N° d'ordre 95 ISAL0106 année 1995

THESE

présentée

Devant L'INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE LYON

pour obtenir

LE GRADE DE DOCTEUR

FORMATION DOCTORALE : Ingénierie Informatique

par

Guillaume BESLON
(ingénieur INSA)

CONTROLE SENSORI-MOTEUR PAR RESEAUX NEUROMIMETIQUES MODULAIRES

- APPROCHE POUR LE PILOTAGE REACTIF EN ATELIER FLEXIBLE -

Soutenue le 30 / 11 / 1995 devant la Commission d'Examen

Jury :
- Melle. Frédérique Biennier
- Mr. Jacques Brémont (Rapporteur)
- Mr. Philippe Coiffet (Rapporteur)
- Mr. Michel Cosnard
- Mr. Joël Favrel
- Mr. Jacques Ferber (Rapporteur)
- Mr. Alain Jutard (Rapporteur)
- Mr. Christian Jutten
- Mr. Yves Martinez

SEPTEMBRE 1995

INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE LYON

Directeur : J. ROCHAT

Professeurs :

S.	**AUDISIO**	PHYSICOCHIMIE INDUSTRIELLE
J.C.	**BABOUX**	GEMPPM[1]
J.	**BAHUAUD**	MECANIQUE DES SOLIDES
B.	**BALLAND**	PHYSIQUE DE LA MATIERE
D.	**BARBIER**	PHYSIQUE DE LA MATIERE
G.	**BAYADA**	EQUIPE MOD. MATH. CALC. SC.
C.	**BERGER (Mlle)**	CETHIL/EQ. PHYSIQUE INDUST.
M.	**BETEMPS**	AUTOMATIQUE INDUSTRIELLE
J.M.	**BLANCHARD**	LCPAE[2]
C.	**BOISSON**	VIBRATION ACOUSTIQUES
M.	**BOIVIN**	MECANIQUE DES SOLIDES
H.	**BOTTA**	EQUIPE DEVELOPPEMENT URBAIN
G.	**BOULAYE**	L.I.S.I.[3]
J.	**BRAUG**	CETHIL/EQ. EQUIPEMENT DE L'HABITAT
M.	**BRUNET**	MECANIQUE DES SOLIDES
J.C.	**BUREAU**	THERMOCHIMIE MINERALE
J.P.	**CHANTE**	CPA[4]
B.	**CHOCAT**	METHODES
B.	**CLAUDEL**	LCPAE[2]
M.	**COUSIN**	BETONS ET STRUCTURES
L.	**CRONENBERGER**	CHIMIE BIOLOGIQUE
M.	**DIOT**	THERMOCHIMIE MINERALE
A.	**DOUTHEAU**	CHIMIE ORGANIQUE
J.C.	**DUPUY**	PHYSIQUE DE LA MATIERE
H.	**EMPTOZ**	RECONNAISSANCE DE FORMES ET VISION
C.	**ESNOUF**	GEMPPM[1]
G.	**FANTOZZI**	GEMPPM[1]
J.	**FAUCHON**	C.A.S.M.[5]
J.	**FAVREL**	PRISMa[6]
G.	**FERRARIS-BESSO**	MECANIQUE DES STRUCTURES
Y.	**FETIVEAU**	GENIE ELECTRIQUE ET FERROELECTRICITE
L.	**FLAMAND**	MECANIQUE DES CONTACTS
P.	**FLEISCHMANN**	GEMPPM[1]

[1] Groupe d'Etude Métallurgie Physique et Physique des Matériaux

[2] Laboratoire de Chimie Physique Appliquée et Environnement

[3] Laboratoire d'Ingénierie des Systèmes d'Information

[4] Composants de Puissance et Applications

[5] Conception et Analyse des Systèmes Mécaniques

[6] Equipe de Productique et Informatique des Systèmes Manufacturiers

A.	**FLORY**	L.I.S.I.[2]
R.	**FOUGERES**	GEMPPM[1]
L.	**FRECON**	D.E.L.I.A.[3]
R.	**GAUTHIER**	PHYSIQUE DE LA MATIERE
M.	**GERY**	CETHIL/EQ. EQUIPEMENT DE L'HABITAT
G.	**GIMENEZ**	CREATIS[4]
P.	**GOBIN**	GEMPPM[1]
P.	**GONNARD**	GENIE ELECTRIQUE ET FERROELECTRICITE
R.	**GOUTTE**	CREATIS[4]
G.	**GRANGE**	GENIE ELECTRIQUE ET FERROELECTRICITE
G.	**GUENIN**	GEMPPM[1.]
G.	**GUILLOT**	PHYSIQUE DE LA MATIERE
A.	**GUINET**	PRISMa[5]
C.	**GUITTARD**	D.E.L.I.A.[3]
J.L.	**GUYADER**	VIBRATIONS-ACOUSTIQUES
J.P.	**GUYOMAR**	GENIE ELECTRIQUE ET FERROELECTRICITE
J.M.	**JOLION**	RECONNAISSANCE DE FORMES ET VISION
J.	**JOUBERT**	GENIE MECANIQUE
J.F.	**JULLIEN**	BETONS ET STRUCTURES
A.	**JUTARD**	AUTOMATIQUE INDUSTRIELLE
R.	**KASTNER**	GEOTECHNIQUE
H.	**KLEIMANN**	GENIE ELECTRIQUE ET FERROELECTRICITE
J.	**KOULOUMDJIAN**	L.I.S.I.[2]
M.	**LAGARDE**	CHIMIE BIOLOGIQUE
M.	**LALANNE**	MECANIQUE DES STRUCTURES
A.	**LALLEMAND**	CETHIL/EQ. ENERGETIQUE ET THERMIQUE
M.	**LALLEMAND (Mme)**	CETHIL/EQ. ENERGETIQUE ET THERMIQUE
P.	**LAREAL**	GEOTECHNIQUE
A.	**LAUGIER**	PHYSIQUE DE LA MATIERE
Ch.	**LAUGIER**	PHYSIOLOGIE ET PHARMACODYNAMIE
P.	**LEJEUNE**	G.M.M.O.[6]
C.	**LESUEUR**	VIBRATIONS-ACOUSTIQUES
A.	**LUBRECHT**	MECANIQUE DES CONTACTS
Y.	**MARTINEZ**	L3i[7]
H.	**MAZILLE**	PHYSICOCHIMIE INDUSTRIELLE
P.	**MERLE**	GEMPPM[1]
J.	**MERLIN**	GEMPPM[1]
J.P.	**MILLET**	PHYSICOCHIMIE INDUSTRIELLE

[1] Groupe d'Etude Métallurgie Physique et Physique des Matériaux

[2] Laboratoire d'Ingénierie des Systèmes d'Information

[3] Développement et Langages Informatiques Avancés

[4] Centre de Recherche et d'Application en Traitement d'Image et du Signal

[5] Equipe de Productique et Informatique des Systèmes Manufacturiers

[6] Génétique Moléculaire des Micro-Organismes

[7] Laboratoire Ingéniérie Informatique Industrielle

M.	**MIRAMOND**	METHODES
N.	**MONGEREAU**	GEOTECHNIQUE
R.	**MOREL**	MECANIQUE DES FLUIDES
P.	**NARDON**	BIOLOGIE APPLIQUEE
A.	**NAVARRO**	LCPAE[1]
M.	**OTTERBEIN**	LCPAE[1]
J.P.	**PASCAULT**	MATERIAUX MACROMOLECULAIRES
J.	**PERA**	MATERIAUX MINERAUX
G.	**PERACHON**	THERMOCHIMIE MINERALE
M.	**PERDRIX**	CREATIS[2]
J.	**PEREZ**	GEMPPM[3]
P.	**PINARD**	PHYSIQUE DE LA MATIERE
J.M.	**PINON**	L.I.S.I.[4]
D.	**PLAY**	C.A.S.M.[5]
J.	**POUSIN**	EQUIPE MOD. MATH. CALC. SC.
P.	**PREVOT**	GRACIMP[6]
R.	**PROST**	CREATIS[2]
M.	**RAYNAUD**	CETHIL/EQUIPE GMC
J.M.	**REYNOUARD**	BETONS ET STRUCTURES
J.	**ROBERT-BAUDOUY (Mme)**	LCPAE[1]
D.	**ROUBY**	GEMPPM[3]
P.	**RUBEL**	L.I.S.I.[4]
J.F.	**SACUDURA**	CETHIL/EQUIPE GMC
H.	**SAUTEREAU**	MATERIAUX MACROMOLECULAIRES
S.	**SCARVADA**	AUTOMATIQUE INDUSTRIELLE
D.	**THOMASSET**	AUTOMATIQUE INDUSTRIELLE
M.	**TROCCAZ**	GENIE ELECTRIQUE ET FERROELECTRICITE
R.	**UNTERREINER**	CREATIS[2]
J.	**VERON**	LCPAE[1]
A.	**VINCENT**	GEMPPM[3]
G.	**VIGIER**	GEMPPM[3]
P.	**VUILLERMOZ**	PHYSIQUE DE LA MATIERE

[1] Laboratoire de Chimie Physique Appliquée et Environnement

[2] Centre de Recherche et d'Application en Traitement d'Image et du Signal

[3] Groupe d'Etude Métallurgie Physique et Physique des Matériaux

[4] Laboratoire d'Ingénierie des Systèmes d'Information

[5] Conception et Analyse des Systèmes Mécaniques

[6] Groupe de Recherche en Apprentissage, Coopération et Interfaces Multimodales

Directeurs de recherche C.N.R.S. :

Y. BERTHIER	MECANIQUE DES CONTACTS
P. CLAUDY	THERMOCHIMIE MINERALE
M. MURAT	GEMPPM[1]
A. NOUAILHAT	PHYSIQUE DE LA MATIERE
M.A. MANDRAND (Mme)	G.M.M.O.[2]

Directeurs de recherche I.N.R.A. :

G. BONNOT	BIOLOGIE
S. GRENIER	BIOLOGIE
Y. MENEZO	BIOLOGIE

Directeurs de recherche I.N.S.E.R.M. :

A.F. PRIGENT (Mme)	CHIMIE BIOLOGIQUE
N. SARDA (Mme)	CHIMIE BIOLOGIQUE

[1] Groupe d'Etude Métallurgie Physique et Physique des Matériaux

[2] Génétique Moléculaire des Micro-Organismes

Mes remerciements vont, en tout premier lieu, à Mademoiselle Frédérique BIENNIER qui a dirigé ces recherches et qui a su être encourageante dans les moments difficiles et exigeante lorsque les circonstances le demandaient. Ce travail lui doit beaucoup et je lui en témoigne ici toute ma gratitude.

Je tiens à remercier aussi Monsieur Joël FAVREL et l'ensemble de l'équipe PRISMa pour m'avoir chaleureusement accueilli parmi eux, avec une attention plus particulière pour Anne LEGAIT dont la gentillesse et la serviabilité n'auront pas décru tout au long de ces trois années. J'adresse aussi tous mes remerciements à Monsieur Yves MARTINEZ, directeur du département informatique de l'INSA de Lyon qui a accepté de participer au jury, ainsi qu'à Monsieur Patrick PREVOT, directeur de la formation doctorale « Ingénierie Informatique ».

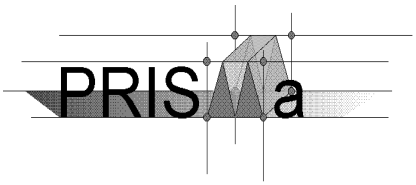
Les quatre rapporteurs, Messieurs Jacques BREMONT, Professeur à l'Université Henri Poincaré/nancy I, Philippe COIFFET, Directeur de Recherche au CNRS, Jacques FERBER, Professeur à l'Université Pierre et Marie Curie/Paris VI et Alain JUTARD, Professeur à l'INSA de Lyon, ont largement contribué, par leurs remarques respectives, à l'amélioration de ce mémoire. Qu'ils en soient ici chaleureusement remerciés ainsi que pour leur participation au jury.

Mes remerciements vont aussi à Messieurs Michel COSNARD, Professeur à l'Ecole Normale Supérieure de Lyon et Christian JUTTEN, Professeur à l'Université Joseph Fourier de Grenoble, membres du jury, dont j'ai apprécié l'intérêt pour ce travail et dont les remarques m'ont permis de préciser certains points du mémoire.

Enfin, je tiens à exprimer toute ma gratitude à l'ensemble des personnes qui, de près ou de loin, ont contribué à ce travail, plus particulièrement Géraldine BENAD, Blandine et Fabrice CETRE et Bertrand BLANQUART qui ont largement contribué à la bonne tenue de ce mémoire, ainsi que l'ensemble des membres du département informatique de l'INSA de Lyon et de l'équipe « Perception et RAISonnement dans les Systèmes Industriels et Humains » du Centre de Recherche en Automatique de Nancy. Sans eux rien n'aurait été possible.

Institut **N**ational des **S**ciences **A**ppliquées
de
Lyon

PRoductique et **I**nformatique des **S**ystèmes **Ma**nufacturiers



CONTROLE SENSORI-MOTEUR PAR RESEAUX NEUROMIMETIQUES MODULAIRES

- Approche Pour le Pilotage Réactif en Atelier Flexible -

par

Guillaume BESLON

CONTROLE SENSORI-MOTEUR PAR RESEAUX NEUROMIMETIQUES MODULAIRES

- Approche Pour le Pilotage Réactif en Atelier Flexible -

Guillaume BESLON

SOMMAIRE

CHAPITRE I : Introduction 22

CHAPITRE II : La Fonction Production : Nouveaux Besoins, Nouveaux Moyens 26

I.1. La fonction production 29

II.2. Le nouvel environnement de la production 30

II.2.1. Standardisation et variété 30
II.2.2. La réactivité : vers une nouvelle forme de concurrence 31
II.2.2.1. Délais de mise à disposition/Délais de conception 32
II.2.2.2. Les flux tendus et le Juste à Temps 33
II.2.2.3. L'ingénierie concourante 34
II.2.3. De nouveaux impératifs pour l'appareil de production 36
II.2.3.1. Flexibilités 36
II.2.3.2. Flexibilité 39
II.2.3.3. Réactivité 40
II.2.3.4. Robustesse 40

II.3. Vers des structures de production flexibles et réactives 41

II.3.1. L'atelier flexible 42
II.3.1.1. Définition 42
II.3.1.2. Constituants 43
II.3.2. Les Architectures de pilotage 44
II.3.2.1. Les architectures centralisées 44
II.3.2.2. Les architectures hiérarchisées 46
II.3.2.3. Décentraliser pour mieux réagir 47
II.3.3. Des contrôleurs locaux intelligents et adaptatifs 52
II.3.3.1. Un environnement plus complexe 52
II.3.3.2. Des contrôleurs flexibles et réactifs 53
II.3.4. De nouvelles techniques au service de la production 54
II.3.4.1. Le contrôle sensori-moteur 54
II.3.4.2. Apprentissage et adaptation : le connexionnisme et le Neurocontrôle 55

II.3.5. Un cas particulier : le système de transport 55
II.3.5.1. Caractéristiques générales du système de transport 55
II.3.5.2. Un système « distribuable » 57
II.3.5.3. Simulation et résultats 58
II.3.5.4. Analyse et extensions 63

II.4. Conclusion 66

CHAPITRE III : Le Pilotage Réactif 68

III.1. Introduction 70

III.2. Origine 71

III.2.1. Les travaux antérieurs 71
III.2.1.1. Cybernétique 71
III.2.1.2. ROAR 72
III.2.2. Une remise en cause des modèles symboliques 74
III.2.3. Du raisonnement au comportement... 75

III.3. Les structures du contrôle réactif 76

III.3.1. Définitions 76
III.3.1.1. Contrôle réactif et intelligence 76
III.3.1.2. Emergence 78
III.3.1.3. « Mise en situation » 79
III.3.1.4. « Mise en Oeuvre » 80
III.3.2. L'architecture « Subsumption » 80
III.3.2.1. Principes 80
III.3.2.2. Organisation pratique 81
III.3.2.3. Applications et variantes 84
III.3.3. AuRA et les modèles vectoriels 84
III.3.3.1. Principe 84
III.3.3.2. L'architecture 85
III.3.4. D'autres modèles 87
III.3.4.1. « Neuro-Ethologie Calculatoire » 87
III.3.4.2. REX, Gapps et Ruler 87
III.3.4.3. « Internalized Plans » 88
III.3.4.4. Systèmes hybrides 88

III.4. Des agents autonomes aux sociétés multi-agents 89

III.5. Application à la productique 92

III.5.1. Assemblage 92
III.5.2. Manutention 93
III.5.3. Usinage 93
III.5.4. Transitique 94
III.5.5. Applications péri-productives 95

III.6. Contrôle réactif et apprentissage 95

III.6.1. Motivations 95
III.6.2. Apprentissage par renforcement 96
III.6.3. Approche statistique 97
III.6.4. Approche génétique 98
III.6.5. Approche neuromimétique 99

III.7. Conclusion 100

CHAPITRE IV : Architectures Neuronales pour le Contrôle de Processus _____ 102

IV.1 Introduction _____ 104

IV.2. Les acteurs de l'apprentissage connexionniste _____ 104

IV.2.1. Caractéristiques « pédagogiques » _____ 105
IV.2.2. Caractéristiques dynamiques _____ 106
IV.2.3. Etendue _____ 107
IV.2.4. Conclusion _____ 107

IV.3. Organisation générale des réseaux de neurones _____ 107

IV.3.1. Introduction _____ 107
IV.3.2. Organisation microscopique : la cellule neuronale _____ 108
IV.3.2.1. Le neurone biologique 108
IV.3.2.2. Le neurone formel 109
IV.3.3. Organisation macroscopique : la topologie _____ 113
IV.3.3.1. Organisation des fonctions dans le cortex 113
IV.3.3.2. Topologie des réseaux de neurones formels 113
IV.3.4. Organisation interne de l'information _____ 116
IV.3.4.1. Un cas particulier : les représentations symboliques 117
IV.3.4.2. Cas des réseaux en couches 118
IV.3.4.3. Le cas des réseaux récurrents 118
IV.3.5. Organisation temporelle : dynamique _____ 118
IV.3.5.1. Les lois d'activation 119
IV.3.5.2. Stabilité et convergence des réseaux de neurones 120
IV.3.6. Apprentissage _____ 121
IV.3.6.1. Apprentissage supervisé 121
IV.3.6.2. Apprentissage non-supervisé et auto-organisation 122
IV.3.6.3. Apprentissage par renforcement 123
IV.3.6.4. Apprentissage « en ligne » et apprentissage « hors-ligne » 123
IV.3.7. Différents modèles de réseaux _____ 123

IV.4. Architectures pour le neurocontrôle _____ 124

IV.4.1. L'identification pour le contrôle de processus _____ 124
IV.4.1.1. Identification d'un processus 124
IV.4.1.2. Identification d'un contrôleur, apprentissage de la commande motrice 125
IV.4.1.3. Identification de la dynamique inverse 127
IV.4.1.4. « Specialized-Inverse-Learning » 128
IV.4.1.5. L'identification comme complément des méthodes classiques 129
IV.4.2. La modélisation adaptative et la rétropropagation à travers le temps _____ 129
IV.4.3. L'apprentissage par renforcement et la critique adaptative _____ 131
IV.4.3.1. Modèle général 132
IV.4.3.2. « Stochastic-Real-Valued-Control » 134
IV.4.3.3. Critique et modélisation adaptative 138
IV.4.3.4. L'adaptation de la critique : TD-Learning 139

IV.5. Conclusion : quels modèles de contrôleurs pour le contrôle réactif _____ 140

CHAPITRE V : Vers des Modèles Neuronaux pour le Contrôle Réactif *144*

V.1. Introduction **146**

V.2. Approches neuronales **146**

V.2.1. Des modèles d'inspiration neurobiologique 147
V.2.1.1. Neuroéthologie calculatoire 147
V.2.1.2. L'intelligence Visuo-motrice 149
V.2.1.3. Les modèles de la colonne corticale 150
V.2.2. Modèles coopératifs séquentiels 151
V.2.3. Modèles de supervision 153
V.2.3.1. Modèles hiérarchiques 153
V.2.3.2. Le réseau « METAPI » 154
V.2.3.3. Modèle Experts/Gating 155
V.2.3.4. La méthode OARE 156
V.2.4. Modèles topologiques 157
V.2.4.1. La méthode LASSO 158
V.2.4.2. Association sensori-motrice par carte de Kohonen 159
V.2.4.3. Construction Epigénétique 159
V.2.4.4. Les organisations fractales 160
V.2.5. Conclusion 161

V.3. L'approche ascendante : une nouvelle voie vers la modularité **162**

V.3.1. Une nouvelle définition de l'émergence 164
V.3.2. Les assemblées de neurones 165
V.3.2.1. Aspect neurobiologique des assemblées de neurones 166
V.3.2.2. Un équivalent connexionniste 167

V.4. Conclusion : vers une organisation « Neuro-Réactive » **168**

CHAPITRE VI : Une Solution Connexionniste Récurrente et Modulaire *172*

VI.1. Introduction **174**

VI.2. Les modèles biologiques : spécialisation et synchronie **174**

VI.2.1. Introduction : le codage impulsionnel de l'influx nerveux 175
VI.2.2. Codage impulsionnel et information temporelle 176
VI.2.3. Caractérisation temporelle des assemblées de neurones 178
VI.2.4. Conclusion : synchronisation et apprentissage 182

VI.3. Le mode ordonnancé : vers un traitement local de l'information **183**

VI.3.1. Aspect temporel des réseaux de neurones 183
VI.3.2. Le mode ordonnancé 184
VI.3.2.1. Algorithme 185
VI.3.2.2. Calcul du vecteur de déséquilibre 186
VI.3.2.3. Architecture 187
VI.3.3. Dynamique du mode ordonnancé 188
VI.3.3.1. Points fixes 189
VI.3.3.2. Convergence 190
VI.3.3.3. Conclusion : vers des dynamiques complexes 192
VI.3.4. Apprentissage et mode ordonnancé 193
VI.3.4.1. Principe : un apprentissage Hebbien 193
VI.3.4.2. Adaptation au mode ordonnancé 196
VI.3.4.3. Le cas des états non-répondants 197

VI.4. Premiers tests et développements 201

VI.4.1. Introduction 201
VI.4.2. Caractéristiques générales des réseaux utilisés pour les tests 202
VI.4.2.1. Environnement de programmation 202
VI.4.2.2. Architecture 202
VI.4.3. Apprentissage du OU EXclusif 202
VI.4.4. Classification : la reconnaissance de couleurs 205
VI.4.5. Apprentissage de séquences booléennes 209
VI.4.6. Neurocontrôle : le pendule inverse 212
VI.4.7. Analyse 220
VI.4.8. Vers d'autres algorithmes... 222
VI.4.8.1. Vers d'autres algorithmes d'apprentissage 222
VI.4.8.2. Toujours un peu plus loin... 223

VI.5. Perspectives : vers l'intégration 224

CHAPITRE VII : Conclusion 226

CHAPITRE VIII : Références Bibliographiques 232

ANNEXES : 252

ANNEXE A1 : Résultats de la simulation du système de transport 256

A1.1. Choix de la loi de gestion 256
A1.2. Influence du nombre de robots 257
A1.3. Influences relatives du nombre de robots et du temps de process maximum 257
A1.4. Production d'un atelier théorique chargé à 100% 259

ANNEXE A2 : Les principaux Modèles Connexionnistes 262

A2.1. Les réseaux «primitifs» 262
A2.2. Les mémoires associatives 263
A2.3. Le modèle de Hopfield et la machine de Boltzmann 264
A2.4. Les cartes topologiques de Kohonen 266
A2.5. Les cartes cognitives de Bart Kosko 268
A2.6. Les réseaux en couches et la rétropropagation du gradient 269
A2.7. Les «Radial Basis Functions» 272
A2.8. L'«Adaptive Resonance Theory» 274
A2.9. D'autres modèles... 275

ANNEXE A3 : Apprentissage du OU EXclusif 278

A3.1. Architecture 278
A3.2. Résultats et analyse 280
A3.3 Conclusion 288

ANNEXE A4 : Classification : La Reconnaissance de Couleurs 292

A4.1. Présentation du problème 292
A4.2. Réseau à valeurs booléennes 294
A4.3. Réseau à entrées réelles 297
A4.4. Conclusion 299

ANNEXE A5 : Apprentissage de séquences booléennes 304

ANNEXE A6 : Neurocontrôle : le pendule inverse 312

A6.1. Présentation du problème 312
A6.2. Travaux antérieurs 314
A6.3. Architecture utilisée et premiers tests 315
A6.4. Mode ordonnancé et unités à valeurs stochastiques 318
A6.5. Conclusion 333

TABLE DES FIGURES

CHAPITRE I : Introduction 22

CHAPITRE II : La Fonction Production : Nouveaux Besoins, Nouveaux Moyens 26

Figure II-1 : La chaîne de production « classique » 33
Figure II-2 : Exemple d'organisation en ingénierie concourante 35
Figure II-3 : Corrélation entre flexibilité en volume et flexibilité en type de produit 37
Figure II-4 : Flexibilité d'un système de production 40
Figure II-5 : Les différents types de production 42
Figure II-6 : Architecture de pilotage centralisée 45
Figure II-7 : Architecture de pilotage hiérarchique 47
Figure II-8 : Décentralisation par les tâches 48
Figure II-9 : Organisation de l'architecture CODECO 49
Figure II-10 : Pilotage guidé par les ressources 51
Figure II-11 : Exemple d'interaction entre plusieurs ressources 52
Figure II-12 : Echange d'information contrôleur/environnement 54
Figure II-13 : L'atelier utilisé pour la simulation 58
Figure II-14 : Les six secteurs de perception d'un robot 59
Figure II-15 : Nombre de pièces produites par l'atelier avec 1 à 13 robots 62
Figure II-16 : Nombre moyen de pièces produites par robot 63
Figure II-17 : Rendement du système de transport 63
Figure II-18 : Coopération inter-robots par ré-émission de balises 65

CHAPITRE III : Le Pilotage Réactif 68

Figure III-1 : L'organisation du contrôle symbolique 74
Figure III-2 : L'architecture classique et le modèle « Subsumption » 81
Figure III-3 : « Augmented Finite State Machine » (AFSM) 82
Figure III-4 : Architecture de contrôle de Genghis 83
Figure III-5 : Détail d'une patte de Genghis 83
Figure III-6 : Les quatre principaux schémas moteurs 85
Figure III-7 : Exemple de comportement issu de la combinaison de schémas 86
Figure III-8 : Le schéma « accostage » 86

CHAPITRE IV : Architectures Neuronales pour le Contrôle de Processus 102

Figure IV-1 : Les acteurs de l'apprentissage 105
Figure IV-2 : La cellule nerveuse 108
Figure IV-3 : Modèle général du neurone formel 109
Figure IV-4 : Les fonctions « signe » et « sigmoïde » 110
Figure IV-5 : Le neurone de Mc Culloch et Pitts 111
Figure IV-6 : Unité ΣΠ 112

Figure IV-7 : Exemple de réseau « feed-forward » 114
Figure IV-8 : Exemple de réseau à trois couches 115
Figure IV-9 : Exemple de réseau récurrent 115
Figure IV-10 : Exemple de réseau totalement connecté 116
Figure IV-11 : Apprentissage supervisé 126
Figure IV-12 : Apprentissage de la dynamique inverse 127
Figure IV-13 : « Specialized inverse learning architecture » 128
Figure IV-14 : Rétropropagation du gradient à travers un modèle 130
Figure IV-15 : Architecture pour la modélisation adaptative 131
Figure IV-16 : La critique adaptative 133
Figure IV-17 : « Stochastic real valued unit » 135
Figure IV-18: Apprentissage « auto-supervisé » 137
Figure IV-19 : Critique adaptative à travers un modèle 138

CHAPITRE V : Vers des Modèles Neuronaux pour le Contrôle Réactif 144

Figure V-1 : Réseau « Pacemaker » et réseau « marche » de la patte avant gauche 148
Figure V-2 : Schémas optique et neurophysiologique de l'oeil composé de la mouche 150
Figure V-3 : Identification du pays d'origine à partir de la plaque minéralogique 152
Figure V-4 : Hiérarchie de réseaux de neurones 154
Figure V-5 : L'architecture modulaire de Jacobs et Jordan 156
Figure V-6 : Architecture du modèle LASSO 158
Figure V-7 : Structure globale et connexions locales dans un réseau fractal 161
Figure V-8 : L'approche descendante 162
Figure V-9 L'approche ascendante 163
Figure V-10 Niveaux hiérarchiques d'un contrôleur neuro-réactif 169
Figure V-11 Niveaux d'organisation des modèles de contrôleurs inspirés des modèles biologiques 170

CHAPITRE VI : Une Solution Connexionniste Récurrente et Modulaire 172

Figure VI-1 : Synchronisation entre deux neurones du cortex visuel 177
Figure VI-2 : Génération d'un rythme moteur par une assemblée de neurone 179
Figure VI-3 : Relation entre le réseau pylorique et le réseau cardiaque 181
Figure VI-4 : Genèse du réseau déglutition 181
Figure VI-5 : Identité des bassins d'attraction : contre-exemple 184
Figure VI-6 : Architecture d'un réseau utilisant le mode ordonnancé 188
Figure VI-7 : Amplitude de variation des poids synaptiques 197
Figure VI-8 : Exemple de cycle bloquant dans un réseau à mode ordonnancé 200
Figure VI-9 : Séquences d'états dans le réseau de la figure VII-8a 200
Figure VI-10 : Elimination des cycles bloquants par un algorithme hebbien 201
Figure VI-11 : Convergence de l'apprentissage en fonction de la taille du réseau 204
Figure VI-12 : Délais moyens de convergence des réseaux 205
Figure VI-13 : Représentation graphique des lots de données 206
Figure VI-14 : La séquence utilisée pour les tests 209
Figure VI-15 : Evolution de l'apprentissage en fonction du temps 211
Figure VI-16 : Le pendule inverse 212
Figure VI-17 : Architecture utilisée lors de l'apprentissage 213
Figure VI-18 : Comportement du contrôleur sur les 120 premières itérations 217
Figure VI-19 : Evolution de x et de θ pendant 50 000 itérations 218

Figure VI-20 : Comportement du système suivant les conditions initiales 218

CHAPITRE VII : Conclusion 226

ANNEXES 252

Figure A2-1 : L'ADALINE de Widrow-Hoff 262
Figure A2-2 : Exemple de mémoire associative 263
Figure A2-3 : Le modèle de Hopfield 265
Figure A2-4 : Topologie des cartes topologiques de Kohonen 267
Figure A2-5 : La fonction du « chapeau mexicain » 267
Figure A2-6 : Exemple de carte cognitive floue 268
Figure A2-7 : Exemple de réseau à trois couches 271
Figure A2-8 : Architecture des RBF-Nets 273
Figure A3-1 : Remise en cause d'un chemin d'activation par la normalisation 281
Figure A3-2 : Convergence de l'apprentissage en fonction de la taille du réseau 282
Figure A3-3 : Délais moyens de convergence des réseaux 283
Figure A3-4 : Comparaison entre le temps moyen d'apprentissage et le nombre moyen d'états non-répondants 283
Figure A3-5 : Comparaison entre le pourcentage de convergence et le nombre moyen d'états non-répondants 284
Figure A3-6 : Temps de convergence en fonction de la date du 1er état non-répondant 285
Figure A3-7 : Répartition des poids synaptiques après apprentissage 286
Figure A3-8 : Représentation des matrices de poids après apprentissage 287
Figure A3-9 : Analogie entre les hyperneurones et les neurones seuils 288

Figure A4-1 : Représentation graphique des lots de données 293
Figure A4-2 : Règles floues pour des fuzzifications en 7 et 9 termes 296
Figure A4-3 : Fonction de décision du réseau de neurones 297
Figure A4-4 : Fonctions triangulaires d'entrée 298
Figure A4-5 : Classes apprises par le réseau à entrées semi-distribuées 299

Figure A5-1 : La séquence utilisée pour les tests 304
Figure A5-2 : Evolution de l'apprentissage en fonction du temps 306
Figure A5-3 : Etat des neurones d'entrée, d'afférence et de sortie pendant le déroulement de la séquence 308

Figure A6-1 : Le pendule inverse 312
Figure A6-2 : Architecture utilisée lors de l'apprentissage 316
Figure A6-3 : Exemple de perturbations générées lors de l'apprentissage 319
Figure A6-4 : Courbe d'apprentissage 324
Figure A6-5 : Comportement du contrôleur pendant les 120 premières itérations 325
Figure A6-6 : Evolution de x et de θ pendant 50 000 itérations 326
Figure A6-7 : Evolution de x et de θ entre 19 500 et 20 500 itérations 326
Figure A6-8 : Trajectoire dans le plan (x,θ) entre 19 500 et 20 500 itérations 327
Figure A6-9 : Trajectoire dans le plan $(\dot{\theta},\theta)$ entre 19 500 et 20 500 itérations 327
Figure A6-10 : Comportement du système suivant les conditions initiales 328
Figure A6-11 : Activité des neurones pour chacun des points de l'espace d'entrée 330

Une métaphore plus fluide s'insinuait en lui, produit de ses anciennes traversées marines. Le philosophe qui tentait de considérer dans son ensemble l'entendement humain voyait sous lui une masse soumise à des courbes calculables, striée de courants dont on eût pu dresser la carte, creusée de plis profonds par les poussées de l'air et la pesante inertie des eaux. Il en allait des figures assumées par l'esprit comme de ces formes nées de l'eau indifférenciée qui s'assaillent ou se relaient à la surface du gouffre ; chaque concept s'affaissait finalement dans son propre contraire, comme deux houles qui se heurtent s'annihilent en une seule et même écume blanche. Zénon regardait fuir ce flot désordonné, emportant comme des épaves le peu de vérités sensibles dont nous nous croyons sûrs. Parfois, il lui semblait entrevoir sous le flux une substance immobile, qui serait aux idées ce que les idées sont aux mots. Mais rien ne prouvait que ce substratum fût la dernière couche, ni que cette fixité ne cachât point un mouvement trop rapide pour l'intellect humain.

M. Yourcenar, *« L'Oeuvre au Noir »*
Gallimard, 1968.

CHAPITRE I :

INTRODUCTION

Introduction

Jusqu'à une période récente, les performances d'un système de production ont été estimées au moyen du seul critère de productivité. De fait, tant que l'entreprise est assurée de vendre rapidement l'ensemble de sa production, ce critère est directement lié au profit réalisé. Or, cette situation, caractéristique des premières décennies de l'ère industrielle, a, depuis maintenant une vingtaine d'années, tendance à disparaître et l'écoulement des stocks produits devient de plus en plus hypothétique tant la demande du marché évolue rapidement. Dès lors, le seul critère de productivité, dissocié du critère économique, devient insuffisant et des critères tels que la flexibilité et la réactivité deviennent cruciaux. Leur fonction est, schématiquement, de permettre au système de production de suivre les évolutions du marché afin d'assurer en permanence l'adéquation entre l'offre « produit » et la demande client. Ce contexte explique par ailleurs la nécessité de réduire les délais entre l'étude de marché (dont la durée de validité est souvent limitée à quelques années voire, pour certains produits, à quelques mois) et la mise en vente du produit.

Pour accroître la flexibilité et la réactivité du système de production, assurer sa pérennité et limiter le temps de mise sur le marché de nouveaux produits, nous proposons une architecture de contrôle des systèmes de production dans laquelle chaque ressource possède une fonction de décision totalement autonome et n'intégrant aucune information concernant les autres éléments composant le système de production. Cette architecture permet de limiter les délais de reconfiguration lorsque, pour une raison quelconque, l'environnement de production évolue. Dans un premier temps, nous proposons d'appliquer prioritairement cette architecture au système de transport de l'appareil de production. Ce service de transport présente, en effet, des caractéristiques particulières qui en font l'un des éléments dont les capacités de réactivité et de flexibilité sont les plus sollicitées. En outre, la décentralisation des processus de transport permet d'exploiter, en milieu de production, les nombreux travaux de recherche menés sur la robotique mobile. Une première étude - en simulation - illustre les possibilités offertes par ce type d'organisation.

Cette phase de définition « macroscopique » de l'architecture doit être complétée par une seconde étape, plus spécifiquement dédiée à la définition d'une architecture de contrôleur pour les ressources. En effet, les simulations montrent que les capacités d'un système décentralisé sont intimement liées à l'adéquation des contrôleurs locaux avec leur environnement ainsi qu'à leur capacité à développer des procédures spécifiques pour répondre à des besoins particuliers ou pour gérer les conflits inter-ressources. Ces deux caractéristiques conduisent donc le concepteur d'un contrôleur de ressources à rechercher une architecture duale réunissant des règles comportementales adaptées à l'environnement ainsi que des procédures d'apprentissage permettant aux contrôleurs de produire des comportements spécifiques, voire de modifier le jeu de règles initial lorsque l'environnement évolue au cours du temps. L'algorithme d'apprentissage permet alors de créer des architectures de pilotage auto-organisées.

Ce mémoire présente un ensemble de travaux relatifs à la création d'une architecture de pilotage décentralisée pour les systèmes de production, avec une application plus spécifique aux systèmes de transport en atelier flexible. Dans un premier temps (chapitre II), nous présenterons les contraintes auxquelles sont soumis les systèmes de production modernes et nous proposerons une organisation architecturale décentralisée. Au terme de ce chapitre, nous présenterons les premiers résultats obtenus, en simulation, en appliquant ces principes au système de transport. Outre qu'elle montre les réelles capacités de l'architecture décentralisée, cette simulation nous permettra de dresser un premier cahier des charges pour les contrôleurs locaux. Celui-ci met en particulier l'accent sur leurs capacités d'autonomie et d'adaptation.

Les deux chapitres suivants sont plus spécifiquement dédiés à l'exposé des bases des deux techniques proposées pour l'implantation des contrôleurs locaux, à savoir le contrôle réactif et les architectures neuromimétiques. Après une étude bibliographique, chaque chapitre présente en conclusion les principes retenus pour l'architecture de contrôle. Dans le chapitre III, nous effectuons une première étude des structures du contrôle réactif ; le chapitre IV, après une introduction aux algorithmes connexionnistes, présente les différentes architectures développées pour le neurocontrôle. Au terme de ce chapitre, nous sommes amenés à définir un cahier des charges pour une architecture Neuro-Réactive. Il apparaît, en effet, qu'aucun des modèles de réseaux étudiés ne répond aux principes architecturaux du contrôle réactif, en particulier pour le postulat de modularité.

Le chapitre IV nous ayant indiqué la démarche à suivre pour intégrer le contrôle réactif et le neurocontrôle au sein d'une même architecture, nous avons effectué une étude bibliographique de modèles neuromimétiques en privilégiant les architectures s'approchant des principes organisationnels définis, c'est-à-dire, essentiellement, les différents modèles de réseaux modulaires. Les résultats de cette étude sont présentés chapitre V, après quoi nous définissons l'organisation générale retenue pour notre contrôleur (section V.3).

Le chapitre VI, enfin, utilise l'ensemble des connaissances développées précédemment pour proposer un modèle de réseau Neuro-Réactif : le mode ordonnancé. Cet algorithme étant fortement inspiré par les développements récents de la neurobiologie, ce chapitre présente succinctement les données neurobiologiques utilisées avant d'exposer en détail l'algorithme proposé. Une première phase de tests est ensuite effectuée afin d'étudier le comportement du mode ordonnancé et de vérifier qu'il correspond bien au cahier des charges établi dans les chapitres précédents. Parmi les tests effectués, on trouvera d'ailleurs une application au neurocontrôle. Les performances obtenues par le mode ordonnancé permettent d'envisager les phases d'intégration de l'algorithme dans un agent autonome, puis de cet agent dans un environnement de production.

CHAPITRE II :

LA FONCTION PRODUCTION :

NOUVEAUX BESOINS, NOUVEAUX MOYENS

Au cours des vingt dernières années, l'environnement des systèmes de production a considérablement évolué. D'une production de masse cherchant quasi-exclusivement à minimiser les coûts de production, l'entreprise est passée progressivement à une production plus souple, déterminée par la demande. Cette évolution est, en particulier, caractérisée par la recherche permanente d'une flexibilité et d'une réactivité maximum afin de permettre une réduction drastique des temps de mise sur le marché.

Pour accroître la flexibilité et la réactivité de l'appareil de production, nous proposons une architecture de pilotage décentralisée dans laquelle les ressources n'agissent qu'en fonction de leurs perceptions. Cette architecture permet d'assurer l'indépendance totale des différentes ressources et, ainsi, l'évolutivité de l'atelier. Un exemple d'application sur le système de transport est proposé afin d'illustrer cette architecture.

La Fonction Production : Nouveaux Besoins, Nouveaux Moyens

Insérées dans une économie de marché de plus en plus redoutable, les entreprises de production doivent faire face à la demande de leur clientèle, mais aussi à une concurrence de plus en plus forte à mesure que s'ouvrent les barrières douanières (marché commun ou accords commerciaux internationaux). Face à cette double motivation, les entreprises ont été conduites à lutter sur des terrains autres que ceux, traditionnels, du coût de revient et de la qualité, conduisant ainsi à engendrer de nouveaux besoins parfois totalement artificiels chez leur clientèle. Désormais le seul coût de revient n'est plus déterminant pour conquérir un marché et les entreprises doivent aussi savoir s'adapter en fabriquant des produits spécifiques. Mieux, elles doivent être capables de devancer l'attente de leurs clients voire de créer leurs goûts. Ces contraintes particulières imposent aux entreprises de restructurer leur appareil de production et, en particulier, de limiter au maximum le « time to market » (temps de mise sur le marché).

Dans ce chapitre, nous allons dans un premier temps étudier plus précisément ces nouvelles tendances et leur impact sur l'appareil de production. Cela nous permettra de dégager les grandes lignes des restructurations que ces tendances entraînent pour l'organisation décisionnelle des ateliers flexibles. Nous pourrons alors proposer, dans un second temps, une architecture de pilotage décentralisée basée sur l'auto-organisation de ressources autonomes. Pour clore ce chapitre nous aborderons succinctement une description de deux approches issues de l'intelligence artificielle - le contrôle réactif et les mécanismes d'apprentissage neuromimétiques - qui nous semble parfaitement indiquées pour réaliser les contrôleurs locaux de chacune des ressources. On remarquera que ces deux techniques sont plus orientées vers la robotique que vers la productique mais il n'y a pas là d'antagonisme : la décentralisation des processus de décision conduit à donner aux machines elles-mêmes (aux « robots », au sens de Philippe Coiffet[1] [COI 93]) une plus forte capacité de gestion. C'est donc bien vers les recherches sur la robotique « intelligente » et coopérante qu'il convient de se tourner pour développer, dans un atelier de production, des capacités d'auto-organisation. Nous présenterons d'ailleurs pour conclure ce chapitre (section II.3.5) un exemple d'application des principes d'auto-organisation qui, bien que très simplifié, permet d'illustrer l'architecture décrite dans ce chapitre et de préciser le cahier des charges des contrôleurs locaux.

---

[1] Selon Philippe Coiffet un robot est une machine répondant à trois caractéristiques essentielles (d'après [COI 93], chapitre 6, pages 97 et 98) :

- C'est une machine physique modifiant *physiquement* son environnement pour atteindre un but.
- C'est une machine *versatile*, c'est-à-dire qu'elle peut exécuter une tâche donnée de plusieurs manières différentes.
- Elle est capable de s'adapter seule aux variations de l'environnement de telle sorte que la tâche soit correctement exécutée en dépit de ces variations.

II.1. La fonction production

Les entreprises industrielles sont des systèmes complexes, décomposables en plusieurs fonctions essentielles, toutes interdépendantes. Parmi toutes ces fonctions, la fonction production occupe, du moins dans les entreprises de transformation, une place essentielle. C'est en effet la fonction générant le plus directement la valeur ajoutée par transformation de matière et/ou assemblage de composants. Elle conduit ainsi à la création du produit fini propre à la distribution vers le (les) client(s) de l'entreprise.

La fonction production peut être décomposable en quatre entités principales. Bien qu'importantes pour la suite de notre exposé (nous nous y référerons régulièrement), celles-ci ne peuvent généralement être définies que de manière intuitive [HAT 92] :

- **Les entités :** l'entité est l'objet directement manipulé par la fonction production. Suivant les étapes de la production il peut donc s'agir de matière première, de composants, de produits semi-finis ou de produits finis.

- **Les ressources :** les ressources du système de production sont l'ensemble des moyens ou des composants mis en oeuvre pour transformer les entités et passer de la matière première au produit fini. La décomposition d'un atelier en un ensemble de ressources est le plus souvent arbitraire et liée à un niveau de décomposition particulier.

- **Les tâches :** le concept de tâche est intimement lié aux deux notions précédentes. En effet, une tâche est l'application directe d'une ressource à une entité. Cette notion est donc étroitement dépendante du niveau de décomposition utilisé pour les ressources.

- **Le pilotage :** la fonction production est un processus complexe appliquant à une ou plusieurs entités un ensemble de tâches et utilisant une ou plusieurs ressources. L'organisation temporelle de ces tâches est confiée à un pilote chargé de séquencer ces opérations dans le temps (ordonnancement) et dans l'espace (présence conjointe des entités et des ressources nécessaires).

L'activité de la fonction production, bien que d'importance majeure au sein de l'entreprise, doit être corrélée avec de nombreuses autres fonctions. Ainsi les diverses fonctions commerciales de l'entreprise (commande et marketing) ont de plus en plus d'influence sur l'activité de la fonction production et celle-ci doit être dimensionnée pour pouvoir répondre efficacement à leurs sollicitations.

Le travail que nous présentons ici concerne un type de production particulier, à savoir la production manufacturière et de transformation. Dans ce contexte, la production nécessite la mise en oeuvre de techniques et d'outils plus ou moins complexes et la phase de définition d'un produit s'accompagne le plus souvent d'une phase « méthode » de conception de l'outil de production lui-même. Par la suite la phase de production proprement dite est soumise à un séquencement complexe d'opérations internes (ordonnancement et pilotage de la production) ou externe (fonction achat, stockage et expédition). Il en résulte une organisation complexe étroitement soumise aux fonctions commerciales de l'entreprise qui provoquent, périodiquement, sa remise en cause.

II.2. Le nouvel environnement de la production

Depuis le dix-neuvième siècle la production industrielle n'a cessé de se diversifier et de devenir plus complexe. En cela, elle est étroitement liée à une évolution de la demande, évolution qu'elle contribue elle-même à créer en un cercle vicieux à l'échelle nationale et, depuis quelques dizaines d'années, mondiale. Schématiquement on peut considérer que les systèmes de production se sont progressivement structurés afin de se mettre en parfaite adéquation avec les modèles économiques et sociaux successifs. Ainsi, si l'on exclut les premières décennies de l'ère industrielle qui ont vu les milieux sociaux et industriels cohabiter sans autre interaction que l'utilisation par les seconds de la force de travail des premiers (la consommation des biens industriels étant réservée à une élite sociale), on peut grossièrement isoler trois phases successives dans l'évolution des modèles production/consommation. Chacune de ces phases est accompagnée d'un modèle de production particulier [COH 92].

La première de ces trois phases, le « Fordisme », représente l'avènement du phénomène de consommation de masse donc de la production en grande série de produits standardisés. Dans l'esprit de Ford cette organisation doit permettre une réduction des coûts de production et, ainsi, ouvrir de nouveaux marchés. L'ouvrier devient ainsi un consommateur à la demande homogène.

Dans un deuxième temps le principe de réduction des coûts de production s'est avéré insuffisant pour conquérir de nouvelles parts de marché. Les entreprises ont alors dû s'adapter aux besoins de leur clientèle. Elles ont ainsi cherché à élargir leurs gammes de produits afin d'exploiter commercialement les particularités des individus-clients.

Avec l'épuisement de ces deux modèles (concurrence sur les prix et élargissement des gammes de produits), on assiste depuis une vingtaine d'années à l'émergence d'un troisième modèle dans lequel le facteur temps apparaît comme prépondérant. L'entreprise ne doit plus seulement fournir, au meilleur prix, un produit adapté à la demande du client, elle doit fournir LE produit désiré dans les délais les plus courts.

Chacun de ces modèles entraîne pour l'appareil de production une organisation particulière afin de répondre au mieux à la demande. Dans la section suivante, nous rappelons brièvement l'organisation propre aux deux premiers modèles. Nous étudierons ensuite les importantes restructurations imposées par le troisième.

II.2.1. Standardisation et variété

Le modèle de standardisation développé par Ford dans les années vingt était accompagné d'un modèle de production, le taylorisme, permettant, au moyen de processus séquentiels, la production en grandes séries. Techniquement, le taylorisme est associé au développement des chaînes de production. C'est aussi l'introduction massive de la mécanisation au sein de la fonction production et la division des tâches en groupes fonctionnels. Structurellement, le taylorisme est accompagné d'une organisation pyramidale de l'entreprise, ce qui entraîne un cloisonnement intensif des différentes fonctions. Cette organisation est entièrement orientée vers la réduction des prix, élément central de concurrence, au prix d'une importante dissociation entre la fonction production et le marché. Le modèle de Ford et le Taylorisme sont donc directement liés à une conception homogène des marchés, parfaitement en accord avec l'essor d'une classe moyenne, ouvrière, dont les réflexes de consommation sont eux-mêmes standardisés.

Cette conception constitue cependant le tendon d'Achille du « Fordisme ». En effet, l'homogénéité des marchés n'est le plus souvent que l'effet d'une offre elle-même standardisée (entre autre pour des besoins techniques). La diversification des gammes de produit, coûteuse et difficile à mettre en oeuvre avec l'organisation tayloriste, a donc progressivement conduit les entreprises à faire évoluer leur organisation. Techniquement, le développement des machines à commande numérique a permis de remettre en cause un des principes fondamentaux du taylorisme (une ligne/un produit) et, parallèlement, a autorisé la reconversion des lignes. Le modèle de variété a ainsi conduit à l'apparition des outils de production « flexibles », à la différenciation retardée puis à la conception modulaire des produits.

Structurellement les entreprises ont dû progressivement décloisonner les différentes fonctions : plus encore que la remise en cause du taylorisme, le passage de la standardisation à la variété s'accompagne d'une influence grandissante de l'état de l'environnement (le marché) sur la fonction production. Les entreprises n'ont plus la liberté de voir le marché s'adapter à leurs produits, elles doivent être en permanence à l'écoute des consommateurs. Schématiquement, le passage de la standardisation à la variété, c'est le passage de la publicité au marketing.

II.2.2. La réactivité : vers une nouvelle forme de concurrence

Après avoir été successivement axée sur la compétitivité puis sur la diversité, la concurrence entre les entreprises s'est intensifiée autours de services connexes à la production. Ainsi les services avals (livraison, après-vente) ont été l'objet d'une concurrence tenace. Les délais de livraisons, en particulier, ont été des arguments publicitaires importants (« demain chez vous », « livraison en 48h chrono » en sont des exemples caractéristiques). Plus récemment encore le service marketing et, avec lui, l'ensemble des services amonts (études, méthodes), ont acquis une importance particulière. En effet, il ne s'agit plus pour l'entreprise de suivre la demande de sa clientèle mais bien de la devancer afin de fournir au client, au moment où la demande se crée, le produit adapté. Les exemples ne manquent pas : les industries du jouet, du produit périscolaire ou du textile connaissent bien les phénomènes de mode. Plus caractéristique encore, dans l'industrie automobile, le développement par Renault de la « Twingo » illustre bien cette nouvelle tendance.

Ces nouvelles tendances s'accompagnent, pour la fonction production, de contraintes extrêmement fortes, en particulier en termes de délais. Ainsi, alors que les modèles de standardisation et de variété s'accompagnaient d'une certaine pérennité de l'appareil de production, la réactivité s'accompagne d'une remise en cause perpétuelle de l'outil de production réduisant d'autant le délai dévolu à sa conception. De plus, l'action conjointe de la réduction des délais de livraison et de la livraison à la carte (donc de l'absence de stocks), entraîne des contraintes fortes sur les temps de fabrication.

II.2.2.1. Délais de mise à disposition/Délais de conception

Le passage d'un modèle de variété à un modèle de réactivité se caractérise principalement par une étroite dépendance entre l'organisation de l'entreprise (à tous les niveaux) et les variations de son environnement (clientèle, concurrence) [COH 92]. L'entreprise doit donc être capable de s'adapter et d'anticiper les variations du marché en remettant en question sa structure interne et, en particulier, celle de son secteur « production ».

Une des caractéristiques dominantes des trois modèles de production cités est la surenchère qu'ils imposent à la fonction production. En effet, l'apparition du modèle de variété n'a pas remis en cause la recherche d'une productivité maximale. De même le modèle de réactivité (qui prolonge le modèle variété) s'accompagne d'une concurrence féroce sur les prix et sur les délais de mise à disposition. Or la cohabitation entre ces différentes formes de concurrence ne fait que renforcer la complexité du problème.

La minoration des délais de mise à disposition, qui était à l'origine obtenue grâce à un stock de produits finis, s'est vue compromise par l'apparition de la production personnalisée et de la production unitaire (One of a Kind Production). Dans ce nouveau contexte le maintien de stocks de produits finis constitue un risque important pour l'entreprise, risque généralement inacceptable. Dès lors, le délai de mise à disposition ne correspond plus au simple délai de livraison mais à l'ensemble du cycle de fabrication/livraison. Les premières réponses apportées par les entreprises à cette double demande concernaient le produit lui-même : conception modulaire, différenciation retardée... Dans un second temps la structure même des systèmes de production a été remise en cause avec l'irruption du modèle japonais du « Juste A Temps » (JAT).

Parallèlement à la réduction des temps de fabrication, l'adaptation permanente aux fluctuations du marché entraîne une réduction drastique des délais dévolus à la conception des produits. En effet, si le but des entreprises est de fournir à leur clientèle le produit attendu au moment même où le besoin est exprimé (voire provoquer l'expression du besoin), le temps de conception correspond au délai d'anticipation de la demande du client. Or la conception d'un produit manufacturé est traditionnellement le fruit d'une longue séquence d'opérations entre le marketing et le lancement de la fabrication (figure II-1). Cette organisation séquentielle entraîne des délais importants, incompatibles avec les fluctuations du marché. C'est pour réduire les délais de conception/lancement/fabrication/distribution (le « time to market ») que sont apparues, toujours en provenance du Japon, les méthodes de conception parallèle des produits (le « Concurrent Engineering »).

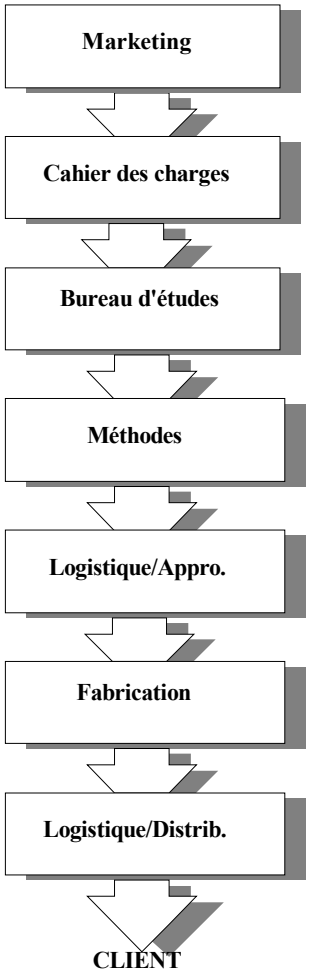


Figure II-1 : *la chaîne de production « classique »*

II.2.2.2. Les flux tendus et le Juste à Temps

Les méthodes classiques de gestion de flux, en particulier les méthodes de type M.R.P.[2], appartiennent à la famille de flux poussés. Elles font en effet fortement appel à une programmation prévisionnelle, centralisée, destinée à anticiper la demande de composants. Dans une logique clientéliste de production (la production est destinée à répondre à une demande directe et non à alimenter des stocks), ce mode de gestion se révèle souvent peu rentable parce qu'il crée des stocks à plusieurs niveaux.

Le modèle des flux tendus, ou flux tirés, est issu d'une logique totalement différente. La production est alors dictée par les besoins et les ateliers ne produisent que le strict nécessaire afin de réduire au maximum les gaspillages, c'est-à-dire les opérations ne conduisant pas à produire de la valeur ajoutée (telles que les opérations de stockage). Le principe de gestion par flux tirés le plus célèbre est sans conteste la méthode Kanban (du japonais « étiquette »). C'est une méthode décentralisée de gestion des flux basée sur des ordres de fabrication implicites, les étiquettes. Elles sont traitées manuellement ce qui implique directement les salariés et contribue ainsi à leur intéressement [GIA 94].

---

[2] Material Requirement Planning (MRP1), Management Ressource Planning (MRP2)

En fait les méthodes de flux tendus s'insèrent dans un contexte plus global de gestion de l'outil de production : le Juste à Temps (JAT). La philosophie du Juste à Temps reprend les principes de lutte contre les stocks intempestifs et contre toute forme de gaspillage au niveau de l'appareil de production. Par gaspillage on entend ici non seulement toute opération non productrice de valeur ajoutée, mais aussi toute non-utilisation du potentiel productif des opérateurs. Dans cette optique, les travailleurs doivent avoir une formation polyvalente et un haut niveau de qualification afin de pouvoir se déplacer d'un poste à un autre et assumer localement (au niveau d'un poste de travail) des responsabilités plus globales (assurer le fonctionnement de la chaîne de production). Le temps inoccupé (par exemple pendant les périodes de sous-production) est alors utilisé à la maintenance prédictive de l'appareil de production ce qui limite les rebuts et favorise la qualité de la fabrication. Par contre, s'agissant des ressources matérielles, les entreprises utilisant le Juste à Temps sont généralement surdimensionnées afin de favoriser la flexibilité et la réactivité de la production.

II.2.2.3. L'ingénierie concourante

Afin de réduire au maximum le « time to market », et donc de mieux anticiper la demande de la clientèle, les entreprises sont amenées à réduire au maximum les délais entre la conception marketing d'un produit et son arrivée sur le marché. D'autant plus qu'au besoin de fournir le produit à temps, s'ajoute un phénomène de dynamique commerciale autour des concepts novateurs et des entreprises qui les ont développés. Le cas de l'automobile et de la société Renault, avec les modèles « Espace » et « Twingo », est caractéristique de cet état d'esprit.

Ainsi la Twingo, qui a fait figure de modèle particulièrement novateur lors de sa sortie, a été conçue en trois ans et demie, délais légèrement supérieur à ceux de l'industrie japonaise, alors qu'en Europe cinq ans sont généralement nécessaires entre l'étude marketing et la mise sur le marché d'un nouveau modèle. Le cas de la Renault Espace est, lui aussi particulier : en ayant conçu son modèle suffisamment tôt, Renault a réussi, avec l'aide de Matra, à se créer une situation de monopole sur le concept du monospace, et son concurrent direct, le groupe PSA, ne parvient que difficilement à pénétrer sur ce marché. Sur des marchés à forts volumes, faibles marges et à rotation rapide, l'importance du facteur temps est encore plus prépondérante : en électronique et en informatique, par exemple, un retard de quelques mois sur un concurrent peut compromettre gravement l'avenir d'un produit.

Pour réduire ces délais, les entreprises ont dû développer une approche organisationnelle orientée vers une conception simultanée des produits, des méthodes, des outils de fabrication et du soutien logistique (approvisionnement, stockage, distribution, ...) [JAG 93]. Pour paralléliser les différentes tâches, trois approches peuvent être suivies :

- conception simultanée du produit et de son système de fabrication,
- découpage en sous-projets réalisables en parallèle par des équipes différentes,
- création de groupes de travail transfonctionnels afin de réaliser une tâche.

Les deux derniers points sont similaires au problème de la parallélisation des algorithmes informatiques. Dans le second cas, l'accent est mis sur le déroulement parallèle des tâches avec seulement quelques points de synchronisation. Dans le troisième cas, le travail en commun permet de dégrossir un problème ou d'affiner une solution en évitant les aller-retours entre services. Il est important de noter que ces trois approches peuvent être utilisées simultanément.

L'ingénierie concourante reposent sur le partage du système d'information et sur l'exploitation directe, à tous les niveaux, du travail en cours (figure II-2). La mise en correspondance des données manipulées aux différents niveaux constitue d'ailleurs la principale difficulté rencontrée par les entreprises utilisant l'ingénierie simultanée. L'utilisation de systèmes d'information complexes basés sur des hyperdocuments et sur le multimédia semble être, dans ce domaine, une voie intéressante [BIE 90a][BIE 95b] mais aucune solution satisfaisante n'a pour l'instant été trouvée pour uniformiser les données issues des différents services. En effet, celle-ci doit avant tout passer par une intégration des différents outils informatiques (CAO/CFAO/GPAO[3]/...) via les données utilisées afin de pouvoir générer, pour chacun d'eux, les informations nécessaires. Cette difficulté propre à l'ingénierie concourante ne doit cependant pas effacer l'intérêt de cette organisation qui permet à l'entreprise de suivre, en temps réel, les évolutions d'un marché très perturbé.

D'un autre côté, l'ingénierie concourante ne doit pas être une réponse unique à la réduction des délais de conception. L'ingénierie simultanée doit s'accompagner d'une réduction maximale des délais propres à chacune des étapes de conception/fabrication. L'appareil de production, en particulier, doit, pour faciliter la mise en place des procédures d'ingénierie concourante, améliorer ses capacités de flexibilités et de réactivité afin de limiter les coûteuses réorganisations d'ateliers. En effet, même si elle peut en partie être effectuée parallèlement à la conception du produit lui-même, la mise en place d'un nouvel atelier de production demande d'importants temps de réglage.

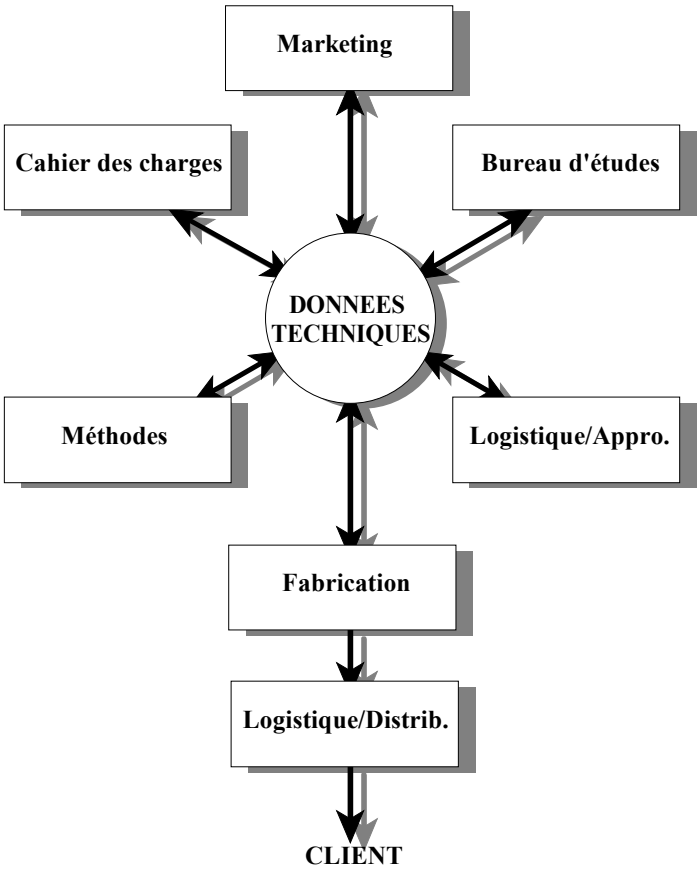


Figure II-2 : *Exemple d'organisation en Ingénierie Concourante*

[3] Conception Assistée par Ordinateur, Conception/Fabrication Assistées par Ordinateur et Gestion de Production Assistée par Ordinateur

II.2.3. De nouveaux impératifs pour l'appareil de production

La mutation de la demande et de la concurrence se caractérise, nous venons de le voir, par une influence directe de l'environnement extérieur sur le système de production. Au niveau global (entreprise, usine), le passage du modèle de variété au modèle de réactivité entraîne une restructuration totale ou partielle (suivant les typologies de production et les caractéristiques du marché), caractérisée entre autres par une production à flux tendus dans une optique de Juste à Temps et par une conception simultanée des produits et des moyens (voir sections précédentes).

En ce qui concerne la fonction production elle-même, l'évolution des mentalités a provoqué l'irruption brutale de l'incertain au coeur même des systèmes de production, avec l'impératif, pour ceux-ci, de gérer localement cette nouvelle donnée. Les mutations structurelles des entreprises modernes ont en effet eu pour conséquence principale une volonté générale de responsabiliser et de rendre autonomes les différents acteurs de la production [LEG 92]. Dès lors qu'ils ont accédé à l'autonomie, les ateliers ont dû « apprendre » à intégrer, dans leur mode de fonctionnement, des perturbations plus ou moins « attendues » sans pour autant baisser les cadences ou augmenter les coûts de production. La décentralisation des processus de décision a conduit à généraliser la recherche d'une flexibilité et d'une réactivité maximales, sans que la définition de ces deux termes ait toujours été précisé préalablement. D'autres architectures, telles que les boucles de pro-action, peuvent être utilisées pour répondre aux aléas [CAM 95].

Chacune de ces notions est très ouverte et il en existe au moins autant de définitions que d'auteurs ayant travaillé sur ce sujet ; certaines d'entre elles recouvrent d'ailleurs les deux termes. C'est pourquoi, avant de les utiliser dans la suite de notre étude, nous allons en proposer une définition synthétique. Celle-ci n'entend pas être meilleure que celles proposées dans la littérature mais nous lui avons trouvé l'avantage de la simplicité car elle différencie clairement deux types de perturbations. De plus, elle reste en accord avec la plupart des définitions usuelles.

II.2.3.1. Flexibilités

Si nous avons délibérément choisi, dans la première partie de ce chapitre, d'utiliser le terme « flexibilité » dans une forme plurielle, c'est pour rendre compte de son ambiguïté. En effet, dans sa définition générale, la flexibilité traduit l'aptitude d'un système à changer facilement pour pouvoir s'adapter aux circonstances. Appliquée aux systèmes de production cette définition nous conduit à considérer la flexibilité comme : *L'aptitude d'un système de production à évoluer et à s'adapter aux circonstances - à son environnement et à son évolution interne - sans dégrader sa fonction première de production.*

Cette définition ne nous autorise qu'à une vision globale de la flexibilité. En effet, elle fait appel à des notions elles-mêmes très ambiguës : quel est le degré de dégradation accepté dans un système de production ? Qu'est-ce que l'environnement d'un système de production ?

En pratique, la définition est morcelée en plusieurs formes de flexibilité. C'est pourquoi nous avions choisi d'utiliser le terme au pluriel.

Ainsi, de nombreux auteurs s'accordent à distinguer, dans un système de production, la flexibilité en volume et la flexibilité en type de produit. La première exprime en effet la

capacité d'un système de production à s'adapter, sans dégradation de productivité, à des tailles de séries différentes. La seconde caractérise la possibilité, pour le même système, de traiter différents types de produits, toujours sans perte notable de productivité. En pratique, ces deux aspects sont intimement liés. Si on excepte les situations extrêmes que sont la production unitaire (OKP[4]) et la très grande série (pour lesquelles la flexibilité en volume et la flexibilité en type n'ont, respectivement, plus aucun sens), la flexibilité en type de produit doit obligatoirement, si elle est exploitée, s'accompagner d'une flexibilité en volume (sauf dans le cas où des produits différents sont fabriqués en un même nombre d'exemplaires, ce qui est le cas dans la sous-traitance par exemple). De même une flexibilité en volume (si elle est réellement exploitée) doit être accompagnée d'une flexibilité en type de produit sous peine de créer des temps morts importants entre chaque changement de production. Le temps réel de production est alors largement inférieur au temps de mobilisation de l'outil de production (figure II-3).

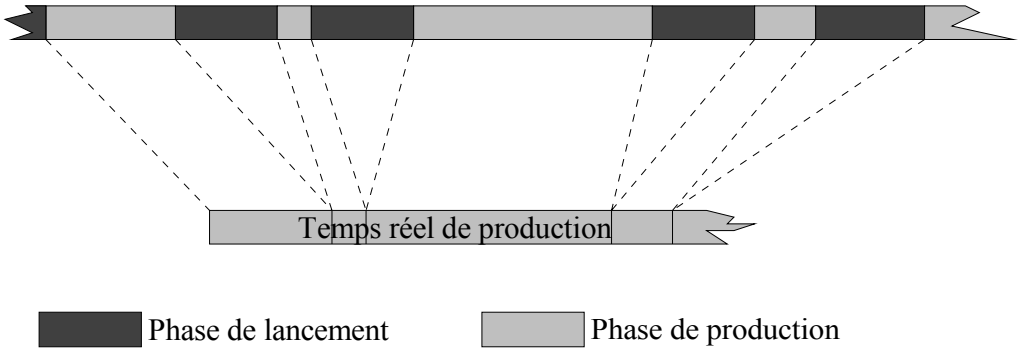


Figure II-3 : *Corrélation entre flexibilité en volume et flexibilité en type de produit :*

Lorsque la flexibilité en volume est utilisé en l'absence de flexibilité en type de produit, les temps de lancement répétés augmentent considérablement le temps total de production ce qui provoque une baisse de productivité (le temps réel des phases de production restant constant).

On trouve aussi dans la littérature une décomposition plus fonctionnelle de la flexibilité :

- **Flexibilité de moyens.** Elle caractérise la potentialité physique, pour un système de production, d'usiner et d'assembler des pièces différentes en quantité variable. C'est essentiellement la variété des gammes exécutables par le système. La flexibilité de moyens est essentiellement liée à la structure physique du système, par exemple à l'utilisation de machines-outils à commande numérique (MOCN).
- **Flexibilité de pilotage.** C'est l'exploitation réelle, par le système de pilotage, des possibilités de la flexibilité de moyens, c'est-à-dire, entre autres, sa capacité à fournir des gammes de remplacement, des modes dégradés, ou, plus simplement, à fournir un ordonnancement adapté à tous les niveaux, à la fabrication de pièces différentes (présence simultanée des outils, matières et gammes sur une machine).
- **Flexibilité d'évolution.** On se place ici dans une optique à plus long terme. Un système de production ne constitue pas, nous l'avons vu, un système figé dans le temps. La flexibilité d'évolution caractérise donc sa capacité à évoluer sans heurt, par exemple sa capacité à utiliser de nouveaux outils, de nouvelles gammes, ou à intégrer

[4] One of a Kind Production

une nouvelle machine sans ralentir la production. La flexibilité d'évolution est liée à la fois au système physique (réaménagement des systèmes de transport, place disponible, ...) et au système de pilotage (possibilité d'intégrer de nouveaux emplacements, de nouvelles gammes ou de nouvelles machines sans remettre en cause tout le processus de pilotage).

Cette deuxième décomposition n'entend pas être meilleure que la première et chacune d'entre elles a ses avantages et ses inconvénients. Elle permet cependant de distinguer des niveaux différents dans la recherche de la flexibilité. Ainsi une gestion de flux utilisant le Juste à Temps entraîne un besoin en flexibilité de moyens et, en corollaire, en flexibilité de pilotage. Par contre, la mise en place d'une procédure d'ingénierie concourante est grandement facilitée par une bonne flexibilité d'évolution. Celle-ci permet en effet de lancer immédiatement de nouvelles fabrications sans remettre en cause une grande partie des systèmes déjà en place dans l'entreprise.

Une troisième décomposition, très proche de la précédente, est proposée dans [COH 92] qui distingue les aspects statiques et dynamiques de la flexibilité.

La flexibilité statique correspond aux opportunités d'évolution du système de production ainsi qu'à une potentialité d'absorption des événements perturbateurs. Elle nécessite généralement la mise en oeuvre de surcapacités ou d'équipements de secours afin de parer à d'éventuelles hausses de la demande ou à des arrêts machine intempestifs (la flexibilité statique est similaire à la flexibilité de moyens déjà citée).

Inversement la flexibilité dynamique correspond à « la capacité de réagir continûment, dans le temps, aux variations de l'environnement ». Cette forme de flexibilité suppose que le système de production réponde à des variations quelconques de son environnement (donc supposées aléatoires) avec un temps de réponse inférieur (ou au moins similaire) au temps de modification de l'environnement (principe du « temps requis »).

L'intérêt principal de la décomposition en flexibilité statique ou dynamique serait de dissocier leur impact sur les coûts de revient. Toujours d'après [COH 92] la flexibilité statique génère des surcoûts pour l'entreprise (espace, équipements de secours, ...) tandis que la flexibilité dynamique, ne reposant pas sur des équipements mais sur une gestion plus fine de l'enchaînement des opérations de production, permettrait au contraire de diminuer les coûts de production en favorisant une utilisation plus intensive des ressources humaines et matérielles. La flexibilité dynamique rejoint en cela les principes du Juste A Temps (JAT) : élimination des gaspillages en redistribuant les ressources non utilisées vers d'autres secteurs de production.

Cette décomposition, bien qu'intéressante d'un point de vue fonctionnel, ne recouvre qu'une réalité très incertaine. Ainsi, comme pour la flexibilité en volume et la flexibilité en type, les deux formes sont souvent indissociables. Comment mettre en oeuvre la flexibilité dynamique si le système de production est figé, sans aucune possibilité physique de modification des séquences et/ou des gammes. Par exemple, dans le cas d'une ligne de transfert, il est impossible, si un poste est stoppé par un manque d'approvisionnement, de redistribuer le travail vers d'autres postes amonts ou avals. Dans ce cas c'est l'ensemble de la chaîne qui s'arrête. Quant à redistribuer le travail sur d'autres chaînes, ce n'est possible que si les ressources sont mobiles et si les ouvriers sont formés pour cela (main-d'oeuvre plus qualifiée). Ce n'est donc possible que si les outils de la flexibilité statique ont déjà été mis en place.

En outre, parler de la flexibilité dynamique comme d'une procédure ne générant aucun surcoût est une approximation trop rapide. C'est oublier, en effet, les coûts de développement des outils de pilotage complexes, auxquels s'ajoutent les coûts salariaux et de formation du personnel qualifié chargé d'appliquer ces procédures de gestion. De plus, la mise en oeuvre de la flexibilité dynamique engendre également des coûts dus à la flexibilité statique nécessaire.

Dans un autre ordre d'idée la flexibilité dynamique est définie par rapport à un concept « d'environnement générateur d'aléas » qui, pour un système de production, demande à être précisé. On doit ainsi distinguer les aléas générés par les différents systèmes de gestion de l'entreprise, c'est-à-dire les ordres inattendus. En effet, ceux-ci ne doivent pas être absorbés par le système de production mais suivis et exécutés. En d'autres termes, le système de production, dont la fonction est d'exécuter un plan de production généré par les diverses fonctions de gestion de l'entreprise, doit avant tout faire une distinction entre les aléas *modifiant le plan de production* et ceux *perturbant le plan de production*.

En fait la rupture - artificielle - entre flexibilité statique et flexibilité dynamique est essentiellement d'origine historique. Le passage du modèle de standardisation au modèle de variété s'est traduit, au niveau du système de production, par le développement de la flexibilité statique. Dans un second temps, la transition vers le modèle de réactivité a entraîné, en réduisant les délais à tous les niveaux, l'apparition de la flexibilité dynamique.

Les différentes définitions proposées ci-dessus décomposent la flexibilité de production à partir des différents moyens mis en oeuvre dans les ateliers (matériel et logiciel). Pour la suite de notre travail nous préférerons une définition basée sur l'origine des événements. Elle permet en effet de distinguer les moyens nécessaires à leur gestion en termes de stratégie (exploitation ou élimination) et non en termes d'outils matériels ou logiciels.

Afin de limiter au maximum les confusions dues à l'ambiguïté des différentes définitions, nous n'utiliserons par la suite le terme de flexibilité que dans un sens restreint. En complément nous utiliserons le terme de réactivité, leur union permettant de retrouver le sens général de la flexibilité de production (voir ci-dessus). Ces deux termes seront définis dans les paragraphes suivants, ainsi que la robustesse, une troisième caractéristique des systèmes de production, étroitement liée aux deux précédentes.

II.2.3.2. Flexibilité

Dans la suite de notre travail, nous appellerons **flexibilité** *la capacité d'un système de production à évoluer et à s'adapter pour répondre immédiatement et sans dégradation des performances à des ordres de fabrication différents*.

Suivant cette définition la flexibilité correspond à la capacité d'absorption des événements (perturbateurs ou non) issus des niveaux décisionnels supérieurs de l'entreprise. Ces événements, bien que pouvant être très divers, correspondent toujours à des ordres de fabrication (ou à des modifications d'ordres de fabrication). Ils entraînent donc une modification de la production de l'atelier.

La flexibilité est donc une caractéristique externe des systèmes de production : l'étendue de leur domaine de variation. Elle suppose en particulier que les fonctions connexes d'approvisionnement et de distribution sont, elles aussi, capables de s'adapter aux variations de la production (figure II-4).

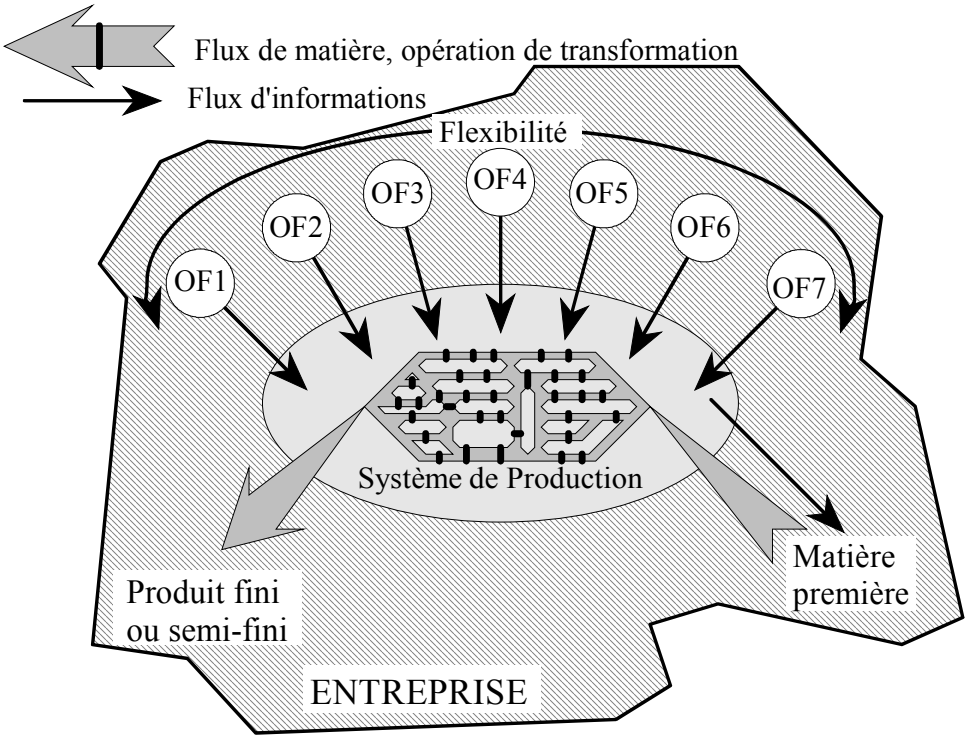


Figure II-4 : *Flexibilité d'un système de production*

La flexibilité correspond à la variabilité des ordres de fabrication (OF) supportés, sans dysfonctionnement, par le système de production, c'est-à-dire à la multiplicité des flux de matière et des opérations de transformation.

II.2.3.3. Réactivité

Si la flexibilité est la capacité, pour un système de production, à accepter des ordres de fabrication différents, *la réactivité est la capacité d'absorption des perturbations (par perturbation on entend : l'ensemble des événements inattendus, à l'exception des ordres de fabrication) toujours sans dégradation de la productivité*.

A l'inverse de la flexibilité qui représente un potentiel d'utilisation du système de production (la marge de manoeuvre des niveaux décisionnels supérieurs), la réactivité est une caractéristique interne du système. Ainsi un système totalement réactif est aussi totalement transparent. Tout ordre de fabrication conduit à la production des pièces désirées en temps voulu quels que soient les événements perturbateurs apparus au cours de la production (bris d'outil, panne de machine, ...).

II.2.3.4. Robustesse

La robustesse est, comme la réactivité, une caractéristique interne des systèmes de production. Toutes deux sont d'ailleurs fortement liées. En effet, les perturbations apparaissant au cours de la production peuvent être d'origine interne (panne, bris d'outil, ...) ou externe (rupture d'approvisionnement, erreur d'étiquetage, ...). Si les secondes ne peuvent pas être directement éliminées (par la politique interne de l'atelier), les premières peuvent, par contre être évitées en utilisant un matériel plus robuste. *La robustesse est donc la capacité*

d'un système de production à transformer des matières premières en produits finis ou semi-finis sans générer d'événements perturbateurs, c'est-à-dire sans dégradation(s) de ses constituants. La robustesse peut donc être considérée comme l'application aux systèmes de production du principe « d'homéostasie » développé dans le cadre de la cybernétique. L'outil principal de la robustesse est la maintenance préventive.

La notion de robustesse est souvent associée avec celle de « tolérance » qui est, elle aussi, fortement liée à la réactivité. *La tolérance est la capacité du système de production à maintenir son fonctionnement (sans modifications) malgré l'apparition d'événements perturbateurs*. C'est donc une caractéristique particulière du système : des aléas peuvent apparaître en cours de production (contrairement à la robustesse) mais le système est conçu et dimensionné afin qu'ils n'aient aucun caractère perturbateur pour la production. La réunion de ces deux aspects conduit à une définition plus large de la robustesse : *la robustesse d'un système de production est sa capacité à réaliser les objectifs de production (c'est-à-dire à suivre le plan de production) en absorbant les perturbations (internes ou externes).*

II.3. Vers des structures de production flexibles et réactives

La transition vers une architecture fédératrice décentralisée a entraîné, au niveau du système de production, un accroissement important des capacités de décision. En effet, la flexibilité et la réactivité imposent un traitement local de l'ensemble des événements entrants. Seuls les flux d'information indispensables au bon fonctionnement d'un système de production doivent donc être conservés : les ordres de fabrication et les commandes de matières premières. Tout autre échange d'information entraîne une hausse importante des délais. De plus les échanges de matières doivent être limités au maximum (zéro stock) afin de limiter les coûts de production.

L'utilisation des machines-outils à commande numérique (MOCN) et l'automatisation ont permis de réaliser un compromis entre la productivité des lignes de transfert et la flexibilité/réactivité des machines servies par l'opérateur humain (figure II-5).

La décomposition des différents types de production n'est en fait que le fruit de l'introduction progressive de l'informatique au coeur du système de production. *L'automatisation flexible est ainsi largement due à l'incorporation de la flexibilité propre à l'informatique dans les schémas effecteurs d'une machine ou dans les schémas de pilotage des systèmes de production*. D'après [HAT 92], cette évolution peut être décomposée en trois étapes majeures.

Dans un premier temps l'apparition de l'informatique industrielle a permis la création de processus automatiques plus complexes et plus souples que ceux supportés antérieurement (automatismes mécaniques et automates programmables de la première génération). L'application de ces processus aux systèmes de production a conduit à l'apparition des MOCN qui permettent d'approcher la productivité et la flexibilité de l'opérateur humain (figure II-5, flèches 1). La réactivité, par contre, est loin d'être aussi élevée mais les machines à commande numérique compensent ce défaut par une excellente robustesse.

Dans un second temps, ces mêmes processus automatiques ont permis la réalisation de systèmes robotisés de plus en plus performants. Les MOCN ont alors pu être regroupées en cellules (petits groupes de machines regroupées autour d'un mécanisme robotisé et orienté vers la fabrication d'une gamme de pièce) puis, pour des productions en plus grandes séries, en lignes flexibles. Cette structure permet en effet de conserver la cadence propre aux lignes

de transfert (en limitant l'appel aux procédures complexes d'ordonnancement) tout en produisant, grâce aux MOCN, des pièces différentes au sein d'une même famille (figure II-5, flèches 2).

Enfin, dans un dernier temps, l'introduction directe de l'ordinateur dans la gestion du cycle de production a favorisé l'utilisation des techniques avancées de l'informatique (système d'information, intelligence artificielle, ...). Celles-ci ont alors apporté à l'automatisation flexible les processus de décision qui lui manquaient pour décentraliser au maximum les processus de décision et se libérer des structures linéaires ou parallèles caractéristiques des lignes ou des cellules flexibles. Dès lors, une pièce entrant dans un atelier peut suivre des cycles d'usinage complexes grâce au gain de puissance apporté par l'informatique au système de pilotage humain (figure II-5, flèches 3). L'atelier flexible constitue ainsi un compromis efficace entre la flexibilité/réactivité des machines servies manuellement et la productivité des lignes de transfert.

