 1). Le

chromosome ainsi obtenu est représenté par une simple chaîne de 0 et de 1 (Goldberg 1989).

6.1.3.3 La population initiale

Une fois le codage choisi, une population initiale formée de solutions admissibles du problème doit être déterminée. Ce mécanisme de génération de la population initiale doit être capable de produire une population d'individus non homogène qui servira de point de départ pour les générations futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global. Plusieurs mécanismes de génération de la population initiale sont utilisés dans la littérature (Caux et al. 1995). La méthode la plus classique consiste à générer aléatoirement les chromosomes constituant la population initiale. Cette méthode répond à la nécessité d'avoir une population variée permettant d'explorer des zones diverses de l'espace de recherche. Des heuristiques peuvent aussi être utilisées pour générer des solutions admissibles. Cependant, une telle pratique peut amener l'algorithme génétique à converger vers des optimums locaux. Le problème principal lors de la génération de la population initiale, est le choix de la taille de la population. Les biologistes ont introduit le concept de *diversité requise* de la population. Ainsi, pour survivre, une espèce doit être suffisamment hétérogène. Par ailleurs, une population trop grande augmente le temps de calcul pour faire un cycle génétique. Il faut donc trouver le bon compromis entre taille de la population et le nombre de cycles génétiques que nous voulons réaliser.

6.1.3.4 La fonction d'évaluation "*fitness*"

Chaque individu de la population correspond à une solution donnée du problème à résoudre. C'est afin de mesurer l'adéquation de chaque individu qu'on utilise une *fonction d'évaluation* qui permet de tester ou de noter individuellement chaque individu, et ceci afin de lui affecter une *valeur sélective* (*fitness*). La complexité de la fonction d'évaluation dépend essentiellement du problème posé et de ses contraintes.

6.1.3.5 Les opérateurs génétiques

Les opérateurs génétiques sont les éléments critiques pour une optimisation efficace. De leurs mécanismes dépendent à la fois le maintien de la diversité des individus composant la population et la convergence de l'AG vers l'optimum global. Dans ce qui suit, les détails des opérateurs les plus couramment utilisés sont donnés.

6.1.3.5.1 Sélection La sélection est un procédé dans lequel chaque individu est choisi en fonction de sa valeur d'évaluation. Cet opérateur génétique inspiré de la sélection naturelle (appelé aussi opérateur de reproduction), est un processus qui permet de choisir parmi la population courante d'individus, les plus adaptés pour se présenter au croisement et à la mutation. Ce choix est crucial pour l'évolution de la performance globale de la population. Vladimir dans (Vladimir 1996) démontre l'importance de l'opérateur de sélection pour la convergence de l'AG vers un optimum global. Parmi les nombreuses techniques de sélection existantes, nous retenons :

1- La sélection par classement : elle consiste à ranger les individus de la population dans un ordre croissant (ou décroissant selon l'objectif) et à retenir un nombre fixé de chromosomes. Ainsi, seuls les individus les plus forts sont conservés. L'inconvénient majeur de cette méthode est la convergence prématurée de l'algorithme génétique vers des optimums locaux. Il est parfois nécessaire de garder quelques individus jugés faibles pour créer la diversité au niveau de la population. De plus, les individus faibles contiennent parfois des gènes susceptibles de contribuer à l'apparition de bonnes solutions.

2- La sélection par la roulette : elle consiste à créer une roue de loterie biaisée (Goldberg 1989) pour laquelle chaque individu de la population occupe une section de la roue proportionnelle à sa valeur d'évaluation. Ainsi, même les individus les plus faibles ont une chance, même minime, de survivre. Si la population d'individus est de taille égale à N, alors la probabilité de sélection d'un individu x_i notée $p(x_i)$ est égale à :

$$p(x_i) = \frac{F(x_i)}{\sum_{k=1}^{N} F(x_k)} \tag{6.1}$$

avec F représentant la fonction d'évaluation des individus.

La probabilité de sélectionner l'individu x_i pour la nouvelle génération de population sera d'autant plus grande que $p(x_i)$ est grande. Avec une telle sélection, un individu fort peut être choisi plusieurs fois. En revanche, un individu faible a moins de chance d'être sélectionné.

3- La sélection par tournoi : elle consiste à choisir aléatoirement deux ou plusieurs individus et à sélectionner le plus fort. Ce processus est répété plusieurs fois jusqu'à l'obtention de N individus. L'avantage d'une telle sélection est d'éviter qu'un individu très fort soit sélectionné plusieurs fois. On pourra toutefois introduire la notion d'élitisme dans cette méthode. Si l'individu le plus fort n'a pas été sélectionné, il est copié dans la génération suivante à la place d'un autre choisi aléatoirement.

6.1.3.5.2 Croisement L'opérateur de croisement "*crossover*", comme son nom l'indique, consiste à croiser deux individus (les parents) afin de donner naissance à un ou à deux individus (le ou les fils) qui auront les caractéristiques des deux parents à la fois. Un principe simple de croisement consiste à prendre aléatoirement une partie des gènes de chacun des deux parents et à les affecter aux fils. Le processus de croisement est essentiel pour explorer efficacement l'espace des solutions possibles.

Dans le but de ne pas renouveler à chaque cycle génétique, tous les individus de la population antérieure, on associe à l'algorithme génétique une probabilité de croisement notée P_{cross} qui permet de décider si les parents seront croisés entre eux ou s'ils seront tout simplement recopiés dans la population suivante.

Il existe dans la littérature plusieurs opérateurs de croisement. Ils différent selon le type de codage adopté et la nature du problème traité. Nous allons décrire dans ce qui suit succinctement les opérateurs de croisement les plus utilisés. Nous les classons en deux catégories : le croisement binaire et le croisement réel.

1- Croisement binaire : cet opérateur génétique a pour objectif de transformer aléatoirement une partie du *génome*. Ces transformations fournissent des heuristiques générales pour l'exploration des solutions possibles car ils apportent de la nouveauté dans le système.

- **croisement en 1-point** c'est le croisement le plus simple et le plus connu dans la littérature. Il consiste à choisir au hasard un point de croisement pour chaque couple de chromosomes. Les sous-chaînes situées après ce point sont par la suite interchangées pour former les deux fils (cf. figure. 6.3),

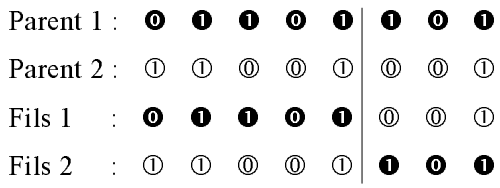


FIG. 6.3 – Croisement en un point de deux chromosomes

- **croisement en n-points** ce type de croisement est utilisé en choisissant aléatoirement n points de coupure pour dissocier chaque parent en $n+1$ fragments. Pour former un fils, il suffit de concaténer alternativement $n+1$ sous-chaînes à partir des deux parents. Ce croisement cherche à explorer tout l'espace des solutions possibles en créant des descendants ayant des caractéristiques très éloignées des parents.

2- Croisement réel : le codage réel des chromosomes nécessite l'utilisation d'opérateurs génétiques spécifiques pour manipuler les chromosomes. En effet, les opérateurs développés pour le codage binaire, peuvent s'ils sont utilisés dans le cadre d'un codage réel, générer des solutions non admissibles qui ne respectent pas les contraintes du problème étudié (Guvenir & Erel 1998). Nous allons décrire dans ce qui suit, un type de croisement uniformément continu. Cet opérateur de croisement nous servira par la suite pour l'optimisation paramétrique effectuée sur l'architecture de contrôle proposée. Plusieurs autres types d'opérateurs de croisement sont donnés dans (Woods 1997).

Croisement uniformément continu ce croisement est proposé par (Michalewicz 1992) et amélioré par (Guvenir & Erel 1998) pour le cas d'une classification de produits suivant plusieurs critères (e.g., coût, durée de vie, possibilité de remplacement, volume) dans l'objectif de faciliter le contrôle et le développement d'un système de gestion des stocks. Les auteurs ont suggéré cet opérateur pour produire des chromosomes valides. Un chromosome $X = (x_1,\ x_2,\ ...,\ x_n)$ est valide lorsque : $\sum_{i=1}^{n} x_i = 1$.

La génération de descendants valides $X' = (x'_1,\ x'_2,\ ...,\ x'_n)$ et $Y' = (y'_1,\ y'_2,\ ...,\ y'_n)$ à partir des chromosomes $X = (x_1,\ x_2,\ ...,\ x_n)$ et $Y = (y_1,\ y_2,\ ...,\ y_n)$ valides, s'obtient ainsi : $x'_i = sx_i + (1-s)y_i$ et $y'_i = (1-s)x_i + sy_i$, où s une constante choisie aléatoirement, à chaque opération de croisement, dans l'intervalle [-0.5, 0.5].

6.1.3.5.3 Mutation Des mutations apparaissent au hasard dans le génome des organismes vivants qui subit la pression sélective du milieu (Darwin 1859). Quand une mutation est favorable, elle confère un nouveau caractère, et l'organisme qui l'acquiert a plus de chance de se reproduire et de le transmettre à sa descendance. À l'inverse, lorsqu'elle est défavorable, l'organisme meurt et le nouveau caractère disparaît. Ainsi, au fil des générations, de nouveaux caractères peuvent apparaître, se maintenir et s'accumuler en donnant ainsi naissance à de nouvelles espèces.

L'opérateur de mutation, tel qu'il est défini dans les AG, correspond à une modification aléatoire d'une partie d'un chromosome. C'est un phénomène qui joue le rôle de bruit et empêche l'optimisation de se figer. Il permet ainsi d'assurer une recherche aussi bien globale que locale, selon le poids et le nombre de gènes mutés. Il existe de nombreuses manières de faire muter un chromosome. Pour un problème utilisant le codage binaire, la mutation la plus connue consiste à inverser la valeur d'un bit choisi aléatoirement (Goldberg 1989).

La mutation joue un rôle secondaire par rapport au croisement, car la mutation ne suit pas forcement une logique justifiée. Pour ne pas trop perturber la convergence de

l'AG, tout en laissant apparaître des chromosomes mutants durant le déroulement de l'optimisation, on attribue généralement une faible probabilité P_{muta} d'application de l'opérateur de mutation sur les chromosomes (P_{muta} tend vers zéro).

6.1.3.6 Les algorithmes génétiques appliqués au contrôle de robots mobiles

Les algorithmes génétique ont inspiré beaucoup de travaux d'optimisation d'architectures de contrôle pour robots mobiles. Nous pouvons citer, par exemple les travaux de Liu dans (Liu et al. 1999) qui propose d'apprendre à une multitude de robots, comment appliquer efficacement leurs forces sur un objet pour le pousser vers un objectif. Le codage proposé consiste en une concaténation des chromosomes de chaque agent (représentant la commande à lui affecter) dans un chromosome unique. L'apprentissage se fait sur ce dernier chromosome. L'évolution de l'optimisation se fait via un superviseur qui tend à satisfaire certains critères liés d'une part à l'évolution globale de l'objet à pousser, et d'autre part à celle des robots mobiles (e.g., le caractère lisse des trajectoires qui doivent être obtenues par l'objet poussé, la maximisation des forces engendrées par les robots, etc.). L'algorithme proposé par Liu reste néanmoins à un niveau d'abstraction très élevé par rapport aux actions locales des robots autour de l'objet à pousser. Lucidarme dans (Lucidarme et al. 2002) applique un apprentissage non supervisé pour obtenir un groupe de robots qui soit capable d'explorer efficacement un environnement inconnu. Chaque robot s'auto-évalue alors, en utilisant une fonction qui mesure la distance moyenne qu'il a parcouru depuis la dernière mutation ou le dernier croisement. Le robot arrive ainsi à juger de la pertinence de ses propres paramètres de contrôle. Il est à noter que dans ce cas de figure, les opérations de croisement ne se font que si les robots se rencontrent physiquement, ce qui peut ralentir considérablement l'apprentissage. Barberá dans (Barberá & Skarmeta 2002) définit les règles floues qui régissent le contrôleur d'un robot mobile en utilisant les AG. Ceci dispense donc le concepteur d'avoir recours à un expert pour établir ces règles. Grefenstette dans (Grefenstette & Schultz 1995) développe un mécanisme d'apprentissage des règles réactives pour le cas d'une tâche de navigation en présence d'obstacles. Les règles réactives apprises consistent à trouver l'action (vitesses de translation et de rotation) à exécuter par le robot en fonction de ses informations capteurs. Les règles à apprendre se présentent sous la forme suivante :

Si (front_sonar $<$ 30 **Et** distance $>$ 10) **Alors** rotation de 20°/s.

Si (front_ir $<$ 5) **Alors** vitesse = -1 cm/s.

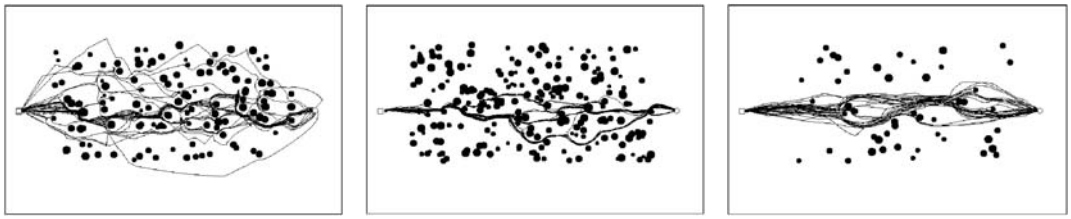
Si ...

Dans cette formulation du problème de navigation, chaque règle est considérée comme un gène, et un ensemble de cinq règles est considéré comme un chromosome qui permet

de réaliser la navigation. Pour chaque cycle d'apprentissage, les positions des obstacles sont changées. Les règles sont évaluées individuellement afin de leur affecter une *fitness*. A la fin de l'apprentissage, Grefenstette retient les cinq règles les plus performantes. Dorigo dans (Dorigo & Schnepf 1991) utilise les algorithmes génétiques pour apprendre à un robot à suivre un objectif mobile tout en évitant les obstacles. Les algorithmes génétiques sont spécifiquement utilisés ici pour déterminer quand le robot doit activer un comportement au lieu d'un autre.

Ram dans (Ram et al. 1994) utilise les algorithmes génétiques pour trouver les gains optimaux régissant des schémas moteurs (*move-to-goal*, *avoid-static-obstacle*, *noise*) qui servent pour la navigation d'un robot mobile en présence d'obstacles. Parmi ces gains nous citons : *goal-gain* : force avec laquelle le robot est attiré par la cible ; *obstacle gain* : force avec laquelle le robot s'éloigne des obstacles ; *obstacle sphere-of-influence* : distance de l'obstacle maximale à laquelle le robot est repoussé ; *noise-gain* : amplitude du déplacement aléatoire ; *noise persistance* : nombre d'échantillons de temps où le vecteur de bruit est gardé constant. Ces gains contrôlent la direction et la vitesse linéaire d'évolution du robot. En fonction des gains trouvés par optimisation, Ram désigne trois comportements principaux qu'il considère comme des niches écologiques "*ecological niches*". Ces niches écologiques sont discriminées entre autres, via des considérations visuelles des trajectoires obtenues par les robots (cf. figure. 6.4). Les types de robots obtenus sont :

- *safe* : robot navigant avec le moins d'impacts possibles,
- *fast* : robot navigant le plus rapidement,
- *direct* : robot navigant avec un grand taux d'action pour aller tout droit.



(a) Robot de type "*safe*" (b) Robot de type "*fast*" (c) Robot de type "*direct*"

FIG. 6.4 – Trajectoires obtenues pour différents types de robots, et pour différents encombrement d'obstacles (les points noirs) dans l'environnement (Ram et al. 1994).

6.2 Optimisation paramétrique de l'architecture de contrôle proposée

Nous introduisons dans ce qui suit, une méthodologie d'optimisation paramétrique fondée sur les algorithmes génétiques. Celle-ci sera appliquée à l'architecture de contrôle proposée au chapitre V (cf. figure. 5.2, page 113). Nous rappelons que cette dernière utilise le PSAHH pour coordonner les comportements élémentaires entre-eux, et qu'elle est dédiée pour induire une coopération effective d'un ensemble de robots réactifs. Ces robots partagent l'objectif commun de pousser efficacement un objet vers une cible, et ceci dans la mesure du possible en évitant de se gêner. La réalisation de la TCPO exige donc pour le robot de satisfaire plusieurs critères simultanément (e.g., pousser l'objet, éviter les autres robots et répondre aux signaux altruistes) ce qui justifie l'utilisation des blocs de fusion $\Sigma_i|_{i=1..3}$ (cf. figure. 5.2, page 113).

Le choix des gains $\mathbf{g}_{ij}|_{i=1..3,j=1..3}$ propres aux blocs de fusion $\Sigma_i|_{i=1..3}$ peut être fait empiriquement (cf. §5.2.2, page 115). Ces gains sont choisis alors de telle sorte à pouvoir *visualiser l'émergence* d'un certain nombre de comportements émanant du groupe de robots. Cependant, cette détermination des gains peut être très fastidieuse, et ce vu d'une part, la multitude des paramètres à fixer[8] et d'autre part la non-exhaustivité des tests effectués[9] pour valider la pertinence des paramètres obtenus. Par conséquent, afin d'éviter une trop longue phase d'essais-erreurs pour trouver les gains les plus adaptés aux blocs de fusion $\Sigma_i|_{i=1..3}$, nous proposons d'utiliser les algorithmes génétiques pour réaliser cette optimisation paramétrique. C'est plus précisément sur les valeurs des gains $\mathbf{g}_{3j}|_{j=1..3}$ du bloc de fusion Σ_3 que se porte notre optimisation. Ce bloc de fusion a comme objectif de satisfaire plusieurs critères simultanément : attirer le robot le plus rapidement vers l'objet à pousser (contribution du comportement d'*attraction à la boite*), éviter les autres robots ou obstacles (contribution du comportement d'*évitement d'obstacles*), et finalement si le robot détecte des signaux altruistes, y répondre en conséquence (contribution du comportement de *réponse aux signaux altruistes*).

6.2.1 Méthodologie proposée pour l'optimisation

Le codage des chromosomes s'est fait en considérant chaque triplet de gains $\mathbf{g}_{31}$, $\mathbf{g}_{32}$, $\mathbf{g}_{33}$ affectés au bloc de fusion Σ_3, comme étant un chromosome élémentaire de la popu-

[8] Paramètres de gains qui sont de plus couplés entre-eux.

[9] Tests pour les différents modes de fonctionnement du système multi-robots, qui peut manifester des caractéristiques hautement dynamiques.

lation d'individus[10]. Nous rappelons que les gains affectés pour chaque chromosome sont contraints par l'équation 5.1, page 108, qui stipule que la somme des gains d'un bloc de fusion est égale à 1. La figure 6.5 montre la méthodologie appliquée pour évaluer chaque chromosome de la population d'individus.

L'un des points essentiels qui doit caractériser une phase d'évaluation de chromosomes, concerne le crédit qui peut être attribué aux fitness trouvées. Dans notre cas particulier, cette fitness doit représenter le degré d'adéquation du chromosome (paramètres de gains) à réaliser la fonction attendue du bloc de fusion Σ_3 (cf. §5.2.1, page 112). A cette fin, nous avons choisi de confronter chaque chromosome de la population à un nombre important de situations que le robot peut rencontrer dans son environnement dynamique.

L'évaluation de la *fitness instantanée* $\boldsymbol{Fitness}_{kij}$ (cf. figure. 6.5) d'un chromosome se fait en fonction :

- du Contact ou non du Robot avec la Boite à pousser, ce qui est représenté par la variable booléenne $\boldsymbol{CRB}_{kij}$ ($\boldsymbol{CRB}_{kij}$ = '1' si le robot est en contact avec la boite, et '0' sinon),
- des Déplacements relatifs du Robot par rapport : à la Boite, aux Obstacles et finalement par rapport aux Signaux altruistes, en l'occurrence $\boldsymbol{DRB}_{kij}$, $\boldsymbol{DRO}_{kij}$, $\boldsymbol{DRSA}_{kij}$, $\boldsymbol{DRSR}_{kij}$ respectivement, et ce s'ils sont détectés par les capteurs du robot. Ainsi, ces déplacements relatifs nous donnent une information, si en appliquant la commande $\mathcal{C}_j$, le robot va plutôt se rapprocher ou s'éloigner des éléments caractéristiques détectés (boite, obstacles, signaux altruistes).

Plus spécifiquement, le calcul de $\boldsymbol{Fitness}_{kij}$ à chaque pas d'échantillonnage est obtenu en utilisant l'agrégation d'objectifs (Hajela & Lin 1992), ce qui donne :

$$\begin{aligned}\boldsymbol{Fitness}_{kij} = \alpha_1 \times \boldsymbol{CRB}_{kij} + \alpha_2 \times \boldsymbol{DRB}_{kij} - \alpha_3 \times \boldsymbol{DRO}_{kij} + \\ \alpha_4 \times \boldsymbol{DRSA}_{kij} - \alpha_5 \times \boldsymbol{DRSR}_{kij}.\end{aligned} \tag{6.2}$$

Avec : $\alpha_i|_{i=1..5}$ correspondent aux priorités accordées pour chaque critère élémentaire affectant le comportement global d'attraction à la boite. Nous avons choisi de donner plus de priorité respectivement, pour le contact avec la boite α_1=2 qui donne par conséquent

[10] Ces gains couplés aux commandes générées par les comportements élémentaires d'*attraction à la boite*, d'*évitement d'obstacles* et de *réponse aux signaux altruistes* respectivement, caractérisent complètement le mouvement du robot quand le bloc de fusion Σ_3 est actif.

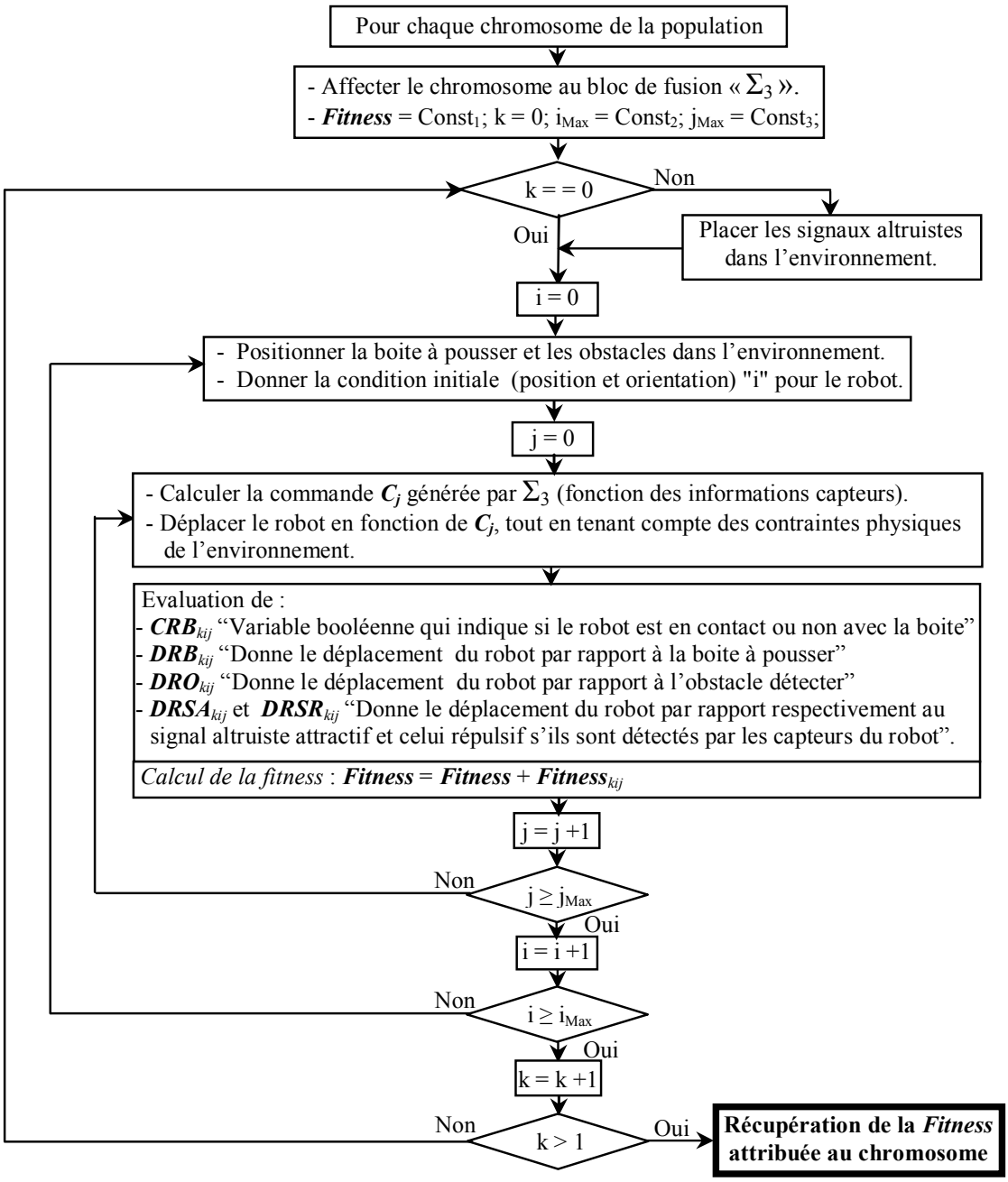


FIG. 6.5 – Méthodologie appliquée pour évaluer les chromosomes attribués à Σ_3

plus de poids aux chromosomes qui font déplacer le robot plus rapidement vers l'objet à pousser, α_2=1.4 pour aller vers la boite, α_3=1 pour éviter les obstacles, et finalement

α_4 et α_5=0.4 pour tenir compte des signaux altruistes.

Il est à noter que les variables liées aux déplacements relatifs du robot par rapport aux autres agents (i.e., $\boldsymbol{DRB}_{kij}$, $\boldsymbol{DRO}_{kij}$, $\boldsymbol{DRSA}_{kij}$ et $\boldsymbol{DRSR}_{kij}$) sont positives quand le robot se rapproche des agents concernés (i.e., respectivement la boite, les obstacles et les signaux altruistes) et ont une valeur négative si le robot s'éloigne d'eux.

La *fitness finale* affectée pour chaque chromosome de la population est calculée comme suit :

$$\boldsymbol{Fitness} = \sum_{k=0}^{1} \sum_{i=0}^{i_{Max}} \sum_{j=0}^{j_{Max}} \boldsymbol{Fitness}_{kij} \tag{6.3}$$

Avec : l'indice "k" représente les itérations faites avec ou sans les signaux altruistes qui entourent la boite, "i" représente les différentes positions et orientations initiales prises par le robot pour chaque début de simulation, et "j" le nombre de pas de simulation (ou de commande) que le robot doit réaliser avant d'arrêter la i^{eme} simulation.

6.2.1.1 L'environnement de test

Pour appliquer la méthodologie d'optimisation paramétrique proposée, nous avons utilisé le simulateur *MiRoCo* (cf. §7.3, page 159). *MiRoCo* nous a donc permis d'évaluer chaque chromosome d'une manière intensive, c'est-à-dire via un nombre important de situations susceptibles d'être rencontrées par le robot dans son environnement (cf. figure. 6.5).

La figure 6.6, montre l'environnement de test choisi pour exécuter l'évaluation de chaque chromosome. Cette figure nous montre aussi une configuration initiale (position, orientation) prise par le robot pour commencer sa navigation vers la boite.

La modélisation précise de l'environnement sous *MiRoCo* nous a permis de simuler des interactions entre agents (robot, obstacles, boite, signaux altruistes) d'une manière très réaliste. *MiRoCo* nous a donné aussi la possibilité d'avoir une bonne approximation de la structure du mini-robot ALICE (cf. §8.1.3, page 178). Ainsi, nous avons pu simuler la forme conique du champ d'émission des capteurs infrarouges, les déplacements précis du robot dans son environnement[11], etc.

[11] Cela consiste au fait que, si le déplacement du robot est gêné par un obstacle, alors le robot sera bloqué à sa position courante tant qu'il n'applique pas de commande appropriée pour se débloquer.

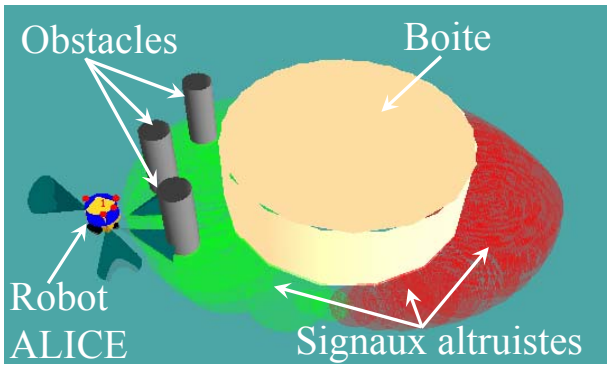


FIG. 6.6 – Configuration adoptée pour l'environnement de test "*setup*", afin d'évaluer les performances des chromosomes.

6.2.1.2 Les opérateurs génétiques utilisés

La réalisation des cycles génétiques (cf. figure. 6.2) passe inéluctablement par l'utilisation d'opérateurs génétiques (cf. §6.1.3.5). Pour le cas de la formulation du problème d'optimisation, nous avons dû utiliser des opérateurs génétiques appropriés. Plus spécifiquement les opérateurs utilisés sont adaptés pour manipuler des chromosomes à gènes réels (Guvenir & Erel 1998), et qui plus est, sont contraints par l'équation 5.1, page 108.

6.2.1.2.1 Opérateur de sélection Cet opérateur est réalisé, en utilisant la roulette de sélection de Goldberg (Goldberg 1989).