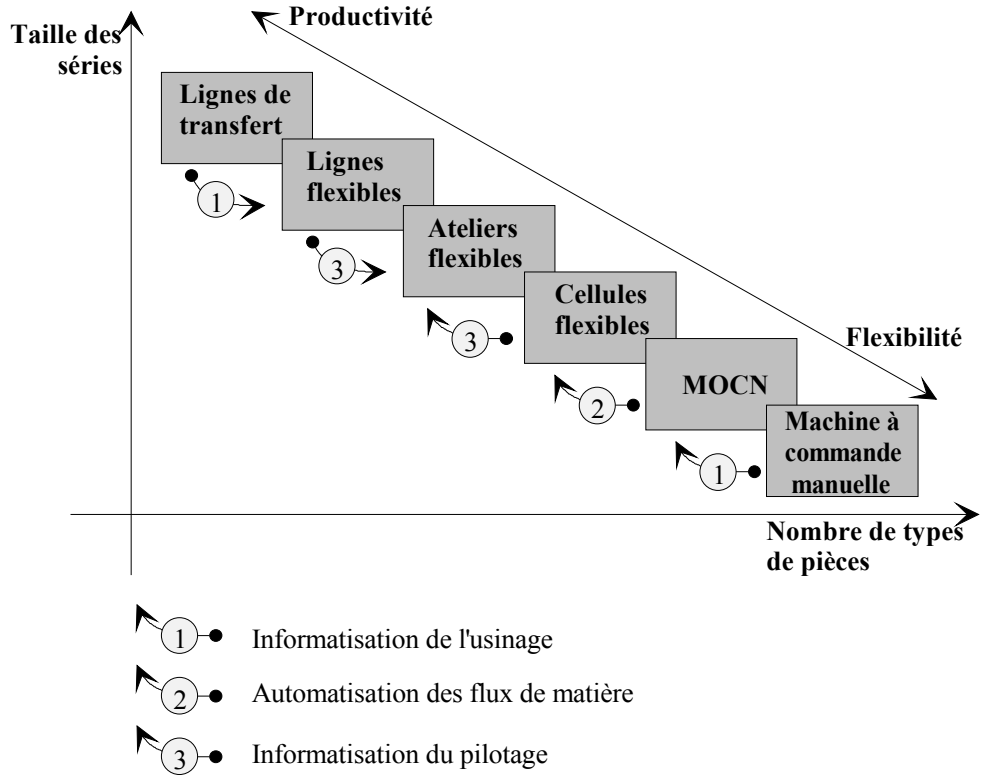


Figure II-5 : *Les différents types de production*

II.3.1. L'atelier flexible

II.3.1.1. Définition

D'après [GIA 94] un atelier flexible (ou Flexible Manufacturing System, FMS) est « *un ensemble de machines reliées par un système de manutention automatique, le tout contrôlé par un système central informatisé* ». Cette définition est en parfaite cohérence avec les trois phases énumérées ci-dessus. Cependant elle doit être complétée : ce ne sont pas ses composants qui font d'un système de production un atelier flexible mais ses performances en termes de flexibilité et de réactivité.

Un atelier flexible doit donc être capable de produire, à une cadence relativement élevée (intermédiaire entre la ligne de transfert et les ateliers manuels), des pièces différentes dans un ordre et/ou avec des lots de taille aléatoire. La variabilité des pièces est limitée par des contraintes de conception mais elle doit être supérieure à celle obtenue par la simple utilisation des MOCN. En effet, l'atelier flexible doit permettre des cycles de fabrication complexes, variables, passant par des machines différentes ou dans un ordre différent. Ces caractéristiques le distinguent de la ligne flexible (un cycle de fabrication, plusieurs usinages possibles) et de la cellule flexible qui n'effectue généralement qu'un nombre restreint d'opérations sur une même pièce.

Pour Roger Bonetto [BON 87], on doit ajouter à la définition précédente deux critères supplémentaires : le niveau d'automatisation et la désynchronisation du travail entre l'homme et la machine.

II.3.1.2. Constituants

Pour définir les différents éléments constitutifs d'un atelier flexible nous devons revenir à la définition d'un système de production. Nous avons vu que celui-ci est décomposable en quatre notions essentielles (les entités, les ressources, les tâches et le pilotage). Dans le cadre de la production flexible, et, plus précisément de l'atelier flexible, nous pouvons préciser ces différentes notions.

- **Les entités :** D'après la définition de l'atelier flexible (voire même de la flexibilité), les entités produites doivent pouvoir être toutes différentes. Cependant leur variabilité est limitée en raison de contraintes techniques évidentes (matière, taille des pièces, ...). De plus les ateliers flexibles sont plus particulièrement adaptés aux entreprises manufacturières et de transformation. Les entités manipulées seront donc généralement des pièces mécaniques usinées. Malgré ces restrictions, nous sommes loin des principes du « total FMS[5] » développé au Japon au début des années quatre-vingts (cet atelier devait pouvoir usiner, 24h/24, n'importe quelle pièce dans un cube d'acier [BON 87]). Les pièces usinées dans un atelier flexible appartiennent généralement à une famille particulière.

- **Les Ressources :** D'après [BON 87] un atelier flexible doit être constitué de trois éléments principaux : un système de fabrication, un système de manutention et un système de montage. Cette décomposition n'est cependant que rarement appliquée. On trouve souvent des ateliers de fabrication distincts des ateliers d'assemblage. Quoi qu'il en soit, les ressources sont généralement constituées de MOCN, de robots de manutention ou d'assemblage et d'un système de convoyage complexe, auxquels il faut ajouter les outils de réglage et de test.

- **Les Tâches :** Comme nous l'avons vu section II.1, une tâche est l'application d'une ressource à une entité. L'atelier flexible est donc susceptible d'effectuer un nombre très important de tâches et il est impossible d'en dresser une liste exhaustive.

- **Le Pilotage :** La principale évolution ayant donné naissance aux ateliers flexibles est l'informatisation et l'utilisation de processus de décision complexes pour le pilotage

---

[5] Flexible Manufacturing System

des systèmes de production. De fait, la principale caractéristique d'un atelier flexible tient à son système de pilotage. Celui-ci doit être capable de gérer dynamiquement un grand nombre de tâches, d'entités et de ressources. Il doit veiller à la parfaite coordination de tous les composants de l'atelier tout en gérant les aléas apparaissant de façon quasi-permanente au cours de la production. Le système de pilotage d'un atelier flexible comprend généralement deux unités distinctes : **l'ordonnancement** chargé de la planification prédictive et **le suivi** chargé de l'application du plan. De la répartition exacte des fonctions entre ces deux unités dépend la politique de gestion de l'atelier (court, moyen ou long terme) et sa capacité à gérer l'aléa (réactivité).

Le but du travail que nous présentons ici est la définition d'architectures de production adaptées aux principes de l'ingénierie concourante et du JAT. L'atelier flexible semble être le système de production le mieux adapté mais il doit lui-même être conçu pour répondre aux besoins de ces nouvelles approches. Cela signifie que les ressources et le pilotage (seules entités stables dans un atelier flexible) doivent pouvoir supporter de nouvelles productions sans restructurations majeures et sans surcoûts importants. Dans le chapitre suivant nous allons présenter les différentes architectures de pilotage utilisées dans les ateliers flexibles. Elles ont, en effet, une action primordiale sur les capacités de l'atelier. Nous verrons par la suite comment le choix de certaines ressources (système de manutention) peut, lui aussi, avoir une influence importante sur la flexibilité et sur la réactivité (section II.3.5).

II.3.2. Les Architectures de pilotage

Les tâches imparties au système de pilotage sont très diverses. Il doit, en effet, effectuer le transfert des ordres de haut niveau émanant des fonctions « décisionnelles » de l'entreprise (essentiellement la GPAO) vers le système physique de production. Pour cela, ces ordres doivent être décomposés en opérations élémentaires regroupées dans la gamme de fabrication. L'ensemble des opérations doit alors être séquencé (ordonnancement) en fonction des machines disponibles. Le système de pilotage doit ensuite s'assurer de la bonne marche de la production et transmettre, vers la GPAO, un compte-rendu de fabrication.

Cette variété d'opérations impose pour le pilote une décomposition modulaire complexe en fonction des règles de gestion choisies. Nous appelons architecture de pilotage d'un atelier flexible sa décomposition en différents groupes fonctionnels et la répartition, dans les différents groupes, des fonctions de décision.

Au fur et à mesure du développement des méthodes informatiques et de la montée en puissance du matériel, l'organisation des pilotes d'atelier flexible s'est affinée pour aboutir à des architectures modulaires complexes, basées sur des principes de gestion de plus en plus performants.

II.3.2.1. Les architectures centralisées

Nous ne les présentons que pour mémoire. Issues des premiers balbutiements de l'informatique moderne cette architecture concentre la gestion de l'atelier au sein d'un unique contrôleur « qui voit tout et fait tout ».

L'ensemble des ordres de fabrication donne lieu à un ordonnancement prévisionnel des différentes tâches (intégrant donc l'ensemble des ressources et des entités). Le calcul de l'ordonnancement est basé sur une charge moyenne des machines ce qui permet de conserver

une certaine souplesse d'exécution (maintenance, réparation, ...). Le pilote peut ainsi continuer à suivre le plan prévisionnel malgré l'apparition de perturbations, dans la mesure où celles-ci restent limitées. Des perturbations trop importantes entraînent cependant un décalage important par rapport à la planification prévisionnelle. Un nouvel ordonnancement doit alors être calculé pour la période de temps restant (figure II-6).

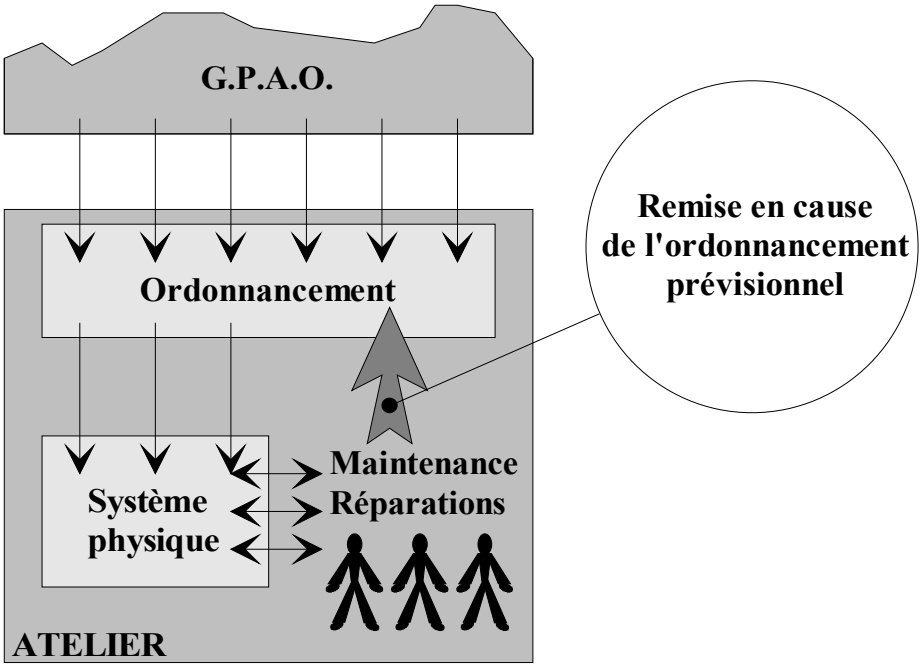


Figure II-6 : *Architecture centralisée*

La maintenance et la gestion des perturbations sont le plus souvent gérées manuellement.

L'inconvénient majeur de cette politique de gestion est le temps de calcul de l'ordonnancement. Etant donné le nombre important de paramètres mis en jeu (nombre de ressources, d'entités, de tâches, ...), le calcul de l'ordonnancement nécessite généralement plusieurs jours. Il est souvent simplifié aux dépens de la flexibilité (limitation du nombre de pièces différentes fabriquées simultanément). L'ordonnancement est ensuite appliqué pendant une période au moins supérieure au temps de calcul (généralement quelques semaines). De plus, la charge des machines est volontairement limitée afin de permettre les opérations de maintenance. Ce type de pilotage n'est, en effet, absolument pas réactif et l'ordonnancement ne peut être appliqué qu'au prix d'une robustesse à toute épreuve. L'apparition trop fréquente de perturbations au cours de la production entraîne en effet une explosion combinatoire : le temps de calcul des ordonnancements successifs bloque rapidement la production.

Ce type de contrôleur monolithique est immédiatement saturé lorsque la taille des ateliers croît ou lorsque des aléas apparaissent en cours de production (du fait de l'explosion combinatoire). De plus il ne permet pas de modifier les cycles de production ou la structure de l'atelier. Ceux-ci sont en effet des paramètres essentiels dans le calcul de l'ordonnancement prévisionnel et leur modification risque de rendre le logiciel inutilisable. Enfin, ce type de planification prévisionnelle sur plusieurs semaines est contraire aux principes du flux tiré et ne peut en aucun cas permettre la fabrication en Juste A Temps.

Diamétralement opposées aux principes des structures flexibles, les architectures centralisées ont été rapidement abandonnées au profit de structures plus complexes mais aussi plus souples d'utilisation.

II.3.2.2. Les architectures hiérarchisées

Tout en conservant le principe de la planification prévisionnelle, les architectures hiérarchisées opèrent une division successive du travail au fur et à mesure qu'on descend dans les couches hiérarchiques. A chaque étape, le travail à effectuer est réparti entre plusieurs modules fonctionnels qui traitent chacun un aspect particulier du problème (principe de délégation). Il n'y a pas dans cette architecture de liaisons transversales entre les différents modules (figure II-7).

Par rapport au modèle centralisé, la décomposition hiérarchique permet de limiter l'impact des perturbations. Ainsi, dans l'exemple présenté figure II-7, les perturbations causées par l'arrêt d'une machine peuvent rester confinées au module « usinage » si une deuxième machine peut recevoir le surplus de travail correspondant. L'apparition de perturbations de plus grande ampleur entraîne cependant la remise en cause des décisions de haut niveau. C'est en particulier le cas lorsqu'un dépassement de temps trop important intervient. Le fonctionnement de l'ensemble de l'atelier est alors perturbé et un ré-ordonnancement devient nécessaire.

Le principe de fonctionnement de chacun des modules est relativement simple. Chacun d'eux est doté de deux fonctions principales :

- **Fonction planification :** Les ordres reçus des niveaux supérieurs sont traités et décomposés en sous-ordres transmis au niveau inférieur.

- **Fonction d'observation :** Chaque module du niveau inférieur envoie un compte-rendu. Ceux-ci sont collectés, traités, et forment un compte-rendu global transmis au niveau supérieur.

Dans les premiers systèmes de pilotage utilisant cette structure, seul le niveau supérieur était doté d'une réelle capacité de décision et les délégations successives n'étaient que des raffinements de ses choix initiaux. Une telle architecture est totalement incompatible avec les besoins de réactivité des ateliers modernes. En effet, le moindre incident, quel que soit le niveau auquel il apparaisse, doit, s’il perturbe le plan de production, remonter l'ensemble de la hiérarchie jusqu'à la planification. Ce principe pose alors les mêmes problèmes que l'architecture centralisée.

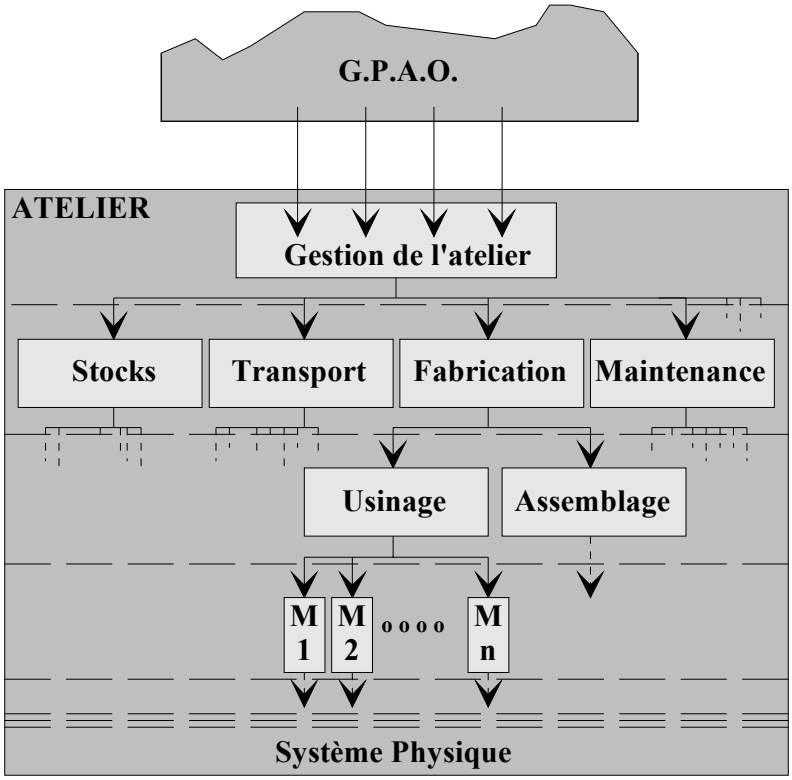


Figure II-7 : *Architecture hiérarchique*

La structure décisionnelle se présente sous la forme d'un organigramme.

Pour éviter de remettre en cause l'ensemble du plan de production dès qu'un événement perturbateur apparaît au cours du cycle de production, les modules successifs peuvent être dotés de capacité décisionnelle. Dès lors, la remontée des informations peut être stoppée dès qu'un module dispose de l'ensemble des « commandes » nécessaires pour absorber la perturbation. Cependant des liaisons transversales deviennent rapidement nécessaires, en particulier vers le système de transport (souvent concerné par la gestion des perturbations) ou vers la maintenance. Le rôle prépondérant de la planification est alors remis en cause et l'architecture hiérarchisée se mue en une architecture décentralisée coopérative, plus souple mais aussi plus difficile à gérer.

II.3.2.3. Décentraliser pour mieux réagir

Le passage d'une architecture de pilotage hiérarchisée à une architecture décentralisée, motivé par la dualité ordonnancement/réactivité, s'est amorcé dès le début des années quatre-vingts pour progressivement prendre corps avec le développement de méthodes de conception adaptées telles que la méthode CODECO[6].

Déjà, en 1983, le « total FMS » que nous avons cité plus haut utilise une architecture de pilotage dans laquelle les modules ne sont plus résolument organisés hiérarchiquement : chaque tâche est gérée par un module indépendant et l'ensemble est

[6] COnduite DEcentralisée COordonnée

organisé sous la forme d'un graphe de communication sans qu'une quelconque hiérarchie ne soit imposée a priori (figure II-8).

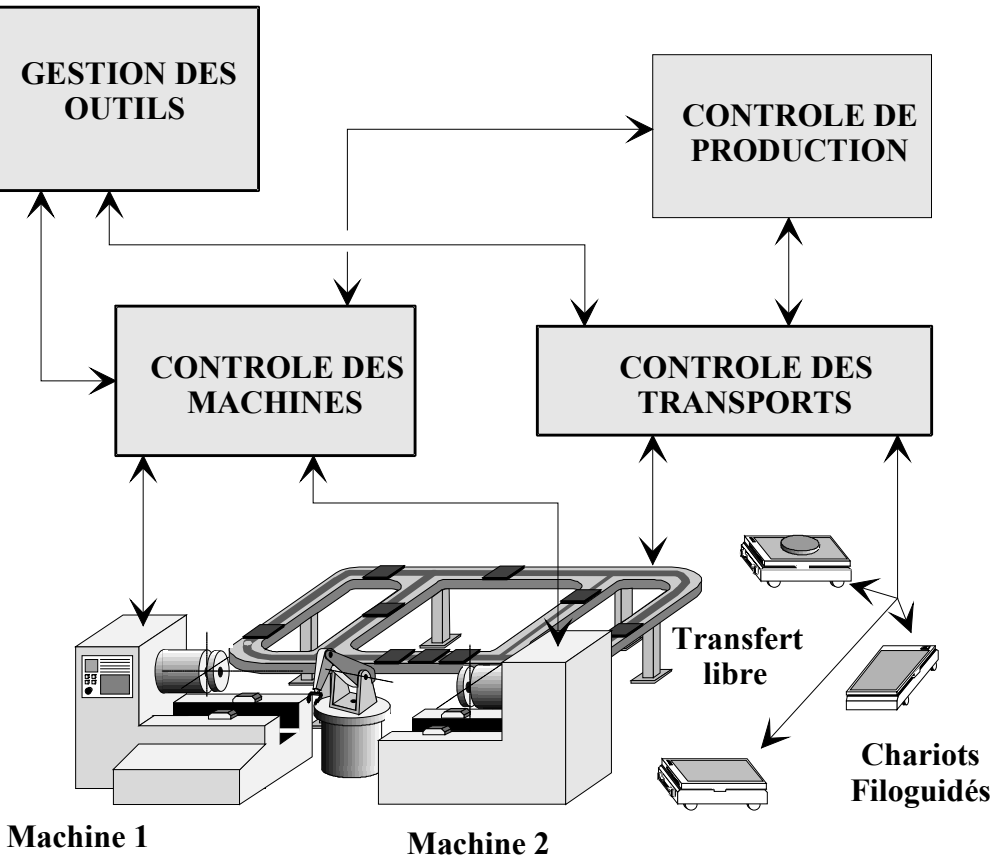


Figure II-8 : *Décentralisation par les tâches*

Bien qu'elle ne soit pas imposée, la structure hiérarchique est intrinsèque à ce type d'architecture : les tâches de suivi de la production ou de lancement des opérations apparaissent comme des entités de haut niveau distribuant le travail aux autres modules. L'existence de voies de communication latérales entre les modules de plus bas niveau offre cependant à chacun d'eux une marge de manoeuvre beaucoup plus importante que dans les modèles strictement hiérarchisés et la « décentralisation par les tâches » constitue déjà une progression importante vers une autonomie des différents modules décisionnels.

Plus récemment plusieurs architectures génériques délibérément orientées vers une décentralisation du processus de décision en atelier ont été proposées. Ainsi, en France, les années quatre-vingts ont vu la naissance de l'architecture CODECO (COnduite DEcentralisée COrdonnée d'atelier) développée au Laboratoire d'Automatique de Grenoble [KAL 85]. Dans ce modèle le processus de décision est réparti sur deux niveaux hiérarchiques : au niveau supérieur un « coordinateur d'atelier » et au niveau inférieur un ensemble de modules de « conduite locale » coopérant directement les uns avec les autres pour absorber les perturbations (figure II-9). De plus, cette organisation s'accompagne d'une décomposition verticale de chaque processus de décision intégrant directement les différents modes de communication : hiérarchique (c'est à dire avec les niveaux décisionnels supérieurs ou inférieurs) ou transversal pour la coopération entre modules d'un même niveau (essentiellement entre les conduites locales d'un même atelier mais aussi, éventuellement, entre coordonnateurs d'atelier différents).

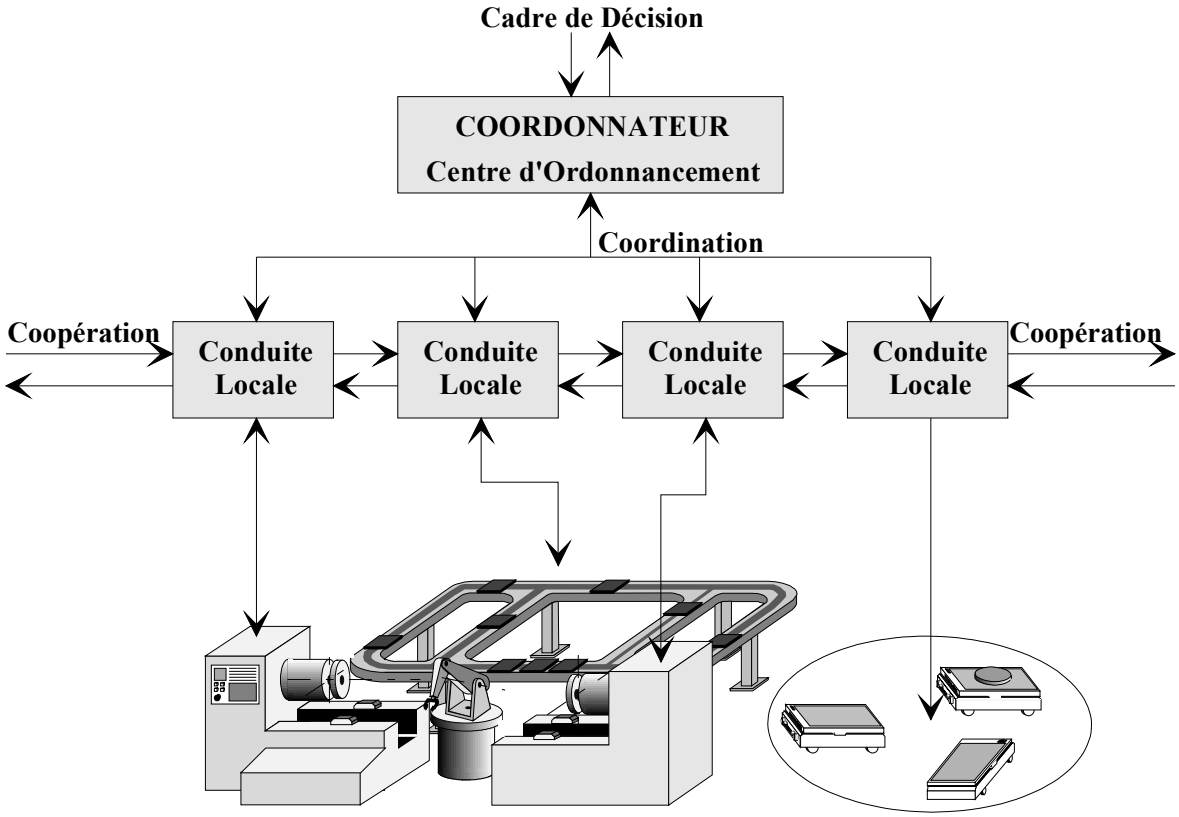


Figure II-9 : *Organisation de l'architecture CODECO*

Le passage d'une architecture centralisée/hiérarchisée à une architecture décentralisée est en grande partie rendue possible par l'introduction des nouvelles techniques de l'informatique (systèmes d'information, bases de données partagées, intelligence artificielle) dans les ateliers de production. L'intelligence artificielle autorise en effet chaque centre décisionnel à planifier localement ses actions en fonction des informations locales (l'état de la ressource associée), latérales (états des ressources voisines) et des informations générales (en particulier les gammes de fabrication) stockées dans le système d'information.

Plusieurs architectures décentralisées, similaires au modèle CODECO, ont été proposées dans la littérature, la plupart d'entre elles conservant une organisation hiérarchique en deux niveaux : supervision/coordination et coopération (voir par exemple [CRA 89]). Les systèmes de pilotage totalement décentralisés sont restés marginaux, principalement en raison des problèmes de synchronisation entre les contrôleurs locaux. De fait la gestion décentralisée pose un certain nombre de problèmes : chaque contrôleur doit avoir une vue suffisamment large de l'atelier (pour que la gestion de celui-ci soit au moins sub-optimale) mais la complexité de son processus de décision est directement fonction de la quantité d'information reçue.

Une solution élégante a été proposée au Laboratoire d'Automatique et d'Informatique de Marseille : le système de pilotage devant - entre autre - assurer la synchronisation entre les flux de matières (matière première, produits finis ou semi-finis) et les flux d'information (gammes de fabrication, gammes d'assemblage, ...), D. Brun-Picard propose d'associer l'information à la matière [BRU 93]. A chaque unité de matière circulant dans l'atelier est associé un « jeton » logique contenant l'ensemble des informations sur le cycle de vie de l'entité matérielle. Chaque poste recevant la « pièce » dispose alors directement des informations nécessaires à son traitement et à sa destination ultérieure. Malheureusement une

telle organisation, bien qu'elle permette d'assurer que chaque contrôleur local dispose de l'information minimale nécessaire, ne leur permet pas d'anticiper les actions en fonction des travaux en cours pour prévoir, par exemple, les transferts d'outils, le chargement des programmes d'usinage (pour une machine outil) ou la présence, en fin d'usinage, d'un chariot ou d'une palette (pour le système de transport).

Malgré les difficultés inhérentes aux systèmes décentralisés, il peut être intéressant, pour augmenter la réactivité d'un système de production, de supprimer totalement - lorsque le contexte de production le permet - les modules de supervision et de passer d'un modèle décentralisé coordonné (type CODECO) à un modèle coopérant dans lequel chaque module de conduite locale est totalement autonome dans sa fonction de décision et ne réagit qu'en fonction de ses propres observations, c'est-à-dire en fonction de son **environnement**.

Dans le modèle de réactivité décrit ci-dessus, une telle organisation est intéressante à plusieurs points de vue :

- La masse des aléas apparaissant dans un système de production complexe (tel qu'un atelier flexible) est telle que le fonctionnement en mode normal ne représente qu'une petite partie du temps de production. Dès lors le mode « normal » d'un système de production (i.e. le mode courant) est, de fait, le mode dégradé[7], c'est-à-dire les procédures de reprise et d'élimination des perturbations. Il est donc souvent illusoire de chercher à calculer un ordonnancement global (qu'il soit prévisionnel ou calculé en temps réel) sachant que, selon une très forte probabilité, les données physiques utilisées pour le calcul ne seront plus valides lors de son utilisation.

- La mise en place d'une architecture de contrôle d'atelier ou de cellule flexible est un processus coûteux financièrement et qui demande un temps d'étude important afin de maîtriser l'ensemble des interactions entre les sous-systèmes. La recherche permanente de la flexibilité des moyens de production est, d'ailleurs, en grande partie justifiée par le coût de mise en place d'un système de production dédié à une série particulière (les moyens de production ne sont dédiés que pour la grande ou la très grande série). Le développement d'architectures totalement décentralisées dans lesquelles chaque contrôleur est dédié à une ressource permet d'associer ou d'éliminer à volonté les ressources adaptées à une production particulière ce qui permet, dans un contexte d'ingénierie concourante, de limiter les temps de conception de l'outil de production.

- La remise en cause des modèles hiérarchiques ne se limite pas aux couches basses de l'appareil de production. Depuis quelques d'années, c'est l'ensemble de la structure de l'entreprise (traditionnellement décomposée en secteurs et ateliers fortement cloisonnés) qui est remise en question : la recherche de la réactivité impose en effet la transition vers une architecture fédératrice. Le contrôle décentralisé vers les ressources permet de supprimer partiellement ces cloisonnements en autorisant une mobilité des structures : une ressource peut alors « migrer » d'un atelier à l'autre voire être partagée par plusieurs ateliers.

---

[7] D'après Michel D'Iribarne, conférence pour l'inauguration de l'AIP Lorraine et les dix ans des AIP (Nancy 1993)

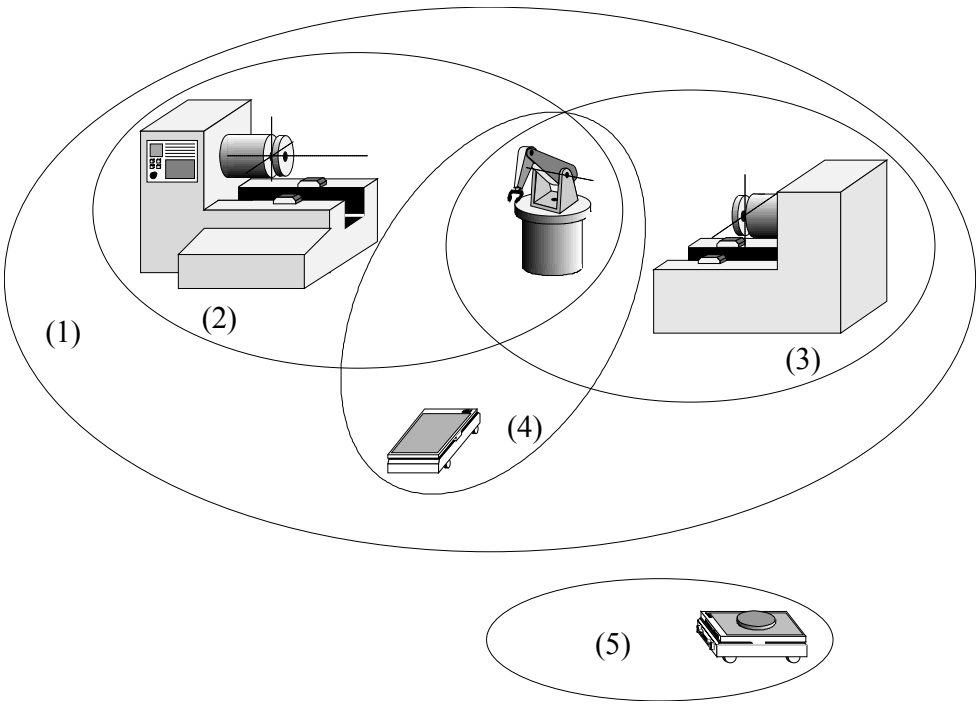


Figure II-10 : *Pilotage guidé par les ressources*

Chaque ressource est contrôlée localement en fonction de son environnement perçu et de son état interne (la perception de chaque contrôleur est directement fonction de son état grâce à un processus de scrutation[8])
(1) : environnement du robot de chargement
(2-3) : environnement des machines outils
(4-5) : environnement des chariots de transport

L'architecture décentralisée « sensori-motrice » permet à chaque contrôleur d'anticiper sur les actions à venir en scrutant l'état de son environnement direct (machines voisines, robots, systèmes de transport) ou indirect (base de donnée partagée, plan de production issu de la GPAO, ...). Par contre, cette structure impose une redéfinition complète des contrôleurs locaux par rapport aux modèles traditionnellement utilisés dans les systèmes de production : ce mode de gestion impose une répartition plus étendue des processus de décision, en particulier vers les systèmes de transitique qui, seuls, peuvent avoir une perception globale de l'état du système. Ainsi c'est au contrôleur du système de manutention (par exemple un chariot mobile) que revient le choix de la machine cible ; l'arrêt momentané d'une machine pour maintenance ou réparation doit donc être géré par le contrôleur du chariot qui va re-diriger le travail vers une autre machine ou vers un stockage temporaire.

La suite de ce chapitre est consacrée à une étude plus précise des contrôleurs locaux. Nous proposerons en particulier l'utilisation des nouvelles techniques de l'intelligence artificielle pour la gestion processus de décision locaux. Nous montrerons enfin les contraintes particulières que cette architecture entraîne sur la conception d'un système de transport en atelier flexible.

---

[8] « Situated Perception »

II.3.3. Des contrôleurs locaux intelligents et adaptatifs

II.3.3.1. Un environnement plus complexe

Dans le modèle décentralisé « sensori-moteur », chaque ressource (i.e. chaque contrôleur) doit disposer de fonctions de perception et de scrutation de façon à acquérir les informations nécessaires à son bon fonctionnement. Cette capacité doit aussi être supportée par les différents mécanismes sensoriels dont dispose le contrôleur, que ces mécanismes soient physiques (en particulier la vision) ou logiciels (interrogation de bases de données). L'environnement propre à chaque ressource est donc composé d'éléments physiques, perçus par des capteurs, et d'éléments « virtuels », contenus dans les différentes bases de données de l'atelier. Ces dernières contiennent des informations de haut niveau qui permettent aux ressources d'orienter les processus de scrutation (attente d'un type de pièce particulier) et d'anticiper sur le travail à venir (chargement d'un outil sur un barillet, ...). Cependant, en cas de conflit entre les deux sources d'information (réelles et virtuelles), la ressource prend prioritairement en compte l'information directement perçue. Celle-ci peut alors éventuellement être utilisée pour la mise à jour de la base de donnée. La figure II-11 présente un exemple d'interactions entre plusieurs ressources.

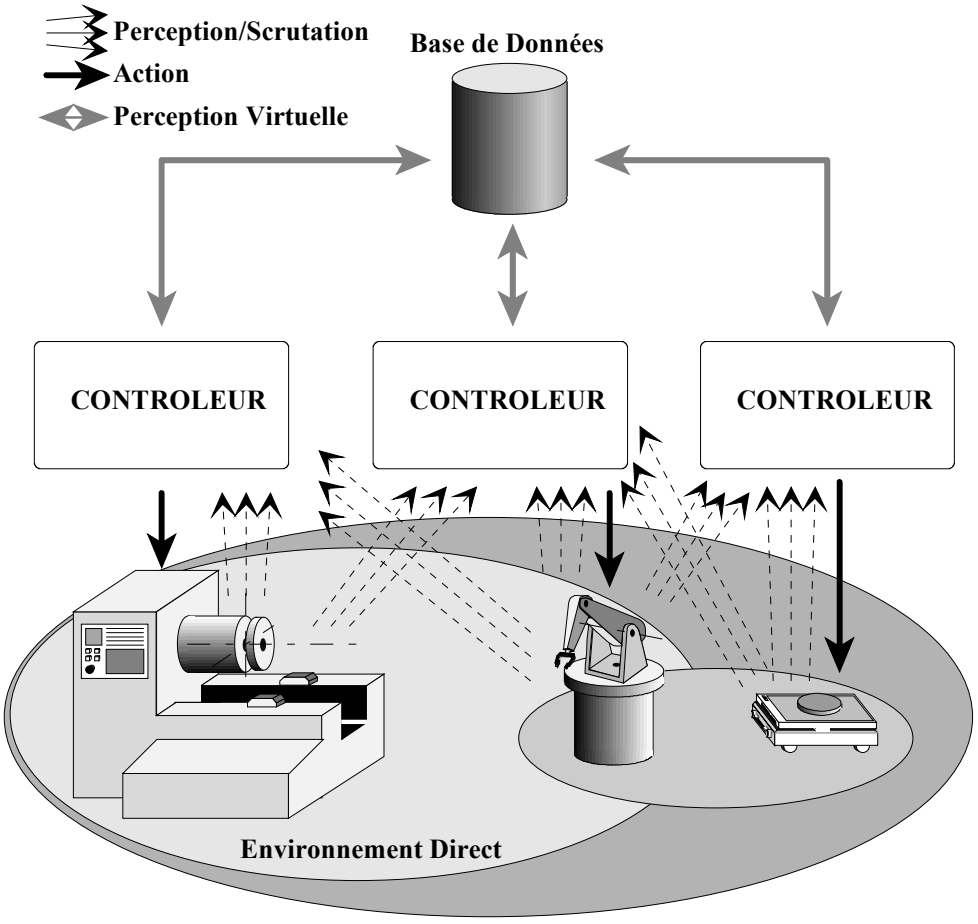


Figure II-11 : *Exemple d'interaction entre plusieurs ressources : chargement d'une pièce sur une machine outil.*

Bien que le chariot mobile ne soit pas encore perçu par le contrôleur de la machine outil, celle-ci peut se préparer à recevoir la pièce par une simple scrutation des ordres généraux émanant de la GPAO (plan de production).

II.3.3.2. Des contrôleurs flexibles et réactifs

La complexité des informations « sensorielles » traitées par un contrôleur local dans une architecture totalement décentralisée est beaucoup plus importante que celle des informations logiques utilisées par les systèmes actuels. C'est pourquoi il est indispensable, préalablement à la création d'une telle organisation, de proposer des modèles de contrôleurs adaptés.

Chaque contrôleur local doit être capable de réaliser l'interface entre ses perceptions, quelles qu'elles soient (réelles ou virtuelles[9], internes ou externes, ...), et la ressource physique à laquelle il est associé. En outre il doit interpréter ses perceptions pour prévoir l'évolution de son environnement et les diriger (scrutation, perception active) de façon à accélérer les traitements ultérieurs. Ses performances sont donc directement liées à l'existence de deux boucles sensori-motrices différentes. La première est ouverte sur l'environnement par l'intermédiaire de l'action (grande boucle) et la seconde (petite boucle), interne, est dédiée au processus de scrutation[10] (figure II-12).

Le but de l'architecture décentralisée est de permettre un fonctionnement robuste, réactif et flexible des systèmes de production. Contrairement aux modèles centralisés ou hiérarchisés qui sont intrinsèquement rigides, ce modèle autorise en effet l'exploitation immédiate des potentialités de chacun des contrôleurs locaux pour l'élimination immédiate (et avant que leur impact ne soit trop important) des perturbations apparaissant continuellement dans l'atelier. Par contre, ce principe impose, pour chaque contrôleur, des capacités de flexibilité/réactivité et une robustesse maximale qui est généralement hors de portée des architectures de contrôle classiques.

La mise en place de modules interagissant par des boucles sensori-motrices complexes avec un environnement faiblement structuré (à l'inverse des modèles théoriques utilisés pour la planification prévisionnelle) impose l'utilisation de nouvelles techniques de contrôle permettant un couplage étroit entre perception et action (action motrice ou perception active). C'est pourquoi nous proposons d'organiser les contrôleurs autour de deux techniques modernes de l'intelligence artificielle : les architectures de contrôle réactif (« Behavior-Based-Control ») et le Neurocontrôle. L'utilisation de techniques complémentaires n'est cependant pas exclue, par exemple pour doter certaines ressources de capacités cognitives plus avancées (manipulation symbolique par des systèmes experts et utilisation d'architectures issues de l'Intelligence Artificielle Distribuée, ...), mais nous avons concentré notre propos sur les systèmes plus résolument réactifs.

---

[9] Les perceptions réelles proviennent des capteurs et de l'environnement physique. Les perceptions virtuelles sont issues de processus de lecture vers une base de données ou un espace logique partagé par les ressources (« blackboard »). Voir figures II-11 et II-12.

[10] Yves de Ribaupierre : Conférence d'ouverture de NSI'94 (Chamonix, mai 1994)

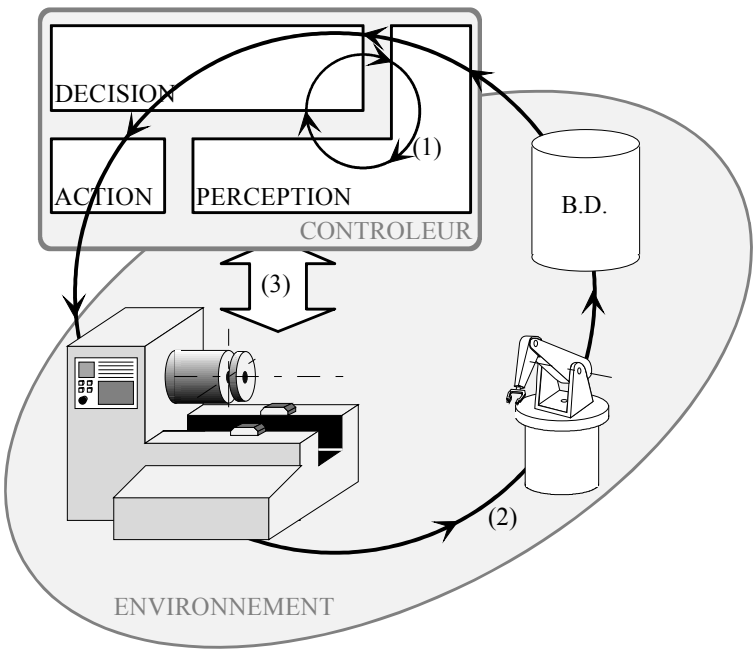


Figure II-12 : *Echanges d'information contrôleur/environnement*[11]

Le contrôleur est associé à une machine outil mais son environnement inclut un robot de chargement et une base de données BD (représentés en blanc).

(1) : Petite boucle sensori-motrice
(2) : Grande boucle sensori-motrice
(3) : Information état/commande

II.3.4. De nouvelles techniques au service de la production

Les progrès récents de l'intelligence artificielle permettent de créer des architectures de contrôle sensori-moteur complexes capables d'interagir « intelligemment » avec un environnement complexe et non structuré (typiquement le milieu naturel). Ces architectures n'ont été, jusqu'à présent, que très occasionnellement utilisées en environnement de production. L'architecture décentralisée « sensori-motrice » constitue le cadre d'application idéal de ces nouvelles techniques.

II.3.4.1. Le contrôle sensori-moteur

Les années quatre-vingts ont vu l'apparition d'une nouvelle branche de l'intelligence artificielle délibérément orientée vers la réalisation de processus automatiques « intelligents » mais, à l'inverse de l'intelligence artificielle « classique », en privilégiant l'aspect comportemental à l'aspect décisionnel [BRO 86]. L'un des buts de cette approche est de permettre la réalisation de robots aptes à évoluer dans un environnement dynamique et non structuré (donc réactifs et robustes) tout en limitant leur coût afin d'autoriser la création d'essaims de robots.

---

[11] La décomposition architecturale du contrôleur en trois modules « perception », « décision » et « action » n'est utilisée qu'à titre illustratif. Les principes organisationnels retenus pour les contrôleurs sensori-moteurs sont présentés dans le chapitre III.

Les principes fondateurs de cette nouvelle approche (dénommée par ses fondateurs « Behavior-Based-Control » ou « Réactive Control ») reposent sur un couplage étroit entre la perception et l'action : les architectures construites suivant ce principe privilégient l'usage de boucles sensori-motrices courtes mais nombreuses par rapport aux processus de décision basés sur une analyse plus fine des informations perçues moyennant une extraction/manipulation de symboles. Cette organisation sensori-motrice des traitements permet de construire, pour un coût relativement modique, des architectures de commande répondant en temps réel à des stimuli complexes.

Ces architectures, bien qu'elles aient démontré leur capacité à contrôler des robots (robots mobiles ou bras robots), sont jusqu'à maintenant restées en marge des milieux industriels. Elles représentent pourtant une solution potentielle au problème de la décentralisation dont elles partagent le principe fondamental : l'action du robot (de la ressource) est directement fonction de l'environnement perçu. Le chapitre suivant (chapitre III) est entièrement dédié à la description de ces nouvelles architectures et à leurs applications en productique.

II.3.4.2. Apprentissage et adaptation : le connexionnisme et le Neurocontrôle

L'utilisation de contrôleurs locaux permet d'assurer la réactivité et la flexibilité du système de production (entre autres par la portabilité des ressources) mais il est difficile, en raison de la complexité des systèmes de production, d'en garantir la productivité. C'est pourquoi ce type d'architecture semble a priori essentiellement destiné à la production unitaire et à la petite série pour lesquelles le gain en temps de conception prime sur la perte de productivité. Pour permettre d'assurer la productivité en moyenne série, nous proposons d'adjoindre aux contrôleurs des mécanismes d'apprentissage permettant d'adapter leur comportement à la production en cours de façon à accélérer le traitement des pièces ou des opérations les plus courantes. L'utilisation de mécanismes d'apprentissage permet, en outre, de simplifier la construction des contrôleurs locaux et, en autorisant l'acquisition de nouveaux comportements et/ou la modification des comportements existants, d'augmenter encore les capacités de flexibilité et de réactivité [WER 87b]. Nous verrons au cours du chapitre III que les architectures réactives souffrent de l'absence de tels mécanismes.

Parmi tous les mécanismes d'apprentissage disponibles, nous avons retenus les architectures connexionnistes ou « réseaux de neurones » qui présentent plusieurs intérêts dont celui d'être intrinsèquement des architectures sensori-motrices. En outre les réseaux de neurones ont déjà été couramment appliqués à des problèmes de contrôle de processus (essentiellement dans un cadre de recherche, en particulier pour le contrôle de bras manipulateurs [MAR 90b][RAB 92] et pour la saisie au vol [MIL 87]). C'est pourquoi nous proposons, pour les contrôleurs locaux, une architecture unifiée basée à la fois sur les principes généraux du « Behavior-Based Control » (organisation macroscopique) et sur les modèles connexionnistes (organisation microscopique).

II.3.5. Un cas particulier : le système de transport

II.3.5.1. Caractéristiques générales du système de transport

Parmi les différentes ressources composant un atelier flexible, le système de transport (ou système de « transitique ») occupe une place très particulière. Il est, en effet,

particulièrement exposé aux aléas puisque tout événement, interne ou externe, prévu ou imprévu, influence à court ou moyen terme le système de transport en modifiant la destination des pièces circulant dans l'atelier (redirection vers une nouvelle machine, transport d'outils,...), en générant des en-cours imprévus (pièces sur palettes) et, plus généralement, en alourdissant la charge du système. En outre, le système de transport, qui met en oeuvre des éléments mécaniques très mobiles (palettes, chariots filoguidés, etc. ...) est lui-même générateur de perturbations qu'il doit gérer de façon interne afin de ne pas bloquer les tâches de production en cours. Il doit donc disposer d'une bonne capacité de réactivité « interne » pour absorber les aléas qu'il génère. Plus encore que pour les autres ressources, l'expression « le mode de fonctionnement normal est le mode dégradé » s'applique au système de transport. De ce fait, il est généralement impossible de planifier l'enchaînement des actions de transport et la prise de décision doit, ici plus qu'ailleurs, être effectuée en temps réel en fonction de l'état de l'atelier.

De plus, si on excepte les systèmes de pilotage centralisés, le système de convoyage constitue souvent le seul élément physique fédérateur d'un atelier. Sa conception est donc soumise à de nombreux choix antérieurs (nombre et type de machines, disposition des cellules, etc.). Elle doit donc intervenir - séquentiellement - après les opérations de conception précise de l'atelier, d'équilibrage, ... En outre, l'architecture du système de transport risque d'être remise en cause dès qu'un ou plusieurs éléments de l'atelier doivent être modifiés. L'ajout d'une machine nouvelle peut ainsi entraîner la refonte complète du système de convoyage.

La plupart des systèmes de transport actuellement utilisés en milieu industriel (pour ne pas dire tous) est basée sur une infrastructure reliant l'ensemble des points à desservir par un ou plusieurs anneaux. C'est le cas pour les transferts libres (ou « marguerites ») mais aussi pour les chariots filoguidés à guidage visuel (ligne peinte sur le sol), électromagnétique (câble enterré) ou laser. Ce type de structure permet de résoudre le problème de la réactivité du transport en desservant systématiquement et dans un ordre préétabli l'ensemble des postes de l'atelier (ou un sous-ensemble dans le cas d'anneaux imbriqués). Pour être efficace, ce type d'organisation impose une robustesse à toute épreuve : l'existence d'une voie de communication unique entraîne, lorsqu'elle est coupée, l'arrêt immédiat de l'ensemble du système de transport et, conséquemment, de l'ensemble des ressources de l'atelier. De plus, l'existence d'une infrastructure « lourde » (au moins en ce qui concerne son étendue) ne permet pas de modifier aisément l'organisation interne de l'atelier (en particulier de rajouter une ou plusieurs machines) ni de faire migrer une machine d'un atelier à l'autre. Cette caractéristique contribue à figer la structure de l'entreprise et sa décomposition en ateliers cloisonnés, limitant ainsi les possibilités de coopération entre ateliers (l'appel à une ressource extérieure - le plus souvent un opérateur - devenant généralement indispensable).

Ce type d'architecture de convoyage, souvent considéré comme le plus réactif, n'atteint de bonnes performances qu'au prix d'une robustesse coûteuse et d'une rigidité difficilement compatible avec la mise en oeuvre d'un processus d'ingénierie simultanée ou avec la création d'une « fédération d'ateliers coopérants » [LEG 92]. En outre, comme nous l'avons déjà montré, ce type d'organisation est probablement une des raisons essentielles qui ont poussé vers l'utilisation de moyens de production toujours plus flexibles. Un système de transport plus souple joint à des techniques telles que la TGAO (Technologie de Groupes Assistée par Ordinateur) ou l'optimisation par réseaux de neurones [BIE 95a], permettrait à plusieurs ateliers de partager une machine dédiée à un type d'usinage particulier. Cette solution serait probablement moins onéreuse que l'utilisation, dans chaque atelier, de machines outils extrêmement flexibles capables de réaliser plusieurs types d'usinage.

II.3.5.2. Un système « distribuable »

Les caractéristiques particulières du système de transport, exposées ci-dessus, auxquelles s'ajoutent les relatives simplicité et permissivité de la tâche associée (le but premier, souvent unique, est que chaque pièce atteigne sa destination en temps voulu, généralement sans contrainte de chemin) nous ont conduits à y voir le sous-système le plus à même de recevoir les premiers modèles d'architecture totalement décentralisée. En effet, une architecture basée sur des robots mobiles totalement indépendants permettrait, tout en conservant une bonne réactivité (chaque robot peut réagir instantanément dès l'apparition d'une perturbation puisque son comportement n'affecte pas le reste du système, un robot peut être ajouté ou enlevé, ...), d'assouplir la structure du système de production. L'ajout ou la suppression d'une machine se traduit alors directement par une modification de l'environnement des robots sans avoir à modifier la fonction de décision ; de même un robot peut, le cas échéant, migrer vers un autre atelier. En outre, depuis une vingtaine d'années, les recherches en robotique intelligente et, plus particulièrement, en contrôle réactif, se sont majoritairement portées sur les Robots Autonomes d'Intervention (ROAR[12]), faisant progresser les capacités des robots mobiles de façon souvent spectaculaire. Conçus pour être utilisés en milieu hostile et non structuré, ces robots n'en sont que mieux adaptés aux environnements de production qui, malgré les efforts incessants pour limiter les aléas, ne sont, au mieux, que partiellement structurés [ARK 90b]. Le système multi-robots « MARTA » (projet européen ESPRIT) conçu pour des tâches de transitique dans les gares, les aéroports, les docks, ..., est d'ailleurs basé sur cette caractéristique propre aux environnements « humains ». L'aspect structuré permet d'effectuer une planification a priori de la trajectoire de chaque robot mais chacun d'entre eux conserve une capacité de décision afin de gérer localement toutes les tâches locales d'interactions avec l'environnement (dont les autres robots) [GIR 93].

Les principes architecturaux décrits dans ce chapitre sont essentiellement basés sur le développement de l'autonomie décisionnelle des ressources de l'atelier. La majorité des travaux sur des processus autonomes ayant porté sur la robotique mobile, nous avons choisi de nous inspirer des modèles de contrôleurs correspondants (plus particulièrement des modèles réactifs) pour développer un modèle générique de contrôleur de ressources. De même, afin de tester préalablement cette architecture sur un sous-système de l'atelier, nous l'avons appliquée initialement au seul système de transport : outre l'intérêt propre lié à sa décentralisation, cela nous permet d'exploiter plus directement les travaux antérieurs. C'est pourquoi, afin d'illustrer les principes architecturaux décrits dans ce chapitre, nous avons développé un premier outil de simulation d'un système de transport constitué de robots mobiles autonomes.

Le but de cette simulation n'est pas de concevoir un système de transport mais de servir de base d'étude pour un système distribué composé de robots extrêmement simples devant résoudre des tâches multiples (transport de matière entre différents points de l'atelier) et, en particulier, de tester les règles d'interaction entre les différentes composantes de l'atelier (machines, robots mobiles et tables d'entrée/sortie). Les « robots » simulés ici sont donc contrôlés par de simples règles perception-action et ne sont dotés d'aucune capacité cognitive ni de mécanisme d'apprentissage.

---

[12] Remote Operating Autonomous Robots

II.3.5.3. Simulation et résultats

L'atelier utilisé pour la simulation comporte trois machines, un point d'entrée et un point de sortie. Le nombre de robots de transport est variable, choisi entre 1 et 13. Chacun des cinq points de chargement/déchargement (voir figure II-13) est signalé par une balise supposée visible depuis tous les points de l'atelier. Cette supposition n'est pas improbable si, par exemple, les balises sont situées en hauteur. Elle contraint cependant fortement la conception physique de l'atelier et nous cherchons à autoriser l'existence de zones « d'ombre » dont les robots pourraient sortir grâce à des procédures de coopération inter-robots (voir section suivante).

Les matières premières (ou les pièces semi-finies) entrent dans l'atelier par la table d'entrée et les pièces usinées en sortent par la table de sortie. Les trajets intermédiaires sont déterminés aléatoirement à chaque fois qu'une pièce est chargée sur un chariot. En moyenne chaque pièce subit trois usinages avant de sortir de l'atelier. A chaque usinage correspond un « temps de process » choisi aléatoirement entre zéro et un « temps de process maximum » fixé entre 100 et 700 « unités de temps »(UT). Ces règles de gestion de l'atelier obligent le système de transport à être totalement réactif.

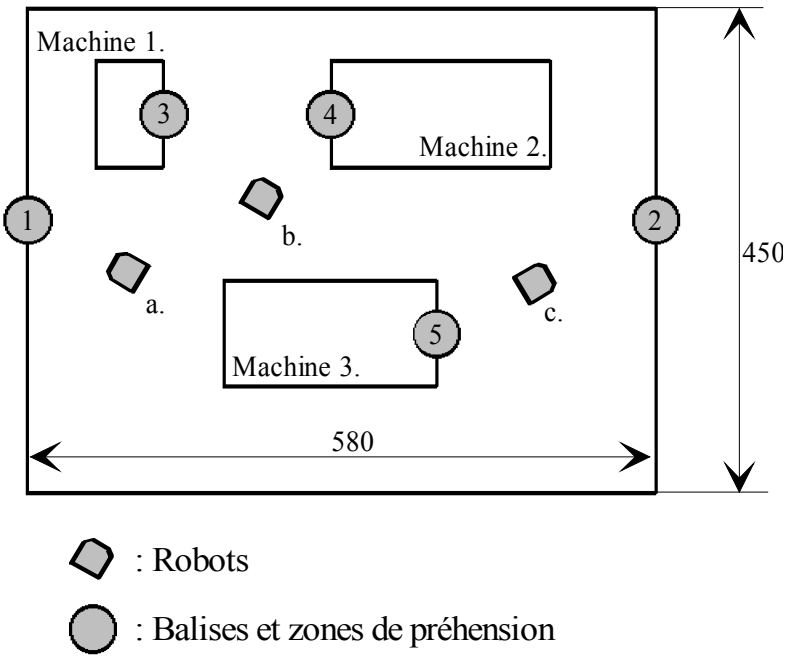


Figure II-13 : *L'atelier utilisé pour la simulation*

Les balises 1 et 2 correspondent respectivement au point d'entrée des pièces dans l'atelier et au point de sortie. Les balises 3, 4 et 5 correspondent aux points de chargement/déchargement des trois machines. Les unités de mesures sont arbitraires (il s'agit en fait de pixels lors de l'affichage).

Le système de transport proprement dit est constitué de un à treize robots. Le principe de guidage par balise nous autorise à concevoir des robots très simples, sans modèle interne de l'environnement : seules les perceptions directes de l'environnement sont utilisées dans le processus de décision. Les robot disposent de six « secteurs de perception » - avant (AV), avant-gauche (AG), avant-droit (AD), gauche (G), droit (D) et arrière (AR) - qui leur permettent de savoir dans quelle direction (c'est-à-dire dans quel secteur) se trouve la balise

recherchée et de détecter la présence d'obstacles à proximité immédiate (figure II-14). Un robot peut choisir son comportement parmi cinq actions différentes : « avancer » (A), « tourner à droite et avancer » (TD), « tourner à gauche et avancer » (TG), « tourner aléatoirement à droite ou à gauche et avancer » (TA) et « tourner fortement , aléatoirement à droite ou à gauche, et avancer » (T2A). Les angles de rotation sont déterminés aléatoirement, de 0,375 à 0,625 radians (21,5° à 35,8°) pour une rotation simple et de 0,75 à 1,25 radians (43° à 71,6°) pour une rotation « forte ». La vitesse des robots est constante, fixée à 2,5 pixels par unité de temps. En trajet direct il faut donc environ 230 UT à un robot pour traverser l'atelier (c'est à dire pour aller de la table d'entrée à la table de sortie).

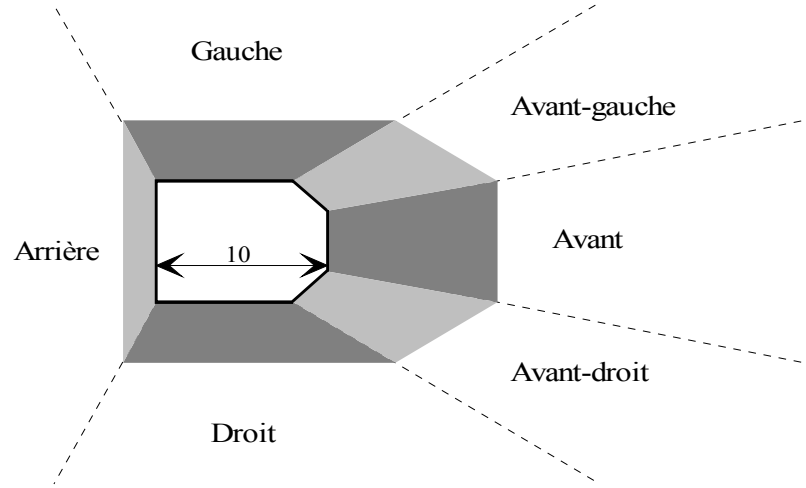


Figure II-14 : *Les six secteurs de perception d'un robot*

Les zones en grisé correspondent aux zones de détection d'obstacles.

La fonction de décision des robots est extrêmement simple. Elle est gouvernée par trois niveaux de règles qui sont :

- Evitement d'obstacle
- Recherche de la balise cible
- Choix de la balise cible

Le premier niveau est prioritaire sur les deux autres : en cas d'obstacle dans le secteur frontal ou dans les secteurs frontaux-latéraux le robot tourne pour éviter l'obstacle correspondant (tableau II-1). Les secteurs frontaux latéraux sont prioritaires sur le secteur frontal car ils permettent d'orienter la rotation.

Secteur de détection de l'obstacle	**AD**	**AG**	**AV**
Action	**TG**	**TD**	**T2A**

Tableau II-1 : *Premier niveau de règles (évitement d'obstacles)*

Le deuxième niveau de règles permet au robot d'atteindre la balise cible en fonction du secteur dans lequel elle est détectée (tableau II-2).

Secteur de détection de la balise	/	AV	AD	AG	D	G	AR
Action	**A**	**A**	**TD**	**TG**	**TD**	**TG**	**TA**

Tableau II-2 : *Deuxième niveau de règles (recherche de la balise cible)*

Dans le cas où plusieurs balises cibles sont détectées, le robot choisit selon une loi de priorité (la balise la plus frontale est prioritaire). En cas de priorités égales (symétries), l'action est déterminée aléatoirement.

Le troisième niveau de règles permet au robot de choisir sa « destination » en fonction de son état courant (chargé ou non) et de l'état des machines de l'atelier (libre ou non, pièce disponible, ...). Il s'agit donc de choisir, à travers le comportement de chacun des robots, une loi globale de gestion des matières transportées. Deux solutions principales sont envisageables la loi de requête (RFS[13]) et la loi d'échange :

- La loi RFS est la plus intuitive : lorsqu'une ressource quelconque (machine ou table d'échange) nécessite un chariot mobile (c'est-à-dire lorsqu'elle a une pièce à transporter), elle émet un signal RFS qui est pris en compte par les robots mobiles libres. Ceux-ci considèrent alors la balise correspondante comme balise cible et se dirigent - éventuellement - vers elle. Lorsqu'un chariot est chargé, il se dirige vers la balise « destination » de la pièce transportée. Cette loi de gestion autorise donc les robots à avoir plusieurs destinations possibles (d'où les règles d'arbitrage présentée en légende du tableau II-2). En outre, elle suppose que les ressources disposent de moyens de communication pour échanger les signaux RFS.

- La loi d'échange, par contre, ne demande aucun moyen de communication inter-ressources et supprime tous les conflits de destination (sauf si une pièce peut avoir plusieurs destinations possibles). Selon cette loi, les robots ne peuvent se déplacer que lorsqu'ils sont chargés et un robot vide doit immédiatement aller chercher une nouvelle pièce à la table d'entrée. Les machines ne peuvent donc que procéder à des échanges de pièces.

Afin de choisir entre ces deux règles de gestion et d'étudier le comportement du système de transport, nous avons effectué des séries de simulation de 20 000 UT chacune. Les performances du système de transport sont estimées par le nombre de pièces produites par l'atelier (qui est directement fonction du rapport « temps d'usinage/temps mort » pour les trois machines). Le comportement du système de transport (et de chacune des machines) étant en grande partie aléatoire, les résultats sont des moyennes calculées à partir de dix simulations.

Dans un premier temps, nous avons effectué une comparaison entre la loi RFS et la loi d'échange. Pour cela un système de transport type constitué de 5 robots a été utilisé dans l'atelier pour chacune des deux règles et pour différents « Temps de Process Maximum » (TPM). Les résultats sont présentés dans le tableau II-3. Le nombre de chariots a été choisi à

[13] Request-For-Service. Cette loi de gestion est proposée dans [DOT 92]

partir des premiers essais effectués car il constitue une bonne moyenne entre un sous-dimensionnement du système de transport et la saturation physique de l'atelier provoquée par un trop grand nombre de robots (voir figure II-15).

TPM (UT)	**100**	**200**	**300**	**400**	**500**	**600**	**700**
Règle RFS	70.7 *6.4*	68.8 *4.0*	59.6 *3.2*	**57.3** ***3.5***	44.9 *5.7*	**43.6** ***7.5***	39.7 *7.2*
Règle d'échange	**84.4** ***3.9***	**71.5** ***4.7***	**62.2** ***4.4***	53.4 *6.0*	**48.6** ***7.6***	37.7 *5.6*	**40.6** ***7.2***

Tableau II-3 : *Comparaison entre la règle RFS et la règle d'échange*

Nombre moyen de pièces produites par l'atelier pour des temps de process moyens différents. Les valeurs en italique représentent les écarts types pour chaque campagne de 10 mesures.

Les résultats présentés dans le tableau II-3 attribuent de meilleures performances à la règle d'échange lorsque les temps de process sont faibles. En revanche les performances sont sensiblement identiques lorsque les temps de process croissent. Or l'abaissement des temps de process conduit à solliciter plus fréquemment le système de transport et souvent à le saturer (lorsque le temps de process augmente, ce sont les machines elles-mêmes qui créent un goulot d'étranglement et ralentissent la production). Il semble donc que la règle d'échange soit plus résistante lorsque le système de transport est fortement chargé, probablement parce qu'elle interdit les trajets à vide. Comme elle est, a priori, plus simple à mettre en oeuvre que la règle RFS (elle ne nécessite pas de communication inter-ressources), c'est cette règle que nous avons choisi pour les campagnes de simulation suivantes. On remarquera cependant que la règle d'échange conduit les robots à emprunter fréquemment (une fois par pièce produite) le chemin conduisant de la table de sortie à la table d'entrée. En conséquence cette règle conduit à saturer physiquement la partie centrale de l'atelier alors que la règle RFS permet à des robots vides d'effectuer, lorsque la partie centrale est saturée, des tâches de transport intermédiaire (par exemple sortie-machine3-sortie-...).

Dans un second temps, nous avons observé le comportement du système de transport lorsque le nombre de robots augmente. Pour cela le Temps de Process maximum a été fixé à 100 UT. Cela permet d'assurer que les machines ne constituent pas un goulot d'étranglement et que la productivité de l'atelier est directement fonction des performances du système de transport (la distance inter-machines la plus courte correspond à un temps de transport de 60 UT). Les résultats de ces campagnes de simulation sont présentés figure II-15.

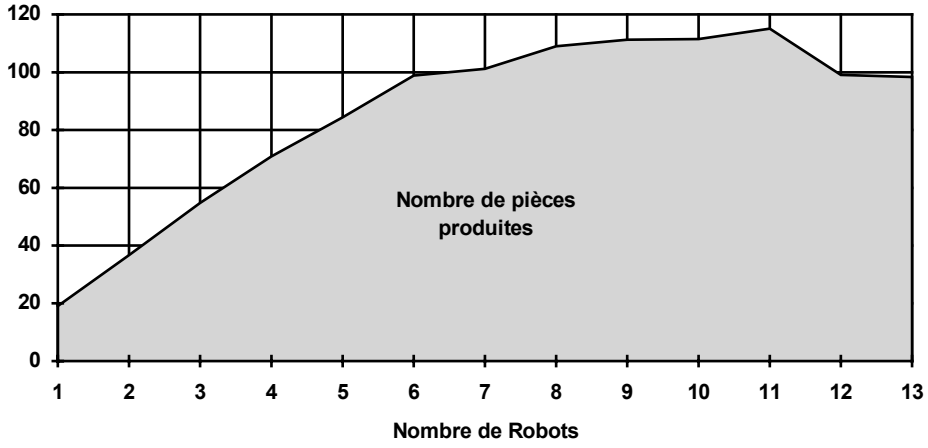


Figure II-15 : *Nombre de pièces produites par l'atelier avec 1 à 13 robots*

Les valeurs exactes sont présentées en annexe A1.

La figure II-15 confirme les observations visuelles, qualitatives, de l'atelier : le système de transport sature progressivement l'espace inter-machines lorsque le nombre de robots croît. Jusqu'à six robots, le nombre de chemins possibles (et l'espace situé entre les machine 2 et 3) permet de rajouter des robots de transport sans que leurs performances individuelles ne décroissent (figure II-16). Par contre, de 6 à 11 robots, les interactions deviennent plus nombreuses et on observe un net ralentissement des performances. Enfin, au delà de 11 robots, l'atelier est saturé en permanence et l'ajout de nouveaux robots provoque une forte baisse des performances du système de transport. Le phénomène de saturation est en grande partie dû à l'organisation physique de notre atelier : la majeure partie des surfaces est inutilisée car trop excentrée par rapport aux chemins reliant les balises entre elles. En effet, les phénomènes de saturation n'apparaissent pas (du moins pas aussi rapidement) lorsque les machines sont mieux réparties dans l'atelier et en particulier lorsque le trajet reliant les tables d'entrée et de sortie n'emprunte pas les mêmes zones que les trajets intermédiaires. Notre but étant, entre autres, de montrer l'adaptabilité du système de transport à des situations de transport complexes, nous avons délibérément choisi de travailler dans une configuration défavorable.

Lors d'une dernière campagne de simulation, nous avons cherché à estimer le rendement de notre architecture de transport. Pour cela nous avons comparé la productivité de l'atelier avec la productivité théorique d'un atelier idéal ayant les mêmes caractéristiques (temps de process, nombre moyen d'opérations) mais dans lequel les machines sont chargées à 100% (c'est-à-dire que l'on suppose que le système de transport est capable de répondre immédiatement à tous les besoins de matière). Le rendement a été calculé pour différentes configurations du système de transport (de 1 à 11 robots) et pour un Temps de Process Maximum variant de 100 à 700 UT (figure II-17). Les résultats obtenus montrent le bon rendement du système (jusqu'à plus de 80%) sous réserve que le système de transport soit correctement dimensionné (plus de 5 robots) et que le temps de transport moyen entre deux balises soit suffisamment faible en regard du temps de process maximum (dans notre atelier, le temps de transport moyen entre deux balises, en supposant que les robots empruntent le trajet direct, est de 125 UT environ).

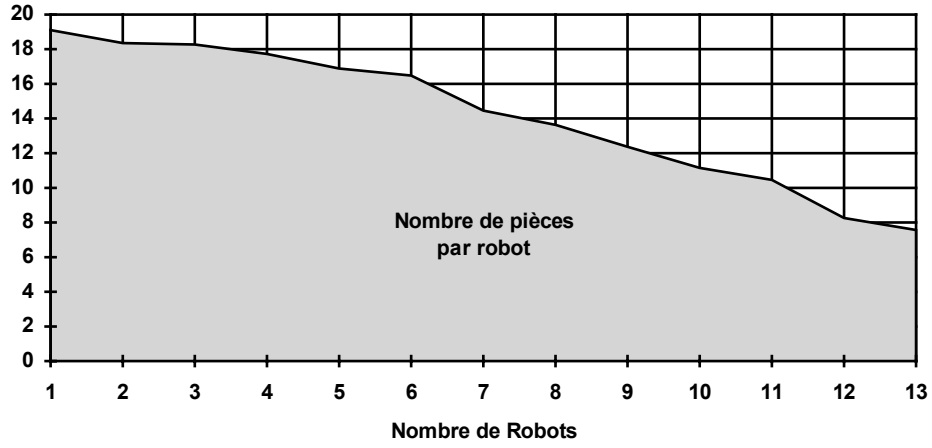


Figure II-16 : *Nombre moyen de pièces produites par robot.*
Les valeurs exactes sont présentées en annexe A1.

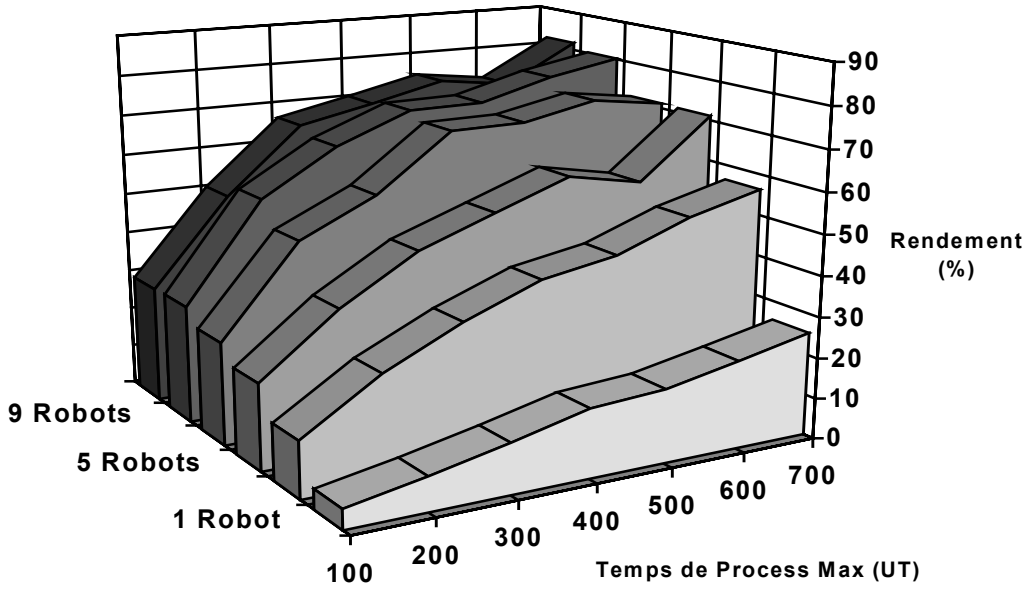


Figure II-17 : *Rendement du système de transport par rapport à un atelier idéal*
Les valeurs exactes (nombre de pièces produites dans chaque configuration) sont présentées en annexe A1.

II.3.5.4. Analyse et extensions

Bien qu'elle soit restée très simplifiée, cette première analyse d'un système de transport totalement décentralisé nous a permis de montrer l'intérêt de cette approche : malgré l'extrême simplicité des robots (pas de précision des mouvements, moyens de perception limités, fonction de décision très simple), le système de transport présente un rendement assez élevé. Malheureusement la comparaison des performances obtenues ici avec un système de transport classique (de type transfert libre ou chariot filoguidé) n'est pas possible (ou, du moins, serait faussée) car le mode de gestion adopté ici pour l'atelier (« ordonnancement »

totalement aléatoire) entraînerait probablement des performances désastreuses pour tout système en boucle fermée (ceux-ci doivent en effet être dimensionnés de façon très précise et l'agencement des machines autour de l'anneau influe fortement sur les performances du système de transport).

Les bonnes performances affichées par notre système de transport sont toutefois à relativiser : s'il permet de charger correctement les machines, il ne permet pas, en revanche, de garantir l'ordre d'arrivée des pièces sur les machines. Ce système n'est donc pas adapté à tous les types de production et d'opérations. Les opérations d'assemblage, par exemple, qui imposent la présence simultanée, sur une même machine, d'éléments différents, risquent d'être difficiles à gérer avec un système de transport distribué. De même, ce système se justifie peu pour les productions en très grandes séries pour lesquelles les temps de conception du système de production deviennent « négligeables » face au moindre gain de productivité. Par contre, dans un contexte de production réactive (petite et moyenne série, évolution permanente des moyens de production), un système de transport multi-robots se justifie pleinement car il permet d'obtenir de bonnes performances à faible coût et en conservant une totale évolutivité. En outre, lorsque l'appareil de production évolue, tous les éléments du système de transport sont directement réutilisables dans le nouveau système et il n'est pas utile d'arrêter l'ensemble de l'atelier pour agir sur une de ses composantes.

Par contre, si la simulation nous a permis d'illustrer les intérêts de la décentralisation des processus de décision, elle a aussi mis en avant certaines insuffisances propres aux robots utilisés. En plus des phénomènes de saturation apparaissant lorsque le nombre de robots dépasse un certain seuil, l'observation qualitative du fonctionnement de l'atelier nous a montré que le comportement des robots demande à être adapté pour éviter des conflits provoquant des retards ou pour optimiser certains déplacements élémentaires (contournement de machines, couloirs de passage, « sens uniques », ...). En outre des moyens de communications inter-robots permettraient probablement d'éviter la plupart des conflits entre les ressources. C'est pourquoi nous proposons d'étendre cette architecture dans deux directions principales. Dans un premier temps nous chercherons à optimiser le comportement des robots par l'utilisation des architectures réactives et par l'apprentissage des caractéristiques de leur environnement (d'où l'utilisation du neurocontrôle). Ainsi, on peut supposer que l'apprentissage de séquences motrices « standards » (évitement de robots arrivant en sens opposé, passage des « coins » des machines, trajets les plus courants, ...) permettrait d'améliorer les performances individuelles de chacun des robots et donc de garantir un temps de transport minimum. Dans un second temps nous chercherons à améliorer les performance du système de transport en autorisant les robots à échanger des informations. Nous envisageons, en particulier, d'associer à chaque robot une balise capable de ré-émettre les signaux caractéristiques des balises reçues. Nous espérons ainsi faciliter l'apparition de « trains de robots » et limiter ainsi les conflits entre robots circulant dans des directions différentes. De plus cette architecture permettrait, par la création de balises intermédiaires (les robots), d'autoriser l'existence de zones d'ombre depuis lesquelles une ou plusieurs balises seraient invisibles (figure II-18).

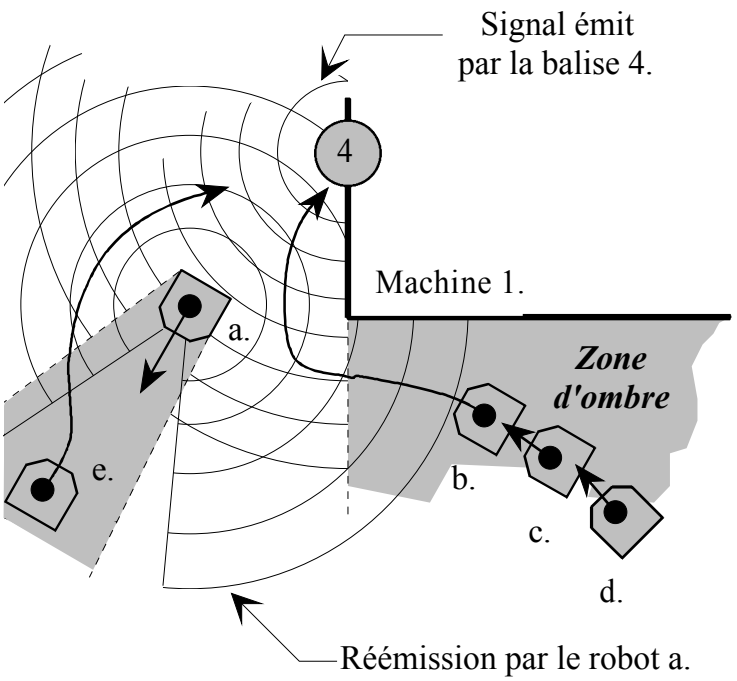


Figure II-18 : *Coopération inter-robots par ré-émission de balises.*

Chaque robot contient une balise qui lui permet de ré-émettre les signaux perçus (la ré-émission est effectuée dans des directions privilégiées. Dans les exemples proposés ici, il n'y a pas de ré-émission dans la zone frontale pour ne pas créer d'attirance directe d'un robot sur un autre).

- *Le robot (b) est situé dans une zone d'ombre de sa balise cible (balise 4) mais le robot (a) relaie la balise 4 et lui permet de sortir de la zone d'ombre et d'éviter les obstacles qui la créent.*
- *Les robots (c) et (d) se dirigent vers la même balise que (b). Celui-ci, en ré-émettant le signal détecté, les attire à lui ce qui permet la création de « trains de robots » et limite les conflits dans l'atelier.*
- *Le robot (e) est situé dans la zone d'ombre du robot (a). Il ne perçoit donc plus le signal de la balise et modifie sa trajectoire ce qui lui permet d'éviter d'entrer en conflit avec (a).*

Dans la suite de ce travail, nous nous intéresserons plus particulièrement à l'optimisation des contrôleurs locaux. Dans ce contexte, cette première étude d'un système de transport totalement décentralisé nous permet d'établir les premières bases du cahier des charges de ces contrôleurs.

La simulation nous montre que ces contrôleurs doivent disposer de règles comportementales réparties en couches successives. Un premier niveau de règles comportementales doit permettre aux ressources de répondre, de façon sub-optimale mais robuste, aux consignes qui leurs sont « imposées » (ou pour lesquelles elles ont été conçues). Ce niveau autorise l'atelier à conserver un régime de fonctionnement minimum, même en cas de dégradation partielle de son état interne ou de son environnement. Un deuxième niveau de règle, spécifique, doit se substituer au premier pour améliorer le rendement de la ressource lorsque son environnement le permet (c'est-à-dire lorsqu'elle n'est pas soumise à une perturbation). Ce deuxième niveau comprend des règles plus complexes (comportant en particulier des séquences d'actions alors que le premier niveau comprenait essentiellement des

actions réflexes) applicables dans des situations spécifiques. Il peut être partiellement défini manuellement mais pour que la ressource puisse disposer d'une autonomie maximale, elle devra être capable de développer automatiquement, c'est-à-dire par l'apprentissage, ses propres règles. C'est à cette condition que, lors d'une évolution du système de production, l'atelier va pouvoir retrouver automatiquement un fonctionnement optimal après un court temps de latence pendant lequel le premier niveau de règles autorise un fonctionnement sub-optimal.

Les contrôleurs locaux devront donc disposer simultanément de ces différents niveaux de règles comportementales gérées, au moins partiellement, par des mécanismes d'apprentissage et d'adaptation. En outre, la remise en cause des règles spécifiques doit elle-même être effectuée par l'algorithme d'apprentissage : la modification de l'environnement se traduit par des variations dans la boucle perception/action mais ces variations n'affectent pas obligatoirement l'ensemble des règles spécifiques. Ainsi, dans le cas des chariots mobiles dans une application de transitique, la fermeture d'un chemin spécifique (par exemple par le déplacement d'une machine) ne doit pas entraîner la suppression de l'ensemble des chemins appris par les robots. Les différentes règles de décision apprises par les contrôleurs devront donc pouvoir être modifiées indépendamment les unes des autres.

II.4. Conclusion

Dans ce premier chapitre, nous avons présenté le contexte générale de notre étude, à savoir les systèmes de productions modernes et les contraintes qui s'exercent sur eux. Nous avons en particulier montré comment, depuis une dizaine d'années, les délais de mise sur le marché influent sur la compétitivité des entreprises et nous avons présenté les principaux outils développés pour les réduire.

Au cours de cette étude préliminaire, nous avons particulièrement porté notre attention sur les capacités de flexibilité et de réactivité intrinsèques des différentes architectures de pilotage d'atelier. Face aux rigidités structurelles inhérentes aux architectures classiques, nous nous sommes tournés vers des modèles d'architectures totalement décentralisées au sein desquelles chacune des ressources de l'atelier possède une liberté décisionnelle totale par rapport à son environnement perçu. Une telle organisation de l'atelier autorise en effet une plus grande réactivité puisque les différentes décisions prises au sein de l'atelier le sont sur la foi d'information « sensorielles » (donc a priori conformes à la réalité physique de l'atelier) et non sur la base d'un processus de prévision/simulation qui introduit dans le processus de décision un décalage temporel au cours duquel les événements ne peuvent être pris en compte. En outre, la modification, à quelque niveau que ce soit, d'un élément de l'atelier va être immédiatement perçue par l'ensemble des ressources comprenant cet élément dans leur environnement et celles-ci vont pouvoir modifier leur comportement en conséquence. La flexibilité de l'atelier se trouve donc, elle aussi, maximisée par la décentralisation.

Dans un second temps nous avons cherché à formaliser les principes de l'architecture décentralisée. Nous avons ainsi été amenés à définir les contrôleurs locaux de chacune des ressources comme des systèmes sensori-moteurs en relation avec un environnement. Cette approche nous a permis de proposer, pour la réalisation de ses contrôleurs, deux techniques de l'intelligence artificielle : le contrôle réactif (qui propose des architectures sensori-motrices performantes mais figées) et les modèles neuromimétiques qui, grâce à leurs capacités

d'apprentissage, permettent de créer/adapter dynamiquement les boucles sensori-motrices lorsque l'environnement d'un contrôleur évolue.

Conscients des difficultés que pose l'intégration d'une architecture totalement décentralisée, nous proposons dans un premier temps d'en appliquer les principes à un sous-système de l'atelier. En raison de ses caractéristiques particulières, nous avons choisi le système de transport interne de l'atelier qui constitue un sous-système quasiment autonome mais dont l'influence, en termes de flexibilité et de réactivité, est particulièrement importante. En outre, le choix du système de transport permet d'utiliser directement les architectures du contrôle réactif et du neurocontrôle qui ont été largement appliquées aux problèmes de la robotique mobile. Une première approche, en simulation, utilisant des robots aux capacités cognitives très limitées illustre les principes généraux et montre que la décentralisation du système de transport ne porte pas préjudice à la productivité de l'atelier. En outre, la simulation nous a permis de préciser les besoins des contrôleurs locaux, en particulier au niveau de l'apprentissage. C'est pourquoi les prochains chapitres de ce mémoire sont consacrés à la définition d'une architecture de contrôle adaptée aux besoins de la productique.