6.2.1.2.2 Opérateur de croisement Le principe de l'utilisation de cet opérateur se présente ainsi : ayant deux chromosomes parents valides (respectant l'équation 5.1, page 108) $x = \langle x_1,\ x_2,\ x_3\rangle$ et $y = \langle y_1,\ y_2,\ y_3\rangle$, le croisement de ces deux chromosomes produira deux chromosomes fils valides $x' = \langle x'_1,\ x'_2,\ x'_3\rangle$ et $y' = \langle y'_1,\ y'_2,\ y'_3\rangle$ en suivant la formulation suivante :

$$\begin{cases} x'_i = sx_i + (1-s)y_i \\ y'_i = (1-s)x_i + sy_i \end{cases} \tag{6.4}$$

avec s une variable aléatoire qui prend ses valeurs dans l'intervalle $[0,\ 0.5]$.

6.2.1.2.3 Opérateur de mutation L'opérateur de mutation, que nous avons utilisé, transforme la valeur d'un gène aléatoirement choisi sur le chromosome, en une valeur égale soit à "0" soit à "1", et ce avec une probabilité identique.

Dans le cas où il y a mutation de chromosome, alors ce chromosome doit être normalisé afin qu'il soit toujours valide. L'opération de normalisation des gènes du chromosome $x = \langle x_1,\ x_2,\ x_3 \rangle$ s'effectue comme suit :

$$x_{i=1..3} = \frac{x_i}{\sum_{j=1}^{3} x_j}. \tag{6.5}$$

6.2.1.3 Valeurs des paramètres de l'algorithme génétique

Il y a plusieurs autres paramètres à fixer pour s'assurer de la convergence d'un algorithme génétique. Ces paramètres correspondent : à la taille de la population d'individus, au nombre maximal de générations, et aux probabilités de croisement et de mutation (P_{cross} et P_{muta}). Les valeurs de ces paramètres dépendent fortement de la problématique étudiée : sa nature (linéaire ou pas), la dimension de l'espace de recherche, etc. Souvent, il faut faire appel à une expérimentation importante pour trouver les valeurs des paramètres les plus pertinentes.

Des travaux ont été menés dans ce domaine par (Eiben et al. 1999), (Jong 1975) et (Lee & Takagi 1994) et ils ont montré la difficulté de fixer ces paramètres. Il existe néanmoins des valeurs proches de celles de la nature qui permettent d'obtenir généralement des résultats appréciables. Par exemple, la probabilité de croisement appartient généralement à l'intervalle [0.60, 0.99]. De même, la probabilité de mutation est souvent choisie dans l'intervalle [0.0001, 0.05]. En effet, une mutation avec une grande probabilité perturbe la convergence en induisant une oscillation de la valeur moyenne du critère à optimiser. En revanche, un faible taux de mutation permet d'assurer une bonne exploration de l'espace de recherche. Afin d'accélérer la convergence de l'algorithme génétique, la taille de la population n'excède pas en général la valeur de 1000 chromosomes. Le choix d'une population à faible effectif conduira probablement à l'obtention d'un optimum local. Par contre, une grande population engendrera un temps de calcul excessif. Bien évidemment, les valeurs couramment utilisées mentionnées ci-dessus ne sont là qu'à titre indicatif et n'ont aucune forme d'universalité.

Pour réaliser efficacement l'optimisation paramétrique nous avons utilisé une population de 1000 chromosomes, avec une probabilité de croisement $P_{cross} = 0.8$, et une probabilité de mutation $P_{muta} = 0.02$.

6.2.1.4 Résultats de l'optimisation paramétrique

La figure 6.7 représente l'évolution de la fitness maximale et de la fitness moyenne de la population d'individus au cours de l'optimisation. Le chromosome optimal est obtenu

après 120 cycles génétiques et il correspond au chromosome suivant :
$\diamond\ \Sigma_3^*$: $x^* = <g_{31}^*, g_{32}^*, g_{33}^*> \equiv <0.181, 0.400, 0.419>$.

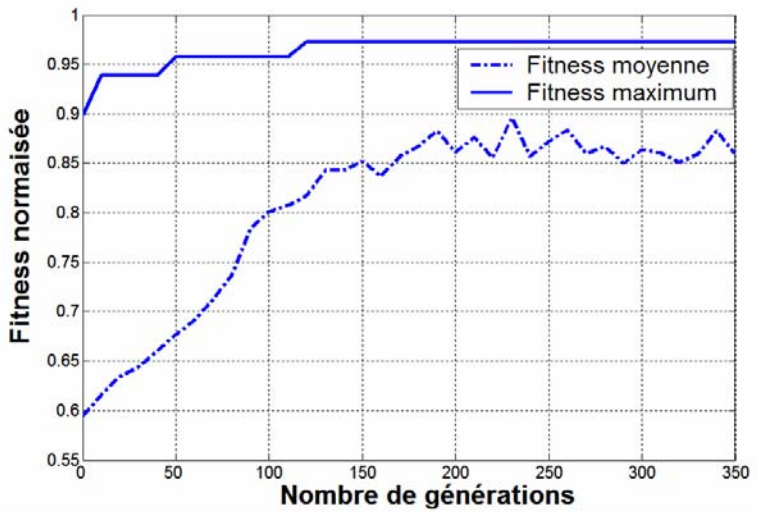


FIG. 6.7 – Évolution de l'optimisation paramétrique

6.2.2 Évaluation de la validité de l'optimisation

Pour valider la pertinence des fitness affectées pour chaque individu de la population[12], nous prenons le chromosome qui a la fitness maximale (optimale), c'est-à-dire $x^* = <0.181, 0.400, 0.419>$ et un autre chromosome $y = <0.596, 0.397, 0.006>$ dont la valeur de fitness se trouve au milieu des 1000 chromosomes appartenant à la population finale. L'évaluation consiste alors, à affecter séquentiellement chacun des deux chromosomes x^* et y au bloc de fusion Σ_3, et à effectuer un grand nombre de simulations avec l'architecture de contrôle globale basée sur le PSAHH pour réaliser la TCPO (cf. figure. 5.2, page 113).

Il est à noter que les autres blocs de fusion, c'est-à-dire Σ_1 et Σ_2 se voient affecter les mêmes paramètres de gains obtenus empiriquement au chapitre V, c'est-à-dire :

- Σ_1 : g_{11}= 0.500, g_{12}= 0.400, g_{13}= 0.100
- Σ_2 : g_{21}= 0.500, g_{22}= 0.450, g_{23}= 0.050

Avec : g_{11}, g_{21} étant les gains affectés respectivement pour les comportements de *poussée de boite* et de *repositionnement*, g_{12}, g_{22} correspondant au comportement d'*évitement*

[12]Sachant que chaque fitness obtenue par la méthodologie d'évaluation explicitée en figure 6.5 est censée montrer le degré d'adéquation du chromosome à réaliser le comportement global d'attraction vers la boite (quand il est affecté bien évidemment au bloc de fusion Σ_3).

d'obstacles et $\boldsymbol{g}_{13}$, $\boldsymbol{g}_{23}$ correspondant au comportement de *réponse aux signaux altruistes*.

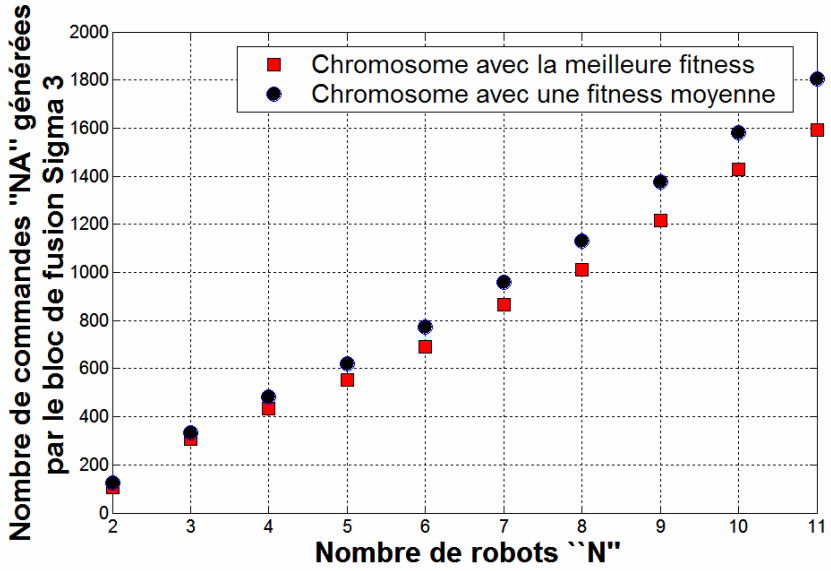
Les simulations groupées, que nous avons effectuées pour réaliser cette évaluation, ont été réalisés avec *Nc* égale à 2 (le nombre minimum de robots qui doivent coopérer pour pousser la boite). Nous augmentons par la suite le nombre de robots N présents dans les simulations de 2 jusqu'à 11 robots. Pour avoir une indication statistique plus précise sur les résultats des simulations, nous faisons pour chaque N, un nombre de simulations $NbSim = 50$. Notons que les deux séries de simulation avec les deux chromosomes x^* et y sont effectuées avec les mêmes conditions initiales, c'est-à-dire avec les mêmes positions et orientations initiales des robots.

La figure 6.8(a) montre la somme du Nombre d'Activations "NA" des blocs de fusion Σ_3 appartenant à chaque robot participant à la TCPO. C'est-à-dire qu'à chaque activation d'un bloc de fusion Σ_3 d'un des robots qui participe à la TCPO, le nombre d'activation "NA" se voit incrémenter de 1. En ce qui concerne la figure 6.8(b), elle montre le temps moyen nécessaire aux robots pour réaliser la TCPO.

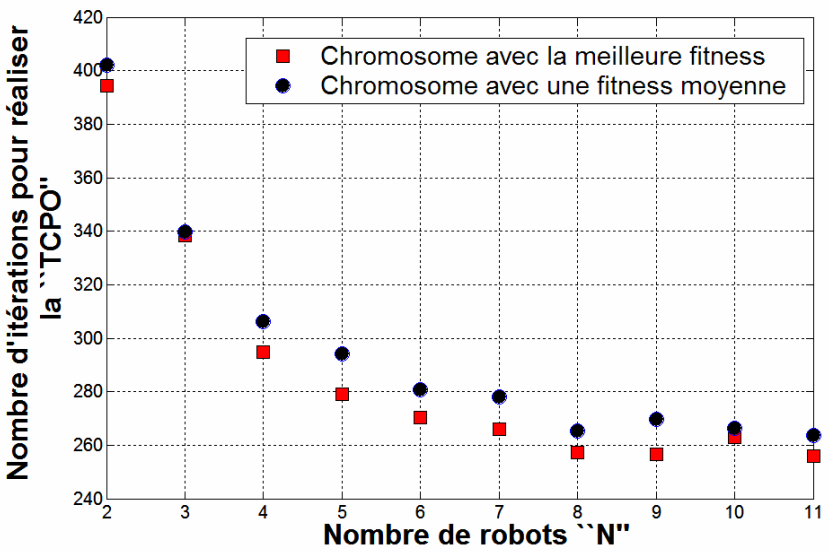
Nous observons que les nombres NA, correspondants à l'utilisation du chromosome x^* avec la meilleure fitness, sont plus petits que les NA correspondants à l'utilisation du chromosome y avec la fitness de moindre importance. Des NA plus petits indiquent que les robots trouvent plus rapidement la boite, et donc contribuent plus concrètement à la pousser. Ceci est confirmé par la figure 6.8(b) qui montre que le temps nécessaire pour réaliser la TCPO devient plus petit quand nous affectons le chromosome à meilleure valeur de fitness au bloc de fusion Σ_3.

6.3 *BibAG* une bibliothèque pour algorithmes génétiques interfacé à *MiRoCo*

Il existe différentes librairies d'algorithmes génétiques mises à disposition pour les besoins de la communauté scientifique. Nous pouvons citer EOlib "*Evolving Objects Library*" de (Keijzer et al. 2001), ainsi que celles proposées dans (Wall 1995) ou dans (Filho et al. 1994). Heitkõter dans (Heitkõter & Beasley 2005) donne une liste importante des différentes bibliothèques proposées dans la littérature. Cependant, la plupart de ces librairies sont en général adaptée plus ou moins à des domaines d'application spécifiques et ne s'appliquent pas directement à notre formulation du problème d'optimisation. Ceci nous a donc amené à développer la bibliothèque nommée "*BibAG*" qui est à la fois fondée



(a) Nombre d'activation "NA" de Σ_3 fonction de N, et du type de chromosome qui est implanté.



(b) Nombre d'itérations pour réaliser la TCPO en fonction de N, et du type de chromosome qui est implanté.

FIG. 6.8 – Comparaison entre l'efficacité des chromosomes

sur une ossature générique d'optimisation par algorithmes génétiques, et qui est complètement adaptée à notre formulation du problème d'optimisation. Le schéma UML[13]

[13] *Unified Modeling Language.*

représenté en figure 6.9 montre la hiérarchie des classes appliquée pour la bibliothèque *BibAG*.

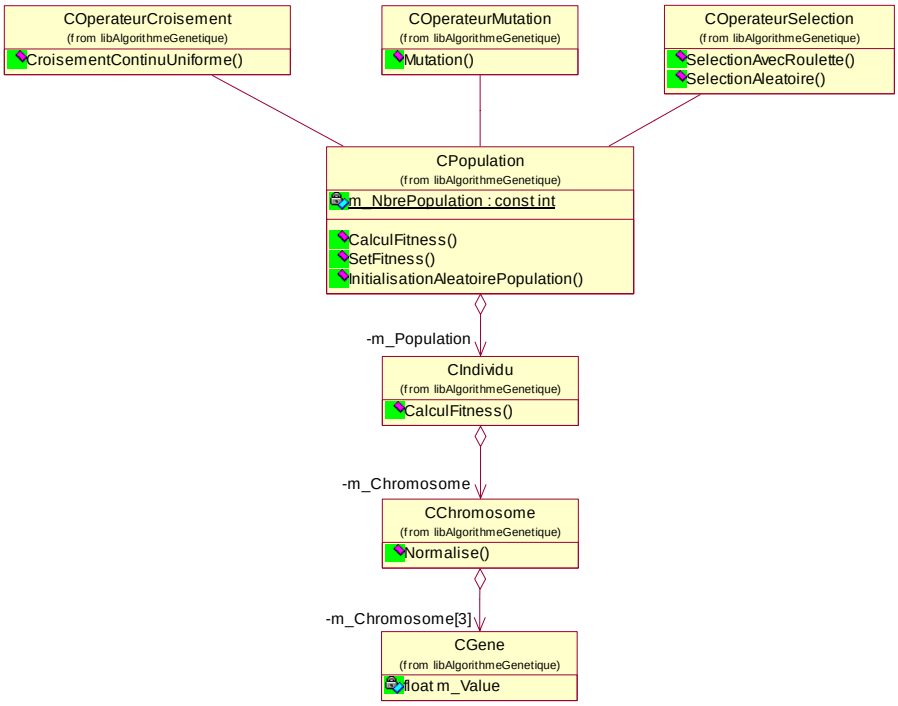


FIG. 6.9 – Diagramme de classes de la bibliothèque "*BibAG*"

La bibliothèque *BibAG* est interfacée au simulateur *MiRoCo* (cf. figure. 6.10) de telle sorte que :

- *MiRoCo* se charge d'évaluer chaque chromosome de la population d'individus. Les chromosomes sont affectés séquentiellement au bloc de fusion Σ_3 afin de le tester massivement sur une multitude de configurations (cf. figure. 6.5, page 139) et ce, dans un environnement de préférence très contraint[14] (cf. figure. 6.6),
- la bibliothèque *BibAG* se charge des autres phases du cycle génétique (cf. figure. 6.2).

6.4 Conclusion

Les algorithmes génétiques nous ont servi dans ce chapitre pour proposer une méthodologie d'optimisation appropriée des valeurs de gains du PSAHH. Nous avons dû pour

[14] En effet, plus l'environnement est encombré, plus l'évolution du robot sera soumise à plus de situations de tests différentes, ce qui implique une évaluation des chromosomes plus conséquente.

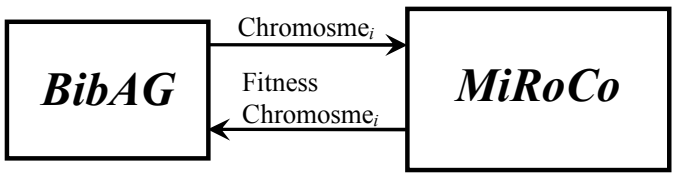


FIG. 6.10 – Interface existante entre la bibliothèque pour algorithmes génétiques "*BibAG*" et le simulateur "*MiRoCo*".

cela utiliser des opérateurs génétiques spécifiques pour, d'une part, manipuler des gènes (gains) à valeurs réelles, et d'autre part, afin de tenir compte des contraintes imposées sur ces gains par le PSAHH.

Nous avons appliqué l'optimisation paramétrique proposée sur l'architecture de contrôle à base de PSAHH et appliqué au cas de la TCPO. Plus spécifiquement l'optimisation s'est portée sur le bloc de fusion Σ_3 qui est chargé d'attirer le robot vers l'objet à pousser. La validation de cette méthode évolutionniste s'est effectuée sur la base d'une étude statistique sur un grand nombre de simulations.

Comme perspective à court terme, nous envisageons de réaliser l'optimisation paramétrique des blocs de fusion Σ_1 et Σ_2 (cf. §5.2, page 113), et ceci en utilisant bien évidemment des métriques appropriées, à l'image de la fitness proposée pour le cas du bloc de fusion Σ_3 (cf. équation. 6.3, page 140).

Troisième partie

Environnement de Simulation et Plate-forme Expérimentale

"*Dans le raisonnement expérimental, l'expérimentateur ne se sépare pas de l'observateur. L'observateur doit être le photographe des phénomènes, son observation doit représenter exactement la nature. Il faut observer sans idée préconçue ; l'esprit de l'observateur doit être passif, c'est-à-dire se taire ; il écoute la nature et écrit sous sa dictée. Mais une fois le fait constaté et le phénomène bien observé, l'idée arrive, le raisonnement intervient et l'expérimentateur apparaît pour interpréter le phénomène*".

Claude Bernard.

"*Il y a deux choses que l'expérience doit apprendre : la première, c'est qu'il faut beaucoup corriger, la seconde, c'est qu'il ne faut pas trop corriger*".

Eugène Delacroix.

Chapitre 7

Simulateur *MiRoCo*

Résumé : *Ce chapitre est consacré d'une manière générale aux mécanismes régissant les simulations de systèmes multi-robots. Pour cela, nous allons faire au préalable un tour d'horizon succinct des différents outils de simulation utilisés dans ce cadre, en mettant en avant leurs caractéristiques et l'objet de leur utilisation. Nous passerons ensuite, aux détails organisationnels et fonctionnels du simulateur MiRoCo "Mini-Robotique Collective" que j'ai conçu et développé tout au long du déroulement des travaux de thèse, afin de tester et valider avec le plus de précision possible les simulations présentées dans les chapitres antérieurs.*

7.1 Introduction

Le besoin de validation par simulation s'avère être un passage obligé avant toute phase de mise en oeuvre effective des solutions proposées pour le contrôle de systèmes complexes. Le contrôle réactif d'un groupe de robots mobiles fait partie de ces systèmes complexes, très difficiles à modéliser donc aussi à contrôler. En effet, le fait de ne pas disposer de modèles qui puissent décrire de manière précise le fonctionnement des systèmes multi-robots, fait que la plupart des travaux effectués dans ce cadre se basent majoritairement : soit sur l'évaluation de métriques caractérisant la réalisation des tâches coopératives à effectuer (Parker 2001), (Ram et al. 1994) ; soit sur l'observation d'apparition de patterns dans l'environnement (Melhuish 2001) ; ou bien sur des études statistiques de l'influence de tel ou de tel paramètre sur le bon fonctionnement des tâches à exécuter (Kube & Zhang 1997), (Balch & Arkin 1995*a*), (Adouane & LeFort-Piat 2004*c*).

Le fonctionnement d'un système multi-robots est principalement lié au type de contrôle qu'on lui applique. Cependant, il est aussi soumis à une multitude d'autres types de paramètres hétérogènes. Ces paramètres sont liés aux phénomènes physiques tels que : les frottement ou les glissements, les interférences des signaux lors des communications, les aléas de fonctionnement des capteurs. Tous ces paramètres réunis font que les phases de validation expérimentale sur les systèmes multi-robots deviennent longues et fastidieuses. En effet, il n'est pas toujours aisé de savoir quelle est l'incidence de tel ou tel paramètre sur les dynamiques d'évolution atteintes par le système. Dans le but de réduire le temps consacré à ces phases de validation expérimentale, on est bien souvent contraint d'isoler les phénomènes (par groupe ou individuellement) afin d'étudier quelle est leur véritable influence sur le système complexe étudié. Dans ce cadre, *l'étude par simulation* s'avère être d'une importance capitale pour l'observation et l'explication des phénomènes, comme peuvent l'être les comportements émergents liés à l'intelligence collective. Les propriétés formelles et les paramètres induisant ces phénomènes sont ainsi isoler et permettent par conséquent d'explorer les possibilités effectives du contrôle proposé, et ce, indépendamment des contraintes matérielles qui peuvent être trop particulières, et fausser ainsi l'analyse induite sur les contrôles proposés.

Les autres avantages liés aux phases de simulation sont nombreux et variés tels que :

- le gain en temps durant les phases de développement et de test,
- l'évitement du problème des pannes inopinées des robots réels (e.g., une soudure qui rompt, un composant défectueux, une alimentation faible),

- l'exhaustivité des possibilités de tests de tout type de paramètres conditionnant les systèmes multi-robots, comme par exemple l'étude de l'influence du paramètre *nombre* de robots participant à la tâche coopérative, c'est-à-dire l'étude de l'effet de masse. Effectivement, ce paramètre ne peut pas être étudié d'une manière exhaustive en expérimentation matérielle, à part si on se restreint à un nombre de robots très réduit (de l'ordre de la dizaine), et même dans ce cas de figure, les résultats seront très dépendants des aléas caractérisant le système multi-robots,
- la réduction des coûts pour le cas des systèmes trop onéreux pour procéder à de multiples phases d'essais-erreurs.

Les points cités ci-dessus et bien d'autres nous ont conforté dans l'idée de passer par des phases de simulations intensives (i.e., en mode *batch*) pour valider des points clés liés aux architectures de contrôle ainsi qu'à l'optimisation paramétrique proposées aux chapitres IV, V et VI.

Afin de contrôler un système en automatique dite traditionnelle, le concepteur doit au préalable disposer de modèles mathématiques (équations différentielles, représentation d'état, etc.) décrivant le système à contrôler ou plus spécifiquement un modèle qui décrit l'évolution des variables à asservir en fonction des commandes appliquées. L'étape qui suit, correspond à la synthèse (toujours basée sur une démarche analytique) du contrôleur qui répond aux performances de robustesse, précision et de rapidité exigées pour le contrôle du système. Néanmoins, pour le cas des systèmes multi-robots, les roboticiens, ne disposant pas de modèle précis d'évolution de ces systèmes[1], font appel bien souvent aux simulations multi-agents pour les modéliser et ainsi les contrôler. Cette modélisation dans une perspective multi-agents consiste à décomposer le système en un ensemble de composants (agents) qui disposent d'une sémantique d'interaction fixée (/établie). Cette modélisation comprend aussi la définition de l'environnement dans lequel évoluent les agents. La simulation de ce modèle permet d'observer le comportement global du système et les comportements locaux des entités sous l'influence de certains paramètres. La validité de ces simulations reste cependant tributaire de la justesse des modèles (cf. figure. 7.1) implémentés dans le simulateur, tels que le modèle d'interaction des robots, le modèle du mouvement des agents, le modèle de communication entre robots.

Le développement du simulateur *MiRoCo* "Mini-Robotique Collective" a été l'occasion pour nous de concevoir un environnement multi-agents, avec toutes les techniques de conception et de programmation que cela impose. Le développement de *MiRoCo* a été

[1] Représentant l'effet des actions locales des robots, sur l'évolution du système, d'une manière globale.

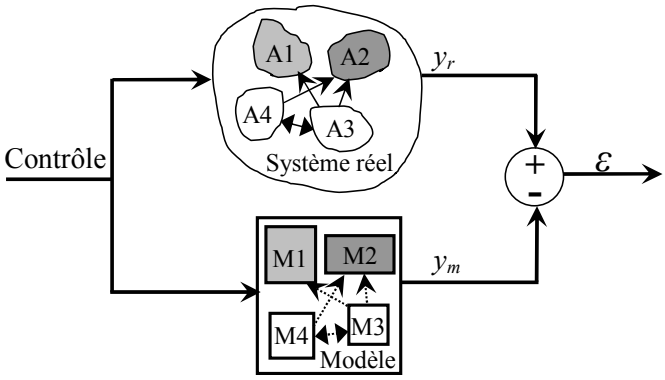


FIG. 7.1 – Modélisation d'un système multi-robots par une simulation multi-agents. Pour une modélisation précise, l'idée est de faire tendre $\varepsilon \longrightarrow 0$.

principalement motivé par le fait de disposer d'un outil qui modélise le plus fidèlement possible ce que nous voulions réellement simuler, en l'occurrence un groupe de robots qui puissent communiquer et interagir d'une manière précise, et ceci afin de réaliser des tâches de coopération. Une fois obtenue les spécificités souhaitées de *MiRoCo*, les tests des architectures de contrôle proposées aux chapitres IV et V, et l'optimisation paramétrique du chapitre VI se sont faits d'une manière toute naturelle, et ceci en gardant à l'esprit que les dynamiques d'interaction générées entre agents, tendent à représenter le plus fidèlement possible le modèle réel du système multi-robots.

Ce chapitre ne se veut pas exhaustif par rapport aux détails techniques du développement de *MiRoCo*, mais plutôt un chapitre traitant certaines spécificités conceptuelles et de modélisation d'un environnement multi-robots. Ces spécificités conditionnent le fait que le modèle simulé se rapproche ou non du système multi-robots réel (cf. figure. 7.1).

7.2 Les environnements de simulation multi-robots

Simuler un phénomène, un système, ou tout autre élément modélisable, pose toujours la question importante du niveau de détail à atteindre pour que les simulations soient satisfaisantes. Même s'il est évident que le monde est sans nulle doute son meilleur modèle[2], il reste qu'il y a toujours des phénomènes physiques plus prédominants que les autres par rapport à l'objet précis des simulations. C'est ce qui nous permet donc de nous restreindre qu'à la modélisation des phénomènes importants. Cependant, même

[2] "*The World is its own Best Model*" (Brooks 1990).

dans ce cas de figure, il faut généralement se restreindre aussi à un niveau de détail qui satisfasse le dilemme temps-précision[3]. En effet, en plus du fait qu'une modélisation précise des phénomènes ralentit considérablement les simulations, il y a aussi le fait qu'il n'est pas nécessaire d'atteindre des niveaux de précision très importants pour pouvoir tirer des conclusions ou valider les méthodes, les concepts ou les solutions apportées.

L'investigation menée au début de nos travaux de recherches sur les différents environnements de simulation que nous pouvions potentiellement utiliser, a montré l'existence de plusieurs types d'outils de simulation pour systèmes multi-robots (ou plus généralement pour systèmes multi-agents situés). Certains outils relativement plus basiques simulent des environnements discrets[4], avec des robots à nombre d'états ou de perceptions réduit. D'autres outils plus sophistiqués vont jusqu'à la modélisation de la physique de l'environnement dans lequel évolue les différents robots. Le choix du degré de sophistication du simulateur à utiliser, dépend de l'objet même des simulations. En effet, le simulateur utilisé pour valider des concepts ou des mécanismes organisationnels d'entités autonomes ne sera pas forcément le même que celui qui sera utilisé pour l'étude des interactions microscopiques entre entités autonomes situées.

Parmi les environnements de simulation qui restreignent les positions des robots à des positions discrètes dans l'environnement (correspondantes aux cases d'un cadre figé de l'environnement), nous pouvons citer le simulateur Alia (cf. figure. 7.2(a)) de Claude Delaye (Delaye 1993) qui simule des robots explorateurs chargés de récolter du minerai sur une planète inconnue. Le but est d'étudier, par le biais de l'évolution artificielle, l'adéquation matérielle et de contrôle de la colonie de robots utilisée pour réaliser la tâche en question. Une fois le minerai trouvé, il est rapporté à la base. Le cheminement de retour est facilité par la dépose à l'aller de "balises" (cf. figure. 7.2(a)). La particularité de ce travail réside dans le fait que les robots explorateurs sont générés automatiquement par la base, suivant un processus de sélection, basé sur leur capacité à rapporter du minerai. Les mutations, que peuvent subir les robots d'une génération à l'autre, sont à la fois comportementales (variation du contrôle qui leur est affecté), mais également morphologiques (comme l'utilisation de roues ou de chenilles, taille de la benne, etc.) et ceci en fonction de la nature supposée du terrain où évoluent les robots.

[3]Par temps-précision nous voulons dire, temps nécessaire pour obtenir les résultats et précision souhaitée de ces derniers.

[4]La discrétisation de l'espace d'état réduit considérablement les dynamiques d'interaction et d'évolution que l'on peut observer. En effet, ceci entraîne par exemple une représentation partielle des états atteignables par le système multi-robots.

Un autre exemple de simulateurs utilisant un nombre d'états discrets, est celui développé et utilisé par Alexis Drogoul (Drogoul 1993) dans le cadre du projet MANTA "*Modeling an Anthil Activity*" au LIP6. Ce simulateur (cf. figure. 7.2(b)) est dédié à l'étude des sociétés de fourmis en se basant sur le modèle de comportement individuel EMF "*EthoModelling Framework*". MANTA a permis de tester des hypothèses concernant l'émergence de structures sociales, telles que la division du travail, la sociogénèse, les relations de dominance et de hiérarchie au sein d'un type particulier de colonies de fourmis (Ectatomma ruidum). L'univers simulé correspond donc à une fourmilière artificielle, telle qu'on peut la voir dans un laboratoire d'éthologie. Chaque fourmi est représentée par un agent pouvant se déplacer de case en case et/ou effectuer des tâches particulières (soins aux oeufs, fourragement, etc.) suivant le principe d'EMF proposé. MANTA offre des possibilités importantes pour spécifier aisément les comportement des agents, ainsi que de nombreuses facilités graphiques. Cependant, cette plate-forme est plus particulièrement dédiée à la représentation et la validation de concepts organisationnel plutôt qu'à la simulation des interactions physiques précises entre individus situés dans un environnement contraint.

La communauté travaillant sur les SMA, dispose d'un grand nombre d'outils de simulation traitant de problèmes complexes via une modélisation par SMA (cf. figure. 7.1). Pour le cas des simulations de systèmes multi-robots, leur transposition aux plate-formes SMA[5] déjà existantes reste quasi immédiate vu la nature déjà distribuée du système multi-robots. Cependant, la plupart des plates-formes SMA offre pas ou peu de composants dédiés pour manipuler efficacement l'évolution d'agents situés dans un environnement contraint. Olivier Simonin par exemple dans (Simonin 2001) a développé un outil de simulation (cf. figure. 7.2(c)) qui s'exécute sur Madkit[6] (Gutknecht et al. 2000), mais il a du développer au préalable une librairie de classes spécialisées qu'il a interfacé avec les classes abstraites définies dans Madkit. Madkit a permis entre autre de gérer et de tester librement l'ordre d'exécution des actions des agents dans l'environnement, de développer facilement plusieurs représentations graphiques de l'exécution d'une même simulation et d'intégrer des sondes affichant des informations sur le système. Cependant, les actions des agents présents dans les simulations effectuées par Simonin (cas des robots découpeurs ou pousseurs par exemple) restent à un niveau d'abstraction assez élevé par rapport à la physique de manipulation tels que pousser, découper les objets présents dans l'environnement.

[5] L'adresse internet "`http://lil.univ-littoral.fr/Mimosa//#Objectifs`" contient des liens vers certaines des plate-formes SMA les plus couramment utilisées en France.

[6] Madkit "`http://www.madkit.org/`" est l'une des plate-forme de simulation SMA la plus aboutie.

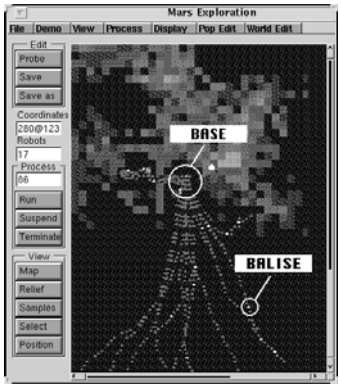
Dans le cadre des simulateurs dédiés pour réaliser la tâche coopérative de poussée d'objets, on trouve aussi "SimbotCity" (cf. figure. 7.2(d)) développé par Ronald Kube (Kube 1997) durant sa thèse afin de simuler entre autre les déplacements d'un objet poussé sous l'effet des actions d'un groupe de robots réactifs (Kube & Zhang 1994). On note que dans le cadre de SimbotCity c'est un modèle géométrique qui décrit la relation existante entre les actions des robots et les déplacements consécutifs de l'objet à pousser.

Les simulateurs dédiés pour les robots footballeurs constituent d'autres types de plate-formes de simulation partagées par plusieurs équipes de recherche de part le monde. Ces équipes traitent généralement des problématiques voisines au contrôle coopératif d'un groupe de robots mobiles. Dans le cadre de la fédération de robots footballeurs "RoboCup[7]" par exemple, il y a en plus des compétitions de véritables robots mobiles, une autre ligue qui se focalise uniquement sur des compétitions en simulation. Le simulateur nommé "RoboCup Soccer Simulator" (cf. figure. 7.2(e)) constitue la base standard et commune du déroulement des compétitions via Internet[8] (Chen et al. 2003). Le fait qu'il y est une plate-forme unique de compétition par simulation, a permis à ce simulateur de bénéficier d'un développement conséquent de la part des différentes équipes de recherche. Cependant, ce simulateur, malgré son aspect attrayant, reste à notre sens trop dédié aux compétitions de robots footballeurs, et les perceptions récupérées par les différents robots ne sont pas aussi modulables et réalistes que celles que nous souhaitons avoir dans nos simulations.

Le simulateur Webots (Michel 2004) (cf. figure. 7.2(f)) commercialisé par la société CYBERBOTICS est à notre sens celui qui correspond le plus à nos besoins en simulation. Il offre une large gamme de possibilités de programmation et d'interface. Il donne aussi à la fois : un affichage très réaliste, une interface d'édition du monde facile à utiliser, et prend en compte plusieurs types de contraintes physiques d'interaction entre les éléments des simulations. Sur un aspect multi-robots, cadre d'utilisation du simulateur qui nous intéresse plus particulièrement, Webots propose des processus de communication inter-robots locaux et globaux satisfaisants. Cependant, le déplacement des entités robotiques s'effectue d'une manière séquentielle, et utilise une base de temps fixe (caractéristiques

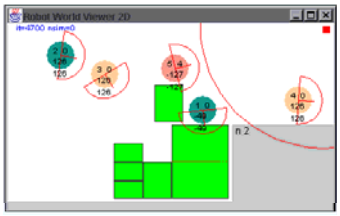
[7]Le but étant de disposé de deux équipes de robots capables de jouer à un jeu collectif dans un environnement commun (jeu proche du football, ou du hockey), et ceci dans le but de tester et de comparer, par des compétitions, différents aspects liés aux stratégies de contrôle et de coopération implémentées sur les robots.

[8]Sous une architecture client/serveur standardisée, les joueurs "clients" reçoivent des informations du "serveur" toutes les 100 ms, contenant entre-autres les positions de tous les agents (11 joueurs par équipe) à l'intérieur du champ de perception des robots, leurs vitesses, etc. A partir de ces informations les clients peuvent calculer des commandes de déplacement ou de manipulation de la balle (e.g., tirer, donner un coup de pied, etc.) à envoyer au serveur.



(a) Alia (Delaye 1993)

(b) Manta (Drogoul 1993)



(c) Simulateur interfacé à Madkit (Simonin 2001)

(d) SimbotSity (Kube 1997)

(e) RoboCup Soccer Simulator (Noda et al. 1998)

(f) Webots (Michel 2004)

FIG. 7.2 – Illustrations de quelques simulateurs présentés dans la littérature

observées pour les anciennes versions de Webots), c'est-à-dire que ceci implique que les entités virtuelles qui évoluent dans l'environnement n'ont pas les mêmes priorités de mouvement et par conséquent d'accès aux ressources (cf. §7.3.2.3.1, page 163).

Il est à noter que la plupart des programmes sources des simulateurs présentés ci-dessus ne sont pas en libre accès, et que les documents qui les décrivent se restreignent généralement à un niveau de détail très faible. Par conséquent, il n'a pas été possible, ni d'utiliser l'un de ces simulateurs afin de l'adapter en fonction de nos exigences en simulation, ni même de nous inspirer efficacement de l'expérience de modélisation acquise par leurs concepteurs afin de concevoir notre propre simulateur *MiRoCo*.

7.3 Développement de *MiRoCo*

Le fait qu'il y est a priori aucun simulateur dans la littérature qui soit, à la fois accessible et qui réponde au cahier des charges que nous nous sommes fixés (cf. §7.3.2.3), nous a amené à développer le simulateur *MiRoCo* (cf. figure. 7.3). Ce simulateur nous a permis d'évaluer la viabilité des architectures de contrôle ainsi que l'optimisation paramétrique proposées aux chapitres précédents pour le cas du système multi-robots traité. Parmi les points importants qui ont caractérisé les résultats obtenus, nous notons d'une part, la précision des simulations effectuées, et d'autre part le caractère intensif des simulations effectuées (plusieurs milliers à chaque fois). Ces deux points nous ont permis d'obtenir des indications statistiques précises sur l'influence de certains paramètres de contrôle sur l'évolution du système multi-robots étudié.

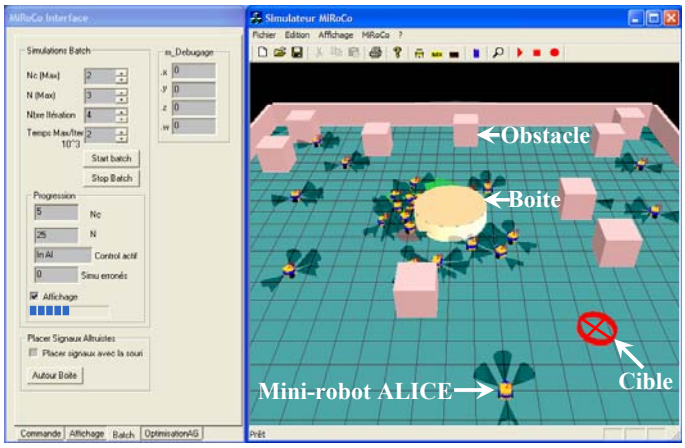


FIG. 7.3 – Simulateur *MiRoCo* "*Mini-Robotique Collective*"

Parmi les points importants à respecter pour simuler correctement un système multi-agents situé, nous pouvons citer :

- la nécessité de séparation des agents de leur environnement. C'est ce qui est caractérisé dans (Magnin 1996) par une forme de dualité entité-environnement,
- la gestion efficace et réaliste de l'évolution temporelle des simulations, car une simulation n'est autre qu'une suite d'interactions consécutives entre agents. Dans le cadre des simulations que nous voulons effectuer, ceci concerne : le déplacement simultané des robots tout en respectant les contraintes de non chevauchement entre agents, l'émission-réception des signaux, le déplacement des robots par rapport aux déplacements des autres agents déplaçables, tels que les objets à pousser dans le cadre de la TCPO,

– la modélisation précise ou du moins réaliste, des lois physiques intervenant lors de l'évolution et des interactions entre agents, car c'est ceci qui conditionne en grande partie, la correspondance entre la dynamique d'évolution du système multi-robots obtenue et celle qui règne dans le monde réel. Magnin dans (Magnin 1996) modélise les interactions entre agents par des règles environnementales (e.g., le rebond d'une balle sur les murs dans le cas des simulations de robots footballeurs). Cependant, ces règles peuvent être d'un nombre très important pour caractériser l'ensemble des interactions possibles, donc il convient de faire attention à leurs définitions pour ne pas risquer d'avoir des contradictions ou des aberrations quant au déroulement des simulations.

Dans ce qui va suivre, nous présentons les grandes lignes des principaux modèles implémentés dans *MiRoCo*, ainsi que l'organisation et les interactions existantes entre les différentes classes constituant le simulateur.

7.3.1 Conception *orientée objet* de *MiRoCo*

Parmi les avantages majeurs d'une conception *orientée objet*, nous trouvons le fait qu'elle crée un lien quasi immédiat entre, d'une part les structures du domaine d'application et leurs mécanismes d'interaction[9] et d'autre part, la structure même du programme informatique (variables, structures, méthodes, etc.). Ceci induit par conséquent une démarche de transposition aisée de ce qui correspond au monde réel vers sa représentation informatique. Nous trouvons aussi une multitude d'autres avantages liés à la conception orientée objet dont bénéficie d'ailleurs énormément le domaine du génie logiciel. Nous pouvons citer parmi eux : la constructibilité ascendante des programmes ; la facilité de maintenance, d'évolution et de modification. *MiRoCo* est issu de cette conception *orientée objet* et il est dédié pour des simulations en robotique collective d'une manière générale.

7.3.2 Classes et organisation des simulations sous *MiRoCo*

La figure 7.4 représente l'ossature organisationnelle globale de *MiRoCo*, qui est décrite ici en utilisant une représentation UML sous forme d'un diagramme de classes[10] (Booch et al. 2000).

[9] Dans notre cas, c'est les structures et les mécanismes d'un système multi-robots.

[10] "Les diagrammes de classes expriment de manière générale la structure statique d'un système, en terme de classes et de relations entre classes. Outre les classes, ils présentent un ensemble d'interfaces et de paquetages, ainsi que leurs relations" (Muller & Gaertner 2000).

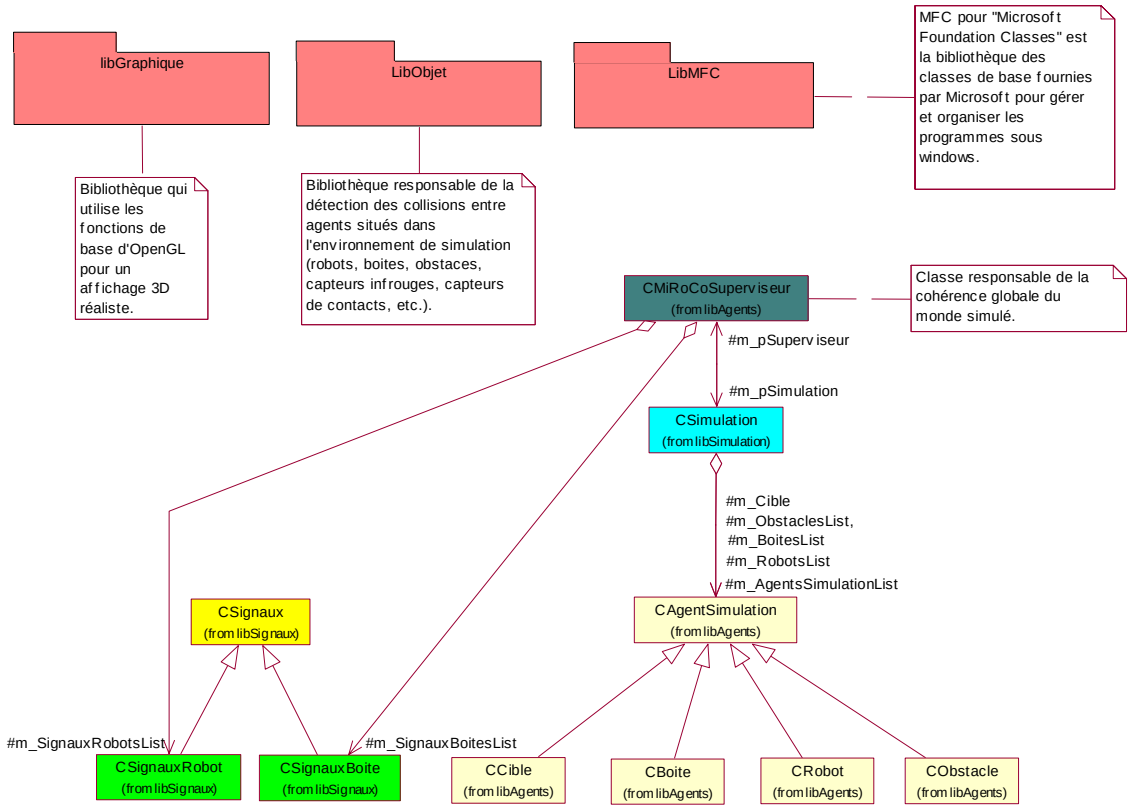


FIG. 7.4 – Schéma UML représentant les principales classes intervenant dans le cadre des simulations sous *MiRoCo*.

Nous décomposons les types d'agents situés dans l'environnement en trois catégories :

- les dynamiques et/ou actifs (les robots) qui ont une complète autonomie de décision et d'action,
- ceux plutôt pseudo-statiques et/ou passifs (e.g., la boite) qui se déplacent uniquement sous l'influence d'autres agents, en l'occurrence sous l'action des robots,
- et finalement ceux qui sont statiques (les obstacles, les murs, la cible, etc.) qui ne changent pas de position ni d'orientation tout au long de la simulation.

7.3.2.1 Agent robot

La figure 7.5 correspond à la représentation logicielle d'un robot physiquement situé dans son environnement. L'agent robot est donc constitué de tous les éléments qui vont lui permettre de percevoir (les capteurs), analyser (le contrôleur) et enfin agir sur son environnement (en utilisant ses moteurs, et son émetteur de signaux).

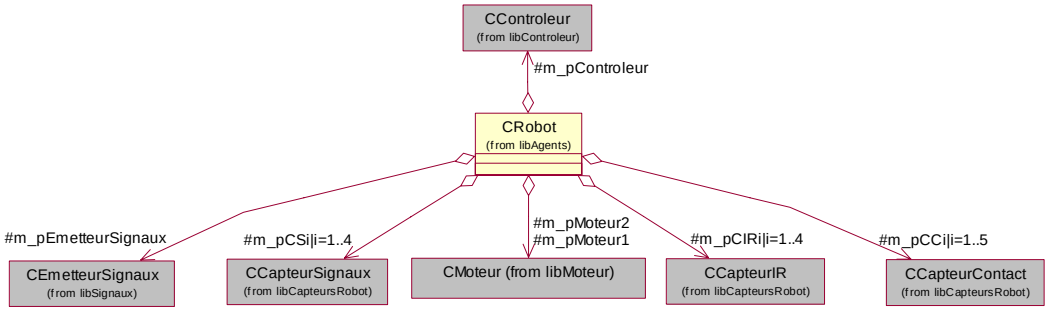


FIG. 7.5 – Schéma UML de l'organisation des classes d'un mini-robot ALICE simulé

Il est à noter, que le robot mobile choisi dans la cadre de notre plate-forme expérimentale, est le mini-robot ALICE (cf. chapitre VIII). C'est pour cette raison que nous avons tenu à ce que le robot simulé (cf. figure. 7.6) corresponde le plus fidèlement à la structure de base d'ALICE, augmenté de certains capteurs nécessaires à la réalisation de la TCPO.

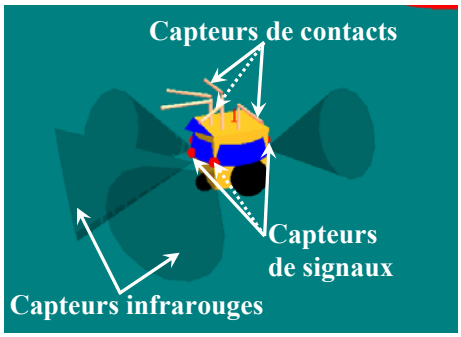


FIG. 7.6 – Capteurs simulés du mini-robot ALICE

7.3.2.2 Agents boite, obstacle, cible

Les autres types d'agents situés dans l'environnement de simulation, correspondent aux :

- agents boite : contrairement à l'agent robot qui peut agir sur son environnement d'une manière autonome, l'agent boite ne peut se déplacer que sous l'impulsion des actions exercées par les autres agents robots. Cette agent dispose néanmoins d'un émetteur de signaux pour que les autres robots puissent le repérer dans l'environnement,
- agents obstacle : ces agents correspondent à des éléments statiques dans l'environnement, et sont censés ne pas être déplacés tout au long de la simulation,

– agent cible : correspond à un cercle avec une position et un rayon fixe dans l'environnement. Cet agent particulier permet par exemple, dans le cas de la TCPO, de connaître la zone où doit être acheminé l'objet à pousser. Nous considérons qu'une simulation est achevée à partir du moment où l'objet à pousser touche le pourtour du cercle représentant la cible.

7.3.2.3 Classe superviseur

Cette classe joue un rôle primordial lors des simulations. En effet, c'est elle qui dicte les lois du monde simulé. Ces lois sont définies de façon à tendre vers les lois physiques qui règnent dans le monde réel. C'est donc cette classe qui fait en sorte que les simulations (interactions entre agents, gestion des signaux dans l'environnement, etc.) soient les plus réalistes et cohérentes possibles. Notons que cette classe pourrait jouer aussi le rôle d'un contrôleur central dans le cas où l'on envisagerait de procéder à ce type de contrôle.

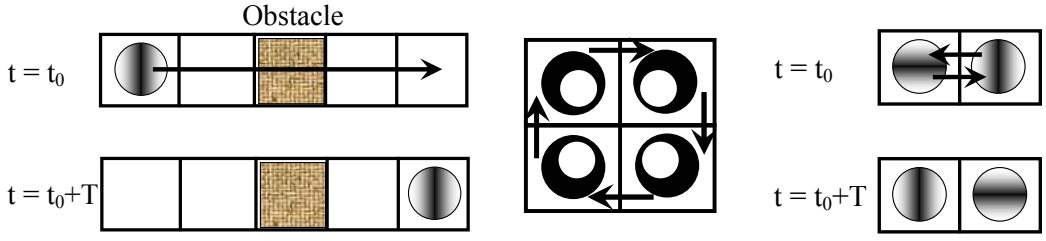
Dans ce qui suit les principaux mécanismes et modèles implémentés dans la classe superviseur sont exposés.

7.3.2.3.1 Moteur physique Pour simuler un environnement multi-robots où les déplacements des robots mènent bien souvent à des conflits spatiaux[11], il est primordial de pouvoir respecter un certain nombre de contraintes afin d'essayer de faire tendre l'exécution des simulations vers ce qui ce passe effectivement dans le monde réel. Ces contraintes sont illustrées dans ce qui suit via un certain nombre d'exemples et de travaux.

– **Contrainte de non-séquentialité des actions des agents :** pour que les agents ne se rentrent pas les uns dans les autres, la solution la plus immédiate consiste à exécuter les actions des agents dans un ordre immuable (séquentiel) et à vérifier à chaque fois que l'agent déplacé ne rentre pas dans les autres agents. Cependant, ceci introduit une relation de précédence indésirable, puisque l'action du robot n-1 est toujours exécutée avant celle du robot n. Ainsi, si deux agents ou plus sont régulièrement en compétition pour une ressource (e.g., surface de la boite à pousser), alors celui ayant son action au rang n-1 prendra systématiquement la ressource aux agents de rangs supérieurs. D'où l'idée inhérente à cette contrainte qui impose que les actions des robots doivent s'exécuter simultanément (en parallèle) afin qu'il n'y ait aucune relation d'hiérarchie d'accès aux ressources si toutefois elles sont sollicitées en même temps,

[11] On appelle conflit spatial, une situation où plusieurs agents tentent d'utiliser simultanément un même espace alors que celui-ci est insuffisant pour le permettre.

– **Contrainte de non chevauchement des agents entre-eux :** cette contrainte est importante au sens que les éléments du monde physique ne peuvent pas se chevaucher. Par conséquent, ce qui est exigé par cette contrainte est le fait que les agents doivent toujours avoir leurs volumes (et/ou surface) complètement disjoints, c'est-à-dire que l'intersection de leurs volumes doit toujours être l'ensemble vide. L'un des moyens couramment utilisé en simulation d'agents autonomes situés, est de décomposer l'environnement d'évolution des agents en un certain nombre de positions discrètes (Drogoul 1993), (Delaye 1993). Ainsi, à chaque pas d'échantillonnage, chaque agent occupera une case bien précise de l'environnement qui sera bien évidemment différente de celles des autres agents qui partagent le même environnement. Cependant, même en discrétisant l'environnement, l'évolution des agents n'est pas à l'abri de certaines situations conflictuelles qui doivent être gérées. La figure 7.7(a) par exemple montre le cas où la position finale d'un agent est plausible, cependant comme l'agent traverse au préalable un obstacle, alors la position finale de l'agent devient complètement aberrante et l'algorithme implémenté doit détecter cela. Dans le cas de la figure 7.7(b), chaque agent souhaite se déplacer dans la case de son voisin, ce qui semble en théorie possible. Néanmoins, lors de la simulation, quel que soit l'agent déplacé en premier, il apparaîtra que sa case voisine est déjà occupée, sauf si l'on prend comme principe d'exécuter toutes les actions en parallèle, avec éventuellement un "retour en arrière" en cas d'incohérence finale. Cet algorithme peut présenter néanmoins des résultats non satisfaisants, par exemple lorsque deux agents souhaitent permuter leurs positions (cf. figure. 7.7(c)), on aura un résultat final cohérent, par contre le croisement des deux agents devrait être impossible, mais ceci n'est pas détecté.



(a) Agent se déplaçant de quatre cases à travers un obstacle !

(b) Échanges de positions non conflictuelles.

(c) Échange de positions conflictuelles !

FIG. 7.7 – Quelques situations conflictuelles induites par les déplacements des agents (Magnin 1996).

Il serait tout à fait possible de présenter d'autres exemples posant problème. Cependant, l'objectif ici n'est pas de faire une présentation exhaustive des différents conflits spatiaux qui peuvent survenir lors de l'utilisation d'un environnement discret, mais l'objectif est de montrer via des exemples assez simplistes[12], des cas de conflits spatiaux qui doivent être résolus afin d'aboutir à des simulations cohérentes et précises.

Par conséquent, afin d'aboutir à la cohérence des mouvements des agents présents dans les simulations, il faut que ces agents, d'une part, ne passent pas les uns au travers des autres, et d'autre part, il faut que leurs actions puissent s'exécuter en même temps et non pas séquentiellement. Cette deuxième contrainte est très importante dans le cas de la TCPO, car nous avons plusieurs agents qui partagent un milieu très confiné (en l'occurrence le pourtour immédiat de l'objet à pousser). Nous notons que dans le cas où on aurait donner une forme de séquentialité (donc de priorité) au mouvement des agents, alors cela conduirait inévitablement à une dynamique d'évolution du système multi-robots tout à fait faussée.

Avant d'expliquer les détails du *moteur physique* que nous avons implémenté sur *MiRoCo*, nous donnons au préalable le modèle de déplacement des agents dans l'environnement continu[13] : à l'instant d'échantillonnage $t+\triangle t$, chaque agent mobile va évoluer de son état initial ($Position_t$, α_t) $\equiv$ (x_t, y_t, α_t) vers un état final ($Position_{t+\triangle t}$, $\alpha_{t+\triangle t}$) $\equiv$ ($x_{t+\triangle t}$, $y_{t+\triangle t}$, $\alpha_{t+\triangle t}$) (cf. figure. 7.8) suivant cette équation :

$$\begin{cases} Position_{t+\Delta t} = Position_t + \vec{V_t} \times \Delta t \\ \alpha_{t+\triangle t} = \alpha_t + (\alpha_t - \angle \vec{V_t}) \end{cases} \tag{7.1}$$

Avec : $\vec{V_t}$ correspondant à la vitesse instantanée de l'agent en question.

L'algorithme 7.1 résume en pseudo-code le *moteur physique* implémenté dans *MiRoCo* qui respecte les deux contraintes mentionnées ci-dessus. La figure 7.9 nous donne une représentation graphique du fonctionnement de cet algorithme pour un cas simple d'un conflit spatial entre deux agents mobiles.

[12] Dus à la discrétisation de l'environnement. Nous notons que pour ce type d'environnement, chaque agent n'a pas de forme explicite vu qu'il doit tenir forcément dans une seule case.

[13] C'est-à-dire qu'un agent peut occuper n'importe quelle position (x, y) de l'environnement.

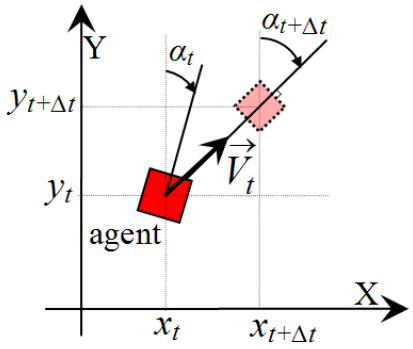
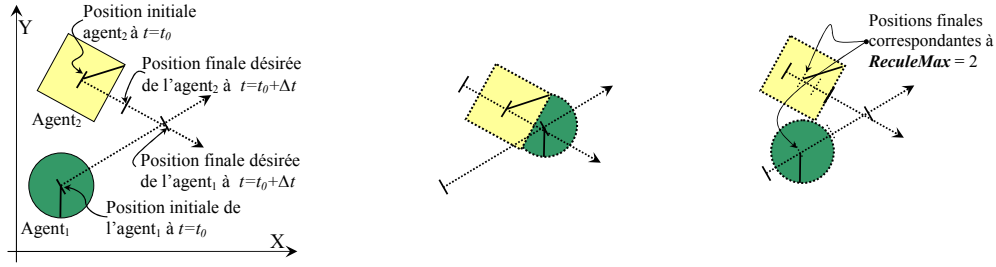


FIG. 7.8 – Modèle de déplacement des agents mobiles dans les simulations



(a) Positions initiales et finales désirées

(b) Mouvement sans l'application du *moteur physique*

(c) Mouvement avec l'application du *moteur physique* (*Précision*=3)

FIG. 7.9 – Application du *moteur physique* implémenté dans *MiRoCo* pour un cas simple de deux agents en mouvement.

Comme cela est indiqué dans l'algorithme 7.1, la précision de l'algorithme augmente proportionnellement avec la valeur de la constante *Précision*. Cette constante permet de décomposer les trajectoires futures (effectuées en un pas d'échantillonnage) des agents mobiles en *Précision* (le nombre) parties égales (cf. figure. 7.9(c)). Une partie est calculée par exemple pour le cas de l'agent$_i$ par $Pas_i = [(1/Précision) \times Vitesse_i \times \Delta t]$. L'idée consiste ensuite à trouver de combien de Pas_i faire reculer la position future de l'agent$_i$ pour qu'il ne soit pas en collision avec aucun autre agent situé[14] (ce qui correspond dans l'algorithme 7.1 à trouver $ReculeMax_i$ de l'agent$_i$). Nous notons que si $ReculeMax_i = Précision$, alors le robot revient à sa position initiale, et si $ReculeMax_i = 0$, alors ceci implique que la position future du robot$_i$ se fait sans aucune collision.