Avant de proposer une architecture réactive pour les contrôleurs locaux, nous allons, dans le prochain chapitre, développer les principes essentiels des architectures réactives ainsi que certains points particulièrement importants pour notre domaine d'application. Dans un second temps nous étudierons plus précisément les modèles neuromimétiques et leur application au contrôle de processus (chapitre IV) ce qui nous permettra de proposer une architecture de contrôleur unissant les deux approches (chapitres V et VI). La définition des architectures de contrôle nous conduira à orienter notre propos dans une optique plus « robotique » que « productique ». En effet, le point clef de notre architecture est la notion d'*autonomie* : avant d'être considérés comme des ressources appartenant à un système de production, il est particulièrement important que les ensembles contrôleurs/ressources soient considérés comme des entités immergées dans un environnement inconnu (au moins partiellement). Ces entités s'accordent donc parfaitement avec la notion de « robots » telle que nous l'avons définie en tête du présent chapitre, c'est-à-dire une machine répondant aux trois caractéristiques suivantes :

- C'est une machine physique modifiant *physiquement* son environnement pour atteindre un but.

- C'est une machine *versatile*, c'est-à-dire qu'elle peut exécuter une tâche donnée de plusieurs manières différentes.

- Elle est capable de s'adapter seule aux variations de l'environnement de telle sorte que la tâche soit correctement exécutée en dépit de ces variations.

(d'après [COI 93], chapitre 6, pages 97 et 98)

CHAPITRE III :

LE PILOTAGE REACTIF

La création d'architectures de pilotage décentralisées pour les systèmes de production demande avant tout que l'on dispose, au niveau de chacun des contrôleurs locaux, de fonctions de décision capables d'assurer l'autonomie des ressources. Le modèle de contrôleur que nous avons proposé au cours du chapitre précédent repose sur des capacités de réaction en fonction de ses perceptions. C'est pour cette raison que nous avons choisi d'utiliser, pour implémenter les contrôleurs, des modèles architecturaux issus du Contrôle réactif ou « Behavior-Based Control ».

Dans ce chapitre nous présentons les principes fondamentaux du contrôle réactif. Les principales architectures actuellement utilisées en robotique seront ensuite décrites plus en détail, après quoi nous exposerons les principaux liens qui unissent le contrôle réactif aux sociétés d'agents, aux mécanismes d'apprentissage et, bien sûr, à la productique.

Le Pilotage Réactif

III.1. Introduction

Depuis 1970, l'Intelligence Artificielle a fait un usage presque exclusif des modèles de raisonnement symbolique. Ceux-ci reposent sur l'hypothèse qu'un processus intelligent peut (doit) être obtenu par manipulation d'un modèle symbolique de l'univers, les différentes étapes de cette manipulation constituant le raisonnement. Cette approche a été utilisée avec succès dans la plupart des domaines intrinsèquement symboliques en particulier avec le développement des systèmes experts. Cependant, dans certains domaines, lorsque l'échange d'informations avec le monde réel est indispensable (par exemple en robotique mobile), les modèles symboliques n'ont donné que de piètres résultats. Pour ce type d'application, ils sont très fortement handicapés par leur rigidité, par une mauvaise robustesse et par la croissance rapide des temps de calcul avec l'augmentation de la taille des modèles (explosion combinatoire).

Vers le milieu des années quatre-vingts, sous l'impulsion de quelques auteurs américains, de nombreux scientifiques ont considéré que, en matière de robotique autonome, les modèles symboliques conduisent à une impasse. En effet, il est apparu irréaliste de concilier l'expansion des modèles symboliques avec les impératifs de taille et d'énergie de la robotique mobile. En quelques années, essentiellement depuis 1986, quelques petits groupes de recherche, rapidement relayés par des équipes du monde entier, ont développé de nouveaux concepts et de nouvelles méthodologies pour le contrôle de robots autonomes. Ils ont ainsi donné naissance au contrôle réactif[14].

Les nouvelles architectures de contrôle développées à partir de ces travaux sont essentiellement caractérisées par un couplage étroit entre la perception et l'action et par un traitement parallèle des différentes tâches. En outre, l'absence de référence à un modèle statique de l'environnement autorise une flexibilité et une adaptabilité maximum.

L'idée première, autour de laquelle s'articulent ces nouvelles structures, est l'idée de ***fonctionnalité émergente***. Les différentes fonctionnalités d'un agent autonome, quel qu'il soit, sont le produit de ses multiples interactions avec un environnement complexe. Dès lors, les spécifications d'un système ne sont explicites que par rapport à un contexte donné dans un environnement donné. Une conséquence directe du principe de « fonctionnalité émergente » est l'impossibilité, pour le concepteur du système autonome, de définir un algorithme de résolution (planification). Il doit au contraire concevoir une boucle d'interaction (entre le système et l'environnement) qui converge vers l'état souhaité.

---

[14] Dans de nombreux articles américains le contrôle réactif (« Reactive Control ») est désigné sous l'appellation « Behavior-Based Control ». Cette dénomination étant pratiquement intraduisible - contrôle comportemental (?) - nous ne l'utiliserons que très rarement dans la suite du texte.

La ***structure distribuée et parallèle*** est la seconde caractéristique de ces architectures. Les processus de décision sont divisés en modules « Tâches » travaillant en parallèle. Cette structure, constituée d'entités autonomes communiquant entre elles sans notion hiérarchique, permet de renforcer la robustesse et la rapidité des traitements. L'accent porte sur l'absence de modules de supervision, de coordination ou de modélisation, modules qui constitueraient des points faibles pour la robustesse de l'ensemble ; la coordination entre les différents modules doit émerger de leurs interactions avec l'environnement.

Compte tenu de sa courte existence, le contrôle réactif a déjà fourni à la robotique autonome un grand nombre d'applications spectaculaires. Il a permis de « révolutionner » la robotique mobile et, dans une certaine mesure, l'intelligence artificielle en montrant l'efficacité de robots « insectoïdes » de conception relativement simple. Cette approche novatrice de la robotique n'est pas encore implantée en milieu industriel. Pourtant, face aux besoins de flexibilité et de réactivité des ateliers de production moderne, elle apporte quelques éléments de réponse : les contrôleurs réactifs présentés ci-après sont autant d'étapes vers la gestion autonome des ressources et donc vers le développement d'une architecture de gestion décentralisée et auto-organisée. Dans ce chapitre nous avons avant tout cherché à fournir au lecteur un aperçu général de ces nouvelles techniques. C'est pourquoi nous nous sommes attachés à définir, à travers ses origines, les principes fondamentaux du contrôle réactif. Nous présenterons ensuite les architectures réactives les plus marquantes avant d'orienter plus délibérément notre propos vers les applications industrielles envisagées.

III.2. Origine

Les origines du contrôle réactif sont diverses. Le facteur déclenchant a probablement été les difficultés rencontrées par les méthodes classiques qui ont engendré une demande de nouvelles méthodologies. Cependant l'existence d'un besoin n'est pas une condition suffisante à l'apparition d'une nouvelle technologie. Les premiers théoriciens du contrôle réactif ont puisé leur inspiration dans les travaux des cybernéticiens et des roboticiens (dont la problématique était similaire) ainsi que dans les sciences du vivant : biologie et neurobiologie d'une part, éthologie et psychologie comportementale d'autre part.

III.2.1. Les travaux antérieurs

L'origine du contrôle réactif, comme celle de la plupart des techniques informatiques, remonte aux années quarante et à la seconde guerre mondiale avec les travaux de Von Neumann [NEU 48], Turing [TUR 50] et Shannon [SHA 50][15]. Cependant, les machines « intelligentes » définies alors sont totalement déconnectées du monde réel et manipulent des données symboliques (le jeu d'échec est souvent pris comme exemple d'application). C'est plutôt dans les travaux de Norbert Wiener [WIE 48] et dans la cybernétique qu'il faut chercher les origines de « l'intelligence comportementale ».

III.2.1.1. Cybernétique

Pendant près de vingt ans (du début des années cinquante à la fin des années soixante) la cybernétique a constitué une importante composante de l'intelligence artificielle, en liaison

---

[15] [PEL 95] propose une traduction française des principaux textes fondateurs des sciences cognitives

avec la théorie du contrôle et avec l'approche systémique. La problématique des cybernéticiens est l'étude mathématique des machines autonomes et des modèles animaux ou humains. Contrairement à l'Intelligence Artificielle « classique », le but premier n'est donc pas la conception fonctionnelle de machines intelligentes, mais leur analyse en termes d'entrée-sortie (en termes de flux selon l'analyse systémique). Il s'agit donc de l'analyse des systèmes (considérés comme des boîtes noires) à partir des caractéristiques mathématiques de leur comportement. De plus cette approche intègre l'aspect dynamique des systèmes étudiés grâce à la notion de rétroaction (Feed-back) qui permet de réguler les systèmes.

Face aux premiers résultats des modèles de raisonnement symbolique, la cybernétique est rapidement apparue obsolète et a été progressivement abandonnée au cours des années soixante. En fait, cette technique a été victime de deux facteurs limitatifs : d'une part, des outils de calcul insuffisants et d'autre part, l'absence de mécanismes de décomposition modulaire des systèmes qui interdit l'agrégation de comportements différents. Depuis une dizaine d'années l'approche cybernétique connaît un regain d'intérêt dû, entre autres, aux liens étroits qu'elle entretient avec l'approche systémique [DUR 94] et avec le développement d'outils d'analyse des systèmes complexes (voir, par exemple, [MOR 95]).

Les liens entre cybernétique et contrôle réactif sont étroits. Les deux approches ont en commun leurs buts et une partie de leurs méthodes (le contrôle réactif profite cependant des récents développements de l'informatique). Ainsi Grey Walter propose en 1953 un modèle de robot mobile (une « tortue ») disposant de capteurs de lumière et de contacteurs. Plus récemment, en 1984, Valentino Braitenberg [BRA 84] a publié une étude de « Psychologie Synthétique » dans laquelle il montre comment une « société artificielle » composée de véhicules simples - munis de capteurs de lumière reliés aux propulseurs via des « neurodes » [DEW 87] - peut présenter des comportements complexes décrits en termes anthropomorphiques par un observateur extérieur (agressivité, amour, fuite, ...).

III.2.1.2. ROAR[16]

Avec le robot **SHAKEY** (développé à la fin des années soixante au SRI [NIL 69]), le contrôle des systèmes autonomes passe du champ de la cybernétique à celui de l'informatique. Dès lors, et jusqu'au milieu des années quatre-vingts, la robotique intelligente a fait un usage presque exclusif des modèles symboliques. Ce paragraphe expose les tendances principales suivies durant ces vingt-cinq années. Pour plus de précision on pourra se rapporter à [COI 93] ou à [IYE 91] qui propose un large recueil d'articles.

Une écrasante majorité des robots mobiles développés au cours de cette période (par exemple CART, développé à Stanford par la même équipe que Shakey, ou Hilare conçu au LAAS de Toulouse [GIR 84][GIR 93]) font appel à des ressources informatiques non embarquées avec lesquelles ils communiquent soit par câble (auquel cas la source de puissance est, elle aussi, extérieure) soit par transmission radio. Ils peuvent ainsi disposer d'une puissance de calcul plus importante, ce qui autorise le traitement d'un volume de données important et en particulier d'images vidéo. Les quelques robots utilisant des calculateurs embarqués (même partiellement) sont beaucoup plus limités. Ainsi le robot Jason de l'université de Berkeley (Californie), dont l'électronique était en grande partie embarquée, ne disposait que de capteurs infrarouges et ultrasonores.

---

[16] Remote Operated Autonomous Robots (Robots Autonomes d'Intervention)

La caractéristique essentielle partagée par l'ensemble de ces systèmes est la simplification importante de leur environnement. La plupart d'entre eux évoluent dans un environnement statique simplifié afin de limiter la complexité des tâches de contrôle. Dans de nombreux cas, l'environnement de travail est exclusivement composé de prismes (éléments polygonaux verticaux) ou de cylindres.

Cette simplification est rendue nécessaire par l'utilisation d'un modèle interne de l'environnement, généralement bidimensionnel, basé sur des primitives géométriques (treillis réguliers ou irréguliers, diagrammes de Voronoï, mémorisation des « polygones-obstacles »). L'utilisation d'environnements « modèles » permet alors de limiter les erreurs de représentation et donc les risques d'incohérence entre le monde réel et le modèle interne. En outre, l'utilisation d'un modèle suppose qu'on dispose d'outils permettant sa mise à jour (typiquement des outils de vision artificielle). Or ces outils sont lents et limités à la reconnaissance d'objets simples ce qui interdit leur utilisation dans un environnement naturel, complexe et dynamique.

En fait, la plupart de ces travaux conduits entre 1970 et 1985 reposaient sur l'hypothèse implicite qu'ils pourraient constituer des bases de travail solides pour des systèmes plus complexes, évoluant en environnement naturel (donc dynamique). En pratique, de tels robots n'ont pas vu le jour (ou, du moins, ils n'utilisent pas ce type de processus de décision) faute d'algorithmes d'extraction de symboles utilisables en milieu naturel.

Dans le cadre du projet américain ALV (Autonomous Land Vehicle[17]), cette approche a été abandonnée faute de pouvoir construire le modèle tridimensionnel d'une route goudronnée à partir d'informations visuelles. Les deux projets ayant abouti dans le cadre de ce contrat (ALVIN et NAVLAB) utilisent donc directement les informations visuelles pour en déduire l'action à effectuer : la trajectoire est calculée par rapport aux coordonnées visuelles et non par rapport à un repère extérieur [THO 88a][TUR 88]. Ces deux systèmes ont obtenu des résultats concluants, parvenant à contrôler des véhicules totalement autonomes à des vitesses relativement importantes (18 km/h pour les premières versions du NAVLAB et jusqu'à 80 km/h pour ALVINN[18] qui utilise un réseau de neurones pour l'apprentissage de la conduite sur routes et autoroutes [POM 93]). Bien qu'ils n'en partagent pas les aspects théoriques, les travaux de Matthew A. Turk et Charles Thorpe [THO 88a][TUR 88] au sein du Martin Marietta Denver Aerospace et de l'Université Carnegie-Mellon peuvent être considérés comme les premières applications du contrôle réactif [BRO 91]. Cette approche du contrôle visuo-moteur est d'ailleurs très proche de la « *Gestalttheorie »* (la théorie de la forme) développée dans les années trente pour expliquer la perception visuelle chez l'homme [GUI 37][19].

La plupart des robots construits depuis la fin des années quatre-vingts, bien que conservant une structure principalement symbolique, utilisent des modules réactifs de plus bas niveau pour protéger le robot (réflexes innés) [PRA 91]. Les derniers modèles de la famille « HILARE » [GIR 93](et les dérivés, MARTHA et VAP, conçus respectivement pour la transitique et l'exploration martienne) ainsi qu'« AMBLER » (un robot arachnoïde à six pattes conçu pour l'exploration du sol martien [BAR 89a]) sont construits suivant ce principe. Les

---

[17] Véhicule Terrestre Autonome

[18] Malgré la similitude des appellations, on remarquera qu'ALVINN (modèle neuromimétique pour le pilotage du NAVLAB) est différent d'ALVIN, projet concurrent du NAVLAB !

[19] Il est intéressant de noter que les critères visuels de la *Gestalttheorie* ont été récemment confirmés par l'étude neurobiologique du système visuel [SIN 93].

conditions sont cependant très particulières : MARTHA travaille dans un environnement partiellement connu et aménagé (mais dynamique) tandis que VAP et Ambler évoluent dans un environnement statique. De plus, pour ces deux dernières applications, la vitesse n'est pas un critère fondamental : VAP doit explorer une distance d'environ cent kilomètres en 13 mois et Ambler est conçu pour parcourir environ un kilomètre par jour.

III.2.2. Une remise en cause des modèles symboliques

Le principe fondamental de l'Intelligence Artificielle classique est que l'intelligence est une conséquence de la manipulation complexe d'informations symboliques représentant l'environnement. Dans le cadre de la robotique intelligente le modèle symbolique constitue donc un intermédiaire quasi incontournable entre perception et action. Le modèle symbolique est relié au monde réel par des modules d'extraction de symboles et par des modules actionneurs gérant la motricité de l'ensemble. En pratique le modèle est souvent créé à partir d'observations préalables de l'environnement auquel cas la perception directe est utilisée pour valider, et éventuellement compléter, la base de connaissances existante (figure III-1).

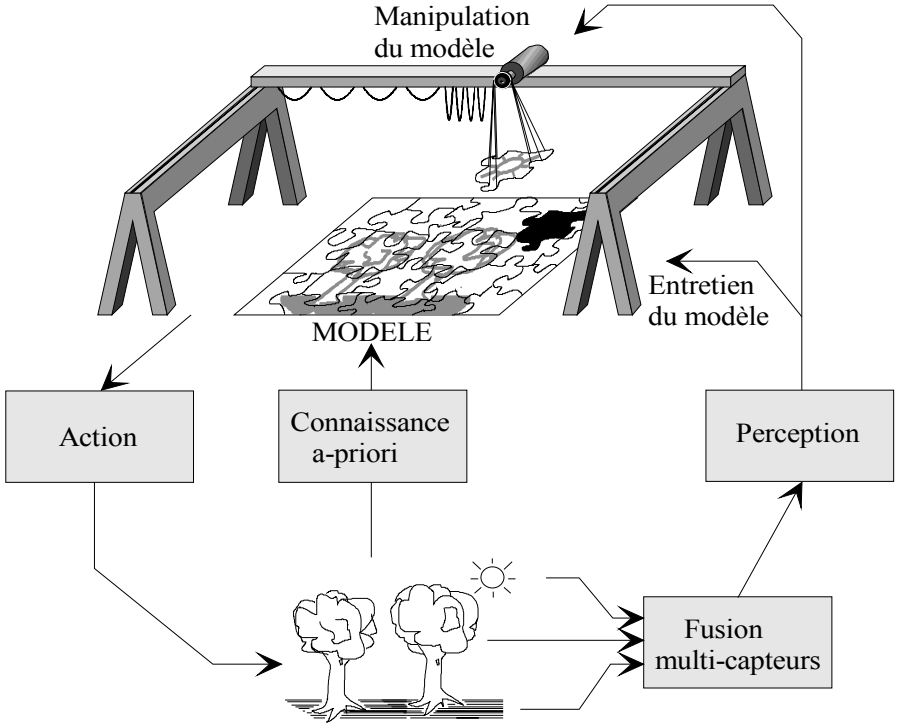


Figure III-1 : *L'organisation du contrôle symbolique*

Le modèle symbolique et les algorithmes de manipulation (moteur d'inférence, planification, ...) constituent la partie « intelligente » du système.

Les difficultés rencontrées par cette architecture ne sont pas directement dues à la manipulation du modèle ; c'est en fait l'absence d'interface efficace entre l'environnement réel et le modèle interne qui provoque la remise en cause de l'approche symbolique. En effet, on est à l'heure actuelle incapable d'extraire d'un l'environnement complexe les informations nécessaires à l'entretien du modèle (et, a fortiori, à sa création). De plus, les simplifications propres à la modélisation interdisent l'exploitation directe des informations symboliques. La boucle « environnement-perception-décision-action » de la figure III-1 est donc coupée en deux faute de modules « perception » et « action » : les modèles décisionnels sont efficaces en

simulation ou en environnement simplifié (« environnement-modèle ») mais ils sont inutilisables lorsque le robot est plongé dans un milieu naturel inconnu.

III.2.3. Du raisonnement au comportement...

Le contrôle réactif ne s'est pas exclusivement construit sur la négation des architectures symboliques mais aussi sur l'exploitation artificielle de modèles biologiques du comportement et de l'intelligence. Avant de clore ce paragraphe sur les origines du contrôle réactif, nous allons donc évoquer rapidement cet aspect.

Chacun d'entre nous utilise ses propres introspections pour construire une « théorie » de l'intelligence et de la pensée. Ainsi, si on excepte les partisans du dualisme matière/pensée, chacun d'entre nous construit un modèle, généralement très simple, largement influencé par l'utilisation du langage et de ses composantes symboliques. Les méthodes classiques de l'intelligence artificielle font fortement appel à cette vision simplificatrice (fonctionnaliste) d'un cerveau manipulant des symboles de haut niveau.

Face à cette approche fonctionnaliste, on peut opposer l'ensemble des principes matérialistes et, en particulier, le matérialisme « identité » [DEL 94] selon lequel la conscience (et l'intelligence) est une propriété émergente de la complexité du système nerveux (qui est non réductible à un ensemble de règles) et de ses relations avec un environnement complexe. Selon une démarche proche de l'analyse systémique, l'intelligence ne peut être étudiée que par rapport à un contexte, à un environnement extérieur qui alimente la conscience en stimuli et lui permet de « s'exprimer » à travers ses actions. L'intelligence n'est plus alors une « simple » question de raisonnement (le « comment ») mais elle s'exprime au contraire à travers un ensemble de comportements (le « quoi ») produits par des interactions complexes au sein d'un ensemble de cellules nerveuses qui ne contiennent pas (ou du moins pas explicitement) le raisonnement conduisant à ce comportement.

Les principes fondamentaux du contrôle réactif sont basés sur ces nouvelles approches de la conscience : plutôt que de chercher à développer un *raisonnement* capable de coordonner les actions d'un robot dans un but précis, la conception d'un contrôleur réactif demande avant tout d'identifier les *comportements* que le robot devra suivre pour résoudre la tâche qui lui est impartie. Cette démarche est à l'origine des liens étroits qui rapprochent le contrôle réactif des sciences biologiques, plus particulièrement l'éthologie [BEE 90]. Dans une moindre mesure (en raison de sa complexité), l'étude des comportements humains a aussi été utilisée comme source d'inspiration. Les travaux de Jean Piaget, en particulier, parce qu'ils concernent le développement sensori-moteur de l'enfant et les transitions vers le stade cognitif, permettent, par analogie, de hiérarchiser les comportements suivant les étapes successives du développement de l'intelligence[20].

---

[20]Piaget distingue six stades essentiels dans le développement de l'enfant, caractérisés par l'apparition successive de nouvelles structures mentales sur la base (le substrat) des structures apparues antérieurement : 1° Le stade des réflexes, ou montages héréditaires, ainsi que des premières tendances instinctives (nutrition) et des premières émotions. 2° Le stade des premières habitudes motrices et des premières perceptions organisées, ainsi que des premiers sentiments différenciés. 3° Le stade de l'intelligence sensori-motrice ou pratique et des régulations affectives élémentaires. 4° Le stade de l'intelligence intuitive, des sentiments inter-individuels spontanés et des rapports sociaux de soumission à l'adulte. 5° Le stade des opérations intellectuelles concrètes et des sentiments sociaux de coopération. 6° Le stade des opérations intellectuelles abstraites, de la formation de la personnalité et de l'insertion affective et intellectuelle dans la société des adultes [PIA 64].

Outre leur utilisation pour la définition des comportements élémentaires, les modèles biologiques ont aussi été à l'origine de plusieurs méthodes d'implantation des comportements élémentaires dans un contrôleur. On peut citer, par exemple, les modèles neurobiologiques développés par Randall Beer ou Nicolas Franceschini (section V.2) ou les schémas moteurs utilisés dans l'architecture AuRA (section III.3.3). L'utilisation des modèles biologiques comme source d'inspiration n'est cependant pas exempte de risque : rien ne prouve, en effet, que les modèles biologiques soient optimisés. Il est au contraire fort probable que certaines composantes du système nerveux des animaux les plus évolués (y compris l'homo-sapiens-sapiens) soient des vestiges des animaux primitifs conservés par l'évolution (cerveau reptilien, bulbe rachidien, ...). De plus, les systèmes biologiques sont des organisations extrêmement complexes et il est difficile d'attribuer les fonctions cognitives à l'un ou l'autre des niveaux d'intégration (atomique, moléculaire, macromoléculaire, cellulaire, macro-cellulaire, ...). Il est beaucoup plus probable que tous ces niveaux collaborent ; or les représentations informatiques sont généralement limitées à un seul niveau d'organisation.

Cependant que les systèmes biologiques sont plus utilisés comme sources d'inspirations que comme modèles de contrôleurs. Les architectures réactives, privilégiant le résultat sur la structure, ont donc été adaptées selon les besoins et les moyens disponibles même si cela a dû les éloigner des modèles biologiques. Dans ce contexte même une théorie éthologique, psychologique ou neurobiologique erronée peut être utilisée si elle permet d'améliorer les performances des robots construits. C'est d'ailleurs le cas pour l'architecture AuRA qui repose sur des modèles comportementaux obsolètes [ARK 90c].

III.3. Les structures du contrôle réactif

Après une réflexion similaire à celle présentée ci-dessus, deux équipes de recherche américaines (animées par Rodney A. Brooks au Laboratoire d'Intelligence Artificielle du MIT [BRO 86] et par Ronald C. Arkin à l'université du Massachusetts [ARK 87]) créèrent parallèlement les premiers systèmes réactifs. La naissance de ce nouveau type d'architecture est consacrée en 1986 par la parution d'un article de R.A. Brooks dans lequel il développe la plupart des principes fondamentaux du « Behavior-Based-Control ». Ces principes ont été précisés au cours des années suivantes et, dès le début des années quatre-vingt-dix, plusieurs ouvrages collectifs ont contribué à leur diffusion ([MAE 90a]).

III.3.1. Définitions

Les différentes architectures réactives cherchent à éliminer les problèmes de temps de réponse et d'extraction de symboles en supprimant purement et simplement les modèles internes. Le contrôle réactif offre en remplacement des principes architecturaux destinés à remplacer l'intelligence symbolique par l'intelligence comportementale.

III.3.1.1. Contrôle réactif et intelligence

D'après Ron Arkin et Rod Grupen [ARK 93b],

« Le contrôle réactif est une technique permettant de relier étroitement perception et action, traditionnellement dédiée à la motricité, afin d'obtenir une réponse temps réel aux stimulations provenant d'un environnement dynamique et non-structuré.[21] *»*

Le contrôle réactif ne doit donc pas être confondu avec l'ordonnancement réactif ou la planification réactive qui appliquent les contraintes du temps réel sur une structure symbolique. Il ne doit pas non plus être identifié à la planification prévisionnelle[22] qui associe a priori une séquence d'actions à chacune des situations potentielles sans que l'évolution de l'environnement influence le déroulement de cette séquence (exécution aveugle).

Ne serait-ce que par comparaison avec les méthodes classiques, le contrôle réactif est indissociable de la notion d'intelligence. Face à l'approche anthropomorphique de l'intelligence (en particulier du test de Turing qui définit l'intelligence par comparaison avec la référence humaine[23] [TUR 50]), Rodney A. Brooks, l'un des principaux artisans du contrôle réactif, oppose deux arguments plus fonctionnels :

- Le développement de l'intelligence ne concerne qu'une petite fraction finale du processus d'évolution ayant conduit à l'apparition de l'homo-sapiens. On peut donc supposer que les opérations élémentaires de perception et de motricité dans un environnement dynamique sont un substrat indispensable à l'apparition de l'intelligence. Les animaux peu évolués nous donnent un exemple de ce substrat et peuvent donc servir d'exemples pour la construction de robots « pré-intelligents ».

- Peu nous importe que les robots soient « intelligents » s'ils se comportent de façon intelligente, c'est-à-dire si un observateur extérieur, ignorant totalement leurs règles de fonctionnement, les juge intelligents d'après l'observation de leur comportement lorqu'ils sont plongés dans leur environnement. Ce principe est généralement illustré par une parabole : le trajet (complexe) effectué par fourmi sur long d'une grève est le reflet de la complexité de l'environnement sur le comportement de la fourmi et non l'expression d'une réflexion intelligence.

Rodney Brooks propose donc une nouvelle approche de l'intelligence, considérant d'une part que le processus mental lui-même est moins important que ses conséquences visibles, d'autre part que celles-ci sont liées au milieu dans lequel baigne le système. L'intelligence est déterminée par la dynamique des interactions avec le monde [BRO 91].

---

[21] Reactive Control is a technique for tightly coupling perception and action, typically in the context of motor behaviors, to produce timely robotic response in dynamic and unstructured worlds. ([ARK 93a], p.7)

[22] Situated Action

[23] Un utilisateur communique, via un télétype, avec un ordinateur ou avec une tierce personne. L'ordinateur (l'algorithme) peut être considéré comme intelligent si, au bout d'un certain temps, il est impossible de faire la distinction entre l'homme et la machine.

La portée du test de Turing est fortement contestée : d'une part parce qu'il ne permet pas de juger des capacités de métacognition de la machine, d'autre part parce qu'il n'implique pas la compréhension de la tâche (voir l'expérience fictive dite « de la chambre chinoise » [DEL 94]). En outre, il est fortement réducteur en raison de son aspect anthropomorphique [DUB 90].

III.3.1.2. Emergence

Le concept d'émergence, intimement lié à la relation entre le sujet observé et son environnement, est probablement aussi vieux que la problématique scientifique mais il n'a été réellement pris en compte que depuis une trentaine d'année, en particulier grâce au développement de la systémique et des sciences de la complexité [MOR 95]. Il s'agit de déterminer si le comportement d'un système est le fruit de son organisation interne ou si, comme le suppose la thèse de l'émergence, il est issu des interactions avec son environnement. Cette question est particulièrement liée aux sciences de la vie puisqu'un système vivant interagit en permanence avec un environnement complexe. La thèse de l'émergence conduit alors à penser que la complexité du comportement est le reflet de la complexité de l'environnement. Un comportement « intelligent » peut alors être simplement dû à une parfaite adaptation à l'environnement, ce qui conduit, à travers le darwinisme, à voir dans la sélection naturelle et dans la génétique les fondements d'une intelligence non consciente parce qu'émergente.

Bien qu'elle soit souvent définie en termes de relation sujet/environnement (donc deux entités), l'émergence est, en pratique, fortement liée aux systèmes multi-agents et à « l'intelligence collective » [BON 94]. C'est en effet dans les systèmes distribués comptant un nombre important d'éléments (tels les colonies d'insectes sociaux) que le phénomène émergent prend toute son ampleur : l'ensemble de la colonie présente des fonctionnalités (approvisionnement du groupe, soins au couvain, construction d'un nid, ...) dont ne dispose aucun des agents locaux (qui, dans le cas des insectes sociaux, sont quasiment dépourvus de capacités cognitives). En effet, dans le cas d'une colonie, les phénomènes émergents propres aux interactions sujet/environnement (le sujet étant ici un agent particulier) sont renforcés par la richesse des interactions entre agents[24].

Le principe d'émergence a des conséquences directes dans le domaine de la robotique : si le comportement d'un agent autonome est une propriété émergente issue d'interactions intensives entre l'agent et un environnement dynamique complexe, alors les spécifications d'un système ne suffisent pas, à elles seules, à expliquer (ou spécifier) son comportement. Celui-ci dépend, pour une large part, des propriétés statiques et dynamiques de l'environnement [MAE 90b].

Le principe essentiel du contrôle réactif est l'exploitation des caractéristiques de l'environnement pour guider un agent autonome vers le but qui lui a été fixé. Là où l'intelligence artificielle classique résout un problème par une méthode algorithmique (c'est-à-dire en édictant la méthode conduisant au but tout en éliminant les perturbations dues à l'environnement), le concepteur d'un système réactif doit, au contraire, rechercher une « dynamique » reliant l'environnement et l'agent (les agents). C'est alors la convergence de cette dynamique (sous-entendu vers le but fixé) qui permettra au(x) robot(s) de remplir la tâche qui lui (leur) est impartie. Cependant, si la thèse émergente donne un cadre conceptuel à cette approche (dite approche montante), elle ne résout pas tous les problèmes pratiques. L'émergence est le produit de la confrontation d'entités complexes et de leur parfaite adéquation les unes aux autres. Or cette adéquation peut être, dans certains cas, difficile à obtenir car elle demande l'ajustement de nombreux paramètres. C'est pourquoi nous proposons

---

[24] La richesse des interactions s'expliquant, entre autre, par la communauté de lieu de vie, de but, de moyens, de sens, ... qui favorise les rencontres et la communication.

l'utilisation d'algorithmes d'apprentissage permettant l'acquisition ou, au moins, l'adaptation, des comportements afin de simplifier l'accès à l'émergence (voir section III.6).

III.3.1.3. « Mise en situation[25] »

Outre l'émergence, le contrôle réactif est basé sur un principe de « mise en situation » [BRO 86], dont l'idée fondamentale est exprimée par Rodney Brooks au travers d'une simple phrase :

Le monde est son propre, meilleur, modèle.[26]

L'utilisation du monde réel directement perçu comme source d'information (à la place d'une représentation interne) permet de faire disparaître totalement les problèmes liés à l'extraction de symboles et à la correspondance avec le monde virtuel : les perceptions étant directement utilisées pour décider des actions à entreprendre, le passage par une phase symbolique n'est plus nécessaire.

En fait, la mise en situation correspond à l'utilisation d'un repère égocentrique (« Robocentrique ») pour l'expression de l'environnement. Les différentes composantes du monde réel ne sont manipulées qu'en fonction de leurs interactions directes (spatiales ou fonctionnelles) avec le robot. Philip Agre illustre cette représentation fonctionnelle de l'environnement par l'exemple suivant, présenté dans le contexte du jeu « Pengi » mais suffisamment explicite pour être écarté de son contexte initial [AGR 90] :

« Si une guêpe me tourne autour (pour la troisième fois de la journée), je dois l'inclure, dans une éventuelle représentation du monde, comme « la guêpe qui me poursuit » et non comme « la troisième guêpe ». Peu m'importe que celle-ci soit la même que celle qui me poursuivait cinq minutes plus tôt, c'est l'interaction immédiate avec CETTE guêpe (celle que j'observe ici et maintenant) qui m'intéresse (en supposant que mon but ne soit pas d'effectuer une étude statistique sur les hyménoptères). »

La mise en situation permet de simplifier le traitement de l'information en remplaçant l'*intention* par l'*attention* qui peut être gérée grâce à des processus parallèles relativement simples (chacun d'entre eux étant en attente d'une seule classe d'événements). Le gain est alors double : d'une part, le temps de calcul (donc de réaction) peut être largement réduit par l'utilisation du parallélisme (les événements entrants sont directement exploités par les processus de traitement adaptés puisque les autres processus ne les « attendent » pas), d'autre part cette structure permet de coder chacun des traitements et de gérer, à un niveau supérieur, leurs interactions séparément au moyen de processus logiques relativement simples. On remarquera que cette architecture implique un découplage des différents stimuli. Chaque module doit, en effet, recevoir l'information sémantiquement signifiante pour la classe d'événements qui le concerne.

---

[25] Le terme anglais, « Situatedness », est intraduisible en français

[26] The world is its own, best, model. ([BRO 91], page 583)

III.3.1.4. « Mise en Oeuvre »

La « mise en oeuvre » (*Embodiment*) d'un processus décisionnel, qu'il soit réactif ou symbolique, est l'implantation de ce processus dans un robot réel (par opposition aux simulations) évoluant dans son environnement réel. Ce terme n'a de sens que dans un contexte de recherche : dans un environnement industriel, elle découle directement de l'utilisation du robot dans le système de production.

La « mise en oeuvre » est fortement liée au concept d'émergence : si l'intelligence comportementale est issue de l'interaction entre l'environnement et le système, alors, dans le cas d'un environnement simulé, l'intelligence du système est limitée par les possibilités du simulateur. De même, l'adaptation de l'environnement au robot - standardisation des formes, limitation de l'aspect dynamique ou stochastique - limite directement l'intelligence comportementale du système. La « mise en oeuvre » est donc l'étape la plus importante lors de la conception d'un robot mobile : elle seule permet la validation effective d'un algorithme. Dans un environnement dynamique, elle est le seul moyen de vérifier que le temps de réponse est suffisant et que la dynamique du système est en accord avec celle de l'environnement (simultanéités, succession d'événements, ...). Elle ne permet cependant pas de tester totalement un système en un temps fini puisque l'éventail des possibilités (l'espace d'états) est en général infini.

La mise en oeuvre contraint fortement le concepteur d'un système réactif car elle l'oblige à tenir compte de tous les détails de conception du système sans se limiter aux fonctions élevées. Or il apparaît souvent, lors de la réalisation physique d'un système robotique, que les fonctions élevées ne sont réellement utilisées qu'au cours d'une mince fraction du temps et que la majorité des problèmes provient des modules supposés peu complexes (essentiellement la perception et l'action).

III.3.2. L'architecture « Subsumption »

Historiquement l'architecture « Subsumption » est considérée comme le premier modèle de contrôle réactif. Initialement présentée en 1986, elle s'est ensuite rapidement répandue dans la communauté scientifique et plusieurs auteurs ont proposé des variantes de l'architecture initiale. Nous présentons ici les différents principes théoriques sur lesquels cette architecture est basée, après quoi nous détaillerons l'organisation générale d'un contrôleur « Subsumption ».

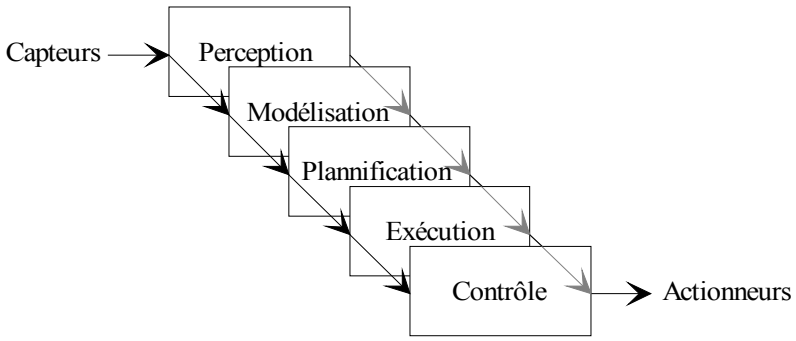
III.3.2.1. Principes

L'architecture « Subsumption », développée au MIT par Rodney A. Brooks entre 1984 et 1986, exploite les trois principes fondamentaux du contrôle réactif (émergence, mise en situation, mise en oeuvre) pour répondre à un cahier des charges affichant clairement sa différence avec les systèmes conçus antérieurement : les robots doivent être simples et peu chers tout en restant capables de se déplacer et d'effectuer un travail précis dans un environnement quelconque sans intervention humaine.

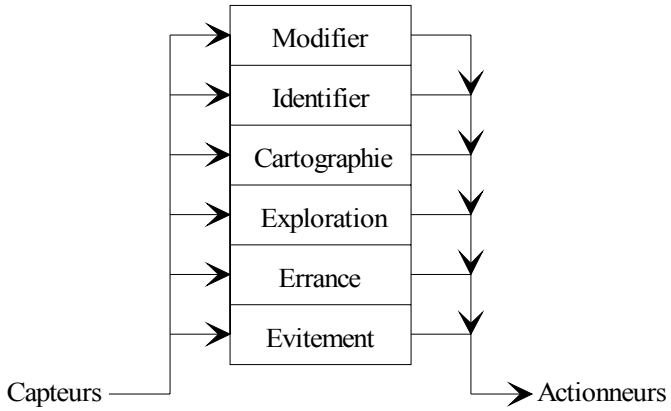
Pour répondre à ce cahier des charges, R.A. Brooks rejette la décomposition horizontale classique, basée sur des modules de traitement séquentiel de l'information (figure III-2a), et propose une décomposition verticale dans laquelle chaque module contrôle tous les éléments physiques du robot pour la réalisation d'une tâche particulière (figure III-2b). Les différents modules peuvent alors calculer en parallèle des solutions spécifiques dont une ou plusieurs

vont ensuite dicter le comportement du robot. Plutôt que d'organiser les modules en fonction de solutions algorithmiques choisies a priori, l'architecture est donc construite suivant les comportements externes souhaités et leur classement par « degré de compétence ».

Chaque niveau de compétence inclut, dans ses fondements, les niveaux inférieurs. L'organisation générale correspond donc à ajouter, à chaque niveau de compétence, des contraintes supplémentaires au substrat formé par les niveaux inférieurs. Chaque niveau *subsume*[27] donc les niveaux inférieurs, d'où le nom donné par Brooks à cette architecture.



(a) : Architecture classique



(b) : Architecture *Subsumption*

Figure III-2 : *L'architecture classique (a) et le modèle « Subsumption » (b)*

La décomposition verticale présente de nombreux avantages. En plus du gain en temps de réponse et en robustesse provoqué par l'utilisation du parallélisme, le principe de « Subsumption » permet de concevoir incrémentalement l'architecture de contrôle du robot. Il « suffit » d'ajouter une couche, c'est-à-dire un comportement, sur le substrat préalablement réalisé et testé pour augmenter les compétences de l'ensemble.

III.3.2.2. Organisation pratique

La construction incrémentale de l'architecture est basée sur la construction progressive, par couches successives, d'un réseau de « machines à états finis augmentée » (AFSM), c'est-à-dire de machines à états finis auxquelles on adjoint des registres d'entrée et une horloge interne (figure III-3) [BRO 89]. Chaque AFSM peut changer d'état suite à l'arrivée d'un message ou

[27] Subsumer : Considérer une entité comme faisant partie d'un tout (Le Robert de la langue française, 1994).

suite à un top de son horloge. Les sorties des machines sont redistribuées à travers le réseau ou, dans certains cas, directement bouclées à l'intérieur de l'AFSM (figure III-3, boucle (1)).

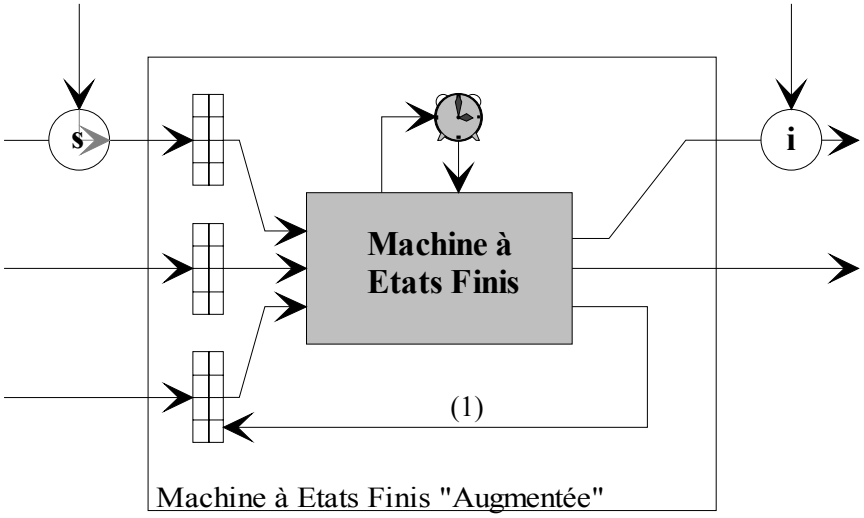


Figure III-3 : *Augmented Finite State Machine (AFSM)*

Une machine à états finis est étendue par l'adjonction de registres d'entrée et d'une horloge. Les différentes machines communiquent simplement par échange de messages binaires (via les registres) ou par suppression (s), inhibition (i) ou remplacement de messages. La sortie d'une machine à états finis peut être directement affectée à son propre registre d'entrée (1).

La construction d'un réseau d'AFSM met en oeuvre quatre mécanismes essentiels : le transfert, la suppression, l'inhibition et le remplacement de messages. Le mécanisme de construction incrémentale est donc supporté très simplement par le système puisqu'il suffit d'ajouter de nouvelles machines aux couches anciennes via l'un ou l'autre de ces mécanismes. Chacune des machines est gérée par sa propre horloge. Les messages sont donc échangés suivant un processus asynchrone. L'ajout de nouveaux modules n'entraîne donc aucune contrainte de synchronisation ou de « hand-shake ». La figure III-4 présente les huit premières couches du contrôleur d'un robot hexapode (GENGHIS) construit au MIT. Le robot correspondant mesure environ 35 cm pour un poids approximatif d'un kilogramme. Chacune des pattes (figure III-5) est contrôlée par deux servomoteurs (α-pos et β-pos) et chacun des axes génère en retour un signal codant la force appliquée (α-force et β-force). Le robot comprend en outre six capteurs infrarouges frontaux, deux inclinomètres et deux antennes qui lui permettent d'anticiper les obstacles.

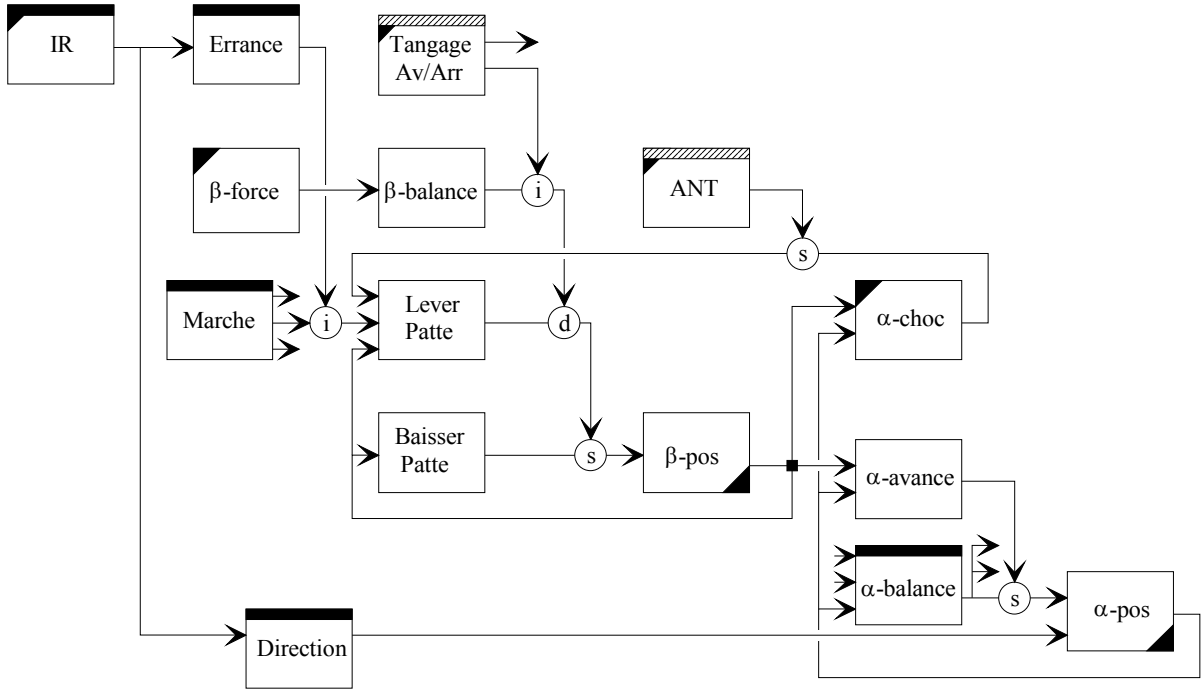


Figure III-4 : *Architecture de contrôle de GENGHIS.*

Le réseau complet comporte 57 AFSM. Les triangles noirs identifient les machines connectées aux capteurs (triangle noir en haut à gauche) ou aux actionneurs (triangle noir en bas à droites). Toutes les AFSM sans bande supérieure sont dupliquées six fois (une par patte) à l'exception des AFSM tangage et ANT (bande hachurée) qui sont dupliquées deux fois et des AFSM avec une bande noire qui sont uniques (synchronisation générale, modules α-balance, marche, errance, IR et direction). Les huit étapes de la construction incrémentale sont : « tenir debout » (modules α-pos et β-pos), « marche » (marche, lever patte, baisser patte, α-avance et α-balance), « marche assurée » (β-force et β-balance), « vitesse » (α-choc), « antennes » (ANT), « tangage » (tangage Av/Arr), « réagir au mouvement » (IR et Errance) et « suivre le mouvement » (Direction).

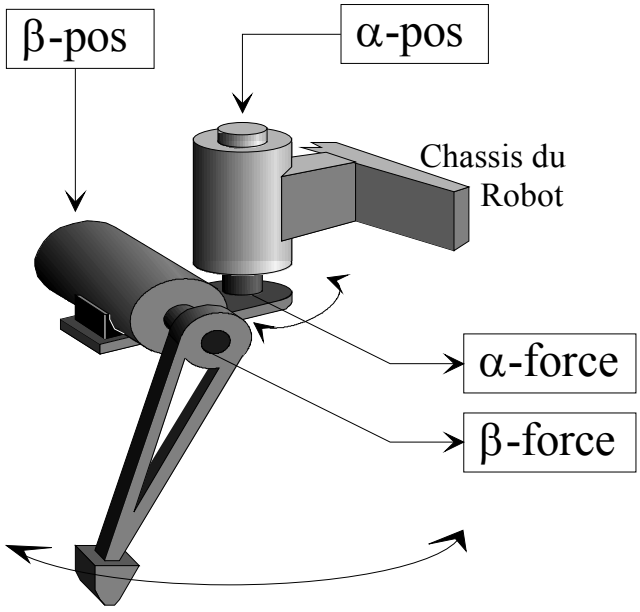


Figure III-5 : *Détail d'une patte de GENGHIS.*

III.3.2.3. Applications et variantes

Outre GENGHIS, l'architecture « Subsumption » peut se prévaloir de nombreuses implantations matérielles sur divers robots autonomes de tailles différentes (le plus petit, SQUIRT, représente un véritable exploit puisqu'il est entièrement autonome et ne pèse que 50 grammes). Le dernier robot construit au MIT par Rodney Brooks (en cours de développement depuis 1993) est COG, un humanoïde disposant de quatre caméras (deux pour la vision large en noir et blanc et deux pour la vision stéréoscopique) asservies pour le suivi oculaire.

Parallèlement aux travaux de R.A. Brooks, de nombreux chercheurs ont proposé des variantes de l'architecture « Subsumption ». Ainsi, au MIT, Pattie Maes a proposé un mécanisme d'arbitrage basé sur l'apprentissage [MAE 90c] et Maja Mataric utilise une architecture similaire pour un mécanisme orienté vers la coopération multi-robots [MAT 92]. En France, R. Zapata utilise une architecture similaire pour le contrôle de robots mobiles se déplaçant à grande vitesse (Robot SNAKE) [ZAP 92] et Bertrand Beaufrère utilise des règles floues pour réaliser les comportements élémentaires [BEA 94].

Bien qu'elle ait prouvé ses capacités pour le contrôle réactif, l'architecture « Subsumption » pose deux problèmes essentiels. L'architecture de contrôle étant définie a priori, il est difficile d'agir dynamiquement sur le ou les but(s) du robot. Il est donc nécessaire de prévoir l'ensemble des alternatives, non seulement en termes d'environnement, mais aussi en termes de but. De plus, même si des langages de développement ont été en partie définis, la construction incrémentale du graphe reste essentiellement empirique, soumise aux capacités - limitées - d'un concepteur humain. Il y a donc un fort besoin de mécanismes d'acquisition de comportements à partir de l'expérience, c'est-à-dire d'algorithmes d'apprentissage.

III.3.3. AuRA et les modèles vectoriels

En marge des travaux menés au MIT, Ronald C. Arkin a suivi une démarche similaire, sans pour autant rejeter systématiquement, comme l'a fait Brooks, l'utilisation de représentations du monde réel pour le contrôle réactif [ARK 87]. Cette architecture (AuRA[28]) a, elle aussi, été mise en oeuvre avec succès sur des robots mobiles autonomes.

III.3.3.1. Principe

Le modèle développé par Arkin à l'Université du Massachusetts puis au Georgia Hight-Tech Institute est basé sur les schémas moteurs définis comme étant « *une représentation mentale acquise par l'expérience et comprenant l'organisation des processus de perception et d'action en vue de répondre à des situations complexes caractérisées par un ensemble de stimuli* » (d'après [ARK 93b], page 70).

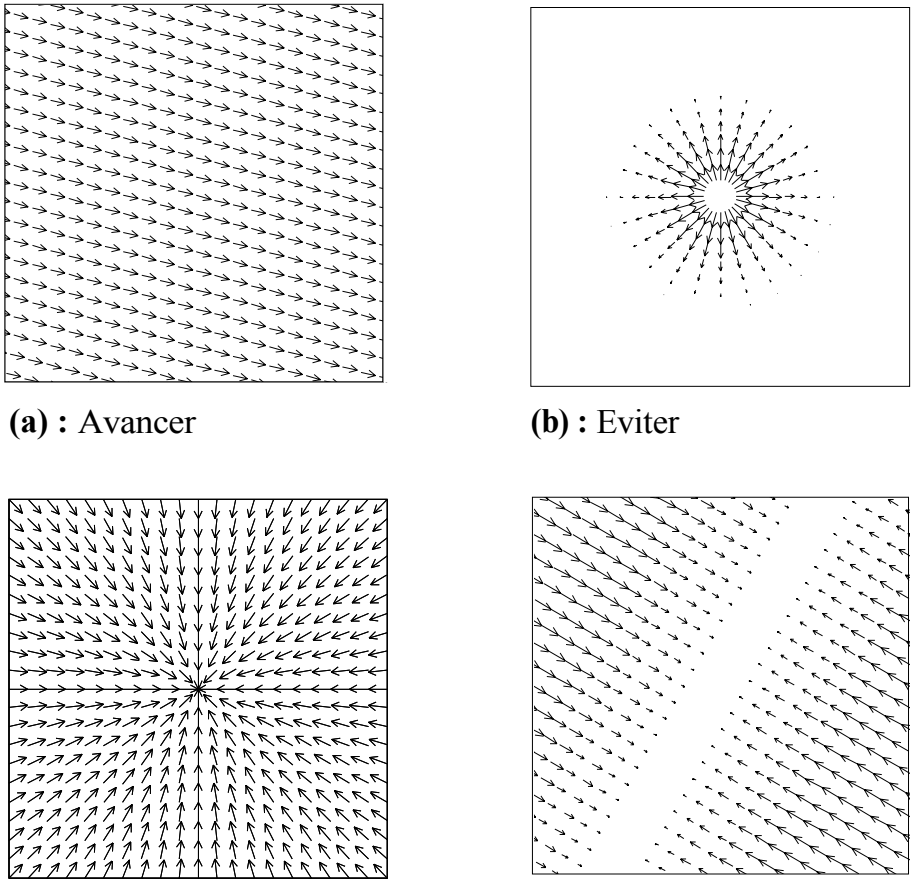
L'utilisation des schémas en robotique date du début des années quatre-vingts mais ils sont généralement restés limités à une petite partie du processus décisionnel sans remettre en question les principes de traitement séquentiel de l'information. AuRA, en revanche, est construit suivant des principes de contrôle sensori-moteurs : les schémas sont directement alimentés par les informations reçues des différents capteurs.

---

[28] **Au**tonomous **R**obot **A**rchitecture

III.3.3.2. L'architecture

Ron Arkin utilise des champs vectoriels pour la représentation de l'environnement du robot. Ils sont construits à partir de modèles standards tels que l'évitement d'obstacles ou la poursuite de cibles (figure III-6). Ces champs vectoriels sont directement exploitables pour la motricité et constituent un intermédiaire entre l'absence totale de représentation (prônée par R.A. Brooks) et les modèles symboliques [ARK 93a][ARK 93b].



(a) : Avancer **(b) :** Eviter

(c) : Atteindre **(d) :** Chemin

Figure III-6 : *Les quatre principaux schémas moteurs.*

a : *Avancer en ligne droite* b : *Eviter un obstacle*
c : *Atteindre la cible* d : *Rester sur le chemin*

Par rapport à la plupart des architectures réactives, les champs vectoriels permettent d'éviter les problèmes d'arbitrage entre les différents comportements : les différents schémas sont simplement additionnés puis normalisés (figure III-7). En outre, les minima locaux (qui constituent un problème crucial en robotique mobile) sont éliminés par l'ajout d'un comportement « bruit ». De plus, des informations symboliques de haut niveau (but, planification prévisionnelle, temps de parcours, obstacles connus...) peuvent intervenir dans le processus de décision par modification directe des schémas moteurs.

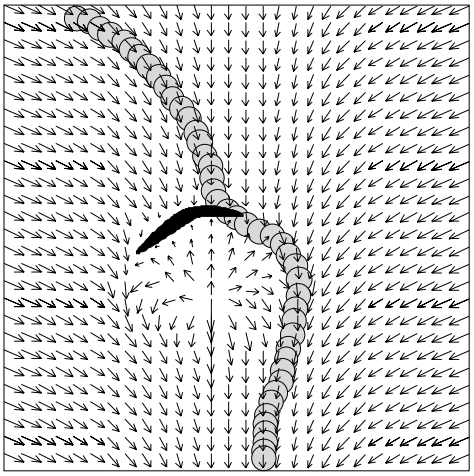


Figure III-7 : *Exemple de comportement issu de la combinaison des schémas « chemin », « avancer » et « évitement d'obstacle ». Les minima locaux (en noir) sont supprimés grâce à un quatrième schéma « bruit » (non représenté sur la figure).*

Un des avantages essentiels de l'approche de Ron Arkin est la portabilité totale des différents comportements. Chacun des modules générateurs de schémas peut être directement réutilisé sur un nouveau robot ou dans un nouvel environnement. L'ensemble des schémas constitue donc une librairie dans laquelle le concepteur d'un robot mobile autonome choisit les comportements les plus adaptés à l'environnement. AuRA peut donc être configuré en vue de l'utilisation dans un milieu spécifique. Ainsi l'application aux problèmes de transitique en milieu industriel a été étudiée par Ron Arkin [ARK 90b][ARK 90c]. Un schéma « accostage » spécifique a d'ailleurs dû être ajouté aux quatre schémas de base pour autoriser l'approche d'un quai de déchargement ou d'une machine outil (figure III-8).

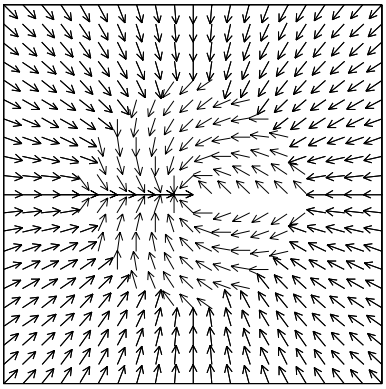


Figure III-8 : *Le schéma « accostage »*

Les schémas moteurs de AuRA peuvent être aisément étendus à des environnements plus complexes. L'utilisation en environnement fortement bruité ou incertain peut être envisagée en pondérant chacun des schémas utilisés suivant son degré de certitude. De même, l'utilisation en environnement à forte composante dynamique est autorisée par le caractère modulaire de l'architecture, ce qui permet de paralléliser les calculs et donc de diminuer le temps de réaction. En outre, Ron Arkin propose d'étendre son modèle aux espaces tridimensionnels en généralisant les comportements classiques et en y ajoutant une bibliothèque spécifique (monter, descendre, suivre une courbe de niveau, ...). Enfin, comme pour l'architecture « Subsumption », de nombreuses variantes ont été proposées. Ashwin Ram, en particulier, a proposé d'étendre le principe avec un mécanisme d'apprentissage [RAM 93].

III.3.4. D'autres modèles

Si les architectures du type « Subsumption » et les modèles vectoriels du même type qu'AuRA représentent la majeure partie des systèmes réactifs développés jusqu'à maintenant, de nombreux autres modèles ont été proposés dans la littérature. Nous présentons ci-après les plus significatifs.

III.3.4.1. « Neuro-Ethologie Calculatoire[29] »

L'analyse qui a conduit à la fois R.A. Brooks et R.C. Arkin à définir leurs systèmes respectifs est essentiellement issue de l'étude comportementale, macroscopique, de modèles vivants. Randall Beer et de nombreuses équipes de recherche proposent une approche basée sur l'étude microscopique des insectes. Cette approche, baptisée par R. Beer « Neuro-Ethologie Calculatoire », est issue d'une étroite collaboration entre roboticiens et neurobiologistes ce qui a permis de construire des architectures neuronales, généralement figées (c'est-à-dire sans apprentissage), utilisables pour le contrôle de robots mobiles. L'application de cette approche au contrôle réactif sera détaillée section V.2.1.1.

III.3.4.2. REX, Gapps et Ruler

L'un des problèmes majeurs des approches réactives, quelles qu'elles soient, est l'absence quasi-complète de méthodologie de conception pour les différents modules comportementaux ou pour les différents schémas moteurs. Chaque concepteur d'un système réactif doit donc faire face à une tâche de conception difficile pour laquelle aucun des outils de l'informatique classique n'est réellement adapté.

Leslie P. Kaelbling et Stanley J. Rosenschein, du groupe Teleos de Palo-Alto, ont proposé deux langages spécifiques pour le développement de modules comportementaux en architecture réactive. Le cadre général de leurs travaux est proche de l'architecture de Brooks : les agents définis dans cette étude sont construits sur une hiérarchie de compétence similaire à la décomposition par « Subsumption » (l'approche de L.P. Kaelbling inclut cependant la possibilité d'encapsuler des niveaux hiérarchiques secondaires dans un module). Par contre, la construction des modules comportementaux est sensiblement différente : L.P. Kaelbling sépare dans chacun des modules la perception et l'action. De plus, cette approche est basée sur un formalisme logique unissant information et sous-buts. Pour L.P. Kaelbling, il y a en effet une dualité évidente entre ces deux entités et, en particulier, entre « choix stratégique » et « oubli » : un choix stratégique est assimilable à l'oubli des autres alternatives [KAE 90].

A partir de ce formalisme les auteurs ont proposé successivement un langage de bas niveau pour la programmation des agents autonomes (REX, basé sur une machine LISP, qui permet la spécification de circuits de pilotage) puis, à un niveau supérieur, deux langages déclaratifs, basés sur REX, pour la définition des modules sensoriels (RULER) et décisionnels (GAPPS, défini à partir de la dualité information/but). Cette architecture a été testée, au centre de Palo-Alto, sur un robot mobile : FLAKEY.

---

[29] Computational Neuro-Ethology

III.3.4.3. « Internalized Plans »

Dans le cadre du projet DARPA ALV (qui a conduit à la création du NAVLAB), David Payton, du « Hughes AI Center », propose une architecture intéressante car elle conjugue la structure en couches des modèles « Subsumption » (ainsi que la construction incrémentale) et la représentation vectorielle de l'environnement.

Le principe de cette architecture est relativement simple : plutôt que de choisir entre les deux écoles (décision/action dictées soit par un modèle soit directement par l'environnement), D. Payton constate la complémentarité des deux approches et propose d'intégrer, dans le modèle de Brooks, des entrées sensorielles fictives appelées « Internalized Plans » [PAY 90]. Celles-ci sont basées sur des données symboliques, par exemple l'énergie et le temps disponible, les contraintes imposées par le concepteur, la connaissance partielle du terrain ou des représentations vectorielles de l'environnement (directement utilisables par les modules comportementaux).

Les intérêts de cette approche sont nombreux : d'une part, elle comble une importante lacune des architectures « Subsumption » (en offrant la possibilité de reconfigurer dynamiquement une partie des comportements en les reliant à des valeurs « paramètres »), d'autre part, elle permet d'accroître la complexité du modèle d'Arkin en ajoutant au graphe vectoriel des données non quantifiables géographiquement (buts non géographiques, données temporelles, ...). Par contre, bien qu'elle apporte un surcroît de puissance à l'architecture « Subsumption », cette approche ne résout aucunement les problèmes de conception inhérents à ce modèle. Au contraire, en multipliant les entrées et en obligeant chaque module à choisir entre perception et modèle(s), il y a lieu de craindre qu'elle n'alourdisse encore la phase de conception.

III.3.4.4. Systèmes hybrides

Malgré les positions apparemment très tranchées qu'adoptent certains auteurs en faveur du modèle symbolique ou du modèle réactif, de nombreux travaux utilisent conjointement les deux approches en essayant de préserver les intérêts principaux de chacune d'elles. Il est même probablement plus juste de considérer qu'à l'exception des quelques travaux purement réactifs (dont les travaux de R.A. Brooks), la grande majorité des robots mobiles développés actuellement sont des systèmes hybrides. Leur construction est motivée par deux constats essentiels :

- La difficulté éprouvée par les modèles réactifs à gérer des buts alloués dynamiquement (d'où un manque de flexibilité intrinsèque) ou des concepts non directement perceptibles (en particulier une connaissance a priori de l'environnement) pousse ce type de système à intégrer des niveaux symboliques.

- Le besoin d'interaction directe avec l'environnement et les problèmes de sécurité ont poussé les systèmes symboliques à s'étendre vers les niveaux inférieurs (création de boucles réflexes) [PRA 91].

Les différences profondes entre contrôle réactif (subsymbolique) et contrôle symbolique sont donc avant tout une question d'école (programmation logique/programmation comportementale) reposant en grande partie sur les fondements cognitifs décrits en début de chapitre. La mise en oeuvre de l'une ou l'autre de ces approches (c'est-à-dire, le plus souvent,

l'orientation générale d'une architecture hybride) est ensuite un choix fonction des circonstances propres à chaque application.

Les modèles hybrides sont généralement constitués d'une entité de planification, généralement hiérarchique et symbolique, capable d'entretenir et d'exploiter le contexte global de la « mission » (en intégrant, entre autres, des informations explicites fournies par un éventuel opérateur humain) et d'un contrôleur réactif (subsymbolique) utilisant parallèlement les stimuli sensoriels et les informations fournies par le niveau symbolique pour piloter le robot en temps réel et de la façon la plus robuste possible.

D'après Ron Arkin, les modèles symboliques hiérarchiques et le contrôle réactif sont totalement compatibles et doivent se partager respectivement la formulation du plan et son exécution (il met cependant en garde contre une hybridation trop rapide qui ne ferait qu'exacerber les aspects négatifs de chacune des deux approches). Selon [ARK 93a], les bases neurobiologiques et psychologiques de systèmes mixtes sont évidentes, bien que probablement très simplificatrices. Il inclut donc l'approche par schémas moteurs dans un modèle plus ambitieux : l'architecture « AuRA » comprend, outre le pilote réactif décrit section III.3.2, un « navigateur » et un superviseur chargé d'interpréter les ordres émanant d'un opérateur et d'assurer l'homéostasie du robot [ARK 93b].

III.4. Des agents autonomes aux sociétés multi-agents

La question de l'efficacité des robots construits suivant la base de comportements simples reste souvent posée : est-il possible d'assurer, par exemple, que le but sera atteint en se basant sur un robot unique ? La plupart des recherches menées sur le « Behavior-Based-Control » ont porté sur le comportement d'un unique robot capable d'évoluer dans un environnement dynamique complexe. A partir de la fin des années quatre-vingts (c'est-à-dire au moment où le contrôle réactif a commencé à s'ancrer plus fortement dans la communauté des roboticiens), de nombreux travaux se sont orientés vers des architectures multi-robots homogènes ou hétérogènes, en liaison avec l'éthologie et l'étude des essaims d'insectes sociaux [BON 94]. Les buts poursuivis par ces travaux sont multiples : d'une part, un intérêt théorique indéniable (dans une société de robots les propriétés émergentes peuvent être considérablement accrues par les multiples interactions entre robots), d'autre part, étant donné le « faible » coût de ce type de robots, la construction d'un essaim[30] semble, a priori, offrir des capacités de travail décuplées pour un coût qui reste inférieur à celui de systèmes plus complexes (en particulier pour des applications de surveillance, d'exploration, de récolte, de nettoyage, ...). Jusqu'à maintenant, ces travaux ont essentiellement fait l'objet de simulations sur ordinateurs mais de plus en plus de système multi-agents « réels » comptant de quelques unités à quelques dizaines de robots sont développés et étudiés, ce qui assure la pérennité des résultats obtenus.

En fait, cet engouement est issu des capacités étonnantes observées chez les insectes vivant en colonies (en particulier les fourmis, les termites et les abeilles). La confrontation de milliers d'entités au comportement très simple permet en effet à la colonie de créer des superstructures complexes (et souvent répétitives) et de les entretenir sans que chacune des entités soit soumise à un superviseur quelconque : la coordination des actions de l'ensemble du

[30] Dans la littérature le terme consacré est un « Swarm ».

groupe est essentiellement due aux interactions des différents membres de la colonie entre eux et avec leur environnement commun (stigmergie[31]).

Schématiquement, la stigmergie est provoquée par un ensemble d'interactions entraînant, dans un milieu initialement homogène (par rapport à une des variables d'état), l'apparition de perturbations qui, en s'amplifiant par un phénomène de feed-back positif, vont provoquer l'apparition d'hétérogénéïtés (concentration de phéromones chez les fourmis, développement de piliers chez les termites, ...). Celles-ci ont alors une influence sur le comportement des agents, ce qui provoque l'apparition de comportements spécifiques. La stigmergie n'est cependant pas le seul mode de fonctionnement des systèmes multi-agents biologiques et on observe dans la nature de nombreux comportements collectifs basés sur des relations diverses entre les agents et avec l'environnement : des sociétés cellulaires [BON 94] aux relations sociales de l'homo-sapiens-sapiens [SCH 95], la palette des comportements collectifs est vaste et il appartient aux sciences de l'artificiel, en coopération avec les sciences de la vie, d'exploiter cette source d'information.

La construction d'un essaim de robots pose cependant de nombreuses questions dont la principale est « comment contrôler plusieurs robots sans pour autant leur faire perdre leur autonomie? » [NUM 92]. D'autres questions d'importance concernent la communication inter-robots (communication explicite par la mise en place d'un langage inter-robots [YAN 92][ARK 93a] ou communication implicite par modification d'un environnement commun [DOR 96]), le but poursuivi (est-ce un but unique pour lequel les robots doivent réellement coopérer ou peut-il être divisé en sous-buts, chacun d'eux étant alloué dynamiquement aux différents robots, par exemple, le transport d'un ensemble - un sac de billes - ou de chacune des parties - des billes - d'un point à un autre). Le degré de communication et de coopération nécessaire entre les agents est en effet totalement différent dans le premier cas (la coopération est souvent indispensable [KUB 92]) et dans le second (la coopération n'est pas nécessaire mais elle permet généralement d'améliorer les performances [BES 95][DRO 92][PAR 92a]). L'étude des différentes formes de coopération (qui peuvent être, tout comme la communication, explicites ou implicites) et des comportements permettant leur apparition, constitue d'ailleurs un des enjeux actuels des systèmes multi-agents [FER 94].

Ronald Arkin et David Hobbs ont proposé un modèle de société multi-agents basé sur l'architecture AuRA [ARK 92][ARK 93a]. Au cours de cette étude, ils ont montré l'intérêt des langages de communication inter-agents. Ceux-ci permettent, en effet, de favoriser la coopération entre les robots. Ron Arkin montre ainsi, dans une tâche de prospection mettant en oeuvre plusieurs robots, que la transmission de l'état des robots permet de ramener plus rapidement l'objet recherché à la base car les robots coopèrent pour le transport. Le gain est limité, de l'ordre de 10%, mais la procédure de communication est très simple : si l'environnement d'un robot ne lui dicte pas de comportement précis, alors il se dirige vers le robot le plus proche ayant déjà repéré le but.

Alexis Drogoul et Jacques Ferber ont développé un système multi-agents pour une application de récolte. Inspirée par les comportements de récolte des fourmis, cette étude a montré comment des robots autonomes réalisant indépendamment la même tâche peuvent

---

[31] De *stigma*, piqûre et *ergon*, travail, oeuvre. La stigmergie est donc une « oeuvre stimulante » provoquant, par sa structure, la coordination des tâches de construction. Dans le cas de la stigmergie, l'ouvrier ne dirige pas son travail, il est guidé par lui (d'après [BON 94])

interagir et progressivement coopérer pour générer des chaînes permettant d'optimiser le transport de matières depuis le lieu de récolte jusqu'à la base [DRO 92].

Dans l'état actuel des recherches, la plupart des travaux sont restés limités à une étude par simulation de la dynamique émergeant d'un essaim de robots (les quelques systèmes implantés sur des robots réels sont le plus souvent restés limités à un faible nombre de robots - deux ou trois - [MAT 92] mais des systèmes de plus grande ampleur sont en cours d'étude). Même s'il est souhaitable de développer des systèmes réels, la phase de simulation reste souvent un préalable indispensable car elle permet d'étudier les comportements émergents plus simplement (et à moindre coût) qu'avec des systèmes réels [LEN 94]. Par contre, la simulation a eu comme conséquence indirecte que de nombreux travaux se sont éloignés des contraintes imposées par le monde réel pour ne s'intéresser qu'à l'aspect théorique des essaims de robots. Si l'apport théorique de ces études est indéniable, en particulier pour la compréhension des phénomènes d'émergence, il est plus limité en termes d'applications réelles. C'est en particulier le cas de l'utilisation d'algorithmes génétiques sur des populations de robots qui a fait l'objet de plusieurs communications [HAR 92][MIT 92][HOF 92]. Malgré un intérêt théorique indéniable, les contraintes de temps induites par l'application de tels systèmes dans un environnement naturel rendent une application pratique difficilement concevable. En outre, l'intérêt théorique qu'apporte le parallèle entre la robotique et la biologie ne doit pas restreindre le champ d'application et les outils de l'ingénierie multi-agents. Les contraintes propres à l'évolution et à la sélection naturelle interdisent l'apparition, en milieu naturel, de solutions nécessitant une dégradation temporaire des performances (les états intermédiaires n'étant pas pérennes, la sélection naturelle ne permet pas de les obtenir). Ces contraintes n'existent pas en ingénierie et il appartient au concepteur d'un système multi-agents de rechercher les solutions optimales en s'inspirant des systèmes naturels et en utilisant les techniques artificielles.

Outre les problèmes de communication et de coopération, l'un des points clefs de la conception de systèmes multi-agents réside dans l'architecture de contrôle utilisée pour chacun des agents. L'émergence est en effet un phénomène difficilement prévisible et généralement dépendant d'un grand nombre de paramètres (afin de permettre l'adéquation entre le comportement des agents et leur environnement). De plus il est pratiquement impossible, dans un environnement multi-agents (a fortiori s'il est réel et non simulé) de prévoir l'ensemble des configurations possibles. La majeure partie des agents développés actuellement sont basés sur l'architecture « Subsumption » (section III.3.2), souvent en utilisant des modules comportementaux spécifiques permettant la coopération [KUB 92][MAT 92]. Cependant cette architecture est, nous l'avons vu, construite manuellement ce qui ne permet pas aux agents de modifier leurs règles de décision en fonction du milieu dans lequel ils sont plongés : les règles de décision et les paramètres qui s'y rapportent doivent être fixés par le concepteur ce qui, dans le cas de coopération ou de communication complexes, peut s'avérer long et fastidieux.

L'étude des essaims d'insectes sociaux et d'autres collectifs biologiques permet de dresser une première ébauche d'architecture de contrôle pour des agents. En effet, les modèles biologiques nous montrent que les comportements sensori-moteurs mis en oeuvre dans les tâches coopératives sont de deux ordres : une composante stimulus-réponse et une composante séquentielle permettant, à partir d'un stimulus unique, de construire un ensemble de réponses ([BON 94], chapitre 3). Les comportements sexuels des insectes appartiennent souvent à la deuxième famille alors que les comportements de défense ou d'alimentation sont plutôt du type stimulus-réponse. La combinaison de ces deux types de relation entrée/sortie permet d'augmenter considérablement les potentialités comportementales et permet à un système d'exploiter, suivant les situations, des comportements réactifs ou des comportements spécifiques.

La deuxième information essentielle que nous fournissent les systèmes biologiques est l'importance de l'apprentissage dans l'organisation des sociétés ([BON 94], chapitre 2). Celui-ci permet en effet à certains individus de se spécialiser dans l'accomplissement de tâches spécifiques ce qui réduit les possibilités de conflits entre agents : selon un principe similaire à celui décrit ci-dessus pour la stigmergie, l'apprentissage par certains agents de tâches spécifiques supprime ces tâches de l'environnement des autres agents ce qui, partant d'un milieu homogène (les agents) conduit à un milieu hétérogène (des agents spécifiques). La spécialisation par l'apprentissage présente, sur la spécialisation a priori, l'intérêt de la souplesse et de la flexibilité. En outre, l'apprentissage permet de modifier dynamiquement les paramètres dont dépend l'apparition d'un comportement émergent [WEI 92].

III.5. Application à la productique

Au cours du chapitre II, nous avons montré comment l'évolution des systèmes de production vers des architectures plus souples c'est-à-dire plus flexibles et plus réactives, provoque une demande croissante de modules de commande présentant une autonomie maximale. Nous avions alors présenté succinctement le contrôle réactif comme étant une architecture potentiellement apte à répondre à cette demande. Les techniques du contrôle réactif permettent, en effet, de prendre en compte localement des informations sensorielles plus évoluées (donc d'améliorer robustesse et flexibilité) et de mettre en oeuvre des boucles sensori-motrices plus complexes.

Les potentialités du contrôle réactif sont nombreuses et quelques applications se développent dans différents secteurs. Cependant les caractéristiques propres de ce type de systèmes les destinent à des applications telles que la manutention, le transport de matière ou les applications péri-productives plutôt qu'à une utilisation directe sur des machines dont le programme doit être très strictement exécuté (MOCN). Dans ce chapitre, nous présentons cinq applications potentielles couvrant la majeure partie du système de production.

III.5.1. Assemblage

Les problèmes liés à l'assemblage de pièces mécaniques sont essentiellement de deux ordres : dans un premier temps un problème d'ordonnancement des tâches et, éventuellement, de choix des composants et, dans un second temps, un problème plus proche de la manutention mais avec des contraintes fortes quant aux positions respectives des différents éléments.

Suivant les typologies de production (taille des séries, personnalisation du produit), l'ordonnancement des tâches d'assemblage sera réalisé hors-ligne (a priori) ou en temps réel. Dans les deux cas, la remise en cause de l'ordonnancement établi est coûteuse en temps de calcul et risque donc de bloquer la production.

Pour limiter l'appel à une re-planification, Chris Malcolm et Tim Smithers [MAL 90a] proposent d'utiliser une architecture hybride pour le contrôle d'une cellule d'assemblage autonome (SOMASS). A partir d'une base de formes géométriques (version 3D des pentaminos), SOMASS doit construire, par assemblage, des formes plus complexes. Pour cela, C. Malcolm propose une architecture hiérarchisée proche de celle généralement utilisée pour ce type de système mais dont le niveau inférieur est composé d'un contrôleur réactif. La principale innovation de ce système réside dans le mode de communication entre le module d'ordonnancement et le module de commande. En effet, d'après les auteurs, il est erroné d'exprimer un ordonnancement sous la forme d'une séquence d'actions. Dans SOMASS,

l'ordonnancement n'exprime donc pas les actions à entreprendre en termes d'actions motrices de haut niveau mais en termes de mouvements de pièces, ce qui donne aux modules inférieurs une importante capacité de décision.

Chris Malcolm compare l'ensemble des couches basses à une « machine virtuelle » à laquelle le superviseur donne des ordres de haut niveau sans se préoccuper de la façon dont ils vont être exécutés. L'utilisation, interne à cette machine virtuelle, d'un contrôleur réactif permet une bonne flexibilité et une grande robustesse grâce à des procédures locales de rattrapage, dont l'influence est limitée aux modules comportementaux.

III.5.2. Manutention

Pendant longtemps les robots de manutention industrielle n'ont été que faiblement et indirectement connectés à leur environnement : l'absence d'une pièce ou son mauvais positionnement provoquait une erreur détectée par de simples contacteurs. Avec l'introduction de la flexibilité et de la réactivité dans les systèmes de production, les opérations de manutention sont devenues fortement dépendantes de leur environnement et du contexte de production. L'utilisation de l'information visuelle s'est donc largement répandue dans les systèmes de production, parfois jointe à d'autres sources d'informations (capteurs de pression, contacteurs optoélectroniques, ...). Parallèlement, les processus de contrôle sont devenus de plus en plus complexes entraînant un important besoin en puissance de calcul pour pouvoir gérer en temps réel la masse d'informations fournies par les capteurs. Plusieurs solutions ont été proposées pour gérer ce problème et accroître la puissance des manipulateurs, en particulier pour leur permettre la saisie « au vol » sur tapis roulants [TUN 94]. Dans un tel contexte, l'approche réactive et les approches de type « situated vision » (extraction, dans l'image, des éléments utiles sans traitement global) devraient permettre d'obtenir des manipulateurs à la fois plus robustes et plus flexibles. Jusqu'à présent les travaux sur les architectures réactives se sont concentrés sur la robotique mobile, mais l'application de ces principes à des bras robots permettrait probablement de généraliser la mise en oeuvre de manipulateurs « intelligents » en environnement de production.

Parmi les applications du contrôle réactif aux bras robots, on peut citer les travaux précurseurs de Thomas Miller [MIL 87] qui utilise un réseau de neurones pour relier les stimuli issus d'une caméra et les « schémas moteurs » permettant de centrer le bras au-dessus d'une cible mouvante, et plus récemment les travaux de Rod Grupen du « Laboratory for Perceptual Robotics » (Université du Massachusetts) sur un projet de perception visuelle reliée à un système bras/mains comportant 25 degrés de liberté (dont 5 pour le système de perception) [ARK 93b]. Même si ce projet est beaucoup plus ambitieux qu'un « simple » système de manutention, il permet d'envisager des transferts de technologie vers des tâches de manutention traditionnellement difficiles (par exemple dans l'industrie agro-alimentaire).

III.5.3. Usinage

Le domaine de l'usinage est généralement réservé à des machines très rigides exécutant un programme fourni par un laboratoire méthode, ou par un logiciel de CAO/CFAO. Dans le cas de la très petite série ou de la production unitaire (OKP[32]), le temps de conception devient alors très coûteux face au temps d'usinage. Quelques équipes de recherche (en particulier le

---

[32] **O**ne-of-a-**K**ind-**P**roduction

CRAN à Nancy avec le projet d'une machine-outil multi-broches, SHIVA) travaillent sur des machines outils commandées par des procédés multi-agents pour usiner des surfaces complexes en petite série. L'architecture proposée relève cependant plus de l'intelligence artificielle distribuée (chaque agent disposant de capacités cognitives « avancées » : planification, langage de communication, ...) que du contrôle réactif. A notre connaissance ces travaux sont encore très peu avancés, en particulier en raison de problèmes de perception. Dans le domaine de l'usinage de surface, l'utilisation d'architectures similaires à celles proposées par Ron Arkin ou David Payton (« Internalized Plans ») pourrait permettre la mise en place de machines-outils autonomes travaillant directement à partir des caractéristiques géométriques de la pièce à usiner (sous réserve cependant que l'on parvienne à extraire suffisamment d'informations « sensorielles » au cours de l'usinage). Cependant, dans le cas de l'usinage, la réactivité du système de pilotage ne peut être exploitée que si la machine elle-même possède des qualités de dextérité lui permettant de réaliser une même tâche (un même usinage) de différentes manières.

III.5.4. Transitique

Nous avons montré au cours du chapitre II, l'intérêt potentiel des robots mobiles autonomes (par opposition aux chariots filoguidés) pour les applications de transitique en milieu manufacturier. Ce secteur constitue le domaine privilégié du contrôle réactif en industrie car il permet de maîtriser le comportement des robots de transport sans faire appel à des ressources de calcul trop importantes. Deux types d'application peuvent être envisagés sur la base de l'utilisation de robots mobiles pour le transport des matériaux. Ronald Arkin, dans son application de AuRA aux systèmes de transports [ARK 90b][ARK 90c], propose d'utiliser les graphes vectoriels pour le pilotage d'un robot mobile. Cette application reste relativement proche des modèles classiques basés sur la planification de chemins (par exemple MARTHA [GIR 93] ou MONAMOVE [GUT 94]). Un tel système reste, en effet, relativement lourd à gérer et cher à mettre en oeuvre. L'intérêt de AuRA sur les modèles utilisant la planification est qu'il autorise une modification importante de l'environnement sans arrêter le système.

En marge de ce type d'architecture, les travaux sur les essaims de robots permettent d'envisager des applications beaucoup plus légères basées sur des robots moins évolués (donc moins chers) mais en plus grand nombre [DOT 93]. De tels robots, similaires à ceux utilisés dans le système de transport présenté chapitre II, peuvent être guidés par des balises visuelles (ou infrarouges), et utiliser une information sensorielle beaucoup plus limitée que les systèmes basés sur une planification prévisionnelle ou réactive. Par rapport à l'architecture de transport déjà présentée, l'utilisation d'un modèle réactif type « Subsumption » permet d'intégrer des comportements de « haut niveau » tels que l'approche d'une zone de chargement/déchargement, l'homéostasie (par exemple le maintien en charge des batteries) ou des comportements de coopération/communication entre les différents robots permettant d'améliorer les performances [DRO 92][FER 94]. En outre, les potentialités de ce type de système peuvent être étendues : l'activité des balises propres à chacune des machines peut, en effet, être modulée en fonction de la charge prévisionnelle de celle-ci, ce qui permet à un robot de choisir dynamiquement la destination d'une pièce en fonction des machines disponibles.

III.5.5. Applications péri-productives

Les environnements de production ne constituent qu'un cas particulier, structuré, des milieux naturels pour lesquels ont été conçus les robots mobiles. Or, ceux-ci ont été développés pour être utilisés pour des travaux qui, pour certains d'entre eux, sont généralisables aux milieux industriels où elles constituent souvent des tâches « péri-productives ». De même qu'ils sont directement utilisables pour les problèmes de transitique, des robots mobiles autonomes relativement peu évolués peuvent être directement utilisés pour des tâches de surveillance (pour lesquels ils offrent plus de souplesse que des systèmes fixes) ou pour des tâches de nettoyage. Plusieurs industriels spécialistes en robotique manufacturière développent actuellement des robots autonomes pour le nettoyage des hangars et des zones inoccupées. Pour ce type d'application, une planification stricte n'est pas indispensable et des contrôleurs réactifs peuvent, grâce à leur excellente robustesse et à leur coût modique, constituer une solution particulièrement efficace.