[14]Ceci peut se faire aussi en utilisant des méthodes analytiques telle que la dichotomie pour accélérer cette recherche.

Algorithme 7.1 *Moteur physique* implémenté dans *MiRoCo*

Entrée : int $Précision$ = const1 ; //Donne la précision de l'algorithme. Plus const1 est grande plus la précision de l'algorithme est plus importante.//

Pour chaque agent mobile dans l'environnement **Faire**
  $PositionInitiale_i = PositionAgent_i$; //Sauvegarder sa position initiale
  $PositionFinale_i = PositionInitiale_i + Vitesse_i \times \Delta t$; //Calcul de sa position finale. Avec $Vitesse_i$ correspondant à la vitesse de l'agent$_i$.//
  $PositionAgent_i = PositionFinale_i$; //Affectation de la position finale à l'agent en question
  $ReculeMax_i = 0$; //$ReculeMax_i$ correspond au nombre d'incréments en moins à faire par l'agent$_i$ par rapport à sa position finale pour ne pas être en collision avec d'autres agents.//
Fin Pour

//Obtention de $ReculeMax_i$ effective pour chaque agent$_i$.//
Pour tous les agents présents dans l'environnement **Faire**
  //NB : les indices $_i$ et $_j$ désignent deux agents distincts.
  int $Recule = 0$;
  Tant que agent$_i$ est en collision avec l'agent$_j$ **Faire**
    $Recule = Recule + 1$;
    $PositionAgent_i = PositionFinale_i$ - $[(Recule / Précision) \times Vitesse_i \times \Delta t]$;
    $PositionAgent_j = PositionFinale_j$ - $[(Recule / Précision) \times Vitesse_j \times \Delta t]$;
  Fin Tant que

  Si $Recule > ReculeMax_i$ **Alors**
    $ReculeMax_i = Recule$;
  Fin Si

  Si $Recule > ReculeMax_j$ **Alors**
    $ReculeMax_j = Recule$;
  Fin Si

  //Réaffection des positions finales des agents afin de faire d'autres tests de collision avec d'autres agents.
  $PositionAgent_i = PositionFinale_i$;
  $PositionAgent_j = PositionFinale_j$;
Fin Pour

//Obtention des positions finales des agents mobiles qui respectent d'une part, l'évolution parallèle (simultanée) des robots et d'autre part, la contrainte de non chevauchement des agents entre-eux.//

Pour tous les agents mobiles dans l'environnement **Faire**
  $PositionAgent_i = PositionFinale_i$ - $[(ReculeMax_i / Précision) \times Vitesse_i \times \Delta t]$;
Fin Pour

7.3.2.3.2 Modèle de déplacement de l'objet à pousser Les déplacements de l'objet à pousser dans le cas de la TCPO s'effectuent entièrement sous l'impulsion des actions exercées par les agents robots. Nous avons modélisé le déplacement de l'objet à pousser de telle sorte à ce que le modèle tienne compte des actions simultanées de

plusieurs robots à la fois (robots qui sont en interaction directe avec l'objet en question à l'instant t). Le modèle est résumé dans ce qui suit :

Le déplacement de l'objet à pousser entre les instants d'échantillonnage t et $t+\Delta t$ est conditionné, d'une part, par le nombre de robots en collision avec la boite "*Nrcb*" à l'instant t, et d'autre part, par la résultante de vitesse $\overrightarrow{V_b}$ engendrée par ces robots à l'instant t (cf. équation. 7.2). Ces deux contraintes se résument ainsi :

Si ($Nrcb < Nc$) **Alors** la boite à pousser reste immobile.

Sinon la boite se déplace en fonction de $\overrightarrow{V_b}$ (cf. figure. 7.10(a)), correspondant à l'application du principe de conservation de la *quantité de mouvement*[15] avant et après impacts pour le cas de collisions élastiques.

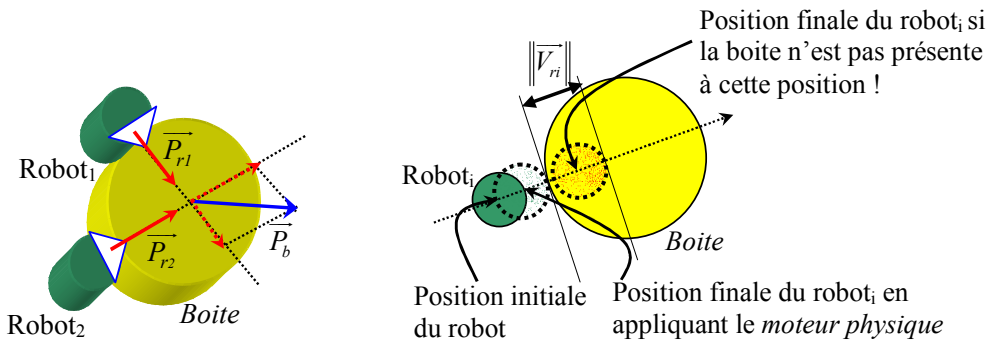
$$\overrightarrow{V_b} = \frac{Nrcb \times m_r}{M_b} \times \sum_{i=1}^{Nrcb} \overrightarrow{V}_{ri} \tag{7.2}$$

Avec :
- m_r masse d'un robot élémentaire,
- M_b masse de la boite à pousser,
- $\overrightarrow{V}_{ri}$ vitesse du robot$_i$ quand il rentre en collision avec la boite à pousser, telle que : $0 < \left\|\overrightarrow{V}_{ri}\right\| \leq (Vitesse_i \times \Delta t)$ (cf. figure. 7.10(b)).

Ainsi, à chaque pas d'échantillonnage, si le nombre de robots en collision avec la boite dépasse Nc, alors celle-ci va se déplacer à la vitesse instantanée $\overrightarrow{V_b}$, déterminée par les multiples collisions des robots avec elle. Dans les simulations effectuées sous *MiRoCo*, ce déplacement peut être entravé s'il y a au moins un agent qui gêne sa progression.

7.3.2.3.3 Modèle d'échange de signaux entre agents Un autre élément important modélisé sous *MiRoCo* concerne les différents signaux émis et/ou reçus par les agents présents dans l'environnement, en l'occurrence les robots, le ou les objet(s) à pousser et la cible à atteindre. Effectivement, l'objet à pousser, comme la cible à atteindre, dispose de balises émettrices caractéristiques qui permettent aux robots de les repérer et de les

[15] La conservation de la quantité de mouvement est un principe fondamental de la mécanique exposé par Isaac Newton en 1687 dans son célèbre ouvrage intitulé "Philosophiae naturalis principia mathematica", qui stipule que la quantité de mouvement d'un système isolé est constante. Dans le cas d'un ensemble de systèmes, et en l'absence de forces externes (systèmes isolés), la quantité de mouvement totale (égale à la somme de toutes les quantités de mouvement) se conserve également.



(a) Conservation de la quantité de mouvement.

(b) Obtention du module de la vitesse du robot au moment de l'impact avec la boite.

FIG. 7.10 – Modèle de déplacement de la boite

distinguer dans l'environnement. Les signaux altruistes constituent l'autre type de signaux utilisés par les robots pour une communication de très bas niveau entre-eux (cf. §4.3.1.7.1, page 98). Ces signaux altruistes doivent être modélisés de la manière la plus réaliste possible afin de permettre une analyse fiable de leur pertinence pour la réalisation des tâches coopératives entreprises.

Les points communs entre les différents types de signaux émis par les agents dans l'environnement sont :

- un champ d'émission (portée) déterminé en fonction de l'agent et/ou de la situation dont ils sont émis,
- une intensité décroissante à chaque fois qu'on s'éloigne de leur centre émetteur.

Dans ce qui va suivre, nous présentons les deux modélisations de signaux utilisées sous *MiRoCo*. Il est à noter que ces deux types de modélisation utilisent une structure qui fait office de tableau noir "*blackboard*" (Englemore & Morgan 1988), (Hayes-Roth 1984) afin de centraliser et de sauvegarder temporairement les informations utiles des signaux. Les robots n'ont alors qu'à émettre une requête vers cette structure pour pouvoir lire les informations locales qui les concernent.

1. **Signaux à décroissance d'intensité discrète :** Dans cette première modélisation, les signaux émis dans l'environnement sont représentés de manière discrète. C'est-à-dire que nous avons décomposé l'environnement en $(N_{li} \times N_{co})$ cases égales, où chaque case correspond à un élément du tableau TAB_s de N_{li} lignes et N_{co} colonnes. Chaque case du tableau contient autant de valeurs d'intensité de signaux

qu'il y a de types d'agents qui émettent des signaux couvrant cette case du tableau[16]. Il est à noter, qu'un signal sauvegardé dans la case du tableau garde une trace sur le type d'agent (robot, boite ou cible) qui a émis le signal. Nous notons aussi que l'intensité du signal est maximale dans la case du tableau contenant la position du centre de l'agent émetteur et elle diminue avec un facteur γ à chaque fois qu'on s'éloigne d'une case (dans toutes les directions) par rapport au centre de l'agent (cf. figure. 7.11).

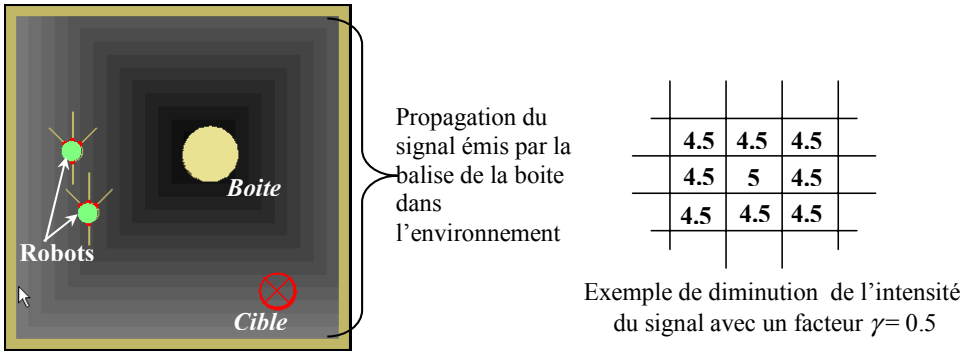


FIG. 7.11 – Émission de signaux à valeurs discrètes dans l'environnement

Ainsi, pour faire fonctionner le capteur de signaux "CS_i" (appartenant à un robot déterminé de la simulation), il n'a qu'à lire à la position (x_i, y_i) qu'il occupe dans l'environnement, la valeur du signal pour lequel il est dédié.

Cette version d'émission de signaux, qui utilise l'environnement comme médium de communication, est fortement inspirée des phéromones émises par les fourmis. Ces phéromones leur permettent entre autres de garder une trace du trajet à faire pour atteindre par exemple une source de nourriture. Cette trace est amplifiée par le passage répété des fourmis, qui sont de plus en plus nombreuses à emprunter ce chemin.

Inconvénients : Cette première approche de modélisation des échanges de signaux entre agents, s'avère être très coûteuse en temps de calcul et en mémoire. Ceci est accentué d'autant plus par le fait que nous voulons disposer d'informations très précises sur l'évolution des intensités des signaux à chaque fois qu'on s'éloigne de leur centre émetteur. En effet, ceci impose l'utilisation d'un tableau TAB_s de très grande dimension. L'autre point négatif de cette modélisation, est la lourdeur de rafraîchissement des cases de TAB_s à chaque pas d'échantillonnage. Ceci est dû

[16]La case en question contiendra s'il y a plusieurs signaux de même nature (i.e., émis par des agents identiques), la valeur de la somme des intensités des signaux émis par ces agents.

principalement aux déplacements des agents émetteurs de signaux et à la disparition de ces signaux dans l'environnement (car ils sont censés être volatiles).

L'implantation de ce type approche d'émission/réception de signaux en utilisant l'environnement comme médium de communication, est aussi techniquement difficile à maîtriser en robotique mobile, car ceci suppose de manipuler des substances chimiques, donc nécessite des capteurs et des récepteurs dédiés.

Tous les inconvénients cités ci-dessus nous ont amené à proposer une autre modélisation pour la gestion de l'émission/réception des signaux dans l'environnement.

2. **Signaux à décroissance d'intensité continue :**

   Dans cette seconde modélisation, nous avons opté pour une gestion des signaux émis et/ou perçus par les différents agents dans l'environnement, nettement moins coûteuse en temps de calcul et en espace mémoire.

   Nous établissons pour cela une liste de signaux présents dans l'environnement, avec pour chaque signal les caractéristiques décrites dans l'algorithme 7.2. Une représentation graphique de ces signaux dans l'environnement est donnée en figure 7.12.

Algorithme 7.2 Méthode d'émission de signaux continus

Entrée : Disposer l'agent$_i$ d'un émetteur de signaux ;
Si l'agent$_i$ doit émettre un signal **Alors**
Ajouter à la liste des signaux présents dans l'environnement un signal caractérisé comme suit :
- un centre "$Centre(x_i, y_i)$" correspondant à la position de l'agent$_i$ (cf. figure. 7.12(a)) ;
- un rayon "*Rayon*" déterminé par l'agent$_i$ (cf. figure. 7.12(a)) ;
- une intensité maximale "$IntensiteSignalCentre$" qui correspondra à l'intensité perçue au centre du signal ;
- une valeur de décroissance de la valeur absolue de l'intensité du signal "*DecroisanceSignal*", et ce quand on s'éloigne de son centre ;
- un identifiant "$Identifiant$" correspondant à celui de l'agent$_i$;

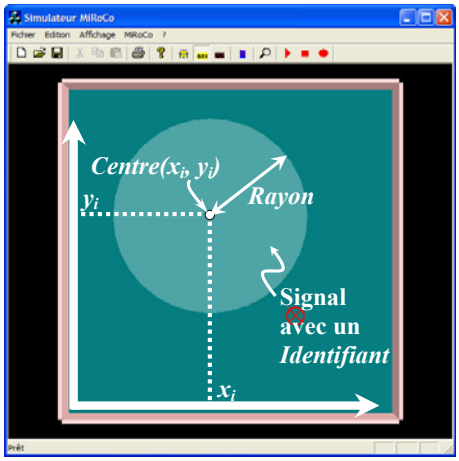
Fin Si

Pour ce qui est de l'algorithme 7.3, il décrit le mécanisme utilisé par les capteurs de signaux, afin de récupérer les bonnes valeurs d'intensité des signaux émis dans l'environnement.

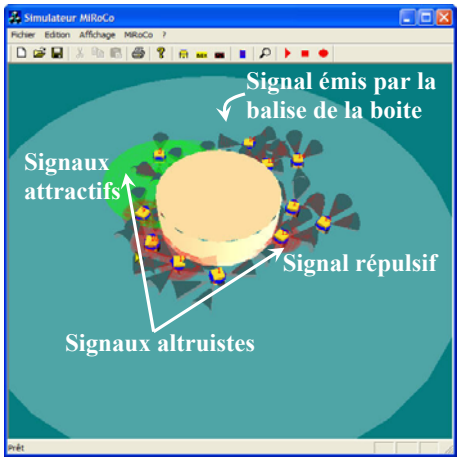
NB : La liste des signaux présents dans l'environnement est rafraîchie à chaque pas d'échantillonnage.

7.3.2.4 Classe Simulation

Cette classe contient les listes correspondant aux différents types d'agents présents dans l'environnement, en l'occurrence la liste des robots, des boites et des obstacles. Cette classe contient aussi une instance de la classe superviseur, et un agent cible. La



(a) Caractéristique d'un signal émis dans l'environnement.



(b) Émission de signaux de différents types dans l'environnement.

FIG. 7.12 – Signaux à décroissance continue

Algorithme 7.3 Méthode de réception des signaux continus

Entrée : Disposer l'agent$_i$ d'un ou plusieurs récepteurs de signaux ;
Pour tous les récepteurs de l'agent$_i$ **Faire**
  Pour tous les signaux présents dans l'environnement **Faire**
    Si ($PositionRcpteur$ - $CentreSignal$) $\leq$ $RayonSignal$ **Alors**
      Si $IntensiteSignalCentre > 0$ **Alors**
        //Dans le cas d'un signal altruiste attractif ou bien d'un signal émanant de la boite et ou de la cible
        $ValeurRecepteur = IntensiteSignalCentre$ - $DecroisanceSignal \times DistanceDuCentreDuSignal$;
      Sinon
        //Sinon c'est un signal répulsif
        $ValeurRecepteur = IntensiteSignalCentre + DecroisanceSignal \times DistanceDuCentreDuSignal$;
      Fin Si
    Fin Si
  Fin Pour
Fin Pour

largeur et longueur de l'environnement simulé, la hauteur et la largeur des murs sont aussi définies dans cette classe. Il faut donc au moins une instance de cette classe pour disposer de tous les éléments nécessaires à l'exécution d'une simulation sous *MiRoCo* (cf. figure. 7.4, page 161).

7.3.2.5 Bibliothèques externes utilisées

7.3.2.5.1 Environnement de programmation Nous avons développé *MiRoCo* en utilisant Visual C++ 6.0. Cette plate-forme de programmation pour C++ propose une architecture de programme, appelée document/vue basée sur les MFC "*Microsoft Foun-*

dation Class Library", qui consiste à séparer les données (document) de leur visualisation (vue). L'idée générale a été alors d'intégrer les classes modélisant le système multi-robots, à l'ossature standard d'une architecture document/vue et ce afin de bénéficier des fonctionnalités des MFC (Horton 1999).

7.3.2.5.2 Affichage L'aspect affichage graphique de l'environnement multi-robots de *MiRoCo* est complètement géré en utilisant la bibliothèque graphique OpenGL "*Open Graphics Library*", ce qui nous a permis de bénéficier de ses différentes potentialités. Parmi elles, nous pouvons citer, le choix des angles de vues "*ViewPoint*" et de leur nombre pour visualiser la scène. Ainsi, nous pouvons par exemple disposer chaque robot d'une caméra individuelle, qui lui permettra (via un traitement d'images approprié) de retirer des informations pertinentes sur son environnement (cf. figure. 7.13).

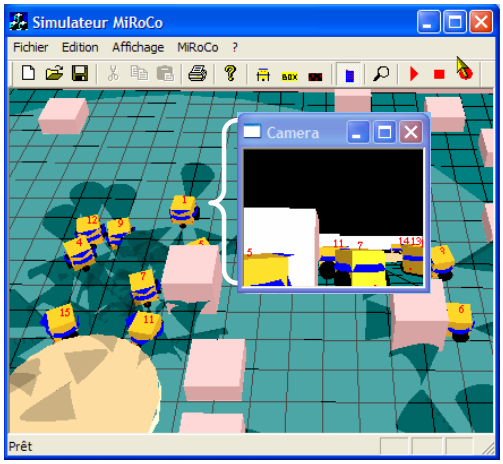


FIG. 7.13 – Angle de vue à partir de la position du robot numéro 1

La scène d'évolution des robots peut être composée de plusieurs volumes élémentaires (cube, sphère, etc.), ou de formes complexes (qui peuvent être construites par accumulation de formes de base). Ces formes pouvant être colorées, texturées, avoir une apparence d'un matériau quelconque, etc. OpenGL offre donc un grand nombre de fonctionnalités d'affichage, rendant ainsi le rendu des scènes (contenant tous les éléments participant aux simulations : robots, boites, obstacles, signaux émis par les agents, etc.) à observer très réaliste. Ceci nous permet par conséquent d'avoir des indications très précises sur la dynamique d'interaction et d'évolution du système multi-robots simulé.

7.3.2.5.3 Détection de collisions Il est nécessaire de disposer d'outils qui détectent automatiquement les collisions éventuelles entre les différents agents situés dans l'environ-

nement. Ce besoin nous a conduit à rechercher une bibliothèque appropriée de détection des collisions (Reggiani et al. 2002). Nous avons opté finalement pour la bibliothèque SOLID[17] (Bergen 2004) qui est diffusée sous une licence "*GNU Library General Public Licence*". Les différents agents qui peuvent être testés grâce à SOLID, peuvent avoir des formes élémentaires diverses telles que des cubes, sphères, cylindres, cônes, etc. Mais ils peuvent aussi être construits par une adjonction de formes élémentaires afin de produire des agents aux formes complexes, comme le cas d'un robot par exemple.

7.3.2.6 Déroulement d'une simulation complète pour le cas de la *TCPO*

Le déroulement des simulations sous *MiRoCo* pour la cas de la TCPO est résumé par les étapes décrites dans l'algorithme 7.4.

Algorithme 7.4 Étapes nécessaires pour l'exécution de la TCPO sous *MiRoCo*

Début de la simulation multi-robots : Donner une configuration initiale à tous les agents présents dans l'environnement ;
Tant que la simulation n'est par terminée **Faire**
- Rafraîchissement les signaux présents dans l'environnement ;
- Déplacement de chaque robot dans l'environnement en fonction, d'une part des stimuli locaux qu'il perçoit et d'autre part de l'architecture de contrôle qu'il utilise ;
- Application du *moteur physique* pour avoir des déplacements cohérents et physiquement réalisables ;
- Déplacement éventuel de la boite si les conditions nécessaires à son déplacement sont vérifiées ;
- Affichage de la scène avec les différents agents ;

Fin Tant que

7.4 Conclusion

La conception modulaire du simulateur *MiRoCo*, nous a permis de maîtriser la complexité inhérente aux simulations de systèmes multi-robots. Les différentes modélisations implémentées dans *MiRoCo* nous ont permis d'obtenir une précision importante par rapport à l'évolution de la dynamique du système multi-robots. *MiRoCo* constitue à présent un outil complet pour simuler des systèmes multi-robots.

Dans le cadre des simulations effectuées aux chapitres IV, V et VI, *MiRoCo* s'est avéré un outil précieux pour étudier et quantifier d'une part, les caractéristiques des architectures de contrôle implémentées sur les robots et d'autre part, les phénomènes et comportements émergents induits par le coopération des robots.

[17] `http://www.win.tue.nl/~gino/solid/`

Chapitre 8

Phase expérimentale

Résumé : *L'un des principaux objectifs de ce chapitre est d'étudier les différentes possibilités de transposition des éléments de contrôle, de conception et d'analyse proposés et traités jusqu'à présent en simulation, vers une implantation matérielle effective, et ceci en tenant compte des contraintes matérielles, logicielles et expérimentales. Après une description succincte de la plate-forme de robotique collective disponible actuellement au LAB, nous nous focaliserons sur l'implémentation effective, d'une part des comportements élémentaires proposés au chapitre IV pour le cas de la TCPO, et d'autre part sur les tests de viabilité des processus de coordination entre comportements proposés aux chapitres IV et V, en l'occurrence le PSAH et le PSAHH respectivement.*

8.1 Plate-forme expérimentale

Afin de mener à bien nos phases d'expérimentation, nous avons mis en place une plate-forme expérimentale (cf. figure. 8.1) dédiée à l'étude et à l'analyse des comportements collectifs d'un groupe de mini-robots autonomes ALICE (cf. §8.1.3).

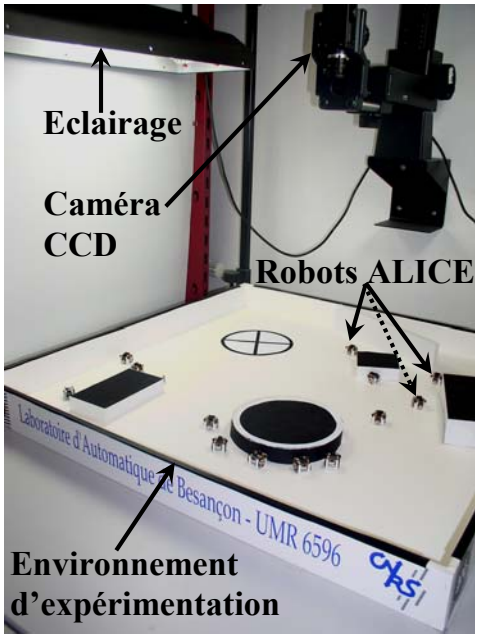


FIG. 8.1 – Plate-forme expérimentale

Les principaux éléments composants cette plate-forme sont décrits dans ce qui suit.

8.1.1 L'environnement

L'environnement choisi pour les expérimentations est une enceinte carrée d'une dimension de 80cm×80cm, qui représente tout de même 1600 fois la surface d'un mini-robot ALICE. Ce rapport entre les dimensions de l'environnement et celles des ALICE suffit amplement à mettre en scène et à contrôler un grand nombre de mini-robots.

8.1.2 L'aspect vision

La nécessité d'un outil qui nous permette de quantifier d'une manière précise et surtout automatique les expérimentations effectuées, nous a poussé à doter la plate-forme

expérimentale d'une chaîne d'acquisition vidéo[1] (cf. figure. 8.2). Cette chaîne d'acquisition, jumelée à l'outil de *tracking* développé (Velghe 2003), nous permet d'acquérir avec précision :

- d'une part, les conditions initiales des expérimentations (cf. figure. 8.3(a)) telles que : le nombre, les positions et les orientations initiales de tous les éléments caractéristiques de l'environnement en l'occurrence les robots, les objets à pousser, les obstacles, les murs, la cible,
- d'autre part, le *tracking* de l'évolution (positions, orientations) des différents éléments mobiles dans l'environnement en fonction du temps (cf. figure. 8.3(b)).

L'outil de *tracking* développé fonctionne en utilisant un seuillage approprié ainsi que plusieurs motifs distinctifs placés sur les différents éléments présents dans l'environnement. Ce *tracking* nous permet un traitement en temps-réel d'une quinzaine d'éléments mobiles dans l'environnement et pourrait éventuellement être utilisé pour un contrôle centralisé des mini-robots ALICE.

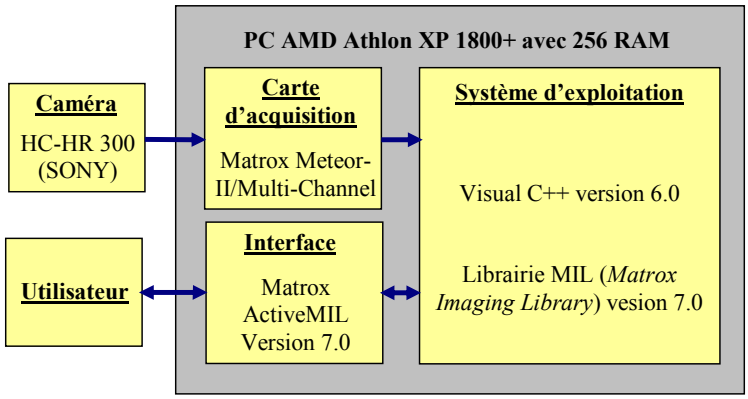


FIG. 8.2 – Chaîne d'acquisition et de traitement vidéo utilisée

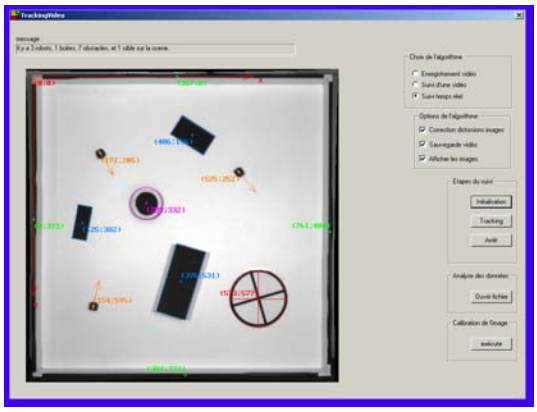
Les expérimentations effectuées sont sauvegardées sur un fichier vidéo et sur un fichier XML[2] dédié (cf. figure. 8.3(c)), et ce pour permettre des analyses post-expérimentales.

L'éclairage de l'environnement est assuré par quatre néons de 24 watts qui diffusent une lumière de type *lumière du jour* (cf. figure. 8.1). Cette lumière permet d'une part, un éclairage uniforme de l'environnement pour un meilleur *tracking* vidéo, et d'autre part,

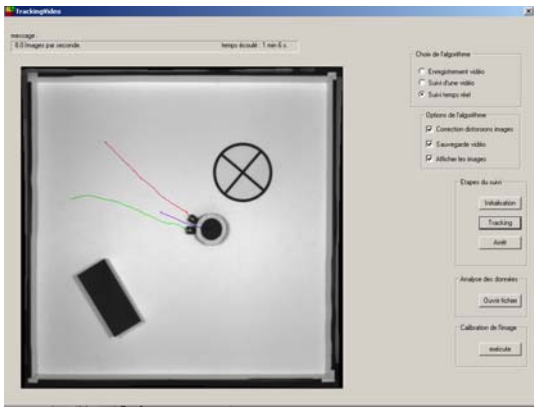
[1] Les autres buts visés de cette chaîne d'acquisition vidéo sont de pouvoir comparer, valider et vérifier la viabilité des architectures de contrôle proposées, et ce grâce entre autres à une analyse plus fine des interactions existantes entre robots lors des expérimentations.

[2] *eXtended Markup Language*.

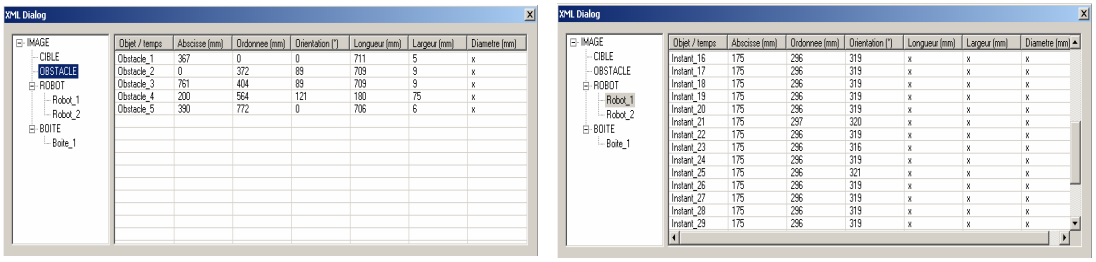
une émission réduite d'infrarouges, qui risquerait si elle est trop importante, de parasiter les capteurs infrarouges propres aux mini-robots ALICE.