III.6. Contrôle réactif et apprentissage

La « théorie » du contrôle réactif tel qu'il a été défini par R.A. Brooks à travers ses différents travaux repose sur une adéquation quasi-parfaite entre un robot et son environnement pour que l'intelligence puisse émerger de leur confrontation. Or cette adéquation est, dans la plupart des architectures décrites ci-dessus, le seul fait du concepteur du robot. Celui-ci doit donc disposer d'un minimum d'informations sur les lois qui régissent l'environnement afin de concevoir le robot en conséquence. Même si cette connaissance est probablement moindre que celle mise en oeuvre pour la construction de modèles symboliques, il est difficile, pour le concepteur, d'appréhender l'ensemble de l'environnement dans lequel évoluera le robot. L'adjonction d'algorithmes d'apprentissage aux différents modèles de contrôle réactif est donc rapidement apparue comme une étape indispensable vers l'intelligence comportementale : l'acquisition de comportements dictés par l'environnement permet de décupler la portée de l'émergence, en particulier dans le cas des sociétés multi-agents.

Dans des communications consacrées aux principes généraux du contrôle réactif, R.A. Brooks [BRO 91] et Ron Arkin [ARK 93a] considèrent l'apprentissage comme un des défis majeurs auxquels le « Behavior-Based-Control » doit répondre pour augmenter les possibilités d'actions de leurs robots.

III.6.1. Motivations

Les motivations pour la construction d'algorithmes d'apprentissage sont, au moins, doubles : d'une part, la recherche d'une plus grande adéquation du robot à son environnement [MAE 92a] et, d'autre part, l'intérêt théorique soulevé par un prolongement des parallèles entre les sciences de l'artificiel (informatique, robotique, cybernétique, ...) et les sciences du vivant (biologie, neurobiologie, génétique, pédagogie, ...) qui leur servent souvent de base théorique.

L'intérêt pratique et les apports potentiels des mécanismes d'apprentissage au « Behavior-Based-Control » ont été exposés dès 1989 par Pattie Maes et Rodney Brooks dans un article qui a marqué le coup d'envoi des travaux dans ce domaine [MAE 90c]. D'après les auteurs, la difficulté majeure rencontrée lors de la conception d'une architecture réactive réside non pas dans la conception des différents modules comportementaux mais dans leur contrôle, c'est-à-dire dans le choix d'un mécanisme d'arbitrage permettant de choisir, à un instant donné quel(s) comportement(s) doit(vent) être actif(s). Ce problème comporte quelques variantes

suivant les architectures (un ou plusieurs comportements actifs simultanément, activation systématique de l'ensemble des comportements et choix d'une action moyenne, ...) mais il reste essentiel dans la plupart d'entre elles. Quels que soient les mécanismes utilisés pour cet arbitrage (choix manuel [BRO 86], langage de description [KAE 90] ou somme normalisée [ARK 87]) le résultat final est une règle arbitraire, stable au cours du cycle de vie du robot.

Cette approche devient rapidement inutilisable lorsque le nombre de comportements élémentaires augmente (c'est-à-dire lorsque la complexité du comportement global désiré croît) ou lorsque l'environnement est non-stationnaire. Dans le premier cas, en effet, le choix a priori devient tributaire du concepteur humain - qui ne peut avoir qu'une vision partielle - ou de principes théoriques incertains (la somme normalisée d'un grand nombre de schémas moteurs risque de conduire à un comportement anarchique si elle n'est pas elle-même soumise à des règles d'arbitrage). Dans le second cas, la solution fixée risque de devoir évoluer au cours du temps. Remarquons que, dans le cas d'un environnement non-stationnaire, c'est l'ensemble du système qui peut être amené à évoluer et non seulement les règles d'arbitrage. Pattie Maes propose donc des lois d'arbitrage reposant non plus sur une étude a priori mais sur une estimation de *pertinence*[33] et de la *sûreté*[34] des différents comportements en fonction de l'état de l'environnement. Les principes d'apprentissage décrits dans [MAE 90c] sont à l'origine d'une famille d'algorithmes que nous qualifierons de « statistiques ».

L'intérêt théorique pour l'apprentissage est largement motivé par l'origine biologique des « thèses » réactives. Comment ne pas exploiter plus encore les travaux des biologistes et pousser l'analogie jusqu'à s'inspirer des différents mécanismes d'apprentissage observés dans la nature ? Cette approche a conduit au développement de deux mécanismes d'apprentissage distincts : les algorithmes génétiques et les réseaux de neurones formels, tous deux antérieurs au « Behavior-Based-Control » (les réseaux de neurones sont apparus au cours de la seconde guerre mondiale et les algorithmes génétiques au milieu des années soixante-dix). Ces deux techniques sont basées sur des principes totalement différents : la théorie de l'évolution et de la sélection naturelle (Darwinisme) pour les algorithmes génétiques et l'apprentissage par l'expérience pour les algorithmes neuromimétiques. Leurs applications au contrôle réactif sont donc elles aussi très différentes en termes de principes et d'objectifs.

III.6.2. Apprentissage par renforcement

Toutes les méthodes utilisées pour l'apprentissage du comportement d'un robot mobile appartiennent à la catégorie d'apprentissage « par renforcement ». Le principe général est extrêmement simple, il a d'ailleurs été initialement énoncé dans les premiers travaux de psychologie (au XIXème siècle) avant d'être repris par les informaticiens dans les années soixante-dix/quatre-vingts [WID 73][BAR 83][SUT 88] : le comportement est noté suivant une heuristique fonction de l'état de l'environnement et l'algorithme d'apprentissage cherche à maximiser les retours positifs et/ou à minimiser les retours négatifs. Le problème posé à l'algorithme d'apprentissage est donc de rechercher une solution (exploration de l'espace des commandes, problème du « crédit structurel[35] ») et, lorsque cette solution (optimale ou sub-optimale) est trouvée, de cesser l'exploration pour renforcer les réflexes choisis. Le problème

---

[33] Relevant Behavior

[34] Reliable Behavior

[35] Structural Credit Assignnment Problem

de l'apprentissage peut être plus ou moins ardu selon que l'estimation fournie par l'heuristique dépend de l'action immédiate ou, au contraire, d'une séquence d'actions plus ou moins longue (problème du « crédit temporel[36] »).

III.6.3. Approche statistique

L'approche statistique de l'apprentissage repose sur un principe extrêmement simple : parmi l'ensemble des actions possibles le robot choisit celle qui d'après son expérience passée a la plus grande chance de produire le résultat le plus positif. L'algorithme doit arbitrer le choix entre la qualité de la réponse (le résultat le plus positif) et sa pertinence (le plus de chance de produire un résultat positif). L'algorithme de ce type le plus répandu est probablement le « Q-Learning » [WAT 92] et ses variantes (Q-Learning avec regroupement[37], Dyna-Q, ...).

Le Q-Learning utilise une matrice de correspondance entre situations et actions. Dans cette matrice sont stockées des valeurs Q représentant la perspective de renforcement. Le choix de l'action à un instant t est donc une fonction (partiellement stochastique) des valeurs Q associées à chaque action possible dans la situation considérée. Lorsque l'action choisie a été exécutée le nouveau signal de renforcement (s'il existe) et la nouvelle valeur de Q sont utilisés pour mettre à jour la valeur initiale (l'utilisation de la nouvelle valeur de Q pour la mise à jour de l'ancienne permet de résoudre le problème du crédit temporel, cette méthode de calcul est connue sous le nom de TD-Learning [SUT 88]). Les variantes du Q-Learning cherchent à réduire le temps d'apprentissage (qui peut être important lorsque la taille de la matrice croît) en généralisant les valeurs Q sur de nouveaux couples situation/action (c'est-à-dire en généralisant les valeurs apprises à des couples voisins ou en générant des séquences d'actions virtuelles).

Plusieurs auteurs ont appliqué le Q-learning au contrôle de robots mobiles (voir par exemple [SEH 94] pour une étude comparative des différentes variantes du Q-Learning appliqué au contrôle de robots mobiles ou [LIN 92] pour une étude comparative du Q-Learning et des modèles neuromimétiques appliqués au contrôle d'agents réactifs).

L'algorithme proposé par Pattie Maes [MAE 90b] est relativement proche, dans son principe, du Q-Learning. Cependant les auteurs attirent l'attention sur la capacité des algorithmes à s'intégrer aux architectures réactives : pour pouvoir s'intégrer à un contrôleur réactif, l'algorithme d'apprentissage doit être distribué sur l'ensemble des modules comportementaux, chacun d'entre eux calculant localement sa probabilité de réussite. Le Q-Learning en particulier ne correspond pas à cette description : les valeurs Q propres à chaque comportement peuvent être stockées localement mais leur calcul et le choix de l'action entreprise font tous deux appel à l'ensemble des valeurs Q associées à la situation (d'une façon générale, un algorithme totalement distribué ne permet théoriquement pas d'utiliser le TD-Learning ce qui pose plusieurs problèmes pratiques, en particulier celui du crédit temporel).

Dans l'apprentissage local, chaque module comportemental mémorise le nombre de réponses positives et de réponses négatives reçues alors qu'il était actif et alors qu'il était inactif. Ces valeurs lui permettent de calculer, d'une part, sa pertinence et, d'autre part, sa qualité (le calcul peut être étendu et intégrer des conditions secondaires). Une fonction

---

[36] Time Credit Assignment Problem

[37] Clustered Q-Learning

stochastique détermine ensuite, en fonction de ces deux valeurs, si le comportement correspondant est activé ou non. Signalons que cet algorithme est très limité : il ne permet pas a priori de résoudre les problèmes de crédit temporel et il se limite à des signaux de renforcement à trois états (positifs, négatifs ou nuls). Il est cependant bien adapté aux problèmes - simples - d'arbitrage entre plusieurs comportements tandis que le Q-Learning est plus indiqué pour l'apprentissage de séquences d'actions.

L'algorithme de P. Maes a été utilisé sur le robot GENGHIS (décrit section III.3.2.2) pour l'apprentissage de la marche hexapode. Le feed-back négatif est généré par deux contacteurs situés sous le robot (à l'avant et à l'arrière) et le feed-back positif par une roue tractée (un signal positif est produit lorsque la roue détecte un mouvement vers l'avant). Le robot est doté de treize comportements (un « avance » et un « recule » par patte auxquels s'ajoute un comportement global réduisant la somme des angles horizontaux des différentes pattes à zéro). Grâce à cet algorithme le robot parvient à acquérir une marche tripode en une dizaine de minutes. Même si la marche tripode nécessite une coordination des douze comportements, dans ce problème la combinatoire est relativement simple. Il est donc difficile de juger cet algorithme à partir de ce seul exemple.

III.6.4. Approche génétique

Comme nous l'avons vu au début de ce chapitre, la différence essentielle distinguant les modèles réactifs et les modèles symboliques repose plus dans l'approche générale que dans une différence essentielle de structure (la majorité des systèmes étant hybride). Non seulement l'architecture des modèles réactifs est inspirée de la structure des systèmes biologiques mais, de plus, la méthodologie incrémentale est très largement inspirée des principes évolutionnistes de Darwin. Les algorithmes génétiques, eux-mêmes largement inspirés des principes évolutionnistes et de la sélection naturelle, sont donc apparus naturellement comme la méthode d'apprentissage idéale pour le « Behavior-Based Control ». La création d'essaims de robots a augmenté l'intérêt des recherches dans ce domaine en renforçant l'analogie naturel/artificiel.

Les algorithmes génétiques ont été proposés en 1975 par J. Holland [HOL 75] comme outils d'optimisation d'inspiration biologique. Les différentes caractéristiques du système à optimiser sont représentées sous la forme d'une structure de données, le génotype, initialisée aléatoirement (le génotype ne comprend que les valeurs à optimiser, les valeurs constantes ou supposées comme telles n'y sont pas représentées). L'initialisation aléatoire permet de créer plusieurs génotypes différents qui constituent une *population*. Cette population de première génération est alors évaluée par rapport au problème posé (dans le cas d'un robot mobile l'évaluation caractérise l'adaptation du robot à son environnement). Cette évaluation est utilisée, de manière statistique, pour produire la deuxième génération. Ce principe, analogue à la sélection naturelle, permet aux algorithmes génétiques de déterminer, parmi la population initiale, le génotype le plus adapté mais il ne permet pas d'améliorer cette population. Pour cela, des algorithmes de mutation (modification aléatoire d'une partie du génome) et de croisement (échange partiel de génome entre deux individus choisis aléatoirement) sont appliqués à chaque changement de génération[38].

Les algorithmes génétiques sont particulièrement intéressants en raison de leur excellente faculté d'exploration. Ils permettent d'explorer rapidement (en quelques générations)

---

[38] Pour une approche plus complète des algorithmes génétiques voir, par exemple, [GOL 94]

un espace de recherche important tout en évitant (grâce aux croisements) les minima locaux. Par contre, la structure du génotype et la fonction d'évaluation, qui influent fortement sur les performances de l'algorithme, peuvent être relativement difficiles à déterminer.

L'application des algorithmes génétiques au contrôle de robots mobiles est, pour les raisons évoquées plus haut, un domaine de recherche très suivi et de nombreuses communications sont relatives à ce domaine. La plupart d'entre elles sont cependant restées au stade de la simulation [MIT 92][HOF 92][HAR 92]. L'utilisation « réelle » des algorithmes génétiques pose, en effet, d'importants problèmes de temps : l'évaluation des générations successives demande de consacrer à chaque élément de chaque génération un temps d'expérimentation non négligeable (une solution intéressante à plusieurs points de vue serait d'utiliser des populations d'individus réels mais les problèmes matériels - en particulier financiers - sont énormes. L'utilisation de robots miniatures tels que celui présenté ci-après constitue cependant une approche prometteuse). A titre d'exemple, Francisco Mondada propose dans [MON 94] d'utiliser les algorithmes génétiques pour fixer les relations dans une structure similaire aux « vehicles » de Valentino Braitenberg (voir section III.2.1.1). Cette étude a été menée sur le robot miniature « KHEPERA ». Pour ce problème, la combinatoire est relativement faible (KHEPERA comporte 8 capteurs de proximité à infrarouges et deux actionneurs) et les temps d'évaluation courts (30 secondes par individu) mais l'apprentissage complet demande de faire évoluer une population d'une centaine d'individus sur un total de 50 générations. Le temps d'apprentissage complet est donc de près de deux jours.

D'après les auteurs, l'approche proposée ci-dessus est assez simpliste (les algorithmes génétiques sont finalement utilisés pour fixer les poids d'un réseau de neurones) ; elle est cependant fréquemment suivie car l'approche connexionniste permet un codage très simple (mais non optimisé) du génome sous la forme des poids d'un réseau de neurones [SPI 92][BES 92b] [PAR 92b]. Dans le cadre du contrôle réactif, les algorithmes génétiques semblent plus indiqués pour l'optimisation des critères d'arbitrage entre comportements de plus haut niveau (d'une manière analogue à l'apprentissage proposé par Pattie Maes) que pour l'apprentissage des poids dans les réseaux de neurones. En effet, ceux-ci disposent déjà de mécanismes d'apprentissage puissants. Les deux techniques peuvent cependant parfaitement cohabiter en parallèle dans un même robot : les réseaux de neurones apprenant les séquences comportementales tandis que les algorithmes génétiques déterminent, suivant la situation, quel(s) réseau(x) doit(vent) prendre le contrôle du robot.

III.6.5. Approche neuromimétique

Il est impossible de clore ce chapitre sur l'apprentissage dans les systèmes réactifs sans aborder la question des réseaux de neurones. A l'heure actuelle, ils constituent un des paradigmes d'apprentissage les plus puissants et ils ont été largement appliqués au contrôle de processus en général et de robots en particulier.

Pourtant, malgré les nombreux travaux reliant les deux domaines, il subsiste une antipathie importante entre les deux approches, en particulier de la part des théoriciens du contrôle réactif qui, pourtant, utilisent des techniques analogues pour l'apprentissage comportemental (Pattie Maes, dans plusieurs articles, utilise des modes de représentation quasi-identiques aux réseaux de neurones [MAE 92a][MAE 92b] et plusieurs auteurs utilisent des approches similaires [CRU 92][CHE 92]). Rodney Brooks est l'un des plus virulents détracteurs des méthodes connexionnistes (voir section III.2). Dans un article plus récent, il propose une critique acerbe des algorithmes neuromimétiques et de la plupart des travaux correspondants (« *Les machines connexionnistes sont testées sur des problèmes préparés par*

les chercheurs eux-mêmes... Il a été fait un usage abusif des méthodes scientifiques[39]*... »* [BRO 91]). Cependant, il ne remet pas totalement en cause le principe d'un apprentissage connexionniste (il a lui-même proposé un algorithme d'apprentissage inspiré des modèles connexionnistes [BRO 90b]) mais plutôt les architectures connexionnistes qui, selon lui, ne sont pas adaptées au contrôle réactif.

Cet argumentaire est partiellement fondé. Toutefois, les algorithmes neuromimétiques sont une des approches les mieux adaptées pour l'apprentissage de comportements moteurs. En effet, l'exemple flagrant de la biologie (argument similaire à celui qui a conduit R.A. Brooks à développer le « Behavior-Based-Control » et l'architecture « Subsumption ») et les nombreuses applications pratiques du connexionnisme au contrôle de robot (citons, en particulier, [COU 94] qui propose un réseau de neurones très simple pour l'apprentissage de la marche hexapode, application similaire à celle présentée par [MAE 90b]) permettent d'envisager des architectures connexionnistes pour le pilotage réactif (architectures « Neuro-Réactives »). Par rapport à la plupart des algorithmes présentés ci-dessus, les réseaux de neurones présentent plusieurs qualités qui en font un instrument tout indiqué pour l'apprentissage du contrôle moteur. Ils sont en particulier doués d'une capacité de généralisation ce qui leur permet d'apprendre rapidement en extrapolant les comportements déjà acquis aux situations nouvelles (ce qui permet de limiter le nombre d'essais nécessaires) et de n'utiliser qu'un espace mémoire limité (par rapport à des algorithmes statistiques type Q-Learning). En revanche, et nous rejoignons en cela Rodney Brooks, la structure « Boîte Noire », centralisée, propre à la plupart des réseaux de neurones, est totalement incompatible avec les principes de modularité/parallélisme du contrôle réactif.

Avant d'aborder plus avant les problèmes et les avantages d'une union « Behavior-Based-Control »/Réseaux Neuromimétiques, il est indispensable de présenter les réseaux de neurones de façon plus approfondie. Le chapitre suivant présente donc l'approche neuromimétique et les différentes architectures de contrôle proposées dans la littérature (chapitre IV). Cette étude nous permettra, ensuite, d'étudier plus en détail les points communs et les discordances que présente cette technique d'apprentissage avec les modèles présentés ci-dessus.

III.7. Conclusion

Nous avions, dans le chapitre II, proposé l'utilisation du contrôle réactif en environnement de production afin d'améliorer la robustesse et la réactivité des systèmes de production. Bien que les travaux aient davantage portés sur des modèles de robots insectoïdes que sur des machines outils ou des robots manipulateurs, l'étude détaillée des principes et des architectures du « Behavior-Based-Control » confirme l'intérêt de ces méthodes, surtout si des voies de communication avec des systèmes d'ordonnancement et de planification se développent (systèmes hybrides). Malgré le faible nombre de travaux publiés, le paragraphe « application à la productique » (section III.5) permet d'ailleurs d'avoir une première idée de l'éventail des applications envisageables, en particulier en transitique, domaine vers lequel nous avons plus précisément dirigé nos investigations. Le transport des matériaux, qui met en oeuvre des robots autonomes faiblement contraints par une planification (seule la destination et,

---

[39] « They were usually tested on problems set up by the researcher. There were many abuses of the scientific method in these tests - the results were not always as the researchers interpreted them. » ([BRO 91], page 579)

éventuellement le temps de transport sont déterminants), permet en effet d'implanter directement les modèles réactifs dans un atelier flexible.

Par contre, cette étude a aussi permis de mettre en évidence les lacunes des modèles réactifs en termes d'adaptation et d'apprentissage alors que ces capacités semblent être indispensables à la parfaite adaptation des contrôleurs réactifs avec leur environnement (en particulier dans le cas des systèmes multi-agents). Aucun des modèles étudiés ne permet de répondre totalement aux critères recherchés (vitesse de convergence, apprentissage structuré, respect des contraintes imposées par l'architecture réactive : modularité, apprentissage en situation et par renforcement, pas de modèle symbolique, comportement sensori-moteur, ...). Malgré l'opposition de principe de certains théoriciens du contrôle réactif, les réseaux de neurones formels semblent être une approche prometteuse, en particulier en raison des origines biologiques qu'ils partagent avec le « Behavior-Based Control », et leurs aptitudes pour le contrôle des systèmes dynamiques complexes. Le chapitre suivant sera donc consacré aux modèles neuromimétiques et au neurocontrôle, ce qui nous permettra, dans un second temps, de d'ébaucher les grandes lignes d'une architecture « Neuro-Réactive ».

CHAPITRE IV :

ARCHITECTURES NEURONALES

POUR LE

CONTROLE DE PROCESSUS

Avant de définir une architecture regroupant contrôle réactif et apprentissage connexionniste, nous nous proposons, dans le présent chapitre, de détailler les architectures neuronales spécifiques au « neurocontrôle ». Cette étude nous permettra, dans un second temps, d'aborder plus précisément la question d'une architecture « Neuro-Réactive » et des critères auxquels doivent répondre les réseaux de neurones utilisés pour l'implémenter. Nous verrons qu'aucun des modèles connexionnistes « classiques » ne satisfait pleinement ces critères, principalement en raison de leur aspect monolithique.

Architectures Neuronales pour le Contrôle de Processus

IV.1 Introduction

Historiquement les premiers modèles neuromimétiques ont été développés dans la mouvance cybernéticienne. Ils devaient permettre de valider et d'utiliser en robotique les premiers modèles neurobiologiques de la conscience. Les cybernéticiens tendaient en effet vers une conception matérialiste réductrice de la conscience (conception béhavioriste selon laquelle la conscience est issue d'une combinaison de boucles réflexes [DEL 94]). L'adaptation informatique des principes organisationnels du cortex devait alors leur permettre de vérifier cette hypothèse.

Le contrôle de processus par réseaux de neurones (ou « neurocontrôle » [WER 89]) dispose en effet de beaucoup d'atouts, tant sur le plan pratique (apprentissage d'une commande non-linéaire, fusion multi-capteurs, possibilité d'apprentissage on-line et donc d'adaptation dans un environnement non-stationnaire, ... [HUN 92]) que sur le plan théorique : le neurocontrôle constitue en effet le lien le plus direct entre l'intelligence artificielle et la neurobiologie [WER 87a][WER 92]. De plus, le contrôle de processus pose des problèmes d'apprentissage souvent plus complexes que ceux posés par les problèmes de classification. Cela a conduit plusieurs auteurs à dépasser le cadre classique de l'apprentissage supervisé (professeur/élève) pour définir de nouvelles méthodes d'apprentissage puis des architectures neuromimétiques regroupant plusieurs réseaux (apprentissage du contexte/apprentissage de la commande). Pour certains auteurs le neurocontrôle serait même dissociable des méthodes neuromimétiques : P.J. Werbos se pose ainsi en « consommateur » de réseaux de neurones et pose le neurocontrôle comme intermédiaire entre le processus d'apprentissage et le processus à contrôler [WER 89].

Le principal apport du neurocontrôle est constitué par des architectures d'apprentissage telles que la « modélisation adaptative » (section IV.4.2) ou la « critique adaptative » (section IV.4.3). C'est pourquoi, après avoir esquissé une taxinomie des différents problèmes rencontrés dans ce domaine (c'est-à-dire des différentes « classes de processus » rencontrées) et après avoir brièvement présenté les principes fondamentaux du connexionnisme, nous consacrerons ce chapitre à l'étude des différentes architectures utilisées pour le contrôle des processus.

IV.2. Les acteurs de l'apprentissage connexionniste

L'apprentissage d'un processus par une architecture connexionniste peut se résumer succinctement à un échange d'informations entre trois éléments principaux : l'environnement (c'est-à-dire le processus à contrôler), l'élève (le contrôleur proprement dit) et le superviseur d'apprentissage, qui constitue le « professeur » (figure IV-1). Cette organisation, bien que très grossière, est suffisante pour nous permettre de définir une taxinomie des différentes applications du neurocontrôle en fonction des messages échangés entre ces trois acteurs.

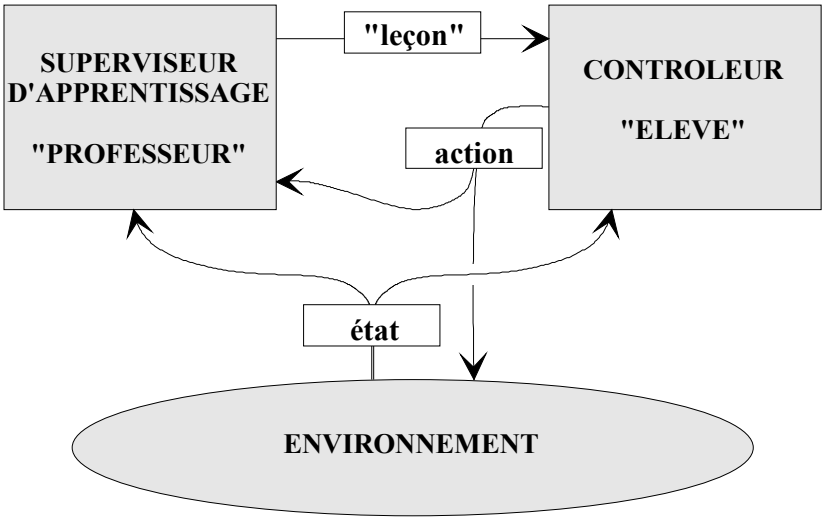


Figure IV-1 : *Les acteurs de l'apprentissage*

IV.2.1. Caractéristiques « pédagogiques »

Nous pouvons ainsi, dans un premier temps, différencier plusieurs classes de problèmes en fonction de la qualité du professeur ou, plus exactement, de la qualité de la « leçon ». Le contrôleur peut en effet recevoir deux types d'information :

- **l'action associée à un état particulier**. Dans ce cas le « professeur » a une connaissance complète du processus à contrôler.
- **la qualité de l'action entreprise**, estimée par le « professeur » en fonction de l'évolution de l'environnement.

Le premier cas correspond à un apprentissage direct du contrôle moteur associé à une perception de l'environnement. Le problème posé par l'apprentissage est alors peu différent d'une tâche de classification par apprentissage supervisé : on peut envisager le contrôleur comme une mémoire hétéro-associative des commandes motrices. Il peut cependant être rendu plus complexe lorsque le contrôleur doit associer une séquence d'actions à un état particulier de l'environnement. Quoi qu'il en soit ce mode d'apprentissage suppose toujours que le professeur ait une parfaite connaissance du problème traité.

Le deuxième cas, par contre, ne permet plus au contrôleur d'utiliser un apprentissage supervisé. Il doit alors explorer l'espace des commandes, en utilisant un apprentissage par renforcement, pour sélectionner parmi les différentes actions celle(s) provoquant l'évolution la plus positive de l'environnement. Ce problème d'exploration de l'espace des commandes est appelé « credit assignment problem ». Il se différencie de l'apprentissage d'une commande motrice par le séquencement des opérations d'apprentissage : ici le professeur doit observer l'évolution de l'environnement du temps t au temps t+1 (évolution provoquée par l'action au temps t) pour pouvoir générer le signal de renforcement alors que dans le premier cas le professeur fournissait directement, au temps t, l'action à entreprendre.

Malheureusement, il est souvent impossible de fournir, à chaque pas de temps, une estimation de la qualité de la réponse. En effet, le but est généralement de maximiser (ou minimiser) un ou plusieurs paramètres (par exemple atteindre une cible ou conserver un état d'équilibre). Dans ce cas, le signal de renforcement ne peut généralement être fourni que lorsque le but a été atteint ou lorsque la contrainte a été transgressée. Le contrôleur doit alors déterminer

quelle est la séquence d'actions conduisant à maximiser le signal de renforcement, sans connaître a priori, parmi les différentes actions composant cette séquence, celles qui ont été déterminantes. C'est donc le contrôleur lui-même qui doit estimer, à chaque pas de temps, le bénéfice accordé aux différentes actions. C'est le « Time-Credit-Assignment-Problem » pour lequel la « leçon » (le signal de renforcement primaire) n'est disponible qu'au terme d'une série plus ou moins longue d'actions.

IV.2.2. Caractéristiques dynamiques

Grâce à la notion de « leçon », nous avons donc pu isoler quatre familles principales d'apprentissage de processus :

- apprentissage d'une commande motrice,
- apprentissage d'une séquence motrice,
- « Credit Assignment Problem »,
- « Time Credit Assignment Problem ».

Celles-ci peuvent encore être subdivisées en fonction des caractéristiques dynamiques de l'espace d'état (ensemble des états pris par l'environnement). Thibault Langlois [LAN 92] propose ainsi une typologie en quatre classes principales, soit, par ordre de difficulté :

- Classe I : environnement statique déterministe

$$x(t+1) = f(t, y(t)) \qquad \textbf{(IV-1)}$$

où $x(t)$ représente l'état de l'environnement et $y(t)$ l'action au temps t .

- Classe II : environnement statique stochastique

$$x(t+1) = f(t, y(t), b(t)) \qquad \textbf{(IV-2)}$$

où $b(t)$ représente un bruit aléatoire.

- Classe III :environnement dynamique déterministe

$$x(t+1) = f(x(t), y(t)) \qquad \textbf{(IV-3)}$$

- Classe IV : environnement dynamique stochastique

$$x(t+1) = f(x(t), y(t), b(t)) \qquad \textbf{(IV-4)}$$

Cependant cette subdivision peut encore être étendue. En effet, le caractère dynamique de l'environnement peut être caractérisé par un nombre entier, son ordre, tel que, si i est l'ordre de l'environnement décrit par la fonction $f()$, on a :

$$x(t+1) = f(t, x(t), x(t-1), \ldots, x(t-(i-1)), y(t), b(t)) \qquad \textbf{(IV-5)}$$

Cela nous permet en effet de différencier deux formes totalement différentes d'environnement, qui représentent la plupart des cas : les environnements d'ordre 1 et les environnements d'ordre supérieur. Ainsi, lorsque i croît de façon importante, on doit considérer que la fonction $f()$ varie au cours du temps (environnement non stationnaire). Nous différencierons ensuite les environnements déterministes des environnements stochastiques.

IV.2.3. Etendue

Enfin, l'étendue des espaces d'état et de commande nous permet d'affiner cette décomposition. On distinguera empiriquement les espaces discrets (ou de faible dimension) et les espaces continus ou de forte dimension. Le cas le plus complexe étant, bien entendu le cas d'une forte dimensionnalité de l'espace d'état et de l'espace de commande. En effet, il devient alors impossible de posséder une connaissance exhaustive du problème et l'exploration ne permet pas de balayer efficacement l'ensemble des possibilités. Cependant la difficulté générée par l'utilisation d'espaces continus est souvent compensée par la précision de la perception et de l'action. C'est en particulier le cas pour le problème du pendule inverse (voir section VI.4.6 et annexe A6), qui est un problème non-linéaire classique mais qui devient cependant linéaire pour peu que le pendule reste proche de la verticale. L'utilisation d'un espace d'état continu et, a fortiori, d'un espace de commande continu permet de simplifier le problème en le restreignant aux zones de contrôle linéaire. A l'inverse, un contrôle type « *Bang-Bang* » (valeurs fixes +10N/-10N) et une perception discrète ne permettent pas de limiter le processus à une zone d'évolution linéaire (pour le pendule inverse un contrôle de type « *Bang-Bang* » et un espace d'état à valeurs réelles permet déjà de limiter le problème aux seules zones de contrôle linéaire).

IV.2.4. Conclusion

Chacune des classes de processus (d'environnement) proposées ci-dessus entraîne, pour l'apprentissage de la commande, des contraintes particulières et donc, nécessite une organisation particulière de l'apprentissage. Dans la suite de ce chapitre, nous présentons brièvement les modèles connexionnistes avant de passer en revue les principales organisations décrites dans la littérature. Nous les avons regroupées en trois familles différentes : les modèles utilisant l'identification (c'est-à-dire copiant un processus dynamique ou un contrôleur existant) et les deux modèles générant directement un contrôle moteur en fonction de l'état de l'environnement.

IV.3. Organisation générale des réseaux de neurones

IV.3.1. Introduction

Les réseaux de neurones, apparus dans la littérature dès 1943 d'abord comme explication du fonctionnement de la conscience[40], sont très librement inspirés de l'organisation « électrique » des cellules du cortex [McC 43]. On retrouve en effet dans leur architecture certaines caractéristiques propres au système nerveux animal. En particulier, les réseaux de neurones sont composés de nombreuses cellules interconnectées (les neurones) en un graphe analogue à celui formé par les dendrites et les axones (d'où le terme générique de « connexionnisme » utilisé pour l'ensemble de ces techniques). Le plus souvent la ressemblance entre les réseaux de neurones et le cortex se limite à ce seul principe organisationnel car les cellules utilisées restent généralement très simplifiées en regard de leurs homologues biologiques (de plus les modèles formels utilisent généralement un graphe régulier ce qui n'est pas le cas pour les cellules réelles).

---

[40] « Pour la première fois dans l'histoire de la science, nous savons en effet comment nous savons »
In the den of the metaphysicist, conférence de J. McCulloch au Centre philosophique de l'université de Virginie, 23 mars 1948.

Un réseau de neurones formels est donc constitué d'un graphe de cellules logiques échangeant des messages à travers un graphe de connexions. Il peut alors être entièrement défini par trois caractéristiques :

- **Les NEURONES** qui le composent, c'est-à-dire les noeuds de la structure. Ce sont des automates généralement très simples dont la logique de fonctionnement est indépendante de l'état global du réseau.

- **Sa TOPOLOGIE** qui correspond à l'organisation des connexions intercellulaires. En général elle est fixée a priori mais certains algorithmes permettent de la modifier dynamiquement.

- **Les LOIS DYNAMIQUES** du réseau qui régissent son évolution (règles d'apprentissage) et son comportement global (essentiellement les règles de mise à jour). Elles définissent comment le réseau apprend puis restitue l'information.

IV.3.2. Organisation microscopique : la cellule neuronale

IV.3.2.1. Le neurone biologique

L'unité de base de l'organisation du système nerveux est la cellule nerveuse ou le neurone. Cette cellule, bien qu'on en ait observé de nombreuses variantes, présente toujours la même fonctionnalité de transfert d'un signal électrique. Sa structure est d'ailleurs parfaitement adaptée à cette tâche ; on peut, grossièrement, la diviser en trois parties : les dendrites, le corps cellulaire (qui porte le noyau) et l'axone (figure IV-2).

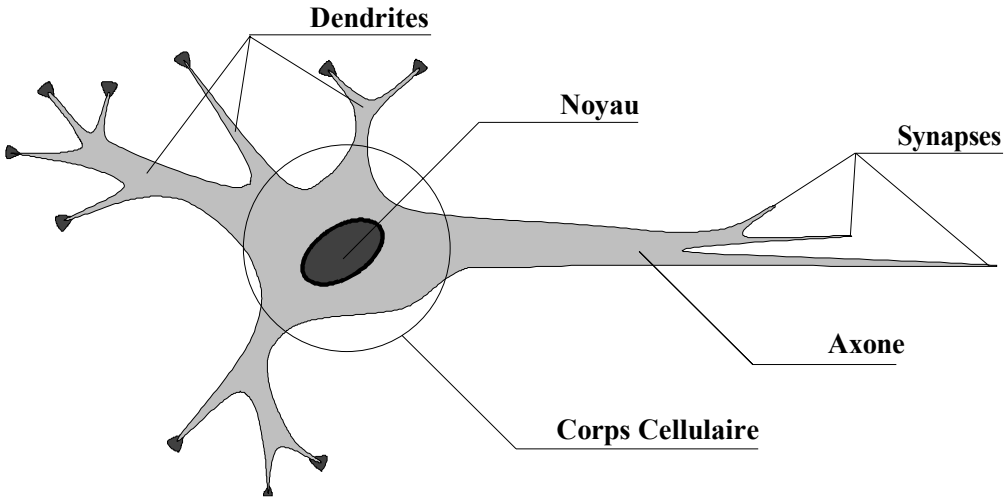


Figure IV-2 : *La cellule nerveuse*

L'influx nerveux, qui a longtemps été considéré comme un signal continu, est en réalité un train de différences de potentiels d'environ 110 mV. Il est collecté sur d'autres cellules par l'intermédiaire des synapses et conduit par les dendrites jusqu'à la cellule elle-même. L'axone, enfin, permet au neurone de le transmettre à distance vers d'autres cellules (de l'ordre du millimètre pour les neurones du cortex, la distance de transmission peut atteindre plusieurs centimètres pour les neurones moteurs). Cependant la fonction du neurone n'est pas une simple transmission de l'influx nerveux. La cellule agit comme un filtre passe-haut : les trains de potentiels recueillis par les axones sont additionnés (la fréquence d'un train de potentiel caractérise

l'intensité de l'influx nerveux) et la cellule n'est excitée que si la somme dépasse une valeur seuil. Elle émet alors un potentiel d'action sur son axone. Les potentiels transmis sont alors proportionnels à ceux reçus jusqu'à ce que la fréquence de décharge atteigne une valeur maximale.

La transmission de l'influx électrique le long des cellules nerveuses fait appel à des mécanismes d'échange moléculaire entre l'intérieur de la cellule (le soma) et l'extérieur. Des pompes ioniques permettent en effet au neurone de maintenir une différence de potentiel de -70 mV entre ces deux milieux (le milieu intérieur est riche en ions potassium tandis que le milieu extérieur est riche en ions sodium). En revanche, la transmission intercellulaire au niveau des synapses fait appel à des mécanismes de transmission chimique à travers l'espace synaptique. L'axone du neurone présynaptique est, en effet, séparé du neurone postsynaptique par un espace de quelques centièmes de millimètres au travers duquel circulent les neuromédiateurs, c'est-à-dire les hormones spécifiques à la transmission de l'influx nerveux d'une cellule à l'autre. Les synapses constituent l'élément clef du stockage de l'information car elles sont capables de moduler leur activité en fonction de l'activité des neurones présynaptiques et postsynaptiques selon une règle proche de la loi de Hebb [HEB 49] (d'autres cellules peuvent influer sur la plasticité des synapses, par exemple les interneurones ou les cellules de Purkinje, mais nous nous limitons ici à une explication simplifiée de ce phénomène).

IV.3.2.2. Le neurone formel

Le neurone formel est un automate caractérisé par un petit nombre de fonctions mathématiques. Il traite un signal d'entrée recueilli à travers ses connexions entrantes pour fournir un signal de sortie calculé par la fonction de transfert. Son état est caractérisé par trois variables E, A et S calculées au moyen de trois fonctions successives : la fonction d'entrée $f()$, la fonction d'activation $g()$ et la fonction de sortie $h()$ (figure IV-3).

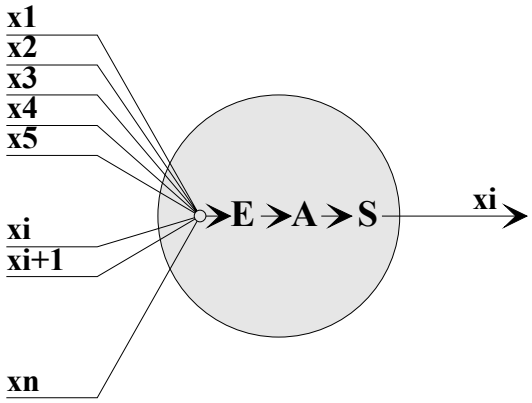


Figure IV-3 : *Modèle général du neurone formel.*

On a donc successivement : $E = f(\vec{x})$, $A = g(E)$ et $S = h(A)$. La définition de ses trois fonctions permet donc de fixer le comportement du neurone.

La fonction d'entrée $f()$:

Dans un réseau de n neurones chacun d'entre eux reçoit en général de 1 à n stimulations en provenance des autres cellules. Ces valeurs, binaires ($\{0,1\}$ ou $\{-1,1\}$) ou réelles, sont en général pondérées puis additionnées par la fonction $f()$:

$$E = f(\vec{x}) = \sum_{j=1}^{n} W_j x_j \qquad \textbf{(IV-6)}$$

W_j est la pondération appliquée par le neurone à l'entrée j. Si le neurone dont on calcule l'entrée est le neurone i, on note W_{ij} la pondération correspondant à l'arc reliant le neurone j au neurone i. On parle alors du poids de la connexion.

Certaines cellules utilisent des fonctions d'entrée différentes. Ainsi, par exemple, les unités Sigma-Pi (ΣΠ) réalisent des sous-produits pondérés entre groupes d'entrées avant d'additionner les résultats obtenus.

La fonction d'activation $g()$:

Alors que la fonction d'entrée varie peu d'une famille de réseaux à l'autre (hormis l'ensemble d'entrée), la fonction d'activation des neurones est généralement l'une des caractéristiques principales d'un type de réseau particulier. En effet l'état A d'un neurone peut être à valeurs binaires ou réelles et la fonction d'activation peut fortement varier d'un type de réseau à l'autre. Les deux familles principales sont la fonction signe, pour les neurones à états binaires, et la fonction sigmoïde pour les neurones à états réels (figure IV-4). Ces deux fonctions sont généralement décalées par l'ajout d'un seuil fixe (noté θ_i ou, par analogie avec une stimulation constante, W_{0i}). D'autres fonctions sont cependant utilisables telles que les fonctions gaussiennes ou les fonctions linéaires à seuil.

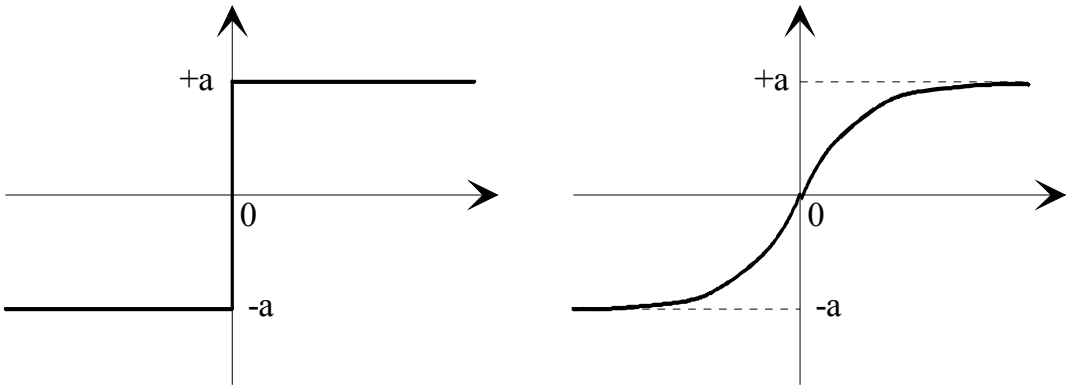


Figure IV-4 : *Les fonctions « Signe » et « Sigmoïde »*

Fonction Signe : $A = \mathrm{a}(2H(E) - 1)$, ***(IV-7)***

Avec $H(\ )$ *fonction de Heaviside :* $H(x) = 1$ *si* $x > 0$,
$H(x) = 0$ *sinon.*

Fonction Sigmoïde : $A = \mathrm{a}\dfrac{(e^E - 1)}{(e^E + 1)}$ ***(IV-8)***

La fonction de sortie $h()$:

L'une des plus importantes particularités de l'automate « neurone » est de n'avoir qu'une sortie pour un grand nombre d'entrées. Binaire ou réelle, la sortie S est en général du même type que les entrées. La fonction de sortie est le plus souvent la fonction identité et on a $S = A$. Dans le cas contraire on construit une fonction g' telle que $g' = h \circ g$ ce qui permet de se ramener au cas précédent.

La fonction de transfert $F(\ )$:

La fonction de transfert du neurone est la combinaison successive des fonctions d'entrée, d'activation et de sortie. Elle permet de calculer l'état du neurone au temps $t+1$ à partir de son état au temps t .

Soit, pour le neurone i : $x_i(t+1) = S = h \circ g \circ f[\vec{x}(t)] = F[\vec{x}(t)]$

et, dans le cas général (neurone à seuil) :

$$x_i(t+1) = g\left(\sum_{k=1}^{n} W_{ik} x_k(t) - \theta_i\right) \qquad \textbf{(IV-9)}$$

Les modèles classiques :

Le neurone de Mc Culloch et Pitts :

Historiquement le premier modèle de neurone formel a été défini en 1943 par Warren Mc Culloch et Walter Pitts [McC 43][41][McC 47]. Il s'agit d'un automate à état binaire ($\{0,1\}$) dont la fonction de transfert est la fonction de Heaviside appliquée à la somme des entrées pondérées auxquelles est ajouté un seuil θ_i (figure IV-5).

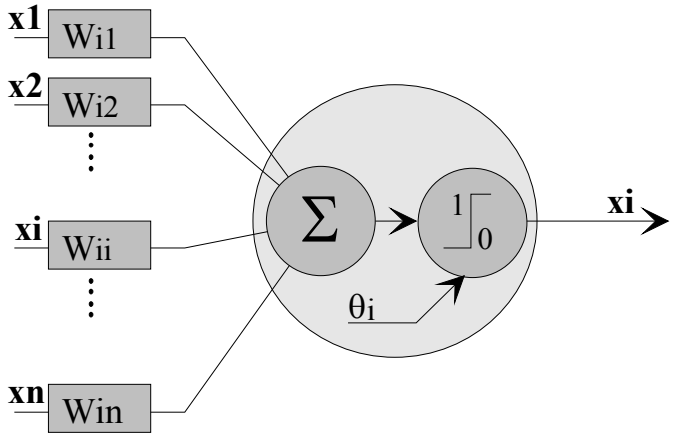


Figure IV-5 : *Le neurone de Mc Culloch et Pitts*

Initialement les poids étaient calculés analytiquement (inversion des matrices d'entrées/sorties) et ce type de neurone n'était pas doté de fonctions d'apprentissage. En 1957,

[41] Une traduction en français est disponible dans [PEL 95]

Frank Rossenblatt a comblé cette lacune avec le Perceptron. Ce réseau est en effet capable d'apprendre une classification sur les entrées [ROS 58]. Le modèle neuronal de Mc Culloch et Pitts, bien que très simple, est encore utilisé pour certaines familles de réseaux, mais il a tendance à disparaître au profit du modèle à valeurs réelles.

Le modèle « continu » :

Dès 1960, avec l'ADALINE (« ADAptive LInear NEtwork »), Bernard Widrow propose un modèle de neurone dans lequel les poids sont appris à partir des valeurs de stimulation tandis que cette valeur réelle est ensuite filtrée par une fonction de Heaviside (voir figure IV-4) pour produire la sortie du neurone [WID 60]. Même si l'ADALINE ne prend pas réellement des valeurs réelles, on peut considérer ce modèle de neurone comme le précurseur du neurone « continu ».

C'est dans les années quatre-vingts, d'abord avec les travaux de John Hopfield [HOP 84] puis avec l'apparition de l'algorithme de la rétropropagation du gradient d'erreur [RUM 86b] que les neurones continus ont conquis leurs lettres de noblesse. Les modèles les plus courants sont les neurones à fonction de transfert dérivable qui permettent de rétropropager l'erreur par dérivation. Dans ce modèle, les entrées sont pondérées puis additionnées. L'état du neurone est ensuite calculé grâce à une fonction dérivable généralement impaire (fonction sygmoïdale) ou paire (fonction gaussienne).

Les unités « ΣΠ » :

Dans certains cas, il peut s'avérer nécessaire d'utiliser des neurones dont la fonction d'entrée n'est pas simplement affine, c'est-à-dire que les neurones ne réalisent pas la simple somme pondérée de leurs entrées mais, par exemple, le produit de plusieurs entrées. Ainsi, dans le le cas des unités ΣΠ, les entrées peuvent être multipliées entre elles (on se limite en général à des produits de deux entrées) avant d'être pondérées puis additionnées (figure IV-6). Il est ainsi possible de créer des unités plus puissantes en introduisant des phénomènes d'exclusion (multiplicateur nul). Les entrées multipliées sont appelées entrées conjuguées.

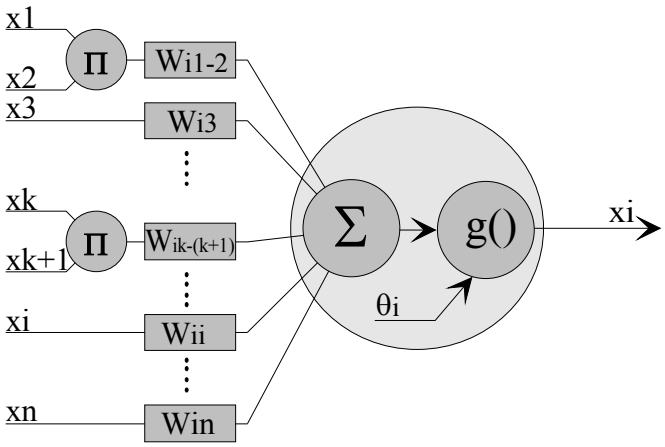


Figure IV-6 : *Unité ΣΠ*

IV.3.3. Organisation macroscopique : la topologie

IV.3.3.1. Organisation des fonctions dans le cortex

Bien que les thèses de localisation des fonctions cérébrales et de la mémoire sur des cellules neuronales[42] aient longtemps été retenues pour expliquer le fonctionnement du cerveau, la progression de la neurobiologie a progressivement réfuté cette vision au profit de la distribution des fonctions sur une population de neurones plus large. L'organisation macroscopique du cerveau montre cependant que les fonctions principales (en particulier les fonctions motrices, sensorielles et somatosensorielles) sont localisées sur des aires cérébrales distinctes. Il est ainsi possible de décomposer la chaîne de perception visuelle en couches traversées successivement par l'influx nerveux porteur de l'information oculaire (aire V1 à V5, aire NT, ...). De même, certaines aires cérébrales sembles êtres spécifiquement dédiées aux fonctions sensori-motrices propres à chacune des zones du corps (bras, jambes, mains, ...). Il est important de préciser cependant que cette localisation des fonctions cérébrales reste limitée à un niveau macroscopique : même si la notion d'aires cérébrales rappelle les principes fondamentaux de l'intelligence artificielle symbolique ou la notion d'homonculus, ces aires ne constituent pas des entités distinctes au niveau microscopiques et la communication inter-aires cérébrales s'effectue au niveau cellulaire par des connexions synaptiques classiques. Les aires cérébrales observées en imagerie cérébrale ont probablement plus une origine topologique (regroupement des fonctions sensori-motrices proches) qu'une origine fonctionnelle (décomposition/agrégation des fonctions sensori-motrices).

On notera que cette présentation a totalement passé sous silence plusieurs aspects importants du système nerveux, à savoir le système limbique et les assemblées de neurones. Le premier, qui régit l'ensemble des communications chimiques (hormonales) dans le cortex, a en effet été totalement ignoré par les concepteurs des réseaux de neurones alors que son importance, bien que mal comprise, semble prépondérante, en particulier pour l'apprentissage. Le deuxième aspect, les assemblées de neurones, traduit la « semi-localisation » des fonctions cérébrales sur les cellules nerveuses, avec une granularité plus faible que les aires cérébrales. Contrairement aux réseaux de neurones formels qui sont des entités essentiellement monolithiques, les fonctions cérébrales sont localisées, au sein d'une même aire, sur des groupes fonctionnels diffus identifiés par leurs rythmes de fonctionnement propres : les « assemblées de neurones ». Celles-ci semblent être particulièrement importantes pour permettre l'exécution parallèle (mais corrélée) de deux traitements différents sur un même stimulus.

IV.3.3.2. Topologie des réseaux de neurones formels

Contrairement aux neurones formel qui sont une modélisation grossière des neurones biologiques, l'organisation topologique des réseaux connexionnistes n'est généralement pas dictée par une quelconque analogie avec les structures neurobiologiques. On ne retrouve donc pas, dans les réseaux de neurones, les notions « d'aires cérébrales » ou de « zones fonctionnelles ». De fait, les règles d'organisation topologique des réseaux de neurones varient énormément d'un modèle connexionniste à l'autre. Nous allons essayer ici de décrire quelques-unes des grandes règles topologiques utilisées.

---

[42] principe de la « Grandmother Cell ». Certaines études récentes montrent que, pour des tâches très particulières (reconnaissance du visage ou de la nourriture), des cellules réagissent spécifiquement à un stimulus particulier.

Cas général :

Bien que les règles d'organisation puissent être très variables, la description de la topologie est toujours donnée très simplement par l'ensemble des fonctions d'entrée des neurones composant le réseau (nous nous limitons ici au cas des fonctions d'entrée affines, ce qui exclue les unités ΣΠ décrites auparavant). En effet, on considère qu'une connexion entre deux neurones i , j (de i vers j) est inexistante lorsque la pondération correspondante est nulle ($W_{ji} = 0$). La topologie d'un réseau de n neurones (réseau de dimension n) peut alors être décrite sous la forme d'une matrice carrée : la matrice des poids W.

Cette écriture permet de décrire très simplement les différentes architectures employées. Elle permet, en particulier, de définir la notion de chemin d'activation dans un réseau de neurones : *un chemin d'activation est une suite de neurones reliés par des connexions non nulles*. Un chemin d'activation peut être bouclé (c'est-à-dire qu'un même neurone peut apparaître plusieurs fois dans la suite). En conséquence, il peut y avoir, dans un même réseau, une infinité de chemins d'activation.

Nous allons maintenant définir, à partir de ces notions, quelques organisations topologiques caractéristiques.

Les réseaux « Feed-Forward » :

Un réseau est dit « feed-forward » (« passe-avant ») si toute suite $W_{ij}, W_{jk}, \ldots, W_{lm}$ ($W_{xy} \neq 0$) est de longueur finie ; c'est-à-dire s'il n'existe pas dans le réseau de chemin d'activation contenant deux fois le même neurone (figure IV-7).

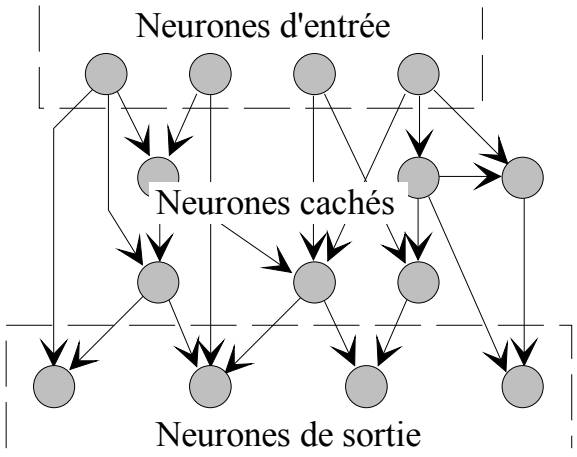


Figure IV-7 : *Exemple de réseau « feed-forward »*

Les modèles les plus simples de réseaux « feed-forward » sont les « réseaux » monocellulaires dans lesquels un unique neurone de sortie reçoit, via des connexions pondérées, toutes les valeurs d'entrée (voir annexe A2, section 1).

Les réseaux en couches :

Il s'agit d'une forme particulière de réseaux « feed-forward » dans laquelle une cellule est reliée aux neurones d'entrée par des chemins de longueur constante. On peut alors regrouper les neurones en couches selon la longueur de ce chemin (figure IV-8). On parle alors de réseaux à deux couches (dans lesquels les neurones de sortie sont directement reliés aux neurones d'entrée), à trois couches (une couche de cellules dites « cachées » sépare les neurones d'entrée des neurones

de sortie), à *k* couches ... Le terme « réseaux en couches » sous-entend généralement que la connectivité d'une couche à la suivante est totale (c'est-à-dire que tout neurone d'une couche est connecté à l'ensemble des neurones de la suivante).

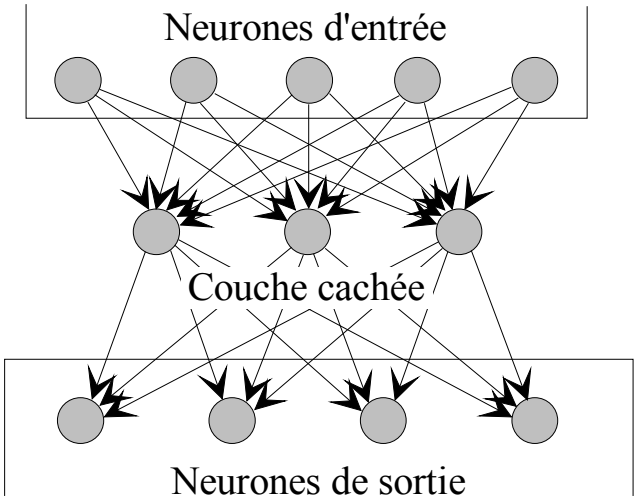


Figure IV-8 : *Exemple de réseau à trois couches*

Les réseaux récurrents :

Par opposition aux réseaux « feed-forward » et, a fortiori, aux réseaux en couches, les réseaux récurrents peuvent contenir des chemins bouclés, passant plusieurs fois par un même neurone (figure IV-9). Grâce à cette structure cyclique, un stimulus entrant peut être partiellement ou totalement remis en question par l'état antérieur du réseau (d'où une notion de contexte) ou par l'arrivée de stimuli postérieurs. Ce type de réseau a donc des capacités théoriques supérieures à celles des réseaux non-récurrents. Malheureusement chaque neurone d'un réseau récurrent peut être soumis à sa propre influence moyennant un chemin plus ou moins long (théoriquement entre une et une infinité de connexions). Les réseaux récurrents présentent donc une dynamique complexe due aux multiples rétroactions internes (voir section IV.3.5.2).

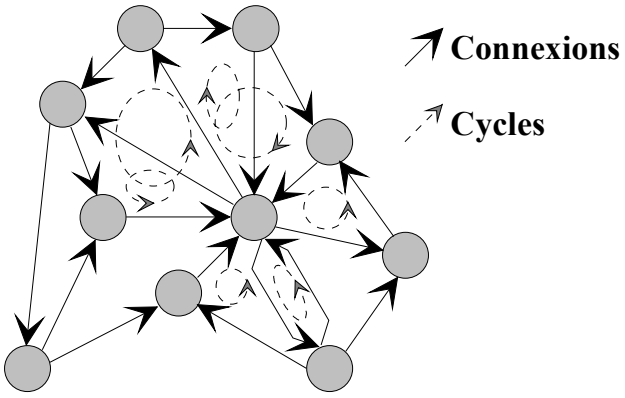


Figure IV-9 : *Exemple de réseau récurrent*
Tous les cycles présents dans le réseau ne sont pas représentés sur la figure.

Un cas particulier, les autoconnexions :

Parmi les composantes de la matrice des poids, les éléments diagonaux ont une signification particulière. En effet, ils représentent l'influence directe de l'état d'une cellule au

temps t sur son état au temps $t+1$. Ces connexions particulières sont appelées « autoconnexions », elles permettent aux neurones de « mémoriser » leurs états courants et de les utiliser pour calculer leurs états futurs en les ajoutant, après pondération, aux stimulations qu'ils reçoivent. Un réseau autoconnecté est donc un réseau dans lequel la matrice des poids est à diagonale non nulle.

Les réseaux totalement connectés :

Si, dans un réseau, la matrice des poids ne comporte aucun élément nul, alors le réseau est dit totalement connecté. C'est-à-dire que tout neurone reçoit et émet des stimulations vers l'ensemble des neurones du réseau. Cette définition exige, en particulier, la présence d'autoconnexions. En pratique, le cas d'un réseau totalement connecté à l'exception des autoconnexions étant relativement fréquent, on précisera en général si celles-ci sont présentes ou non.

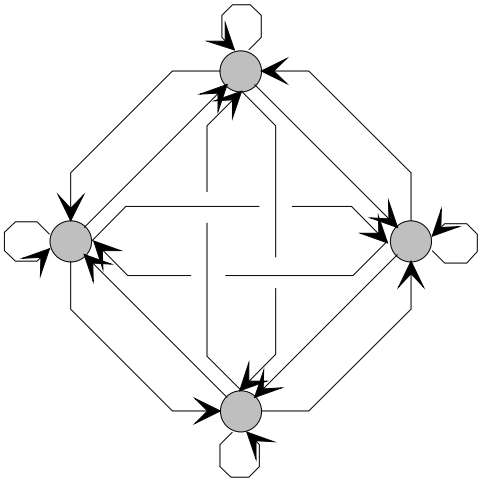


Figure IV-10 : *Exemple de réseau totalement connecté*

Les Réseaux symétriques :

Un réseau de neurones est dit symétrique si pour tout couple de neurones de ce réseau on a :

$$W_{ij} = W_{ji} ,$$

c'est-à-dire si la matrice de poids est symétrique. Tout réseau symétrique est, par définition, un réseau récurrent.

IV.3.4. Organisation interne de l'information

L'intérêt des méthodes connexionnistes provient directement ou indirectement de la répartition de l'information sur la structure. C'est, en effet, cette organisation, couplée à un parallélisme massif (bien que souvent simulé), qui permet aux réseaux de neurones de traiter des informations bruitées (le bruit est éliminé par les nombreux recoupements internes entre les informations) ou incomplètes (l'association, au niveau des connexions, entre les différents éléments permet alors au réseau de restituer l'information complémentaire). De même, la distribution de l'information permet un important gain en robustesse. Cependant nous verrons que cette dilution peut aussi être source de conflits entre les différentes informations et limiter ainsi les capacités des réseaux.

Avant d'étudier plus avant les méthodes connexionnistes, nous allons donc aborder rapidement le problème de la représentation de l'information dans les réseaux de neurones formels. Notre but n'est cependant pas d'étudier en détail leur organisation interne, mais de fournir au lecteur les bases qui lui permettront d'aborder plus aisément les chapitres suivants.

IV.3.4.1. Un cas particulier : les représentations symboliques

Bien que les réseaux de neurones soient souvent opposés aux méthodes classiques, « symboliques », de l'intelligence artificielle, l'utilisation des structures connexionnistes n'entraîne pas systématiquement la distribution de l'information. Ainsi certains modèles de réseaux de neurones manipulent des représentations locales de l'information en associant les cellules à des éléments symboliques.

Ainsi les réseaux de Hopfield [HOP 82] (annexe A2, section 3), utilisés en optimisation, associent à chaque cellule une configuration particulière de leur espace de recherche (par exemple, dans le problème du voyageur de commerce, le numéro d'ordre d'une ville [HOP 85]). Les critères d'optimisation permettent alors de calculer la pondération associée à chacune des connexions.

De même, les mémoires associatives [AMA 72][KOH 77](voir annexe A2, section 2) permettent de coder un ensemble de relations symétriques entre deux groupes de symboles et donc, à partir d'un ou plusieurs de ces symboles, de retrouver par propagation la famille associée. Ainsi, dans un système développé par Yves Cochet et Gérard Paget [COC 88], une base d'images est reliée à un ensemble de descripteurs par l'intermédiaire d'une mémoire associative, ce qui permet à un utilisateur de retrouver une image en activant les descripteurs désirés.

Bart Kosko a proposé un modèle symbolique de réseau de neurones appelé « carte cognitive » (« cognitive map » / « fuzzy cognitive map » [KOS 86a]). Dans ce modèle chaque cellule est associée à un concept particulier. Celles-ci sont reliées par des connexions représentant les relations de cause à effet entre les symboles associés (annexe A2, section 5).

Dans ces réseaux, la représentation de l'information peut être qualifiée de locale car l'ensemble des symboles manipulés est en relation bijective avec l'ensemble des cellules du réseau. On notera cependant que cette qualification n'a de sens que pour un niveau de description donné dans un ensemble symbolique prédéfini [MID 91]. Ce type de représentation a l'avantage de faciliter la communication entre le réseau et l'utilisateur (cela permet, par exemple, d'associer des processus de décision connexionnistes à des bases de données symboliques). En outre, lorsque les connexions sont associées à des relations symboliques (type « cause à effet »), l'utilisation de règles d'apprentissage dans ces modèles peut permettre d'enrichir la base de symboles.

Par contre, le contenu sémantique de chacun des noeuds limite les capacités de ces modèles. En effet, les représentations internes étant fixées, le développement ou l'apprentissage de nouveaux concepts reste impossible (à moins d'ajouter de nouvelles cellules, des mécanismes incrémentaux peuvent être utilisés à cet effet [COC 88] [BIE 90a] [BIE 90b]) et on ne maîtrise pas toujours la localisation de la solution (ainsi, si la solution recherchée est une combinaison de plusieurs concepts, la réponse du réseau doit être analysée manuellement). De même, dans un réseau à représentation locale, les capacités de généralisation doivent être explicitement introduites par la création de connexions entre les concepts voisins [MID 91]. En revanche, pour de tels réseaux, le contenu symbolique des différents neurones a permis des développements mathématiques poussés et on dispose de critères d'apprentissage assez précis (exclusion mutuelle entre différentes combinaisons apprises, capacité en fonction du nombre de neurones, ...).

IV.3.4.2. Cas des réseaux en couches

Dans les réseaux en couche, seules les cellules appartenant à la couche d'entrée ou à la couche de sortie sont associées à une information symbolique. Elles réalisent l'interface entre l'information distribuée sur les couches cachées (plus exactement sur les connexions cachées) et l'utilisateur du réseau. Dans ce cas particulier, chaque couche de connexions peut être assimilée à un fonction mathématique linéaire appliquée aux entrées, celle-ci étant répartie sur l'ensemble des connexions (chaque critère d'entrée est associé, par une pondération, aux critères de sortie). On montre cependant que toute fonction continue bornée peut être identifiée par un réseau à trois couches [FUN 89][HOR 89]. La répartition de l'information entre les connexions d'une même couche (et entre les différentes couches) est étroitement liée à l'algorithme utilisé : l'algorithme le plus couramment utilisé, la rétropropagation du gradient, provoque une « dilution » générale de l'information mais ce n'est pas le cas, par exemple, pour les modèles « RBF-Nets » [CHE 91] (annexe A2, section 7) qui sont basés sur la décomposition de la fonction mathématique globale en fonctions locales, propres à chaque cellule cachée, se référant à un « noyau » (« Kernel ») identifié dans l'espace d'entrée.

La distribution de l'information dans les réseaux utilisant la rétropropagation du gradient est à l'origine de la plupart des contraintes pesant sur ce type de réseaux. Ainsi l'apprentissage doit être organisé de façon à fournir au réseau une information cohérente dans le temps. Il n'est pas possible, avec de tels réseaux, d'apprendre successivement différentes informations. L'apprentissage de nouvelles informations dans un réseau déjà entraîné risque en effet de provoquer un phénomène « d'oubli catastrophique » : les pondérations préalablement fixées sont modifiées et le réseau « désapprend » [FRE 91]. Dans ce type de réseaux, toutes les informations apprises doivent être fournies alternativement au cours d'une même séance d'apprentissage.

IV.3.4.3. Le cas des réseaux récurrents

La représentation de l'information dans un réseau récurrent est sans conteste le plus complexe. En effet, ces réseaux ne peuvent plus être assimilés à un « simple » filtrage de l'information entrante. L'aspect dynamique de ce type de structure leur permet en effet de mémoriser l'information temporelle fournie en entrée (le contexte).

Pour ce type de réseaux le terme de « boîte noire » est encore grandement d'actualité. L'information entrante est, en effet, répartie sur l'ensemble des connexions en fonction de son contenu immédiat et de l'état antérieur du réseau. D'autre part, l'information restituée est elle-même fonction du temps : elle évolue progressivement jusqu'à atteindre un état stable (dans le meilleur des cas), cyclique ou chaotique si le réseau contient des neurones à état réel[43].

IV.3.5. Organisation temporelle : dynamique

La dynamique d'un réseau de neurones est directement fonction de sa topologie (longueur des chemins d'activation) mais aussi des lois internes utilisées pour calculer les états successifs des neurones au cours du temps. Chacune des cellules étant le plus souvent un automate à temps discret ($x_i(t+1) = F\left(x_1(t), x_2(t), ..., x_n(t)\right)$ **(IV-10)**), l'ensemble du réseau est généralement soumis

[43] Si le réseau est constitué de neurones binaires, alors l'espace d'état est fini et dénombrable et le réseau ne peut présenter que des états stables ou des cycles (sous réserve que la fonction de transfert des neurones soit déterministe).

à une horloge interne. Le temps propre de chacun des neurones est alors fonction de la loi d'activation utilisée dans le réseau. Ces notions n'ont cependant d'intérêt réel que pour les réseaux récurrents. Pour les réseaux non-récurrents, le mode de mise à jour utilisé n'influe pas sur la dynamique de réseau.

IV.3.5.1. Les lois d'activation

Le principe d'activation des neurones dans un réseau est généralement très simple. Le réseau R est divisé en k sous-ensembles r_i tels que :

$$R = \bigcup_{i=1}^{k} r_i \text{ , avec } r_i \cap r_j = \varnothing, \forall i \neq j \qquad \textbf{(IV-11)}$$

Ces sous-ensembles sont alors mis à jour successivement de sorte que, entre le temps t et le temps $t+k$ tous les sous-ensembles (et donc tous les neurones) sont mis à jour une et une seule fois. Ce mode « **asynchrone par sous-ensembles** » n'est que très rarement utilisé dans sa forme la plus générale (excepté pour les réseaux en couches dans lesquels les couches sont mises à jour successivement) et la plupart des réseaux actuels utilisent l'un de ses deux extrêmes :

- **Le mode synchrone.** C'est probablement le plus simple et le plus courant des modes de mise à jour actuellement utilisés dans les modèles connexionnistes. Dans ce cas le réseau ne comprend qu'un sous-ensemble de n neurones ($R = r_1$). Tous les neurones sont donc mis à jour simultanément en fonction de l'état du réseau au pas de temps précédent.

  Soit, si $\vec{x}(t) = \{x_1(t), x_2(t), ..., x_n(t)\}$ **(IV-12)** est le vecteur d'état du réseau au temps t , et si $\vec{x}(t+1) = F\left(\vec{x}(t), W, \vec{\theta}\right)$ **(IV-13)** est l'équation dynamique du réseau (où W représente la matrice de poids et $\vec{\theta}$ le n-vecteur des seuils) :

  $$\vec{x}(t+1) = \{x_1(t+1), x_2(t+1), ..., x_n(t+1)\} \qquad \textbf{(IV-14)}$$

  Pour chaque neurone l'équation dynamique peut alors s'écrire :

  $$x_i(t+1) = F\left(\vec{x}(t), W, \vec{\theta}\right) \qquad \textbf{(IV-15)}$$

- **Le mode asynchrone.** Ici le réseau est divisé en n sous-ensembles unitaires. A chaque pas de temps un et un seul neurone est donc mis à jour, l'ensemble du réseau devant être réactualisé en un temps fini. soit :

  $$\vec{x}(t+1) = \{x_1(t), x_2(t), ..., x_i(t+1), ..., x_n(t)\} \qquad \textbf{(IV-16)}$$

  Si le neurone i est mis à jour au temps t .

  Le choix du neurone mis à jour au temps t peut être établi à partir d'une séquence préétablie (mode **Asynchrone Séquentiel**) ou suivant un processus aléatoire (**Asynchrone Aléatoire**). Dans les deux cas, la séquence doit conduire tous les neurones à être mis à jour un même nombre de fois (le mode asynchrone respecte ainsi la contrainte imposée par l'équation IV-11).

IV.3.5.2. Stabilité et convergence des réseaux de neurones

Les réseaux « feed-forward » présentent une dynamique simple : il atteignent toujours un état stable en un temps limité car l'état de chaque neurone n'est évalué qu'une fois (lorsque les états de tous les neurones amonts ont été calculé). Par contre, dans les réseaux récurrents, l'existence de chemins d'activation infinis ne permet pas, dans le cas général, d'assurer la convergence du réseau. C'est à dire qu'il n'est pas possible, dans le cas général, d'assurer que le réseau sera stable après un nombre fini d'itérations (i.e. $\exists t' \ / \ \forall t > t' , \quad \vec{x}(t+1) = \vec{x}(t)$). L'exploitation de ce type de réseaux reste donc soumise à l'étude de leur comportement dynamique et, en particulier, à l'étude de leurs états attracteurs.

On distingue, en connexionnisme, trois types d'attracteurs :

- **Les attracteurs stables :** toutes les cellules ont atteint une valeur fixe et y reviennent lorsqu'on les en écarte faiblement. Le réseau est dit « stable », il a atteint un point fixe (la zone à partir de laquelle le réseau converge vers ce point fixe est son « bassin d'attraction ») : $\exists t' \ / \ \forall t > t' , \quad \vec{x}(t+1) = \vec{x}(t)$

- **Les attracteurs périodiques :** le réseau présente alors une dynamique cyclique : $\exists t' \ / \ \forall t > t' , \quad \vec{x}(t+T) = \vec{x}(t)$ si T est la période de l'attracteur,

- **Les attracteurs chaotiques.** Il s'agit du cas le plus complexe. Le réseau ne se stabilise pas mais il n'est pas possible de trouver de période finie. Ce type d'attracteur ne peut apparaître que lorsque le réseau comporte des cellules à valeur réelle ou des cellules dont la fonction de décision est stochastique (sans quoi l'espace d'état du réseau est fini, les trajectoires sont déterministes et les états chaotiques ne peuvent pas exister). L'étude des attracteurs chaotiques fait appel aux outils de la physique chaotique afin de déterminer leur enveloppe globale.

Dans la plupart des applications, les réseaux de neurones doivent associer une réponse unique à une ou plusieurs stimulations. Les attracteurs périodiques et chaotiques sont alors à éviter. C'est pourquoi les premières études de la dynamique des réseaux neuromimétiques ont été essentiellement tournées vers l'étude de la convergence des réseaux. Elles ont conduit Michael A. Cohen et Stephen Grossberg à énoncer le premier théorème de convergence :

Théorème de Cohen-Grossberg [COH 83] :

Un réseau récurrent est stable si les connexions sont symétriques et si la fonction de transfert des cellules du réseau est croissante, monotone et bornée.

Bart Kosko a étendu ce théorème aux phases d'apprentissage d'un réseau récurrent lorsque les poids sont modifiés suivant une règle hebbienne (théorème de Kosko-Cohen-Grossberg [KOS 88]).

L'utilisation des modèles connexionnistes n'est cependant pas limitée aux réseaux convergents. Les états périodiques ou chaotiques sont de plus en plus exploités voire même recherchés. Ainsi ils peuvent permettre de construire des séquences à partir d'un stimulus unique ou d'intégrer un aspect temporel dans la réponse du réseau [KLO 90]. D'autre part, ils peuvent parfois être analysés comme une réponse partielle du réseau à une information incomplète. Enfin une réponse partiellement stable peut souvent être suffisante. Ainsi, dans le cas des cartes cognitives, un état instable peut correspondre à une réponse partielle ou incomplète sur une partie des concepts tout en autorisant une autre partie à fournir une réponse stable.

IV.3.6. Apprentissage

Bien que la matrice de poids puisse parfois être calculée analytiquement à partir d'un ensemble de couples entrée/sortie (annexe A2.7), c'est leur capacité d'apprentissage par l'exemple qui fait l'intérêt principal des réseaux connexionnistes. Les différentes architectures proposées sont en effet généralement indissociables d'algorithmes d'apprentissage permettant de modifier le comportement du réseau (généralement par une modification de la matrice de poids, plus rarement par une modification de la structure du réseau) en fonction de différents critères (exemples de couples entrée/sortie, estimation de la qualité de la réponse, organisation/similitudes entre les exemples). On classe les différents algorithmes d'apprentissage en trois familles selon les critères utilisés pour l'entraînement du réseau et selon l'information fournie à l'algorithme (voir section IV.2.1)

IV.3.6.1. Apprentissage supervisé

C'est actuellement le mode d'apprentissage le plus couramment utilisé. Son principe est élémentaire : on soumet au réseau un grand nombre d'exemples pour lesquels l'entrée et la sortie associée sont connues et les poids sont modifiés de façon à corriger l'erreur commise par le réseau (c'est-à-dire la différence entre la sortie désirée et la réponse du réseau à l'entrée correspondante). Le plus répandu des algorithmes d'apprentissage supervisé est l'algorithme de rétropropagation du gradient d'erreur qui, appliqué aux réseaux en couches, utilise la dérivée de la fonction de transfert des neurones pour calculer l'erreur en amont d'une couche par rapport à l'erreur en aval [RUM 86b]. Bien que les principes théoriques de l'apprentissage non-supervisé soient antérieurs, l'apprentissage supervisé est, historiquement, le premier à avoir été utilisé pour l'entraînement des modèles neuromimétiques en 1957 avec le Perceptron [ROS 58] puis en 1960 avec l'ADALINE [WID 60] (annexe A2, section 1 et section 6).

IV.3.6.2. Apprentissage non-supervisé et auto-organisation

Contrairement aux modes supervisés, seule une base d'entrées est ici fournie au réseau. Celui-ci doit donc déterminer lui-même ses sorties en fonction des similarités détectées entre les différentes entrées, c'est-à-dire en fonction d'une règle d'auto-organisation. La règle la plus connue est celle définie par Donald O. Hebb dans son livre « The Organization of Behavior » :

« When an axon of a cell A is near enought to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency as one of the cells firing B is increased.[44] ***»***
[HEB 49], chapitre 4.

La règle de Hebb a donné naissance à de nombreux algorithmes d'apprentissage supervisé (comme dans les mémoires associatives où elle est utilisée pour apprendre des connexions reliant les neurones d'entrée aux neurones de sortie, annexe A2.2.) ou non- supervisés. Le plus simple d'entre eux est basé sur le renforcement des connexions entre les cellules d'activités corrélées (c'est-à-dire actives simultanément), soit, en utilisant le formalisme précédent,

$$\Delta W_{ij} = \alpha x_i x_j \qquad \textbf{(IV-17)}$$

Où α est une constante nommée « facteur d'apprentissage »

D'autres algorithmes dérivés de la règle de Hebb sont cités dans la littérature. Ainsi Harry Klopf et Bart Kosko ont proposé de renforcer les connexions conduisant à des changements d'activité corrélés (respectivement dans le « Drive-Reinforcement Learning » [KLO 86] et dans le « Differential Hebbian Learning » [KOS 86b], la différence entre les deux algorithmes réside essentiellement dans le calcul du facteur d'apprentissage), soit, si Δx_i est la variation de l'état du neurone i :

$$\Delta W_{ij} = \alpha \Delta x_i \Delta x_j \qquad \textbf{(IV-18)}$$

Un modèle intermédiaire a été développé par Cheung et Omidvar [CHE 88]. D'après ces auteurs un modèle basé sur la corrélation entre l'état des cellules activantes et la variation d'état des cellules activées s'approche plus des comportements d'apprentissage observés sur les cellules biologiques. Mathématiquement le modèle de Cheung est régulé par la valeur de la connexion. Il s'exprime alors comme suit :

$$\Delta W_{ij} = \alpha x_i W_{ij} \Delta x_j \qquad \textbf{(IV-19)}$$

Contrairement aux modèles de Hebb (équation (IV-17)) et de Klopf (IV-18), le modèle de Cheung ne conduit pas à une modification symétrique des connexions.

D'autres algorithmes d'apprentissage ont été proposés, généralement en association avec une topologie particulière. On citera en particulier les algorithmes de Kohonen pour les cartes

---

[44] ***Quand un axone de la cellule A est assez proche pour exciter une cellule B et quand, de façon répétitive et persistante, il participe à son activation, un certain processus de croissance ou un changement métabolique s'installe dans une cellule ou dans les deux tel que l'efficacité de A, en tant qu'elle est une des cellules qui active B, est augmentée.*** (traduction [PEL 95], chapitre 5, page 170)

auto-organisatrices [KOH 84] (annexe A2, section 4) et les modèles ART1 [CAR 86] et ART2 [CAR 87] développés par Carpenter et Grossberg (annexe A2, section 8).

IV.3.6.3. Apprentissage par renforcement

L'apprentissage par renforcement constitue un modèle intermédiaire entre l'auto-organisation et l'apprentissage supervisé. Dans de nombreux problèmes on ne dispose pas de l'information nécessaire à la construction d'une base d'apprentissage complète. On ne dispose souvent que d'une information qualitative permettant l'évaluation de la réponse calculée, sans pour autant connaître la réponse la plus adaptée. Les algorithmes d'apprentissage par renforcement, essentiellement développés depuis le début des années quatre-vingts, permettent alors d'entraîner le réseau pour qu'il fournisse, à chaque stimulus entrant, la sortie la plus adéquate. Cette forme d'apprentissage (aussi qualifiée d'apprentissage par pénalité/récompense - reward/penalty [WID 73] - ou par essai-erreur) est de plus en plus utilisée, en particulier pour les applications de neurocontrôle qui posent généralement des problèmes de « Credit-Assignment » ou « Temporal-Credit-Assignment » (voir sections IV.2.1 et IV.4.3).

L'apprentissage par renforcement a fait l'objet de nombreuses études en association avec les modèles neuromimétiques (en particulier dans les réseaux en couches [LAN 92]) ou avec des modèles symboliques pour lesquels des méthodes générales ont été mises au point (Q-Learning [WAT 92] ou TD-Learning [SUT 88]). Cependant son application aux réseaux récurrents pose encore de nombreux problèmes pratiques et théoriques. C'est pourquoi de nombreuses études portant sur ce type d'apprentissage sont encore en cours.

IV.3.6.4. Apprentissage « en ligne » et apprentissage « hors-ligne »

Suivant le mode d'apprentissage utilisé, suivant l'information disponible et les conditions propres à l'environnement d'apprentissage, le réseau peut être entraîné in situ. On parle alors d'apprentissage « en ligne » (on-line). C'est généralement le cas pour l'apprentissage par renforcement qui nécessite une réponse de l'environnement.

Dans le cas contraire (apprentissage « hors-ligne » ou « off-line »), le réseau est entraîné à partir d'une base d'apprentissage (généralement une base d'exemples) avant d'être appliqué au problème réel. On distingue alors les phases d'apprentissage des phases d'exploitation. Les apprentissages hors-ligne sont généralement plus sensibles au bruit ou à la dérive car ils ne peuvent pas suivre une évolution éventuelle du problème si celle-ci intervient en cours d'exploitation.

IV.3.7. Différents modèles de réseaux

La composition de l'ensemble des paramètres décrits au cours des sections précédentes (type de cellule, topologie, mode d'apprentissage, dynamique, ...) permet de créer de nombreux modèles de réseaux et d'utiliser, pour un problème donné, celui répondant le mieux aux besoins. La description des différents modèles de réseaux serait fastidieuse et n'entre pas dans le cadre de ce mémoire ; cependant on trouvera en annexe A2 (section 1 à 8) la description des modèles les plus courants.

IV.4. Architectures pour le neurocontrôle

Les modèles neuromimétiques ont été très largement utilisés pour le contrôle de processus (neurocontrôle). L'apprentissage par l'exemple permet en effet de contrôler des systèmes pour lesquels on ne dispose pas toujours de loi mathématique. Cependant les réseaux de neurones ne sont généralement pas capables de résoudre par eux-mêmes les problèmes de « Credit-Assignment » et de « Time-Credit-Assignment ». La suite de ce chapitre présente différentes méthodes d'apprentissage permettant de résoudre ces problèmes. Nous avons principalement porté notre attention sur les architectures permettant de résoudre le « Time-Credit-Assignment-Problem » car celui-ci est particulièrement important dans le cas de l'apprentissage des comportements d'un agent réactif.

IV.4.1. L'identification pour le contrôle de processus

Nous avons vu (section IV.3.4.2) que les réseaux en couches sont capables, pour peu qu'ils soient correctement dimensionnés et que les différents paramètres soient optimisés [FUN 89][HOR 89], d'identifier une fonction mathématique continue bornée avec une précision arbitraire. Cette propriété a conduit à développer, par extension, des applications d'identification de processus dynamiques et de les utiliser en neurocontrôle. Différentes utilisations peuvent être envisagées : l'identification de l'environnement (création d'un modèle neuronal), l'identification du contrôleur et l'identification de la dynamique inverse de l'environnement.

IV.4.1.1. Identification d'un processus

L'identification d'un processus dynamique ne permet pas d'obtenir directement le contrôleur associé mais l'utilisation d'un modèle de l'environnement est souvent nécessaire au développement d'un contrôleur, notamment pour l'apprentissage d'un contrôleur neuronal. C'est en particulier le cas lorsque, pour des raisons techniques ou financières, le système n'est pas accessible pendant toute la durée de l'apprentissage du contrôleur.

Pour ce type d'application, les intérêts des réseaux de neurones par rapport aux méthodes classiques (qui procèdent par ajustements successifs de paramètres sur un modèle mathématique) sont multiples. Ils permettent, en effet, d'identifier des processus non linéaires sans création préalable d'un modèle mathématique. De plus, si le réseau utilise des neurones à fonction de transfert dérivable, le modèle créé est globalement dérivable ce qui permet d'utiliser l'algorithme de la rétropropagation à travers le temps (section IV.4.2).