(a) Acquisition des conditions initiales des expérimentations.



(b) *Tracking* vidéo des éléments mobiles dans la scène, application à la TCPO.



(c) Sauvegarde des informations relatives aux expérimentations sous format XML

FIG. 8.3 – Outil de *tracking* utilisé durant les expérimentations

8.1.3 Les mini-robots ALICE

Dans la perspective de pouvoir implémenter une architecture de contrôle complètement distribuée et réactive sur un grand nombre de robots minimalistes (autant d'un point de vue structurel que décisionnel), notre choix s'est vite porté sur le mini-robot ALICE (cf. figure. 8.4) (Caprari 2003). ALICE offre à la fois une autonomie énergétique et décisionnelle très appréciable relativement à ses dimensions qui ne sont que de 22mm×20mm×20mm. De plus, ce mini-robot permet des expérimentations avec un grand nombre de robots dans des environnement très confinés, et tout cela en gardant des coûts d'investissement et d'exploitation relativement faibles.

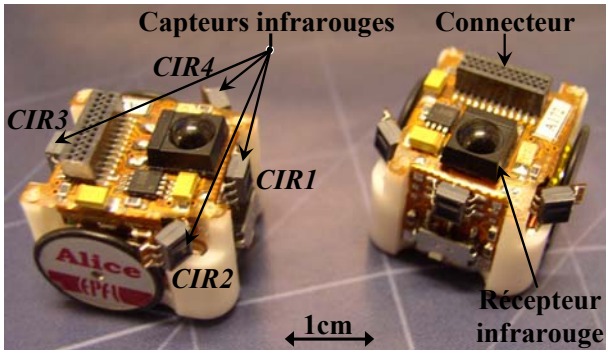


FIG. 8.4 – Mini-robots ALICE

Le développement du mini-robot ALICE par Gilles Caprari durant ses travaux de thèse à l'EPFL[3] était un bon challenge à la fois technologique et conceptuel. En effet, malgré son petit volume, inférieur à 9cm^3, Gilles Caprari a su obtenir une mini-structure robotique mobile et autonome performante. L'autonomie énergétique d'ALICE est de l'ordre de 10 heures, et est assurée grâce à une batterie rechargeable. Quant à son autonomie décisionnelle, elle est obtenue grâce à un PIC16F877 (microcontrôleur à 8 bits de chez MICROCHIP) cadencé à une fréquence d'horloge de 4 MHz et qui dispose d'une mémoire de programme flash EPROM de 8 Kmots. Ceci est bien évidemment loin des puissances de calcul dont sont dotés d'autres robots mobiles sur lesquels sont embarqués des ordinateurs complets à forte puissance de calcul. La structure de base d'ALICE comprend aussi : deux moteurs de montre de type Lavet (moteur pas à pas pouvant tourner dans les deux sens) qui permettent au mini-robot d'avoir une vitesse maximale de 40mm/s (Url-actionneur 2005) ; un connecteur pour interfacer d'autres étages capteurs ou de communication ; un récepteur infrarouge pour télécommande ; quatre capteurs infrarouges de proximité *CIR1*, *CIR2*, *CIR3*, *CIR4* avec les caractéristiques suivantes (Url-capteurIR 2005) :

- ils sont positionnés respectivement à 0°, 45°, 180° et -45° par rapport à la face avant du mini-robot,
- ils ont un champ d'émission de forme conique avec un angle d'ouverture approximativement égal à 60°,
- ils peuvent détecter des obstacles jusqu'à une distance maximale de 4cm. Ceci restreint donc ALICE à des informations très localisées.

[3]École Polytechnique Fédérale de Lausanne.

8.2 Implémentation effective des architectures de contrôle proposées

Après avoir simulé d'une manière intensive et exhaustive les différentes architectures de contrôle proposées aux chapitres IV et V[4], le passage à une validation en situation réelle reste un objectif primordial à satisfaire. Ce passage ne nous donnera certainement pas autant d'indications statistiques sur l'évolution de la dynamique du système multi-robots que celles obtenues avec le simulateur *MiRoCo*[5]. Cependant, le fait de pourvoir implémenter ce qui à été proposé aux chapitres IV et V sur des robots réels, va nous permettre d'avoir une indication précise sur l'adéquation des mécanismes et des stratégies de contrôle proposés pour le cas du contrôle et de la coopération d'un groupe de robots minimalistes.

Nous rappelons que les architectures de contrôle proposées aux chapitres IV et V pour le cas de la TCPO, sont composées des comportements élémentaires suivants : l'*alignement*, la *poussée de boite*, le *repositionnement*, l'*attraction à la boite*, l'*évitement d'obstacles*, les comportements altruistes d'*émission de signaux altruistes* et de *réponse aux signaux altruistes* et finalement le comportement d'*exploration*.

Ces comportements élémentaires fonctionnent selon des mécanismes à base de *stimuli-réponse* (cf. §4.3.1, page 91) et leur implémentation sur une structure robotique (le microcontrôleur d'ALICE) exige de tenir compte à la fois d'aspects matériels et logiciels. C'est deux aspects sont traités dans les deux sections qui suivent.

8.2.1 Aspects matériels

Avant d'implanter la stratégie de stimuli-réponse propre à chacun des comportements élémentaires cités ci-dessus, il faut au préalable parvenir à récupérer de l'environnement les stimuli qui assurent leur fonctionnement. Vu le caractère complètement distribué souhaité du contrôle des robots, ces derniers doivent pouvoir disposer d'une autonomie perceptuelle complète (cf. §1.1.3, page 11). Néanmoins, la structure de base d'ALICE ne permet pas de récupérer de l'environnement les stimuli nécessaires au fonctionnement de certains comportements élémentaires.

[4] Intensive grâce aux simulations effectuées en mode "*batch*" et exhaustive vu les différents tests effectués sur plusieurs éléments influençant étroitement le fonctionnement des architectures de contrôles proposées (e.g., l'utilisation ou non des comportements altruistes, valeurs des gains à affecter aux blocs de fusions du PSAHH).

[5] Vu entre autres les différents dysfonctionnements ou pannes que pourraient subir les robots durant les expérimentations ; au nombre limité de robots en notre possession ; au temps imparti à chaque phase expérimentale.

Les stimuli manquants ainsi que les comportements élémentaires dont ils dépendent sont résumés dans ce qui suit :

- la détection d'angles relatifs que fait le robot, d'une part, avec l'objet à pousser et d'autre part, avec la cible à atteindre (cf. figure. 4.13, page 97). Ces stimuli sont nécessaires au fonctionnement des comportements élémentaires suivants : l'*alignement*, la *poussée de boite*, le *repositionnement* et finalement l'*émission de signaux altruistes*,
- l'émission/réception de signaux de bas niveau pour réaliser les comportements altruistes d'*émission de signaux altruistes* et celui de *réponse aux signaux altruistes*,
- la détection du contact direct entre le robot et la boite à pousser et non pas avec un autre élément présent dans l'environnement, tel un robot, un obstacle ou un mur. Ce stimulus est nécessaire au fonctionnement des comportements élémentaires suivants : l'*alignement*, la *poussée de boite* et le *repositionnement*.

Les perceptions manquantes pour le fonctionnement des architectures de contrôle proposées pour la cas de la TCPO, nous ont donc amené à prospecter et à étudier les différentes possibilités d'intégration de nouveaux étages de capteurs et de communication sur la structure originelle d'ALICE.

Malgré la conception très modulaire d'ALICE (Caprari & Siegwart 2003) qui prévoit un interfaçage aisé d'autres cartes électroniques à sa structure de base (tel un étage capteur ou de communication), nous avons rencontré des difficultés pour intégrer les éléments manquants. Ceci est dû principalement au fait que ALICE est, à la base, dimensionné et optimisé afin d'aboutir à une miniaturisation importante, ce qui implique donc que les éléments (capteurs, émetteurs, actionneurs, etc.) qui peuvent être intégrés, doivent respecter un certain nombre de contraintes, telles que :

- l'encombrement (dimension, poids) minimum à l'échelle des dimensions d'ALICE,
- la basse consommation d'énergie à l'image des différents éléments (moteurs, capteurs infrarouges, ...) constituant ALICE. Ceci est directement lié aux possibilités d'alimentation de la batterie d'accumulateur disponible sur ALICE,
- la disponibilité de broches libres sur le microcontrôleur d'ALICE (le PIC16F877). En effet, l'intégration de nouveaux composants passe inéluctablement par son interfaçage avec le PIC16F877 d'ALICE.

Les points cités ci-dessus ont donc grandement contraint nos investigations pour déterminer les capteurs appropriés qui puissent être adjoints à la structure de base d'ALICE. D'autant plus que ces contraintes éliminent toute possibilité d'intégration de

capteur sophistiqué demandant de grandes ressources de fonctionnement. Ceci nous à donc amené à contourner le problème quant à la sophistication des moyens que nous pouvons mettre en oeuvre, en utilisant deux voies :

1. l'inspiration du fonctionnement de certains animaux,
2. l'exploitation de certaines caractéristiques intrinsèques des éléments présents dans l'environnement (e.g., leur hauteur, couleur, type de matériaux).

En procédant ainsi, plusieurs possibilités de récupération des stimuli manquants s'offrent à nous. Nous citons dans ce qui suit, celles qui nous semblent les plus pertinentes et celles que nous avons retenues.

8.2.1.1 Dispositif pour la mesure d'angles

Le but de ce dispositif est de pouvoir déterminer les angles relatifs θ_1, θ_2 que le robot fait avec deux points caractéristiques dans l'environnement. En l'occurrence pour le cas de la TCPO, θ_1 et θ_2 correspondent respectivement aux angles que fait le robot avec l'objet à pousser et avec la cible à atteindre (cf. figure. 4.13, page 97).

Une des solutions pour caractériser ces deux points dans l'environnement est de disposer sur chacun d'eux, une balise émettrice caractéristique. Afin d'éviter d'éventuelles interférences ou post-traitements importants sur les signaux perçus par les robots, il est fortement souhaitable que ces deux balises émettent des signaux de nature différente, tels que des ultrasons versus des ondes radio. Ceci exige bien évidemment d'intégrer à ALICE des capteurs, d'une part sensible aux signaux émis par les balises émettrices, et d'autre part, qui puissent déduire les orientations relatives du robot par rapport à ces balises.

Concernant la distinction entre les deux balises émettrices, plusieurs combinaisons de capteurs ont été envisagées (capteurs infrarouges versus à ultraviolet, capteurs infrarouges versus magnétiques, etc.). Cependant, à chaque fois la solution a été abandonnée faute : d'encombrement excessif, de coût important, d'impossibilité technique, etc. (Baud 2004).

Pour ce qui est des méthodes possibles pour déduire ces angles relatifs, là aussi plusieurs solutions ont été envisagées. Parmi elles, nous pouvons citer celle qui consiste à placer un capteur de lumière sur l'axe d'un moteur de montre afin qu'il puisse tourner tel un radar. L'idée est de pourvoir obtenir la direction où l'intensité du signal perçu est maximale, car cette valeur maximale n'est censée être récupérée que lorsque le capteur se trouve effectivement de face par rapport à la balise émettrice. Les inconvénients majeurs de cette solution résident dans l'encombrement et la consommation énergétique relativement importants du moteur à intégrer sur ALICE.

Finalement les dispositifs de mesure d'angles retenus, et qui ont fait l'objet d'une implantation matérielle sur le mini-robot ALICE, sont d'une part un anneau de capteurs de lumière pour le cas de l'objet à pousser, et d'autre part un capteur de type boussole numérique pour le cas de la cible.

1. **Anneau de capteurs de lumière :** L'utilisation d'un anneau de capteurs de lumière (cf. figure. 8.5) pour déterminer l'angle relatif que fait le robot avec l'objet à pousser, consiste à exploiter les différentes intensités lumineuses relevées par l'anneau de capteurs afin de déterminer la direction de la balise émettrice[6] (une simple lampe électrique).

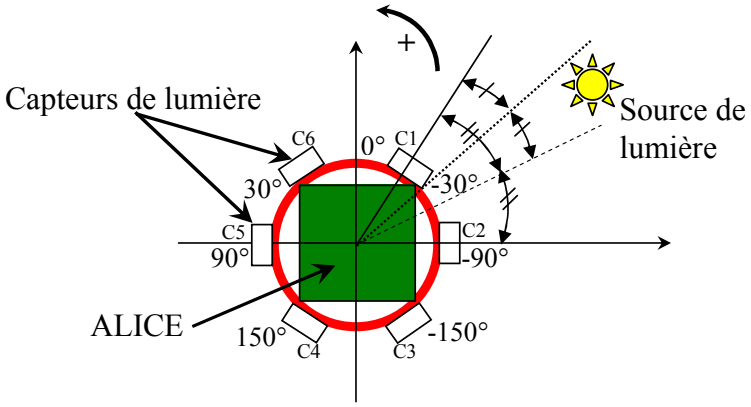


FIG. 8.5 – Anneau de six capteurs de lumière

Comme cela est indiqué sur la figure 8.5, la solution la plus simpliste consiste à déterminer les deux capteurs qui relèvent les intensités du signal les plus importantes, en l'occurrence dans la figure 8.5 ceci correspond aux capteurs positionnés à -30° et à -90° par rapport à la face avant du robot (i.e., respectivement les capteurs C1 et C2). Ceci permet d'affirmer si tous les capteurs sont bien uniformes et que l'environnement n'est pas trop bruité, que la source de lumière est entre -30° et -90°. Nous pouvons affiner davantage cette mesure d'angle en exploitant aussi la valeur maximale entre les deux capteurs C1 et C2. En effet, si c'est C1 qui relève la valeur maximale, alors ceci nous permet d'affirmer que la source de lumière sera plus près de C1 que de C2 et inversement. Dans la figure 8.5, c'est C1 qui relève l'intensité maximale alors ceci se traduit par le fait que la source de lumière est entre -30°

[6] La comparaison de plusieurs valeurs d'intensités de stimuli afin de s'orienter est largement utilisée chez les animaux. Les abeilles, par exemple, remontent un gradient chimique en utilisant la direction indiquée par leur antenne la plus stimulée.

et -60°. Par la suite, nous faisons une simple approximation grossière mais suffisante qui stipule que la source lumineuse est au milieu de l'intervalle [-30°, -60°], c'est-à-dire à un angle relatif de -45°. L'algorithme décrit ci-dessus au travers d'un exemple, tolère donc dans le cas d'un anneau à 6 capteurs, une incertitude égale à ± 15°.

La relation générale reliant l'incertitude maximale $\Delta\theta$ que l'on peut obtenir au nombre de capteurs présents sur l'anneau "$NCapteurs$" (capteurs qui doivent être positionnés uniformément autour du robot), est la suivante.

$$|\Delta\theta| = \frac{360}{4 \times NCapteurs} \tag{8.1}$$

La solution d'un anneau de capteurs est celle qui nous semble la plus simple à mettre en oeuvre et surtout la moins coûteuse en temps de traitement des informations perçues. Les tests préalables de cet anneau de capteurs (cf. figure. 8.6) ont consisté à utiliser 6 photorésistances[7] qui disposent chacune d'un champ d'ouverture d'environ 180°, ce qui permet à l'anneau de capteurs de couvrir largement le plan. Nous précisons que nous avons orienté les photorésistances de 60° par rapport à l'horizontale de façon à éviter au maximum de les soumettre aux réflexions parasites des signaux lumineux avec le sol ou avec tout autre type d'élément présent dans l'environnement.

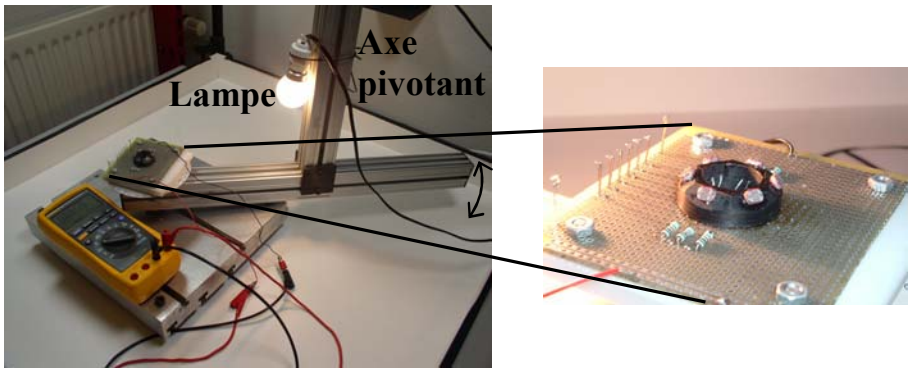
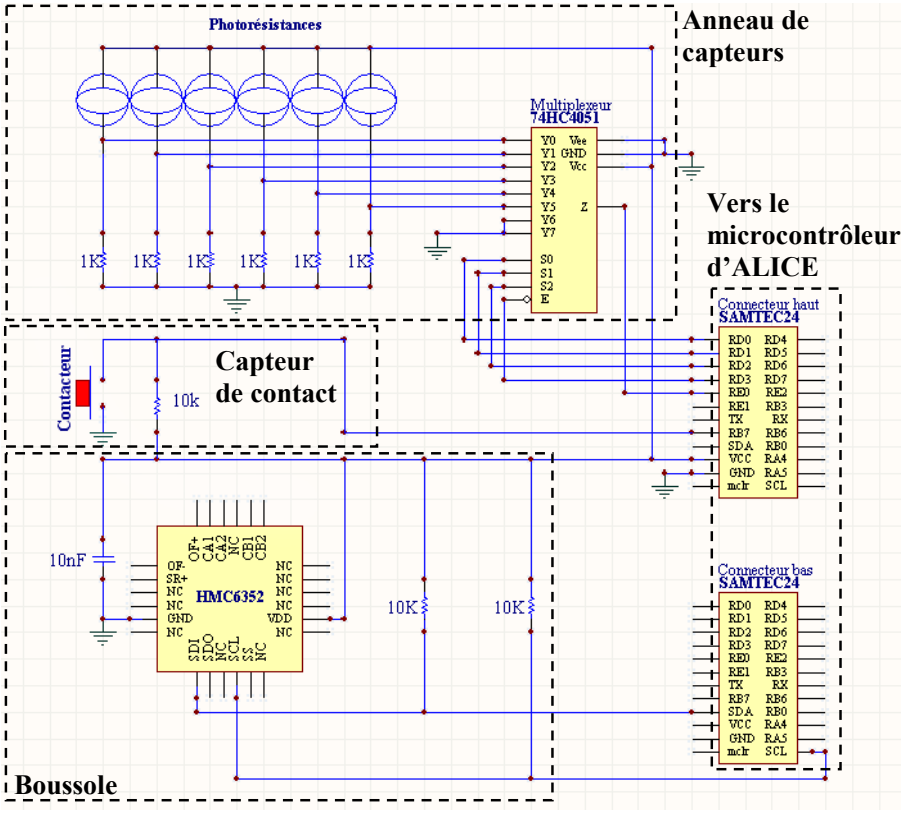


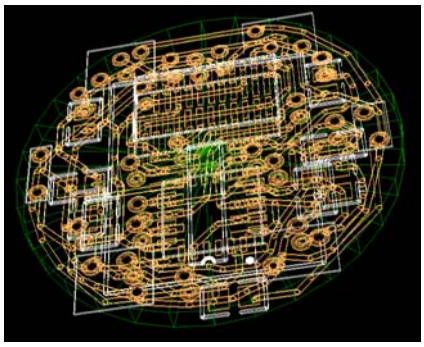
FIG. 8.6 – Plate-forme de tests pour l'anneau de capteurs

L'implantion effective de cet anneau de capteurs sur le mini-robot ALICE s'est effectuée via le développement d'une carte électronique qui contient à la fois l'anneau de capteurs, une boussole numérique et un capteur de contact (cf. figure. 8.7).

[7] Capteur qui dispose d'une résistance variable en fonction de l'intensité lumineuse perçue.



(a) Schéma électronique de l'étage capteur.



(b) Schéma PROTEL de la carte électronique double face développée.

FIG. 8.7 – Carte électronique développée.

La figure 8.8 représente l'étage capteurs réalisé intégrant l'anneau de photorésistances opérationnel. Les tests effectués sur cet anneau de capteurs ont consisté à faire tourner une source lumineuse autour de lui et à relever à intervalles réguliers, les valeurs des tensions aux bornes de ses six photorésistances (cf. figure. 8.9).

Nous remarquons que les tensions obtenues correspondent à nos attentes. En effet, la plage de sensibilité des capteurs ainsi que la forme de leurs réponses à la présence de la source de lumière restent approximativement identiques.

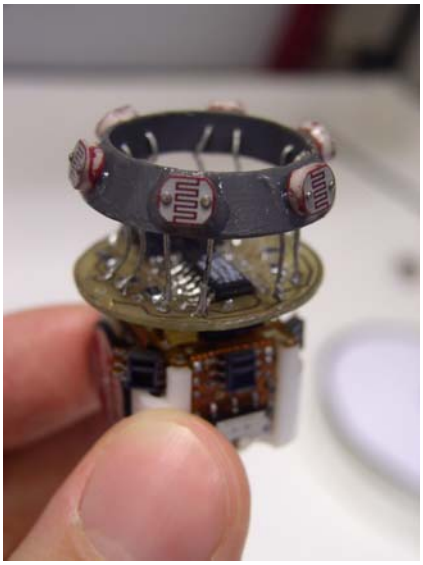


FIG. 8.8 – Étage capteurs ajouté à ALICE

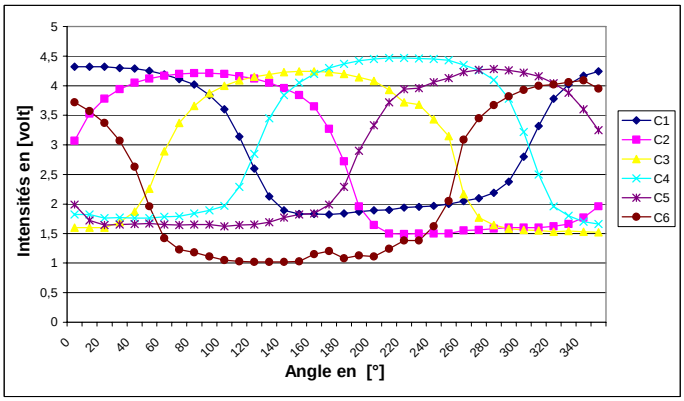
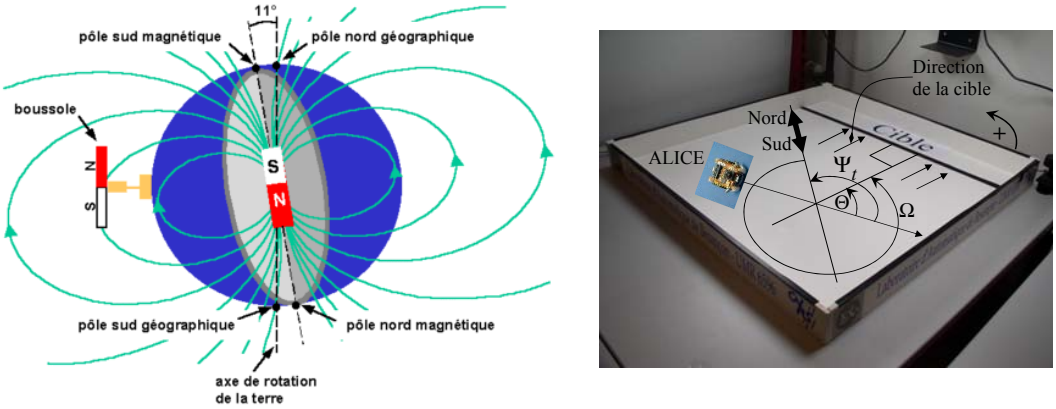


FIG. 8.9 – Tensions relevées aux bornes des photorésistances de l'anneau de capteurs

2. **La boussole numérique :** Le dispositif que nous avons prévu pour indiquer à ALICE son orientation relative par rapport à la cible, consiste à utiliser une boussole numérique qui lui permettra de connaître à chaque instant "t", l'orientation relative Ψ_t du nord magnétique terrestre (cf. figure. 8.10). Ce nord magnétique constituera donc une forme de repère absolu partagé par tous les robots réalisant

la tâche coopérative. Les robots connaissant à l'avance (via leur programme de contrôle) l'orientation effective Ω de la cible par rapport au nord magnétique terrestre, sont ainsi capables de déduire l'angle relatif Θ qu'ils font avec la cible, tel que $\Theta = \Psi_t + \Omega$.



(a) Champ magnétique terrestre (b) Obtention de l'angle relatif Θ

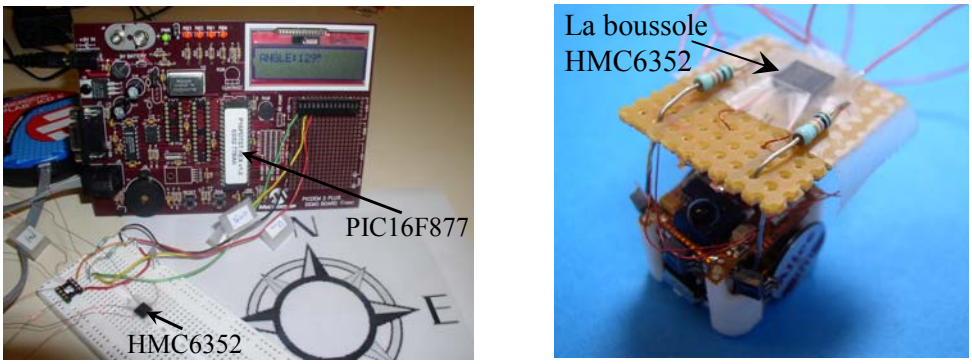
FIG. 8.10 – Utilisation du nord magnétique terrestre comme repère absolu pour orienter les robots.

Nous notons que l'approche décrite ci-dessus s'inspire des moyens qu'ont certains animaux pour s'orienter dans leur environnement. Nous pouvons citer par exemple les pigeons, qui utilisent une boussole biologique lors de leur longue migration. Ce qui est notable aussi par rapport à la méthodologie que nous voulons entreprendre, est le fait que la cible à atteindre ne sera pas caractérisée par un point précis dans l'environnement mais plutôt par une direction fixe dans l'environnement.

Pour ce qui est de l'implémentation effective d'une boussole numérique sur ALICE, nous avons pu trouver dans le commerce le composant HMC6352 de chez HONEYWELL qui correspond parfaitement à nos exigences d'intégration. Les principales caractéristiques du HMC6352 sont les suivantes : une dimension réduite de 6.5mm×6.5mm×1.5mm ; une précision de l'ordre de $\pm 5^\circ$; une résolution de l'ordre de 0.5° ; une facilité d'interfaçage avec le PIC16F877 d'ALICE et cela via une communication sérielle par I^2C "*Inter Integrated Circuit*" ; des routines de calibration intégrées.

Les délais de livraison importants du HMC6352 ont fait que son intégration sur ALICE a pris du retard. Néanmoins, nous avons pu effectuer des tests concluants de cette boussole en l'interfaçant dans un premier temps avec un banc d'essais

dédié aux microcontrôleurs PIC (cf. figure. 8.11(a)), et dans un deuxième temps nous l'avons testé sur l'ALICE même via une carte électronique d'essais (cf. figure. 8.11(b)) en attendant son intégration effective sur l'étage capteur dédié (cf. figure. 8.8). Nous avons pu ainsi vérifier la fiabilité de cette boussole pour relever l'orientation du nord magnétique terrestre et ce malgré les champ électrique parasite présent dans l'environnement d'expérimentation.



(a) Banc d'essais utilisé (b) La boussole interfacée à ALICE

FIG. 8.11 – Tests effectués pour l'intégration de la boussole numérique à ALICE

8.2.1.2 Dispositif pour les comportements altruistes

L'émission et la réception des signaux altruistes (cf. §4.3.1.7, page 98) sont l'un des moyens qui améliore de façon significative les performances de la tâche coopérative de poussée d'objets. Pour réaliser ces fonctions, nous avons choisi d'adapter le protocole de communication proposé par Gilles Caprari (Url-communication 2005) à nos besoins. Ce protocole consiste à utiliser les capteurs infrarouges de proximité disponibles sur ALICE afin d'établir une communication de bas niveau entre robots.

La séquentialité d'utilisation des capteurs infrarouges de proximité disponible sur ALICE est décrite sur la figure 8.12. En effet, la gestion des différents éléments capteurs, de communication et de l'actionnement d'ALICE se fait d'une manière séquentielle et cadencer par l'intermédiaire des interruptions générées par un Timer du PIC16F877 qui surviennent chaque 200μs (Url-programmation 2005). La gestion des capteurs infrarouges de proximité se fait chaque 1ms (5 interruption Timer). La période nécessaire pour pouvoir à la fois : communiquer avec d'autres ALICE ; relever les intensités lumineuses de l'environnement et mesurer les distances séparant le robot des obstacles se fait chaque 50ms. Ceci est satisfaisant dans le cadre du mini-robot ALICE vue sa vitesse relativement faible de 40mm/s. En effet, ceci implique que le rafraîchissement des informations

parvenant des capteurs de proximité se fait à chaque fois qu'ALICE avance au maximum de 2mm.