Considérons un système dynamique d'ordre n correspondant à l'équation dynamique suivante (on considère, dans un premier temps, que le système est déterministe) :

$$x(t+1) = f(t, x(t), x(t-1), \ldots, x(t-(n-1)), y(t), b(t))$$

L'observation du système permet d'obtenir un ensemble de couples $\left[x(t+1), (t, x(t), x(t-1), \ldots, x(t-(n-1)), y(t))\right]$ puis, en utilisant un apprentissage supervisé, d'entraîner le réseau à apprendre à associer les différents couples. Si le réseau est correctement dimensionné et si les exemples sont suffisamment représentatifs des espaces d'état et de commande, il peut apprendre l'ensemble des couples entrée/sortie et généraliser la dynamique du processus à l'ensemble du domaine concerné.

Si l'identification d'un processus dynamique bien connu (dimension, étendue, ordre,...) ne pose pas de problèmes majeurs, l'identification d'un environnement quelconque entraîne d'importantes difficultés, en particulier pour déterminer la topologie adéquate. C'est en particulier le cas lorsque l'ordre dynamique de l'environnement est inconnu a priori car cet ordre détermine le nombre d'états antérieurs devant être soumis, en entrée, au réseau. Sous-estimé, il risque de conduire à des états indéterminés (entrées identiques, sorties différentes) ; surestimé, il augmente considérablement la dimension du problème et donc le temps d'apprentissage. Diverses approches ont été utilisées pour résoudre ce problème. La plus prometteuse d'entre elles est probablement l'utilisation de modèles neuronaux récurrents qui permettent la mémorisation d'un contexte global (section IV.3.4.3).

IV.4.1.2. Identification d'un contrôleur, apprentissage de la commande motrice

L'apprentissage supervisé, défini sections IV.2.1 et IV.3.6.1, permet d'obtenir très simplement un contrôleur capable d'apprendre une commande motrice (ou une séquence motrice) pour peu qu'on dispose d'un ensemble de couples perception/action suffisamment important. Un réseau de neurones utilisé comme mémoire hétéro-associative permet alors d'associer à « chaque » point de l'espace d'entrée la commande optimale [HUN 92]. Les capacités de généralisation des réseaux neuromimétiques peuvent ensuite être mises à profit pour obtenir la commande sur les points ne figurant pas dans la base d'exemples.

Tous les réseaux à apprentissage supervisé sont utilisables pour apprendre une commande motrice. Cependant les réseaux en couches avec rétropropagation du gradient sont particulièrement bien adaptés à ce problème car leur dynamique simple leur permet de répondre en temps réel aux stimulations reçues. De plus les nombreux travaux ayant été menés avec ces réseaux permettent de disposer d'informations qualitatives (plus rarement quantitatives) quant à l'organisation de l'apprentissage. Toutefois, dans le cas d'un contrôle linéaire, l'utilisation d'un modèle multicouches ne s'impose pas et on préférera alors des modèles de type MADALINE (plusieurs ADALINE travaillant en parallèle sur une même entrée) ou des modèles classiques de type ARMA.

Le principal inconvénient du contrôle supervisé provient de la séparation des phases d'apprentissage (figure IV-11b) et d'exploitation (figure IV-11c) (apprentissage « hors-ligne »). En effet, dans un environnement non stationnaire, le contrôleur doit être régulièrement remis en cause ce qui entraîne un réapprentissage complet. Dans les cas les plus difficiles, il n'est même plus possible d'assurer l'intégrité de la base d'apprentissage au moment de l'apprentissage, le système ayant pu évoluer entre la construction de cette base (figure IV-11a) et son utilisation (figure IV-11b). La problématique de l'apprentissage du contrôle supervisée rejoint en cela le problème de l'identification d'un modèle non stationnaire. Or, ce problème n'a pas reçu, a l'heure actuelle, de réponse satisfaisante. En outre, la construction de la base d'apprentissage n'est pas toujours possible puisque le comportement du système n'est pas toujours parfaitement maîtrisé. Ce type de contrôleur est cependant particulièrement adapté au contrôle de la trajectoire d'un bras robot. On citera par exemple les travaux de L. Rabelo [RAB 91] [RAB 92] et les travaux de D. Pomerleau sur le SM2[45] [POM 93][NEW 93].

Plusieurs auteurs proposent d'utiliser ce type de modèle pour contrôler les déplacements d'un robot mobile [OPI 90a] [OPI 90b] [POM 93] [RIV 93] [BID 94]. On remarquera cependant

[45] Space Mobile Manipulator

que ce type de problème est fortement dépendant du contexte (une même entrée peut conduire à différents comportements en fonctions des états antérieurs ; ce problème est donc d'ordre supérieur à un). Dans ce cas, l'utilisation d'un réseau en couches est plus discutable puisque ce type de réseau doit recevoir en entrée autant d'états successifs qu'il est nécessaire pour différencier deux sorties différentes. Or, d'une part cela entraîne une augmentation importante de l'information à traiter (et de la dimension du problème), d'autre part l'ordre du problème est rarement connu a priori.

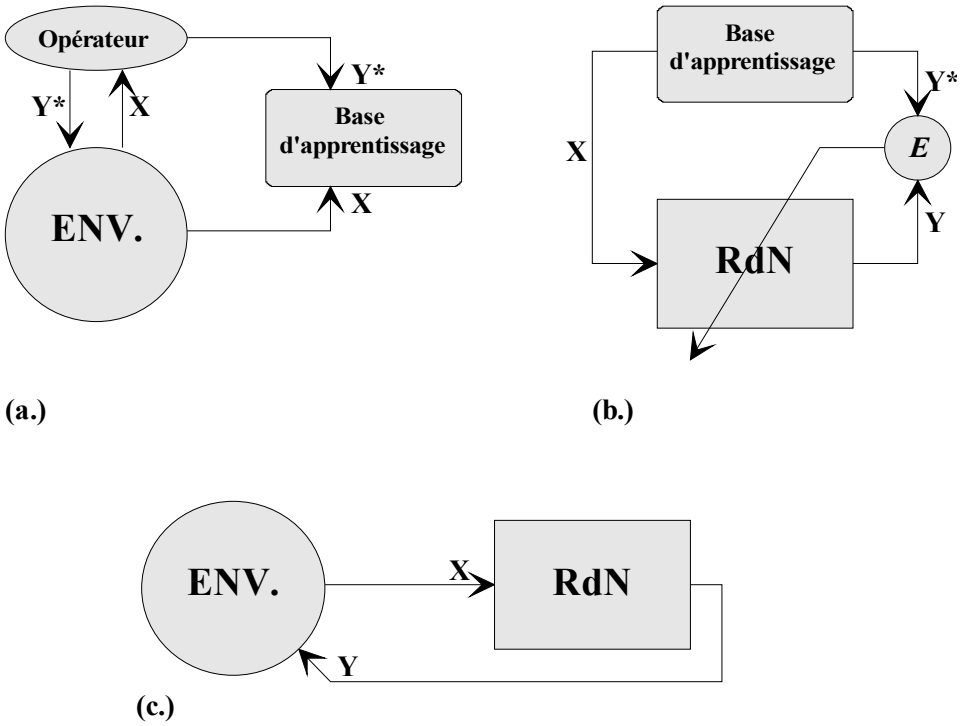


Figure IV-11 : *Apprentissage supervisé.*

L'apprentissage est réalisé grâce à une base construite à partir de l'utilisation réelle du système (a.). En phase d'apprentissage (b.) le réseau (RdN) apprend à associer les sorties désirées Y aux entrées X. En phase d'exploitation enfin (c.), il applique la commande apprise aux points contenus dans la base d'apprentissage et la généralise aux points intermédiaires. La séparation totale des phases (a.) et (c.) interdit généralement l'application de ce type d'apprentissage aux environnements non-stationnaires.*

L'intérêt de l'identification d'un contrôleur existant est assez limité si on utilise, pour construire la base d'apprentissage, un contrôleur existant (le contrôleur neuronal obtenu est forcément moins précis que le modèle initial). Cependant nous avons vu que les réseaux en couches permettent de garantir un temps de réponse relativement court, ce qui n'est généralement pas le cas des contrôleurs utilisant un modèle mathématique. Lorsque le temps de calcul est trop important, l'identification d'un contrôleur par un réseau de neurones peut permettre de contrôler le processus en temps réel.

De plus, dans le cas d'un système non-linéaire, les réseaux de neurones peuvent permettre d'intégrer en un unique contrôleur des modèles mathématiques linéaires ne couvrant qu'une partie de l'espace d'états. Ainsi Marc Schoenauer propose d'utiliser un réseau en couches pour diriger un mobile vers une cible (l'orientation à l'arrivée étant fixée, ce problème est comparable à garer une voiture dans un parking vide) [SCH 93]. Ne disposant pas d'une base de trajectoires, celle-ci est construite au moyen d'un algorithme génétique. On a donc un contrôleur génétique pour chaque origine et l'obtention d'une nouvelle trajectoire demande des temps de calcul incompatibles avec

un contrôle temps réel. L'utilisation du réseau de neurones permet alors de fusionner en un seul contrôleur les différentes trajectoires construites et de les généraliser en temps réel à de nouvelles positions initiales.

Par contre, lorsque le seul « contrôleur » utilisable est un expert humain, il est souvent difficile de formaliser la connaissance de cet expert pour construire un modèle mathématique du contrôle. Dans ce cas l'utilisation d'un réseau de neurones permet d'apprendre directement le contrôle approprié à chaque situation sans passer par une phase de formalisation autre que la constitution de la « leçon », c'est-à-dire de la base d'exemples utilisée pour l'apprentissage.

IV.4.1.3. Identification de la dynamique inverse

Considérant l'équation dynamique du système à contrôler X(t+1)=F(X(t),Y(t)) (où X(t) est l'environnement perçu au temps t, et Y(t) l'action entreprise, en considérant ici que le problème est d'ordre 1), on peut, si cette équation est inversible, utiliser un réseau de neurones pour faire correspondre les couples [X(t),X(t+1)] et Y(t). En phase d'apprentissage (le plus souvent hors-ligne mais l'apprentissage peut être réalisé en ligne), on applique une action Y(t) à partir de l'état X(t) ; l'état X(t+1) observé est utilisé pour entraîner le réseau (figure IV-12a). En phase d'exploitation, à partir d'une trajectoire [X(t),X(t+1)] souhaitée, le réseau permet de retrouver l'action à entreprendre (figure IV-12b). Cette approche présente l'avantage de la simplicité (la plupart des algorithmes d'apprentissage supervisés sont utilisables) et ne nécessite pas de construction préalable d'une base d'exemples [HUN 92]. Ce type de contrôleur suppose qu'on dispose d'un couple état initial/état désiré, ce qui n'est pas toujours le cas par exemple lorsqu'il s'agit de maintenir un système en équilibre.

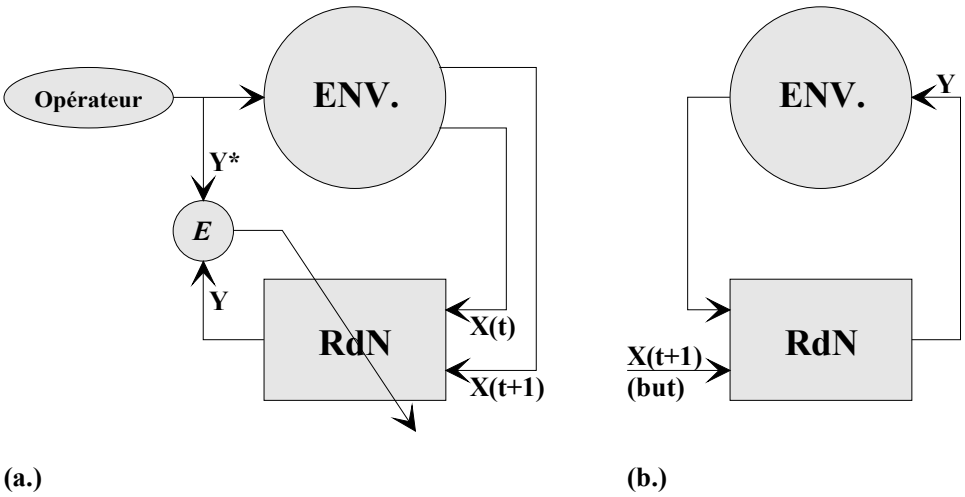


Figure IV-12 : *Apprentissage de la dynamique inverse pour un environnement d'ordre 1.*
(a.) : phase d'apprentissage (b.) : phase d'utilisation

Par contre l'apprentissage de la dynamique inverse pose un certain nombre de problèmes.

D'une part, lorsque l'équation dynamique F() n'est pas inversible (ce qui est en général le cas pour un bras robot car X(t+1)=F(X(t),Y1(t))=F(X(t),Y2(t))...), la méthode n'est pas applicable directement. Plusieurs solutions ont été proposées pour résoudre ce problème. Dans certains cas on peut se ramener à une fonction inversible en rajoutant des contraintes (minimiser les trajets, forcer un sens de parcours, ...).

D'autre part, avec cette architecture, les sessions d'apprentissage ne sont pas dirigées vers un but précis. Si la commande optimale n'est pas connue, au moins partiellement, il faut explorer totalement les espaces d'états et de commandes. L'apprentissage de la dynamique inverse n'est alors applicable qu'aux problèmes de faible dimension. En outre l'apprentissage dans les réseaux de neurones est soumis à une présentation rationnelle des différents exemples. Or les présentations risquent ici d'être anarchiques voire déséquilibrées (certains couples état/action étant statistiquement plus courants que d'autres) ce qui est inacceptable pour la plupart des modèles connexionnistes.

Ces inconvénients ont conduit plusieurs auteurs à modifier les principes de l'apprentissage de la dynamique inverse en appliquant à l'environnement la commande calculée par le réseau à partir de la trajectoire désirée. Le réseau est alors entraîné en fonction des trajectoires désirées (« Specialized-Inverse-Learning ») [PSA 88][JOR 92].

IV.4.1.4. « Specialized-Inverse-Learning »

L'architecture est organisée autour d'un contrôleur neuronal qui reçoit directement en entrée les couples « état » / « état désiré » (ce qui permet de forcer le réseau à produire un contrôle particulier). Ce réseau produit alors une action qui est appliquée parallèlement au système et à un deuxième réseau ayant préalablement été entraîné pour identifier le comportement du système. L'erreur utilisée pour entraîner le réseau est alors calculée en comparant les états désirés et les états obtenus (figure IV-13). Le modèle peut alors être utilisé pour calculer l'erreur lorsque le système physique est inutilisable ou bruité [HUN 92].

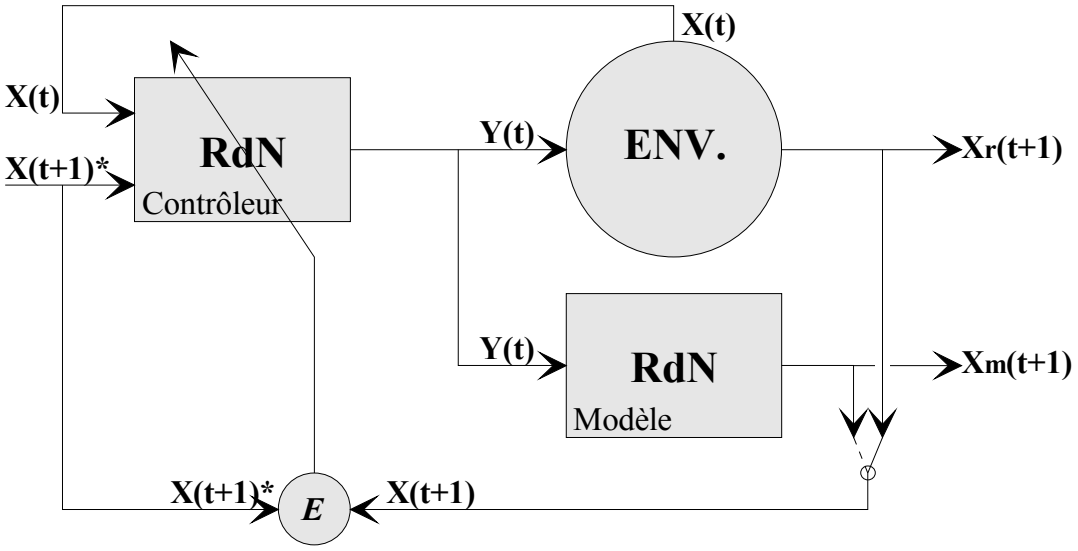


Figure IV-13 : « *Specialized-Inverse-Learning-Architecture »*

L'erreur utilisée pour l'apprentissage du contrôleur est calculée en comparant l'état final désiré X(t+1) avec l'état final X(t+1) obtenu après application de la commande Y(t) (c'est-à-dire, soit l'état $X_r(t+1)$ réellement obtenu après application de la commande, soit l'état futur prévu par le modèle : $X_m(t+1)$).*

L'apprentissage « spécialisé » permet de lever l'essentiel des inconvénients limitant l'apprentissage de la dynamique inverse. L'architecture utilisée autorise le contrôleur à explorer l'espace des commandes pour isoler celle conduisant à la meilleure concordance X(t+1)*/Xr(t+1). Ce modèle est donc à même de résoudre le « Credit-Assignment-Problem » (tandis que, pour le modèle général d'apprentissage de la dynamique inverse, l'opérateur seul dirigeait l'exploration). Par contre l'apprentissage spécialisé suppose que l'on dispose des états successifs par lesquels doit passer l'environnement [X(0), X(1), ... , X(t), ... , X(fin)]. Il ne permet pas, en général, de résoudre le « Time-Credit-Assignment-Problem ». Cependant cette limitation est plus due au choix des modèles neuronaux (réseaux en couches utilisant la rétropropagation du gradient) qu'à une contrainte réelle de l'architecture. Rien n'interdit, en effet, d'utiliser pour le contrôleur un réseau récurrent capable de générer des commandes motrices mais la comparaison X(t+1)*/Xr(t+1) ne permet pas de juger correctement les états intermédiaires. Nous allons présenter maintenant des architectures capables de résoudre ce problème en calculant, à chaque pas de temps, l'erreur commise par le réseau.

IV.4.1.5. L'identification comme complément des méthodes classiques

L'intérêt des réseaux de neurones pour le contrôle de processus dynamiques ne doit pas occulter celui des méthodes classiques d'identification par un modèle mathématique. En effet, ces méthodes ont elles aussi des avantages indéniables, en particulier en matière de sécurité d'évolution. Ainsi, pour des applications aérospatiales, les méthodes mathématiques classiques, qui sont les seules à pouvoir garantir le « zéro-défaut » (sur une faible partie de l'espace d'état), sont souvent préférées aux méthodes connexionnistes qui ne garantissent généralement qu'une « faible dégradation des performances lorsqu'on s'écarte des zones de contrôles optimales » [PEL 92].

Afin de préserver les intérêts respectifs des deux approches, plusieurs solutions ont été proposées. Nous détaillons ici les deux principales :

Afin de conserver la sécurité des méthodes mathématiques et les facultés d'apprentissage des réseaux de neurones, plusieurs auteurs proposent d'utiliser ceux-ci pour l'apprentissage des paramètres du modèle [COX 92]. Pour cela, on exprime l'erreur finale en fonction des paramètres concernés. Signalons cependant que cette approche, si elle permet de garantir la réponse du modèle (les réseaux de neurones ne sont ici qu'un outil de développement), n'offre pas plus de possibilités que les modèles classiques. En effet, ceux-ci sont avant tout limités par le modèle mathématique qui est généralement linéaire.

Une deuxième solution est d'allier en un seul contrôleur des modèles mathématiques et des modèles neuromimétiques suivant les phases de contrôle concernées. Ainsi on peut utiliser des modèles mathématiques linéaires garantissant le « zéro-défaut » sur une partie de l'espace d'états tandis que des réseaux de neurones permettent d'éviter une trop forte dégradation des performances au niveau des jonctions, c'est-à-dire dans les zones non-linéaires. Cette approche a l'avantage d'appliquer à chaque zone de l'espace d'états le type de contrôleur qui semble le plus adapté. Elle rejoint en cela les modèles de réseaux modulaires proposés par Robert A. Jacobs et Michael I. Jordan [JAC 93] [JOR 94].

IV.4.2. La modélisation adaptative et la rétropropagation à travers le temps

Nous avons vu (section IV.4.1.1) que, grâce à leurs capacités d'approximateurs universels, les réseaux de neurones, et en particulier les réseaux en couches, permettent d'identifier avec une

précision arbitraire le processus dynamique à contrôler. Par la suite, la possibilité d'utiliser un tel modèle pour entraîner le réseau a été envisagée (section IV.4.1.3). Or un tel modèle neuronal de l'environnement, s'il est formé de neurones réels à fonctions de transfert dérivables, permet d'obtenir l'erreur amont à partir de l'erreur aval. C'est-à-dire qu'il permet de calculer l'erreur au pas de temps précédent [WER 90]. Il suffit pour cela d'appliquer l'algorithme de la rétropropagation du gradient (sans modification des pondérations) à travers les différentes couches neuronales composant le modèle (figure IV-14).

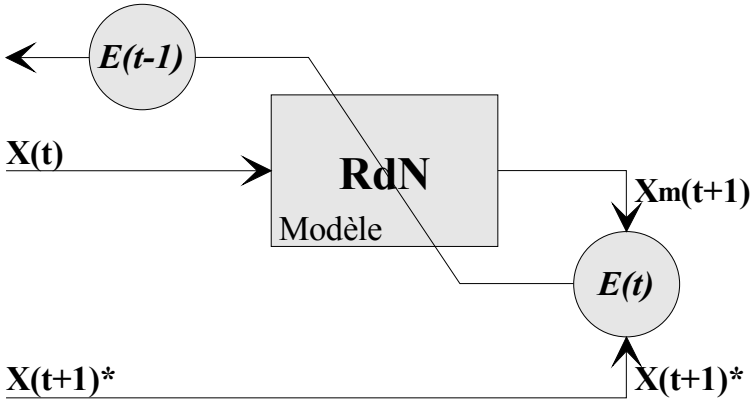


Figure IV-14 : *Rétropropagation du gradient à travers un modèle.*

L'erreur au temps t (différence entre l'état désiré X(t+1) et l'état obtenu Xm(t+1) est rétropropagée à travers le modèle ce qui permet d'obtenir l'erreur au temps t-1.*

Une architecture formée de deux réseaux de neurones à état réel et à fonction de transfert continue (un réseau modèle de l'environnement et un réseau contrôleur) peut donc s'affranchir des problèmes de « Time-Credit-Assignment » puisque le gradient d'erreur peut être obtenu à chaque pas de temps par rétropropagations successives dans le modèle et dans le contrôleur (figure IV-15). Des différentes architectures proposées jusqu'ici, la modélisation adaptative est donc la seule qui autorise la poursuite d'un objectif à long terme (maximiser/minimiser un ou plusieurs paramètres) [WER 89].

Initialement proposée par P.J. Werbos en 1974, la rétropropagation à travers le temps a bénéficié de l'essor des réseaux en couches qui a suivi les travaux de LeCun et Rumelhart sur la rétropropagation [LEC 85][RUM 86b]. Cette technique s'est alors rapidement popularisée et de nombreuses applications ont vu le jour, comme le « Truck-Backer-Upper » de N'Guyen et Widrow. Dans cette application, un réseau de type « MADALINE » est utilisé pour contrôler les mouvements d'un camion en phase d'approche d'un quai de déchargement [NGU 90].

La modélisation adaptative par rétropropagation à travers le temps présente de nombreux avantages : c'est une technique éprouvée utilisant des réseaux maintenant très classiques (réseaux en couches) et permettant une réponse en temps réel. Par contre, elle est fortement handicapée par la lenteur de l'apprentissage (du fait du nombre important de couches à traverser) et surtout par sa faiblesse face au bruit. Celui-ci interdit la création d'un modèle neuronal « parfait » pour rétropropager le gradient d'erreur et l'utilisation répétée d'un modèle approximatif risque d'entraîner une déviation importante dans le calcul de l'erreur.

Ces deux problèmes peuvent généralement être évités en limitant le nombre de rétropropagations (le nombre exact de rétropropagations nécessaires à l'apprentissage étant inconnu, on est souvent contraint à le surestimer). Plusieurs auteurs ont proposé des modifications

de l'algorithme initial, soit en adaptant le nombre de rétropropagations au cours de l'apprentissage, soit en le limitant arbitrairement. Les erreurs les plus faibles ne sont pas prises en compte et l'influence du bruit devient négligeable (les déviations dues au bruit sont essentiellement provoquées par les rétropropagations « profondes ».

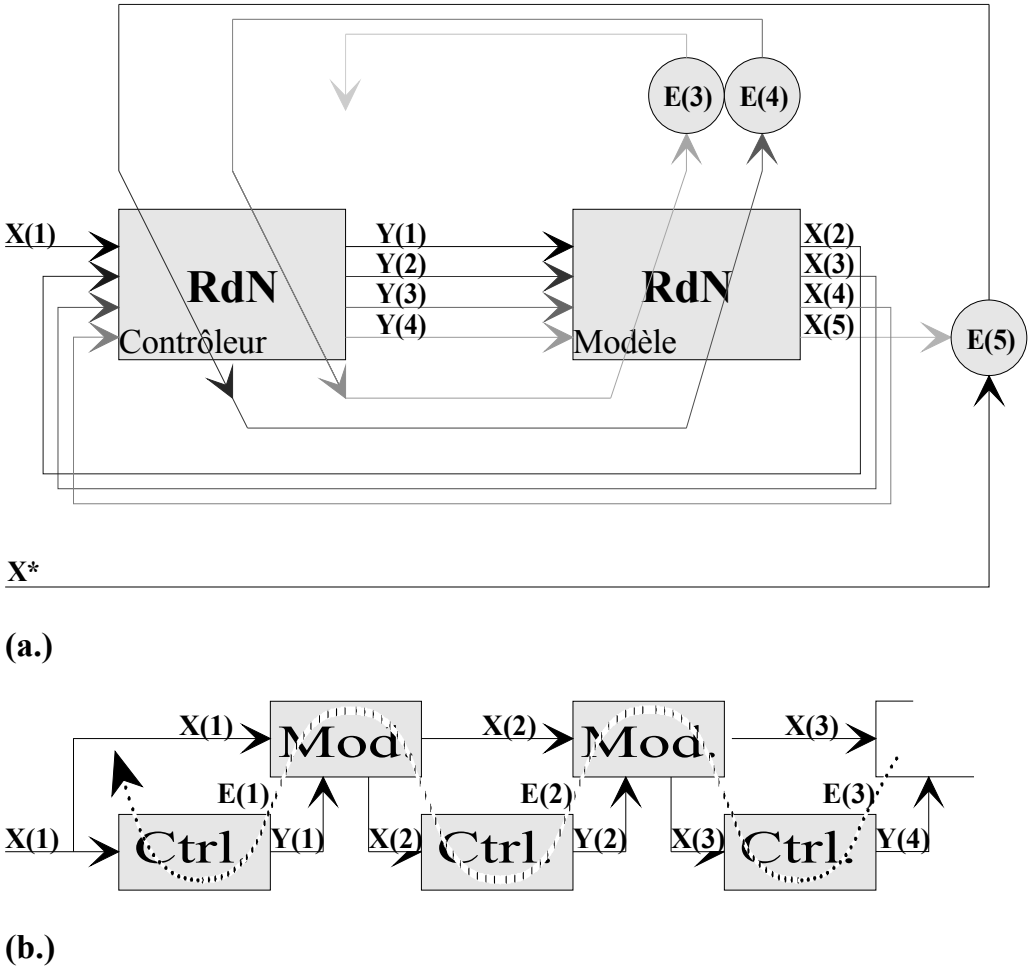


Figure IV-15 : *Architecture pour la modélisation adaptative.*

(a.) *Le modèle neuronal est entraîné directement grâce au système réel. Il est ensuite utilisé pour l'entraînement du contrôleur par rétropropagation à travers le temps : à partir de l'état X(1) le modèle permet d'estimer l'évolution du processus sur plusieurs pas de temps (X(2) à X(5)) en fonction des commandes déterminées, à chaque pas de temps, par le contrôleur (Y(1) à Y(4)). On peut alors calculer l'erreur finale E(5) et la rétropropager à travers le modèle et le contrôleur pour obtenir les erreurs intermédiaires E(4), E(3), ...*

(b.) *L'apprentissage est alors réalisé par rétropropagation à travers une succession de couches comme dans un réseau classique [NGU 90].*

IV.4.3. L'apprentissage par renforcement et la critique adaptative

Le neurocontrôle est intimement lié à l'apprentissage par renforcement. En effet, dans de nombreux cas il n'est pas possible de fournir au contrôleur la commande la plus adaptée mais on peut définir une fonction permettant d'estimer la qualité du contrôle effectué (par exemple la distance par rapport à une cible). Si le signal de renforcement est disponible à chaque pas de temps et s'il est relatif à l'action immédiatement antérieure, il est possible d'apprendre une

commande motrice sans utiliser d'architecture d'apprentissage particulière et le problème posé se rapproche de l'apprentissage supervisé[46] [SAE 91]. Par contre, lorsque le signal de renforcement n'est fourni qu'épisodiquement ou qu'il est relatif à une séquence d'actions, l'apprentissage se heurte au « Credit-Assignment-Problem » et au « Time-Credit-Assignment-Problem » (voir section IV.2) et la simple utilisation d'un algorithme d'apprentissage par renforcement ne suffit pas à entraîner un contrôleur.

Jusqu'à présent la modélisation adaptative est la seule architecture proposée capable de résoudre le « Time-Credit-Assignment-Problem » ; les autres modèles ne permettent pas d'utiliser l'erreur finale pour estimer l'erreur intermédiaire. Les réseaux de neurones sont cependant couramment utilisés pour des applications de prédiction : un algorithme d'apprentissage supervisé permet d'entraîner le réseau à prédire, à partir de l'état de l'environnement, l'évolution d'un paramètre particulier (voir, par exemple, [KOU 92] ou les applications de prévision de la consommation électrique développées pour EDF [DOR 92]).

Ainsi un réseau peut être utilisé pour fournir une estimation de l'erreur finale à partir de l'état de l'environnement. Une architecture comportant un réseau de contrôle et un réseau d'estimation de l'erreur (réseau « critique ») permet alors de contrôler un système dynamique complexe sans connaissance préalable du système. De plus, de toutes les architectures proposées ici, la « critique adaptative » est probablement la plus biologiquement plausible, ce qui renforce encore l'intérêt porté à cette méthode [WER 87a]. Nous présentons ici le modèle général de la critique adaptative avant d'évoquer deux modifications de l'algorithme destinées à améliorer le comportement d'exploration de ce modèle.

IV.4.3.1. Modèle général

L'architecture générale de la critique adaptative, proposée par Barto et *al.* en 1983 avec le modèle *Associative Search Element/Adaptive Critic Element (ASE/ACE)* [BAR 83] est composée de deux réseaux de neurones (figure IV-16). Le premier réseau, le contrôleur proprement dit, fournit à chaque pas de temps une commande. Le second estime en temps réel, à partir d'un signal de renforcement R(t) fourni périodiquement par l'environnement, la probabilité d'échec à chaque pas de temps r(t). L'évolution de cette probabilité permet d'entraîner parallèlement l'un et l'autre des réseaux. Initialement, ce principe était couplé à une représentation distribuée de l'environnement (chaque état est représenté par une valeur booléenne [MIC 68]) afin de linéariser des problèmes intrinsèquement non linéaires et donc d'utiliser des réseaux mono-couches. L'utilisation des réseaux multi-couches a permis de s'affranchir de cette discrétisation et de contrôler directement, par la modélisation adaptative, des systèmes dynamiques non linéaires [AND 89][LAN 92]

Le principe de la critique adaptative est indissociable des algorithmes d'apprentissage par renforcement (les signaux R et r(t) sont des estimations réelles de l'état de l'environnement, et non, comme pour la modélisation adaptative, des erreurs commises par le réseau contrôleur). C'est pourquoi plusieurs auteurs ont proposé des modèles de neurones orientés vers l'acquisition du réflexe pavlovien, c'est-à-dire l'apprentissage des réflexes conditionnés [GRO 87][KLO 92][ALK 89][KAN 88]. Dérivés de l'apprentissage hebbien, ces neurones ont le plus souvent été utilisés pour construire des réseaux très simples, voire limités à une seule cellule

---

[46] L'apprentissage par renforcement ne permet pas de diriger les actions à effectuer et le réseau doit être capable d'explorer l'espace des commandes pour obtenir un renforcement maximum et apprendre une commande optimale.

comme dans le cas de l'*ASE* et de l'*ACE* [BAR 83]. Cette approche représente un renouveau des principes matérialistes déjà évoqués mais elle entraîne de nombreux problèmes et ne permet généralement pas de contrôler des systèmes dynamiques non-linéaires. En effet, l'utilisation de l'apprentissage par renforcement dans des réseaux récurrents entraîne souvent des problèmes de dynamique complexes. De fait, la plus grande partie des applications de contrôle utilisant la critique adaptative développées jusqu'à présent est basée sur la rétropropagation du gradient, tant pour le réseau de contrôle que pour le réseau critique [AND 89][LAN 92].

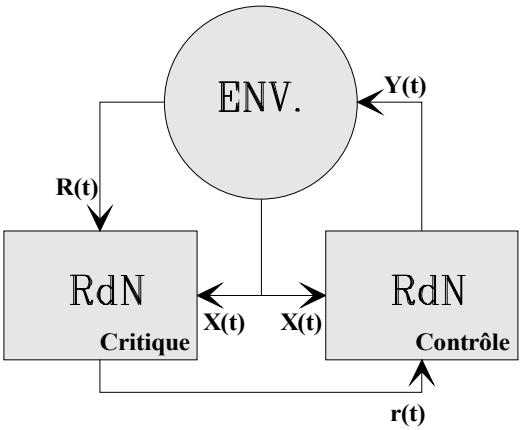


Figure IV-16 : *La critique adaptative*
Les phases d'apprentissage et d'exploitation sont confondues (apprentissage « en ligne »). L'évaluation de l'environnement conduit à la production d'un signal de renforcement « primaire » R(t). Un réseau « Critique » estime alors le risque d'échec r(t) ce qui permet au réseau « Contrôle » d'apprendre, par renforcement, la commande optimale.

Le modèle classique de la critique adaptative présente de nombreux intérêts. Sur le plan théorique, parce qu'il constitue, nous l'avons dit, le modèle le plus proche des principes neurobiologiques. Rappelons cependant que l'utilisation de réseaux d'inspiration neurobiologique a tendance à s'effacer au profit de réseaux dont la structure est biologiquement plausibles (les réseaux multicouches) mais dont les règles d'apprentissage font appel à des procédés globaux biologiquement peu probables. Sur le plan pratique, la critique adaptative présente aussi de nombreux intérêts : sa bonne résistance au bruit, ses capacités à apprendre « on-line » et sa souplesse d'utilisation (elle ne nécessite pratiquement aucune connaissance préalable du système à contrôler), font d'elle le système le plus adapté à contrôler des processus dynamiques dans des environnements bruités et non stationnaires, donc en particulier en milieu réel.

Malgré ses nombreux avantages, la critique adaptative est loin de constituer un modèle idéal pour le neurocontrôle. En effet, dans sa forme classique elle n'est applicable qu'à certaines classes particulières de processus.

Ainsi ce modèle ne résout aucunement les problèmes posés par le contrôle de systèmes dynamiques d'ordre supérieur à un. Pour ces processus les contraintes sont ici les mêmes que pour la modélisation adaptative, du moins tant que les réseaux utilisés sont des réseaux en couches. Par contre, comme nous l'avons dit, les capacités d'apprentissage on-line des modèles utilisant la critique adaptative leur permettent de contrôler des processus dans des environnements non-stationnaires.

En outre, ce type de structure pose de nombreux problèmes quant à la génération de la commande motrice adaptée à chaque état particulier. En effet, la critique adaptative repose sur un principe d'exploration de l'espace des commandes. Cela sous-entend que, lorsqu'une commande

s'est révélée inadaptée, le contrôleur doit en générer une autre, jusqu'à trouver la commande idéale ou la séquence idéale. Dans les modèles classiques, l'exploration est produite par l'initialisation aléatoire des poids du réseau. Cela rend le système dépendant de cette initialisation et risque de dégrader les performances, surtout lorsque l'espace des actions est important. Pour pallier à ce problème, plusieurs auteurs ont adjoint au système un processus de décision stochastique. Ainsi, dans le premier modèle de Barto, Sutton et Anderson [BAR 83], un processus stochastique permet de choisir l'action en fonction de la valeur réelle de l'élément de contrôle ce qui favorise l'exploration de l'espace des commandes.

IV.4.3.2. « Stochastic-Real-Valued-Control »

Le problème de l'exploration de l'espace des commandes devient plus crucial encore lorsque le contrôleur doit déterminer ses actions au moyen de valeurs réelles. Dans ce cas, en effet, le contrôleur doit explorer l'espace de recherche jusqu'à obtenir la valeur optimale. Il doit ensuite stopper l'exploration pour ne pas dégrader le contrôle obtenu. Or l'apprentissage par renforcement utilisé pour la critique adaptative ne permet pas, dans ses formes les plus courantes, d'arrêter l'exploration. Dès 1975, en utilisant des algorithmes d'apprentissage par renforcement pour le modèle CMAC, Albus s'était heurté à ce problème sans parvenir à le résoudre [ALB 75].

L'origine des problèmes de saturation est le signal de renforcement : lorsque le système a appris le contrôle optimal, il continue de recevoir le signal de renforcement. De plus celui-ci a alors une forte tendance positive (puisque les actions entreprises sont optimales). Il provoque donc une dérive progressive vers un contrôle *Bang-Bang*.

Vijaykumar Gullapalli [GUL 90] et Kazushige Saga [SAG 90] ont simultanément proposé une approche stochastique pour résoudre les problèmes de saturation et d'exploration. Similaires dans le principe, ces approches diffèrent radicalement par leur mise en oeuvre.

Le « Stochastic-Real-Valued-Control[47] » repose sur un calcul probabiliste du contrôle appliqué au système. Les valeurs réelles sont donc calculées au moyen d'une fonction aléatoire dont les paramètres (moyenne et variance) sont ajustés en fonction de la réponse de l'environnement. Ce principe était déjà en partie utilisé auparavant mais, jusqu'aux travaux de Gullapalli et Saga, les algorithmes d'apprentissage agissaient exclusivement sur la moyenne sans réduire la variance au cours de l'apprentissage. Ainsi, l'architecture ACE/ASE utilisait déjà une fonction stochastique pour déterminer le signe de la force appliquée au chariot supportant le pendule.

[47] Contrôle Stochastique à Valeur Réelle

En utilisant ce principe, Gullapalli propose des Unités Stochastiques à Valeur Réelle (SRV-Units[48]) disposant, en entrée, de deux séries de poids (w_i et v_i) utilisés respectivement pour calculer, à partir du vecteur d'entrée $x(t)$, la moyenne $\mu(t)$ et la variance $\sigma(t)$ de l'état de la cellule (figure IV-17, équations IV-20 et IV-21). En pratique, les unités calculent la moyenne et le signal de renforcement estimé $\hat{r}(t)$, la variance est ensuite calculée, à partir de $\hat{r}(t)$, au moyen d'une fonction $s()$ (équation IV-22).

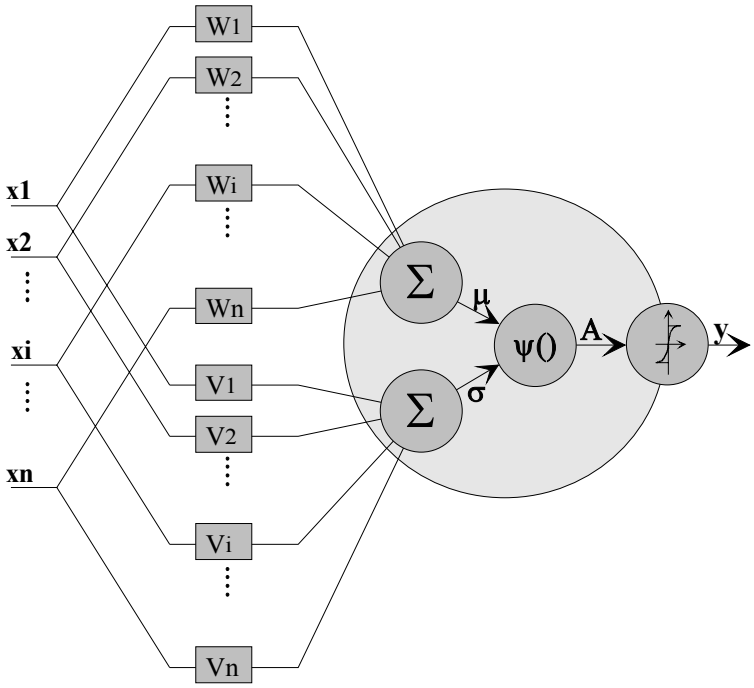


Figure IV-17 : *Stochastic Real Valued Unit*

Le neurones possèdes deux séries de poids : $W_{1..n}$ pour le calcul de l'état moyen μ (équation IV-20) et $V_{1..n}$ pour le calcul du signal de renforcement (équation IV-21) puis de la variance σ (équation IV-22). μ et σ sont ensuite utilisés pour calculer l'état de la cellule par la loi normale Ψ (équation IV-23).

Les lois d'activation de cette unité sont identiques à celles utilisées pour les unités réelles classiques (cf. équation IV-20) :

$$\mu(t) = \sum_{i=1}^{n} w_i(t) x_i(t) \qquad \textbf{(IV-20)}$$

$$\hat{r}(t) = \sum_{i=1}^{n} v_i(t) x_i(t) \qquad \textbf{(IV-21)}$$

La variance est ensuite calculée au moyen de l'équation suivante (IV-22) :

$$\sigma(t) = s(\hat{r}(t)) = \max\left[\left(\frac{1-\hat{r}(t)}{5}\right), 0\right] \qquad \textbf{(IV-22)}$$

[48] Stochastic Real Valued Units

L'état de l'unité est alors déterminé grâce à une loi normale $\psi()$:

$$A(t) = \psi(\mu(t), \sigma(t)) \qquad \textbf{(IV-23)}$$

Le principal apport des travaux de Gullapalli réside dans la fonction d'apprentissage utilisée. En effet, si les fonctions d'activation permettent effectivement de favoriser l'exploration de l'espace des commandes, elles ne permettent pas d'éviter le phénomène de saturation. Pour cela on utilise un terme $\Delta_w(t)$ fonction des différents paramètres :

$$\Delta_w(t) = (r(t) - \hat{r}(t))\left(\frac{A(t) - \mu(t)}{\sigma(t)}\right) \qquad \textbf{(IV-24)}$$

Ce terme (le « bruit normalisé ») n'autorise l'apprentissage que lorsque le bruit appliqué a provoqué une réponse positive de l'environnement. Il est utilisé dans l'équation (IV-25) pour calculer les nouvelles pondérations grâce à une loi hebbienne :

$$w_i(t+1) = w_i(t) + \alpha\Delta_w(t)x_i(t) \qquad \textbf{(IV-25)}$$

Parallèlement les poids v_i sont modifiés grâce à un apprentissage supervisé de type Widrow-Hoff (règle Delta) [WID 60] pour que chacune des unités puisse calculer un signal de renforcement local (équations (IV-26) et (IV-27)) :

$$\Delta_v(t) = r(t) - \hat{r}(t) \qquad \textbf{(IV-26)}$$

$$v_i(t+1) = v_i(t) + \beta\Delta_v(t)x_i(t) \qquad \textbf{(IV-27)}$$

L'auteur propose trois séries de test :

- des unités simples sont utilisées pour l'identification de fonctions booléennes simples (ET logique).
- Un réseau mixant SRV-Units et neurones « classiques » pour l'apprentissage des fonctions ET et OU EXclusif.
- Apprentissage de la commande d'un bras robot au moyen d'un réseau en couches dont les unités de sortie sont de type SRV-Units.

En outre, un « réseau » composé d'une unité SRV a été utilisé par Mark Sheppard pour le contrôle du pendule inverse[49]. D'après l'auteur, les résultats sont supérieurs à ceux obtenus avec des méthodes classiques. Cependant, comme nous le verrons dans la section consacrée au pendule inverse (VI.4.6), de nombreuses imprécisions nous interdisent d'exploiter directement ces résultats.

---

[49] Sheppard, M., Oswald, A., Valenzuela, C., Sullivan, G., Sotudeh, R. - « Reinforcement Learning in Control » - *Rapport interne de l'université de Teeside, Middlesbrough, Cleveland, TSA 3BA, UK* - 10 pages - Non daté - Non référencé

L'algorithme proposé par Gullapalli, en dehors de son indéniable intérêt, pose un certain nombre de problèmes. Le plus important d'entre eux est l'utilisation d'un signal de renforcement primaire $r(t)$ à valeur réelle, qui plus est fourni à chaque pas de temps. Les unités SRV ne résolvent donc pas le problème du « Time-Credit-Assignment ». Cependant rien n'interdit d'utiliser ces cellules pour construire un contrôleur dans une structure de type « critique adaptative » mais cette possibilité n'a pas été testée. De plus, l'utilisation d'un signal de renforcement global (non spécifique aux unités) provoque, lorsque les unités SRV sont utilisées en réseau, des problèmes de stabilité. C'est pourquoi, dans son application de contrôle d'un bras robot, l'auteur utilise un réseau en couches dont seules les unités de sortie sont de type « SRV-Units ». Les couches d'entrée sont formées d'unités à valeur réelle entraînées par un algorithme de rétropropagation du gradient. Dans ce cas, seule la sortie du réseau est calculée au moyen d'une fonction stochastique.

Cette architecture est donc en partie opposée au principe initial suivi par Gullapalli (utiliser des fonctions stochastiques locales). Elle est, par contre, beaucoup plus proche de celle présentée dans [SAG 90].

A l'inverse de Gullapalli, Saga propose d'utiliser deux réseaux distincts. Le premier permet de calculer la moyenne (réponse idéale) tandis que le second calcule la variance. Un générateur de bruit aléatoire, dépendant de la variance calculée, est ensuite utilisé pour calculer la réponse finale du système. L'auteur considère ce type d'architecture comme auto-supervisé[50] : en utilisant un mécanisme stochastique le système génère ses propres réponses (auto-organisation, auto-apprentissage) mais il utilise les mécanismes de l'apprentissage supervisé (avec un facteur d'apprentissage variable, le gain, calculé par le réseau de prédiction) pour entraîner le réseau de contrôle (figure IV-18).

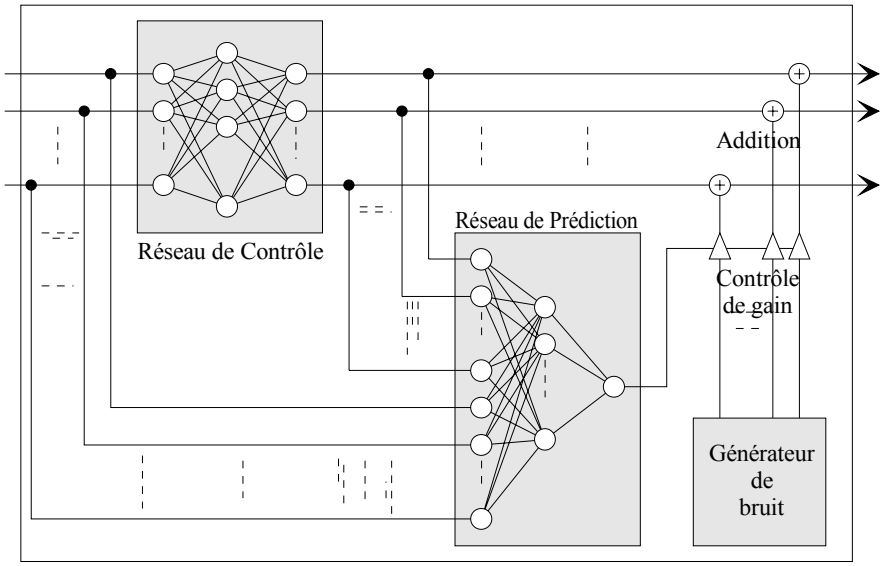


Figure IV-18 : *Apprentissage « auto-supervisé »*

L'architecture « Auto-supervisée » est intéressante parce qu'elle apporte des solutions au problème de l'exploration de l'espace des commandes par un réseau de neurones. Malheureusement, comme les SRV-Units de Gullapalli, ce modèle ne résout pas le « Time-Credit-Assignment-Problem » : le réseau de prédiction apprend grâce à un apprentissage supervisé

[50] Self-Supervised Learning

classique, le gain correspondant à chaque couple entrée-sortie du contrôleur. Dans ce contexte l'utilisation d'une architecture complexe ne se justifie pas totalement. En outre l'application utilisée par l'auteur pour illustrer son algorithme (atteindre une cible visuelle dans une pièce vide) ne montre pas clairement l'intérêt de la méthode : Valentino Braitenberg, dans son étude « vehicles » [BRA 84] [DEW 87] propose des automates correspondant à ces spécifications en reliant simplement les capteurs et les actionneurs.

IV.4.3.3. Critique et modélisation adaptative

Lorsque la dimension du problème est importante, il est difficile d'utiliser uniquement un signal de renforcement réel qui ne permet qu'une recherche statistique de la solution optimale. L'utilisation d'une architecture plus complexe incluant un modèle neuronal doit alors être envisagée. En 1989, P.J. Werbos [WER 89] a proposé un modèle comportant trois réseaux utilisant simultanément la critique adaptative et la modélisation adaptative. Nous n'avons pas trouvé, dans la littérature, d'applications pratiques utilisant cette architecture. Il est donc difficile d'en connaître les capacités exactes. Nous la citons cependant car elle offre une solution intéressante aux problèmes d'exploration de l'espace (« Curse of dimensionnality problem »).

Le but de cette architecture est de fournir au contrôleur une évaluation plus précise que le simple renforcement secondaire (ou primaire). Pour cela le contrôleur est remplacé par un ensemble contrôleur/modèle semblable à celui présenté section IV.4.2. Le modèle est utilisé pour apprendre les associations entre le renforcement secondaire (et l'état de l'environnement) et les différents vecteurs de sortie du contrôleur. Cette architecture permet donc au contrôleur de corriger plus spécifiquement ses poids et autorise une exploration dirigée de l'espace des commandes.

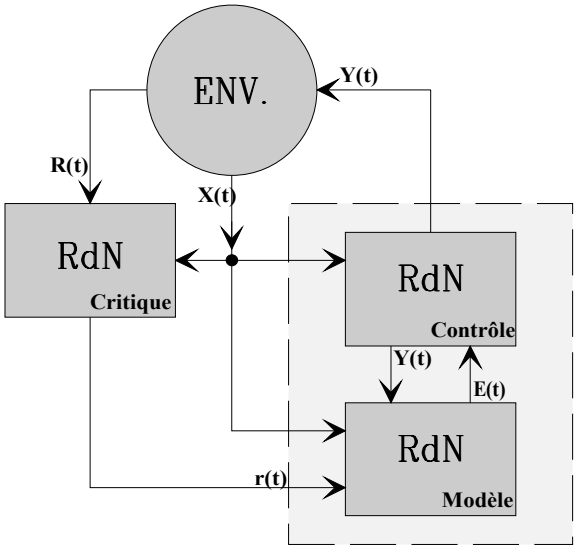


Figure IV-19 : *Critique adaptative à travers un modèle*

Le modèle est utilisé pour produire, à partir du signal de renforcement r(t), une estimation de l'erreur E(t) commise par le réseau de contrôle.

IV.4.3.4. L'adaptation de la critique : TD-Learning

Le principe même de la critique adaptative suppose qu'on soit capable d'estimer le signal de renforcement secondaire, c'est-à-dire de prédire le signal de renforcement final. La méthode de prédiction la plus générale est la méthode des différences temporelles (« Temporal-Difference-Learning » ou « TD-Learning ») [SUT 88]. Elle permet de prévoir l'évolution d'un système à long terme et de donner, à chaque étape $t_0, t_1, t_2, \cdots, t_m$ d'une séquence d'états, une prédiction concernant l'arrivée future, au temps $t_m + 1$, d'un événement z (représenté par un scalaire). Dans le cas de la critique adaptative, l'événement z est le signal de renforcement primaire, externe, et les prédictions sont les signaux de renforcement secondaires, internes. Les prédictions sont notées $P_1, P_2, P_3, \cdots, P_m$ et les états successifs de la séquence $\vec{x}(0), \vec{x}(1), \vec{x}(2), \cdots, \vec{x}(m)$, où $\vec{x}(t) = x_1(t), x_2(t), \cdots, x_n(t)$ est un n-vecteur réel.

Il est possible d'estimer les prédictions au moyen d'un algorithme d'apprentissage supervisé en utilisant, à chaque pas de temps, l'erreur quadratique donnée par $\frac{1}{2}(P_t - z)^2$. En utilisant un réseau à deux couches et la loi d'apprentissage de Widrow-Hoff (voir annexe A2, section 1) on a, si $f(\ )$ est la fonction sigmoïde :

$$P_t = f\left(\sum_{i=1}^{n} x_i(t) W_i\right) \qquad \textbf{(IV-28)}$$

La loi d'apprentissage donne alors, pour l'élément $x(t)$:

$$\Delta W_{t \to t+1} = \alpha(z - P_t) \frac{\partial P_t}{\partial W(t)} \qquad \textbf{(IV-29)}$$

soit, pour l'ensemble de la séquence :

$$\Delta W_{t_0 \to t_m} = \alpha \sum_{t=t_0}^{t_m} (z - P_t) \frac{\partial P_t}{\partial W(t)} \qquad \textbf{(IV-30)}$$

Cependant cette méthode impose d'attendre l'arrivée de l'événement final pour corriger les différentes valeurs de la prédiction. En outre, cette approche ne permet pas d'affiner progressivement les valeurs de la prédiction lorsqu'on se rapproche de la fin de la séquence.

Le principe fondamental du TD-Learning est de présenter l'erreur de prédiction commise au temps t, $z - P_t$, comme une somme de variations de prédiction :

$$z - P_t = \sum_{k=t}^{m} (P_{k+1} - P_k) \qquad \textbf{(IV-31)}$$

avec, par définition : $P_{m+1} = z$

On a alors, en combinant les équations (IV-30) et (IV-31) :

$$\Delta W_{t_0 \to t_m} = \alpha \sum_{t=t_0}^{t_m} \sum_{k=t}^{t_m} \left(P_{k+1} - P_k\right) \frac{\partial P_t}{\partial W(t)}$$

$$\Delta W_{t_0 \to t_m} = \alpha \sum_{k=1}^{t_m} \sum_{t=t_0}^{k} \left(P_{k+1} - P_k\right) \frac{\partial P_t}{\partial W(t)}$$

$$\Delta W_{t_0 \to t_m} = \alpha \sum_{k=1}^{t_m} \left(P_{k+1} - P_k\right) \sum_{t=t_0}^{k} \frac{\partial P_t}{\partial W(t)}$$

A chaque instant t , on peut donc calculer la contribution élémentaire d'un élément de la séquence à l'apprentissage :

$$\Delta W_{t \to t+1} = \alpha \left(P_{t+1} - P_t\right) \sum_{k=t_0}^{t} \frac{\partial P_k}{\partial W(k)} \qquad \textbf{(IV-32)}$$

L'utilisation des différences temporelles permet donc d'apprendre incrémentalement à prédire l'événement z au fur et à mesure du déroulement de la séquence $\vec{x}(0), \vec{x}(1), \vec{x}(2), \cdots, \vec{x}(m)$.

Selon l'équation (IV-32), chaque élément de la séquence contribue identiquement à l'apprentissage. Richard Sutton propose de modifier cette équation en lui joignant un terme de décroissance λ $(0 \le \lambda \le 1)$ qui favorise l'apprentissage des éléments les plus récents par rapport aux éléments anciens (équation IV-33) :

$$\Delta W_{t \to t+1} = \alpha \left(P_{t+1} - P_t\right) \sum_{k=t_0}^{t} \lambda^{t-k} \frac{\partial P_k}{\partial W(k)} \qquad \textbf{(IV-33)}$$

L'algorithme de prédiction correspondant est nommé TD(λ). TD(1) est équivalent à la loi de Widrow-Hoff appliquée à la prédiction (équations IV-30 et IV-32). Richard Sutton montre, sur deux exemples, que TD(0) converge plus rapidement que TD(1). Par ailleurs, cet auteur a étendu la famille d'algorithmes TD(λ) à la prédiction de valeurs cumulées (dans ce cas on cherche à prédire le cumul d'une valeur mesurée à chaque pas de temps), et aux environnements non-linéaires en utilisant, à la place de la règle de Widrow-Hoff, la rétropropagation du gradient dans un réseau multi-couches.

IV.5. Conclusion : quels modèles de contrôleurs pour le contrôle réactif

Nous avons proposé, au terme du chapitre III, le cahier des charges d'un algorithme d'apprentissage pour le « Behavior-Based-Control » et nous avons énuméré plusieurs conditions essentielles :

- Apprentissage par renforcement,
- Modularité/Parallélisme,
- Limitation de l'utilisation de modèles,
- Action sensori-motrice (« situated action »),
- Apprentissage « on-line ».

Afin de proposer l'architecture connexionniste la plus proche, structurellement, des principes du contrôle réactif, nous allons étudier, pour chacun des modèles de contrôleur présenté et pour les modèles connexionnistes les plus courants, les contradictions essentielles (ou, au contraire, les corrélations) avec les contraintes pré-citées. Dans un premier temps, le tableau IV-1 met en relation les différentes architectures de contrôle proposées avec les trois premiers critères définis ci-dessus. En effet, les deux derniers critères dépendent directement du modèle de réseau utilisé et non de l'architecture dans laquelle il s'insère. Le tableau IV-2 présentera ensuite les principaux modèles de réseaux de neurones sous l'angle du « Behavior-Based-Control ».

	Identification d'un contrôleur (IV.4.1.2)	Identification de la dynamique inverse (IV.4.1.3)	« Specialized-inverse-learning » (IV.4.1.4)	Modélisation adaptative (IV.4.2)	Critique adaptative (IV.4.3)
Apprentissage par renforcement	Eventuellement (peu adapté)	NON	NON	**OUI**	**OUI**
Apprentissage en ligne (on-line)	**OUI**	NON	NON	NON	**OUI**
Utilisation d'un modèle	**NON**	OUI	OUI	OUI	**NON**

Tableau IV-1 : *Correspondance architecture neuronale/architecture réactive*

Le tableau IV-1 nous montre de façon évidente que seuls l'identification directe d'un contrôleur (c'est-à-dire l'apprentissage par l'exemple) et l'apprentissage par critique adaptative souscrivent aux conditions imposées par le « Behavior-Based-Control ». Toutefois, l'identification de la commande motrice privilégie l'utilisation d'un apprentissage supervisé. Avec un apprentissage par renforcement, cette architecture est similaire au module « contrôle » d'une architecture de critique adaptative. La différence entre ces deux modèles n'est alors plus que l'aptitude à résoudre les problèmes de « Time-Credit-Assignment » par l'adjonction du réseau de critique.

Si le choix de l'architecture générale ne pose pas de problèmes majeurs, il n'en est pas de même pour le choix du (des) modèles de réseaux. Le tableau IV-2 propose une mise en correspondance des différents types de réseaux présentés dans le tableau synoptique de la section IV.3.7 et les caractéristiques imposées pour l'algorithme d'apprentissage.

	Apprentissage par renforcement	Apprentissage en ligne	Utilisation d'un modèle	Modularité	Architecture Sensori-motrice
Perceptron, ADALINE (A2.1)	**Possible**	**OUI**	**Selon l'architecture**	NON	OUI à l'ordre 1
Mémoires associatives, apprentissage par la règle de Hebb (A2.2)	**Possible pour l'adaptation**	**Possible pour l'adaptation**	Partielle (clés identifiées)	**Partielle (clés hiérarchisées)**	**OUI**
Réseaux de Hopfield, apprentissage par la règle de Hebb (A2.3)	NON	NON	**Selon l'architecture**	NON	**OUI**
Cartes topologiques (A2.4)	NON (auto-organisation)	**OUI**	**Selon l'architecture**	NON	OUI à l'ordre 1
Cartes cognitives (A2.5)	**Possible pour l'adaptation**	**Possible pour l'adaptation**	OUI	**OUI**	NON
Réseaux en couche, apprentissage par rétropropagation. (A2.6)	**Possible (peu adapté)**	**Possible (peu adapté)**	**Selon l'architecture**	NON	OUI à l'ordre 1
Réseaux en couches type RBF-Nets (A2.7)	NON	NON	OUI[51]	**Partielle**[52]	OUI à l'ordre 1
Adaptive Resonance (annexe A2.8)	NON (auto-organisation)	**OUI**	**Selon l'architecture**	NON	**OUI**

Tableau IV-2 : *Correspondance modèle de réseau/architecture réactive*

D'après le tableau IV-2, aucun modèle de réseau ne répond totalement au profil recherché, deux raisons principales :

- Les réseaux en couches, et, d'une manière générale, les réseaux « feed-forward » (non-récurrents), ne permettent pas d'inclure dans leur fonction de décision, le contexte (c'est-à-dire les stimuli antérieurs) ce qui restreint fortement leur fonction de décision. Seuls les réseaux récurrents permettent de mémoriser de façon non explicite les états antérieurs (les mémorisations explicites, qui permettent d'étendre les possibilités des réseaux en couches, sont en contradiction avec les principes du contrôle réactif car elles imposent une discrétisation très rigide de l'écoulement de temps).

- La plupart des réseaux neuromimétiques sont totalement incompatibles avec les principes de modularité. Seuls les cartes cognitives et, dans une moindre mesure, les mémoires associatives et les RBF-Nets autorisent une décomposition modulaire (voir [RON 95], [BIE 92a] et [COC 88]). Par contre, les cartes cognitives et les mémoires associatives ne permettent pas une interaction sensori-motrice directe avec l'environnement : ils utilisent

[51] La disposition des noyaux dans l'espace d'entrée peut être considérée comme un modèle partiel de l'environnement puisque la disposition des noyaux est fonction de la répartition des exemples d'entrée (voir annexe A2, section 7).

[52] Le découpage de l'espace d'entrée en zones allouées à des cellules cachées particulières peut être considéré comme une forme de modularité. Celle-ci reste cependant limitée à cet aspect topologique.

une représentation symbolique (un neurone/un concept ou « clé ») qui interdit les interactions directes avec l'environnement. Les RBF-Nets, pour leur part, sont des réseaux « Feed-Forward » et ne permettent pas de mémoriser le contexte de l'action. Ces réseaux sont d'ailleurs essentiellement adaptés aux applications de classification.

Cette analyse, proche de celle qui a conduit Rodney Brooks à rejeter totalement les modèles connexionnistes, nous mène à la conclusion suivante : si aucun réseau n'est totalement adapté pour une utilisation en contrôle réactif, ils offrent, séparément, des solutions intéressantes (modèles récurrents, modèles symboliques, apprentissage par renforcement, critique adaptative, ...). De plus, de même que l'éthologie a servi de base conceptuelle au « Behavior-Based-Control », la neuroéthologie nous prouve la compatibilité entre le phénomène observé (multi-comportements, schémas moteurs, ...) et l'architecture sous-jacente (système nerveux). L'absence de modèles adaptés nous permet de constater l'insuffisance de nos modèles connexionnistes mais elle ne remet pas en cause l'approche elle-même. Dans la suite de cette étude nous allons étudier quelques zones de convergence entre les modèles réactifs et les réseaux de neurones, particulièrement les réseaux modulaires et les modèles symboliques, afin de définir les principes d'une organisation « Neuro-Réactive » (chapitre V). Nous proposerons ensuite (chapitre VI) une architecture fondée sur ces principes et réalisant, au moins en partie, un premier pont entre les deux approches.

CHAPITRE V :

VERS DES MODELES NEURONAUX

POUR LE

CONTROLE REACTIF

Les deux chapitres précédents (chapitres III et IV) nous ont proposé successivement une étude des modèles de contrôleurs réactifs et une introduction aux méthodes connexionnistes et au neurocontrôle. Cette étude bibliographique générale nous a permis de mettre en avant la nécessité d'une architecture connexionniste spécifique, adaptée aux contraintes du contrôle réactif.

Le critère essentiel étant la modularité, nous proposons ici une description des différentes architectures neuromimétiques modulaires présentées dans la littérature. Parmi les différentes architectures neuronales proposées, nous avons plus particulièrement détaillé les modèles dits de « neuroéthologie calculatoire » et d'« intelligence visuo-motrice » car, bien qu'ils soient dénués de capacité d'apprentissage, ils sont très proches des modèles de contrôleurs recherchés.

Dans un deuxième temps nous analyserons les modèles décrits sous l'angle du contrôle réactif et nous proposerons une alternative basée sur la spécialisation, dans un réseau de neurones, de groupes fonctionnels. Cette approche est très librement inspirée de la neurobiologie et des « assemblées de neurones ».

Vers des Modèles Neuronaux pour le Contrôle Réactif

Au cours des chapitres précédents nous avons étudié deux techniques de contrôle, le « Behavior-Based-Control » et le neurocontrôle, que nous souhaitons utiliser pour améliorer la flexibilité et la réactivité des systèmes de production. Dans ce chapitre, nous proposons de réunir ces deux approches au sein d'une architecture mixte. Pour cela, nous allons dans un premier temps définir le cadre général de ce travail (intérêt, approche, ...) avant de proposer une analyse des différents algorithmes connexionnistes modulaires, la modularité étant une des contraintes principales imposées par le contrôle réactif. Enfin, nous proposerons une architecture, d'inspiration biologique, permettant d'unifier neurocontrôle et Contrôle Réactif.

V.1. Introduction

Le chapitre IV nous a montré que les modèles neuromimétiques les plus courants ne sont pas adaptés aux structures réactives : les réseaux de neurones les plus courants (excepté les réseaux symboliques) sont des organisations intrinsèquement monolithiques qui fournissent un résultat dicté par l'ensemble du réseau. Cette structure, qui a longtemps été qualifiée de « boîte noire », est à l'origine des principales limitations des réseaux de neurones et, en particulier, de l'oubli catastrophique (voir section IV.3.4.2 et [FRE 91]) et de perte d'information due à des conflits entre les différentes entrées proposées au réseau. En outre, cette organisation est une caractéristique propre aux réseaux de neurones formels. L'étude macroscopique des modèles biologiques permet d'identifier de nombreuses zones fonctionnelles plus ou moins bien définies. Ces aires cérébrales sont d'ailleurs à l'origine de l'organisation modulaire du contrôle réactif. Pour permettre la création de réseaux « Neuro-Réactifs », nous devons donc chercher prioritairement à supprimer l'organisation monolithique des réseaux de neurones en privilégiant une représentation partiellement localisée de l'information.

V.2. Approches neuronales

La structure type « boîte noire » des réseaux de neurones est progressivement remise en question par la communauté scientifique. Plusieurs auteurs ont proposé des structures neuronales à information localisée comme les cartes cognitives, les mémoires associatives ou les « RBF-Nets » [RON 95] mais cette approche entre souvent en contradiction avec les modèles réactifs puisque le réseau communique avec son environnement par l'intermédiaire d'une interface symbolique. Grâce à une organisation sensori-motrice (les clés sont soit des stimuli, soit des actions motrices), certains modèles de cartes cognitives peuvent cependant être considérés comme des contrôleurs réactifs. Une deuxième approche consiste à construire une structure modulaire composée de réseaux de neurones spécialisés. Dans ce chapitre, nous présentons successivement ces deux approches, sans nous limiter au contrôle réactif car la plupart des modèles modulaires ont été exclusivement appliqués à des problèmes de reconnaissance de formes. Il s'agit donc d'autant d'approches de la modularité et nous étudierons leur application éventuelle au contrôle sensori-moteur.

V.2.1. Des modèles d'inspiration neurobiologique

Les réseaux de neurones formels tels que nous les avons décrits plus haut sont librement inspirés de l'organisation générale du système nerveux, c'est-à-dire essentiellement des lois d'activation qui régissent les cellules neuronales (somme des entrées, seuillage, décharge, ...). Les travaux que nous présentons ici, au contraire, s'inspirent de la topologie réelle observée dans le système nerveux. Il s'agit donc moins, ici, de construire un modèle neuronal doué d'apprentissage, que de créer manuellement un réseau, déduit soit de l'observation d'un réseau réel, soit du comportement désiré.

V.2.1.1. Neuroéthologie calculatoire

La « neuroéthologie calculatoire[53] » est née à la fin des années quatre-vingts sous l'impulsion essentielle de Randall Beer (Université de Cleveland) en collaboration avec le biologiste Hillel Chiel. La problématique initiale est similaire à celle qui a motivé, quelques années plus tôt, la naissance du contrôle réactif : si l'Intelligence Artificielle utilise une vision exclusivement basée sur notre introspection, c'est essentiellement parce que le système nerveux humain est trop complexe pour être copié directement. Randall Beer s'inspire donc d'un modèle plus simple - c'est-à-dire dont le système nerveux est beaucoup moins complexe - pour construire un robot hexapode, véritable « insecte artificiel » [BEE 90].

L'organisation générale du contrôleur est proche de l'architecture « Subsumption » présentée figure III-4 (les modèles comportementaux sont d'ailleurs similaires) : le réseau de neurones est divisé en dix sous-réseaux, six réseaux « locomotion » (un par patte), un réseau « errance », un réseau « suivi de frontière », un réseau « manger » et un réseau « choix de comportement ». Chaque réseau est construit en s'inspirant au maximum des données biologiques. Cependant, de l'avis même des auteurs, il est impossible de rendre compte de la complexité des réseaux biologiques ni même des cellules nerveuses. Le modèle de cellule utilisé est un neurone à état réel, dont la fonction de transfert est linéaire à seuil (avec un saut initial). Ce modèle de cellule, classique en connexionnisme, est rendu plus complexe par l'adjonction d'un courant intrinsèque et de pertes membranaires. Ces composantes particulières permettent de créer des neurones dont la dynamique est plus complexe. Ils peuvent en particulier décharger spontanément (et régulièrement) sans être soumis à une quelconque activité ce qui permet de créer des neurones *Pacemakers*[54]. L'existence de ce type de neurone dans les réseaux biologiques a d'ailleurs été récemment confirmée par les neurobiologistes au niveau de la commande motrice (génération d'un rythme moteur dans le système gastrique du homard [SEL 83] et de l'aplysie [KAN 88]) et dans certaines aires visuelles (activité rétinienne spontanée chez le foetus [SHA 92]).

Les cellules sont reliées par des connexions pondérées activatrices ou inhibitrices. La figure V-1 présente un des six réseaux de contrôle des pattes (les six réseaux sont identiques) et le réseau de couplage entre les neurones « pacemakers » de chaque patte.

---

[53] Computational Neuro-Ethology

[54] régulateurs, métronomes.

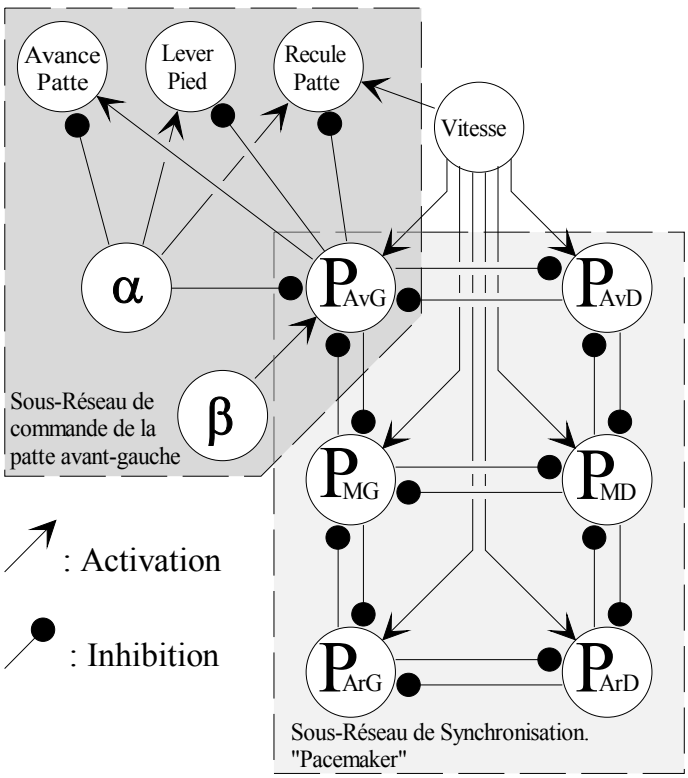


Figure V-1 : *Réseau « Pacemaker » et réseau « marche » de la patte avant gauche*

Le modèle complet comporte six réseaux « marche » identiques. Les connexions inhibitrices du réseau « Pacemaker » permettent de synchroniser les actions des différentes pattes. Les neurones α et β sont des neurones sensoriels correspondant respectivement aux positions avant et arrière de la patte.

Grâce à ce réseau relativement simple (43 neurones pour l'ensemble de la partie « marche »), le robot est capable de marcher à des rythmes différents suivant la valeur du neurone « vitesse ». La vitesse de déplacement de chaque patte prise individuellement étant constante, les variations en vitesse sont dues à l'utilisation de cinq démarches différentes, depuis la marche simple (une patte à la fois), jusqu'à la marche tripode proprement dite. Randall Beer souligne le fait que ces différents comportements sont produits par un réseau unique, sans modification de la topologie ou des poids synaptiques. Le robot passe en effet d'une démarche à l'autre par simple variation de la stimulation appliquée au neurone « vitesse ». De plus, l'étude du réseau a montré que de manière assez surprenante il résiste à des lésions artificielles pratiquées sur son « système nerveux » alors qu'il est apparemment dimensionné au minimum.

Le modèle de Randall Beer est intéressant pour deux raisons essentielles :

- D'une part, il montre clairement l'intérêt des modèles neuronaux pour le contrôle réactif. Pour vérifier les capacités présentées, en simulation, par ce réseau (malgré les nombreuses simplifications : frottements, délais, inertie, bruits, ...), il a été implanté sur un robot tripode réel [BEE 92]. Les résultats obtenus ont confirmé les capacités du système, en particulier quant aux différentes démarches possibles.

- D'autre part, dans le cadre d'une approche neuronale plus générale, la « neuroéthologie calculatoire » est une importante source de renseignements. Les travaux sur « l'insecte artificiel » ont d'ailleurs été à l'origine d'une étude plus générale sur l'application des réseaux de neurones au contrôle réactif [GAL 92]. Cette analyse porte l'accent sur les

aspects dynamiques propres aux systèmes réactifs, ce qui confirme les conclusions énoncées au terme du chapitre IV : grâce à leur dynamique complexe, seuls les réseaux récurrents sont à même de contrôler des systèmes complexes interagissant dynamiquement avec leur environnement.

Par contre, cette architecture est difficile à mettre en oeuvre : le réseau décrit compte 43 neurones soit près de 500 paramètres à ajuster manuellement. L'utilisation d'algorithmes d'apprentissage semble donc indispensable.

L'approche « neuroéthologique » a aussi été suivie par Willard Miranker et Barry Willner (IBM Research Division) qui proposent un modèle neuronal pour la construction des toiles d'araignée [MIR 94]. Ce modèle est basé sur la génération de séquences d'actions par un réseau de neurones spécifique (Wave Propagating Neural System) couplé à des processus réflexes permettant de modifier le déroulement de la séquence (par exemple en inversant le sens du tissage lorsque l'araignée rencontre le bord de la toile). Comme pour le réseau proposé par R. Beer, le modèle neuroéthologique de W. Miranker est un modèle récurrent dont les connexions sont fixées manuellement (le réseau est dépourvu de règles d'apprentissage).

V.2.1.2. L'intelligence Visuo-motrice

La majorité des systèmes vivants (y compris les insectes) utilisent, comme source principale d'information, le flux optique perçu par différents types de capteurs (essentiellement la vision binoculaire - stéréoscopique ou non - et les yeux à facettes). L'information visuelle, extrêmement complexe, traverse une série de couches neuronales pour être transmise aux neurones moteurs. Cette transmission s'accompagne de traitements massivement parallèles complexes qui permettent de transformer l'information visuelle en une réponse motrice efficace grâce à une cartographie polaire [CAM 88]. L'intelligence Visuo-Motrice [FRA 92] cherche à exploiter, en robotique, les principes biologiques de traitement de l'image visuelle pour produire des réflexes moteurs élémentaires. Ce principe permet d'envisager deux types d'approches.

Après avoir étudié l'oeil à facettes de la mouche et la structure neuronale associée, Nicolas Franceschini a implanté, en logique câblée, les principes d'acquisition et de traitement de l'information visuelle sur un robot mobile. Celui-ci est doté d'un oeil à facettes plan omnidirectionnel (96 facettes et autant de photorécepteurs, figure V-2), de deux « yeux » pour la vision frontale (de 13 facettes chacun) et d'un détecteur de cible (caméra vidéo). Les couples de facettes adjacentes permettent de détecter 96 mouvements élémentaires puis, au moyen d'un réseau neuronal câblé, de synthétiser une commande motrice pour ajuster la vitesse à la proximité des obstacles puis pour les éviter [MAR 94]. Le robot est ainsi capable de se déplacer rapidement (jusqu'à 50 cm/s) dans un environnement encombré et d'atteindre une cible située derrière un ensemble d'obstacles. Ce modèle a été étendu - en simulation - pour le guidage d'un aéronef volant à faible altitude au-dessus d'un environnement inconnu. La détection des mouvements élémentaires est alors utilisée pour déterminer l'altitude relative de l'aéronef [MUR 94].

Ce travail, tout comme la « neuroéthologie calculatoire », montre l'intérêt des modèles neuronaux pour le contrôle réactif mais, à l'inverse des modèles de Randall Beer et de Willard Miranker, il est basé sur des réseaux en couches et non sur des réseaux récurrents. Cette architecture « feed-forward » permet un traitement rapide d'une grande quantité d'informations mais elle ne permet pas de gérer des configurations complexes telles que des minima locaux en poursuite de cible : lorsqu'il est nécessaire de s'éloigner de la cible pour éviter un obstacle, le

réseau doit pouvoir mémoriser les perceptions et les actions antérieures pour construire des séquences d'actions.

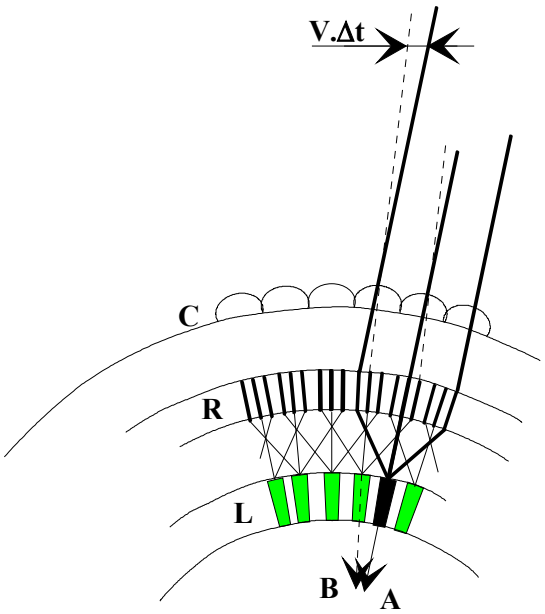


Figure V-2 : *Schéma optique et neurophysiologique de l'oeil composé de la mouche (d'après [BRA 91]).*

avec : C, la cornée ; R, la rétine et L, la « lamina » (premier ganglion optique). La comparaison des signaux A et B aux temps t et $t + \Delta t$ permet, connaissant la vitesse V , de déterminer la distance de l'objet correspondant.

La deuxième approche de l'intelligence visuo-motrice est l'utilisation des modèles neuronaux classiques pour l'apprentissage direct d'une commande sensori-motrice (c'est-à-dire pour l'apprentissage d'une application injective de l'espace des stimuli dans l'espace des commandes). Etant donnée la complexité des stimuli (généralement une image visuelle en niveaux de gris), il est impossible de parcourir entièrement l'espace d'entrée. Les réseaux de neurones, grâce à leur capacité de généralisation, sont donc particulièrement indiqués pour l'apprentissage de ce type de fonction. Cette approche est suivie par de nombreuses équipes de recherche que ce soit en robotique mobile ou pour des bras manipulateurs [MIL 87] [OPI 90b] [BID 94] [NEW 93] [POM 93]. Cependant il s'agit là d'une utilisation classique des réseaux de neurones (non modulaires) où l'apprentissage est guidé par un superviseur humain (voir section IV.4.1.2).

V.2.1.3. Les modèles de la colonne corticale

En 1988, Yves Burnod (Institut de Neurosciences, Paris) a proposé, dans un livre intitulé « An Adaptive Neural Network : The cerebral cortex » [BUR 88], un modèle du cortex basé sur des unités de plus haut niveau que les cellules neuronales : la colonne corticale. Ce modèle a été exploité par la suite - en particulier par Frédéric Alexandre à l'INRIA Lorraine - pour construire des réseaux multicouches dédiés à la reconnaissance de formes (reconnaissance de la parole et reconnaissance visuelle [ALE 88]) et, par Yves Burnod lui-même, pour la construction de modèles sensori-moteurs du cortex [BUR 90]. Le modèle de la colonne corticale n'est pas à proprement parler un réseau de neurones mais plutôt un modèle général décrivant le fonctionnement d'un petit nombre de cellules neuronales géographiquement localisées dans le cortex. Un réseau de colonnes corticales peut donc être vu sous la forme d'un réseau de réseaux

de neurones. Il est donc intéressant de décrire ici ce modèle qui modélise une forme d'interaction entre réseaux de neurones.

Suivant l'analyse de Y. Burnod les modèles neuronaux classiques souffrent d'un éloignement trop important des réseaux biologiques dû en particulier à la grande diversité des neurones biologiques (les réseaux de neurones formels utilisent généralement un unique modèle de neurone). Pour éviter l'explosion combinatoire qu'entraînerait une modélisation plus fine des processus neuronaux, le modèle de Burnod est basé sur un découpage de plus haut niveau (les colonnes corticales) qui, toujours selon Yves Burnod, permet de faire apparaître des invariants structuraux.

Ainsi, le modèle de la colonne corticale a été utilisé par Yves Burnod pour modéliser les transformations visuo-motrices produites par le cortex à partir des informations visuelles, oculomotrices (direction du regard) et proprioceptives [BUR 90][TRO 92a]. L'apprentissage utilise les boucles sensori-motrices pour associer le déplacement du bras avec un mouvement perçu visuellement et avec un rapprochement de la cible. Cette étude a permis de vérifier l'existence, dans les cellules corticales, d'invariants et de directions privilégiées lors de la mise en correspondance des stimuli visuels et des commandes motrices.

L'intérêt essentiel du modèle de Burnod est de présenter les règles de fonctionnement du cortex à un degré d'organisation immédiatement supérieur au niveau cellulaire et neuronal. Il permet donc de dégager des principes d'organisation et de communication inter-réseaux tels que les inhibitions latérales entre colonnes, l'existence de trois états différents caractérisés par une fréquence d'apparition des potentiels d'activité. Ce modèle propose aussi des lois globales pour la dynamique des réseaux.

V.2.2. Modèles coopératifs séquentiels

Les réseaux de neurones sont de plus en plus fréquemment appliqués à des problèmes complexes et structurés. Afin d'exploiter la structure intrinsèque de ce type de problèmes (par exemple la reconnaissance de caractères, de mots, de phrases, ...), plusieurs auteurs ont proposé d'utiliser une architecture regroupant plusieurs réseaux dédiés aux différents sous-buts. Cette approche propose donc de limiter l'aspect boîte noire des réseaux de neurones en les appliquant à une architecture modulaire.

Dans cette section, nous présentons les modèles coopératifs séquentiels qui utilisent les résultats produits successivement ou simultanément par plusieurs réseaux pour déterminer la valeur finale.

Dans de nombreux problèmes de classification ou de reconnaissance de formes (moins fréquemment, dans les applications de contrôle de robots), le traitement de l'information procède par raffinements successifs. Dans ce cas, l'utilisation d'une architecture modulaire permet un traitement par partie ce qui simplifie et, souvent, accélère la procédure d'apprentissage. Il suffit pour cela de disposer les réseaux en cascade suivant l'organisation du problème à traiter. Ainsi, dans le cas de la reconnaissance de la parole ou de l'écriture manuscrite, on peut dégager successivement les sous-systèmes suivants :

- Lettre
- Phonème
- Mot
- Groupe de mots (Sujet/Verbe/COD/COI...)

- Phrase

...

Ce type de traitement séquentiel est utilisé dans de nombreux travaux. On peut citer, à titre d'exemple, [ALL 90] (reconnaissance de formes structurées) ou [MAR 90a] qui l'applique à l'identification du pays d'origine à partir de la plaque minéralogique (figure V-3). Dans ce dernier cas l'architecture est convergente (fusion de données), mais le modèle séquentiel peut aussi être divergent (plusieurs résultats indépendants) ou regrouper les deux approches (décomposition en sous-buts puis fusion des résultats intermédiaires).

L'architecture séquentielle est de plus en plus couramment utilisée pour accélérer les processus connexionnistes en limitant le nombre de connexions. Elle a l'avantage d'être d'un usage relativement simple : l'absence de boucles de rétroaction entre les réseaux permet un apprentissage par partie. En outre, si les sous-réseaux sont non récurrents, elle permet de garantir le temps de réponse (dynamique de type passe-avant). Par contre, le gain de performance est étroitement dépendant de l'architecture utilisée [DOR 94]. Le concepteur d'un tel système doit donc rechercher la décomposition optimale pour laquelle les différents réseaux sont suffisamment simples (buts élémentaires) sans que l'architecture globale ne devienne ingérable [BES 92a]. Certains algorithmes permettent cependant de développer la structure séquentielle automatiquement lors de l'apprentissage. Ainsi, Tristan Salomé et Hughes Bersini ont une architecture coopérative (EMANN[55]) permettant d'apprendre simultanément les connexions synaptiques et la structure du réseau ([BON 94], chapitre 7). Ce modèle est inspiré du réseau dît « Cascade-Correlation-Learning » dont la structure est générée progressivement selon les besoins de l'apprentissage ([SIM 92] présente une application du réseau « Cascade-Correlation-Learning » pour le contrôle de processus).

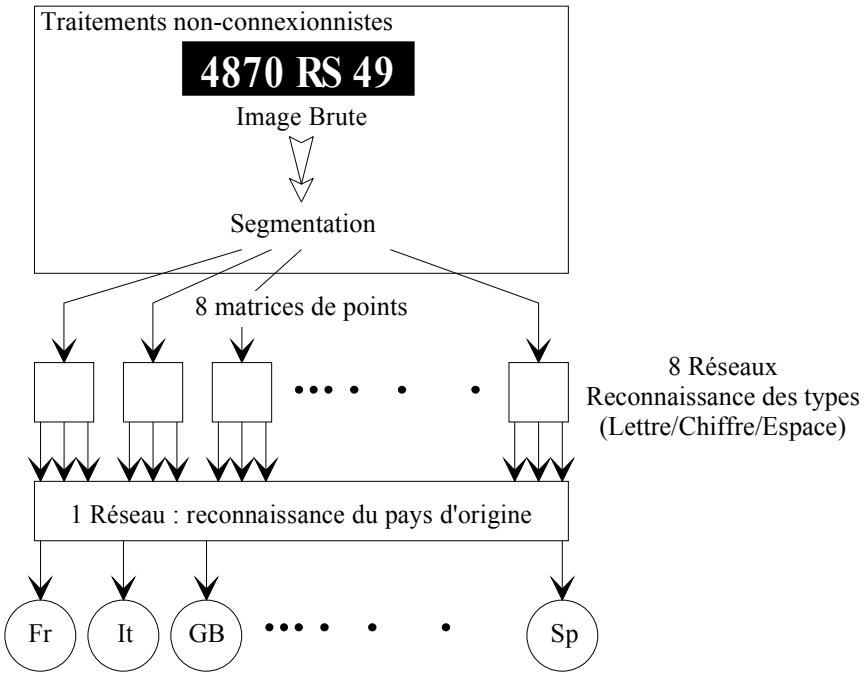


Figure V-3 : *Identification du pays d'origine à partir de la plaque minéralogique*

[55] Evolving Modular Architecture for Neural Networks

Les architectures séquentielles ont été fréquemment appliquées aux problèmes de classification et de reconnaissance de formes. Par contre leur utilisation en neurocontrôle est moins courante : en contrôle de processus il y a souvent interdépendance entre les différents sous-buts (dans le cas d'un bras robot, par exemple, les positions des différentes articulations sont fortement liées) ce que ne permet pas l'architecture séquentielle (les sous-buts doivent être indépendants). Cependant des cascades de réseaux peuvent être utilisées pour différencier les « aires sensorielles » des « aires motrices » dans les modèles de type intelligence visuomotrice (section V.2.1.2) ou pour décomposer le contrôle entre les tâches de planification et de contrôle [RAB 91][RAB 92].

L'utilisation d'une architecture connexionniste coopérative pour le « Behavior-Based-Control » semble difficilement envisageable. En effet, les multiples boucles de rétroaction - caractéristiques des modèles réactifs - ne permettent pas de réaliser simplement un apprentissage par partie (à moins de fixer a priori le graphe de communication entre les différents modules mais nous avons vu que ce graphe est - précisément - la principale source de difficultés des architectures réactives).

V.2.3. Modèles de supervision

Les principales difficultés rencontrées par les architectures coopératives viennent de l'absence de hiérarchie entre les différents modules. Ceux-ci ne communiquant que par échange d'information, la gestion d'un « réseau de réseaux » de grande taille pose des problèmes quasiment identiques à ceux rencontrés par le contrôle réactif (augmentés par la complexité intrinsèque des algorithmes connexionnistes). Nous présentons ici une seconde approche basée sur un processus compétitif plutôt que coopératif. Les réseaux ne sont donc pas utilisés simultanément pour générer, par agrégation, une réponse finale. Ils sont, au contraire, soumis à des lois d'arbitrage gérées, à un niveau hiérarchique supérieur, par un réseau « maître ». Cette architecture est donc à rapprocher de l'algorithme d'apprentissage proposé par Pattie Maes [MAE 90c] pour les contrôleurs réactifs (section III.6.3). L'approche connexionniste présente alors l'avantage d'unifier les modes de représentation entre les modules (comportements) et les méthodes d'arbitrage. En outre, dans certains cas, cette approche permet l'apprentissage simultané des « comportements » et du « maître », ce qui représente un apport certain par rapport aux modèles d'arbitrage statistiques.

V.2.3.1. Modèles hiérarchiques

Les modèles d'arbitrage les plus simples sont formés par une simple hiérarchie de réseaux de neurones. Les niveaux hiérarchiques successifs permettent de séparer le problème initial en une série de sous-problèmes indépendants. Chacun d'eux est ensuite traité, au niveau le plus bas, par un expert approprié (figure V-4). Ce type d'architecture est relativement simple à mettre en oeuvre. Le problème est divisé a priori en plusieurs classes et chaque expert est entraîné à résoudre une classe particulière. Parallèlement le ou les réseaux d'arbitrage apprennent à isoler les différentes classes. Les phases d'apprentissage de chacun des réseaux sont donc totalement indépendantes.

L'action du réseau de supervision sur les experts peut être une sélection simple (la sortie de l'expert le plus adéquat est la seule prise en compte) ou une pondération d'actions (les sorties des experts sont pondérées puis additionnées au moyen d'unités $\Sigma\Pi$). La deuxième solution est généralement plus robuste (en particulier au voisinage des singularités) mais elle peut, dans

certains cas, occasionner des réponses aberrantes. En outre, elle impose d'utiliser un codage des sorties identique pour chacun des experts ce qui n'est pas forcément souhaité.

Du fait de sa simplicité, ce modèle a été très largement appliqué à différents. En reconnaissance vocale par exemple, dans l'architecture « SID » (Source IDentification), un réseau d'arbitrage est utilisé pour classer les sources (les locuteurs) afin de simplifier la tâche des experts (en limitant, pour une même syllabe, les variations du signal temps/fréquence) [HAM 92]. En neurocontrôle cette architecture est particulièrement intéressante pour contrôler des systèmes sur des domaines de fonctionnement différents. Il est alors possible de contrôler des systèmes non-linéaires par une juxtaposition de modules connexionnistes simples de type Perceptron ou ADALINE.

Cependant l'utilisation de cette architecture reste soumise à la capacité du concepteur à isoler les différents domaines de fonctionnement (ou les non-linéarités) ce qui, en neurocontrôle, n'est pas toujours possible. En outre, il est difficile d'attribuer les erreurs à l'un ou l'autre des niveaux hiérarchiques ; autrement dit, si une erreur est commise, doit-elle être attribuée à l'expert correspondant ou au superviseur qui aurait dû choisir un autre expert.

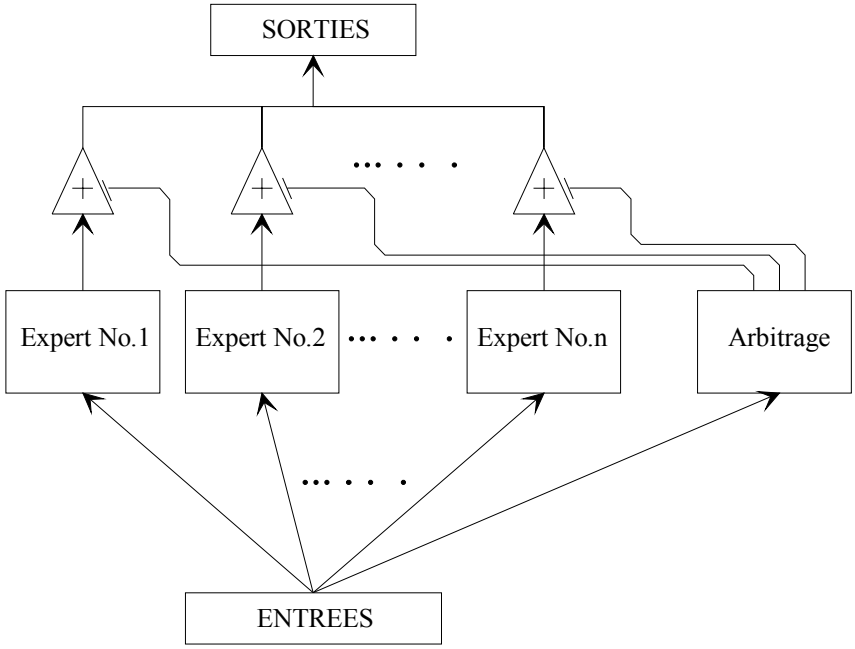


Figure V-4 : *Hiérarchie de réseaux de neurones*

Pour répondre aux problèmes posés par les modèles hiérarchiques, plusieurs architectures ont été développées. Elles permettent un apprentissage simultané des experts et du ou des superviseurs en déterminant la responsabilité de chacune des parties. Nous présentons ici trois approches différentes. Les deux premières (le réseau METAPI et les modèles EXPERTS/GATING) sont relativement proches de l'architecture que nous venons de décrire, par contre, la troisième (OARE) est basée sur un principe totalement différent.

V.2.3.2. Le réseau « METAPI »

L'architecture du réseau METAPI [HAM 92] est directement issue du modèle SID décrit dans le paragraphe précédent mais le réseau de supervision est entraîné après intégration de l'ensemble des experts. Pour cela, J.B. Hampshire et A. Waibel utilisent une adaptation de l'algorithme de rétropropagation de gradient pour les unités ΣΠ [RUM 86b]. Les réseaux experts

ayant été préalablement entraînés séparément, l'erreur est entièrement attribuée au superviseur qui peut ainsi apprendre, pour chacun des cas de la base d'exemples, quel expert est le mieux adapté.

V.2.3.3. Modèle Experts/Gating

Robert A. Jacobs et Michael I. Jordan (MIT) ont proposé une architecture hiérarchique plus complète que les deux précédentes. Le modèle Experts/Gating permet en effet l'apprentissage simultané de l'ensemble de l'architecture. Il offre aussi une possibilité d'encapsulation des niveaux hiérarchiques. Il a, de plus, été utilisé par les auteurs pour des applications de contrôle de systèmes non-linéaires ce qui représente un intérêt supplémentaire dans le cadre du travail présenté ici [JAC 93] [JOR 94].

Structurellement cette architecture diffère peu du modèle METAPI : plusieurs réseaux « Experts » traitent en parallèle un même vecteur d'entrée, puis les réponses sont combinées grâce à un réseau médiateur (« Gating ») par pondération et addition (figure V-5). L'intérêt essentiel du modèle « Experts/Gating » est son algorithme d'apprentissage qui permet d'entraîner simultanément les réseaux experts et le réseau médiateur en laissant les experts s'auto-organiser pour déterminer les domaines de viabilité de chacun d'entre eux. L'algorithme utilisé est relativement simple : connaissant les probabilités a posteriori pour que chaque expert ait fourni la bonne réponse (ces probabilités sont simplement calculées par la formule des probabilités conditionnelles de Bayes en évaluant la sortie de chaque expert par rapport à la sortie désirée), le réseau Gating est entraîné pour favoriser le meilleur expert. Parallèlement chacun des experts reçoit un signal d'erreur proportionnel à la valeur g_i fournie par le Gating. Cet algorithme permet une auto-organisation par compétition (« Winner-Takes-All ») des différents experts.

Un des intérêts fondamentaux de cette approche est de laisser le concepteur du système totalement libre de l'architecture utilisée pour chacun des experts. Il est ainsi possible de développer récursivement le modèle en implantant à la place de chacun des experts un modèle hiérarchique de niveau inférieur, ce qui permet de diviser le problème en zones de fonctionnement puis de subdiviser chacune des zones et ainsi de suite. Cette décomposition hiérarchique permet de combiner un grand nombre de réseaux mono-couche pour contrôler des systèmes non-linéaires (linéaires par partie) Il est aussi possible de compléter le modèle en ajoutant un réseau partagé dont la sortie est systématiquement utilisée (Modular Architecture with a Shared Network, MAS) et d'ajouter des contraintes pour forcer l'architecture à s'organiser suivant des lois générales pré-établies (Constrained Modular Architecture with a Shared Network, CMAS).

Outre le gain de performance, l'intérêt essentiel de l'architecture modulaire est d'éviter l'apparition de « l'oubli catastrophique ». Ainsi, Jacobs et Jordan ont utilisé leur modèle pour entraîner un robot à manipuler successivement des charges de 22 kg et 0 kg. Les apprentissages étant dirigés vers des réseaux différents l'oubli catastrophique est évité alors qu'un réseau monolithique doit passer, à chaque changement de charge, par une nouvelle phase d'apprentissage.

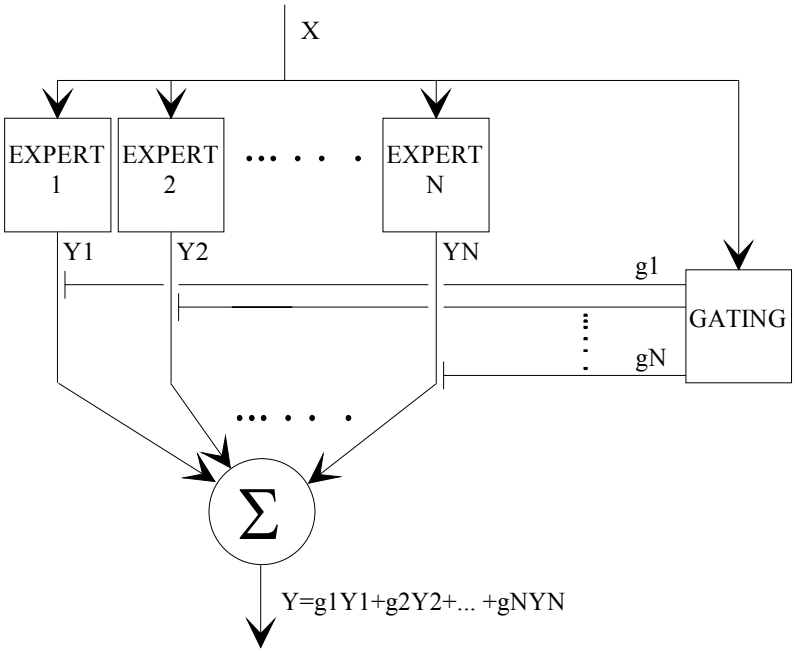


Figure V-5 : *L'architecture modulaire de Jacobs et Jordan*

L'architecture présentée ici comprend n réseaux experts E_i fournissant une sortie Y_i en réponse à l'entrée X. Le réseau médiateur fournit un vecteur $S_1..S_n$ qui est normalisé pour obtenir les valeurs $g_1..g_n$ représentant les probabilités a priori pour que les différents experts fournissent une réponse appropriée. Ces valeurs sont combinées pour calculer la sortie finale de l'ensemble : Y.

Le modèle Experts/Gating est le seul qui autorise réellement une auto-organisation des différents modules parallèlement à leur apprentissage. Il est en cela particulièrement intéressant pour une éventuelle utilisation en contrôle réactif. Malheureusement les applications présentées sont basées sur l'apprentissage de la dynamique inverse (apprentissage supervisé, séparation des phases d'apprentissage et de contrôle) et non sur une architecture type critique adaptative (apprentissage « on-line » et par renforcement). De plus, la phase d'apprentissage est sensiblement allongée en raison du nombre important de réseaux. Enfin, même si les domaines de viabilité des différents experts sont fixés dynamiquement, ils sont largement tributaires du nombre d'experts, de leur complexité et de l'organisation hiérarchique qui reste, obligatoirement, de type « feed-forward » et « winner-takes-all », autant de contraintes structurelles qui sont incompatibles avec les principes architecturaux du contrôle réactif.

V.2.3.4. La méthode OARE

La méthode OARE[56] [POM 93], à l'inverse des trois précédentes, n'est pas totalement connexionniste. Elle a été développée parallèlement à ALVINN (section V.2.1.2) pour permettre de choisir, parmi plusieurs contrôleurs connexionnistes, celui fournissant la réponse la plus sûre. Par rapport aux médiateurs connexionnistes proposés jusqu'ici, le but premier d'OARE est d'éviter de rallonger le temps d'apprentissage (tous les contrôleurs doivent donc être indépendants pour pouvoir être entraînés simultanément et l'arbitre doit ensuite être immédiatement disponible) ou d'alourdir la structure. De plus, l'architecture doit être totalement

[56] Output Appearance Reliability Estimation

modelable dynamiquement. En particulier, il doit être possible d'ajouter de nouveaux réseaux pour faire face à de nouvelles contraintes ou à un nouvel environnement ce que ne permettent pas les modèles présentés jusqu'à maintenant.

Dans OARE, tous les experts utilisent une représentation semi-distribuée, gaussienne, des sorties. L'architecture est basée sur le principe intuitif qu'un réseau mal adapté au problème fournira une réponse mal structurée sur l'ensemble des neurones de sortie. La sortie fournie par un réseau est donc comparée à la gaussienne idéale (i.e. la gaussienne utilisée pour l'apprentissage), la différence permet alors d'estimer la confiance attribuée à cet expert. Le superviseur est alors obtenu directement : la tâche est attribuée au réseau présentant le plus fort indice de confiance.

OARE souffre, paradoxalement, de sa trop grande simplicité qui ne permet pas, dans le cadre du contrôle réactif, d'arbitrer des comportements trop différents (par exemple des comportements utilisant des actionneurs différents) ou d'arbitrer les réseaux en fonction d'un contexte d'entrée et non uniquement en fonction de la qualité de la sortie (il est donc impossible de différencier un réseau proche de l'optimum d'un réseau ayant fortement généralisé sa sortie optimale). Cette méthode reste cependant particulièrement intéressante pour permettre l'accès à des procédures de sécurité en cas de réponse peu fiable. Dean Pomerleau propose d'ailleurs une méthode permettant d'étendre OARE : IRRE[57] [POM 93] semble plus prometteuse encore. Elle consiste en une rétropropagation d'activité dans le réseau après que celui-ci ait produit la réponse. Cette rétropropagation permet de déterminer l'entrée idéale correspondant à la sortie activée. Une simple comparaison entre l'entrée réelle et l'entrée idéale permet ensuite de déterminer le degré de certitude. Celui-ci caractérise alors non seulement la qualité de la réponse mais aussi son adéquation avec l'entrée présentée : si le réseau a trop généralisé, IRRE donne une sûreté faible là où OARE détecte une réponse parfaite.

V.2.4. Modèles topologiques

Excepté les modèles d'inspiration neuroéthologique décrits en tête de ce chapitre, tous les modèles modulaires proposés jusqu'à maintenant utilisent des architectures multi-réseaux coopératives ou compétitives pour structurer le traitement de l'information dans les réseaux neuromimétiques. Dans les réseaux de neurones, le traitement appliqué à l'entrée est indissociable de l'information stockée dans les poids synaptiques. Plusieurs modèles de réseaux (en particulier les cartes de Kohonen) développent un modèle topologique de l'information, mais cette propriété est liée à des principes d'auto-organisation qui supposent un apprentissage non-supervisé (ce qui est incompatible avec le contrôle réactif). Cependant plusieurs auteurs ont proposé d'insérer des réseaux de type Kohonen dans des architectures plus complexes (multicouches) afin d'exploiter les propriétés topologiques de ces modèles dans un contexte d'apprentissage supervisé. Nous présentons ici trois modèles de ce type ainsi que deux approches qui, bien que n'utilisant pas directement des modèles topologiques, sont basées sur des notions relativement proches (regroupement hiérarchique des neurones par zone d'influence).

---

[57] Input Reconstruction Reliability Estimation

V.2.4.1. La méthode LASSO

Le modèle LASSO (« Learning ASsociation by Self-Organization » [MID 91]) combine les intérêts des modèles topologiques à auto-organisation et de l'apprentissage supervisé. Il permet en effet l'apprentissage d'associations entrée/sortie via une couche intermédiaire auto-organisée (carte de Kohonen). Celle-ci constitue donc une représentation structurée (topologiquement) des informations apprises par le réseau.

Le réseau LASSO est composé de trois ensembles de neurones (figure V-6) : les cellules d'entrée (E), les cellules de sortie (S) et les cellules cachées (K, c'est-à-dire les neurones d'association).

Le comportement des cellules de sortie est différent lors de la phase d'apprentissage et lors de la phase d'utilisation (l'apprentissage étant réalisé, par voie de conséquence, « off-line ») : au cours de l'apprentissage, seules les connexions W^{SK} sont utilisées et la carte de Kohonen s'auto-organise en fonction des associations entrée/sortie (on parle alors, pour l'ensemble des neurones d'entrée/sortie, de couche de communication). L'algorithme d'apprentissage est identique à celui utilisé pour les cartes de Kohonen (Annexe A2, section 4). En phase d'utilisation, les poids W^{KS} (non-utilisés pour l'apprentissage) sont fixés égaux aux poids W^{SK} correspondants. Le réseau est alors capable, à partir des seules entrées, de retrouver le neurone image, puis son groupe d'activation sur la carte de Kohonen, puis, par propagation sur les connexions W^{KS}, la sortie associée.

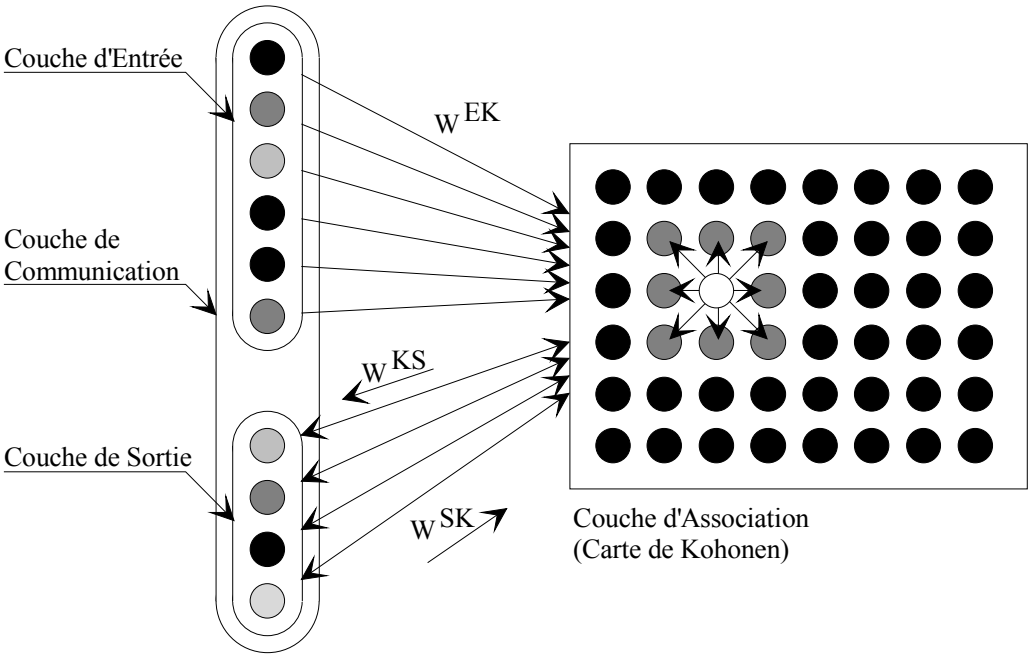


Figure V-6 : *Architecture du modèle LASSO*[58]

Plusieurs autres modèles exploitent les capacités d'auto-organisation topologique des réseaux de neurones. Ceux-ci sont généralement basés soit, comme LASSO, sur une ou plusieurs cartes de Kohonen utilisées comme couche de prétraitement soit sur le modèle ART de

[58] Seules les connexions d'un seul neurone de la couche d'association sont représentées sur la figure V-16.

Carpenter et Grossberg (modèle ARTMAP [CAR 91]). Ces différentes applications sont dédiées à des tâches de reconnaissance de formes, probablement en raison de l'inconvénient majeur des réseaux auto-organisés : l'apprentissage est extrêmement lent. Cette dernière caractéristique interdit l'utilisation de cette méthode pour un apprentissage « en-ligne », donc pour le contrôle réactif.

V.2.4.2. Association sensori-motrice par carte de Kohonen

Le rapprochement entre les modèles topologiques et le contrôle sensori-moteur (intelligence visuo-motrice) repose sur l'hypothèse fondamentale que des cartes topologiques (reflétant les propriétés de l'environnement) permettent de mettre en correspondance les entrées sensorielles et les sorties motrices. Cette hypothèse est fortement influencée par les recherches en neurobiologie (voir section V.2.1.3).

Partant de cette hypothèse, Ritter, Martinez et Schulten ont proposé une application de commande d'un bras robot et de sa pince (cinq degrés de liberté au total) [MAR 90b]. L'information entrante est d'origine visuelle, elle est fournie par deux caméras filmant l'espace d'évolution dans lequel le robot doit saisir des cylindres présentés aléatoirement dans l'espace de recherche.

L'architecture neuronale utilisée est une hiérarchie de cartes de Kohonen : un premier réseau (une carte tri-dimensionnelle) est utilisé pour passer des coordonnées caméra à un système de coordonnées « internes ». Un réseau de type « Perceptron » est alors utilisé pour calculer, pour chaque noeud de la carte, les angles appliqués à chacune des articulations du bras. A chaque noeud de ce premier réseau est associé un deuxième niveau hiérarchique composé d'une carte de Kohonen bi-dimensionnelle recevant en entrée l'angle à l'horizontale du cylindre vu par les caméras et associant cette information aux coordonnées angulaires de la pince par un processus similaire à celui utilisé par le premier réseau.

Les performances du système sont très intéressantes et le robot apprend rapidement à se positionner pour saisir le cylindre. Cependant, il semble que ces performances sont essentiellement dues aux caractéristiques particulières du problème. En effet, la topologie spatiale et la topologie neuronale associée sont très proches l'une de l'autre : deux entrées proches ont naturellement deux images proches (cela est dû entre autres à l'angle séparant les deux caméras - 90° - qui simplifie au maximum les problèmes inhérents à la vision stéréoscopique).

V.2.4.3. Construction Epigénétique

La construction épigénétique n'est pas à proprement parler un modèle topologique. Cette approche permet cependant de regrouper les cellules neuronales hiérarchiquement en créant un réseau de neurones structuré sur une base d'hypergraphe, chaque neurone incluant, dans sa fonction de décision, des stimulations « horizontale » (synaptiques) et des stimulations « verticales » provenant des neurones englobant ou englobés.

L'épigénèse a été initialement proposée par Yves Cochet et Gérard Paget pour construire des thésaurus connexionnistes (basés sur des mémoires associatives) dans des bases de données [COC 88][BIE 90a]. Cette technique a ensuite été étendue par Frédérique Biennier pour obtenir une structure en hypergraphe pour le contrôle de processus industriels [BIE 92a] [BIE 93]. C'est cette dernière version, basée sur des cartes cognitives [KOS 86a], qui est décrite ici.

Les cartes cognitives (comme les mémoires associatives) utilisent une représentation locale de l'information ce qui permet de représenter, au sein d'un même réseau, plusieurs niveaux de description. Par contre, cette architecture impose une connaissance a priori de l'environnement afin de construire le réseau et de fixer les poids (voir annexe A2, section 5). Les algorithmes épigénétiques permettent de modifier dynamiquement cette structure par création de cellules neuronales.

Les « hyper-cartes cognitives épigénétiques » reposent sur des principes similaires aux cartes cognitives classiques : chaque neurone représente un concept ou un fait et les connexions pondérées expriment les liens de cause à effet ou les associations licites (illicites pour des pondérations négatives). L'utilisation de la structure d'hypergraphe permet d'intégrer plusieurs niveaux d'abstraction dans une carte cognitive et d'associer directement certains neurones à un sous-ensemble du réseau. Dans cette structure, chaque neurone reçoit des stimulations provenant de connexions synaptiques directes mais aussi des stimulations provenant des neurones englobants ou englobés. Ces dernières ne sont cependant pas directement issues de l'état des niveaux hiérarchiques supérieurs ou inférieurs mais déduites de leurs entrées.

Un des intérêts essentiels des processus épigénétiques est qu'ils permettent de modifier dynamiquement le graphe neuronal en ajoutant des hyperneurones ou en scindant des unités en « sous-neurones » (parallèlement, bien sûr, à un processus d'adaptation des poids synaptiques).

L'épigénèse peut être comparée à un processus d'arbitrage entre différents sous-réseaux. La structure d'hypergraphe permet alors de réaliser des raisonnements partiels progressivement affinés au fur et à mesure de la transmission d'activité vers les niveaux inférieurs. Elle permet donc au réseau de fournir rapidement des réponses partielles (ne mettant en jeu que les cellules de haut niveau) puis, après des calculs plus longs car mettant en oeuvre plus de cellules, la réponse finale. Cette approche apparaît donc comme une voie intéressante vers des réseaux « structurés ». Malheureusement les procédures d'apprentissage disponibles sur ce type de réseau sont, pour l'instant, limitées à la seule adaptation des poids synaptiques. L'épigénèse souffre donc du même mal que les réseaux construits suivant les principes de la neuroéthologie calculatoire à savoir une « explosion » du nombre de paramètres lorsque la taille du réseau augmente.

V.2.4.4. Les organisations fractales

Les réseaux de neurones s'insèrent dans un contexte scientifique fortement marqué par le développement des sciences de la complexité et, en particulier, de l'étude des phénomènes chaotiques et des organisations fractales. Une des caractéristiques essentielles d'un système fractal (mais non unique) est de présenter une structure similaire quel que soit le niveau d'échelle auquel il est observé. L'engouement pour ces nouvelles structures a influencé plusieurs chercheurs en intelligence artificielle qui les ont exploitées tant en logique classique qu'en connexionnisme [BAR 89b].

Yoram Baram a proposé en 1989 une organisation fractale permettant de structurer l'information dans les réseaux de neurones en forçant les connexions à présenter, à tous les niveaux hiérarchiques, une topologie identique. La répétition de la connectivité sur les niveaux successifs conduit l'auteur à qualifier ce type de réseau de « fractal ». D'un point de vue connexionniste, il s'agit d'un ensemble de sous-réseaux de Hopfield obtenus par élagage d'une grande partie des connexions du réseau complet.

D'après Yoram Baram et Daniel Dubois [DUB 90], la structure fractale est en parfait accord avec les fondements biologiques du connexionnisme puisque les aires corticales ne sont

pas toutes interconnectées : alors que dans le cas général deux neurones ne sont pas interconnectés, le graphe des connexions permet de transmettre l'information entre deux points quelconques du cortex par l'intermédiaire d'un nombre de neurones relativement faible. Globalement cette caractéristique est effectivement partagée par la structure fractale présentée figure V-7 (huit connexions séparent les neurones les plus éloignés) mais au prix d'une connectivité forcée ne tenant aucun compte des caractéristiques particulières du problème posé. Le modèle fractal constitue donc une étude théorique intéressante mais peu applicable en pratique en raison des contraintes structurelles imposées par la régularité du graphe. Ce problème peut cependant être résolu en autorisant dans la structure des décompositions successives fonction des résultats intermédiaires obtenus aux niveaux supérieurs. Une telle structure serait alors relativement proche de celle construite par les algorithmes épigénétiques.

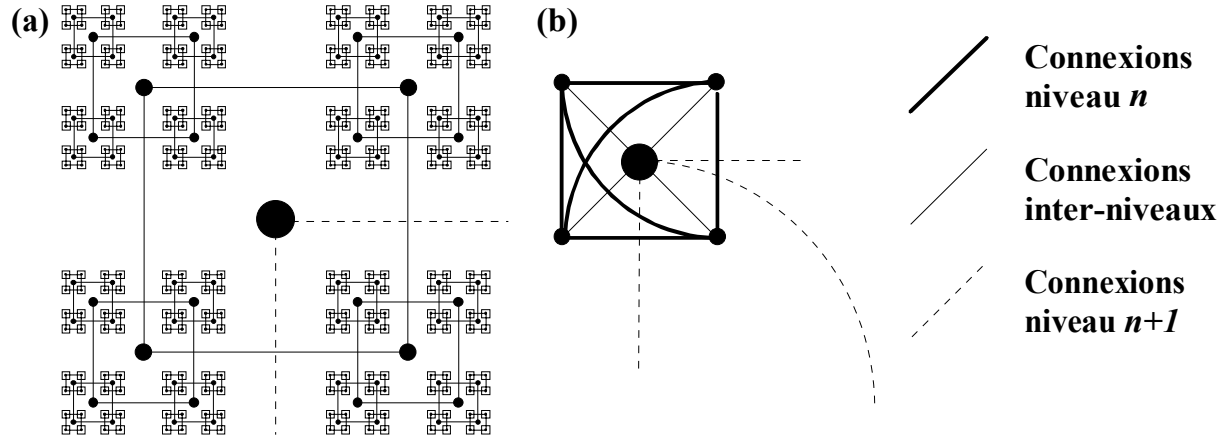


Figure V-7 : *Structure globale et connexions locales dans un réseau fractal*

(a) : La structure fractale est constituée de groupes hiérarchiques de quatre neurones formant des sous-réseaux totalement connectés.
(b) : Chaque sous-réseau est connecté à un des neurones du niveau supérieur.

V.2.5. Conclusion

La plupart des modèles cités ci-dessus implantent, dans une architecture connexionniste, des structures pré-définies, généralement manuellement, par le concepteur. Celui-ci doit donc organiser a priori un graphe complexe formé de réseaux ou de « sous-réseaux » communiquant par de multiples canaux, parfois mal définis. En outre, les quelques méthodes exclusivement basées sur l'apprentissage (modèle de Jacobs et Jordan, OARE) reposent sur un simple arbitrage entre plusieurs réseaux. Elles ne permettent que la parallélisation des tâches sans autoriser les traitements séquentiels multi-réseaux.

Aucune des approches présentées ci-dessus n'est donc totalement satisfaisante pour la réalisation de contrôleurs réactifs doués d'apprentissage : les structures créées sont, soit, trop pauvres structurellement, soit, construites manuellement ce qui limite l'intérêt de l'approche neuronale vis-à-vis des architectures « classiques » décrites dans le chapitre III. (« Subsumption architecture », AuRA, ...). Ainsi, si l'intérêt théorique des études de Randall Beer sur les modèles de neuroéthologie calculatoire est indéniable, leur intérêt pratique semble plus limité : le contrôle de la marche d'un robot hexapode nécessite le réglage de plusieurs centaines de paramètres et le choix de la topologie d'un réseau complexe. Les modèles de ce type demandent donc, pour être réellement utilisables, qu'on leur adjoigne des algorithmes d'apprentissage permettant de fixer par l'exemple la connectivité du réseau ainsi que la majeure partie des paramètres (pondérations, seuils, ...). Cependant les différentes architectures proposées ci-dessus présentent un intérêt didactique indéniable et, à partir de l'étude des structures utilisées, nous avons pu dégager une

approche permettant l'acquisition, par l'apprentissage, de connaissances sensori-motrices et leur stockage structuré (c'est-à-dire respectant l'organisation spatiale ou temporelle des stimuli et des actions correspondantes) dans un réseau de neurones récurrent.

V.3. L'approche ascendante : une nouvelle voie vers la modularité

La structuration a priori de l'information stockée dans le réseau (c'est-à-dire l'affectation d'une « parcelle d'information » à une « partie du réseau » lors de la conception du système), si elle permet de simplifier les procédures de coopération/coordination inter-réseaux, n'est pas toujours aisée à mettre en oeuvre. Le contrôleur doit en effet avant tout être décomposé en sous-systèmes identifiés, auxquels sont attribués des réseaux neuronaux spécifiques entraînés préalablement à l'intégration finale. Or il n'est pas toujours possible d'analyser le problème de contrôle au point d'en identifier toutes les composantes et leurs interactions (on le constate, en particulier, à travers les études cognitives visant à identifier, chez l'homme, les processus conduisant à la prise de décision en milieu manufacturier [RAS 82]). Pour pallier aux approximations effectuées lors de la conception du contrôleur, la phase d'intégration des modules doit donc être accompagnée - et suivie - de processus d'adaptation et d'auto-organisation pour permettre à l'architecture de s'adapter in situ aux contraintes particulières de l'environnement. Cette approche est qualifiée de « descendante » (ou « top-down ») car la décomposition en fonctions élémentaires est réalisée a priori à partir des fonctionnalités globales recherchées [ERC 91] (figure V-8).

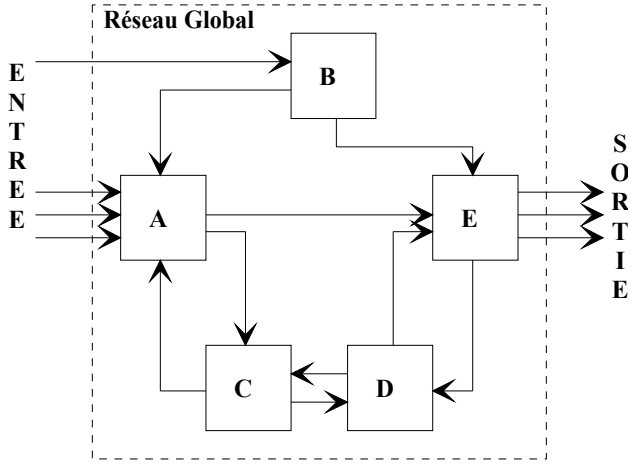


Figure V-8 : *L'approche descendante*

Le réseau global (le « contrôleur ») est construit par intégration de modules spécifiques (A, B, C, D et E) entraînés séparément pour résoudre des sous-problèmes préalablement identifiés.

L'approche ascendante permet, au premier abord, de construire des architectures connexionnistes complexes intégrant un grand nombre de modules intercommuniquant par un graphe complexe pour former un « réseau de réseaux » ou « métaréseau de neurones » [BES 92a]. Malgré son apparente simplicité, cette approche se révèle souvent extrêmement complexe à mettre en oeuvre et, si des architectures complexes incluant un grand nombre de modules ont été définies, le processus d'intégration a rarement mis en oeuvre plus de quelques unités (généralement deux ou trois modules). En effet, les modèles connexionnistes classiques se prêtent mal aux contraintes posées par l'intégration au sein d'une architecture complexe : la plupart des réseaux (en particulier les réseaux basés sur la rétropropagation du gradient) ne permettent pas de multiplier les phases d'apprentissage ce qui interdit l'adaptation post-

intégration. En outre, la création d'architectures complexes, comportant des boucles fermées (feedback) de différentes tailles, pose rapidement des problèmes de synchronisation et de dynamique qui s'ajoutent aux problèmes dynamiques inhérents aux réseaux eux-mêmes (dans le cas de modules composés de réseaux récurrents). C'est pourquoi, sans nier l'intérêt de l'approche descendante lorsque le concepteur dispose d'une bonne maîtrise de l'application et de son cycle de vie, nous ne pouvons pas l'utiliser pour le contrôle réactif : l'apprentissage est utilisé ici pour pallier la méconnaissance de la structure décisionnelle. De plus, l'environnement du contrôleur peut être amené à varier en raison de sa dynamique propre ou à cause des actions entreprises par le contrôleur lui-même (par exemple en raison du déplacement d'un robot mobile). L'apprentissage doit donc être permanent (« on-line ») ce que ne permet généralement pas l'approche descendante pour laquelle toutes les fonctions du système doivent avoir été identifiées à l'avance.

A l'opposé de cette approche, on peut chercher, à partir d'un unique réseau (sur-dimensionné), à faire émerger des îlots de réponses, c'est-à-dire à réaliser un apprentissage par partie, chaque information étant apprise par un nombre restreint de neurones et non, comme dans un réseau classique, distribuée sur l'ensemble des connexions. Plutôt que d'intégrer des réseaux de neurones en un « métaréseau » on cherche donc ici à faire apparaître, lors de l'apprentissage, des sous-réseaux spécialisés identifiés par la nature des connexions internes ou externes et non par une volonté a priori. Par opposition avec l'approche précédente et parce que cette démarche cherche à faire émerger des « sous-structures » au sein d'un réseau global, nous avons qualifié cette approche d'ascendante (« bottom-up ») [ERC 91].

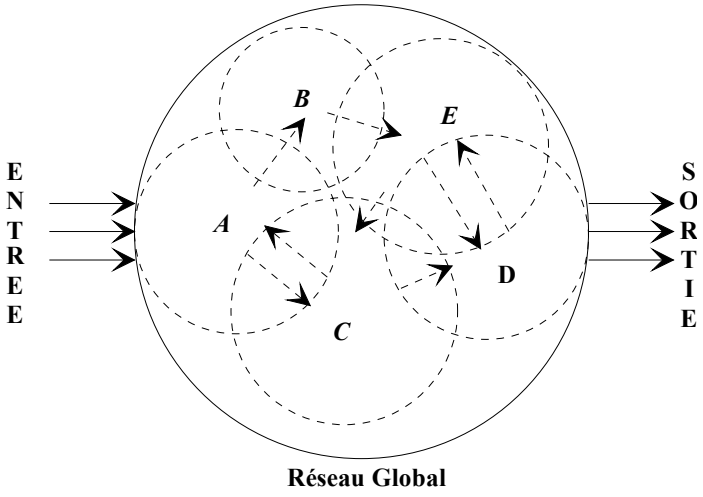


Figure V-9 : *L'approche ascendante*

A partir d'un unique réseau, on cherche à faire apparaître des îlots de réponse spécialisés ce qui permet d'éviter les interactions indésirables entre différents traitements ou différentes sources d'information.

Les structures recherchées (ou « îlots de réponse ») sont similaires à celles mises en oeuvre par Randall Beer pour la commande de son robot insectoïde : l'attribution d'une tâche à une entité particulière n'est pas due à une décomposition arborescente composée de traitements successifs et/ou parallèles mais à l'apparition, lors de l'apprentissage, d'une connectivité particulière. La décomposition en modules fonctionnels est donc une constatation effectuée a posteriori (lorsque la complexité de la structure le permet) et non une volonté a priori du concepteur. Cette approche se démarque donc de la neuroéthologie calculatoire (pour laquelle la décomposition a priori est utilisée pour faciliter la tâche du concepteur) et du modèle de Jacobs et Jordan dans lequel la décomposition modulaire doit s'adapter dynamiquement à une structure

figée. De plus la communication inter-modules ne met en oeuvre aucun mécanisme spécifique (unités ΣΠ, communication sortie/entrée figée, ...) mais, au contraire, utilise directement les connexions synaptiques utilisées pour la communication intra-modulaire. Ainsi, les différents modules peuvent échanger des informations intermédiaires et se regrouper (ou se scinder) suivant les caractéristiques de l'environnement. Par contre, une telle organisation (proche des structures épigénétiques ou fractales) ne permet pas d'isoler systématiquement différents modules fonctionnels au sein du réseau obtenu. Il est donc plus juste, dans la plupart des cas, de parler d'information semi-distribuée dans le réseau que de structure totalement modulaire.

Malheureusement les méthodes ascendantes n'ont, jusqu'à maintenant, été que très peu étudiées par la communauté connexionniste et il n'existe ni algorithme d'apprentissage ni structure neuronale adapté. La suite de ce chapitre présente les principales notions propres à cette approche (émergence homogène, assemblées de neurones, ...). Nous pourrons ainsi aborder directement, dans le prochain chapitre, les algorithmes connexionnistes spécifiques développés pour provoquer l'émergence de ces « îlots de réponse ».

V.3.1. Une nouvelle définition de l'émergence

Nous avons déjà donné, dans un chapitre précédent (section III.3.1.2), une définition de la notion « d'émergence » au sens du contrôle réactif : celle-ci était alors considérée comme la faculté d'un système sensori-moteur à interagir avec son environnement perçu, de sorte que le comportement apparent soit plus complexe que celui induit par les seules lois régissant le comportement du système. Le travail essentiel du concepteur d'un système réactif est alors de rechercher la boucle perception/action la plus appropriée à l'environnement dans lequel le système devra évoluer et à la tâche qu'il devra accomplir.

Une seconde définition de l'émergence, que l'on peut qualifier d'émergence *Homogène* (par rapport à l'émergence *Hétérogène*, c'est-à-dire produite par les relations entre système et environnement), complète le champ d'application de cette notion. Selon cette deuxième définition, l'émergence exprime l'apparition des propriétés nouvelles du fait de l'agrégation (sous-entendu, d'un grand nombre) d'éléments identiques au sein d'un ensemble [WEI 95].

Cette deuxième définition, qui n'est qu'une seconde facette d'un même phénomène, est issue des travaux menés depuis bientôt cinquante ans sur la gestion des systèmes complexes tels que les sociétés d'acteurs (sociétés économiques ou sociales), les colonies d'insectes sociaux (fourmis, termites, ...) ou les organisations cellulaires (système cortical, système immunitaire, ...).

L'émergence homogène a, elle aussi, fortement influencé les travaux sur le contrôle réactif : non seulement l'intelligence émerge de la confrontation entre le robot et son environnement, mais le comportement intelligent du robot peut aussi être une conséquence de l'action conjointe, au sein d'un même contrôleur, de nombreux modules décisionnels dédiés chacun à une fonctionnalité particulière. A ce titre, elle revêt une importance particulière pour le développement de contrôleurs réactifs connexionnistes. Nous avons vu, en effet, que les modèles neuronaux artificiels classiques, tels qu'ils ont été développés depuis 1950, sont inutilisables en raison de l'absence de structure interne intermédiaire entre la simple cellule et le réseau pris dans son ensemble. Les réseaux de neurones doivent donc être à même de développer une structure interne modulaire plus propice à l'émergence de fonctions complexes.

Dans la description des modèles connexionnistes structurés (section V.2) nous avons proposé un modèle basé sur la colonne corticale (modèle de Burnod, section V.2.1.3). Cette

approche constitue un bon exemple des structures de haut niveau pouvant être mises en oeuvre dans les modèles neuronaux artificiels. De plus, son origine neurobiologique tend à donner à cette architecture la pérennité due à l'exemple cortical et le modèle de Burnod permet une première approche de l'émergence homogène des fonctions « intelligentes ». Par contre, pour ce qui est du contrôle réactif, ce modèle souffre d'une limitation majeure : il ne propose pas d'organisation de bas niveau (cellulaire) capable de supporter les fonctionnalités d'une colonne corticale. Or, différentes études concernant des fonctionnalités sensori-motrices d'animaux peu évolués (aplysie [KAN 88], homard [SEL 83], mouche [FRA 92], ...) tendent à attribuer au niveau cellulaire un rôle au moins aussi important qu'au niveau « colonne » (au sens d'une structure autonome regroupant plusieurs centaines de neurones) et, dans de nombreux cas, des neurones associés à des états particuliers ont pu être mis en évidence. L'existence de cellules spécialisées contredit en partie les modèles de la colonne corticale, du moins pour les fonctions sensori-motrices primaires. Par contre le modèle de Burnod est certainement plus représentatif de l'organisation du cortex frontal, dédié aux activités mentales de plus haut niveau (activités supérieures aux fonctionnalités recherchées pour le contrôle réactif).

Comme nous l'avons souligné précédemment, l'approche connexionniste la plus adaptée au processus d'émergence homogène (et, par voie de conséquence, à l'approche ascendante) est probablement la neuroéthologie calculatoire de Randall Beer. Les réseaux de contrôleurs construits manuellement dans le cadre de cette approche présentent en effet toutes les fonctionnalités recherchées à l'exception de l'apprentissage. C'est pourquoi, afin de proposer un algorithme d'apprentissage adapté à cette architecture, nous proposons un modèle théorique des « sous-réseaux » mis en oeuvre dans cette approche.

V.3.2. Les assemblées de neurones

Dans notre présentation des modèles connexionnistes, nous avons souligné le fait que les réseaux de neurones artificiels, bien qu'inspirés de la structure du système nerveux animal, n'en copient qu'une petite partie c'est-à-dire la cellule nerveuse grossièrement représentée par le neurone artificiel. La connectivité des réseaux de neurones artificiels, si elle se rapproche du modèle biologique (par rapport aux arbres symboliques), en reste fort éloignée puisqu'elle n'est généralement régie que par des lois globales (réseaux en couches, réseaux totalement connectés, ...). Le système nerveux présente des structures beaucoup plus fines au sein desquelles l'information circule plus ou moins aisément sans suivre (du moins apparemment) de règles générales. Ce sont ces mêmes structures qui ont inspiré les premiers travaux de Rodney A. Brooks et le développement du « Behavior-Based-Control » ainsi que la neuroéthologie calculatoire ou les modèles de la colonne corticale. C'est pourquoi nous avons choisi de baser l'organisation macroscopique du contrôleur réactif neuronal sur des principes similaires à ceux observés par les neurobiologistes.

Plutôt que de rechercher directement à faire émerger un comportement global, « intelligent », par l'agrégation d'un grand nombre de cellules neuronales, nous avons donc cherché à développer un processus de spécialisation des cellules neuronales en « sous-réseaux » dont l'agrégation provoque, ensuite, l'émergence des fonctionnalités recherchées. Ce type de sous-réseau, ou « assemblée de neurones[59] », a été historiquement proposé par Donald O. Hebb [HEB 49] comme un ensemble de cellules pouvant présenter, suite à une stimulation une activité

[59] Traduction littérale du terme américain « Neural Assemblies »

indépendante subsistant brièvement après la fin de cette stimulation[60]. Cette définition, si elle est probablement partiellement vérifiée, n'est pas suffisante pour définir la structure des assemblées de neurones et pour expliquer leur formation au cours du processus d'apprentissage.

V.3.2.1. Aspect neurobiologique des assemblées de neurones

L'étude des assemblées de neurones présente, pour la neurophysiologie, de multiples intérêts. En effet, ces organisations, relativement indépendantes du reste du système nerveux, permettent d'étudier les mécanismes neuronaux de l'apprentissage (habituation/sensibilisation) tout en limitant les interactions avec d'autres cellules neuronales [KAN 88]. De plus, ce type de structure (circuits neuronaux CPG[61]) a probablement un rôle essentiel pour la génération des principaux rythmes moteurs [MEY 91] et plusieurs auteurs s'accordent pour voir dans les assemblées de neurones un niveau fondamental du traitement de l'information par le cortex, en particulier pour les opérations sensori-motrices élémentaires [DIN 90]. C'est pourquoi, en particulier depuis la fin des années quatre-vingts, les assemblées de neurones ont fait l'objet de recherches poussées. Notre but n'est pas, ici, de décrire ces travaux de façon détaillée (certaines caractéristiques particulières des assemblées de neurones seront plus précisément exposées au cours du chapitre suivant) et nous ne les exploitons ici que pour proposer une « définition » des assemblées de neurones fondée sur la neurobiologie.

Si la plupart des auteurs considèrent les assemblées de neurones comme un élément essentiel des systèmes nerveux biologiques, rares sont ceux qui donnent une définition précise de ces structures : elles sont la plupart du temps évoquées en termes vagues (« ganglions », « groupes de neurones » ou même « réseaux de neurones ») peu explicites quant à leurs caractéristiques exactes.

Dans un article de vulgarisation, Laure Schalchli [SCH 91] différencie les assemblées de neurones par la connectivité intra-assemblée : plus « forte » que la connectivité inter-assemblées, elle permettrait aux sous-réseaux de présenter une activité interne cohérente, c'est-à-dire que tous les neurones inclus dans une assemblée déchargeraient suivant un rythme bien défini. Cette définition, proche de celle proposée par D.O. Hebb, n'est cependant pas totalement satisfaisante : si on s'en tient à la notion de rythme interne, plusieurs études ont montré qu'un même neurone pouvait appartenir, à différentes périodes, à plusieurs assemblées différentes [HOO 89], voire même qu'un réseau (i.e. un rythme d'activité) peut apparaître spontanément de la fusion de deux autres assemblées de neurones puis disparaître [MEY 91]. Les assemblées de neurones ne constituent donc pas des entités fixes, caractéristiques d'une structure figée. Des études ultérieures ont, par ailleurs, montré que le remaniement de la structure pouvait être lié à l'activité d'un ou plusieurs neurones et, en particulier, à l'activité de neurones sensoriels [COM 94].

---

[60] « Any frequently repeated, particular stimulation will lead to the slow development of a « cell-assembly », a diffuse structure comprising cells in the cortex and diencephalon (an also, perhaps, in the basal ganglia of the cerebelum), capable of acting briefly as a closed system, delivering facilitation to other such systems and usually having a specific motor facilitation. » ([HEB 49], Introduction, p. XIX)

« It is proposed first that a repeated stimulation of specific receptors will lead slowly to the formation of an « assembly » of association-area cells which can act briefly as a closed system after stimulation has ceased; this prolongs the time during which the structural changes of learning can occur and constitutes the simplest instance of a representative process (image or idea). » ([HEB 49], chapitre 4., p. 60)

[61] Central Pattern Generator

Dans la majeure partie des cas, les assemblées de neurones biologiques sont identifiées grâce à l'étude fine de l'activité de chacune des cellules nerveuses (dans le cas du système stomato-gastrique du homard, qui a servi de support à la plupart des travaux cités, l'étude des assemblées de neurones a été postérieure à l'identification précise des fonctionnalités de chaque cellule [SEL 83]). Celles-ci sont ensuite regroupées en assemblées suivant leur participation à l'une ou l'autre des fonctionnalités du réseau global. Les assemblées de neurones sont donc avant tout des structures sémantiques associées, par l'observateur humain, à une fonction précise (organique, physique, intellectuelle, ...). Ces différentes fonctions n'étant pas continues dans le temps mais malgré tout intimement liées les unes aux autres, les assemblées de neurones ne peuvent pas être des entités fixes mais *doivent* évoluer au cours du temps. En conséquence les rythmes d'activités qui les caractérisent (i.e. ceux des fonctions régulées et/ou ceux des stimuli observés) *ne peuvent pas* être stables au cours du temps. Dans un des premiers articles montrant l'évolutivité des assemblées de neurones ([HOO 89]), S.L. Hooper propose une approche similaire en basant la décomposition d'un réseau en assemblées de neurones sur la décomposition fonctionnelle du comportement global[62]. Hubert Dinse propose lui aussi une définition fonctionnelle des assemblées de neurones [DIN 90] : constatant l'évolutivité des cartes sensori-motrices observées dans le système nerveux central des primates (en particulier celles concernant la représentation neuroanatomique de la main), il lie l'existence des assemblées de neurones avec les relations action motrice/stimuli (boucle de feedback externe). L'auteur utilise d'ailleurs, pour définir les assemblées de neurones, le terme de « groupe fonctionnel ».

V.3.2.2. Un équivalent connexionniste

Telles que nous venons de les décrire, les assemblées de neurones biologiques n'ont pas d'équivalent connexionniste. Elles représentent en effet un intermédiaire entre la représentation localisée de l'information (de type carte cognitive) et l'information totalement distribuée (de type réseau en couches ou réseau totalement connecté), c'est-à-dire entre les réseaux dont la topologie est fixée a priori en fonction de l'information codée et ceux dont la topologie est uniquement soumise à des lois générales d'organisation sans relation avec l'information.

Comme nous l'avons vu précédemment il y a un intérêt certain à « imiter » dans les réseaux de neurones formels les structures que sont les assemblées de neurones. Celles-ci permettent de décomposer le comportement global en éléments comportementaux distincts, éventuellement actifs simultanément. Une première approche de ces structures nous a conduits à rechercher, dans les réseaux de neurones, une topologie basée sur deux niveaux de poids différents : des pondérations intra-assemblée fortes et des pondérations inter-assemblées faibles. Ainsi l'influence d'une assemblée de neurones sur le reste du réseau est soumise à la coordination de l'activité des neurones internes. Cependant nous avons progressivement remis cette approche en question : les informations neurobiologiques dont nous disposons nous montrent qu'il n'y a pas de frontières (en terme de topologie et/ou de pondération des connexions) clairement définies entre les différentes assemblées de neurones. En effet, comme nous l'avons vu dans le paragraphe précédent, elles peuvent fusionner ou, au contraire, se scinder, sous l'influence d'un unique neurone (donc, a priori, d'une connexion suffisamment pondérée pour pouvoir contrarier l'état d'un groupe de neurones fortement interconnectés). Les assemblées de neurones ne sont donc pas uniquement liées à des contraintes topologiques qui n'expriment pas directement le caractère sémantique de la différenciation entre les différentes cellules.

---

[62] « Many behaviors involve the coordination of several different groups of neurons, known as neural networks, with each network generating specific aspects of the total behavior » ([HOO 89], page 1587)

Pour construire un réseau de neurones capable de se structurer en assemblées de neurones on doit donc disposer d'un algorithme d'apprentissage permettant d'extraire puis de coder localement (c'est-à-dire sur un petit nombre de cellules) l'information sémantique contenue dans les stimuli reçus et dans leurs éventuelles corrélations avec les actions motrices entreprises (contraintes de précédence, d'exclusion, ...). Les assemblées ne sont donc pas des structures clairement identifiées (comme c'est le cas dans la plupart des réseaux connexionnistes modulaires) mais des groupes fonctionnels déterminés a posteriori par l'étude des différentes cellules neuronales composant le réseau.

V.4. Conclusion : vers une organisation « Neuro-Réactive »

Les paragraphes précédents nous ont permis de préciser la structure neuronale mise en oeuvre pour la réalisation d'un contrôleur réactif basé sur un processus connexionniste. Dans le cadre de l'architecture de contrôle présentée dans le chapitre II, cette structure nous permet de réaliser l'interface entre le niveau « cellulaire » (neuronal) et le « haut niveau », le contrôleur lui-même. En outre, elle permet de relier les architectures neuronales et le « Behavior-Based-Control » : un contrôleur connexionniste basé sur des principes de localisation fonctionnelle (dans des assemblées de neurones distinctes) serait en effet conforme aux spécifications proposées pour un algorithme d'apprentissage adapté au contrôle réactif (pas d'utilisation d'un modèle explicite, modularité, architecture sensori-motrice).

La réunion des architectures réactives et neuronales nous conduit à la conception d'un contrôleur basé sur trois niveaux sémantiques successifs (figure V-10) auxquels on peut ajouter, dans le cas d'une application de pilotage décentralisé en atelier, un quatrième niveau regroupant les contrôleurs locaux de l'ensemble des ressources. Le passage d'un niveau au suivant s'accompagne d'un processus d'émergence conduisant à l'apparition de fonctions de plus en plus évoluées.

- Au niveau inférieur (niveau cellulaire), l'information (les stimuli) est soumise à un réseau de neurones qui associe les stimuli et les actions motrices. Ce réseau est entraîné au moyen d'un algorithme d'apprentissage spécifique permettant l'émergence de groupes fonctionnels identifiables. Si l'environnement l'impose (problème de « Time-Credit-Assignment »), le réseau peut être inclus dans une architecture de type « critique adaptative ». A ce niveau l'existence de connexions récurrentes est indispensable pour permettre au réseau de réaliser des séquences d'action ou de conserver une mémoire des couples perception/action antérieurs [GAL 92]. En outre, le réseau devra être capable de s'adapter aux évolutions partielles de son environnement sans remettre en cause l'ensemble des informations stockées. Il ne devra donc pas être sensible au phénomène d'oubli catastrophique.

- Le niveau intermédiaire regroupe l'ensemble des groupes fonctionnels ainsi que certaines cellules neuronales isolées (leur état étant déterminant pour l'activité de l'ensemble du réseau, ces neurones constituent des « entités fonctionnelles mono-cellulaires »). C'est donc à ce niveau que sont produits les premiers comportements identifiables (réflexes sensori-moteurs). Ils sont produits indépendamment par les différentes assemblées de neurones qui correspondent aux modules élémentaires utilisés dans l'architecture réactive de type « Subsumption ». De la juxtaposition d'un grand nombre de comportements élémentaires émerge le comportement global du système, observable au niveau supérieur.

- Le troisième niveau est le niveau contrôleur lui-même dont le comportement global est la conséquence directe de la complexité des niveaux inférieurs (émergence homogène) et de

leurs interactions avec l'environnement (émergence hétérogène). A ce niveau l'architecture, bien que basée sur une approche connexionniste, est proche du modèle de Brooks (section III.3.2) : les premiers comportements qui apparaissent au cours de l'apprentissage servent de substrat à leurs successeurs. De plus, comme pour l'architecture « Subsumption » les principes organisationnels intra-modulaires et inter-modulaires sont identiques. Dans l'architecture originale les modules sont composés d'AFSM échangeant des messages internes ou externes. De même, dans notre modèle, les modules sont composés de neurones (éventuellement un seul) et communiquent par connexions synaptiques intra-assemblée ou inter-assemblées.

- Dans le cadre d'une application industrielle (par exemple un système de transport), le système global représente un quatrième niveau : le niveau multi-agents qui, par interaction entre les différents contrôleurs (chacun d'entre eux percevant les autres comme inclus dans son environnement), doit répondre aux sollicitations des systèmes de gestion de production/planification afin de mener à bien la tâche qui lui est impartie.

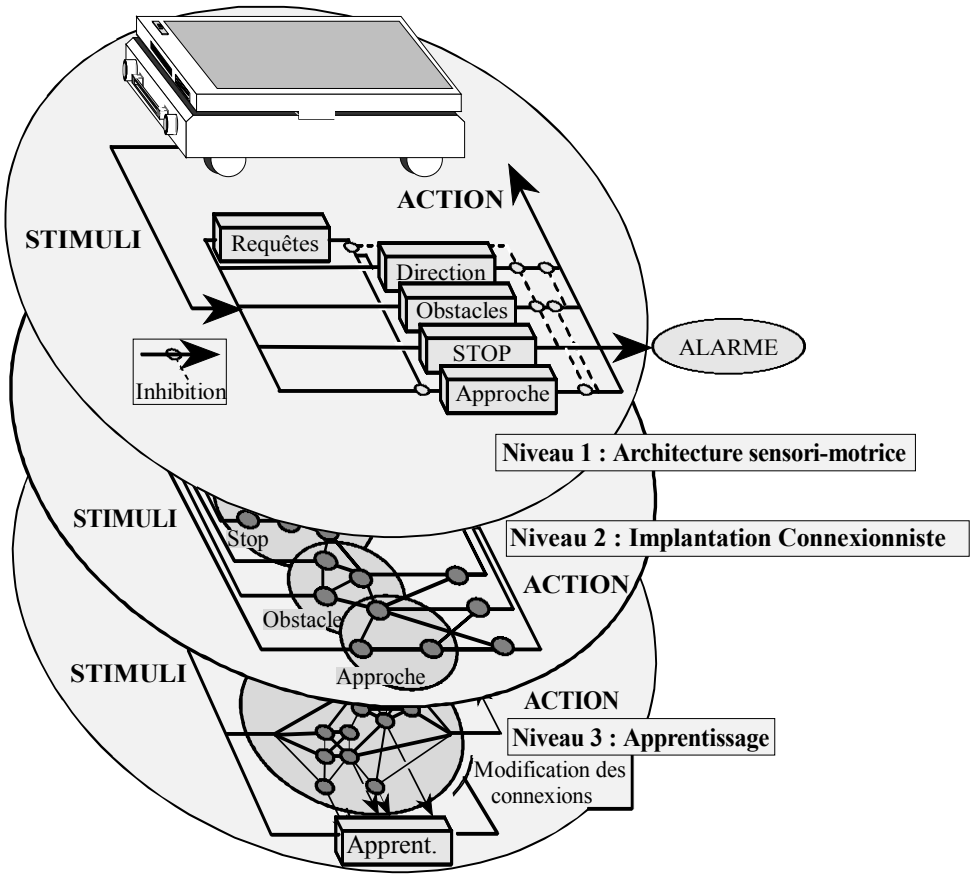


Figure V-10 : *Niveaux hiérarchiques d'un contrôleur « Neuro-Réactif »*

Cette architecture décomposée en trois (ou quatre) niveaux permet d'unifier les différentes approches suivies tout au long de notre travail, c'est-à-dire la productique, le contrôle réactif et les méthodes neuromimétiques (figure V-11). En outre, hormis l'intérêt pratique que revêt cette unification, il nous paraît important, d'un point de vue cognitif, de noter que les différents niveaux hiérarchiques sont inspirés des sciences de la vie (éthologie, biologie et neurobiologie). Cette architecture permet d'unifier des techniques informatiques issues de différents niveaux d'observation des modèles neurobiologiques : du niveau microscopique pour les modèles neuromimétiques jusqu'au niveau macroscopique et comportemental pour le contrôle réactif et l'approche multi-agents.

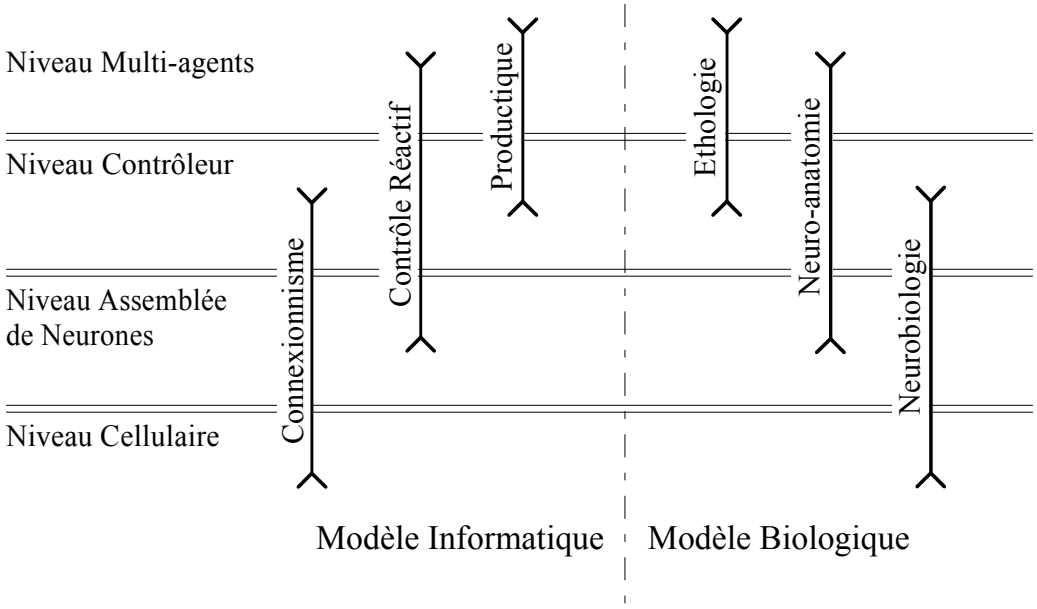


Figure V-11 : *Niveaux d'organisation des architectures de contrôleur inspirés des modèles biologiques.*

Les capacités cognitives des systèmes biologiques émergent par couches successives depuis le niveau cellulaire jusqu'au niveau de la « société ». On retrouve la même structuration par niveaux dans les modèles artificiels qui s'en inspirent mais il manque aux sciences de l'artificiel un « liant » permettant d'unifier ces différents niveaux. Les assemblées de neurones permettent d'unifier les modèles de contrôleurs inspirés de la neurobiologie (modèles neuromimétiques et neurocontrôle), ceux d'inspiration neuroanatomique (contrôle réactif) et les sociétés multi-agents issus, entre autres, de l'éthologie.

L'unification des différents niveaux en une architecture complète reste cependant soumise à la capacité du niveau cellulaire à s'auto-organiser en groupes fonctionnels. Or nous avons vu, à plusieurs reprises, que les modèles neuromimétiques n'ont pas, à l'heure actuelle, cette propriété. C'est pourquoi, nous allons, dans le chapitre suivant, proposer un algorithme connexionniste basé sur l'approche ascendante. Afin de préserver l'uniformité du modèle, cet algorithme repose sur des principes organisationnels proches du « Behavior-Based-Control » et de l'architecture « Subsumption ». En outre, il est librement inspiré des recherches en neurobiologie (en particulier les synchronisations temporelles observées dans les assemblées de neurones) ce qui permet de conserver, à tous les niveaux, l'analogie entre le modèle informatique et le modèle cognitif/neurobiologique.

CHAPITRE VI :

UNE SOLUTION CONNEXIONNISTE

RECURRENTE ET MODULAIRE

Les chapitres précédents nous ont permis de mettre en place une architecture de pilotage décentralisée pour les systèmes de production et de poser les premières bases des contrôleurs locaux utilisés. L'ensemble de ce travail repose donc maintenant sur la définition d'une architecture « Neuro-Réactive ».

Dans ce dernier chapitre nous proposons une architecture connexionniste librement inspirée de modèles neurobiologiques développés ces dernières années. Cette architecture repose sur un stockage par chemins de l'information (Path Coding) tandis que les modèles classiques privilégient généralement un codage global sur la matrice des connexions (Pattern Coding). Les chemins d'activation sont isolés grâce à un nouveau mode de gestion du temps dans les réseaux de neurones.

Après avoir défini en détail l'algorithme proposé, nous détaillerons les différents tests de validation ce qui nous permettra de montrer que l'architecture décrite répond bien aux exigences conjointes du connexionnisme et du contrôle réactif.

Une Solution Connexionniste Récurrente et Modulaire

VI.1. Introduction

Les chapitres précédents nous ont permis de construire progressivement une architecture de contrôleur neuronal basée sur différents niveaux d'organisation. Cette architecture repose sur des principes d'émergence homogène ou hétérogène introduits séparément en intelligence artificielle au cours des dix dernières années dans deux disciplines différentes : le connexionnisme et le contrôle réactif. L'unification de ces deux approches est obtenue par l'utilisation de réseaux modulaires, construits au moyen d'une spécialisation ascendante d'un réseau de neurones en groupes fonctionnels : les assemblées de neurones. La clef de voûte de cette architecture est donc le réseau de neurones lui-même et son aptitude à se scinder, suivant les stimuli perçus et leurs relations avec les actions motrices entreprises, en modules identifiables dédiés à des composantes particulières du comportement global. Nous avons décrit dans le chapitre précédent plusieurs architectures modulaires mais, soit en raison de leur rigidité structurelle (topologie figée), soit par manque d'algorithme d'apprentissage, aucune d'entre elles ne s'est révélée adaptée pour la construction de contrôleurs « Neuro-Réactifs ». C'est pourquoi nous avons pris le parti de construire un algorithme connexionniste spécifique pour l'approche ascendante. Le présent chapitre est consacré à la description de cet algorithme dit « mode ordonnancé ».

Le mode ordonnancé est librement inspiré des modèles neurobiologique. En effet, l'utilisation des travaux des neurobiologistes nous a semblé intéressante à plusieurs points de vue :

- Nous cherchons à unifier deux approches qui sont elles-mêmes directement inspirées de la biologie mais à des degrés d'analyse différents. D'où l'idée de rechercher dans la biologie elle-même la liaison entre le contrôle réactif et le connexionnisme.

- Les modèles connexionnistes, malgré leurs origines corticales, ne présentent pas, comme celle-ci, la capacité de se structurer par l'apprentissage. C'est pourquoi nous avons cherché à mettre en évidence une ou plusieurs différences cruciales entre le modèle biologique et sa copie informatique, différences susceptibles d'interdire l'émergence des assemblées de neurones.

Avant de décrire en détail le mode ordonnancé nous allons donc proposer une analyse succincte des modèles biologiques utilisés afin de présenter les bases qui nous ont permis d'élaborer notre architecture.

VI.2. Les modèles biologiques : spécialisation et synchronie

Nous avons déjà évoqué, à plusieurs reprises, la structure du système nerveux central, que ce soit au niveau cellulaire (section IV.3.2.1) ou, dans le chapitre précédent, au niveau des

groupes fonctionnels. Dans le présent chapitre nous nous intéresserons plus précisément aux caractéristiques temporelles du traitement neuronal de l'information. Le facteur temps semble en effet, de l'avis des neurobiologistes, avoir une influence prépondérante sur l'apprentissage. C'est pourquoi nous allons, dans un premier temps, décrire le codage temporel de l'influx nerveux avant d'étudier son impact sur la transmission et le traitement de l'information. Nous verrons ensuite l'influence du facteur temps sur la spécialisation des « groupes fonctionnels » et sur les phénomènes d'apprentissage propres au système nerveux.

VI.2.1. Introduction : le codage impulsionnel de l'influx nerveux

Au sein du système nerveux l'information véhiculée le long des axones puis transmise, via les synapses, aux dendrites est codée sous la forme d'un signal électrique, provoqué par des échanges ioniques membranaires. La transmission synaptique, par contre, s'effectue par voie chimique au moyen de neurotransmetteurs dont les plus connus sont l'acétylcholine et la sérotonine. Les échanges ioniques ne permettent pas à la cellule nerveuse de maintenir une activité électrique constante. Les neurones génèrent donc des impulsions électriques successives, les potentiels d'action, circulant, sous la forme de trains, des dendrites vers le corps cellulaire puis vers l'extrémité de l'axone. Ce mode de transmission de l'information n'est pas le seul à être mis-en-oeuvre dans le cortex : d'autre mécanismes peuvent être utilisé (par exemple un codage en intensité sur certaines cellules du cortex visuel).

Bien qu'impulsionnel, l'influx nerveux transmis par les neurones conserve un aspect quantitatif : on admet communément que la fréquence des potentiels d'action permet aux neurones de coder une valeur numérique correspondant à l'intensité du signal émis sur l'axone (certaines cellules nerveuses particulières - comme les neurones sensoriels de la rétine - utilisent un codage basé sur des potentiels gradués). De fait, la fréquence de décharge influe directement sur la quantité de neurotransmetteurs libérée au niveau des synapses et donc sur l'intensité du signal reçu par les neurones successeurs. Les contraintes physiques limitent la fréquence de décharge aux environs de 700 Hz, mais, dans des conditions normales, les trains de potentiels contiennent généralement moins de cent impulsions par seconde soit une fréquence d'activité comprise entre 0 et 100 Hz. Notons que certains neurones présentent une activité spontanée dont la fréquence peut atteindre 70 Hz dans certaines aires cérébrales.

Ce mode de codage de l'intensité du signal nerveux n'est pas sans influence sur le traitement de l'information. En effet, même si ce codage permet, théoriquement, de représenter une infinité de niveaux de stimulation (en comptant le nombre d'impulsions sur une fenêtre de temps suffisamment large), la fenêtre d'observation est limitée en pratique d'une part par les contraintes du temps réel et d'autre part, par le flux permanent d'information circulant le long des terminaisons nerveuses. Ce flux risque de provoquer, lorsqu'on allonge la fenêtre d'observation, un chevauchement des informations, a priori incompatible avec la précision des traitements corticaux. Devant la rapidité du traitement cortical de l'information, il est peu probable que cette fenêtre dépasse 100 ms (le système visuel, en particulier, est capable d'isoler des images séparées par des intervalles de temps inférieurs à un dixième de seconde). Or, en 100 ms un neurone ne peut émettre, au maximum, que 10 potentiels d'action, c'est-à-dire qu'il peut transmettre une information codée sur 10 niveaux au maximum, sous réserve que le neurone aval soit capable de compter avec précision le nombre d'impulsions reçues, ce qui est peu probable.

Cette caractéristique particulière des réseaux de neurones biologiques est loin d'être mise en oeuvre chez leurs homologues artificiels : à mi-chemin entre un codage binaire et un codage à valeurs réelles (seule alternative proposée par les algorithmes connexionnistes), les cellules nerveuses ne sont, apparemment, capable de coder l'information que sur une dizaine de niveaux

d'activité. En pratique il semble même réaliste de considérer que, compte tenu des délais mis en oeuvre et de leur faible précision, les neurones ne peuvent coder l'information que sur deux ou trois niveaux [THO 88b].

VI.2.2. Codage impulsionnel et information temporelle

Le système visuel des mammifères (en particulier des primates et de l'homme) est une composante particulière du cortex dont on mesure, introspectivement, la puissance de calcul. En effet des problèmes visuels complexes, tels que la reconnaissance d'un visage ou d'une scène, demandent moins de 100 ms de traitement. Or le traitement des images rétiniennes dans le cerveau met un oeuvre un traitement multicouches de l'information qui, bien que fortement parallèle au niveau de chaque couche de traitement, implique, au moins, une dizaine de traitements séquentiels pour traverser les différentes aires visuelles qui séparent l'image initiale du cortex visuel proprement dit (aires V1 à V5, aire NT, ...). Ces aires effectuent des traitements complexes de détection de contours, d'orientations, de mouvements, etc. Chaque couche doit donc être traversée en 10 ms au maximum. Même en émettant des trains de potentiels à la fréquence maximale (100 Hz), les cellules neuronales ne peuvent recevoir (et émettre), dans ce délai, qu'un seul potentiel d'action. L'information « intensité lumineuse », qui est, a priori, transmise tout au long de la chaîne visuelle, doit donc être véhiculée autrement que par le codage fréquentiel.

Ce constat d'insuffisance du codage « intensité/fréquence » a conduit plusieurs auteurs à proposer, pour le traitement visuel, un codage intensité/délai. Selon cette théorie, l'information « intensité » est codée par le délai temporel séparant les potentiels d'action des différents neurones. Le principe du codage « intensité/délai » repose sur un modèle de neurone dit « integrate and fire[63] » dont le seuil de décharge est plus ou moins rapidement atteint suivant l'intensité de la stimulation reçue. Pour chaque couche de traitement, l'arrivée d'un stimulus visuel provoque une suite de décharges asynchrones dont les décalages permettent de coder les différences d'intensité relatives entre les différents éléments de l'image. Cette hypothèse est corroborée par plusieurs observations neurophysiologiques montrant, dans les aires visuelles, des différences de temps de réponse pouvant atteindre plusieurs dizaines de millisecondes suivant certaines caractéristiques particulières de stimulus [TRO 92b]. Il n'est cependant pas possible de préjuger de l'information transmise : les études conduites à Toulouse par Simon Thorpe montrent que l'ordre d'arrivée des potentiels d'action représente déjà un codage très efficace, et qu'il n'est pas forcément nécessaire de dater précisément leur réception par les couches successives [THO 94].

Si le codage intensité/délai reste relativement simple à appréhender (et, éventuellement, à mettre en oeuvre), plusieurs auteurs ont proposé une analyse différente et attribuent au codage impulsionnel une fonction de plus haut niveau. Le neurobiologiste allemand Von der Malsburg [MAL 90b] a proposé une théorie selon laquelle la synchronisation temporelle entre les potentiels d'action de neurones appartenant à des aires visuelles différentes caractérise l'appartenance de ces neurones à un même objet. Des décharges simultanées permettraient, selon cette théorie, de lier sémantiquement les représentations multimodales d'un unique objet de la scène visuelle. De fait, les synchronisations semblent être liées à des critères perceptifs proches de la Gestalttheorie [GUI 37] [SIN 93].

---

[63] intégration puis décharge

Depuis quelques années, grâce à l'utilisation de micro-électrodes permettant l'enregistrement in-vivo des potentiels d'action au niveau d'une cellule, les neurobiologistes ont multiplié les observations corroborant cette théorie. Plusieurs équipes ont ainsi pu mettre en évidence des synchronisations d'activité entre neurones proches puis entre des cellules distantes (séparées par plusieurs millimètres) dont les champs récepteurs ne se recouvrent pas. De plus, ces travaux ont montré que la synchronisation caractérise une propriété globale du stimulus. Ainsi C.M. Gray (à l'institut Max-Planck de Francfort) a montré que deux stimuli distincts (deux barres verticales alignées) ne provoquent de synchronisation que lorsqu'ils se déplacent dans la même direction et que cette synchronisation est maximale lorsque les deux barres sont remplacée par une barre unique de même dimension (figure VI-1) [GRA 89].

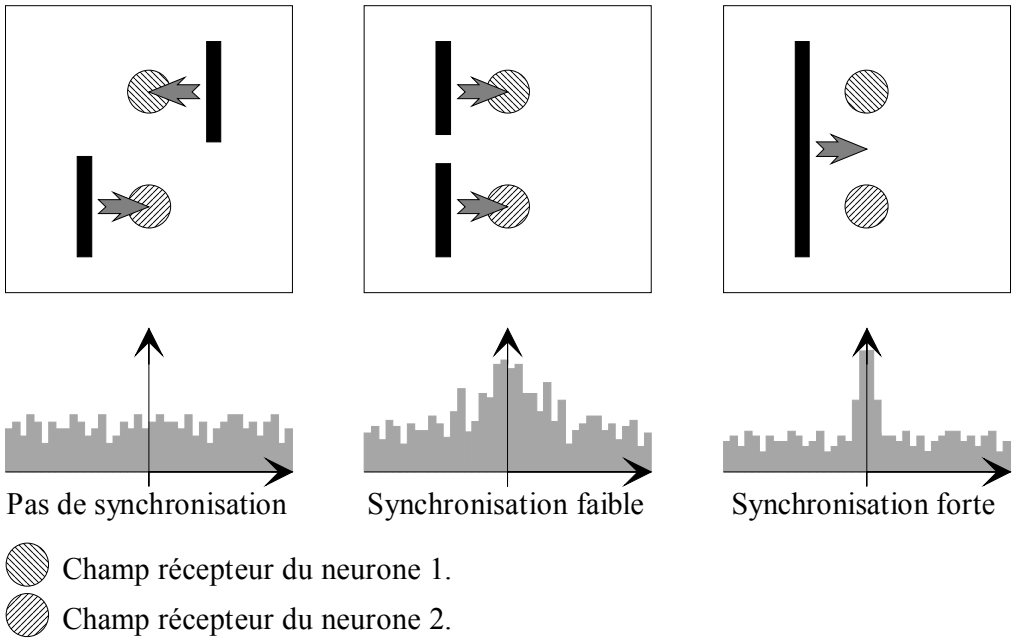


Figure VI-1 : *Synchronisation entre deux neurones du cortex visuel.*

La synchronisation entre les potentiels d'action émis par deux neurones distincts (séparés de 7 mm) est directement liée aux caractéristiques globales du stimulus. Les courbes représentent les histogrammes de corrélation temporelle entre les deux neurones (d'après [GRA 89] et [BUL 92]).

Plus récemment, des travaux ont montré que les synchronisations observées entre des neurones distants sont directement soumises à l'activité de neurones appartenant à des aires visuelles différentes. Ainsi l'activité des neurones de l'aire V1 est modulée par l'aire PMLS (chez le chat) [BUL 92], d'où l'idée que des aires particulières du système visuel coordonnent, via les synchronisations, l'activité des différentes représentations d'un même objet visuel afin de lier entre elles ses différentes composantes (couleur, déplacement, orientation, ...) tout au long du traitement multimodal de l'image visuelle (« Binding Problem »). D'autre part, alors que dans un premier temps les phénomènes de synchronisation n'avaient été observés que dans le cortex visuel, ils ont été récemment observés dans plusieurs autres aires cérébrales dont le cortex moteur de singes [SIN 93]

D'après les auteurs cités ci-dessus, la synchronisation des potentiels d'action entre différents neurones serait intimement liée avec l'existence des assemblées de neurones. De fait, les neurones d'une même assemblée présentent souvent des synchronisations fortes. Cependant, l'origine et la fonction de ces oscillations synchrones sont encore mal déterminées. L'existence

de mécanismes d'apprentissage basés sur l'activité simultanée des cellules neuronales (apprentissage hebbien), qui conduit à renforcer les connexions entre neurones synchrones, pourrait expliquer ce phénomène (la synchronisation, en favorisant l'apprentissage, serait alors à l'origine du regroupement de certaines cellules en groupes fonctionnels [SIN 93]). Certains auteurs voient dans la synchronisation un mécanisme permettant de moduler l'activité des assemblées neuronales sur leurs cellules cibles, ce qui permettrait un codage rapide de l'information (basé sur le degré de synchronisation des neurones d'une même assemblée) [BUL 92].

VI.2.3. Caractérisation temporelle des assemblées de neurones

La plupart des comportements moteurs observés aussi bien chez l'homme que chez l'animal mettent en oeuvre de nombreux rythmes moteurs régulés par des fonctions de perception (éventuellement somato-sensorielles). Ces différents rythmes sont généralement différenciés bien qu'interdépendants. Au cours du chapitre précédent, nous avons montré que le cortex présente une structure modulaire dans laquelle des groupes fonctionnels, les assemblées de neurones, génèrent séparément des aspects spécifiques du comportement global et que celui-ci émerge de la *coopération coordonnée* de ces différents groupes. En outre, nous avons proposé, sous réserve de développer un algorithme adapté, d'utiliser une architecture similaire pour construire des contrôleurs « Neuro-Réactifs ».

Des études neurophysiologiques récentes (voir les paragraphes précédents) montrent l'influence prépondérante, bien que souvent négligée, du facteur temps dans la transmission de l'influx nerveux. En outre les mécanismes de synchronisation temporelle semblent intimement liés avec le regroupement fonctionnel des cellules nerveuses (soit comme origine, soit comme conséquence). C'est pourquoi, pour le développement de notre algorithme, il nous a semblé opportun de détailler ici le comportement temporel des assemblées de neurones biologiques.