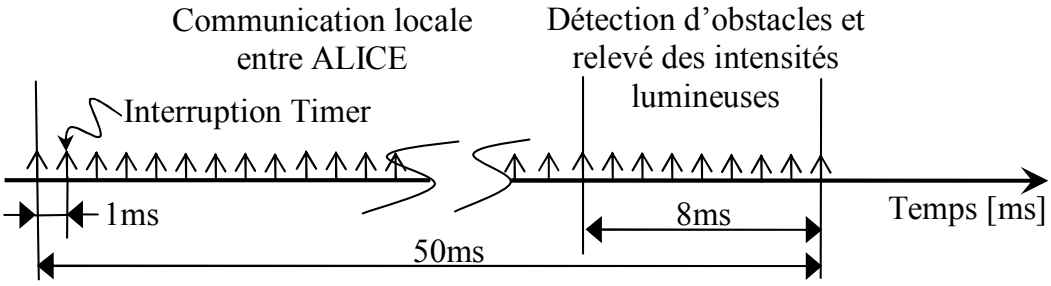


FIG. 8.12 – Séquentialité de l'utilisation des capteurs infrarouges de proximité

Les caractéristiques du protocole de communication entre ALICE sont les suivantes :

- les messages échangés entre robots contiennent deux bits *start*, une information (*valeur*) sur 4 bits, 2 bits pour indiquer l'identifiant de l'émetteur de la communication et 2 bits *stop* (cf. figure. 8.13).

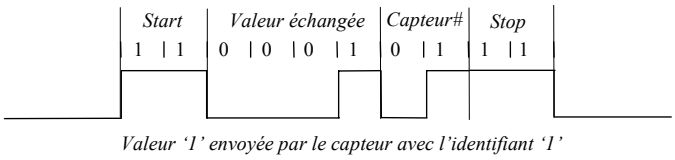


FIG. 8.13 – Type d'informations transmises entre robots

- dans le cas où deux robots veulent envoyer un message au même moment, alors dans ce cas de figure, le premier robot qui détectera le début d'une communication va se mettre en *esclave* pour recevoir le message, il reportera ainsi l'émission de son message jusqu'à ce qu'il devienne le *maître*,
- le robot esclave (récepteur) récupère le message transmis par l'intermédiaire d'un seul capteur de proximité à la fois.

Nous avons utilisé le protocole de communication décrit ci-dessus pour transmettre une information de répulsion et/ou d'attraction telle que si la *valeur* reçue $\in [1, 7]$ alors l'information est interprétée comme une répulsion et si cette *valeur* $\in [8, 14]$ alors c'est une attraction. Les différents tests effectués avec ce protocole de communication nous ont permis d'implémenter les comportements altruistes d'une manière satisfaisante.

Nous notons que la fiabilité des communications échangées entre ALICE sont de l'ordre de 95%[8] ce qui représente globalement de bon résultats.

8.2.1.3 Dispositif pour la détection de l'objet à pousser

La réalisation de la tâche coopérative de poussée d'objets nécessite de pouvoir distinguer l'objet à pousser des autres éléments présents dans l'environnement (e.g., robots, obstacles, murs). Plusieurs solutions s'offrent à nous, parmi elles, celles qui consistent à fixer les caractéristiques intrinsèques de l'objet à pousser, telles que sa hauteur, sa couleur ou sa forme afin qu'il soit facilement détectable par les ALICE.

Nous avons choisi finalement de fixer la hauteur de l'objet à pousser de telle sorte qu'elle soit la plus élevée des éléments présents dans l'environnement. Ainsi, un simple capteur de contact posé sur ALICE en hauteur pourrait déterminer d'une manière quasi certaine la présence de l'objet à pousser. Ce capteur de contact déclenche une interruption dès qu'une force est exercée dessus (contacteur fermé (cf. figure. 8.7(a))). Nous avons testé ce contacteur à l'aide de d'un montage simple (cf. figure. 8.14) qui nous a permis de vérifier la possibilité d'intégration de ce dispositif sur ALICE.

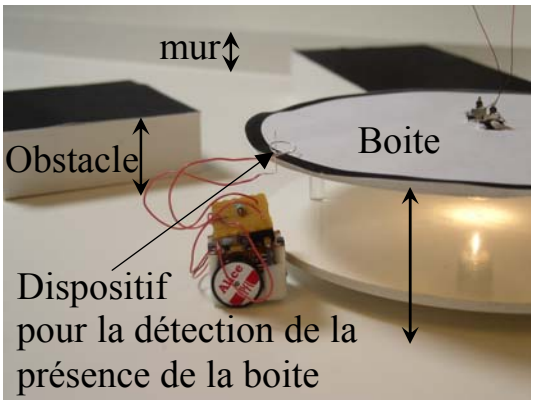


FIG. 8.14 – Détection du contact du robot avec l'objet à pousser

Pour une fiabilité accrue de la détection de l'objet à pousser, nous envisageons de procéder à une fusion des informations parvenant du capteur de contact et celles récupérées par l'anneau de capteurs (cf. figure. 8.8). L'anneau de capteurs servira ainsi aussi pour détecter le dépassement d'un seuil d'intensité lumineuse[9] qui indiquera par conséquent au robot, l'imminence du contact avec l'objet à pousser.

[8] Mesure effectuée grâce à "Hyper Terminal", application Windows qui nous a permis d'afficher en temps réel les messages reçus par ALICE.

[9] Nous rappelons que l'objet à pousser dispose d'une balise qui émet de la lumière.

8.2.2 Aspects logiciels

Les aspects logiciels de l'implémentation des architectures de contrôle proposées consistent, d'une part, à interfacer les capteurs ajoutés au microcontrôleur d'ALICE (partie que nous ne détaillons pas dans ce qui suit), et d'autre part, à implémenter tous les comportements élémentaires définis au chapitre IV (cf. §4.3.1, page 91) ainsi que les mécanismes de coordination entre comportements sur le microcontrôleur d'ALICE. Pour cette deuxième partie relative à l'implémentation logicielle des architectures de contrôle proposées aux chapitres précédents, elle a consisté à programmer et à intégrer les programmes de contrôle sur ALICE de telle manière à respecter d'une part l'architecture logicielle déjà existante (Url-programmation 2005) et d'autre part à optimiser le code des programmes afin qu'ils puissent être à la fois exécuter rapidement par le PIC16F877 d'ALICE et tenir dans les 8Kmots d'espace mémoire disponible sur ce PIC16F877.

L'implémentation de l'architecture de contrôle a été réalisée majoritairement en langage C à l'aide d'un compilateur dédié aux microcontrôleurs de la famille MICROCHIP, en l'occurrence le CCS-C (Url-compilateur 2005). Ceci a grandement facilité la transposition des parties de programmes déjà existantes au niveau du simulateur *MiRoCo* vers le CCS-C.

8.2.3 Expérimentations

Nous avons testé tout d'abord chaque comportement élémentaire afin de valider son adéquation par rapport à la structure matérielle et logicielle d'ALICE. Par la suite nous avons rassemblé l'ensemble de ces comportements élémentaires dans une structure unique de contrôle pour réaliser la TCPO. L'implantation des comportements élémentaires ont été concluants, au sens que les comportements réalisés par les robots correspondent à nos attentes (ceux observés en simulation).

La figure 8.15 représente l'exécution d'une TCPO par 8 mini-robots ALICE. Les robots arrivent ainsi à réaliser la tâche coopérative de poussée de boite sans trop se gêner. Nous notons que les informations d'angles ainsi que celles relatives au contact ou non des robots avec la boite (nécessaires au fonctionnement de l'architecture de contrôle) sont transmises aux robots dans cette expérimentation par l'intermédiaire d'une télécommande.

8.3 Conclusion

La plate-forme expérimentale que nous avons développée, se prête bien à l'étude de la coopération d'un groupe de mini-robots autonomes. Les phases expérimentales nous ont

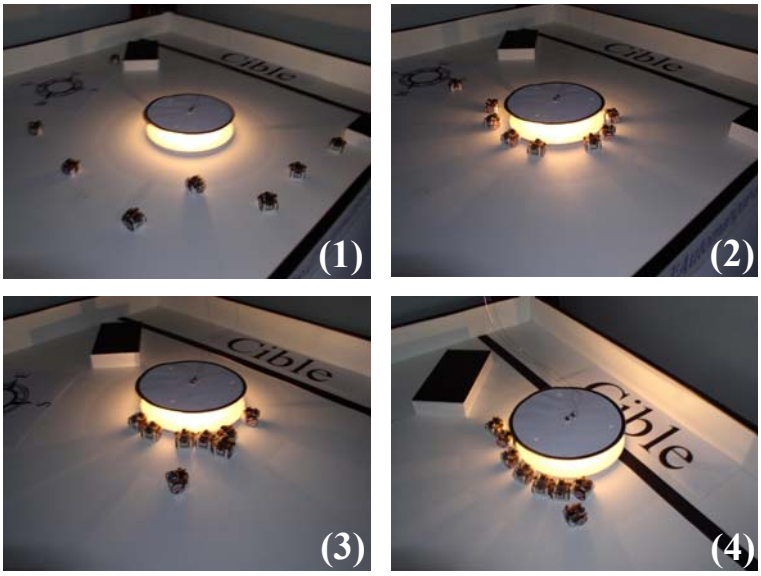


FIG. 8.15 – Expérimentation de l'architecture de contrôle avec 8 mini-robots poussant un objet cylindrique d'une position initiale vers la cible à atteindre.

tout d'abord amené à proposer et à développer de nouveaux capteurs qui puissent être intégrés aisément sur les mini-robots ALICE. Par la suite, les expérimentations nous ont permis de vérifier la viabilité des comportements élémentaires ainsi que les mécanismes de coordination proposés aux chapitres précédents.

Les perspectives à court terme sont liés à la réalisation de nombreuses expérimentations afin d'évaluer plus avant les architectures de contrôle proposées. Ceci sera effectif dès la disponibilité de l'ensemble des capteurs nécessaires à la réalisation de la TCPO au niveau de chaque mini-robot.

Conclusion générale et perspectives

Conclusion générale :

La voie investiguée dans le cadre de nos travaux de thèse correspond à une volonté de prospecter plus avant les potentialités inhérentes au contrôle distribué et réactif d'un groupe d'entités robotiques aussi minimalistes que possible, avec comme objectif l'obtention d'une forme d'intelligence collective. Cette vision du contrôle ne se fait pas toujours sans encombre, vue qu'elle implique parfois de revoir notre démarche de conception et de contrôle des entités robotiques, favorisant par exemple l'effet de masse et la flexibilité du contrôle à sa sophistication et à sa rigidité.

Nous nous sommes attachés au travers de la partie I de ce manuscrit à montrer la pluridisciplinarité des approches et des méthodes traitant du contrôle d'un groupe d'entités autonomes d'une manière générale, et d'entités robotiques d'une manière particulière. L'objectif était de pouvoir s'inspirer des différentes approches, méthodes et outils développés par ces diverses disciplines scientifiques (robotique, éthologie, systèmes multi-agents) afin de pouvoir proposer le contrôle le plus viable et le plus approprié pour faire coopérer un groupe de robots mobiles.

L'état de l'art réalisé sur les principales architectures de contrôle proposées dans la littérature, nous a montré l'engouement certain de la communauté scientifique pour les architectures de contrôle comportementales. Au sein de ces architectures de contrôle, cohabite généralement un ensemble de comportements élémentaires. Ces derniers doivent intervenir à des moments précis afin de contrôler convenablement l'évolution des tâches entreprises par le robot. Ceci stipule donc l'existence au sein de l'architecture de contrôle de mécanismes appropriés, qui permettent de coordonner l'activité de ces comportements. C'est d'ailleurs sur cet aspect du contrôle que se positionne la principale contribution de nos travaux de thèse, qui ont permis la définition des mécanismes de coordination à base de PSAH et/ou de PSAHH décrits aux chapitres IV et V respectivement.

Le PSAH "Processus de Sélection d'Action Hiérarchique" correspond à un mécanisme élémentaire de coordination entre comportements. Ce processus utilise la notion de Positions Refuges "$\mathbb{PR}$" caractérisant chaque comportement élémentaire, pour coordonner d'une manière hiérarchique l'activité de l'architecture de contrôle.

Le PSAH a été, par la suite, amélioré en intégrant des mécanismes de fusion d'actions et un mécanisme d'adaptation des gains, pour donner lieu au PSAH Hybride "PSAHH". Ce dernier processus de coordination plus flexible, intuitif et évolutif que le PSAH de base, s'est avéré être très adapté pour le contrôle de systèmes multi-robots avec des dynamiques d'évolution importantes.

L'évaluation de la viabilité du PSAH et du PSAHH s'est faite principalement au travers d'architectures comportementales complexes et distribuées, dédiées à la réalisation de la Tâche Coopérative de Poussée d'Objets "TCPO". Ceci nous a donc amené à développer des comportements élémentaires génériques (l'*évitement d'obstacles*, l'*exploration*, etc.) et d'autres dédiés (le *repositionnement*, la *poussée de boite*, etc.) afin de mener à bien, la réalisation de la TCPO. Le développement de ces architectures de contrôle nous a conduit aussi à proposer des comportements altruistes entre robots afin d'améliorer leur coopération. L'adjonction de ces comportements altruistes, qui se résume à une forme de communication de bas niveau entre robots, nous a conforté dans l'idée que malgré la simplicité des informations échangées entre robots, ceci permet d'une part, d'éviter des situations conflictuelles, et d'autre part, de générer une forme de coopération implicite entre robots, deux éléments qui conduisent à accélérer la réalisation de la tâche coopérative.

La pertinence des architectures de contrôle proposées a été testée d'une manière intensive au travers d'études statistiques basées sur la réalisation d'un grand nombre de simulations. Celles-ci ont permis d'expliciter au travers de leur nombre élevé et leurs diverses conditions initiales, les différents modes de fonctionnement du système multi-robots. Malgré le caractère simple des comportements élémentaires (nombre de commandes discrètes, réactifs), nous arrivons à obtenir des architectures de contrôle évoluées particulièrement adaptées à des systèmes multi-robots manifestant des dynamiques d'évolutions élevées.

L'élément principal qui peut rendre l'utilisation du PSAHH laborieuse, correspond à la phase de détermination des gains spécifiques aux blocs de fusion. C'est pour cette raison que nous proposons au chapitre VI, une méthodologie appropriée pour optimiser automatiquement les valeurs de ces gains. Cette optimisation paramétrique est obtenue en utilisant une méthode évolutionniste à base d'algorithmes génétiques. Des opérateurs génétiques appropriés ont été utilisés pour manipuler des chromosomes disposant

de gènes à valeur réelle et soumis à des contraints émanants de l'architecture de contrôle.

Du point de vue des outils logiciels et matériels réalisés dans le cadre de nos travaux de recherches, nous avons développé le simulateur *MiRoCo* (*Mini-Robotique Collective*) qui nous a permis de part les différentes modélisations implémentées, d'obtenir une précision importante de la dynamique d'évolution engendrée par le système multi-robots. Ainsi nous avons pu évaluer objectivement la pertinence des contrôles proposés. *MiRoCo* constitue à présent un outil complet pour simuler des systèmes multi-robots. L'autre outil de simulation que nous avons développé, correspond à la bibliothèque d'algorithmes génétiques *BibAG* qui interfacée avec *MiRoCo*, nous permet de réaliser des optimisations appropriées des architectures de contrôle proposées. Nous avons clôturé ce manuscrit par le chapitre VIII qui présente la plate-forme expérimentale développée ainsi que les expérimentations effectuées. Ces expérimentations nous ont permis de vérifier la viabilité des comportements élémentaires ainsi que les mécanismes de coordination entre comportements proposés.

Perspectives :

Les perspectives inhérentes à nos travaux de recherches sont variées et étendues, et ce à l'image du champ de recherche investigué durant nos travaux de thèse. Cela va donc de la modélisation au contrôle des systèmes multi-robots, en passant par l'optimisation paramétrique, la simulation, et finalement la manipulation. Ces différentes perspectives sont détaillées dans ce qui suit.

Architecture de contrôle :

Les perspectives à court terme concernant l'évolution des architectures de contrôle proposées, correspondent à l'utilisation de comportements élémentaires plus évolués (générant par exemple des commandes continues) afin de tester l'adéquation du PSAH et du PSAHH à ce type d'architectures de contrôle.

Effectivement, les comportements élémentaires constituant les briques de base des architectures comportementales, nous devons par conséquent leur accorder une attention toute particulière lors des phases de développement. Ces comportements doivent non seulement être complètement adaptés à la structure du robot mobile, mais doivent aussi exploiter au maximum les perceptions qui leurs parviennent de leur environnement. Sur ce deuxième point particulièrement, il serait nettement plus souhaitable de représenter l'application $\mathcal{F}$ de l'ensemble des stimuli $\mathbb{S}$ vers l'ensemble des commandes $\mathbb{C}$ (cf. §4.1.1,

page 78), non plus avec $c = \mathcal{F}(s)$, mais plutôt par $c = \mathcal{F}(s, \dot{s}, \ldots)$. Ainsi, les réponses du comportement élémentaire tiendraient compte de l'historique des stimuli perçus à l'image par exemple d'un régulateur de type proportionnel dérivé. Le comportement de *repositionnement* par exemple défini pour le cas de la TCPO (cf. §4.3.1.4, page 93), pourrait être redéfini afin que ces fonctions $\boldsymbol{f}$ et $\boldsymbol{g}$ ne dépendent pas que de θ, mais également de $\dot{\theta}$.

L'utilisation d'approches s'apparentant par exemple au DVZ[10] "*Deformable Virtual Zone*" proposé par René Zapata (Zapata & Lepinay 1994), serait un des moyens efficace pour obtenir un comportement élémentaire d'*évitement d'obstacles* générant des commandes qui soient à la fois continues et qui tiennent compte de l'historique des perceptions (Cacitti & Zapata 2001).

Dans une perspective à court terme, nous envisageons aussi :

- d'adjoindre au PSAH et au PSAHH de base, d'autres éléments de contrôle qui interviendraient pour la coordination des comportements élémentaires. Pour cela nous souhaitons intégrer des mécanismes adaptés pour représenter l'état interne du robot tels que : le nombre de fois qu'un comportement s'est activé en un temps précis, la mémorisation à court terme de la séquentialité des comportements activés, l'utilisation de l'état d'avancement des tâches entreprises (à l'image du *progress estimator* proposé dans (Mataric´ et al. 1995)). Tous ces éléments cités ci-dessus, peuvent être efficacement utilisés pour améliorer l'exécution des tâches coopératives entreprises,
- d'appliquer le PSAH et le PSAHH pour contrôler d'autres types de tâches coopératives génériques, telles que le fourragement ou la navigation en groupe. L'élaboration de ces nouvelles architectures de contrôle nous amènera certainement à réutiliser certains comportements élémentaires déjà développés, cependant d'autres comportements plus dédiés aux tâches génériques envisagées seront certainement à développer.

Ce qui peut être aussi réalisé sans demander un grand effort de conception, correspond au fait d'utiliser le PSAH et le PSAHH au contrôle d'entités autonomes volantes. Les comportements élémentaires, dans ce cas de figure, disposeront de positions refuges

[10] Cette approche consiste à affecter à un corps rigide (le robot) un volume caractérisant l'étendue de ses perceptions, et à utiliser les déformations de ce volume induites par la présence d'obstacles dans l'environnement immédiat du robot, pour générer les commandes les plus appropriées pour contrôler son mouvement.

$\mathbb{PR}$ contenues dans un volume (l'espace) au lieu d'être contenues dans le plan.

Les autres points, qui nous paraissent importants à explorer davantage, correspondent au fait de :

- **exploiter plus le lien entre les perceptions et les actions :** l'idée est d'investiguer plus en profondeur la relation existante entre les perceptions et les actions pour le contrôle des robots mobiles. Une reconstruction de l'environnement pourrait être envisagée sans toutefois toucher au caractère réactif du contrôle. Nous entendons par cela, le fait d'avoir une autre couche de contrôle qui serait plus cognitive et qui influencerait le robot dans ses décisions de plus haut niveau,
- **s'inspirer plus du vivant :** il est certain que nous avons encore beaucoup à apprendre des mécanismes organisationnels des sociétés d'insectes. Celles-ci constituent à notre sens l'illustration parfaite des capacités organisationnelles et d'adaptation d'un contrôle distribué et réactif d'un système complexe. Les équipes pluridisciplinaires, gravitant autour d'études relatives au contrôle d'un groupe d'entités autonomes, doivent à notre sens provoquer une synergie plus importante pour aboutir un jour à la coopération de robots autonomes, d'une manière aussi flexible et robuste que le font les insectes sociaux.

Optimisation paramétrique :

Parmi les éléments pouvant être optimisés afin d'affiner le fonctionnement des architectures de contrôle proposées, nous mentionnons la nécessité d'obtenir la forme optimale des Positions Refuges "$\mathbb{PR}$" propre à chaque comportement élémentaire. Ces formes devraient être fonction de la dynamique désirée du système multi-robots et de la définition de l'application $\mathcal{F}$ liant les perceptions aux actions (cf. figure.4.8, page 85 ; pour un exemple de la forme des $\mathbb{PR}$ affectée empiriquement au cas du comportement d'*évitement d'obstacles*).

Afin d'avoir aussi une adéquation plus importante entre les mécanismes de coordination des comportements et les comportements élémentaires eux-mêmes, nous envisageons en correspondance avec la méthode d'optimisation proposée au chapitre VI, de pouvoir coder les paramètres des comportements élémentaires et ceux liés au bloc de fusion dans un chromosome unique. Les paramètres de contrôle ainsi obtenus seront certainement plus adéquats pour l'architecture de contrôle vue dans sa globalité.

Pour les gains des blocs de fusion du PSAHH, nous envisageons d'investiguer les méthodes d'optimisation multi-critères de type Pareto-optimales (Fonseca & Peter 1995),

(Zitzler & Thiele 1998), (Talbi 1999), (Pirjanian 1999) afin d'obtenir un ensemble de solutions satisfaisantes pour différents types de dynamiques du système multi-robots. L'idée ensuite est de pouvoir attribuer dynamiquement ces gains aux blocs de fusion correspondants, en fonction des situations rencontrées par le robot et de la dynamique du système multi-robots perçue. Cette perspective est une analogie au fait que nous adaptons notre vigilance quand nous marchons en fonction du terrain sur lequel nous sommes (e.g., montagneux, sablonneux ou complètement plat). Les problématiques majeures que nous sommes appelés à résoudre ici, correspondent d'une part, à pouvoir caractériser les différents environnements du robot en utilisant uniquement ses perceptions, et d'autre part, à définir une correspondance entre ses perceptions et les solutions Pareto-optimales trouvées.

Simulation :

Plusieurs améliorations sont envisageables pour rendre les simulations sous *MiRoCo* d'autant plus rapides, plus précises et plus conviviales qu'elles le sont actuellement. Parmi les pistes envisageables, citons le fait d'optimiser davantage les programmes du *moteur physique* afin d'avoir des temps de calculs moins importants, ou bien l'identification plus précise des caractéristiques structurelles du mini-robot ALICE (capteurs, actionneurs, communication, etc.) pour améliorer la modélisation faite d'ALICE dans *MiRoCo*. Nous pourrons ainsi obtenir des simulations plus précises des dynamiques des système multi-robots. Il serait aussi envisageable d'intégrer la librairie ODE "*Open Dynamique Engines*" à *MiRoCo* afin de lui permettre une utilisation aisée de plusieurs modélisations liées à la physique d'interaction entre agents (frottement, amortissement, etc.).

Du point de vue de la convivialité d'utilisation de *MiRoCo*, là aussi nous pouvons envisager plusieurs améliorations qui peuvent se résumer comme suit :

- l'ajout d'une interface utilisateur, qui permettra d'agencer et de modifier les architectures de contrôle proposées d'une manière plus conviviale, c'est-à-dire sans recourir à chaque fois à une modification des lignes de code. L'idée est d'utiliser des blocs de comportements pré-programmés et des liens appropriés pour construire intuitivement (e.g., à l'aide de la souris) l'architecture de contrôle à implémenter sur les robots,
- la génération automatique du code machine à implémenter sur les véritables robots, et ce à partir du contrôle implémenté et testé au préalable en simulation. Ceci dans le cadre du mini-robot ALICE (cf. chapitre VIII) correspondrait à générer le code machine du microcontrôleur PIC16F877.

Manipulation :

D'un point de vue expérimental, la réalisation de nombreuses expérimentations pour le cas de la tâche coopérative de poussée d'objets, est un objectif à très court terme. En effet, l'architecture de contrôle proposée pour le cas de la TCPO n'a pu être testée expérimentalement d'une manière étendue, et ce vu la nécessité d'intégration de certains nouveaux capteurs sur les mini-robots ALICE qui n'ont pas pu se faire à temps.

Toujours dans le cadre des expérimentations que l'on pourrait réaliser dans un avenir proche, nous envisageons de réaliser un convoyage de pièces dans un mini-atelier de production comprenant plusieurs postes de travail. L'idée consistera à définir d'une manière globale et automatique (par un superviseur par exemple) l'objet à pousser et le poste de travail à atteindre pour que le groupe de robots minimalistes se mettent à coopérer pour réaliser ce convoyage automatique des pièces.

Modélisation :

La modélisation des systèmes multi-robots n'a pas été traitée dans ce rapport de thèse. Néanmoins, nous avons entamé des collaborations dans ce sens avec une équipe de recherche de l'EPFL (École Polytechnique Fédérale de Lausanne), afin de pouvoir représenter d'une manière probabiliste l'évolution du système multi-robots réalisant la TCPO.

Une modélisation fine des systèmes multi-robots est sans doute un point central à atteindre, pour arriver un jour à contrôler des systèmes multi-robots comme on le fait pour un moteur électrique. Pour cela, des collaborations très étroites avec des physiciens, mathématiciens, éthologues doivent être certainement renforcées.

Annexe A

Principales définitions liées à un système multi-robots

Introduction

Les définitions qui seront données dans ce qui suit, ont largement profité de plusieurs travaux d'équipes travaillant sur les Systèmes Multi-Agents **SMA**, qui ont fait et font toujours des efforts considérables sur les aspects liés à la normalisation et aux nomenclatures utilisés dans le cadre des SMA. Parmi elles, nous pouvons citer le groupe de travail CollInE[1].

Agent

Un agent[2] peut être défini comme une entité (physique ou abstraite) capable d'agir sur elle-même et sur son environnement, disposant d'une représentation partielle ou globale de son environnement, pouvant communiquer avec d'autres agents et dont le comportement est la conséquence de ses observations, de sa connaissance et de ses interactions avec les autres agents (Ferber & Ghallab 1988).

Système Multi-Agents (SMA)

Un Système Multi-Agents est un système composé d'un grand nombre d'agents, qui ont un comportement collectif qui permet d'atteindre une fonction désirée.

[1] Collectif Interaction Émergence, `http://www.irit.fr/COLLINE/`

[2] L'étymologie du mot agent provient du latin *agens* (participe de *agere*), qui signifie tout ce qui agit.

Tâche

Une tâche à réaliser est une certaine configuration de l'état du monde qu'un agent s'engage à atteindre et/ou à maintenir.

Stimulus et/ou Perception

Un stimulus/perception est un moyen de recevoir de l'information d'un environnement physique ou social (les autres agents).

Environnement

L'environnement d'un agent désigne tout ce qui est extérieur à l'agent. On trouve plusieurs définitions proposées entre autres dans (Russell & Norving 1995), (Woolridge 2000) ou bien dans (Lind 2001) qui se rejoignent globalement. Les types d'environnement auxquels on est confronté dans nos travaux se résument ainsi :

- *environnement continu* (par opposition à "*discret*"). Dans un environnement continu, le nombre d'actions et de perceptions possibles dans cet environnement est infini. Dans un environnement discret, le système possède des perceptions distinctes, clairement définies qui décrivent l'environnement,
- *environnement dynamique* (par opposition à "*statique*"). L'état d'un environnement dynamique dépend des actions des agents (au nombre généralement important) qui se trouvent dans cet environnement. Ainsi, les actions d'un agent sont complètement tributaires des actions des autres agents, ce qui donne une évolution dynamique de l'environnement. Contrairement à cela, un environnement statique est peu tributaire des actions de agents qui le composent (cas où les agents n'interagissent pas entre-eux).

Représentation du monde

Les représentations du monde d'un agent - ou croyances - désignent des connaissances qui sont souvent partiales et partielles sur les compétences d'autres agents (l'environnement social), sur l'environnement physique ou sur l'agent lui-même. L'agent doit toujours pouvoir accéder à ces représentations pour décider de son comportement et, éventuellement, il doit pouvoir les modifier.

Agent situé "*situatedness*"

Rodney Brooks dans (Brooks 1991) est parmi les premiers à définir le terme situatedness[3]. Maja Mataric´ dans (Mataric´ 2001) définit ce terme comme le fait que les comportements d'une entité sont étroitement affectés par son environnement[4]. Ainsi un robot qui évolue dans un environnement statique ne sera pas considéré comme situé, alors qu'une entité robotique par exemple dans un système multi-robots sera inévitablement caractérisée de située vue la grande dynamique d'interactions inhérente à ce genre de systèmes.

Agent incarné dans un corps "*embodiment*"

L'embodiment[5] est une forme de *situatedness* plus contrainte (Mataric´ 2001), car en plus du fait que l'entité autonome soit située dans un environnement dynamique, celle-ci disposera aussi d'un corps qui induira par voie de conséquence des contraintes supplémentaires par rapport à l'évolution de l'entité autonome dans l'environnement dynamique.