L'étude neurophysiologique des assemblées de neurones biologiques nécessite l'identification préalable de l'ensemble des cellules nerveuses participant à la fonction décrite. C'est pourquoi ces travaux n'ont été rendus possibles que très récemment et n'ont concerné que des fonctions primaires observées chez des animaux peu évolués (calmar géant, aplysie, homard, ...). Chez ces animaux, les cellules nerveuses sont en effet moins nombreuses et plus grosses que chez les animaux évolués ce qui facilite d'autant les expérimentations. Les travaux que nous rapportons ici ont été effectués par l'équipe du professeur M. Moulin (Laboratoire de Neurobiologie et de Physiologie Comparée, à Arcachon) sur la régulation de l'appareil stomatogastrique du homard [HOO 89] [MEY 91][COM 94].

Les neurones responsables de la régulation des mouvements du tube digestif chez le homard sont bien connus depuis plus de 10 ans et l'existence de mouvements rythmiques clairement identifiés a permis l'identification de quatre groupes fonctionnels (CPG : Central Pattern Generators[64], figure VI-2a) dédiés à la production de l'un ou l'autre des rythmes moteurs (rythmes gastrique, pylorique, oesophageal et cardiaque). Ces rythmes sont générés par deux processus distincts : réseau oscillateur ou activité endogène de neurones « Pacemakers ». Chacune des quatre assemblées de neurones est alors considérée comme une structure immuable, fonctionnellement identifiée par le rythme partagé par l'ensemble des cellules qui la compose (figure VI-2b).

---

[64] Générateurs Rythmiques

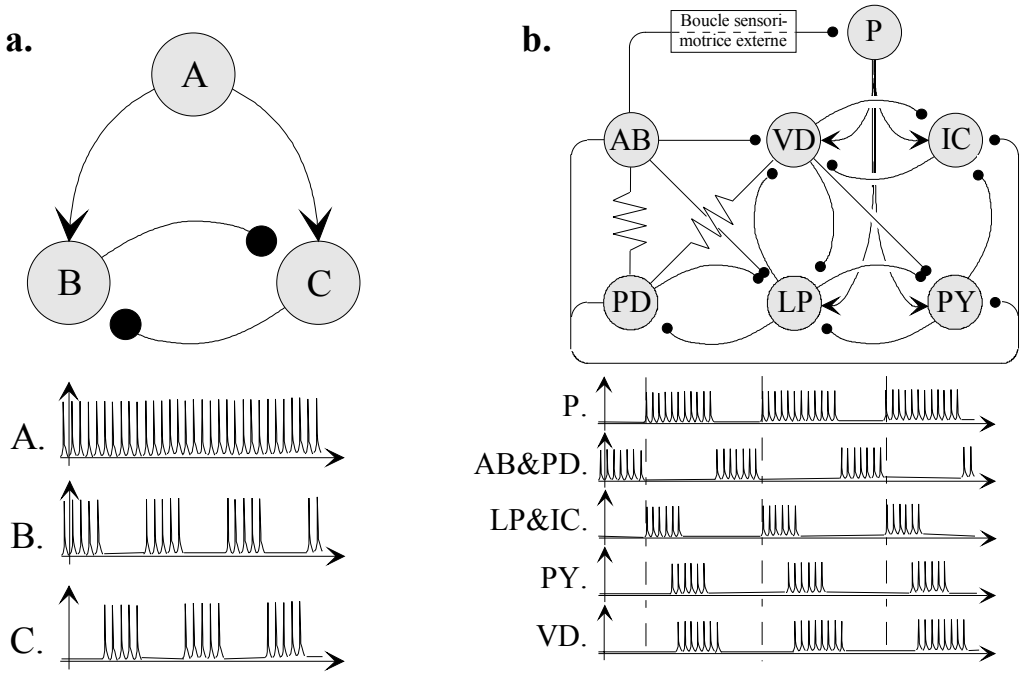


Figure VI-2 : *Génération d'un rythme moteur par une assemblée de neurones.*

a. : Principe général. Les connexions « flèches » correspondent aux activations et les connexions « boules » aux inhibitions (ces deux types de synapses sont chimiques). Les neurones B et C constituent un oscillateur entretenu et régulé par le neurone A.

b. : Le réseau Pylorique chez le homard (les résistances correspondent aux synapses « électriques »). Les connexions reliant ce réseau aux autres assemblées de neurones ne sont pas représentées sur le schéma. Tous les neurones présentent un rythme similaire (rythme pylorique, ligne pointillée) et déchargent à la même fréquence, ce qui est caractéristique de l'appartenance à une même assemblée. Les neurones PD et PY, représentés comme des cellules uniques, sont en réalité au nombre de 2 et 8. On remarquera les similitudes de principe avec le réseau de Randall Beer (figure V-3, section V.2.1.1).

A première vue, le système stomatogastrique du homard est donc contrôlé par des assemblées de neurones clairement identifiées (bien que fortement interconnectées) générant les différents rythmes moteurs. Cette analyse est confirmée par des études portant sur d'autres animaux et sur la régulation de différentes fonctions vitales. De fait, il est généralement admis que les « CPG » sont constitués d'un petit nombre de neurones interconnectés suivant une topologie bien définie et qu'ils sont relativement indépendants les uns des autres. Cette première analyse, favorisant l'indépendance des modules entre eux, est très proche de la plupart des modèles connexionnistes modulaires (en particulier du modèle de Burnod [BUR 88]). Cependant des analyses plus fines viennent contredire largement l'identification fonctionnelle immuable des assemblées de neurones.

Dès 1989, l'équipe du professeur Moulin a pu montrer que certaines cellules nerveuses, bien qu'appartenant à un groupe fonctionnel précis, peuvent, transitoirement, adapter leur rythme d'activité à celui d'une autre assemblée de neurones. Ainsi le neurone « VD », qui présente normalement un rythme pylorique (figure VI-2) se rattache au réseau cardiaque lorsque sous l'impulsion d'un neurone sensoriel celui-ci devient actif (figure VI-3) puis se rattache au réseau pylorique lorsque la stimulation cesse [HOO 89]. Les neurones n'appartiennent donc pas en permanence à la même assemblée de neurones. Autrement dit, les assemblées de neurones ne

communiquent pas uniquement par l'intermédiaire de connexions synaptiques : des neurones particuliers peuvent, dans certains cas, servir d'intermédiaires.

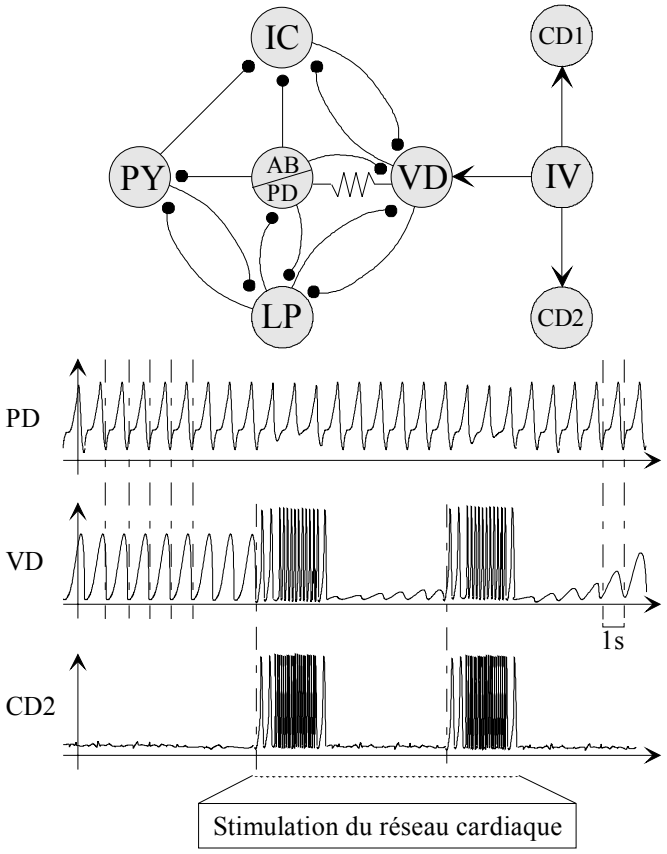


Figure VI-3 : *Relation entre le réseau Pylorique et le réseau Cardiaque.*

Le neurone VD, soumis à l'influence du réseau pylorique et du réseau cardiaque, présente alternativement les caractéristiques propres à chacun de ces deux réseaux. L'activité du réseau cardiaque (et donc du neurone VD) est dictée par un neurone sensoriel (dont l'activité est schématisée en pointillés). Afin de simplifier la figure on n'a représenté ici que les neurones principaux de chacun des réseaux : PD pour le réseau pylorique et CD2 pour le réseau cardiaque(d'après [HOO 89]).

La séparation en entités fonctionnelles strictement isolées s'est vue contredite de façon plus virulente lorsque, en 1991, P. Meyrand a montré que des réseaux différents pouvaient fusionner totalement pour fonder, temporairement, une entité globale caractérisée par un rythme propre [MEY 91]. Chez le homard, tandis qu'en temps normal les réseaux pylorique, gastrique et oesophageal sont clairement identifiés, l'activité d'un groupe de neurones particuliers (Pyloric Suppressors, « PS »), provoque la disparition des rythmes correspondants et l'apparition d'un rythme nouveau, propre à la déglutition (figure VI-4). Si on s'en tient à la définition temporelle des assemblées de neurones (caractérisation par le rythme d'activité), il y a donc eu, sans modification des connexions, disparition de trois assemblées de neurones et création *de novo* d'un groupe fonctionnel totalement différent.

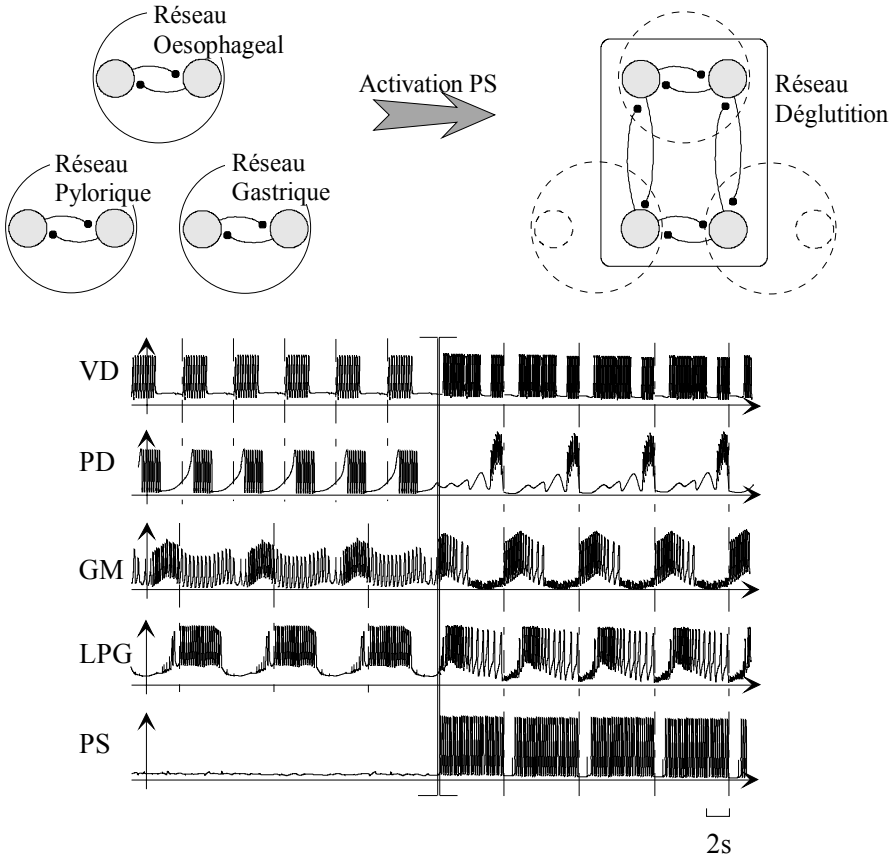


Figure VI-4 : *Genèse du réseau déglutition (d'après [MEY 91])*

Le neurone PS impose son propre rythme d'activité (rythme « déglutition ») à trois assemblées différentes : oesophageale (non représentée), pylorique (neurones VD et PD) et gastrique (neurones GM et LPG). L'activation du neurone PS provoque donc l'apparition d'une assemblée de neurones transitoire formée de neurones de trois assemblées distinctes.

Ces travaux nous apportent un enseignement important quant à la structure des assemblées de neurones : elles ne constituent pas, chez le homard et probablement chez d'autres animaux (au moins les invertébrés), des entités rigides. Elles sont, au contraire, intimement liées au comportement de l'animal voire même à son environnement. En effet la plupart des études montrent que le passage d'un rythme d'activité à un autre, plus ou moins transitoire, est intimement lié à l'activité de certains neurones sensoriels ou moteurs. Ainsi, nous avons vu que l'activité du réseau cardiaque est soumise à un neurone sensoriel particulier (effet réflexe). De même le neurone PS, responsable de l'apparition du rythme « déglutition », contrôle directement les muscles d'ouverture de la valve oesophageale. D'autres études ont permis de mettre en évidence les relations existantes entre les neurones sensoriels et l'activité des assemblées de neurones : le rythme gastrique, par exemple, est régulé par des mécanismes proprioceptifs qui peuvent, selon les cas, le renforcer, ou le remanier totalement [COM 94]. Les mécanismes de synchronisation permettent donc de modifier les assemblées de neurones pour créer (ou détruire) des comportements moteurs complexes tout en évitant la superposition destructrice de rythmes incompatibles. Les « processus décisionnels » permettant le passage d'un rythme à l'autre sont soumis à l'environnement perçu au travers des neurones sensoriels mais aussi, par anticipation, aux actions provoquant une modification de cet environnement (ouverture de la valve oesophageale) ou à des stimuli proprioceptifs générés par les actions entreprises. Ainsi, même des systèmes relativement simples, tels que le système stomatogastrique d'un crustacé, sont

soumis à des boucles sensori-motrices complexes provoquant des remaniements importants de leur activité électrique. Il n'est pas prouvé, à l'heure actuelle, que ces conclusions soient extensibles aux systèmes plus complexes tels que le cortex des mammifères. Toutefois ces travaux nous montrent la souplesse et la flexibilité des assemblées de neurones biologiques. Celles-ci sont avant tout caractérisées par la fonction des différents neurones qui les composent. En outre il est maintenant établi que les assemblées de neurones ne constituent pas des entités disjointes mais qu'au contraire, elles peuvent se chevaucher et partager une, plusieurs, voire l'ensemble des cellules qui les composent.

VI.2.4. Conclusion : synchronisation et apprentissage

Bien qu'ils diffèrent en de nombreux points, l'ensemble des travaux cités ci-dessus nous conduit à introduire, dans le codage neuronal de l'information, une dimension supplémentaire : alors que, depuis plus de cinquante ans, seule la dimension spatiale du codage connexionniste était prise en compte, les développements récents de la neurobiologie montrent l'influence prépondérante de l'information temporelle exprimée en termes de synchronisation, de rythme d'activité ou de séquence d'états. Ainsi, contrairement à ce qui était communément admis, les dates de décharge des cellules neuronales n'ont plus uniquement un contenu informationnel local (codage fréquentiel de l'activité de la cellule), mais, au contraire, un contenu informationnel global, fonction des dates de décharge des autres cellules du réseau. De plus, tous les auteurs cités insistent sur l'importance du facteur temps dans les mécanismes d'apprentissage et dans la formation des assemblées de neurones : les rythmes d'activité seraient à l'origine d'un regroupement fonctionnel des cellules nerveuses en assemblées et les synchronisations permettraient de faciliter l'apprentissage. En effet, ces deux mécanismes étant fortement générateurs de corrélations d'activités entre les cellules neuronales, ils influencent favorablement la croissance des connexions synaptiques au sein d'une même assemblée de neurones (les corrélations temporelles étant plus fortes entre neurones appartenant à une même assemblée).

La mise en évidence, par les neurobiologistes, d'un codage temporel de l'information nerveuse remet partiellement en cause les principes fondateurs du connexionnisme : dans les réseaux de neurones formels, l'information est uniquement codée sous la forme d'« influx nerveux » circulant le long des connexions synaptiques. Pourtant, malgré quelques tentatives isolées [HOR 91] [GRO 91a] [MOZ 92] et quelques simulations surtout destinées à valider les modèles des neurobiologistes [KÖN 92] [HAN 95], le codage spatio-temporel de l'information nerveuse n'a pas été exploitée dans la communauté connexionniste ; probablement parce que ce principe augmente considérablement la complexité des modèles, en particulier au niveau de la dynamique. [MOZ 92] utilise pour cela des unités à valeur complexe - voir pour cela [ROC 94] - ce qui permet de calculer l'amplitude et le déphasage du signal synaptique malgré l'absence de connexions « feed-back ». L'importance des progrès réalisés dans ce domaine ces dernières années laisse cependant présager d'un important développement de paradigmes connexionnistes nouveaux, basés sur le codage temporel de l'information, dans un avenir proche.

Pour notre part, le codage temporel revêt un intérêt particulier en raison de ses liens étroits avec les assemblées de neurones et avec l'émergence, dans le cortex, de structures neuronales spécialisées. C'est pourquoi nous avons choisi de nous en inspirer pour concevoir une architecture connexionniste répondant aux principes fondamentaux du contrôle réactif.

VI.3. Le mode ordonnancé : vers un traitement local de l'information

Notre but n'est ni de modéliser le système nerveux ni de valider les théories développées par les neurobiologistes. Aussi, nous n'avons pas cherché à mettre en place une architecture connexionniste calquée sur des réalités physiologiques. Au contraire nous avons cherché à exploiter les principes développés ci-dessus sans trop nous éloigner des modèles de réseaux de neurones formels classiques. C'est pourquoi nous nous sommes avant tout intéressés à la gestion du temps dans les modèles connexionnistes afin de dégager les modifications essentielles qui, apportées à ces modèles, permettent de localiser le codage de l'information et de faire émerger des assemblées de neurones. Nous verrons au terme de ce chapitre que cette approche nous a probablement conduits à négliger une partie des intérêts du codage temporel et qu'elle a probablement eu pour conséquence une insuffisance structurelle du modèle proposé.

VI.3.1. Aspect temporel des réseaux de neurones

Nous avons vu, dans le chapitre IV, que les réseaux de neurones formels sont gérés au moyen de séquences de mise à jour gérées par une horloge globale. Le choix de ces séquences et leur déroulement constituent le ***mode de mise à jour*** d'un réseau de neurones.

Il existe deux modes de mise à jour principaux : le mode synchrone (tous les neurones sont mis à jour simultanément à chaque pas de temps) et le mode asynchrone (les neurones sont mis à jour séquentiellement suivant un ordre préétabli - asynchrone séquentiel - ou aléatoire - asynchrone aléatoire -). Dans un cas comme dans l'autre, l'information temporelle est totalement inexistante puisque tous les neurones obéissent à une horloge globale : le mode synchrone ignore totalement l'information temporelle (puisque tous les neurones sont artificiellement synchronisés) tandis que, dans le cas du mode asynchrone, le choix artificiel des séquences de mise à jour ne permet pas de leur attribuer un contenu sémantique puisque les séquences de mise à jour ne dépendent pas des flux traversant le réseau. En outre, le mode asynchrone est contraint par une règle supplémentaire, communément admise, qui implique l'existence de cycles de mise à jour, de n pas de temps (n étant le nombre de neurones du réseau). Au cours d'un cycle, tous les neurones doivent être mis à jour une et une seule fois.

Concernant l'influence des modes de mise à jour sur le codage spatial de l'information dans les réseaux de neurones, il a été démontré que, pour une même topologie et des poids synaptiques identiques, les modes synchrone et asynchrone possèdent les mêmes points fixes [KAM 90]. Toutefois, il est aisé de montrer que, soumis aux mêmes stimulations, ils ne vont pas systématiquement converger vers le même point fixe, c'est-à-dire que les bassins d'attraction dépendent, dans certains cas, du mode de mise à jour utilisé (figure VI-5).

Si les bassins d'attraction sont différents, c'est a fortiori le cas des trajectoires suivies dans l'espace d'état. Alors, les influences conjointes du mode de mise à jour et de l'algorithme d'apprentissage vont entraîner des évolutions différentes et les réseaux vont converger (s'ils convergent) vers des organisations topologiquement différentes mais répondant aux mêmes contraintes d'entrées/sorties[65]. Cependant, les principes même des modes synchrone et asynchrone (pas de séquence de mise à jour ou des séquences détachées du contenu

---

[65] Tout dépend cependant de l'algorithme d'apprentissage utilisé : si il est basé sur un calcul d'erreur effectué après stabilisation du réseau - donc en un point fixe - il semble évident que les réseaux vont évoluer de façon strictement identique. Les algorithmes type Rétropropagation du Gradient sont d'ailleurs basés sur ce principe.

informationnel, cycles de mise à jour, ...) ne permettent pas de localiser l'information apprise sur un ensemble limité de neurones (une « assemblée de neurones »). Au contraire, ils contribuent à répartir l'information sur l'ensemble des noeuds du réseau en sollicitant, pour chaque couple entrée/sortie, la participation de l'ensemble des neurones du réseau. L'accès à tout élément d'information stocké dans le réseau implique la consultation de toutes les cellules et de l'ensemble des connexions (nous ne considérons pas ici les problèmes de robustesse inhérents à l'architecture connexionniste). Le réseau peut alors être considéré comme une carte contenant, dans sa globalité, l'information apprise (« ***pattern coding*** »). Ce type de codage est d'ailleurs responsable de la plupart des limitations propres aux réseaux de neurones (oubli catastrophique, états parasites, orthogonalité des « patterns » dans les mémoires associatives, ...).

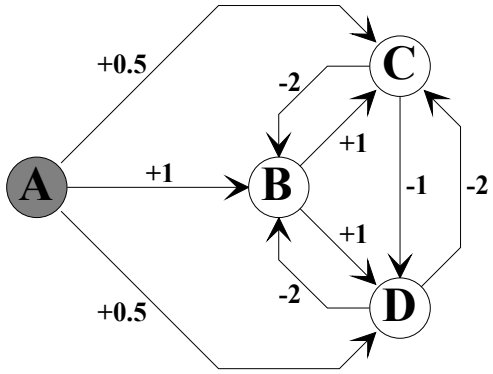


Figure VI-5 : *Contre-exemple sur l'identité des bassins d'attraction selon le mode de mise à jour utilisé.*

En présence d'un neurone A actif (état +1), le réseau, composé de neurones binaires, comporte deux points fixes ($A\overline{B}C\overline{D}$ et $A\overline{B}\overline{C}D$). Le mode synchrone provoque l'activation immédiate de tous les neurones du réseau (état $ABCD$)puis, en raison des connexions inhibitrices, la désactivation des neurones B et C. Le mode synchrone conduit donc systématiquement vers le point fixe $\overline{A}BC\overline{D}$ et l'état $A\overline{B}\overline{C}D$ appartient à son bassin d'attraction. Par contre, si le mode asynchrone est utilisé, les deux points fixes peuvent être atteints (directement ou indirectement). Dans ce cas, l'état $A\overline{B}\overline{C}D$ peut appartenir à l'un ou l'autre des bassins d'attraction suivant la séquence de mise à jour utilisée.

VI.3.2. Le mode ordonnancé

Le principe initial du mode ordonnancé est de libérer les cellules neuronales de l'emprise d'une horloge globale, c'est-à-dire d'autoriser certains neurones à réagir plus rapidement que d'autres proportionnellement à l'intensité du stimulus reçu. Dès lors, des neurones plus faiblement stimulés peuvent ne pas être pris en compte lors du calcul de la réponse et les mécanismes d'apprentissage vont favoriser les connexions utiles propres à chaque couple entrée/sortie. De plus, afin d'autoriser certains neurones (même stimulés au dessus du seuil de décharge) à rester inactifs, il faut supprimer les cycles de mise à jour qui obligent des neurones stimulés différemment à s'activer lors du même cycle. Contrairement aux réseaux classiques qui diluent l'information sur l'ensemble des connexions, un réseau construit suivant ce principe va favoriser la croissance d'un nombre restreint de connexions utiles reliant les neurones d'entrée aux neurones de sortie via un nombre variable de cellules cachées. (« ***path coding*** »). Il est en cela comparable aux réseaux sémantiques à propagation d'activation [COL 75] dans lesquels l'influx nerveux active progressivement des noeuds sémantiques (faits) reliés par des connexions pondérées ou non (règles). Des différences notables empêchent cependant de pousser plus loin la comparaison : dans les réseaux à propagation d'activation, la topologie est fixée manuellement et

les noeuds sont clairement identifiés. Ce n'est pas le cas ici puisque toutes les connexions sont potentiellement utilisables.

VI.3.2.1. Algorithme

Pour permettre au réseau de réaliser des séquences d'activation, nous avons développé une variante du mode asynchrone, basée sur un principe d'ordonnancement dynamique des activations neuronales (d'où l'appellation « mode ordonnancé »). A chaque pas de temps un et un seul neurone va changer d'état (tandis que, dans le mode asynchrone classique, à chaque pas de temps, un neurone est mis à jour, ce qui n'implique pas forcément un changement d'état). Ce neurone est choisi grâce à une loi d'ordonnancement basée sur un calcul de déséquilibre[66].

Dans un réseau de n neurones binaires d'état x_i, $x_i = \{0,1\}$, $i = 1..n$, l'état du réseau au temps t, peut être représenté par un vecteur binaire :

$$\vec{x}(t) = \{x_1(t), x_2(t), ..., x_i(t), ..., x_n(t),\} \quad \textbf{(VI-1)}$$

Chaque neurone possède une valeur de déséquilibre $d_i(t)$ (déséquilibre du neurone i au temps t), caractérisant la force avec laquelle les influx qu'il reçoit (ou qu'il a reçus) tendent à le faire changer d'état. L'ensemble des déséquilibres forme un vecteur appelé vecteur de déséquilibre représentatif de la dynamique du réseau au temps t :

$$\vec{d}(t) = \{d_1(t), d_2(t), ..., d_i(t), ..., d_n(t),\} \quad \textbf{(VI-2)}$$

Selon le mode ordonnancé, on a alors :

$$\vec{x}(t+1) = \{x_1(t), x_2(t), ..., x_i(t+1), ..., x_n(t),\} \quad \textbf{(VI-3)}$$

où le neurone i est celui dont le déséquilibre au temps t est le plus important, à condition qu'il soit strictement supérieur à zéro :

$$d_i(t) = MAX_{j=1..n}(d_j(t)), \quad d_i(t) > 0 \quad \textbf{(VI-4)}$$

Les neurones étant à état binaire, on a, pour le neurone i :

---

[66] D'autre lois d'ordonnancement sont envisageables, incluant par exemple des contraintes topologiques, mais elles n'ont pas été étudiées dans le cadre de ce travail.

$$x_i(t+1) = 1 - x_i(t) \qquad \textbf{(VI-5)}$$

Lorsque tous les déséquilibres sont négatifs ou nuls, aucun neurone ne peut plus changer d'état ; le réseau a alors atteint un de ses points fixes.

Le mode ordonnancé n'est pas restreint par les cycles de mise à jour utilisés pour le mode asynchrone : un neurone peut, bien qu'étant stimulé positivement, ne jamais être activé.

VI.3.2.2. Calcul du vecteur de déséquilibre

Le vecteur de déséquilibre le plus simple est le vecteur de **déséquilibre instantané** : c'est la différence, au temps t , entre les stimulations reçues par un neurone et le seuil de stimulation lui permettant de conserver son état actuel. C'est ce vecteur que nous avons utilisé lors des tests présentés ci-après.

Soit, si, au temps t , un neurone binaire i , possédant un seuil d'activation θ_i, reçoit de l'ensemble du réseau une stimulation $E_i(t)$ telle que :

$$E_i(t) = \sum_{j=1}^{n} W_{ij} x_j(t) \qquad \textbf{(VI-6)}$$

Cette stimulation dicte au neurone un état d'équilibre souhaitable $\hat{x}_i(t+1)$, tel que :

$$\hat{x}_i(t+1) = H\left(E_i(t) - \theta_i\right) = H\left(\sum_{j=1}^{n} W_{ij} x_i(t) - \theta_i\right) \qquad \textbf{(VI-7)}$$

La fonction de transfert des neurones étant la fonction de Heaviside seuillée, l'influx $\hat{E}_i$ tel que :

$$\hat{E}_i(t) = \theta_i \qquad \textbf{(VI-8)}$$

constitue une valeur limite pour le comportement des cellules : pour être (ou devenir) actif, le neurone doit recevoir un influx supérieur à $\hat{E}_i$; dans le cas contraire, il doit recevoir un influx inférieur. Le déséquilibre instantané du neurone est donc donné par la différence entre l'influx réellement reçu et l'influx minimum nécessaire à la conservation de l'état, multipliée par +/-1 selon le sens de la variation d'état potentielle. Soit :

$$\begin{cases} d_i(t) = \sum_{j=1}^{n} W_{ij} x_j(t) - \theta_i & si\ x_i(t) = 0 \\ d_i(t) = -\sum_{j=1}^{n} W_{ij} x_j(t) + \theta_i & si\ x_i(t) = 1 \end{cases} \qquad \textbf{(VI-9)}$$

Soit,

$$d_i(t) = (1 - 2x_i(t))\left[\sum_{j=1}^{n} W_{ij} x_j(t) - \theta_i\right] \qquad \textbf{(VI-10)}$$

Le vecteur des déséquilibres instantanés peut donc être calculé très simplement à partir du vecteur d'état et des entrées seuillées de chacun des neurones. Les déséquilibres instantanés peuvent être additionnés et la loi d'ordonnancement peut obéir au **déséquilibre cumulé** :

$$D_i(t) = \gamma D_i(t-1) + (1-\gamma) d_i(t) \qquad \textbf{(VI-11)}$$

où γ représente le facteur d'atténuation du déséquilibre.

Cependant le déséquilibre cumulé peut ne pas être du même signe que le déséquilibre instantané. La mise à jour d'un neurone sur la seule foi du déséquilibre cumulé peut donc conduire à augmenter son déséquilibre instantané.

VI.3.2.3. Architecture

Pour que le mode ordonnancé permette l'émergence, dans un réseau de neurones, de structures du type « assemblées de neurones » ou « boucles neuronales », il est indispensable que le réseau lui-même respecte un certain nombre de contraintes liées au principe de « **path coding** ».

Le mode ordonnancé est utilisé pour créer et mémoriser des boucles entrée/sortie plus ou moins complexes selon la complexité de la fonction apprise. Ces chemins doivent, en particulier, pouvoir comporter des cycles et mémoriser des états intermédiaires (pour pouvoir, en neurocontrôle, générer des comportements plus complexes que de simples arcs entrée/sortie). C'est pourquoi nous proposons d'utiliser des réseaux structurés en trois groupes de neurones (Voir figure VI-6) :

- Neurones d'entrée : leur état est directement fonction de l'environnement du réseau. Les neurones d'entrée sont totalement indépendants les uns des autres.

- Neurones d'afférence : ils sont utilisés pour relier les neurones d'entrée (groupe précédent) aux neurones de sortie (groupe suivant) par des chemins de longueur variable, éventuellement bouclés. Les neurones d'afférence forment donc un « réseau caché » récurrent.

- Neurones de sortie : ils reçoivent des connexions des neurones d'afférence et des neurones d'entrée. Leur état est directement utilisé pour déterminer l'action sur l'environnement.

En pratique, on utilisera pour décrire ces trois groupes de neurones les termes de « couche d'entrée », « couche cachée » (ou « couche d'afférence ») et « couche de sortie ». Il s'agit cependant d'un abus de langage puisque, en raison des connexions internes dans la « couche cachée », un tel réseau n'est pas conforme à la définition d'un réseau en couches telle qu'elle est donnée au chapitre IV (section 4.3.3.2).

La structure d'un tel réseau permet de distinguer plusieurs types de connexions suivant l'appartenance des neurones pré-synaptiques et post-synaptiques à l'un ou l'autre des groupes de neurones :

- Le fuseau d'entrée relie la couche d'entrée aux neurones d'afférence. Chaque neurone d'entrée est potentiellement connecté à l'ensemble des neurones cachés.

- Les connexions afférentes sont les connexions internes à la couche cachée. Chaque neurone d'afférence est connecté à l'ensemble de ses pairs avec ou sans autoconnexion. Afin de rester cohérent avec la notion de chemin d'activation (orienté de l'entrée vers la sortie), les connexions afférentes ne sont pas symétriques.
- Le fuseau de sortie relie les neurones d'afférence aux neurones de sortie. Comme pour le fuseau d'entrée, tout neurone de sortie reçoit - potentiellement - une connexion en provenance de chacun des neurones cachés.
- Connexions réflexes : nous regroupons sous cette appellation les connexions reliant directement la couche d'entrée à la couche de sortie. Ces connexions permettent d'introduire, dans un contrôleur, des comportements innés, prévus par le concepteur (comportements réflexes). La pondération des connexions réflexes est donc fixée a priori et non déterminée par l'apprentissage. D'autres formes de connexions réflexes sont utilisables. Nous avons ainsi utilisé des connexions inhibitrices internes à la couche de sortie afin de permettre un comportement type « Winner-Takes-All » (section VI.4.5).

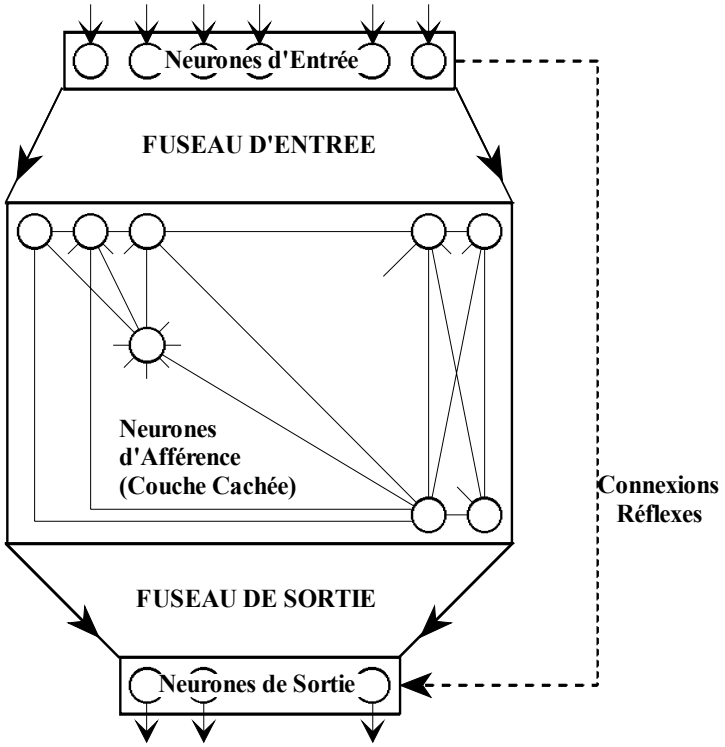


Figure VI-6 : *Architecture d'un réseau utilisant le mode ordonnancé.*

VI.3.3. Dynamique du mode ordonnancé

Nous avons présenté, au cours du chapitre IV, les principaux théorèmes concernant la dynamique des réseaux de neurones (sections IV.3.5.1 et IV.3.5.2). Nous présentons dans ce chapitre l'extension de ces théorèmes aux réseaux utilisant le mode ordonnancé basé sur les déséquilibres instantanés.

Lorsque la topologie du réseau est fixée (c'est-à-dire lorsque les poids synaptiques W_{ij} et les seuils θ_i sont fixés $i = 1..n,\ j = 1..n$) son espace d'état est un ensemble fini de vecteurs d'états. Il peut donc présenter deux types de dynamique :

- Points fixes (ou convergence vers un point fixe)

$$\exists t' \ / \ \forall t > t', \quad \vec{x}(t+1) = \vec{x}(t) \text{ (c'est-à-dire } MAX_{j=1..n}\left(d_j(t)\right) \leq 0\text{)} \qquad \textbf{(VI-12)}$$

- Etats cycliques

$$\exists t' > 1 \text{ tel que } \exists T / \forall t > t', \vec{x}(t+T) = \vec{x}(t) \qquad \textbf{(VI-13)}$$

où T est la période de l'attracteur

VI.3.3.1. Points fixes

Nous avons déjà vu que les points fixes d'un réseau ne dépendent pas du mode de mise à jour utilisé (synchrone ou asynchrone) [KAM 90]. Ce résultat peut être directement étendu au mode ordonnancé :

Si $\vec{x}$ est un point fixe pour le mode synchrone ou pour le mode asynchrone, alors on a :

$$\exists t' \ / \ \forall t > t', \quad \vec{x}(t+1) = \vec{x}(t)$$

D'après la fonction de transfert des neurones (équation VI-7), on a donc :

$$\forall i = 1..n, x_i(t+1) = H\left(\sum_{j=1}^{n} W_{ij} x_j(t) - \theta_i\right) = x_i(t) \qquad \textbf{(VI-14)}$$

D'après (VI-14), on déduit que

$$\forall i, i = 1..n, sign\left(1 - 2x_i(t)\right) = -sign\left(\sum_{j=1}^{n} W_{ij} x_j(t) - \theta_i\right)$$

et, d'après (VI-10),

$$\forall i, i = 1..n, d_i(t) = \left(1 - 2x_i(t)\right)\left[\sum_{j=1}^{n} W_{ij} x_j(t) - \theta_i\right]$$

$\vec{x}$ est donc un point fixe pour le mode ordonnancé.

Si $\vec{x}$ est un point fixe pour le mode asynchrone, alors le vecteur de déséquilibre instantané associé est un vecteur à valeur négative. $\vec{x}$ est donc aussi un point fixe pour le mode ordonnancé.

Réciproquement, si $\vec{x}$ est un point fixe pour mode ordonnancé, le vecteur de déséquilibre instantané est à valeur négative et, d'après (VI-14), $\vec{x}$ est un point fixe pour le mode asynchrone

VI.3.3.2. Convergence

Le théorème de Cohen-Grossberg ([COH 83], voir section IV.3.5.2.) peut lui aussi être étendu au mode ordonnancé : en utilisant, pour un réseau de n neurones binaires, à connexions symétriques[67] ($W_{ji} = W_{ij}$) et dont les autoconnexions sont positives ou nulles, la fonction d'énergie suivante[68] :

$$e(\vec{x}) = \tfrac{1}{2}\vec{x}^T W \vec{x} + \vec{x}^T \vec{\theta} \qquad \textbf{(VI-15)}$$

On montre très simplement que la mise à jour d'un neurone conduit l'énergie à décroître tout en restant positive et que, par conséquent, le réseau converge vers un point fixe.

En supposant qu'au temps t le neurone correspondant au premier élément du vecteur d'état (donc le neurone « 1 ») est mis à jour[69], la fonction d'énergie s'écrit,

au temps t :

$$e(\vec{x}(t)) = -\tfrac{1}{2}\left[W_{11}x_1^2(t) + 2x_1(t)\sum_{i=2}^{n} W_{1i}x_i(t) + \sum_{j=2}^{n}\left(x_j(t)\sum_{i=2}^{n} W_{ji}x_i(t)\right)\right]$$
$$+ \theta_1 x_1(t) + \sum_{i=2}^{n}\theta_i x_i(t)$$

et au temps $t+1$:

$$e(\vec{x}(t+1)) = -\tfrac{1}{2}\left[W_{11}x_1^2(t+1) + 2x_1(t+1)\sum_{i=2}^{n} W_{1i}x_i(t+1) + \sum_{j=2}^{n}\left(x_j(t+1)\sum_{i=2}^{n} W_{ji}x_i(t+1)\right)\right]$$
$$+ \theta_1 x_1(t+1) + \sum_{i=2}^{n}\theta_i x_i(t+1)$$

---

[67] Ce type de réseau ne correspond pas à l'architecture décrite au section VI.3.2.2. : la condition de symétrie des connexions, nécessaire au théorème de Cohen-Grossberg, n'y est pas respectée. Nous verrons dans la section suivante que la convergence n'est pas systématiquement recherchée dans ce type d'architecture. Le théorème de Cohen-Grossberg est donc d'un intérêt limité même s'il permet une première approche de la dynamique du mode ordonnancé.

[68] Cette fonction d'énergie est désormais d'utilisation classique pour prouver la convergence des réseaux récurrents. Elle a été introduite par J. Hopfield ([HOP 82]).

[69] La généralisation aux autres éléments du réseau est obtenue par une simple permutation circulaire.

Or, pour des neurones binaires, et selon les principes du mode ordonnancé, on a $x_1(t+1) = 1 - x_1(t)$ et $x_i(t+1) = x_i(t)$ pour $i = 2..n$. La fonction d'énergie au temps $t+1$ devient donc :

$$e(\vec{x}(t+1)) = -\tfrac{1}{2}\left[W_{11} - 2W_{11}x_1(t) + W_{11}x_1^2(t) + 2\sum_{i=2}^{n} W_{1i}x_i(t) - 2x_1(t)\sum_{i=2}^{n} W_{1i}x_i(t) \right.$$
$$\left. + \sum_{j=2}^{n}\left(x_j(t)\sum_{i=2}^{n} W_{ji}x_i(t) \right) \right] + \theta_1 - \theta_1 x_1(t) + \sum_{i=2}^{n} \theta_i x_i(t)$$

La variation d'énergie entre les temps t et $t+1$ est alors :

$$e(\vec{x}(t+1)) - e(\vec{x}(t)) = -\tfrac{1}{2}\left[W_{11} - 2W_{11}x_1(t) + 2\sum_{i=2}^{n} W_{1i}x_i(t) \right.$$
$$\left. - 4x_1(t)\sum_{i=2}^{n} W_{1i}x_i(t) \right] + \theta_1 - 2\theta_1 x_1(t)$$

soit,

$$e(\vec{x}(t+1)) - e(\vec{x}(t)) = (1 - 2x_1(t))\left[\theta_1 - \frac{W_{11}}{2} - \sum_{i=2}^{n} W_{1i}x_i(t) \right]$$

Or, d'après (VI-10) on a :

$$d_1(t) = (1 - 2x_1(t))\left[W_{11}x_1(t) + \sum_{i=2}^{n} W_{1i}x_i(t) - \theta_1 \right]$$

Soit :

$$e(\vec{x}(t+1)) - e(\vec{x}(t)) = -d_1(t) - \tfrac{1}{2}W_{11}(1 - 2x_1(t))^2$$

et, comme $\vec{x}_1(t) \in \{0,1\}$, on obtient :

$$e(\vec{x}(t+1)) - e(\vec{x}(t)) = \Delta e_{t \to t+1} = -d_1(t) - \tfrac{1}{2}W_{11} \qquad \textbf{(VI-16)}$$

Dans un réseau symétrique, à unités binaires, ne possédant que des autoconnexions activatrices et soumis au mode ordonnancé, le changement d'état d'une cellule entraîne toujours une baisse d'énergie égale au déséquilibre de cette cellule (nécessairement positif) plus la moitié de l'autoconnexion de cette cellule. La fonction d'énergie étant définie positive, la dynamique de

ce type de réseau conduit nécessairement vers un point stable. **Le théorème de Cohen-Grossberg est donc généralisable aux réseaux utilisant le mode ordonnancé**[70]**.**

Dans le cas d'un réseau sans autoconnexion on a, à chaque variation d'état d'un neurone, une perte d'énergie égale au déséquilibre de ce neurone (donc, suivant les principes du mode ordonnancé, égale au plus grand déséquilibre des cellules du réseau). Ce résultat nous permet d'interpréter la dynamique du mode ordonnancé en termes de descente en gradient. L'espace d'état du réseau est un espace à n dimensions (n : nombre de neurones). Chaque point de cet espace est caractérisé par la valeur de l'énergie e en ce point (Equation VI-15). Présenter une entrée au réseau revient à placer celui-ci en un point particulier de l'espace d'état (fonction du vecteur d'entrée et de l'état antérieur du réseau) à partir duquel il va suivre une trajectoire caractéristique de sa dynamique[71]. Lorsque les conditions du théorème de Cohen-Grossberg sont satisfaites, cette trajectoire conduit le réseau vers un point d'énergie minimale (qui peut être un minimum local) selon la règle de progression suivante :

Deux points successifs de la trajectoire dynamique ne différent que par une de leurs composantes (caractéristique propre au mode synchrone). Dans le cas du mode ordonnancé cette composante est celle qui permet la plus forte descente d'énergie lors du passage d'un point à l'autre. **Le mode ordonnancé permet donc d'effectuer une descente d'énergie par dimensions successives, en choisissant à chaque pas de temps la dimension offrant le plus fort gradient de descente.**

VI.3.3.3. Conclusion : vers des dynamiques complexes

Les résultats présentés ci-dessus (extension du théorème de Cohen-Grossberg et similitude des points fixes du réseau) bien qu'ils constituent une première approche de la dynamique des réseaux utilisant le mode ordonnancé, sont d'une portée limitée. En effet, les notions de point fixe et de convergence sont intimement liées aux principes du « ***pattern coding*** » : le point fixe est considéré comme le résultat final d'un calcul neuronal et les preuves de convergence permettent d'assurer l'utilisateur de l'obtention de ce résultat. Dans ce contexte, les conditions nécessaires à la convergence (matrice de poids symétrique et à diagonale nulle) contraignent le réseau sans en entraver le bon fonctionnement. Par contre, ces contraintes sont totalement incompatibles avec l'architecture décrite section VI.3.2.3. Le théorème de Cohen-Grossberg est donc inapplicable et la stabilité du réseau n'est pas assurée.

En outre, l'existence de groupes de neurones spécifiquement dédiés aux tâches d'entrée/sortie nous conduit à considérer la stabilité sous un angle nouveau : contrairement aux réseaux récurrents existants (réseau de Hopfield, machine de Boltzmann, mémoires associatives), la stabilité globale n'est pas nécessaire pour que le réseau fournisse une réponse (puisque seul un sous-ensemble de la population neuronale est interprété comme étant une sortie). Au contraire, l'apparition de stabilités « précoces » (c'est à dire antérieures à l'activation d'une cellule de sortie) peut, dans certains cas, se révéler préjudiciable au bon fonctionnement du réseau. De même, l'existence de cycles internes au réseau « afférent » peut être utile pour

---

[70] Par rapport au théorème initialement démontré dans [COH 83], l'utilisation de neurones binaires (à valeur dans {0,1}), permet d'étendre la portée du théorème aux réseaux ne comportant que des autoconnexions activatrices ou nulles (dans le théorème initial le réseau considéré ne comporte pas d'autoconnexions).

[71] Dans le cas de neurones binaires, le réseau ne peut atteindre que certains points particuliers de l'espace d'état.

produire, à partir d'une entrée unique, une séquence d'actions plus ou moins complexes (section VI.4.5), à condition toutefois que ces cycles ne soient pas « bloquants » pour le reste du réseau, c'est-à-dire qu'ils ne génèrent pas des déséquilibres plus importants que ceux permettant la propagation de l'influx nerveux. Dans le cas contraire, le principe du mode ordonnancé risque de conduire le réseau à ne plus propager l'influx nerveux vers la sortie mais, au contraire, à réquisitionner toutes les ressources pour la gestion du cycle interne.

Bien que, comme le montre l'identité des points fixes et l'extension du théorème de Cohen-Grossberg, le mode ordonnancé ne génère pas de dynamique plus complexe que les modes synchrones ou asynchrones, l'architecture du réseau utilisé entraîne l'apparition d'états complexes difficilement interprétables. De tels états (stabilités précoces et cycles internes) ne permettant pas l'obtention d'une réponse en sortie du réseau ; nous les regrouperons sous l'appellation d'« états non-répondants ». Bien qu'ils puissent être, dans certains cas, recherchés (lorsque le réseau doit mémoriser des états intermédiaires sans générer de réponse immédiate), ils sont généralement préjudiciables au bon fonctionnement du réseau et nous verrons (section VI.3.4.3) comment ils peuvent être éliminés au cours de l'apprentissage.

VI.3.4. Apprentissage et mode ordonnancé

Outre ses composantes architecturales et dynamiques, un réseau de neurones est caractérisé par l'algorithme utilisé pour le calcul des poids synaptiques (voir section IV.3.6). Bien que, dans certains cas, les poids puissent être déterminés analytiquement (essentiellement pour le réseau de Hopfield [HOP 85] et pour les mémoires associatives [KAM 90]), ils sont la plupart du temps déterminés progressivement au cours de la phase d'apprentissage. C'est en particulier le cas pour le mode ordonnancé qui a été conçu pour mettre en oeuvre, par l'apprentissage, un contrôleur réactif.

Comme nous l'avons vu au cours des chapitres précédents (chapitres IV et V), l'utilisation des réseaux de neurones pour le « Behavior-Based-Control » entraîne d'importantes contraintes, non seulement quant à la structure du réseau utilisé (d'où le développement du mode ordonnancé), mais aussi quant à l'algorithme d'apprentissage. Celui-ci doit, en particulier, appartenir à la famille des apprentissages par renforcement (sections IV.2.1, IV.3.6.3 et IV.4.3) afin d'autoriser l'apprentissage à partir d'une simple estimation des performances globales. En outre, pour être compatible avec les principes du mode ordonnancé, c'est-à-dire pour autoriser l'apprentissage de structures locales dans un réseau global, cet algorithme doit déterminer localement les modifications à apporter aux poids synaptiques. C'est pourquoi nous avons choisi d'utiliser un algorithme d'apprentissage de type Hebbien, c'est-à-dire dérivé des principes énoncés en 1949 par D.O. Hebb [HEB 49].

VI.3.4.1. Principe : un apprentissage Hebbien

L'apprentissage hebbien (voir section IV.3.6.2) a été initialement proposé par Donald O. Hebb en 1949 pour expliquer les mécanismes d'apprentissage dans le système nerveux. Il n'a donc pas une origine connexionniste (rappelons que D.O. Hebb n'était pas un informaticien mais un psychologue, fortement influencé par les travaux de Lorente de Nó sur les boucles neuronales). Face à la distribution de l'information dans le système nerveux, Hebb s'est attaché à proposer un algorithme d'apprentissage ne nécessitant pas l'intervention d'un « centre d'apprentissage » spécialisé. C'est pourquoi, il base ce mécanisme sur l'activité locale des neurones : la synapse va avoir tendance à croître lorsqu'elle est active, c'est-à-dire lorsqu'elle

relie deux neurones dont l'activité est corrélée (le neurone pré-synaptique ayant tendance à activer le neurone post-synaptique[72]).

La règle d'apprentissage énoncée par Donald O. Hebb en 1949 n'est pas une formulation mathématique mais une formulation générale. C'est pourquoi elle a donné naissance à plusieurs algorithmes d'apprentissage, tous se réclamant de leur origine « hebbienne », basés sur des interprétations mathématiques de l'énoncé initial. Le plus connu et le plus répandu d'entre eux (le seul à être qualifié d'apprentissage « hebbien » bien qu'il contredise une partie du texte initial) est basé sur les corrélations directes d'activité neuronale. On a alors :

$$\Delta W_{ij}(t) = \alpha x_i(t) x_j(t) \qquad \textbf{(VI-17)}$$

où α est le facteur d'apprentissage.

Alors que le facteur d'apprentissage est généralement constant et positif (la règle de Hebb étant utilisée comme mécanisme d'auto-organisation), nous l'utiliserons ici en liaison avec un générateur de renforcement. α peut alors prendre des valeurs positives ou négatives en fonction du signal de renforcement.

Cette formulation ne respecte pas totalement le texte de Hebb : alors que dans le modèle initial les modifications métaboliques des synapses sont « orientées » suivant l'ordre d'activation des cellules neuronales, cette distinction n'apparaît pas dans le modèle mathématique qui entraîne une modification symétrique des poids synaptiques ($\Delta W_{ij}(t) = \Delta W_{ji}(t)$). De plus, ce modèle est partiellement mis en cause par la neuropsychologie et, en particulier, par les principes du conditionnement pavlovien (il ne permet pas d'expliquer les intervalles inter-stimuli [GRO 87]). C'est pourquoi plusieurs auteurs ont proposé des modèles mathématiques plus complexes, permettant à la fois de se rapprocher de l'énoncé original et de représenter plus finement les processus d'apprentissage observés chez l'animal. Harry Klopf [KLO 86] a ainsi proposé un modèle d'apprentissage (« Drive-Reinforcement-Learning[73] ») basé sur la corrélation des changements d'activité des neurones pré- et post-synaptiques. Par rapport au modèle hebbien classique, le « Drive-Reinforcement-Learning » ne se limite pas aux corrélations instantanées (activité simultanée) : Klopf introduit un mécanisme de « trace » inspiré de l'« Adaptive Search Element » (ASE) de Sutton et Barto ([BAR 83], voir section IV.4.3.1), qui permet de tenir compte, pour l'apprentissage, de l'intervalle interstimulus (équation VI-18).

---

[72] « *When an axon of a cell A is near enought to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency as one of the cells firing B is increased.* » ([HEB 49], chapitre 4)

[73] Cet algorithme d'apprentissage a été proposé simultanément et indépendamment par A.H. Klopf [KLO 86] sous l'appellation « *Drive-Reinforcement Learning* » et par B. Kosko [KOS 86] sous l'appellation « *Differential Hebbian Learning* ».

$$\Delta W_{ij} = \sum_{k=1}^{\tau} c_k \left| W_{ij}(t-k) \right| \left[x_j(t-k) - x_j(t-k-1) \right] \left[x_i(t) - x_i(t-1) \right] \qquad \textbf{(VI-18)}$$

où :

j est le neurone pré-synaptique et i le neurone post-synaptique,

τ est l'intervalle inter-stimulus maximum pour lequel les mécanismes de conditionnement sont efficaces,

c_k est un facteur d'apprentissage fonction de l'intervalle inter-stimulus (permettant de pondérer l'efficacité de la trace).

L'apprentissage est fonction de l'efficacité antérieure de la synapse (d'où l'introduction du terme « $\left| W_{ij}(t-k) \right|$ » dans l'équation VI-18).

D'après Harry Klopf, le « Drive-Reinforcement-Learning » permet de modéliser plus finement les mécanismes de conditionnement observés chez l'animal. Cet algorithme a d'ailleurs été utilisé par la suite pour l'apprentissage de tâches sensori-motrices analogues à celles proposées aux rats de laboratoire par les éthologistes (recherche de nourriture dans un labyrinthe en « T »). D'autre part, l'auteur propose plusieurs « variantes » de l'algorithme initial en utilisant différemment les variations négatives de l'état du neurone pré-synaptique (en les éliminant ou en les utilisant uniquement pour annuler les effets des variations positives). L'apprentissage des connexions inhibitrices (liées aux variations négatives de l'état des neurones) est en effet un des points faibles de la loi de Hebb. Déjà en 1949 D.O. Hebb n'avait pas traité le problème de l'inhibition (d'après [AND 88], introduction au chapitre 4).

Une autre interprétation de la loi de Hebb, a été proposée en 1988 par J.Y. Cheung et M. Omidvar sous l'appellation « Most-Recent-Trace-Learning » (MRT) :

$$\Delta W_{ij} = c W_{ij} x_j(t) \Delta x_i(t) \qquad \textbf{(VI-19)}$$

Contrairement au précédent, l'apprentissage « MRT » n'a pas été construit pour modéliser les processus d'apprentissage biologiques. Il n'a donc pas été validé par une comparaison avec les mécanismes de conditionnement pavloviens. Par contre, il ne nécessite pas de mémoriser, pour chaque neurone, l'ensemble des traces des connexions entrantes.

Des trois mécanismes d'apprentissage hebbien présentés ci-dessus, seul le troisième (MRT Learning) semble directement applicable au mode ordonnancé, d'une part parce que celui-ci suppose en effet des connexions asymétriques (donc un apprentissage lui-même asymétrique, contrairement à ce que propose l'équation VI-17) et d'autre part parce que le temps, dans un réseau ordonnancé, ne peut pas être considéré comme « continu » (le temps local d'une connexion est fortement influencé par l'existence - ou non - de déséquilibres plus importants dans les autres connexions) ce qui rend difficile l'utilisation d'un mécanisme de trace[74]. Dans la section suivante nous détaillons l'adaptation des mécanismes hebbiens au mode ordonnancé.

---

[74] Les influences relatives des mécanismes de trace et du mode ordonnancé demanderaient cependant, à elles seules, une étude plus complète. En effet, le temps écoulé entre l'activation du neurone pré-synaptique et du neurone post-synaptique peut être considéré comme caractéristique de l'importance relative de cette connexion, ce qui permettrait de remplacer le terme « W_{ij} » dans l'équation (VI-18) par un mécanisme de trace spécifique.

VI.3.4.2. Adaptation au mode ordonnancé

L'utilisation conjointe de la loi de Hebb et de la mise à jour ordonnancée permet de développer des mécanismes d'apprentissage spécifiques, tenant compte des corrélations d'activité (ou de variation d'activité suivant le modèle retenu) et des séquences d'activation générées. Il est alors possible, avec une règle hebbienne classique (équation VI-17) de modifier asymétriquement les connexions (VI-20) :

Si t_i est la date de la dernière variation d'état du neurone *i* et t_j celle du neurone *j*, alors le poids W_{ij} de la connexion du neurone *j* vers le neurone *i* est modifié de sorte que :

$$\Delta W_{ij}(t_i) = \alpha H(t_i - t_j) x_i(t_i) x_j(t_j) \qquad \textbf{(VI-20)}$$

Le « Drive-Reinforcement-Learning » a été modifié de façon similaire. En outre il a été simplifié (seules les corrélations entre les dernières modifications d'état sont prises en compte) et le terme « $|W_{ij}(t-k)|$ » a été supprimé car il entraîne une croissance trop rapide de certains poids, ce qui bloque l'apprentissage du reste du réseau. On obtient alors l'algorithme d'apprentissage suivant :

$$\Delta W_{ij}(t_i) = \alpha H(t_i - t_j) \Delta x_i(t_i) \Delta x_j(t_j) \qquad \textbf{(VI-21)}$$

L'équation (VI-21) entraîne toutefois, comme l'algorithme initial de Klopf, la prise en compte, pour l'apprentissage, de corrélations de changements d'activité n'ayant aucun contenu sémantique réel en terme de lien de causalité[75]. Ainsi, lorsqu'un neurone « A » passe de l'état actif (« 1 ») à l'état inactif (« 0 »), un lien de causalité (négatif) supposé relie ce changement d'état aux variations ultérieures des autres cellules du réseau. C'est-à-dire que l'activation d'une cellule « B » conduit à créer une connexion inhibitrice de A vers B. Or, il est impossible de déterminer, à la seule vue des changements d'état, la nature exacte du lien causal liant les neurones A et B : l'activation du neurone B peut être à due la suppression de l'activité du neurone A (AB est alors une synapse inhibitrice) ou à l'activation d'un inter-neurone (AB activatrice). C'est pourquoi, de même que Harry Klopf propose de n'utiliser que les variations positives des neurones pré-synaptiques, nous utiliserons un algorithme d'apprentissage plus proche du « MRT Learning » (Equation VI-22).

$$\Delta W_{ij}(t_i) = \alpha H(t_i - t_j) x_j(t_j) \Delta x_i(t_i) \qquad \textbf{(VI-22)}$$

Aucun des modèles hebbiens présentés ci-dessus (équations VI-17 à VI-22) ne permet une convergence en loi de l'algorithme d'apprentissage : lorsque la fonction Entrée/Sortie recherchée (le « comportement » en neurocontrôle) est apprise, les poids continuent de croître (ou, pour certains d'entre eux, de décroître). En diminuant progressivement le facteur « α » au cours de l'apprentissage, il est possible d'obtenir une convergence en valeur mais cela oblige à structurer la phase d'apprentissage de façon à ce que toutes les configurations possibles soient périodiquement présentées au réseau, ce qui n'est pas cohérent avec l'apprentissage on-line d'un contrôleur. C'est pourquoi nous avons utilisé un terme modérateur - identique à celui des modèles ART (Annexe A2, section 8) - permettant de limiter l'accroissement des poids

---

[75] Sous réserve que les neurones soient à état binaire ($x_i \in \{0,1\}$).

synaptiques. Sous réserve que les poids soient initialisés entre -1 et +1, l'algorithme d'apprentissage est alors donné par :

$$\Delta W_{ij}(t_i) = \alpha\left(1 - \left|W_{ij}(t_i - 1)\right|\right) H(t_i - t_j)\, x_j(t_j)\, \Delta x_i(t_i) \qquad \textbf{(VI-23)}$$

L'équation (VI-23) permet de limiter la plasticité des connexions et de stabiliser les poids (autours de ± 1) lorsque la fonction recherchée est apprise. Par contre, elle ne permet pas de remettre en cause l'apprentissage lorsque la fonction évolue au cours du temps. En effet, les poids proches de ± 1 sont très rigides et ne peuvent pas évoluer rapidement pour suivre, par exemple, une variation de l'environnement du réseau. Nous avons donc modifié le terme modérateur de Grossberg de façon à augmenter la plasticité synaptique lorsque de nouvelles contraintes apparaissent, c'est-à-dire lorsqu'un couple entrée/sortie nouveau vient contredire un ou plusieurs poids existants (Equation VI-24).

$$\Delta W_{ij}(t_i) = \left[1 - \left(sign\left(\alpha H(t_i - t_j)\, x_j(t_j)\, \Delta x_i(t_i)\right) W_{ij}(t_i - 1)\right)\right] \alpha H(t_i - t_j)\, x_j(t_j)\, \Delta x_i(t_i) \qquad \textbf{(VI-24)}$$

Comme dans l'équation (VI-23) le terme modérateur permet de limiter la croissance des poids à l'intervalle $[-1 \quad 1]$ (on remarquera cependant que l'utilisation du terme modérateur n'est possible que dans l'intervalle $[-1 \quad 1]$). Par contre, contrairement à l'équation VI-23, il autorise des modifications rapides des poids synaptiques lorsque des connexions activatrices (+1) ou inhibitrices (-1) sont remises en cause lors de l'apprentissage (figure VI-7).

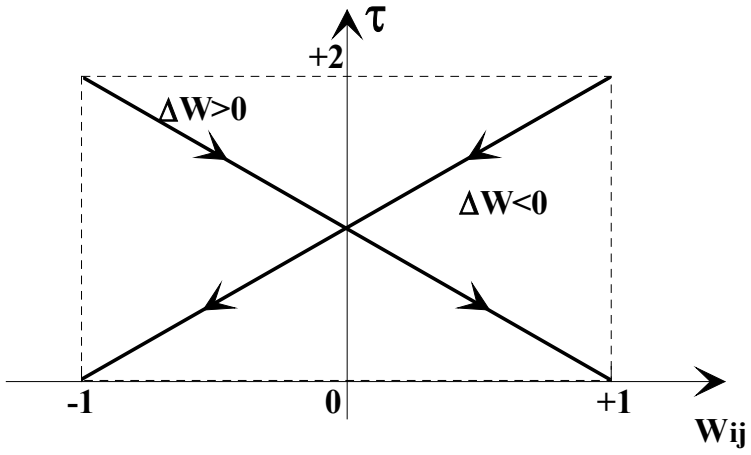


Figure VI-7 : *amplitude de variation des poids synaptiques*

Le terme modérateur (τ) varie suivant la valeur du poids synaptique et le signe de ΔW.

Les différents algorithmes décrits dans cette section sont avant tout issus de raisonnements théoriques. Les tests effectués sur le mode ordonnancé (particulièrement l'apprentissage de fonction) nous ont permis d'avoir des informations pratiques sur leur comportement. Pour plus d'informations, on se rapportera aux sections VI.4.3 à VI.4.6.

VI.3.4.3. Le cas des états non-répondants

Nous avons proposé dans la section précédente différentes interprétations de l'apprentissage hebbien. L'utilisation de ces mécanismes d'apprentissage dans le contexte du

mode ordonnancé (et du codage « par chemins » de l'information sensori-motrice) est cependant soumise à une condition particulière : que l'influx nerveux circule effectivement de l'entrée à la sortie du réseau, c'est-à-dire que l'apprentissage conduise effectivement au renforcement de boucles sensori-motrices. Or, nous avons vu (section VI.3.2.3) que l'architecture des réseaux utilisés par le mode ordonnancé peut conduire à l'apparition d'*états non-répondants* pour lesquels l'influx entrant dans le réseau n'est pas transmis jusqu'à la sortie.

Les états non-répondants susceptibles d'apparaître dans un réseau sont de deux types différents :

- **Les stabilités précoces :** le réseau atteint un point fixe sans agir sur les cellules de sortie.

- **Les cycles bloquants :** le réseau présente un cycle interne mettant en jeu au moins deux neurones dont les déséquilibres sont trop importants pour permettre la propagation de l'influx nerveux vers d'autres cellules du réseau. Les cycles bloquants sont donc une conséquence négative de l'utilisation du mode ordonnancé.

Suivant les conditions d'utilisation du réseau, les états non-répondants peuvent être acceptés ou non. Ainsi, en neurocontrôle, l'existence de stabilités précoces permet au réseau de mémoriser l'évolution de son environnement (le contexte) sans modifier sa sortie afin d'agir en conséquence lors de modifications ultérieures. Par contre, dans des applications de mémorisation hétéro-associatives (plus proches de la reconnaissance de forme), les états non-répondants ne sont plus acceptables : à chaque vecteur d'entrée présenté au réseau correspond un unique vecteur de sortie. C'est pourquoi nous avons inclus, dans les algorithmes d'apprentissage, des procédures permettant d'éliminer les états non-répondants.

Elimination des stabilités précoces :

Des deux formes d'états non-répondants, la stabilité précoce est probablement la plus simple et la plus facile à détecter : il y a stabilité précoce lorsque l'ensemble des déséquilibres du réseau est négatif ou nul (le réseau a atteint un point fixe) avant qu'une réponse interprétable n'ait été observée sur la couche de sortie. L'apparition de stabilité(s) précoce(s) dans le réseau est donc directement liée à l'existence de chemins reliant les entrées aux sorties, c'est-à-dire aux valeurs relatives des poids (W_{ij}) et des seuils (θ_i) dans le réseau. Les stabilités précoces peuvent donc être éliminées très simplement en augmentant légèrement les poids du réseau ou en diminuant légèrement les seuils des neurones (ce qui correspond à augmenter l'ensemble des déséquilibres dans le réseau). L'action sur les seuils est préférable car elle ne modifie pas les chemins existants : tous les déséquilibres étant augmentés de la même valeur ($\Delta d = -\Delta\theta$) leur ordre (et donc l'ordre de mise à jour) n'est pas modifié. Ce n'est pas le cas lorsqu'on agit sur les poids car la variation de déséquilibre d'un neurone est alors fonction du nombre de neurones actifs (c'est-à-dire des neurones $i, (i = 1..n)$, tels que $x_i = 1$):

$$\Delta d = \sum_{i=1}^{n} x_i \Delta W \qquad \textbf{(VI-25)}$$

La procédure proposée pour l'élimination des stabilités précoces suppose cependant que celles-ci soient générées par une insuffisance dans la propagation de l'influx nerveux. Cependant, des états stables peuvent aussi être générés par la saturation du réseau (par exemple dans un réseau ne comprenant que des connexions positives). Dans ce cas il faut non pas augmenter les poids (ou diminuer les seuils) mais diminuer les poids (respectivement augmenter les seuils), ce

qui correspond au fonctionnement normal des lois hebbiennes : l'activation simultanée de deux neurones, si elle conduit à un signal de renforcement négatif, entraîne la diminution de la connexion synaptique qui les relie. La stabilité par saturation peut donc être directement évitée par les lois hebbiennes.

Elimination des cycles bloquants :

Les cycles bloquants sont des phénomènes beaucoup plus complexes que les stabilités précoces. En effet, l'origine des cycles bloquants est double. Ils sont à la fois une conséquence de la topologie du réseau (qui ne respecte pas les conditions du théorème de Cohen-Grossberg) et un épiphénomène propre à l'utilisation du mode ordonnancé avec un déséquilibre instantané : un cycle comportant des déséquilibres importants peut compromettre la propagation de l'activité (figure VI-8). De plus, par rapport aux stabilités précoces qui peuvent être considérées comme des phénomènes globaux (concernant l'ensemble des neurones du réseau), les cycles bloquant sont essentiellement des phénomènes locaux ne concernant par un petit nombre de connexions.

Contrairement aux stabilités précoces les cycles bloquants ne peuvent donc pas être éliminés par des procédures globales d'adaptation des paramètres du réseau. Une première solution pour leur élimination semble être l'utilisation d'un déséquilibre cumulé (section VI.3.2.2, équation VI-11). Celui-ci permet en effet, par accumulation, de mettre à jour des neurones dont le déséquilibre instantané reste faible. Ainsi, dans le cas du réseau présenté figure VI-8a, l'utilisation du déséquilibre cumulé avec un facteur d'atténuation $\gamma = 0.5$ permet d'activer le neurone « 4 » grâce à l'accumulation du déséquilibre instantané sur deux pas de temps successifs (figure VI-9). Malheureusement, nous avons vu que le cumul du déséquilibre conduit le réseau à développer des séquences d'activation complexes en provoquant la mise à jour de neurones dont le déséquilibre instantané est négatif ou nul (ce qui est le cas, figure VI-9b, pour le neurone 4 lorsqu'il est activé au temps 5) ou en générant des stabilités temporaires lorsque le déséquilibre cumulé est fortement négatif et que le déséquilibre instantané est faiblement positif (figure VI-9b, temps 9). En outre, l'utilisation du déséquilibre cumulé permet d'éviter les cycles bloquants dus à la mise à jour ordonnancée mais en aucun cas il ne permet d'éviter les cycles provoqués par l'apparition de topologies particulières dans le réseau (figure VI-8b).

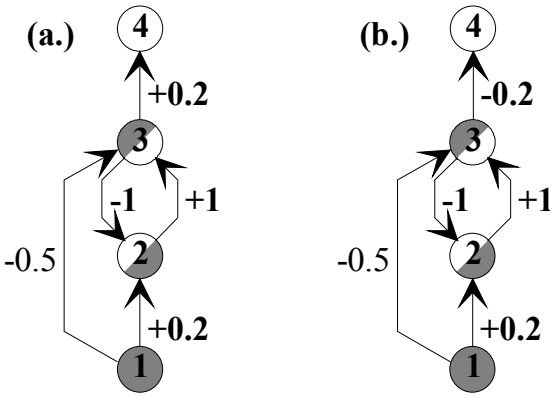


Figure VI-8 : *Exemple de cycles bloquants (entre les neurones 2 et 3) dans un réseau à mode ordonnancé.*

(a.) : Bien qu'il soit régulièrement en déséquilibre, le neurone 4 n'est pas activé car on a, suivant les états internes du cycle, $d_2 > d_4$ *ou* $d_3 > d_4$.

(b.) : Cycle bloquant dû à la topologie du réseau.

(a.)

t	0	1	2	3	4	5	6	7	8	9	10
Neurone 1	■	■	■	■	■	■	■	■	■	■	■
Neurone 2		■	■			■	■			■	■
Neurone 3			■	■			■	■			■
Neurone 4											

(b.)

t	0	1	2	3	4	5	6	7	8	9	10
Neurone 1	■	■	■	■	■	■	■	■	■	■	■
Neurone 2		■	■				■	■			
Neurone 3			■	■				■	■	■	
Neurone 4						■	■	■	■	■	■

: neurones inactif ■ : neurones actifs

Figure VI-9 : *Séquences d'états dans le réseau de la figure VI-8a*

(a.) : séquences de mise à jour obtenue avec le Déséquilibre instantané
(b.) : séquence de mise à jour obtenue avec le déséquilibre cumulé, $\gamma = 0.5$

Les cycles étant des phénomènes locaux, nous avons cherché des procédures d'apprentissage locales permettant d'éviter leur réapparition. Or, les états cycliques sont provoqués par l'existence de boucles de feed-back négatives internes au réseau. C'est pourquoi nous avons utilisé un algorithme permettant de supprimer ces boucles.

Lorsqu'un cycle est détecté, c'est-à-dire lorsqu'un même état est répété au cours des séquences de mise à jour, un algorithme hebbien (correspondant à l'équation VI-20) est utilisé pour renforcer les connexions du cycle, ce qui conduit à limiter les déséquilibres correspondant et donc à autoriser la propagation de l'influx vers d'autres neurones (figure VI-10). Si le cycle est dû à la topologie du réseau (cas présenté figure VI-8b), cet algorithme conduit à l'apparition de stabilités puis, par la diminution générale des seuils du réseau, à la propagation de l'influx.

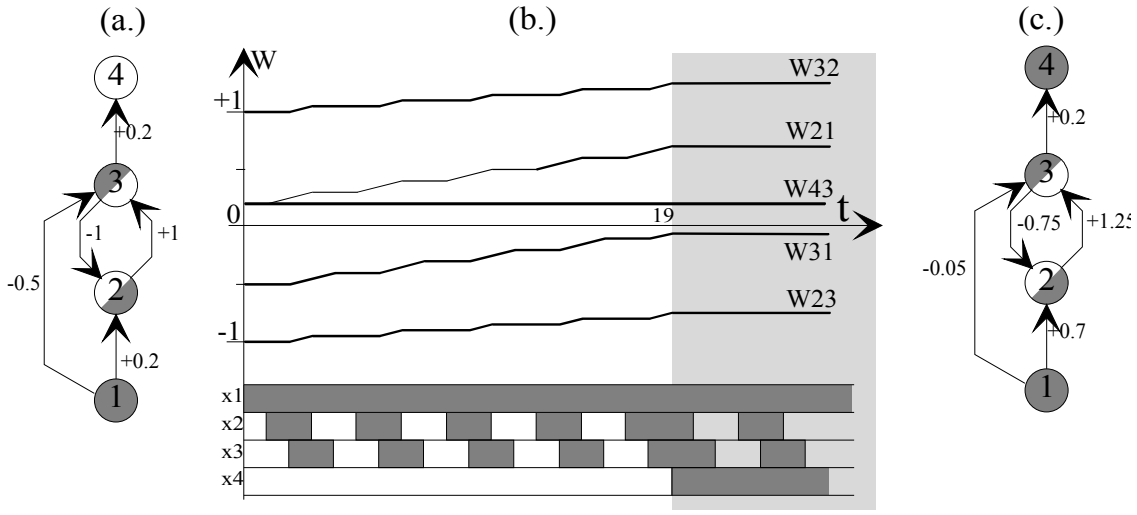


Figure VI-10 : *Elimination des cycles bloquants par un algorithme hebbien.*

(a.) Réseau initial. Le cycle provoque l'activation simultanée des neurones 1, 2 et 3.

(b.) L'apprentissage hebbien ($\alpha = 0.05$) permet de renforcer progressivement les connexions du cycle (les connexions des neurones 2 et 3 vers le neurone 1 ne sont pas renforcées grâce à la fonction de Heaviside utilisée pour l'apprentissage, cf. équation VI-20). Au temps $t = 19$, les connexions ont été suffisamment modifiées pour débloquer le cycle car on a $d_4 > d_2$ et $d_3 < 0$.

(c.) Le nouveau réseau présente toujours un cycle mais il n'est plus bloquant.

VI.4. Premiers tests et développements

VI.4.1. Introduction

Au cours des sections précédentes (VI.3.2, VI.3.3 et VI.3.4), nous avons décrit les différentes composantes d'un algorithme connexionniste susceptible de permettre l'émergence d'assemblées de neurones dans un réseau récurrent. Nous allons maintenant présenter succinctement les différents tests effectués pour valider notre modèle de réseau. Les annexes A2, A3, A4 et A5 présentent plus en détail les différentes applications testées et analysent plus finement les résultats obtenus.

Nous avons appliqué le mode ordonnancé à quatre problèmes différents choisis pour leur complexité croissante mais aussi parce qu'ils permettent de vérifier la compatibilité du réseau avec le cahier des charges et de comparer les résultats obtenus grâce au mode ordonnancé et par d'autres approches. Il s'agit, successivement, de l'apprentissage de la fonction OU Exclusif, d'une application de reconnaissance couleurs, de l'apprentissage de séquences booléennes bouclées et du contrôle d'un pendule inverse.

VI.4.2. Caractéristiques générales des réseaux utilisés pour les tests

VI.4.2.1. Environnement de programmation

Les différents tests présentés ci-après ont été effectués sur ordinateur compatible PC (type « 386 » et « 486 ») en utilisant le langage de manipulation de matrices Matlab de « The MathWorks Inc. », choisi en raison des nombreuses fonctions de haut niveau (manipulation de matrice, visualisation, ...) qu'il offre, par rapport aux langages procéduraux type « C ». Le langage Matlab nous permet, en effet, d'utiliser directement les équations du mode ordonnancé dans les programmes. Nous avons ainsi pu tester directement différentes variantes de l'algorithme initial. Par contre, Matlab étant un langage interprété, la vitesse d'exécution a parfois été un facteur limitatif et nous envisageons, à l'avenir, de transposer les algorithmes vers un langage compilé type C++.

Outre les langages de programmation classiques, de nombreux logiciels de simulation de réseaux de neurones sont disponibles sur le marché (SNNS [ZEL 94], Neural-Basic [HAM 95], Matlab Neural-Networks Toolbox [DEM 92], ...). Cependant, bien qu'ils soient généralement interfaçables en « C », ce qui permet d'ajouter des fonctionnalités particulières, ils sont généralement limités à l'utilisation des réseaux « classiques » (réseaux en couches, cartes de Kohonen, réseaux de Hopfield) et ne permettent pas d'agir suffisamment librement sur les principes « standards » des réseaux neuronaux (en particulier le mode de mise à jour).

VI.4.2.2. Architecture

Dans tous les réseaux utilisés, l'architecture est identique à celle présentée section VI.3.2.3 : les unités, à état binaire, sont réparties en trois couches (entrée, afférence et sortie). La couche d'afférence est totalement connectée (sans autoconnexion). Les connexions sont donc réparties en trois familles distinctes : le fuseau d'entrée (connexions entre les neurones d'entrée et les neurones d'afférence), les connexions afférentes (connexions cachées internes à la couche d'afférence) et le fuseau de sortie (connexions entre les neurones d'afférence et les neurones de sortie). Nous avons vu (section VI.3.2.3) que des connexions réflexes peuvent être ajoutées à la topologie initiale. Cette possibilité n'a été utilisée que pour l'apprentissage de séquences booléennes afin de favoriser l'apparition d'un comportement type « Winner-Takes-All » entre les cellules de sortie.

Hormis les éventuelles connexions réflexes, les poids synaptiques sont initialisés aléatoirement dans un intervalle fixé dont les bornes sont des paramètres propres au réseau utilisé. Les neurones du réseau sont initialement à l'état inactif (le vecteur $\vec{x}$ est initialement un vecteur nul) et on considère que le réseau fournit une réponse lorsqu'il active une des cellules de sortie. Une variable booléenne doit donc être codée par deux neurones différents : un pour « vrai » et un pour « faux ».

VI.4.3. Apprentissage du OU EXclusif

Nous avons utilisé l'apprentissage du OU Exclusif comme première phase de test afin d'étudier le comportement du réseau sur une application simple, ne nécessitant que peu de neurones cachés. En outre, le OU Exclusif est un test élémentaire largement utilisé en connexionnisme car il permet de vérifier qu'un réseau peut distinguer deux classes non-linéairement séparables. Des éléments de comparaison sont donc disponibles ce qui nous permet

d'avoir un premier aperçu des performances du mode ordonnancé. Les résultats présentés ci-après sont détaillés dans l'annexe A2.

L'utilisation de neurones binaires ne permettant pas de coder efficacement une variable booléenne avec un seul neurone, nous utilisons un réseau possédant quatre neurones d'entrée (deux neurones par variable booléenne) et deux neurones de sortie (« VRAI » et « FAUX »). Le tableau VI-1 présente la table d'entrée/sortie du réseau. Le nombre de neurones d'afférence est variable (de 2 à 30 neurones). Les poids sont initialisés aléatoirement entre -0.8 et +0.8.

x	*y*	ENTREE	SORTIE	*x* xor *y*
VRAI	VRAI	0-1-0-1	1-0	FAUX
VRAI	FAUX	0-1-1-0	0-1	VRAI
FAUX	VRAI	1-0-0-1	0-1	VRAI
FAUX	FAUX	0-1-0-1	1-0	FAUX

Tableau VI-1 : *Table d'entrée/sortie du réseau pour le OU EXclusif*

Le signal de renforcement, utilisé comme facteur d'apprentissage, est de +0.005 en cas de réponse correcte et de -0.05 en cas de réponse erronée. Les états non répondants sont supprimés par une diminution des seuils de 0.1 en cas de stabilité précoce. Nous n'avons pas utilisé de procédure spécifique pour les cycles bloquants en raison de la faible taille des réseaux utilisés. Chaque session d'apprentissage comprend entre 100 et 500 présentations de couples entrée/sortie choisis aléatoirement dans la table d'entrée/sortie (tableau VI-1). L'apprentissage est considéré comme terminé lorsque le réseau a répondu correctement pendant plus de 25, 50 ou 100 présentations successives (critères APP25, APP50 et APP100). Nous avons testé les différents algorithmes d'apprentissage présentés section VI.3.4.2, (équation VI-20 à VI-24). Seuls les algorithmes d'apprentissage n'utilisant pas de termes modérateurs ont permis d'apprendre le OU Exclusif[76]. Les résultats obtenus pour chacun des trois critères sont présentés figure VI-11.

Celle-ci nous montre que le réseau parvient à apprendre le OU EXclusif mais avec une forte dépendance aux conditions initiales lorsqu'il est faiblement dimensionné. La différence entre les trois critères d'apprentissage montre que l'algorithme d'apprentissage converge de façon irrégulière (l'apprentissage est dit « instable ») : lorsque la fonction OU EXclusif a été apprise, la prolongation de l'apprentissage provoque l'apparition sporadique d'erreurs ce qui retarde, voire interdit, la convergence pour les critères APP50 et APP100.

[76] L'utilisation d'un terme modérateur provoque la normalisation des poids ce qui est incompatible avec l'apprentissage du OU Exclusif (voir annexe A2, section 2)

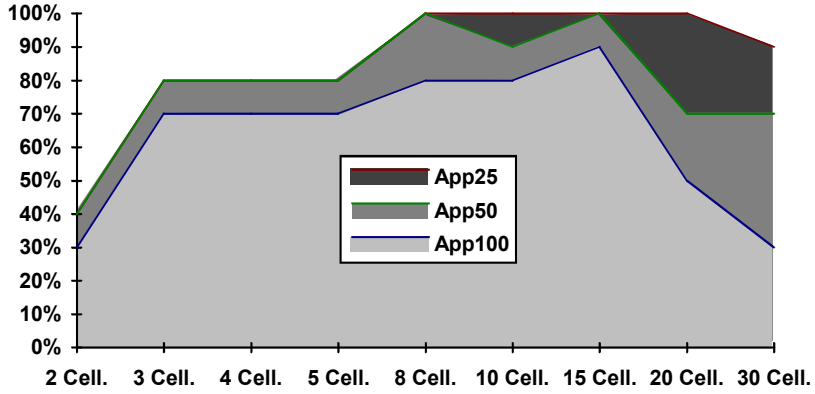


Figure VI-11 : *Convergence de l'apprentissage en fonction de la taille du réseau*

Chaque valeur correspond au pourcentage de réseaux ayant convergé en moins de 500 itérations pour chacun des trois critères (10 essais pour chaque configuration).

La figure VI-12 présente le temps de convergence nécessaire à l'apprentissage du OU EXclusif pour chacun des trois critères de convergence. Le nombre d'itérations nécessaire, de l'ordre de quelques dizaines, est similaire ou inférieur à celui obtenu pour d'autres types de réseaux (d'après Pearlmuter [PEA 89], Gullapalli [GUL 90] et Ichikawa [ICH 92][77]). Le nombre de cellules sur la couche cachée apparaît cependant comme un facteur important pour le temps d'apprentissage.

Les différents tests effectués montrent que l'apprentissage est fortement dépendant de l'apparition d'états non-répondants. Or, dans plus de 20% des cas observés (tous réseaux confondus), la première itération correspond à un état non-répondant. Ce comportement est dû aux valeurs d'initialisation utilisées et à la faible taille des réseaux : statistiquement, dans un réseau de n neurones cachés et avec les valeurs d'initialisation utilisées, un neurone de sortie a une probabilité de $1/2^n$ d'être totalement isolé (c'est-à-dire desservi exclusivement par des connexions négatives). L'utilisation d'un apprentissage par renforcement ne permettant pas de créer des chemins d'activation lorsque ceux-ci sont inexistants, ces configurations ne permettent pas l'apprentissage autrement que par une activation spontanée de la cellule de sortie isolée.

Pour assurer l'existence de chemins d'activation pour toutes les configurations possibles, il suffit d'initialiser positivement les poids (ou, au moins, que la moyenne des poids soit positive). Ainsi, dans les tests suivants, les poids sont généralement initialisés entre 0 et 0.2. L'annexe A2 (section 2) présente une analyse plus complète du problème des états non-répondants et de l'instabilité de l'apprentissage.