Robotique collective versus coopérative

Le point commun entre la robotique dite *collective* et la robotique *coopérative* est que les deux expressions stipulent impérativement l'existence de plus d'un seul robot pour l'exécution de la tâche désirée. Cependant, la robotique coopérative est forcément collective, alors qu'une tâche collective ne stipule pas forcement la coopération entre les entités robotiques (e.g., l'exploration d'un environnement inconnu par un groupe de robots ne stipule pas systématiquement une coopération effective entre les robots).

[3] "*The robots are situated in the world, they do not deal with abstract descriptions, but with the "here" and "now" of the environment that directly influences the behavior of the system*".

[4] "*Situatedness refers to having one's behavior strongly affected by the environment*".

[5] "*The robots have bodies and experience the world directly-their actions are part of a dynamic with the world, and the actions have immediate feedback on the robots' own sensations*" (Brooks 1991).

Annexe B

Comportements élémentaires continus

Introduction

Pour définir des comportements élémentaires réactifs (i.e., fonctionnant par stimuli-réponse), il n'y a dans la littérature aucune méthode générale permettant d'aboutir aux applications $\mathcal{F}$ (cf. §4.1.1, page 78) qui font un lien direct entre les stimuli perçus et les actions (commandes) que les comportements doivent générer.

Pour le cas du mini-robot ALICE (cf. chapitre VIII), plusieurs comportements élémentaires ont été déjà programmés avec succès (Caprari 2003, p.77), et ce en utilisant une approche ad hoc qui fait un lien étroit entre les capacités structurelles (actionneurs, capteurs (nombre, type)) du robot et les commandes qu'il doit générer afin de réaliser correctement le comportement désiré (e.g., évitement d'obstacle, suivi de murs).

Dans ce qui suit, nous donnons les détails d'implémentation de deux comportements élémentaires continus, qui correspondent aux comportements d'*évitement d'obstacles* et d'*attraction à la lumière*. Il est à noter que ces comportements ne prétendent en aucun cas à l'optimalité de leurs actions. En effet, leur vocation principale est de générer des actions satisfaisantes par rapport à leurs objectifs (en l'occurrence l'évitement d'obstacle et l'attraction à la lumière). La simplicité de leur implémentation est due aussi à la nature très rudimentaire des perceptions utilisées, et à la volonté d'avoir des comportements élémentaires les plus simples possibles.

Comportement d'évitement d'obstacles

Les stimuli qui parviennent à ce comportement, sont purement locaux et dépendent des quatre capteurs infrarouges de proximité existants au niveau du mini-robot ALICE (cf. figure. 4.2, page 79), en l'occurrence *CIR1*, *CIR2*, *CIR3*, *CIR4*. Chaque capteur, qui détecte un obstacle, peut donner une information sur la distance qui sépare le robot de celui-ci. Cette information sur la distance $\text{Dist}_{CIRi|i=1..4}$ est bornée et comprise dans l'intervalle $[0, \text{DistMax}_{CIRi|i=1..4}]$[1]. Les commandes $\mathcal{C}$ générées par ce comportement correspondent à une commande de rotation *Rot* (en degré) et une commande de translation *Tra* (en centimètre) (cf. §4.1.2, page 79).

Chaque information capteur (schéma perceptuel (Arkin 1989*b*)) est utilisée pour établir une commande partielle $\mathcal{C}_{CIRi|i=1..4}$ ($\mathcal{C}_{Roti}$, $\mathcal{C}_{Trai}$). L'étape finale pour l'obtention de la commande $\mathcal{C} = (\mathcal{C}_{Rot}, \mathcal{C}_{Tra})$ (commande à appliquer aux actionneurs du robot) consiste à calculer la moyenne de ces commandes partielles (cf. algorithme. B.1).

Le design des comportements élémentaires contenus dans les architectures de contrôle comportementales, doit s'attacher au fait que les commandes qu'ils sont susceptibles de générer, doivent respecter les contraintes structurelles du robot où ils sont implantés. Nous réalisons ceci dans notre cas en imposant que les commandes générées $\mathcal{C}$ soient bornées. Ceci est réalisé en fixant convenablement les constantes $\text{Const}_i|_{i=1..3}$ (cf. algorithme. B.1).

Comportements d'attraction à la lumière

Ce comportement consiste à attirer le robot vers une source de lumière. Les capteurs infrarouges *CIR1*, *CIR2*, *CIR3*, *CIR4* du mini-robot ALICE sont utilisés ici en simple réception, et les commandes à envoyer aux actionneurs dépendent de la contribution des deux commandes partielles correspondantes aux capteurs qui relèvent les intensités lumineuses les plus élevées (cf. algorithme. B.2).

Il est à noter que "ValeurMoyenneSignaux" (cf. algorithme. B.2) correspond à la moyenne des intensités relevées par les quatre capteurs infrarouges, est utilisée pour avoir une référence, d'une part, par rapport à la lumière ambiante dans l'environnement, et d'autre part, par rapport aux différentes intensités relevées par les capteurs infrarouges.

[1] DistMax_{CIRi}, pour le cas des ALICE, sont de l'ordre de 4 cm.

Algorithme B.1 Comportement d'*évitement d'obstacles* (continu)

1: //Initialisations
2: $\mathcal{C}_{Rot} = 0$; $\mathcal{C}_{Tra} = \text{Const}_0$; //Composantes de la commande finale $\mathcal{C}$
3: $\mathcal{C}_{Tra1}$, $\mathcal{C}_{Tra2}$, $\mathcal{C}_{Tra3}$, $\mathcal{C}_{Tra4} = 0$; //Composantes des commandes partielles
4: $\mathcal{C}_{Rot1}$, $\mathcal{C}_{Rot2}$, $\mathcal{C}_{Rot3}$, $\mathcal{C}_{Rot4} = 0$;
5: $\mathcal{C}_{RotMax} = \text{Const}_1$; //Rotation max autorisée pour la commande $\mathcal{C}$
6: $\mathcal{C}_{TraMax} = \text{Const}_2$; //Translation max autorisée pour la commande $\mathcal{C}$
7: *Attenuation* = Const_3 ; //Valeur comprise entre]0, 1[, utilisée pour atténuer l'influence de certaines commandes partielles par rapport à d'autres.//
8: NbreCapteursActifs=0 ;
9: ---
10: //Tests de détection ou pas d'obstacles et calcul des commandes partielles
11: **Si** *CIR1* détecte un obstacle **Alors**
12: $\quad\mathcal{C}_{Tra1} = (\text{Dist}_{CIR1}/\text{DistMax}_{CIR1}) \times \mathcal{C}_{TraMax}$;
13: $\quad\mathcal{C}_{Rot1} = (1 - (\text{Dist}_{CIR1}/\text{DistMax}_{CIR1})) \times \mathcal{C}_{RotMax} \times \mathit{Attenuation}$;
14: $\quad$NbreCapteursActifs = NbreCapteursActifs + 1 ;
15: **Fin Si**
16: **Si** *CIR2* détecte un obstacle **Alors**
17: $\quad\mathcal{C}_{Tra2} = (\text{Dist}_{CIR2}/\text{DistMax}_{CIR2}) \times \mathcal{C}_{TraMax}$;
18: $\quad\mathcal{C}_{Rot2} = (1 - (\text{Dist}_{CIR2}/\text{DistMax}_{CIR2})) \times \mathcal{C}_{RotMax}$;
19: $\quad$NbreCapteursActifs = NbreCapteursActifs + 1 ;
20: **Fin Si**
21: **Si** *CIR3* détecte un obstacle **Alors**
22: $\quad\mathcal{C}_{Tra3} = (\text{Dist}_{CIR3}/\text{DistMax}_{CIR3}) \times \mathcal{C}_{TraMax}$;
23: $\quad\mathcal{C}_{Rot3} = -(1 - (\text{Dist}_{CIR3}/\text{DistMax}_{CIR3})) \times \mathcal{C}_{RotMax} \times \mathit{Attenuation}$;
24: $\quad$NbreCapteursActifs = NbreCapteursActifs + 1 ;
25: **Fin Si**
26: **Si** *CIR4* détecte un obstacle **Alors**
27: $\quad\mathcal{C}_{Tra4} = (\text{Dist}_{CIR4}/\text{DistMax}_{CIR4}) \times \mathcal{C}_{TraMax}$;
28: $\quad\mathcal{C}_{Rot4} = -(1 - (\text{Dist}_{CIR4}/\text{DistMax}_{CIR4})) \times \mathcal{C}_{RotMax}$;
29: $\quad$NbreCapteursActifs = NbreCapteursActifs + 1 ;
30: **Fin Si**
31: ---
32: //Calcul de la commande finale
33: **Si** NbreCapteursActifs $\neq 0$ **Alors**
34: $\quad\mathcal{C}_{Tra} = (\mathcal{C}_{Tra1} + \mathcal{C}_{Tra2} + \mathcal{C}_{Tra3} + \mathcal{C}_{Tra4})$ / NbreCapteursActifs ;
35: $\quad\mathcal{C}_{Rot} = (\mathcal{C}_{Rot1} + \mathcal{C}_{Rot2} + \mathcal{C}_{Rot3} + \mathcal{C}_{Rot4})$ / NbreCapteursActifs ;
36: $\quad$**Retourner** $\mathcal{C} = (\mathcal{C}_{Rot}, \mathcal{C}_{Tra})$;
37: **Sinon**
38: $\quad$**Retourner** $\mathcal{C} = (\mathcal{C}_{Rot}, \mathcal{C}_{Tra})$;
39: **Fin Si**

Algorithme B.2 Comportement d'*attraction à la lumière* (continu)

Requiert : $\text{Int}_{CIRi|i=1..4}$ et "$Ind1$ et $Ind2$" ; //Correspondent respectivement aux intensités lumineuses relevées par chaque capteur infrarouge, et aux indices des deux capteurs qui relèvent les plus grandes intensités lumineuses de l'environnement.//
//Initialisations
ValeurMoyenneSignaux = $\sum_{i=1..4} \text{Int}_{CIRi}/4$;
$\mathcal{C}_{RotMax} = \text{Const}_1$; //Rotation max autorisée pour la commande finale $\mathcal{C}$
$\mathcal{C}_{TraMax} = \text{Const}_2$; //Translation max autorisée pour la commande finale $\mathcal{C}$
$Attenuation = \text{Const}_3$; //Valeur comprise entre]0, 1[, utilisée pour atténuer l'influence de certaines commandes partielles par rapport à d'autres.//

//Calcul des contributions potentielles de chaque schéma perceptuel.

//Contribution éventuelle du capteur *CIR1* "capteur de devant"
$\mathcal{C}_{Tra1} = (\text{Int}_{CIR1}/\text{ValeurMoyenneSignaux}) \times \mathcal{C}_{TraMax}$;
$\mathcal{C}_{Rot1} = 0$;

//Contribution éventuelle du capteur *CIR2* "capteur droit"
$\mathcal{C}_{Tra2} = (\text{Int}_{CIR2}/\text{ValeurMoyenneSignaux}) \times \mathcal{C}_{TraMax}$;
$\mathcal{C}_{Rot2} = \text{-}\ (\text{Int}_{CIR2}/\text{ValeurMoyenneSignaux}) \times \mathcal{C}_{RotMax}$;

//Contribution éventuelle du capteur *CIR3* "capteur arrière"
$\mathcal{C}_{Tra3} = 0$;
Si $Ind1$ ou $Ind2$ correspond à l'indice du capteur "*CIR2*" **Alors**
  $\mathcal{C}_{Rot3} = \text{-}\ (\text{Int}_{CIR3}/\text{ValeurMoyenneSignaux}) \times \mathcal{C}_{RotMax} \times Attenuation$;
Fin Si
Si $Ind1$ ou $Ind2$ correspond à l'indice du capteur "*CIR4*" **Alors**
  $\mathcal{C}_{Rot3} = (\text{Int}_{CIR3}/\text{ValeurMoyenneSignaux}) \times \mathcal{C}_{RotMax} \times Attenuation$;
Fin Si

//Contribution éventuelle du capteur *CIR4* "capteur gauche"
$\mathcal{C}_{Tra4} = (\text{Int}_{CIR4}/\text{ValeurMoyenneSignaux}) \times \mathcal{C}_{TraMax}$;
$\mathcal{C}_{Rot4} = (\text{Int}_{CIR4}/\text{ValeurMoyenneSignaux}) \times \mathcal{C}_{RotMax}$;

//Le calcul des composantes de la commande finale sont donnés comme suit :
$\mathcal{C}_{Tra} = \mathcal{C}_{Tra\text{"}Ind1\text{"}} + \mathcal{C}_{Traj\text{"}Ind2\text{"}}$;
$\mathcal{C}_{Rot} = \mathcal{C}_{Rot\text{"}Ind1\text{"}} + \mathcal{C}_{Rot\text{"}Ind2\text{"}}$;
Retourner $\mathcal{C} = (\mathcal{C}_{Rot}, \mathcal{C}_{Tra})$;

Bibliographie

Adouane, L. & LeFort-Piat, N. (2003*a*), Bio-inspired behaviours of a group of microrobots for cooperative box-pushing task, *in* '6th Japan-France Congress on Mechatronics and 4th Asia-Europe Congress on MECATRONICS', Tokyo Denki University Saytama, Japan, pp. 483–488.

Adouane, L. & LeFort-Piat, N. (2003*b*), Emergence d'une intelligence de groupe à partir d'entités microrobotiques frustres, *in* '6 èmes Journées du Pôle Microrobotique, 2 èmes Journées du RTP Microrobotique', Bourges-France, pp. 40–48.

Adouane, L. & LeFort-Piat, N. (2004*a*), Emergence of group intelligence from minimalist control of mobile mini-robots, *in* '35th International Symposium On Robotics ISR', Paris-France. In CD.

Adouane, L. & LeFort-Piat, N. (2004*b*), Evolutionary parameters optimization for an hybrid control architecture of multicriteria tasks, *in* 'International Conference On Robotics And Biomimetics ROBIO', Shenyang-China. In CD, N°365.

Adouane, L. & LeFort-Piat, N. (2004*c*), Hybrid behavioral control architecture for the cooperation of minimalist mobile robots, *in* 'International Conference On Robotics And Automation', New Orleans-USA, pp. 3735–3740.

Adouane, L. & LeFort-Piat, N. (2005*a*), 'Behavioral and distributed control architecture for minimalist mobile robots', *Journal Européen des Systèmes Automatisés*. In press.

Adouane, L. & LeFort-Piat, N. (2005*b*), Methodology of parameters optimization for an hybrid architecture of control, *in* '16th IFAC World Congress', Prague-Czech Republic. In press.

Aguilar, L. E. (1997), Commande robuste et coordination de mouvements de robots mobiles, PhD thesis, LAAS N°97452.

Ahmadabadi, M. N. & Nakano, E. (2001), 'A "constrain and move" approach to distributed object manipulation', *IEEE Transactions On Robotics And Automation* **17**(2), pp.157–172.

Alami, R., Fleury, S., Herrb, M., Ingrand, F. & Robert, F. (1998*a*), 'Multi-robot cooperation in the martha project', *IEEE Robotics and Automation Magazine* **5**(1), pp.36–47.

Alami, R., Ingrand, F. & Qutub, S. (1998*b*), A scheme for coordinating multi-robot planning activities and plans execution, *in* '13th European Conference on Artificial Intelligence', Brighton, UK.

Albus, J. (1991), 'Outline for a theory of intelligence', *IEEE Transactions on Systems* **21**(3), pp.473–509.

Anderson, T. & Donath, M. (1990), 'Animal behavior as a paradigm for developing robot autonomy', *Robotics and Autonomous Systems* **6**, pp.145–168.

Arbib, M. A. (1981), 'Perceptual structures and distributed motor control', *in Handbook of Physiology, Section 2 : The Nervous System, II, Motor Control, Part 1 (V.B. Brooks, Ed.). American Physiological Society* pp. 1449–1480.

Arkin, R. (1989*a*), Towards the unification of navigational planning and reactive control, *in* 'AAAI Spring Symposium on Robot Navigation', pp. 1–5.

Arkin, R. & MacKenzie, D. (1984), 'Temporal coordination of perceptual algorithms for mobile robot navigation', *IEEE Transactions on Robotics and Automation* **10**(3), pp.276–286.

Arkin, R. C. (1987), Motor schema based navigation for a mobile robot : An approach to programming by behavior, *in* 'Conference on Robotics and Automation', Raleigh, pp. 264–271.

Arkin, R. C. (1989*b*), 'Motor schema-based mobile robot navigation', *International Journal of Robotics Research* **8**(4), pp.92–112.

Arkin, R. C. (1992), 'Cooperation without communication : Multi-agent schema based robot navigation', *Journal of Robotic Systems* **9**(3), pp.351–364.

Arkin, R. C. (1998), *Behavior-Based Robotics*, The MIT Press.

Arkin, R. C., Fujita, M., Takagi, T. & Hasegawa, R. (2001), Ethological modeling and architecture for an entertainment robot, *in* 'International Conference On Robotics And Automation', pp. 453–458.

Arrúe, B., Cuesta, F., Braunstingl, R. & Ollero, A. (1997), Fuzzy behaviors combination to control a non-holonomic mobile robot using virtual perception memory, *in* 'Proceedings of the 6th IEEE International Conf. on Fuzzy Systems', Barcelona, Spain, pp. 1239–1244.

Ashby, R. (1960), *Design for a brain*, New-York : Wiley.

Avouris, N. & Gasser, L. (1992), *Distributed Artificial Intelligence : Theory and Praxis*, Vol. 5, Kluwer Academic Publishers.

Balch, T. (1997), Clay : Integrating motor schemas and reinforcement learning, Technical report, College of Computing GIT-CC-97-11.

Balch, T. & Arkin, R. C. (1993), Avoiding the past : A simple but effective strategy for reactive navigation, *in* 'International Conference On Robotics And Automation', Vol. 1, Atlanta, pp. 678–685.

Balch, T. & Arkin, R. C. (1995*a*), 'Communication in reactive multiagent robotic systems', *Autonomous Robots* **1**(1), pp.27–52.

Balch, T. & Arkin, R. C. (1995*b*), Motor schema-based formation control for multiagent robot teams, *in* V. Lesser & L. Gasser, eds, 'Proceedings of the First International Conference on Multiagent Systems (ICMAS'95)', AAAI Press, San Francisco, CA, USA, pp. 10–16.

Baldassarre, G., Nolfi, S. & Parisi, D. (2003), 'Evolution of collective behaviour in a team of physically linked robots', *In R. Gunther, A. Guillot, and J.-A. Meyer, editors, Applications of Evolutionary Computing, Springer Verlag, Heidelberg, Germany,* pp. 581–592.

Barberá, H. M. & Skarmeta, A. G. (2002), 'A framework for defining and learning fuzzy behaviours for autonomous mobile robots', *International Journal of Intelligent Systems* **17**(1), pp.1–20.

Baud, A. (2004), Conception et réalisation d'un module de localisation et de communication pour les mini-robots alice, Projet de Fin d'Études, Laboratoire d'Automatique de Besançon - Institut Supérieur de l'Electronique et du Numérique de Lille.

Beckers, R., Deneubourg, J. & Goss, S. (1992), 'Trails and u-turns in the selection of the shortest path by the ant lasius niger', *Journal of theoretical biology* **159**, pp.397–415.

Beekman, M., Sumpter, D. & Ratnieks, F. (2001), Phase transition between disordered and ordered foraging in pharaohs' ants, *in* 'Proc. Nat. Acad. Sci.', Vol. 98, pp. 9703–9706.

Bellman, R., Holland, J. & Kalaba, R. (1959), 'On an application of dynamic programming to the synthesis of logical systems', *Journal of the ACM (JACM) archive* **6**(4), pp.486–493.

Bergen, G. V. D. (2004), *Collision Detection in Interactive 3D Environments*, Elsevier.

Bühler, H. (1994), *Réglage par logique floue*, Presse Polytechnique et Universitaires Romandes.

Bonabeau, E. & Theraulaz, G. (1994), *Intelligence collective*, Hermes, Paris.

Bonabeau, E., Dorigo, M. & Theraulaz, G. (1999), *Swarm intelligence 'from Natural to Artificial systems'*, Oxford University Press.

Booch, G., Rumbaugh, J. & Jacobson, I. (2000), *Le guide de l'utilisateur UML*, Eyrolles.

Borenstein, J. & Koren, Y. (1989), 'Real-time obstacle avoidance for fast mobile robots', *IEEE Transactions on Systems, Man, and Cybernetics* **19**(5), pp.1179–1187.

Borenstein, J. & Koren, Y. (1991), 'The vector field histogram – fast obstacle-avoidance for mobile robots', *IEEE Journal of Robotics and Automation* **7**(3), pp.278–288.

Braitenberg, V. (1984), *Vehicles : Experiments in synthetic psychology*, Cambridge, MA : MIT Press.

Bray, D. (1992), 'Cell mouvemements', *Pub : Garland Publishing Inc.*

Brooks, R. A. (1986), 'A robust layered control system for a mobile robot', *IEEE Journal of Robotics and Automation* **RA-2**, pp.14–23.

Brooks, R. A. (1987), Planning is just a way of avoiding figuring out what to do next, *in* 'Working Paper MIT, n° 303'.

Brooks, R. A. (1990), 'Elephant don't play chess', *Robotics and Automation Systems* **6**, pp.3–15.

Brooks, R. A. (1991), 'New approaches to robotics', *Science* **253**, pp.1227–1232.

Cacitti, A. & Zapata, R. (2001), Reactive behaviours of mobile manipulators based on the dvz approach, *in* 'International Conference on Robotics and Automation', Seoul-Korea, pp. 680–685.

Calenbuhr, V. & Deneubourg, J. (1989), Modélisation du comportement du suivi de la piste chez les fourmis, *in* 'Actes coll. Insectes Sociaux', Vol. 5, pp. 207–213.

Caloud, P., Choi, W., Latombe, J.-C., Pape, C. L. & Yim, M. (1990), Indoor automation with many mobile robots, *in* 'Intelligent Robots and Systems IROS', Ibaraki-Japan, pp. 67–72.

Cao, Y., Fukunaga, A. S. & Kahng, A. B. (1997), 'Cooperative mobile robotics : Antecedents and directions', *Autonomous Robots* **4**, pp.1–23.

Caprari, G. (2003), Autonomous Micro-Robots : Applications and Limitations, PhD thesis, Faculté Sciences et Techniques de l'Ingénieur, École Polytechnique Fédérale de Lausanne. Thèse N°2753.

Caprari, G. & Siegwart, R. (2003), Design and control of the mobile micro robot alice, *in* 'Proceedings of the 2nd International Symposium on Autonomous Minirobots for Research and Edutainment AMiRE'2003', Brisbane, Australia, pp. 23–32.

Caprari, G., Estier, T. & Siegwart, R. (2002), 'Fascination of down scaling - alice the sugar cube robot', *Journal of Micro-Mechatronics* **VSP-Utrecht**, pp.177–189.

Causse, O. & Pampagnin, L. (1995), Management of a multi-robot system in a public environment, *in* 'IROS', pp. 245–252.

Caux, C., Pierreval, H. & Portmann, M. (1995), 'Les algorithmes génétiques et leurs applications aux problèmes d'ordonnancement', *APII* **29**(4-5), pp.409–443.

Chankong, V. & Haimes, Y. Y. (1983), *Multiobjective Decision Making Theory and Methodology*, Vol. 8, NorthHolland.

Chauvin, R. (1982), *Les Sociétés Animales*, Presses Universitaires de France, Paris.

Chauvin, R. (1989), *Des Animaux et des Hommes*, Seghers, Paris.

Chen, M., Dorer, K., Foroughi, E., Heintz, F., Huang, Z., Kapetanakis, S., Kostiadis, K., Kummeneje, J., Murray, J., Noda, I., Obst, O., Riley, P., Steffens, T., Wang, Y. & Yin, X. (2003), Robocup soccer server, 2003. user manual for soccer server version 7.07 and later, Technical report.

Chen, Q. & Luh, J. Y. S. (1994), Coordination and control of a group of small mobile robots, *in* 'International Conference On Robotics And Automation', pp. 2315–2320.

Cohen, W. (1996), 'Adaptive mapping and navigation by teams of simple robots', *Robotics and Autonomous Systems* **18**, pp.411–434.

Connell, J. H. (1990), *Minimalist Mobile Robotics*, Academic Press, Londres.

Cornetz, V. (1911), 'La conservation de l'orientation chez les fourmis', *Revue Suisse de Zoologie*.

Cullen, J. & Shaw, E. (1965), 'Methods for measuring the three-dimensional structure of fish schools', *Animal Behavior* **13**, pp.534–543.

Culloch, W. & Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *Bulletin of Mathematical Biophysics* **5**, pp.115–133.

Darwin, C. (1859), *On the Origine of Species. A facsmilie of the first edition*, Harvard University Press, Cambridge, Massachusetts.

Delaye, C. (1993), Structures et organisations des systèmes multi-agents autonomes et adaptatifs, PhD thesis, Université Paris VI.

Deneubourg, J. (1977), Application de l'ordre par fluctuations à la description de certaines étapes de la construction du nid chez les termites, *in* 'Ins. Soc', Vol. 24, pp. 117–133.

Dittmar, A. & Delhomme, G. (2001), Les microcapteurs et microsystèmes du règne végétal et du règne animal, une source d'inspiration pour la microrobotique, *in* 'Quatrième journées du pôle MicroRobotique , INSA Lyon, France'.

Donald, B. R., Jennings, J. & Rus, D. (1994), Analyzing teams of cooperating mobile robots, *in* 'International Conference on Robotics and Automation', pp. 1896–1903.

Dorigo, M. & Schnepf, U. (1991), Organisation of robot behaviour through genetic learning processes, *in* 'Fifth IEEE International Conference on Advanced Robotics', Pisa, Italy, pp. 1456–1460.

Dreyfus, H. L. (1992), *What Computers Still Can't Do : A Critique of Artificial Reason*, The MIT Press. Revision of What Computers Can't Do, MIT Press, 1979.

Drogoul, A. (1993), De La Simulation Multi-Agent A La Résolution Collective de Problèmes – Une Étude De l'Émergence De Structures D'Organisation Dans Les Systèmes Multi-Agents, PhD thesis, Université Paris VI.

Eiben, A. E., Hinterding, R. & Michalewicz, Z. (1999), 'Parameter control in evolutionary algorithms', *IEEE Trans. on Evolutionary Computation* **3**(2), pp.124–141.

Englemore, E. & Morgan, T. (1988), *Blackboard sytems*, adisson-Wesley.

Erceau, J. & Ferber, J. (1991), 'L'intelligence artificielle distribuée', *La Recherche* (233), pp.750–758.

Ferber, J. & Ghallab, M. (1988), Problématique des univers multi-agents intelligents, *in* 'Actes des Journes nationales du PRC-GRECO "Intelligence Artificielle"', Toulouse, pp. 295–320.

Ferrell, C. (1995), 'Global behavior via cooperative local control', *Autonomous Robots* **2**(2), pp.105–125.

Filho, L. R., Treleaven, P. C. & Alippi, C. (1994), 'Genetic-algorithm programming environments', *Computer* **27**(6), pp.28–43.

Filliat, D., Kodjabachian, J. & Meyer, J. (1999), Incremental evolution of neural controllers for navigation in a 6-legged robot, *in* I. Sugisaka & T. (Eds.), eds, 'Proceedings of the Fourth International Symposium on Artificial Life and Robotics', Oita Univ. Press.

Fiorino, H. (1999), *État de l'art sur la coopération Multi-Robot*, LAAS N° 99237.

Firby, R. (1987), An investigation into reactive planning in complex domains, *in* 'Sixth National Conference on Artificial Intelligence', Seattle, pp. 202–206.

Fonseca, C. M. & Peter, J. (1995), 'An overview of evolutionary algorithms in multiobjective optimization', *Evolutionary Computaion* **3**(1), pp.1–16.

Forrest, S. (1991), *Emergent Computation : self-organizing, collective, and cooperative phenomena in natural and computing networks*, MIT Press.

Fraisse, P., Zapata, R., Zarrad, W. & Andreu, D. (2004), 'Remote secure decentralized control strategy for mobile robots', *Journal of Advanced Robotics*. In press.

Freeman, J. & Skapura, D. (1992), *Neural Networks : Algorithms, Applications and Programming Techniques*, Reading, PA : Addison-Wesley Publishing Company.

Fukuda, T., Takagawa, I., Sekiyama, K. & Hasegawa, Y. (2000), 'Hybrid approach of centralized control and distributed control for flexible transfer system', *Distributed Manipulation Kluwer Academic Publishers,* pp. 65–85.

Gadagkar, R. (1994), 'The evolution of altruism in insects - a case study.', *In : Pers. in Entomological Res. (ed. O.P. Agarwal). Scientific Publishers, Jodhpur,* pp. 263–275.

Gat, E. (1998), Three-layer architecture, *in* 'D. Kortenkamp, R.P Bonnasso and R. Murphy, erd, Artificial Intelligence and Mobile Robotics, AAAI Press', pp. 195–210.

Gerkey, B. P. & Mataric´, M. J. (2002*a*), Pusher-watcher : An approach to fault-tolerant tightly-coupled robot coordination, *in* 'International Conference on Robotics and Automation', Washington, DC, pp. 464–469.

Gerkey, B. P. & Mataric´, M. J. (2002*b*), 'Sold ! : Auction methods for multi-robot coordination', *IEEE Transactions on Robotics and Automation-Special issue on Advances in Multi-Robot Systems* **18**(5), pp.758–786.

Gervet, J. (1967), *L'effet de groupe dans la société polygyne de poliste (Hyménoptères Vespides)*, Colloques Internationaux du CNRS.

Giurfa, M. (2003), Comportement et cognition : ce que nous appernd un mini cerveau, *in* 'Quatrièmes Journées Nationales de la Recherche en Robotique'.

Godjevac, J. (1997), *Neuro-Fuzzy Controllers - Design and Application*, Presse Polytechnique et Universitaires Romandes.

Godzinska, E. (1986), Extra-colony altruism in the bumblebees : Misbehaviour or adapatation ?, *in* 'Actes Coll. Insectes Sociaux', Vol. 5, pp. 161–167.

Goldberg, D. & Mataric´, M. J. (1999), Coordinating mobile robot group behavior using a model of interaction dynamics, *in* 'Proceedings, Autonomous Agents', Seattle, WA, pp. 100–107.

Goldberg, D. E. (1989), *Genetic Algorithm in Search, Optimisation and Machine Learning*, Addison Wesley.

Grassé, P. P. (1959), 'La construction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. la théorie de la stigmergie : essais d'interprétation du comportement des termites constructeurs', *Ins . Soc* **6**, pp.41–84.

Grefenstette, J. & Schultz, A. (1995), An evolutionary approach to learning in robots, *in* 'Machine Learning Workshop on Robot Learning', New Brunswick, NJ.

Gutknecht, O., Ferber, J. & Michel, F. (2000), Madkit : une expérience d'architecture de plate-forme multi-agent générique, *in* H. La Réunion, ed., '8ème Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents (JFIADSMA'2000)', pp. 223–236.

Guvenir, H. A. & Erel, E. (1998), 'Multicreteria inventory classification using a genetic algorithm', *European Journal Of Operational Research* pp. 29–37.

Hajela, P. & Lin, C. Y. (1992), 'Genetic search strategies in multicriterion optimal design', *Structural Optimization* **4**, pp.99–107.

Harvey, I., Husbands, P., Cliff, D., Thompson, A. & Jakobi, N. (1996), 'Evolutionary robotics at sussex', *Robotics and Manufacturing : Recent Trends in Research and Applications* **6**, pp.293–298.

Hayes-Roth, B. (1984), *A BlackBoard Model of Control. Technical Report*, Standford University.

Head, H. & Holmes, G. (1911), 'Sensory disturbances from cerebral lesions', *Brain*.

Heitkõter, J. & Beasley, D. (2005), 'The hitch-hiker's guide to evolutionary computation. `http://surf.de.uu.net/encore/`'.

Hirata, Y., Hatsukari, T., Kosuge, K., Asama, H., Kaetsu, H. & Kawabata, K. (2002), 'Transportation of an object by multiple distributed robot helpers in cooperation with a human', *Transactions of the Japan Society of Mechanical Engineers* **68**(668), pp.181–188.

Hirata, Y., Kosuge, K., Asama, H., Kaetsu, H. & Kawabata, K. (2000), Coordinated transportation of a single object by multiple mobile robots without position information of each robot, *in* 'Proc. 2000 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems', pp. 2024–2029.

Holland, J. (1975), *Adaption in Natural and Artificial Systems*, University of Michigan Press.

Hornby, G., Fujita, M., Takamura, S., Yamamoto, T. & Hanagata, O. (1999), Autonomous evolution of gaits with the sony quadruped robot, *in* 'Proceedings of 1999 Genetic and Evolutionary Computation Conference (GECCO)', pp. 1297–1304.

Horton, I. (1999), *Visual C++ 6*, Eyrolles.

Huang, H., Liang, C.-C. & Lin, C.-W. (2001), Construction and soccer dynamics analysis for an integrated multi-agent soccer robot system, *in* 'Proc. Natl. Sci. Counc. ROC(A)', Vol. 25, pp. 84–93.

Hugues, L. (2000), Collective grounded representations for robots, *in* 'Fifth Symposium on Distributed Autonomous Robotics Systems'.

Huntsberger, T., Pirjanian, P., Trebi-Ollennu, A., Nayar, H. D., Aghazarian, H., Ganino, A., Garrett, M., Joshi, S. & Schenker, P. (2003), 'Campout : A control architecture for tightly coupled coordination of multi-robot systems for planetary surface exploration', *IEEE Trans. Systems, Man and Cybernetics, Part A : Systems and Humans, Special Issue on Collective Intelligence* **33**(5), pp.550–559.

Jones, H. L. & Snyder, M. (2001), Supervisory control of multiple robots based on a real-time strategy game interaction paradigm, *in* 'International Conference on Systems, Man, and Cybernetics', pp. 383–388.

Jong, K. D. (1975), An Analysis of the behaviour of a class of genetic algorithms, PhD thesis, University of Michigan.

Jost, C., Garnier, S., Jeanson, R., Asadpour, M., Gautrais, J. & Theraulaz, G. (2004), The embodiment of cockroach beahviour in a micro-robot, *in* '35th International Symposium On Robotics ISR', Paris-France. In CD.

Jung, D. & Zelinsky, A. (2000), 'Grounded symbolic communication between heterogeneous cooperating robots', *Autonomous Robots journal, special issue on Heterogeneous Multi-robot Systems, Kluwer Academic Publishers, Balch, Tucker and Parker, Lynne E. (eds.)* **8**(3), pp.269–292.

Keijzer, M., Morelo, J., Romero, G. & Schoenauer, M. (2001), Evolving objects : a general purpose evolutionary computation library, *in* 'Conférence internationale sur l'évolution artificielle', Université de bourgogne, le creusot, france.

Keller, L. (1997), 'Indiscriminate altruism : Unduly nice parents and siblings', *Trends in Ecology and Evolution* **12**, pp.99–103.

Khalil, W. & Dombre, E. (1999), *Modélisation, identification et commande des robots*, Hermès, Paris.

Khatib, M. (1996), Contrôle du mouvement d'un robot mobile par retour sensoriel, PhD thesis, LAAS N°96510.

Khatib, O. (1980), Commande Dynamique dans l'Espace Opérationnel des Robots Manipulateurs en Présence d'Obsacles, PhD thesis, Ecole Nationale Supérieure De L'aeronautique et de L'espace - France.

Khatib, O. (1986), 'Real-time obstacle avoidance for manipulators and mobile robots', *The International Journal of Robotics Research* **5**, pp.90–99.

Khatib, O., Yokoi, K., Chang, K., Ruspini, D., Holmberg, R., Casal, A. & Baader, A. (1995), Force strategies for cooperative tasks in multiple mobile manipulation systems, *in* 'Int. Symp. of Robotics Research', Munich.

Koren, Y. & Borenstein, J. (1991), Potential field methods and their inherent limitations for mobile robot navigation, *in* 'International Conference on Robotics and Automation', pp. 1398–1404.

Kosecká, J. & Bajcsy, R. (1993), Discrete event systems for autonomous mobile agents, *in* 'Intelligent Robotic Systems', Zakopane, pp. 90–98.

Kreithen, M. (1983), 'Orientation strategies in birds', *Behavioral Energetics : the cost of survival in vertebrates, Ohio State University press,* pp. 3–28.

Krogh, B. H. & Thorpe, C. E. (1986), Integrated path planning and dynamic steering control for autonomous vehicles, *in* 'International Conference on Robotics and Automation', pp. 1664–1669.

Kube, C. (1997), Collective Robotics : From Local Perception to Global Action, Phd thesis, Univeristy of Alberta Canada.

Kube, C. & Bonabeau, E. (2000), 'Cooperative transport by ants and robots', *Robotics and Autonomous Systems* **30**(1/2), pp.85–101.

Kube, C. & Zhang, H. (1994), Stagnation recovery behaviours for collective robotics, *in* 'International Conference on Intelligent Robots and Systems', pp. 1893–1890.

Kube, C. & Zhang, H. (1996), The use of perceptual cues in multi-robot box-pushing, *in* 'International Conference on Robotics and Automation', pp. 2085–2090.

Kube, C. & Zhang, H. (1997), 'Task modelling in collective robotics', *Autonomous Robots* **4**(1), pp.53–72.

Labidi, S. & Lejouad, W. (1993), De l'intelligence artificielle distribuée aux systèmes multi-agents, Rappot n°2004, INRIA Sopia-Antipolis. PROGRAMME 2 : Calcul symbolique, programmation et génie logiciel.

Landau, S., Doncieux, S., Drogoul, A. & Meyer, J.-A. (2002), 'SFERES : Un framework pour la conception de systèmes multi-agents adaptatifs', *Technique et Science Informatiques* **21**(4), pp.427–446.

Laumond, J.-P. (2001), *La robotique mobile*, Hermès.

Lee, A. M. & Takagi, H. (1994), 'A framework for studying the effects of dynamic crossover, mutation and population sizing in genetic algorithms', *Artificial Intelligence* p. 1011. lecture note.

Lenat, D. (1975), Beings : Knowledge as interacting experts, *in* Kauffman, ed., 'IJCAI-4', Los Angeles, pp. 126–133.

Lerman, K., Galstyan, A., Martinoli, A. & Ijspeert, A. (2001), 'A macroscopic analytical model of collaboration in distributed robotic systems', *Artificial Life* **7**(4), pp.375–393.

Leroy, S. (1998), Outils géométriques pour la planification de trajectoires de robots mobiles non holonomes, PhD thesis, LAAS N°98534.

Lichtensteiger, L. (2000), Towards optimal sensor morphology for specific tasks : Evolution of an artificial compound eye for estimating time to contact, *in* 'Proceedings of SPIE', Vol. 4196, pp. 138–146.

Lind, J. (2001), *Iterative Software Engineering for Multiagent Systems*, Springer Verlag, Volume 1994. Lecture Notes in Artificial Intelligence, Heidelberg.

Lioni, A. (1999), Auto-assemblage et transport collectif chez les Oecophylla, PhD thesis, Université Libre de Bruxelles.

Liu, J., Wu, J. & Lai, X. (1999), Analytical and experimental results on multiagent cooperative behavior evolution, *in* P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao & A. Zalzala, eds, 'Proceedings of the Congress on Evolutionary Computation', Vol. 3, pp. 1732–1739.

Lorenz, K. (1984), *Les Fondements de l'Ethologie*, Flammarion, Paris.

Lucidarme, P., Liégeois, A., Vercher, J.-L. & Bootsma, R. (2002), Un algorithme evolutionniste pour l'auto apprentissage de groupes de robots mobiles autonomes, *in* 'Proc. NSI'02', La Londe des Maures, article 56, Session Modèles 2.

Lynch, K. M. (1992), The mechanics of fine manipulation by pushing, *in* 'Proceeding of the IEEE International Conference on Robotics and Automation', Nice-France, pp. 2269–2276.

Lynch, K. M. (1999), 'Locally controllable manipulation by stable pushing', *Transactions on Robotics and Automation* **15**(2), pp.318–327.

Lynch, K. M. & Mason, M. T. (1996), 'Stable pushing : Mechanics, controllability, and planning', *International Journal of Robotics Research* **15**(6), pp.533–556.

Maes, P. (1989), The dynamics of action selection, *in* 'Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI)', Detroit, pp. 991–97.

Maes, P. (1991*a*), Adaptive action selection, *in* 'Actes de Cognitive Science Conference'.

Maes, P. (1991*b*), The Agent Network Architecture (ANA), *in* 'Actes de AAAI Spring Symposium on Integrated Intelligent Architectures', Vol. 2, pp. 115–120.

Maes, P. (1991*c*), A bottom-up mechanism for action selection in an artificial creature, *in* edited by S. Wilson & M. P. J. Arcady-Meyer, eds, 'From Animals to Animats : Proceedings of the Adaptive Behavior Conference', pp. 238–246.

Magnin, L. (1996), Modélisation et simulation de l'environnement dans les systèmes multi-agents, PhD thesis, Université Paris VI.

Mahadevan, S. & Connell, J. (1991), Scaling reinforcement learning to robotics by exploiting the subsumption architecture, *in* 'Eighth International Workshop on Machine Learning', pp. 328–337.

Marken, R. & Powers, W. (1989), 'Random-walk chemotaxis : Trial and errors as a control process', *Behavioural Neuroscience* **103**(6), pp.1348–1355.

Martinoli, A. & Mondada, F. (1995), Collective and cooperative group beahviours : Biologically inspired expermients in robotics, *in* 'Proceedings of the Fourth Symposium on Experimental Robotics ISER', Vol. 223, Stanford, USA, pp. 3–10.

Mason, M. T. (1986), 'Mechanics and planning of manipulator pushing operations', *The International Journal of Robotics Research* **5**(3), pp.53–71.

Mataric´, M. J. (1992), Minimizing complexity in controlling a mobile robot population, *in* 'IEEE International Conference on Robotics and Automation', Nice-France, pp. 830–835.

Mataric´, M. J. (1994), Reward functions for accelerated learning, *in* 'in Machine Learning : Proceedings of the Eleventh International Conference', Morgan Kaufmann, San Francisco, CA, pp. 181–189.

Mataric´, M. J. (2001), 'Learning in behavior-based multi-robot systems : Policies, models, and other agents', *Cognitive Systems Research, special issue on Multi-disciplinary studies of multi-agent learning,* **2**(1), pp.81–93.

Mataric´, M. J. (2004), Interaction and Intelligent Behavior, PhD thesis, MIT.

Mataric´, M. J., Nilsson, M. & Simsarian, K. (1995), Cooperative multi-robots box-pushing, *in* 'IEEE International Conference on Intelligent Robots and Systems', Vol. 3, Pittsburgh, PA, pp. 556–561.

Melhuish, C. (2001), *Strategies for Collective Minimalist Mobile Robots*, Vol. ERS 6, Professsional Engineering Publishing.

Michalewicz, Z. (1992), *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Berlin.

Michel, O. (2004), 'Cyberbotics ltd - webotstm : Professional mobile robot simulation', *International Journal of Advanced Robotic Systems* **1**(1), pp.39–42.

Michie, D. & Chambers, R. (1968), 'Boxes : an experiment in adaptive control', *In Machine Intelligence 2* pp. 137–152.

Minsky, M. & Papert, S. (1969), *Perceptrons*, Expanded Edition. MIT Press, 1969 revised 1988.

Müller, J., Petin, J., Morel, G., Vachon, B., Pegard, C., Brassart, E., Hutin, N., Dembelé, S., Janex, A., Morelo, B., Ravassard, J. & Bourjault, A. (1998), Conception de systèmes de transport collectif d'objets, *in* 'Premières Journées du Pôle Microrobotique', pp. 61–66.

Muller, P.-A. & Gaertner, N. (2000), *Modélisation objet avec UML*, Eyrolles.

Muñoz, A. (2003), Coopération située : une approche constructiviste de la conception de colonies de robots, PhD thesis, Université Pierre et Marie Curie, Paris VI.

Nakamura, M. & Kurumatani, K. (1997), *Langton,C., Shimohara, K. (eds.), Artificial Life V*, The MIT Press, chapter Formation Mechanism of Pheromone Pattern and Control of Foraging Behavior in an Ant Colony Model, pp. 67–74.

Noda, I., Matsubara, H., Hiraki, K. & Frank, I. (1998), Soccer server : a tool for research on multi-agent systems, *in* 'Applied Artificial Intelligence'.

Noreils, F. R. (1993), 'Toward a robot architecture integrating cooperation between mobile robots : Application to indoor environment', *The International Journal of Robotics Research* **12**(1), pp.79–98.

Ono, N. & Fukuta, Y. (1996), A cmac-based reinforcement learning approach to the multi-agent block-pushing problems, *in* 'MECATRONICS'.

Ozaki, K., Asama, H. & Endo, I. (1997), 'Distributed and cooperative object pushing by multiple mobile robots based on communication', *Advanced Robotics* **11**(5), pp.501–517.

Parker, L. E. (1994), Heterogeneous Multi-Robot Cooperation, PhD thesis, Massachusetts Institute of Technology Ph.D. Dissertation.

Parker, L. E. (1997), 'L-Alliance : Task-oriented multi-robot learning in behavior-based systems', , *Advanced Robotics, Special Issue on Selected Papers from IROS '96* **11**(4), pp.305–322.

Parker, L. E. (1998), 'Alliance : An architecture for fault tolerant multi-robot cooperation', *IEEE Transactions on Robotics and Automation* **14**(2), pp.220–240.

Parker, L. E. (2000), 'Lifelong adaptation in heterogeneous multi-robot teams : Response to continual variation in individual robot performance', *Autonomous Robots* **8**(3), pp.239–267.

Parker, L. E. (2001), 'Evaluating success in autonomous multi-robot teams : Experiences from Alliance architecture implementations', *Journal of Theoretical and Experimental Artificial Intelligence* **13**, pp.95–98.

Passera, L. (1984), *L'Organisation Sociale des Fourmis*, Privat, Toulouse.

Pfeifer, R. (2000), On the role of morphology and materials in adaptive behavior, *in* D. F. H. R. In : J.-A. Meyer, A. Berthoz & S. W. (eds.), eds, 'From animals to animats 6. Proc. of the 6th Int. Conf. on Simulation of Adaptive Behavior. Cambridge, Mass. : MIT Press', pp. 23–32.

Pirjanian, P. (1999), 'Multiple objective behavior-based control', *Journal of Robotics and Autonomous Systems, Special Issue*.

Pirjanian, P. & Mataric´, M. (1999), 'Multiple objective vs. fuzzy behavior coordination', Lecture Notes in Computer Science on Fuzzy Logic Techniques for Autonomous Vehicle Navigation, eds. D. Drainkov and A. Saffiotti.

Premvuti, S. & Yuta, S. (1990), Consideration on the cooperation of multiple autonomous mobile robots, *in* 'International Workshop on Intelligent Robots and Systems', Ikabari-Japan, pp. 59–63.

Pruski, A. (1996), *Robotique mobile–la planification de trajectoire*, Hermes.

Qutub, S. (1998), Un paradigm générique pour la navigation coopérative de robots mobiles autonomes, PhD thesis, LAAS, France.

Ram, A., Arkin, R., Boone, G. & Pearce, M. (1994), 'Using genetic algorithms to learn reactive control parameters for autonomous robotic navigation', *Adaptive Behavior* **2**(3), pp.277–305.

Reggiani, M., Mazzoli, M. & Caselli, S. (2002), An Experimental Evaluation of Collision Detection Packages for Robot Motion Planning, *in* 'Conference on Intelligent Robots and Systems', Lausanne, Switzerland.

Reynolds, C. (1987), 'Flocks, herds, and schools : A distributed behavioral model', *Computer Graphics* **21**(4), pp.25–34.

Robert, F. (1996), Coopération multi-robots par insertion incrémentale de plans, Phd thesis, LAAS.

Rosenblatt, J. & Thorpe, C. (1995), Combining multiple goals in a behavior-based architecture, *in* 'IROS'95', Vol. 1, pp. 136–141.

Rumelhart, D., Hinton, G. & Williams, R. (1986), 'Learning internal representations by error propagation', *Parallel Distributed Processing, D. Rumelhart and J. McClelland Eds. Cambridge : MIT Press* **1**, pp.318–362.

Rus, D., Donald, B. & Jennings, J. (1995), Moving furniture with teams of autonomous robots, *in* 'IROS'95', Vol. 1, pp. 235–242.

Russell, S. & Norving, P. (1995), *Artificial Intelligence : a Modern Approach*, Prentice-Hall.

Saffiotti, A., Ruspini, E. & Konolige, K. (1993), Robust execution of robot plans using fuzzy logic, *in* Springer-Velag, ed., 'in Fuzzy Logic in Artificial Intelligence : IJCAI'93 Workshop', Chambéry-France, pp. 24–37.

Sasaki, J., Ota, J., Yoshida, E., Kurabyashi, D. & Arai, T. (1995), Cooperating grasping of a large object by multiple mobile robots, *in* 'International conference On Robotics and Automation', pp. 1205–1210.

Searle, J. (1980), 'Minds, brains and programs', *Behavioral and Brain Sciences* **3**, pp.417–423.

Simonin, O. (2001), Le modèle satisfaction-altruisme : coopération et résolution de conflits entre agents situés réactifs, application à la robotique, PhD thesis, Univesité Montpellier II.

Simsarian, K. T. & Mataric´, M. J. (1995), Learning to cooperate using two six-legged mobile robots, *in* 'Third European Workshop of Learning Robots', Heraklion, Crete, Greece,, pp. 453–462.

Singh, K. & Fujimura, K. (1993), Map making by cooperating mobile robots, *in* 'International Conference On Robotics And Automation', Los Alamitos, pp. 254–258.

Steels, L. (1990), Cooperation between distributed agents through self-organisation, *in* Y. Demazeau & J.-P. Müller, eds, 'Decentralized A.I. : Proc. of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Cambridge, England', North-Holland, Amsterdam, pp. 175–196.

Stilwell, D. & Bay, J. (1993), Toward the development of a material transport system using swarms of ant-like robots, *in* 'International Conference on Robotics and Automation', pp. 766–771.

Sty, K. (2001), Using situated communication in distributed autonomous mobile robots, *in* 'Seventh Scandinavian Conference on Artificial Intelligence (SCAI01)'.

Sugawara, K. & Watanabe, T. (2002), Swarming robots - foraging behavior of simple multi-robot system, *in* 'IROS'02', Lausanne, Switzerland, pp. 2702–2707.

Sugihara, K. & Suzuki, I. (1990), Distributed motion coordination of multiple mobile robots, *in* 'Proceedings of the 5th IEEE International Symposium on Intelligent Control', pp. 138–143.

Sutton, R. (1988), 'Learning to predict by the methode of temporal differences', *Machine Learning* **3**, pp.9–44.

Sutton, R. S. & Barto, A. G. (1998), *Reinforcement Learning : An Introduction Richard*, MIT Press, Cambridge.

Talbi, E.-G. (1999), Métaheuristiques pour l'optimisation combinatoire multi-objectifs, Technical report, CNET (France Telecom).

Tangamchit, P., Dolan, J. & Khosla, P. (2002), The necessity of average rewards in cooperative multi-robot learning, *in* 'International Conference On Robotics And Automation'.

Theraulaz, G. & Bonabeau, E. (1995), Coordination in distributed building, Science, 269, pp. 686–688.

Tournassoud, P. (1992), *Planification et contrôle en robotique – Application aux robots mobiles et manipulateurs*, Hermes.

Url-actionneur (2005), '`http://asl.epfl.ch/research/systems/Alice/documents//Amotors.pdf`'.

Url-capteurIR (2005), '`http://asl.epfl.ch/research/systems/Alice/documents/Asensors`'.

Url-communication (2005), 'http://asl.epfl.ch/research/systems/Alice/documents/A_IRcomm'.

Url-compilateur (2005), 'http://www.ccsinfo.com/picc.shtml'.

Url-programmation (2005), 'http://asl.epfl.ch/research/systems/Alice/documents/Aprogr'.

Vaughan, R. T., Stoy, K., Sukhatme, G. S. & Mataric, M. J. (2000*a*), Blazing a trail : insect-inspired resource transportation by a robot team, *in* 'Proc. Int. Symp. Distributed Autonomous Robot Systems'.

Vaughan, R. T., Støy, K., Sukhatme, G. S. & Mataric, M. J. (2000*b*), Whistling in the dark : cooperative trail following in uncertain localization space, *in* 'Proceedings of the fourth international conference on Autonomous agents'.

Velghe, J. (2003), Localisation et suivi d'objets par vision, Diplôme d'Études Approfondies - DEA-IAP, Laboratoire d'Automatique de Besançon.

Ávila, G. R., Guisset, J. & Deneubourg, J. (2003), 'Synchronization in light-controlled oscillator', *Elsevier, in press.*

Vladimir, E. (1996), The role of selection in genetic algorithms, Technical Report FIT-TR-1996-04.

Volpe, R., Balaram, J., Ohm, T. & Ivlev, R. (1997), 'Rocky 7 : A next generation mars rover prototype', *Advenced Robotics* **11**(4), pp.341–358.

Wall, M. (1995), 'Overview of galib : http ://lancet.mit.edu/ga'.

Wang, P. (1991), 'Navigation strategies for multiple autonomous mobile robots moving in formation', *Journal of Robotics Systems* **8**(2), pp.177–195.

Wehner, R. (1987), ''matched filters' - neural models of the external world', *Journal of Comparative Physiology A - Sensory Neural and Behavioral Physiology* **161**, pp.511–531.

Woods, G. W. (1997), A Hybrid Genetic Algorithm that Adapts to Binary and Real Coded Operators, PhD thesis, Departement of Computer Science RMIT.

Woolridge, M. (2000), On the sources of complexity in agent design - in applied artificial intelligence, Vol. 14, pp. 623–644.

Yamada, S. & Saito, J. (2001), 'Adaptive action selection without explicit communication for multirobot box-pushing', *IEEE Transaction On Systems, Man And Cybernetics-Part C : Application And Reviews* **31**(3), pp.398–404.

Yamaguchi, H., Arai, T. & Beni, G. (2001), 'A distributed control scheme for multiple robotic vehicles to make group formations', *Robotics and Autonomous Systems* **36**, pp.125–147.

Yen, J. & Pfluger, N. (1995), 'A fuzzy logic based extension to Payton and Rosenblatt's command fusion method for mobile robot navigation', *IEEE Transaction on Systems, Man, and Cybernetics* **25**(6), pp.971–978.

Zadeh, L. A. (1965), 'Fuzzy sets', *Information and Control* **8**, pp.338–353.

Zapata, R. & Lepinay, P. (1994), 'Reactive behaviors of fast mobile robots', *Journal of Robotic Systems* **11**(1), pp.13–20.

Zitzler, E. & Thiele, L. (1998), Multiobjective optimization using evolutionary algorithms - a comparative case study, *in* 'Proceedings of the 5th International Conference in Parallel Problem Solving from Nature, PPSN5', pp. 292–301.

Résumé : Contrôler un système multi-robots hautement dynamique au sein duquel évolue un grand nombre d'entités autonomes réactives est un challenge à la fois scientifique et technologique en plein essor. En effet, ceci exige non seulement d'utiliser des entités robotiques les plus élémentaires possibles mais nécessite également au niveau du contrôle, de s'éloigner davantage des conceptions centralisées et cognitives. La démarche consiste à focaliser la conception du contrôle sur l'individu élémentaire constituant le système multi-robots en prenant en compte les différentes interactions locales de cet individu avec les autres entités robotiques avec lesquelles il est censé coopérer. Des effets de masse maîtrisés peuvent être ainsi obtenus et vont permettre d'augmenter à la fois la vitesse, la flexibilité et la robustesse d'exécution des tâches complexes entreprises. Les travaux de recherche présentés dans ce mémoire partent du principe d'une conception ascendante (Bottom-Up) des architectures de contrôle et ce afin de briser la complexité inhérente aux systèmes multi-robots. Plus spécifiquement, nous proposons un Processus de Sélection d'Action Hiérarchique appelé **PSAH** qui permet à l'échelle du robot de coordonner l'activité d'un ensemble de primitives élémentaires (comportements) d'une manière hiérarchique et flexible, et à l'échelle du groupe de robots d'atteindre une coordination entre robots favorisant des buts globaux. Les performances du PSAH ont été améliorées par la suite via l'adjonction d'un mécanisme de fusion d'actions approprié conduisant à un nouveau processus de sélection appelé **PSAHH** (PSAH-Hybride). Les formalismes des algorithmes génétiques ont été utilisés par la suite pour proposer une méthodologie permettant l'obtention des paramètres prépondérants pour le fonctionnement du PSAHH. La validation des résultats s'est effectuée au travers d'expérimentations sur des mini-robots ALICE et plus largement sur un ensemble d'études statistiques réalisées sur un grand nombre de données obtenu grâce au simulateur ***MiRoCo*** (*Mini-Robotique Collective*). Ce simulateur a été conçu et développé dans le cadre de nos travaux de thèse dans le but de simuler d'une manière précise et rigoureuse des systèmes multi-robots à forte dynamique d'interaction.

Mots-clés : Robotique mobile coopérative, Architecture de contrôle comportementale, Optimisation paramétrique, Algorithme génétique, Simulation multi-robots.

Abstract : The control of a highly dynamic multi-robots system where evolves a large number of reactive autonomous entities constitutes a promising scientific and technological challenge. Indeed, this requires to use robotics entities as elementary as possible, but also requires in term of control, to move more away from centralized and cognitive designs. The approach consists in focusing the control design more on the elementary individual constituting the multi-robots system while taking into account its different local interactions with other robotics entities, which are supposed to cooperate between them. Mastered mass effects can be gotten thus and will permit to enhance at the same time the speed, flexibility and the robustness of the executed complex tasks. The works of research presented in this thesis report start from the principle of a Bottom-Up architectures of control in order to break the inherent complexity of the multi-robots systems. More specifically, we propose a Hierarchical Action Selection Process named **HASP** which permits at the scale of the robot to coordinate the activity of a set of elementary primitives (behaviors) in a hierarchical and flexible manner, and at the scale of the group of robots, allows a coordination promoting global goals. The performances of the HASP were improved after via the addition of an appropriate mechanism of fusion of actions leading thus to the **HHASP** (Hybrid-HASP). The formalisms of the genetic algorithms have been used afterwards to propose a methodology, which allow to obtain preponderant parameters for the working of the HHASP. The validation of the results was made through experimentations with mini-robots ALICE and more intensively on statistical studies achieved on a big number of data gotten thanks to ***MiRoCo*** (*Mini-Robotique Collective*) simulator. This simulator have been designed and developed in the framework of our thesis works in order to simulate accurately and rigorously multi-robots systems, which have a strong dynamic of interaction.

Key-words : Cooperative mobile robotics, Behavioral architecture of control, Parameters optimization, Genetic algorithms, Multi-robots simulation.