---

[77] Pearlmuter propose un modèle de réseau récurrent (avec autoconnexions), dont la topologie est fixée par l'utilisateur, apprenant le OU EXclusif en 100 itérations (en moyenne) [PEA 89]. D'après l'auteur ce résultat est comparable aux résultats obtenus dans un réseau en couches utilisant la rétropropagation. Ichikawa utilise un algorithme génétique pour fixer les poids d'un réseau en couches en vue d'apprendre le OU EXclusif. La convergence est obtenue après 30 générations de 20 réseaux [ICH 92]. Lors des tests des SRV-Units, V. Gullapalli n'obtient une convergence qu'après plusieurs milliers d'itérations [GUL 90]

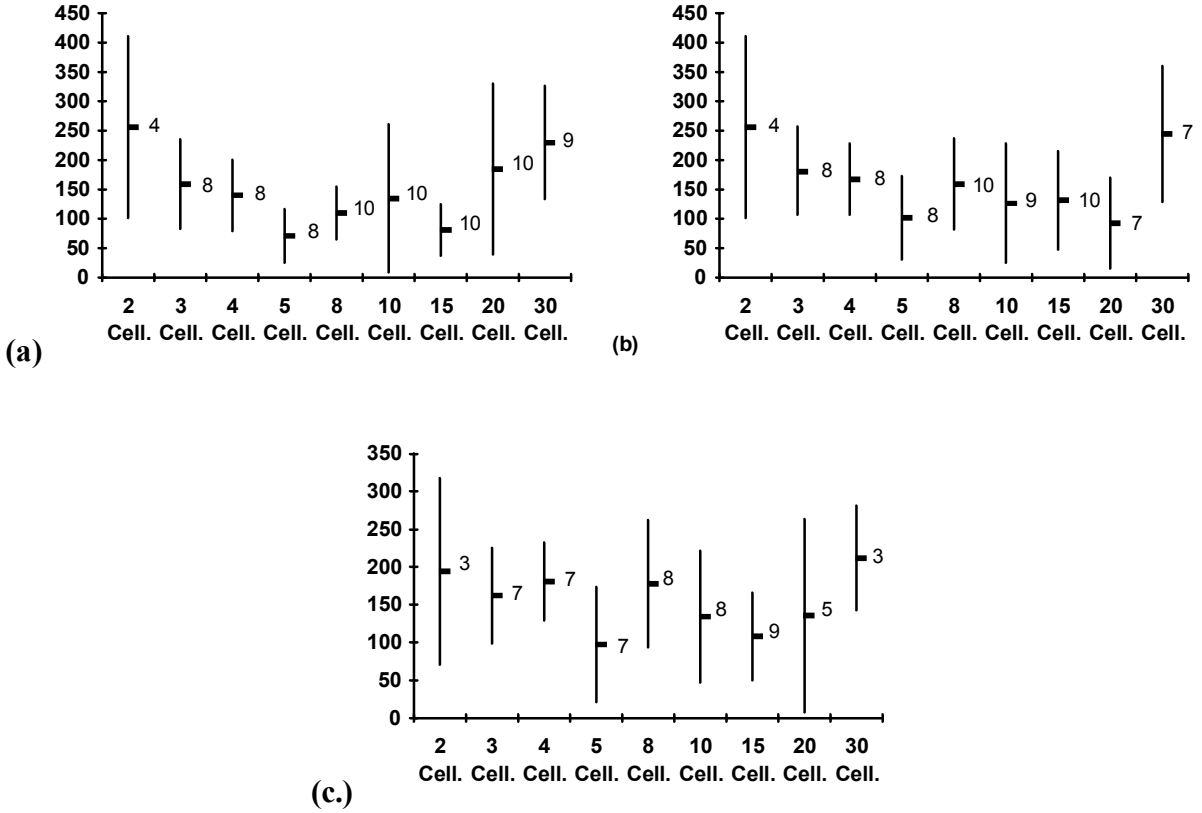


Figure VI-12 : *Moyenne et écart type sur les réseaux ayant convergé en moins de 500 itérations (pour chaque configuration, le nombre de réseaux ayant convergé est indiqué sur la figure).*
(a.) : Critère APP25 (b.) : Critère APP50 (c.) : Critère APP100

L'analyse des matrices de poids et des chemins d'activation appris par le réseau illustre la représentation des données dans le réseau : le réseau concentre l'information apprise sur une petite partie des connexions ce qui lui permet de n'utiliser qu'un nombre restreint de neurones : quelle que soit la taille initiale du réseau, le nombre de neurones utilisés est inférieur à dix. Cependant le OU Exclusif est un problème trop limité pour que l'on puisse en déduire des informations pertinentes concernant l'émergence d'assemblées de neurones.

VI.4.4. Classification : la reconnaissance de couleurs

A partir des connaissances acquises sur l'apprentissage du OU EXclusif, nous avons cherché à appliquer le mode ordonnancé à un problème plus concret en comparant nos résultats avec ceux fournis par d'autres méthodes.

Dans le cadre d'une collaboration avec le Centre de Recherche en Automatique de Nancy (équipe Electricité, Electronique et Automatique), nous avons utilisé le mode ordonnancé pour une application de reconnaissance couleurs dans un contexte de vision artificielle. Le but de cette application est l'identification robuste d'objets de forme identique, donc uniquement différenciables par leur couleur. Cette identification doit être effectuée dans des conditions d'éclairage variables, en particulier en intensité. Cette application a fait l'objet d'une étude approfondie dans le cadre de l'Atelier Inter-établissement de Productique de Lorraine et plusieurs méthodes de classification ont été utilisées (méthodes statistiques, réseaux de neurones, classification floue, algorithmes génétiques) [BOM 94]. Il est donc possible de comparer les

résultats obtenus avec le mode ordonnancé sur les mêmes lots de données que ceux utilisés précédemment. De plus, ce problème permet de tester les capacités de généralisation de l'algorithme, ce qui n'était pas possible avec des fonctions booléennes.

L'application développée à l'AIP Lorraine comprend un bras robot, un tapis roulant et un système de vision couleur par caméra CCD. Le problème posé est la reconnaissance des différents objets en présence, c'est-à-dire des pièces (rouges, bleues ou jaunes) et du tapis roulant (gris), à partir du signal RVB[78] fourni par la caméra (le champ de la caméra contient une zone de blanc de référence). Ce signal étant difficile à exploiter (il contient à la fois l'information luminance et l'information chrominance), un pré-traitement a été effectué afin de passer du repère RVB au repère CIE LUV[79] qui sépare l'information luminance (L^*) des informations chrominance (u^* et v^*). Cela permet d'éliminer l'information luminance, non pertinente pour notre problème, et donc de réduire la dimension du problème de classification en passant d'un espace tridimensionnel (R, V, B) à un espace bidimensionnel (u^* et v^*). Le passage du repère RVB au repère CIE LUV est présenté en annexe A3, section 1.

Les données utilisées pour l'apprentissage sont issues de 6 phases d'acquisition (correspondant à six valeurs d'éclairement différentes) au cours desquelles 400 mesures ont été effectuées pour chaque couleur, soit un total de 9 600 points (figure VI-13). Ceux-ci ont ensuite été répartis en deux lots de données de taille identique (un lot de mémorisation et un lot de généralisation).

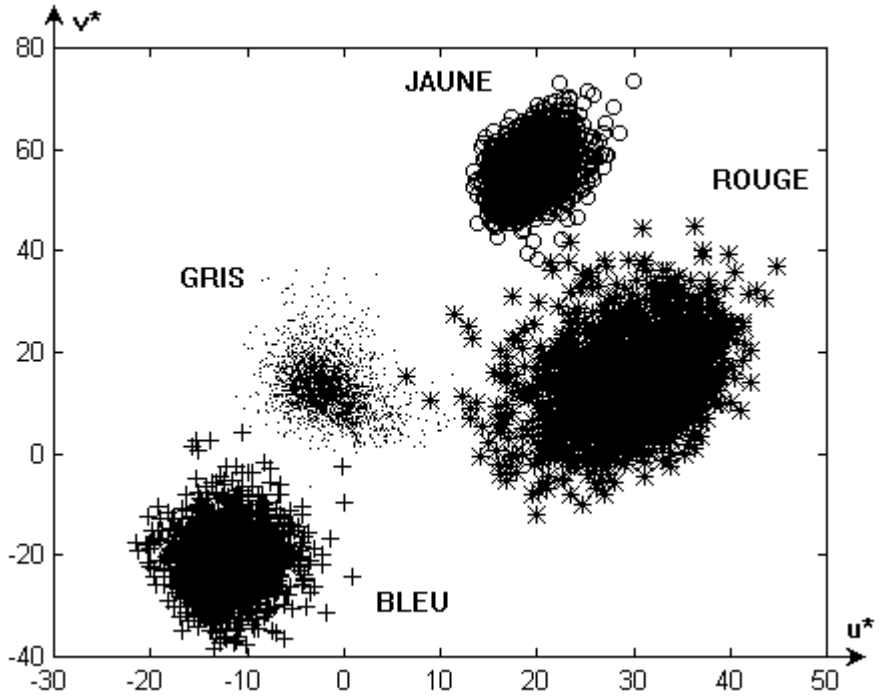


Figure VI-13 : *Représentation des lots de données dans le repère (u*,v*)*

[78] Rouge, Vert, Bleu.

[79] Le repère CIE LUV est un repère standard de la Commission Internationale de l'Eclairage,

Plusieurs méthodes de classification ont déjà été appliquées à ce problème [BOM 94] :

- Un classificateur Bayesien (CB)
- Un réseau de neurones type LVQ1 (Learning Vector Quantization)
- Des classificateurs flous en 6, 7 (CF7) et 9 (CF9) termes linguistiques avec éventuellement ajustement manuel des fonctions d'appartenance (7 termes, CF7M) ou ajustement par algorithmes génétiques (6 termes, CF6G) [RON 94].

Les résultats obtenus sont les suivants (tableau VI-2) :

	CF7	CF7M	CF9	CF6G	LVQ1	CB
Erreur de mémorisation	0.60%	0.16%	0.14%	0.06%	0.42%	0.06%
Erreur de généralisation	0.49%	0.28%	0.18%	0.08%	0.24%	0.11%

Tableau VI-2 : *Performances comparées des différentes méthodes de classification appliquées au problème de la reconnaissance couleurs, d'après [BOM 94] et des communications personnelles de Laurent Rondeau (fuzzification en 6 termes avec ajustement des règles par algorithmes génétiques). Le lot de mémorisation, comme le lot de généralisation, compte 4 800 points.*

Etant données les caractéristiques particulières du problème (séparation de quatre classes dans un espace réel à deux dimensions), nous avons appliqué le mode ordonnancé de deux manières différentes :

- Avec un réseau d'unités à valeur binaire, en discrétisant l'espace d'entrée en 20*20 zones (20 neurones d'entrée pour u^* et 20 neurones d'entrée pour v^*).

- Avec un réseau dont seules les cellules d'entrée sont à valeur réelle ($x_i \in [0\,1]$), les cellules de la couche cachée et de la couche de sortie étant à valeur binaire. Les valeurs d'entrées sont semi-distribuées sur 42 neurones au moyen de 42 fonctions triangulaires de u^* ou v^* (21 neurones pour chaque variable d'entrée) [LOR 91]. Cette représentation permet d'affiner les frontières interclasses sans remettre en cause les principes fondamentaux du mode ordonnancé.

En outre, les réseaux utilisés comptent 4 cellules de sortie (une par couleur) et 20 neurones d'afférence. Nous avons tenu compte, pour ce réseau, des problèmes de stabilité précoce évoqués pour le OU EXclusif. Les poids synaptiques sont donc initialisés aléatoirement entre 0 et 0.2 (fuseau d'entrée et connexions afférentes) ou entre 0.1 et 0.3 (fuseau de sortie). Tous les seuils sont fixés à zéro. Cette initialisation est suffisante pour éviter les états non-répondants. Nous n'avons donc pas utilisé d'algorithme spécifique pour les éviter (un état non répondant est considéré comme une erreur du réseau). Les exemples de la base d'apprentissage sont présentés dans un ordre totalement aléatoire. Le signal de renforcement est de +0.05 en cas de succès, -0.05 en cas d'erreur et -0.1 en cas d'état non répondant. L'algorithme d'apprentissage utilisé est le MRT Learning orienté avec un terme modérateur orienté (équation VI-24). Contrairement aux résultats obtenus avec le OU EXclusif, cet algorithme est ici le plus performant. Pour chacun des deux réseaux nous avons obtenu les résultats suivants :

- réseau à entrées binaires (après 20 000 présentations d'exemples, soit environ 4.5 fois la base d'apprentissage) :

  Erreur de mémorisation : 0.0890 % (sur 4 800 points)

  Erreur de généralisation : 0.1335 % (sur 4 800 points)

- réseau à entrées réelles (après 4 800 présentations d'exemple, soit environ une fois de la base d'apprentissage) :

  Erreur de mémorisation : 0.0667 % (sur 4 800 points)

  Erreur de généralisation : 0.0890 % (sur 4 800 points)

Les résultats obtenus avec le mode ordonnancé sont donc équivalents à ceux obtenus avec les méthodes de classification les plus performantes. On remarquera cependant que, comme pour le OU Exclusif, l'apprentissage est instable. Dans le cas du réseau à entrées réelles, la prolongation de l'apprentissage provoque une baisse de performances : après deux présentations de la base d'exemple, l'erreur de mémorisation remonte à 0.1112 % et l'erreur de généralisation à 0.1335 %. Ce comportement est dû à la normalisation des poids : la fonction de décision du réseau dépend directement des pondérations du fuseau d'entrée et la normalisation de celui-ci restreint les possibilités de placement des frontières interclasses.

L'étude des matrices de poids montre que le réseau n'utilise que 8 neurones sur 20 pour reconnaître les quatre couleurs, chacune d'elles étant associée à une paire de cellules que l'on peut considérer comme un « groupe fonctionnel » associé à une couleur particulière. Cette caractéristique illustre l'intérêt du mode ordonnancé par rapport à l'ensemble des classificateurs utilisés :

- sa capacité à s'auto-dimensionner permet de limiter les temps de calcul. D'après V. Bombardier [BOM 94] les temps de calcul[80] s'échelonnent entre 48h pour le LVQ1 et quelques minutes pour l'approche floue (mais celle-ci demande ensuite un ajustement manuel pour donner de bons résultats, l'ajustement par un algorithme génétique demandant plus de 24h de calcul). En utilisant le mode ordonnancé il faut moins de 15 minutes pour 4 600 itérations (c'est-à-dire une présentation complète de la base d'apprentissage). En 7 minutes les performances sont donc déjà de 0.16 % / 0.29 % pour le réseau à entrées semi-distribuées et les performances maximum sont obtenues en un quart d'heure.
- La représentation semi-distribuée permet au réseau de tenir compte d'éléments peu représentés dans la base d'apprentissage : pour chaque couleur, la grande majorité des points de la base est regroupée à proximité immédiate du centre de la classe (ils correspondent aux conditions optimales d'éclairement). Il est probable que cette répartition des exemples d'apprentissage soit à l'origine des faibles performances affichées par le LVQ1 (la base d'apprentissage étant très mal répartie, les zones faiblement représentées sont difficilement apprises). Par contre, le mode ordonnancé permet de tenir compte de ces points particuliers en générant des chemins spécifiques pour lesquels l'apprentissage est accéléré par l'utilisation d'un terme modérateur orienté.
- Le mode ordonnancé est simple à mettre en oeuvre : l'auto-dimensionnement induit par la représentation semi-distribuée des connaissances dans les « assemblées de neurones »

---

[80] Tous les algorithmes ont été testés sur la même machine (compatible PC type 486/66 avec 32 Mo de mémoire vive) et sous le même environnement : « Matlab/Windows ».

permet de simplifier le paramètrage du réseau : seules les valeurs d'initialisation des poids et les facteurs d'apprentissage sont déterminants pour le fonctionnement du réseau. On notera que l'initialisation des poids a une influence importante sur l'apparition d'états non-répondants (contrairement au cas du OU EXclusif, l'apprentissage n'a pas été accompagné, ici, par l'apparition d'états non-répondants).

VI.4.5. Apprentissage de séquences booléennes

Que ce soit dans la définition générale d'un contrôleur Neuro-Réactif (section V.4), dans l'étude des systèmes multi-agents (section III.4) ou dans l'étude d'un système de transport totalement décentralisé (section II.3.5), nous avons vu que le contrôleur utilisé doit être capable de générer des séquences comportementales éventuellement bouclées. Or, l'apprentissage par renforcement de séquences bouclées est un problème complexe car les différents états de la séquence ne sont pas différenciables par les entrées du réseau. Celui-ci doit donc apprendre à reconnaître des états qui, bien que possédant les mêmes entrées, ne conduisent pas à la même sortie. Il doit pour cela disposer d'une mémoire à long terme (les connexions synaptiques) mais aussi d'une mémoire à court terme lui permettant de suivre le déroulement de la séquence (les activités neuronales).

La séquence utilisée lors des tests comporte quatre états A, B, C, D mais le passage de l'un à l'autre dépend de l'avancement dans la séquence (figure VI-14). L'état courant est présenté en entrée et le réseau qui doit produire l'état suivant. Celui-ci est alors présenté en entrée et ainsi de suite...

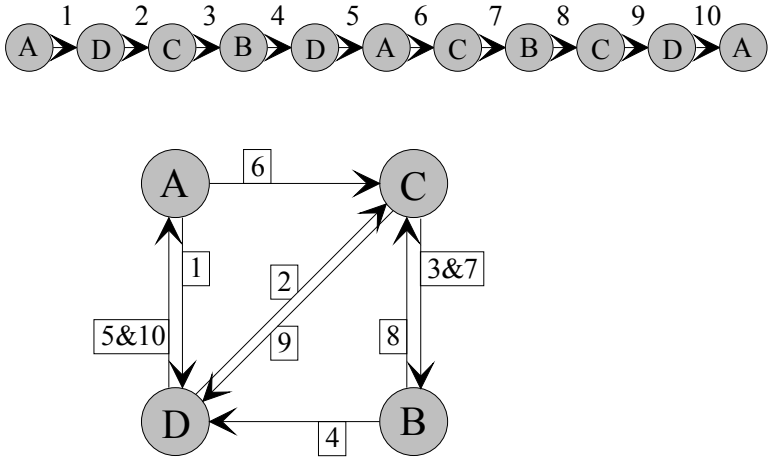


Figure VI-14 : *La séquence utilisée pour les tests*

La séquence est d'ordre 2 (les états T-1 et T-2 sont nécessaires pour déterminer l'état T) et compte 11 pas pour quatre états différents.

Le réseau utilisé pour l'apprentissage de cette séquence compte 4 neurones d'entrée, 4 neurones de sortie et 50 neurones cachés. Les poids sont initialisés entre -0.1 et +0.1 (pour le fuseau d'entrée et les connexions afférentes) et entre 0 et +0.2 pour le fuseau de sortie. Le signal de renforcement est fixé à +0.005 en cas de réussite et -0.005 en cas d'erreur. Contrairement aux applications de reconnaissance de formes et de classification, les neurones d'afférence et de sortie ne sont pas remis à zéro lorsqu'un nouveau vecteur d'entrée est présenté au réseau. On considère que le réseau fournit une réponse lorsqu'un neurone de sortie est mis à jour et que cela conduit la couche de sortie à ne posséder qu'un seul élément actif. Cette connaissance a priori du fonctionnement du système est mise en oeuvre par des connexions réflexes (donc non apprises) inhibitrices entre les différents neurones de la couche de sortie. Etant donnée la combinatoire de ce problème, nous avons guidé l'apprentissage par la méthode des « sorties forcées[81] » [WIL 89].

L'utilisation du mode ordonnancé nous permet de présenter la séquence par éléments successifs tant que le réseau répond correctement (apprentissage par partie). En cas d'échec, les neurones sont ré-initialisés et la séquence est reprise au premier élément. Ce type d'apprentissage « par partie » (les éléments successifs de la séquence sont appris les uns après les autres) est rendu possible par la localisation des informations sur un petit nombre de connexions. Dès lors, de nouveaux pas de la séquence peuvent être appris sans risquer un oubli catastrophique.

Le réseau utilisant la sortie forcée a réussi à apprendre la séquence complète dans neuf cas sur dix. Le temps d'apprentissage moyen est de 254.3 itérations. L'écart type est de 127 itérations (le réseau ayant convergé le plus rapidement a appris la séquence complète en 86 itérations et le plus lent en 473 itérations). Le réseau n'ayant pas convergé est parvenu à apprendre les dix premiers pas de la séquence en 120 itérations (l'apprentissage a été stoppé après 1 000 itérations). Les courbes d'apprentissage montrent que le réseau apprend très rapidement les premiers pas de la séquence (pour lesquels il n'y a pas de conflits dus aux bouclages), après quoi l'apprentissage ralentit car le réseau doit parfois remettre en cause les phases d'apprentissage antérieures (figure VI-15).

Afin de tester la robustesse de l'apprentissage, nous avons présenté au réseau l'état initial « A » et nous avons observé combien de pas de la séquence sont retrouvés sans présentation des entrées associées. Lors de ce test, le réseau est parvenu à retrouver les six premiers états de la séquence (« ABCBDA ») ce qui correspond à une bonne robustesse. Ce test, joint à la capacité du réseau à différencier les boucles de la séquence, nous permet d'affirmer qu'il a bien développé une mémoire à court terme basée sur l'état de ses cellules d'afférence.

Bien qu'il soit pratiquement impossible d'étudier manuellement la représentation de l'information dans le réseau (qui compte près de 2 800 poids différents), nous avons cherché à identifier les fonctions des différents neurones utilisés (26 sur 50 pour le réseau étudié). Pour cela nous avons étudié les séquences d'activation permettant au réseau de retrouver l'ensemble de la séquence. Les résultats exacts sont présentés en annexe A4. On note principalement l'existence de différentes classes de neurones selon qu'ils sont associés à une entrée particulière (A, B, C ou D) ou à une sous-séquence particulière. On peut ainsi identifier les règles permettant au réseau de reconnaître les différents états bouclés. La structure observée est relativement proche de celle d'une carte cognitive : les cellules sont reliées par des liens de cause à effet dépendant directement de la séquence apprise.

---

[81] « Teacher-Forced Learning » : lors de l'apprentissage, des corrélations fictives sont associées aux connexions aboutissant à la sortie désirée ce qui facilite le développement des chemins d'activation correspondants. En outre, la sortie forcée permet d'éliminer efficacement les états non-répondants.

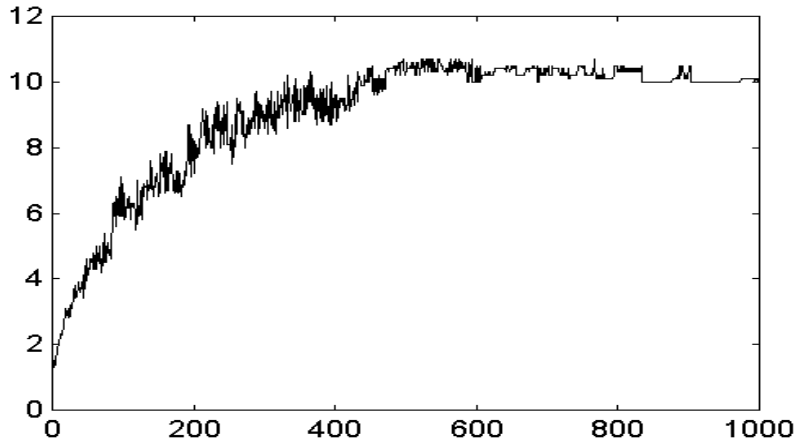


Figure VI-15 : *évolution de l'apprentissage en fonction du temps.*

La courbe représentée est une moyenne sur les 10 sessions d'apprentissage. Dans un premier temps (itérations 0 à 100), l'apprentissage est très rapide et la courbe présente peu d'oscillations car le réseau n'a pas encore dû apprendre les boucles. De 100 à 450 itérations environ, la courbe d'apprentissage oscille fortement car la progression dans la séquence entraîne des conflits avec les premiers pas appris. De 450 à 1 000 itérations, 9 des 10 sessions ont appris la séquence.

Comparé à d'autres modèles connexionnistes d'apprentissage de séquences, le mode ordonnancé ne permet pas d'augmenter la capacité d'apprentissage ni d'apprendre des séquences bouclées d'un ordre supérieur. Ses performances sont même inférieures à celles du modèle d'apprentissage de séquences motrices proposé par Bernard Ans [ANS 94]. Cependant nous n'avons pas relevé, dans la littérature, de modèles connexionnistes capables d'apprendre ce type de séquences sans représentation explicite des boucles internes : dans le réseau proposé par Bernard Ans, certaines cellules sont explicitement réservées à la mémorisation des différentes boucles internes. D'autres modèles, comme celui proposé par W. Miranker [MIR 94], codent explicitement la séquence par l'activation d'une série de neurones identifiant les pas successifs, neurones connectés directement aux cellules de sortie appropriées. Au contraire, le mode ordonnancé permet d'apprendre une séquence de façon implicite, sans utiliser de cellules spécifiques : la représentation semi-localisée de l'information et l'émergence d'assemblées de neurones permettent à un réseau totalement connecté de se spécialiser « par partie » et d'acquérir spontanément une structure interne semblable à celle mise en place explicitement, par l'homme, dans un réseau de neurones devant résoudre un problème similaire.

VI.4.6. Neurocontrôle : le pendule inverse

Les trois phases de test précédentes étaient essentiellement destinées à tester les potentialités du mode ordonnancé et à en étudier le comportement avant de l'utiliser pour des applications plus complexes. Nous avons ensuite appliqué le mode ordonnancé à un premier problème de neurocontrôle : l'apprentissage du contrôle d'un pendule inverse[82]. Cette application, très largement utilisée pour tester les algorithmes connexionnistes appliqués au contrôle de systèmes non linéaires ([BAR 83], [AND 89], [SAE 90], [LIN 91], [THU 91], [LAN 92]), fait figure d'étalon (« Benchmark ») dans le domaine du contrôle [GEV 93].

Le pendule inverse est constitué d'un chariot mobile se déplaçant sur une piste linéaire et portant un balancier pouvant pivoter autour d'un axe fixé sur le chariot (figure VI-16). Le système est simplement commandé en appliquant au chariot une force longitudinale F de norme constante, le but étant de déterminer, à chaque instant, le sens de cette force afin de maintenir le balancier en équilibre sans que le chariot ne sorte de la piste.

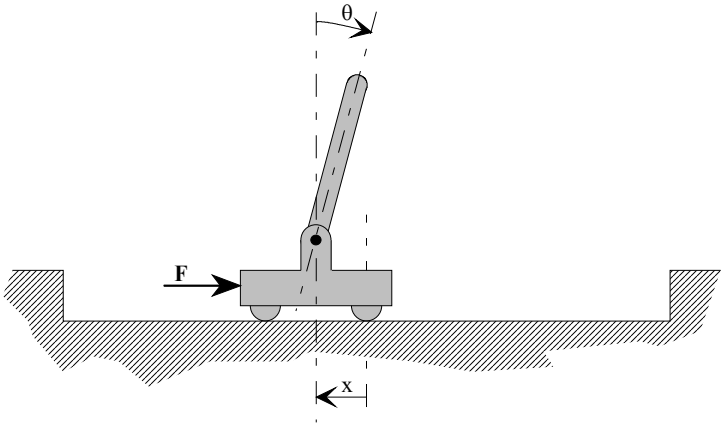


Figure VI-16 : *Le pendule inverse*

L'état du système est donné par quatre variables : θ, l'angle du balancier avec la verticale ; $\dot{\theta}$, la vitesse angulaire du balancier ; x la position de l'axe par rapport au centre de la piste ; $\dot{x}$ et la vitesse du chariot. Le domaine de viabilité du système est défini par rapport à l'angle du balancier (chute du balancier) et à la position de l'axe (le chariot sort de la piste). Les valeurs limites standardisées sont[83] : $x \in [-2.4m \quad 2.4m] \quad \theta \in [-12° \quad 12°]$

Nous avons utilisé l'architecture et les paramètres proposés dans [BAR 83] (section IV.4.3.1) mais le réseau de contrôle (l'Adaptive Search Element) est remplacé par un réseau utilisant le mode ordonnancé. Toutefois, ce réseau reçoit directement les quatre variables d'état et non, comme dans [BAR 83], l'identification de la « Boîte » dans laquelle se trouve le système à l'instant courant. Ces variables sont à valeurs discrètes, le maillage étant identique à celui utilisé

---

[82] « Cartpole problem » ou « inverted pendulum »

[83] Conformément à la standardisation proposée par [GEV 93] nous utilisons les paramètres de [BAR 83]. Le temps d'intégration de la commande, en particulier, est de 0.02 seconde.

pour construire les « Boîtes » de l'Adaptive Critic Element[84] (voir annexe A6). La figure VI-17 présente l'architecture générale retenue. L'apprentissage se déroule par essais successifs : au début de chaque essai le pendule est ré-initialisé à proximité de l'origine[85] et l'essai dure jusqu'à ce que le balancier tombe ou que le chariot sorte de la piste

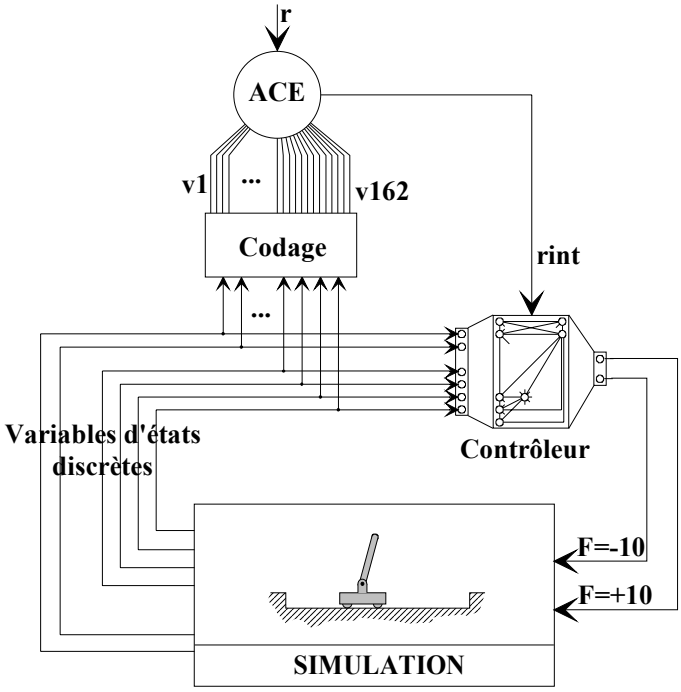


Figure VI-17 : *Architecture utilisée lors de l'apprentissage.*

Le simulateur fournit au contrôleur 15 variables binaires codant l'état du système. Le contrôleur est un réseau à mode ordonnancé qui génère, en sortie, l'action choisie. Les variables d'état sont par ailleurs codées sous forme de 162 « Boîtes » (v1 à v162) utilisées par une cellule ACE pour générer un signal de renforcement secondaire (rint) en prévision du renforcement primaire (r). Le renforcement secondaire est utilisé pour l'apprentissage du contrôleur.

Les premiers résultats obtenus avec cette architecture montrent, comme les trois tests précédents, que l'architecture connexionniste utilisée permet d'apprendre rapidement à contrôler le pendule pendant une à deux minutes (maximum 6 000 commandes). Cependant, dans le cas du pendule inverse, les sessions d'apprentissage sont suffisamment longues (jusqu'à 10 000 itérations par essais) pour que l'instabilité de l'apprentissage génère sporadiquement des comportements inadaptés et provoque ainsi la chute du pendule.

Mode ordonnancé et « SRV-Units »

Les instabilités apparaissant lors de l'apprentissage sont dues à un comportement similaire à la saturation des sorties observée par Albus pour le réseau CMAC [ALB 75]. C'est pourquoi nous avons utilisé, pour résoudre ce problème, une approche similaire au « Stochastic-

---

[84] Le principe de l'ACE est décrit en annexe A5, section 3.

[85] $x_0 \in [-0.012\ \text{m} + 0.012\ \text{m}]$, $\theta_0 \in [-0.3° + 0.3°]$, $\dot{x}_0 \in [-0.001\ \text{m.s}^{-1} + 0.001\ \text{m.s}^{-1}]$, $\dot{\theta}_0 \in [-0.03°.\text{s}^{-1} + 0.03°.\text{s}^{-1}]$

Real-Valued-Control » proposé par Vijaykumar Gullapalli [GUL 90] : l'état des neurones est soumis à un processus stochastique dépendant d'une estimation de la qualité de l'état courant et l'apprentissage est lui-même soumis à cette estimation (voir section IV.4.3.2).

Dans le cas du mode ordonnancé, le problème est légèrement différent puisque les cellules utilisées sont à état binaire (la fonction stochastique n'est donc pas directement réutilisable) et que le signal de renforcement est calculée par une cellule « ACE » spécialisée. Il n'est pas possible d'appliquer directement les principes développés par V. Gullapalli dans un réseau à mode ordonnancé.

En outre, la notion de ***path-coding*** doit être prise en compte : le comportement stochastique ne doit pas être appliqué directement à l'état des neurones mais à la valeur de leur déséquilibre. Ainsi, lorsque la variance est importante, le réseau est autorisé à explorer des chemins différents en modifiant les séquences d'activations dans le réseau (équations VI-26 à VI-28). Plutôt que de SRV-Units, nous parlerons donc ici de *chemins d'activation stochastiques*.

Le déséquilibre des neurones est donc calculé par une loi normale ($\Psi(\mu,\sigma)$) avec les paramètres suivants (moyenne et variance) :

$$\mu_{d_i}(t) = (1 - 2x_i(t))\left[\sum_{j=1}^{n} W_{ij} x_j(t) - \theta_i\right] \qquad \textbf{(VI-26)}$$

$$\sigma_d(t) = \frac{-p(t)}{5} \qquad \textbf{(VI-27)}$$

Soit :

$$d_i(t) = \Psi(\mu_{d_i}, \sigma_d) \qquad \textbf{(VI-28)}$$

Ce qui permet d'introduire un paramètre stochastique dans les séquences de mise à jour utilisées.

Nous avons aussi adapté la fonction d'apprentissage de V. Gullapalli (du moins sa composante issue du calcul de la moyenne et de la variance) tout en conservant le principe initial de l'apprentissage (MRT-Learning avec un terme modérateur), soit :

$$\Delta W_{ij} = \alpha . \hat{r} . \sigma_d (1 - |W_{ij}|) H(t_i - t_j) x_j \Delta x_i \qquad \textbf{(VI-29)}$$

Les poids sont donc modifiés suivant une règle identique à l'adaptation de la règle MRT-Learning utilisée jusqu'à maintenant mais avec un signal de renforcement global $\hat{R}$ combinant le renforcement secondaire et l'évaluation de l'état. Le renforcement secondaire étant lui-même fonction de l'évaluation de l'état (d'après les règles de fonctionnement de l'ACE), on a, pour le renforcement global :

$$\hat{R}(t) = \frac{-p(t)[r(t) + (\gamma p(t) - p(t-1))]}{5} \qquad \textbf{(VI-30)}$$

Le principe des *chemins d'activations stochastiques* a été appliqué à un réseau comptant 15 neurones d'entrée, 40 neurones d'afférence et 2 neurones de sortie. Les valeurs d'initialisation sont identiques à celles utilisées pour la reconnaissance des couleurs (entre 0.1 et 0.3 pour le fuseau de sortie, entre 0 et 0.2 pour les autres connexions). 10 tests ont été effectués, en considérant que la tâche est apprise lorsque le contrôleur parvient à maîtriser le pendule pendant 30 000 itérations (soit un comportement simulé de dix minutes)[86]. Au début de chaque essai le système est ré-initialisé autour de l'origine. Les sessions d'apprentissage ont été limitées à 2 000 essais. Sur les dix contrôleurs testés, cinq sont parvenus à contrôler le pendule pendant 30 000 itérations au moins et 3 sont parvenus à contrôler le pendule pendant plus de 12 000 itération (quatre minutes) (tableau VI-3). On notera que tous les contrôleurs qui sont parvenus à contrôler le pendule ont atteint ce but après moins de 300 essais : la moyenne[87] des temps d'apprentissage est de 181.4 essais (écart type : 54.08). La limite des 12 000 itérations est atteinte en moyenne en 250.6 essais (écart type : 136.03).

Contrôleur	1	2	3	4	5	6	7	8	9	10
Temps de contrôle max.	30 000	29 176	1 927	30 000	30 000	30 000	26 589	12 422	30 000	2 840
Date de contrôle max.	123	342	429	106	284	214	683	271	180	330

Tableau VI-3 : *temps d'apprentissage pour chacun des dix contrôleurs testés.*

Les résultats obtenus en utilisant des chemins d'activation stochastiques sont donc très bons en ce qui concerne le nombre d'essais nécessaires à l'apprentissage de la loi de contrôle. Par contre, cette technique ne permet pas de résoudre totalement le problème des instabilités lors de l'apprentissage ce qui n'est en rien étonnant puisque nous n'avons fait que réduire la modification des poids lorsque le réseau est dans une « bonne configuration » sans l'annuler pour autant. Il est d'ailleurs probable qu'en utilisant d'autres paramètres que ceux de Gullapalli pour le calcul de la variance des déséquilibres instantanés les résultats pourraient être encore améliorés. En outre, l'ACE ne permet pas de prolonger les sessions d'apprentissage suffisamment longtemps : après quelques centaines d'essais, les poids de l'ACE sont saturés et le signal de renforcement secondaire se dégrade. Etant donnés les temps nécessaires à l'apprentissage et la mobilisation des ressources informatiques que cela induit, il ne nous a pas été possible de tester efficacement d'autres configurations.

La comparaison des performances obtenues par rapport à d'autres modèles de réseaux est rendue difficile par la grande diversité des conditions de test. Le seul modèle avec lequel la comparaison soit possible est l'architecture ASE/ACE de Barto, Sutton et Anderson. D'après

---

[86] Le nombre de tests effectués est trop faible pour pouvoir effectuer une étude statistique poussée des résultats. En outre, la limitation à 30 000 itérations est inférieure à celle généralement utilisée (100 000 itérations). Ces limites sont dues aux temps de calcul induits par le grand nombre d'itérations : chaque essai demande entre 5 et 20 heures de calcul sur un PC 486/66. Il ne nous a pas été possible de mobiliser suffisamment de ressources pour permettre de multiplier les tests ou de les prolonger jusqu'à 100 000 itérations.

[87] sur les 5 apprentissages ayant convergé avant 30 000 itérations.

[BAR 83] l'ASE/ACE permet d'obtenir un contrôleur efficace en moins de 100 essais. Hormis le problème de l'instabilité à l'apprentissage (et la saturation de l'ACE), les temps de convergence affichés par le mode ordonnancé sont du même ordre de grandeur alors que, dans le cas présent, le contrôleur reçoit directement les valeurs des quatre variables d'état (l'utilisation des « Boîtes » dans l'architecture ASE/ACE permet de convertir le problème du pendule inverse en un problème de contrôle linéaire).

Comparés avec d'autres modèles d'apprentissage les résultats semblent particulièrement performants : le contrôleur de T. Langlois [LAN 92] est efficace après, en moyenne, 565 essais (chaque essai est stoppé après 6 000 itérations et la validation est faite sans apprentissage) et le taux de réseaux ayant convergé en 1 000 essais est similaire à celui obtenu avec le mode ordonnancé. Le contrôleur de C.W. Anderson converge après 6 000 essais [AND 89], S. Thurn [THU 91] ne parvient à contrôler le pendule qu'une trentaine de secondes (1 755 itérations)... Cependant ces modèles doivent apprendre parallèlement une fonction de contrôle non-linéaire et une fonction d'apprentissage non-linéaire (alors que nous avons utilisé l'ACE qui apprend une fonction linéaire). En outre, ils utilisent tous des variables d'états à valeurs réelles... Autant de différences qui rendent caduques d'éventuelles comparaisons. En outre, l'instabilité de l'apprentissage reste un problème important pour le mode ordonnancé.

Etude du contrôleur obtenu

D'après [GEV 93], l'apprentissage de la commande du pendule dans les conditions optimales n'est pas suffisant pour estimer la qualité d'un algorithme et on doit aussi tester la qualité du contrôleur obtenu. Les tests complémentaires suivants ont été effectués sur le premier contrôleur obtenu :

- Contrôle continu sur 180 000 itérations[88] ce qui permet de montrer que le réseau est bien capable de contrôler le pendule pendant un temps considéré comme « infini » (figures VI-18 et VI-19).

- Contrôle du pendule à partir d'une position initiale quelconque (vitesse nulle) afin de déterminer la zone de l'espace d'état depuis laquelle le contrôleur parvient à ramener le pendule dans le régime de fonctionnement normal[89]. Ce type de test est proposé par Schlomo Geva [GEV 93] pour tester la robustesse du contrôleur. [GEV 93] propose d'utiliser des vitesses initiales non nulles mais dans un espace d'état plus large (autorisant le pendule à basculer jusqu'à 45°). Les limites que nous avons utilisées ne permettent pas au système de se stabiliser s'il est initialisé avec une forte vitesse angulaire. Le réseau parvient à contrôler le pendule quel que soit l'angle initial (entre +0.2 et -0.2 radian) sous réserve que la position du chariot reste dans la zone de fonctionnement normal représentée par la zone en grisé sur la figure VI-20. En dehors

---

[88] ces 180 000 itérations représentent, une heure de temps de contrôle simulé. Sur un 486/66 ce calcul demande environ 4 heures sous Matlab. Le contrôleur, sous réserve d'être codé dans un langage compilé (type « C ») serait donc capable de contrôler le pendule est temps réel.

[89] Pour limiter les temps de calcul nous avons considéré que le contrôleur parvient à maîtriser le pendule lorsqu'après 1 000 itérations le système n'est pas sorti de l'espace d'état. Nous avons choisi cette solution après avoir constaté, sur une série de test prolongés jusqu'à 10 000 itérations, que toutes les erreurs ont lieu dans les 500 premières itérations. De fait, sur les 1 000 points essayés, nous avons observé 367 erreurs dont 1% entre 900 et 1 000 et 3% entre 800 et 900.

de cette limite, les oscillations générées sur le pendule risque de le faire sortir de l'espace d'état. La chute est alors fonction de l'amplitude de l'oscillation, c'est-à-dire du temps que dure la correction. Pour obtenir des contrôleurs robustes en x et en θ, il est aurait été nécessaire de modifier la phase d'apprentissage, soit en perturbant le système, soit en le réinitialisant dans une position aléatoire.

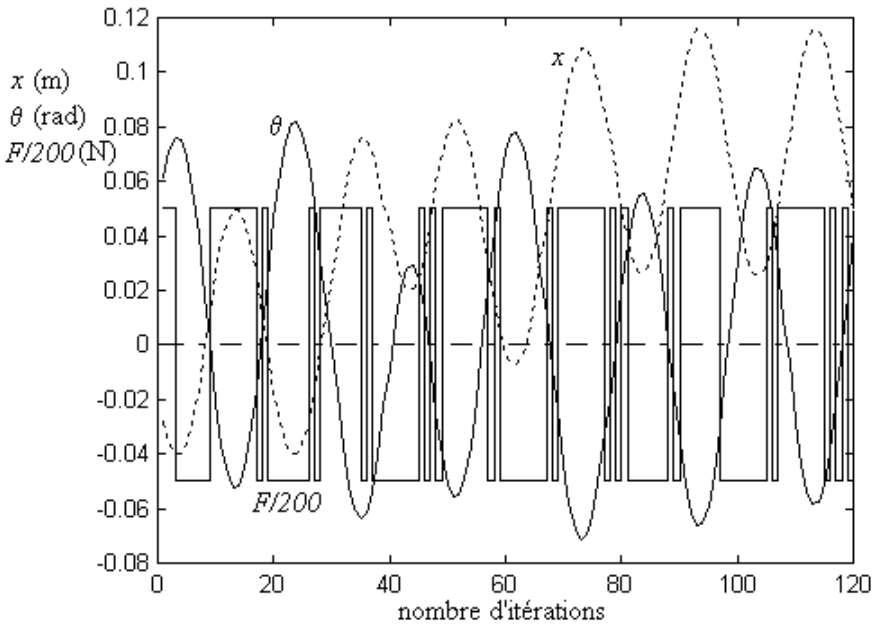


Figure VI-18 : *Comportement du contrôleur sur les 120 premières itérations.*

La courbe en pointillés représente la variable x, la courbe pleine la variable θ et les créneaux représentent le 1/200ème de la force appliquée.
En régime normal (c'est-à-dire quand x reste faible), le réseau développe deux types de comportement : de longs créneaux (6 ou 7 itérations) lui permettent de ramener le pendule vers la verticale tandis que des créneaux plus courts (une itération) permettent d'amortir le pendule quand la vitesse angulaire est trop importante.

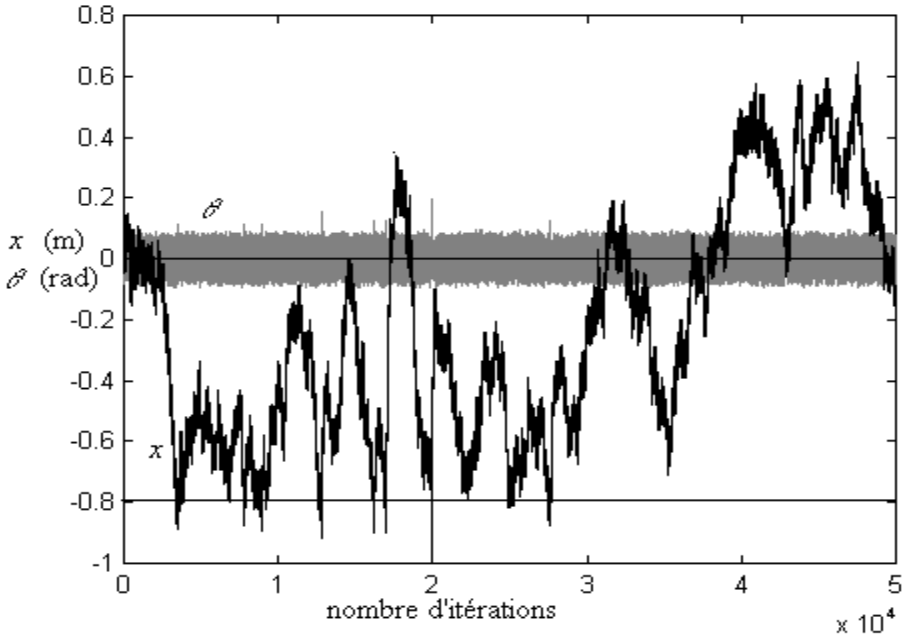


Figure VI-19 : *Evolution de x (noir) et de θ (gris) pendant 50 000 itérations.*

Tant que le pendule est en régime normal (-0.8<x<0.8), le réseau contrôle le système en fonction de l'angle du pendule mais lorsque x sort de ces valeurs, le pendule sort de son régime de fonctionnement normal et le contrôleur corrige brutalement la déviation de qui provoque des oscillations en x et en θ (par exemple après 400, 800, 900 itérations).

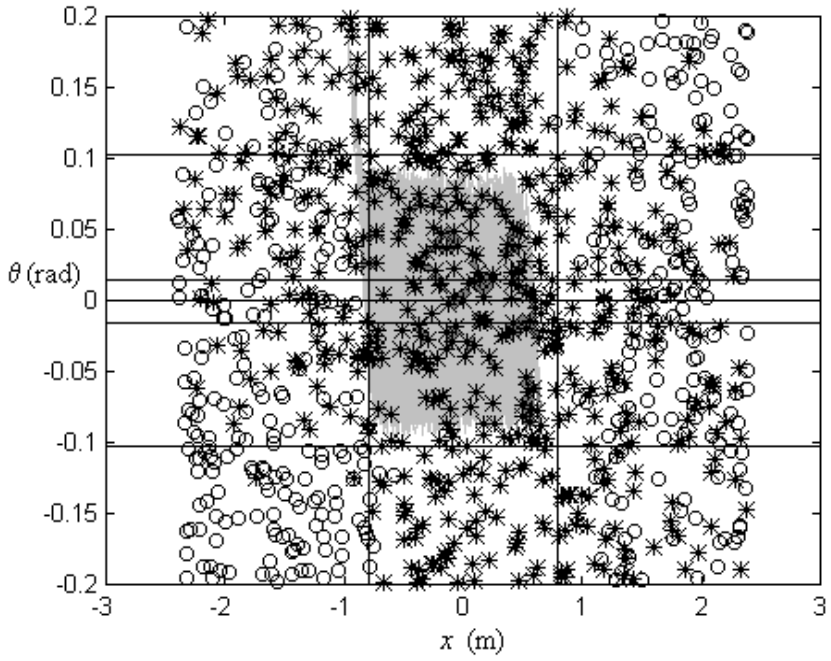


Figure VI-20 : *Comportement du système suivant les conditions initiales. La zone grisée représente les points parcourus sur les 180 000 itérations testées précédemment.*

***** *: pas d'erreur avant 1 000 itérations,* ***o*** *: erreur avant 1 000 itérations.*

Représentation des connaissances

Comme pour les tests précédents, nous avons étudié la représentation des informations stockées dans le réseau. Pour cela nous avons utilisé la même approche que pour l'apprentissage de séquences, c'est-à-dire une visualisation des corrélations d'activité entre les différents neurones pour chacun des états d'entrée. Cependant la complexité du problème est ici largement supérieure puisque le réseau compte 15 neurones d'entrée et 162 configurations possibles. Les résultats obtenus lors de cette étude sont détaillés en annexe A6, section 4.

Comme pour l'ensemble des tests effectués jusqu'à maintenant, on constate que le réseau s'est auto-dimensionné : seuls 26 neurones sur 40 sont utilisés pour contrôler le pendule inverse. Les résultats les plus parlants sont la découverte, parmi les cellules d'afférence, de neurones caractéristiques de certains régimes de fonctionnement : on observe ainsi des cellules dont l'activité est liée au risque de chute, dans un sens ou dans l'autre, au risque de sortie de la piste ou, au contraire, à la « sécurité » (mode de fonctionnement normal : chariot centré et pendule proche de la verticale). Tous ces neurones spécifiques interagissent, à la manière d'une carte cognitive, pour contrôler le pendule.

L'analyse manuelle des réseaux obtenus après apprentissage montre donc que le réseau développe bien une représentation semi-distribuée des connaissances internes puisque des groupes fonctionnels regroupant un petit nombre de neurones émergent lors de l'apprentissage. Cependant, conformément aux prévisions, ces groupes sont diffus et il est difficile de les mettre en évidence par la seule observation « manuelle » du réseau. Le développement de procédures automatiques de détection des corrélations temporelles entre neurones d'entrée, neurones cachés et neurones de sortie (par exemple basées sur l'analyse delta [FEC 93]) est probablement le seul moyen fiable qui permette d'affirmer l'émergence d'assemblées de neurones dans le réseau.

L'émergence de groupes fonctionnels permet l'apprentissage par partie. Dans le cas du pendule inverse, cette caractéristique permet d'accélérer l'apprentissage : le réseau développe des comportements pour le contrôle du pendule puis, lorsque ces comportements sont acquis, il parvient à maîtriser l'angle pendant un temps suffisant pour « permettre » au chariot de se décaler et d'atteindre le bord de la piste. Le réseau apprend alors de nouveaux comportements, spécifiques, permettant de corriger la dérive du chariot. Grâce au mode ordonnancé, cette deuxième phase s'effectue sans remettre totalement en cause la première ce qui n'est pas le cas avec des réseaux monolithiques comme les réseaux en couches (d'où l'importance des temps d'apprentissage affichés par [AND 89][90]).

Conclusion

L'utilisation, pour tester les applications du « mode ordonnancé » au neurocontrôle, d'un problème étalon comme le pendule inverse nous a permis de progresser plus rapidement, tant pour l'amélioration du contrôleur que pour son analyse. En effet, l'importance de la littérature portant sur ce système nous a permis de mieux identifier les causes des instabilités apparaissant

[90] Cette analyse est corroborée par le comportement du système présenté par C.W. Anderson : si le système est systématiquement initialisé en 0,0,0,0, le réseau ne parvient pas à maîtriser la dérive en *x*. L'apprentissage est donc effectué en initialisant le système aléatoirement dans l'espace d'état ce qui revient à organiser l'apprentissage de façon à présenter « simultanément » l'ensemble des situations possibles.

lors de l'apprentissage et d'y remédier en adaptant au besoin les solutions proposées pour les réseaux « classiques ».

Les résultats de ces tests sont prometteurs puisque le mode ordonnancé parvient à apprendre rapidement à contrôler le pendule inverse : le nombre d'essais nécessaires est comparable à celui obtenu avec d'autres architectures. En outre, l'analyse des matrices de poids obtenues nous montre que le réseau a bien développé, en cours d'apprentissage, une représentation semi-distribuée des connaissances. Par contre, malgré les améliorations apportées par l'utilisation des « chemins d'activation stochastiques », l'apprentissage reste instable et, dans la moitié des cas, le réseau n'a pas atteint les 30 000 itérations fixées comme limite à la phase d'apprentissage. Cependant il semble que ce comportement ne soit pas exclusivement le fait du mode ordonnancé mais aussi de la cellule ACE dont les poids sont rapidement saturés et qui, après quelques centaines d'itérations, ne fournit plus au contrôleur qu'un signal de renforcement dégradé. De plus, compte tenu des temps de calcul nécessaires à l'aboutissement d'une série de tests (plus de cinq jours de calcul en phase d'apprentissage sur un 486DX/66[91]), il ne nous a pas été possible d'optimiser les paramètres du réseau. Il est probable que des réglages appropriés, pour le réseau et pour l'ACE, voire l'utilisation d'un autre modèle de réseau « critique » (par exemple un réseau en couches), permettraient d'améliorer les performances de l'apprentissage, en particulier quant au nombre de réseaux convergents.

VI.4.7. Analyse

Les quatre tests que nous venons de proposer nous ont permis de raffiner progressivement l'architecture du mode ordonnancé afin de résoudre les différents problèmes que nous avons rencontrés. Parmi ceux-ci, les plus importants sont les états non-répondants et l'instabilité de l'apprentissage qui nous ont conduits à modifier l'algorithme initialement proposé (utilisation de chemins d'activation stochastiques) et à préciser les domaines de viabilité des différents paramètres (valeurs d'initialisation des poids). Ces raffinements progressifs se sont traduits par une augmentation constante de la complexité des tests proposés avec, dans tous les cas, de bonnes performances lors de l'apprentissage. Au terme de cette étude, il nous est donc possible de dégager les points forts du modèle connexionniste proposé :

- **Vitesse d'apprentissage :** le mode ordonnancé permet d'apprendre très rapidement des fonctions entrée/sortie complexes car la taille du réseau n'a pas d'influence directe sur la longueur des chemins d'activation.
- **Auto-dimensionnement :** le réseau peut être sur-dimensionné. Seul un sous-ensemble des neurones cachés sera recruté lors de l'apprentissage.
- **Simplicité du paramétrage :** à l'exception de l'apprentissage du OU EXclusif pour lequel nous n'avions pas une connaissance approfondie du mode de fonctionnement du réseau, les paramètres utilisés n'ont pratiquement pas été modifiés pour passer d'une application à une autre. En outre, les paramètres déterminants sont peu nombreux : il s'agit essentiellement des valeurs d'initialisation et du facteur d'apprentissage. L'utilisation des chemins d'activation stochastiques augmente cependant le nombre de paramètres.

---

[91] Pour un total de 1 789 287 itérations, soit près de dix heures de temps simulé. Le temps d'apprentissage est donc dû avant tout au nombre d'essais effectués. Compte tenu des conditions dans lesquelles ces tests ont été effectués (utilisation d'un langage interprété), le rapport temps simulé/temps réel obtenu (environ un douzième) permet d'envisager un apprentissage en temps réel si le réseau est codé dans un langage compilé plus performant (type « C »).

- **Possibilité d'utiliser simultanément l'apprentissage par renforcement et les « sorties forcées »**
- **Possibilité d'insérer des connexions réflexes** dans le réseau (par exemple pour favoriser certains comportements ou pour mettre en oeuvre la règle « Winner-Takes-All »)
- **Développement d'une représentation semi-distribuée des connaissances.** En liaison avec l'auto-dimensionnement du réseau, l'étude des matrices de poids montre que le réseau concentre l'information sur un petit nombre de poids. L'étude des séquences d'activation confirme cette information : les neurones utilisés par le réseau sont associés à des configurations d'entrée/sortie spécifiques correspondant souvent à des règles symboliques identifiables.
- **La capacité à apprendre par partie**, illustrée par l'apprentissage de séquences et, indirectement, par le pendule inverse (voir ci-dessous), est une conséquence de la représentation semi-distribuée des connaissances : l'ajout de nouvelles règles comportementales n'a qu'une influence limitée sur les règles déjà apprises. En raison des problèmes d'oubli catastrophique, l'apprentissage par partie est impossible dans les réseaux « classiques » distribuant totalement les connaissances dans la matrice de poids synaptiques.

Cependant, les réseaux utilisés présentent aussi un certain nombre de points faibles qu'il est nécessaire d'éliminer avant de passer à des applications plus complexes et, en particulier, à l'intégration de l'algorithme dans des agents réactifs simulés ou réels :

- Le principal problème est, nous l'avons vu, **l'instabilité de l'apprentissage**. Même s'il est souvent possible d'arrêter l'apprentissage lorsque des critères d'apprentissage sont atteints, ce n'est pas souhaitable pour des agents réactifs qui doivent apprendre « on-line » pour pouvoir s'adapter immédiatement à tout changement de leur environnement. Ce problème est caractéristique des apprentissages hebbiens qui ne permettent pas la convergence en loi.
- **L'exploration de l'espace des commandes**. L'apprentissage par renforcement ne permet pas au réseau de créer de nouveaux chemins d'activation. Deux solutions ont été proposées, les chemins d'activation stochastiques et la sortie forcée. En outre, l'initialisation des poids permet d'assurer l'existence potentielle de l'ensemble des chemins possibles.
- **Les états non-répondants.** Aucune des procédures utilisées pour l'élimination des états non-répondants ne s'est avérée réellement efficace. Seule l'initialisation des poids, qui permet d'éviter de créer initialement des états non-répondants, permet pour l'instant de résoudre ce problème.
- **La dépendance par rapport aux conditions initiales**. Ce problème est directement lié aux deux précédents : si, lors de l'initialisation, le réseau généré présente des états non-répondants ou des neurones inactivables (neurones auxquels n'aboutissent que les connexions inhibitrices), l'algorithme d'apprentissage ne permet pas de remettre en cause efficacement ces caractéristiques et l'apprentissage de la fonction entrée/sortie peut être rendu impossible. Il en va de même si, au cours de l'apprentissage, le réseau génère de telles configurations.

Malgré ces problèmes, essentiellement dus à l'inadéquation des algorithmes d'apprentissage utilisés, le mode ordonnancé apparaît comme un réseau performant. De plus, il répond bien à son cahier des charges : l'auto-dimensionnement et la représentation semi-distribuée des connaissances permettent de conserver une réserve de neurones pour des apprentissages ultérieurs et de modifier une partie du comportement du réseau sans perturber le fonctionnement de l'ensemble.

VI.4.8. Vers d'autres algorithmes...

Le principe du mode ordonnancé est librement inspiré des recherches en neurobiologie qui montrent l'importance du facteur temporel dans les réseaux de neurones et, plus particulièrement, pour la création et l'identification des assemblées de neurones. Cependant, le mode ordonnancé est avant tout un modèle connexionniste *artificiel* intégrant un facteur temporel local, plus complexe que ceux habituellement utilisés par les réseaux de neurones formels. Les différents tests effectués montrent que ce premier pas vers l'intégration du temps dans les réseaux de neurones permet déjà d'obtenir des comportements d'apprentissage performants. Ceux-ci sont cependant limités par l'instabilité des algorithmes de modification des poids synaptiques. Dans cette section, nous proposons quelques extensions possibles du mode ordonnancé destinées, soit à améliorer les performances de l'apprentissage, soit à augmenter encore l'intégration du facteur temps dans le réseau.

VI.4.8.1. Vers d'autres algorithmes d'apprentissage

Nous avons vu, lors de l'étude des réseaux ayant appris le OU EXclusif et avant de présenter le principe des chemins d'activation stochastiques, que les instabilités de l'apprentissage sont essentiellement dues à l'équation utilisée pour l'apprentissage hebbien. En effet, celle-ci génère une modification « standard » des poids synaptiques qui est ensuite appliquée systématiquement, positivement ou négativement, à toutes les connexions reliant des neurones dont l'activité est corrélée. Or, cette modification des poids ne tient pas compte du nombre de connexions activatrices ou inhibitrices aboutissant au neurone, ni de leurs pondérations respectives.

L'apprentissage MRT-Learning initialement proposé par J.Y. Cheung intègre cette composante en introduisant, dans l'équation d'apprentissage, un terme égal à la valeur du poids modifié (section VI.3.4.1, équation VI-19). Cependant cette solution entraîne une croissance exponentielle des poids synaptiques, ce qui n'est pas souhaitable.

Le « Drive-Reinforcement-Learning » de H. Klopf (section VI.3.4.1, équation VI-18) utilise un terme de « trace » destiné à évaluer la corrélation entre deux neurones par rapport au temps écoulé entre leurs activations respectives. Nous avons testé ce principe sur le mode ordonnancé (pendule inverse) sans constater d'amélioration par rapport à l'algorithme initial. En fait, il est probable que le principe de la « trace » ne soit pas adapté au mode ordonnancé car il ne permet pas au réseau de développer plusieurs chemins en parallèle (dans ce cas les connexions transversales entre les chemins risquent d'être fortement modifiées).

Pour résoudre le problème des instabilités apparaissant lors de l'apprentissage, nous proposons d'utiliser dans l'algorithme d'apprentissage un terme fonction de la part prise par la connexion pour la génération du déséquilibre global du neurone. Ainsi les connexions générant le plus de déséquilibres vont être plus fortement modifiées, ce qui revient à favoriser l'apprentissage des chemins effectivement utilisés pour la propagation de l'influx nerveux sans pour autant provoquer « l'emballement » des connexions synaptiques. Plusieurs variantes sont possibles : il est possible de prendre en compte le déséquilibre absolu (par rapport à zéro) ou le déséquilibre relatif (par rapport au déséquilibre immédiatement inférieur). Une troisième solution consiste à prendre en compte les variations de déséquilibre provoquées par l'activation ou la désactivation d'un neurone. Cependant cette famille d'algorithmes n'a pas encore fait l'objet de tests et doit être validée par une expérimentation pratique.

La recherche d'un algorithme d'apprentissage pour le mode ordonnancé peut aussi être guidé par les progrès de la neurobiologie. En effet, comme nous l'avons souligné section VI.3.4.1, l'apprentissage hebbien n'intègre pas, originellement, de mécanismes de décroissance des poids synaptiques. Pour notre part, nous avons considéré la décroissance des poids comme étant liée à un simple paramètre (le renforcement) et au sens de la corrélation du changement d'activité. Or rien, dans les développements de la neurobiologie, n'indique que le mécanisme de décroissance des poids soit identique au mécanisme de croissance synaptique. Au contraire, il semble que ces mécanismes diffèrent complètement de par leur nature (facilitation/habituation), de par leurs seuils (des activations positives peuvent disparaître - malgré l'existence de corrélations d'activité - si un seuil de stimulation n'est pas atteint) et de par les mécanismes chimiques et les neurotransmetteurs utilisés (GABA/Sérotonine). Il n'entre pas dans notre propos de détailler ici les principes d'apprentissage des neurones biologiques mais il est probablement possible de retirer des informations pertinentes par l'étude des phénomènes d'habituation et de facilitation mis en jeu dans les réseaux de neurones biologiques.

VI.4.8.2. Toujours un peu plus loin...

Sous ce titre, nous voulons proposer des extensions au mode ordonnancé qui peuvent servir de base à des développements futurs.

La première extension possible est le passage des neurones binaires aux neurones réels. Nous avons déjà étudié cette possibilité (voir annexe A6) et l'extension des algorithmes de mise à jour ne pose pas de problèmes majeurs. Par contre, les mécanismes d'apprentissage utilisés ne sont pas du tout adaptés à l'utilisation de neurones à état réel et tant que des algorithmes permettant la convergence en loi des connexions synaptiques ne seront pas disponibles, il paraît prématuré d'étendre le mode ordonnancé aux neurones réels.

Une deuxième extension possible est le passage au modèle de neurones dit « integrate and fire », c'est-à-dire à l'utilisation d'une fonction de transfert plus complexe, intégrant dans le temps les stimulations reçues (avec un coefficient de fuite) et déchargeant lorsqu'un seuil d'intégration est dépassé. Ces modèles de neurones permettent de copier le comportement oscillatoire des neurones réels et donc de générer des rythmes d'activité et des synchronisations dans le réseau (voir section VI.2.2). Cependant ces modèles deviennent vite très complexes et le nombre de paramètres à ajuster devient très important ce qui complique encore l'apprentissage (voir, par exemple, le réseau de Randall Beer, section V.2.1.1). En outre, la gestion du temps dans ces modèles de réseaux dépasse largement, en complexité, le simple principe de la mise à jour ordonnancée. En fait on peut considérer que le mode ordonnancé est une solution intermédiaire entre les modes de mise à jour classiques (synchrone/asynchrone) et les modèles « integrate and fire ». Les tests montrent d'ailleurs que cet algorithme possède certaines des qualités propres aux modèles oscillatoires (émergence d'assemblées de neurones) tout en conservant la simplicité de mise en oeuvre des modèles classiques.

La troisième et dernière extension que nous proposerons ici est une modification de l'architecture de critique adaptative. En effet, la critique adaptative peut être considérée comme un modèle de réseau modulaire dans lequel un module calcule le signal de renforcement interne tandis que l'autre recherche la fonction entrée/sortie permettant de maximiser ce signal. Cette architecture est typiquement construite suivant une approche descendante (définition manuelle des modules puis intégration) mais l'apprentissage est réalisé après intégration. Les temps d'apprentissage nécessaires pour l'apprentissage d'une loi de contrôle par la critique adaptative montrent d'ailleurs combien l'apprentissage est difficile avec cette procédure(voir section VI.4.6).

Pour accélérer l'apprentissage, nous proposons de construire une architecture de type critique adaptative au moyen de l'approche ascendante (un réseau unique se spécialisant en une partie critique et une partie contrôle) et du mode ordonnancé. Cependant les difficultés pratiques d'une telle approche sont très importantes : d'une part les signaux de renforcement nécessaires à l'apprentissage de la critique sont opposés à ceux nécessaires à l'apprentissage du contrôle, d'autre part la création d'une boucle de feedback positif d'un réseau sur lui-même via l'algorithme d'apprentissage est difficile à maîtriser. Pour l'instant, toutes les tentatives pour mettre en place ce type d'architecture ont avorté en raison de saturations apparaissant soit au niveau du signal de renforcement, soit au niveau des poids du réseau. Cependant une telle architecture présente un intérêt indéniable (elle évite la redondance d'informations entre le réseau de critique et le réseau de contrôle) et constitue un sujet de recherche largement ouvert.

VI.5. Perspectives : vers l'intégration

Au cours de ce chapitre, nous avons proposé une architecture connexionniste pour le contrôle réactif, en accord avec les spécifications développées dans les chapitres II à V. Par rapport aux modèles connexionnistes classiques utilisés en neurocontrôle (c'est-à-dire, essentiellement, la rétropropagation du gradient dans les réseaux en couches), le cahier des charges du modèle connexionniste comprenait quatre points clefs (le cinquième point, l'absence de modèle de l'environnement, concerne plus l'organisation générale de l'apprentissage que le réseau lui-même. Dans le cadre de ce travail, nous avons choisi un modèle d'apprentissage par critique adaptative, ce qui répond à ce point particulier) :

- Un apprentissage par renforcement,
- Un apprentissage en ligne,
- Modularité,
- Architecture sensori-motrice.

En outre, les chapitres antérieurs nous ont montré que le contrôleur connexionniste obtenu devra être capable d'apprendre aussi bien des connexions réflexes élémentaires que des comportements spécifiques basés sur des séquences d'actions.

L'apprentissage par renforcement et l'apprentissage en ligne, tout comme l'architecture sensori-motrice, sont des contraintes dont nous pouvons tenir compte lors de la conception du contrôleur. Par contre, le chapitre V nous a montré que les modèles connexionnistes modulaires actuellement disponibles sont incompatibles avec les besoins du contrôle sensori-moteur. Après avoir constaté, au terme du chapitre précédent, que les systèmes biologiques présentent une architecture modulaire similaire à celle que nous souhaitons mettre en place, nous avons débuté le présent chapitre par une étude rapide des modèles neurobiologiques afin de dégager des règles générales d'organisation permettant la spécialisation modulaire lors de l'apprentissage. Cette étude nous a permis d'ébaucher un modèle de réseau capable de se spécialiser grâce à une loi de mise à jour spécifique. Enfin, nous avons proposé une série de quatre tests permettant à la fois d'illustrer le fonctionnement du réseau et d'en préciser l'architecture.

Au terme de ce chapitre, la question se pose donc de savoir si le réseau développé répond bien au cahier des charges défini précédemment. Concernant l'apprentissage par renforcement et l'apprentissage on-line, la réponse ne fait aucun doute puisque le réseau a été défini comme tel. L'organisation en une architecture sensori-motrice dépend de la façon dont le réseau échange des informations avec son environnement. La quatrième phase de test (le pendule inverse) montre clairement qu'un réseau utilisant le mode ordonnancé peut communiquer directement avec son

environnement sans passer par une interface symbolique. De même, la diversité des tests proposés (dont l'apprentissage de séquences) nous montre qu'un même modèle de réseau est capable d'apprendre des fonctions sensori-motrices complexes et des séquences de comportements.

La question de la modularité est plus complexe et il est difficile d'affirmer que le réseau obtenu est effectivement organisé en modules fonctionnels. Cependant, les études des réseaux obtenus après apprentissage fournissent nombre d'indices (auto-dimensionnement, neurones réalisant des règles comportementales, neurones de mémorisation temporaire, neurones « seuil », ...) qui nous permettent d'affirmer que le mode ordonnancé permet bien de générer, par l'apprentissage, une représentation semi-distribuée de la loi d'entrée/sortie.

Le mode ordonnancé nous a donc permis de répondre favorablement au cahier des charges d'un contrôleur « Neuro-Réactif ». Pour autant, et bien que ce mémoire sanctionne la fin d'un cycle de recherche, le domaine reste largement ouvert : nous avons vu que les réseaux proposés présentent des faiblesses qu'il faut encore éliminer. De plus, il est maintenant nécessaire de franchir une étape supplémentaire et d'intégrer ce modèle de réseau dans des systèmes multi-agents, d'abord sur la base de l'application de transport présentée chapitre II, puis au niveau du pilotage global de l'atelier. En effet, cette application nous avait permis de montrer l'intérêt du pilotage réactif pour une conduite robuste et décentralisée mais le passage à un niveau stratégique « multi-agents » reste soumis à l'intégration du mécanisme d'apprentissage. Celui-ci doit en effet permettre aux agents d'acquérir des comportements spécifiques en fonction du contexte de production (variabilité des tâches, charge des machines, enchaînement d'actions, taille des obstacles, nombre de robots, ...). En d'autres termes, l'intégration doit faire évoluer l'organisation distribuée réactive vers une organisation coopérative au sein de laquelle les agents s'adaptent à leur environnement. Or, si les tests proposés montrent bien l'adéquation entre le réseau de neurones et les pré-requis architecturaux (modularité, apprentissage par partie, ...), le comportement d'émergence propre à l'apprentissage ne peut être observé (et validé) que dans le cadre du système global. La validation de l'architecture permettra alors de passer à la phase finale, c'est-à-dire à la création d'une architecture de contrôle décentralisée en atelier flexible et son intégration dans le projet régional « CORINE » (COnduite Robuste et Intelligente en Entreprise, région Rhône-Alpes) pour accroître la robustesse des systèmes de production.

En ce qui concerne la première étape, c'est-à-dire le contrôle du système de transport, l'intégration se déroulera en trois phases. Dans un premier temps, la simulation sera utilisée pour permettre l'acquisition de comportements génériques (typiquement des applications de transport non-conflictuelles, c'est-à-dire mono-robot) qui garantissent la portabilité du système vers un environnement de production quelconque. Les robots seront alors soumis (toujours en simulation) à un environnement de production. Après une phase d'adaptation « on-line » (ce qui correspond, en atelier, à une phase productive mais non optimale), les chariots mobiles devront pouvoir éviter (ou, au moins, gérer) les conflits en se spécialisant et en limitant les passages aux points de saturation physique (le centre de l'atelier dans l'exemple présenté chapitre II, voir section II.3.5.3). L'architecture modulaire propre au réseau de neurones autorisant la cohabitation des comportements génériques et spécifiques avec un minimum de dégradation, la flexibilité et la réactivité du système de transport sont garanties par la disponibilité permanente des comportements génériques. Les comportements génériques développés en simulation pourront alors être ré-utilisés pour la transition vers le système physique.

CHAPITRE VII :

CONCLUSION

Conclusion

Le but premier de ce travail était de développer des architectures de pilotage flexibles et réactives pour les systèmes de production. La flexibilité d'évolution, en particulier, apparaît comme un facteur déterminant permettant de diminuer le temps de mise sur le marché d'un produit et donc d'accroître la compétitivité de l'entreprise.

L'étude des différentes architectures utilisées pour le pilotage des ateliers de production nous a montré la rigidité structurelle qu'entraîne la centralisation de la commande. C'est pourquoi nous avons proposé une architecture totalement décentralisée dans laquelle chaque ressource de l'atelier dispose d'une autonomie totale et ne réagit que par rapport à ses perceptions. L'application de ces principes à un sous-système de l'atelier (le système de transport) nous a permis d'illustrer les capacités de ce type d'architecture.

La décentralisation des processus de décision suppose que chaque ressource dispose d'un contrôleur local capable, d'une part de répondre en temps réel à des stimulations complexes (contrôle sensori-moteur), d'autre part d'améliorer progressivement ses comportements propres, éventuellement en développant des comportements spécifiques. C'est à cette condition, en effet, que l'atelier va pouvoir répondre immédiatement à une demande de production (grâce aux comportements réflexes), puis, si cette demande persiste, développer des comportements spécifiques permettant d'augmenter la productivité. En outre, une telle organisation permet de réagir immédiatement aux aléas apparaissant lors d'un cycle de production (donc après apprentissage), grâce à un « mode dégradé » composé des règles sensori-motrices élémentaires.

Après avoir développé l'architecture générale, nous avons proposé, pour la réalisation des contrôleurs locaux, l'utilisation conjointe de deux approches du contrôle de processus :

- le contrôle réactif qui permet l'accès à l'autonomie décisionnelle grâce à des modèles comportementaux basés sur un ensemble de réflexes et sur leur adéquation avec l'environnement du système,

- les algorithmes neuromimétiques qui offrent des potentialités d'apprentissage et d'adaptation. En outre, les modèles connexionnistes ont déjà été très largement appliqués au contrôle de processus.

Après une étude bibliographique du contrôle réactif et des algorithmes neuromimétiques, nous avons cherché à réunir ces deux approches au sein d'une architecture permettant l'apprentissage de comportements réflexes et de séquences comportementales stéréotypées. Or, il apparaît que les différents modèles de réseaux de neurones proposés ne sont pas adaptés pour ce type d'architecture : leur structure monolithique est, en effet, incompatible avec l'architecture modulaire d'un contrôleur réactif. Pour pallier à cette insuffisance, nous avons effectué une étude bibliographique des différents modèles de réseaux modulaires. Malheureusement, la

plupart d'entre eux sont basés sur une définition a priori de l'architecture (ou d'une partie de l'architecture), ce qui est contraire au principe d'apprentissage recherché (apprentissage en ligne et par renforcement). Nous avons donc proposé une architecture inspirée des modèles neurobiologiques « d'assemblées de neurones ».

L'avantage d'une telle architecture est double : d'une part les assemblées de neurones sont des entités diffuses composées de plusieurs cellules d'un même réseau (il n'est donc pas nécessaire de décomposer le réseau en modules clairement identifiés, la modularité est ici caractérisée par la fonction plus que par l'appartenance à un groupe), d'autre part cette structure permet d'établir un lien entre l'approche neuromimétique basée sur les microstructures cervicales et le contrôle réactif largement inspiré des macrostructures du cerveau. Par contre, cette approche n'est utilisable que si nous disposons d'un algorithme permettant de faire émerger des assemblées de neurones dans un réseau au cours de l'apprentissage.

Avant de proposer un algorithme propre à engendrer l'apparition d'assemblées de neurones dans un réseau, nous avons étudié plus précisément les modèles neurobiologiques disponibles. Cette étude nous a permis de mettre en évidence les relations étroites existant entre les assemblées de neurones et le facteur temporel. C'est pourquoi, en nous inspirant librement de la neurobiologie, nous avons proposé un modèle de réseau de neurones basé sur un nouveau mode de gestion du temps, c'est-à-dire, dans un réseau de neurones, sur un nouveau mode de mise à jour : le mode ordonnancé. Après avoir défini l'ensemble de l'architecture de ce modèle, nous avons l'avons testé sur différents problèmes dont l'apprentissage de séquences booléennes et un « Benchmark » : l'apprentissage du contrôle d'un pendule inverse. Les résultats obtenus nous montrent que le réseau, outre qu'il présente de bonnes facultés d'apprentissage, développe bien, au cours de l'apprentissage, une structure interne qualifiée de semi-distribuée. En effet, les cellules cachées se spécialisent en cours d'apprentissage et forment des groupes fonctionnels diffus similaires à des assemblées de neurones. En outre, cette structure permet au réseau d'apprendre par partie une fonction d'entrée/sortie en rajoutant, sur la base des comportements déjà appris, de nouvelles règles stockées sur des neurones inutilisés jusqu'alors. Ce comportement est analogue, dans son principe, à l'architecture « Subsumption » qui nous avait servi de modèle de contrôleur réactif.

Notre étude nous a conduits, à partir d'une problématique « productique », à proposer un modèle de contrôleur réactif. Nous avons donc suivi une méthodologie peu usitée en intelligence artificielle puisque nous avons travaillé par raffinements successifs, à partir d'un problème d'ordre pratique, pour aboutir à une architecture connexionniste adaptée. A chaque étape de la conception nous avons proposé des solutions spécifiques en fonctions des contraintes fixées à l'étape précédente. Cette approche permet de développer une architecture parfaitement adaptée au problème posé mais elle nécessite une attention constante pour assurer la cohérence de l'ensemble. Pour cela nous avons été guidés par les modèles biologiques. Le mode ordonnancé réuni en effet, par sa conception, deux approches inspirées des modèles biologiques mais issues de niveaux d'observation différents (éthologie/neurobiologie).

Parvenus au terme de ce mémoire, nous ne pouvons que souhaiter qu'il ne constitue pas pour autant le terme de ce projet. En effet, l'ensemble de l'approche proposée se révèle riche en opportunités de recherches. Ainsi, l'intégration du contrôleur dans un agent réactif, puis de celui-ci dans une « société multi-agents », permettra aux systèmes de production (ou à tout autre système construit sur le même principe) de s'auto-organiser en fonction des consignes qui lui seront fixées, ce qui offre un important potentiel de réactivité et de flexibilité. C'est pourquoi la phase d'intégration, dans un premier temps sur le sous-système de transport, puis étendue à l'ensemble du système de production, constitue la prochaine étape de ce projet. Outre l'intérêt

pratique que revêt cette approche, elle présente un intérêt théorique indéniable dans divers domaines :

- L'utilisation d'agents doués d'apprentissage au sein d'une société multi-agents permet de simplifier la phase de conception en laissant à l'algorithme le soin d'optimiser les comportements conduisant à l'émergence. En outre, l'exemple de l'éthologie nous montre que certains comportements de groupes sont fortement influencés par les facultés d'apprentissage des différents individus (spécialisation géographique ou fonctionnelle, mémorisation des actions antérieures, ...)

- Les architectures réactives peuvent ainsi bénéficier d'algorithmes d'apprentissage compatibles avec leur organisation interne. Cela permet à un agent de s'adapter à son environnement et de maintenir ses capacités émergentes lorsque celui-ci évolue.

- Le mode ordonnancé offre aux méthodes neuromimétiques une approche novatrice permettant à un réseau de s'auto-organiser et de développer une représentation semi-distribuée des connaissances. Il offre ainsi une voie vers des algorithmes plus performants (parce que résistants à l'oubli catastrophique et permettant l'apprentissage par partie) et son extension à d'autres modèles connexionnistes (neurones à état réel, réseaux en couches, ...) présente un intérêt indéniable.

Les perspectives de recherche ouvertes par ce travail sont donc nombreuses et touchent de nombreux domaines. C'est pourquoi notre volonté est de le prolonger dans les directions les plus prometteuses et, en liaison avec des laboratoires de productique, d'informatique et d'automatique, de contribuer par nos recherches à la création et à l'exploitation d'architectures « Neuro-Réactives ».

CHAPITRE VIII :

REFERENCES BIBLIOGRAPHIQUES

Références Bibliographiques

[AGR 90] AGRE, P.E., CHAPMAN, D. - « What Are Plans for ? » - *In : Maes, P. (Eds), Designing Autonomous Agents, Theory and Practice from Biology to Engineering and Back* - Amsterdam: Elsevier Sciences Publishers, 1990.- p. 17-34

[ALB 75] ALBUS, J.S. - « A New Approach to Manipulator Control : The Cerebellar Model Articulation Controller (CMAC) » - *Journal of Dynamical Systems, Measurement and Control* - Septembre 1975, Vol. 97 - p. 220-227

[ALE 88] ALEXANDRE, F., BURNOD, Y., GUYOT, F., HATON, J.P. - « La Colonne Corticale : Nouvelle Unité de Base pour des Réseaux Multicouches » - *Actes de NeuroNîmes'88, Nîmes, 15-17 Novembre 1988* - Nanterre: EC2 Eds., 1988 - p. 21-33

[ALK 89] ALKON, D. - « Mémorisation et Neurones » - *Pour la Science* - Septembre 1989, N°. 143 - p. 38-46

[ALL 90] ALLEN, E., MURALI, M., DICAPRIO, P. - « A Modular Architecture for Object Recognition Using Neural Networks » - *Actes de la conférence INNC'90, Paris, 9-13 juillet 1990* - Vol. 1 - Dordrecht: Kluwer Academic Publishers, 1990 - p. 35-37

[AMA 72] AMARI, S. - «Learning Patterns and Patterns Sequences by Self-Organizing Net of Threshold Elements » - *IEEE Transactions on Computers* - Novembre 1972, Vol. C-21 - p. 1197-1206

[AND 88] ANDERSON, J.A., ROSENFELD, E. - *Neurocomputing* - Cambridge: MIT Press, 1988 - 729 p.

[AND 89] ANDERSON, C.W. - «Learning to Control an Inverted Pendulum Using Neural Networks » - *IEEE Control System Magazine* - avril 1989 - p. 31-37

[AND 90] ANDERSON, T.L., DONATH, M. - «Animal Behavior as a Paradigm for Developing Robot Autonomy » - *In Maes, P. (Eds) - Designing Autonomous Agents, Theory and Practice from Biology to Engineering and Back* - Amsterdam: Elsevier Sciences Publishers, 1990 - p. 145-168.

[ANS 94] ANS, B., GILHODES, J.C., VELAY, J.L., COITON, Y. - «Un Réseau de Neurones pour l'Apprentissage et la Programmation de Formes Motrices » - *Actes des journées NSI'94, Chamonix, 2-5 Mai 1994* - p. 123-126

[ARK 87] ARKIN, R.C. - «Motor Schema Based Navigation for a Mobile Robot » - *Actes de IEEE International Conference on Robotics and Automation* - Los Alamitos: IEEE Press Eds., 1987 - p. 264-271

[ARK 90a] ARKIN, R.C. - «Integrating Behavioral, Perceptual, and World Knowledge in Reactive Navigation » - *In Maes, P. (Eds), Designing Autonomous Agents, Theory and Practice from Biology to Engineering and Back* - Amsterdam: Elsevier Sciences Publishers, 1990 - p. 105-122.

[ARK 90b] ARKIN, R.C., MURPHY, R.R. - «Autonomous Navigation in a Manufacturing Environnement » - *IEEE Transactions on Robotics and Automation* - Août 1990, Vol. 6, N°. 1 - p. 445-454.

[ARK 90c] ARKIN, R.C. - «The Impact of Cybernetics on the Design of a Mobile Robot System: A Case Study » - *IEEE Transactions on Systems, Man and Cybernetics* - 1990, Vol. 20, N°. 6 - p. 1245-1257.

[ARK 92] ARKIN, R.C., HOBBS, J.D. - «Dimensions of Communication and Social Organization in Multi-agents Robotic Systems » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge: MIT Press, 1993 - p. 487-493

[ARK 93a] ARKIN, R.C., BALCH, T., NITZ, E. - «Communication of Behavioral State in Multi-agent Retrieval Tasks » - *Actes de IEEE International Conference on Robotics and Automation, Atlanta, 2-6 Mai 1993* - Los Alamitos : IEEE Eds, 1993 - Vol. 3 - p. 588-594.

[ARK 93b] ARKINS, R.C., GRUPEN, R. - «Behavior-Based Reactive Robotic Systems » - *Tutorial T4, IEEE International Conference on Robotics and Automation, Atlanta, 2-7 Mai 1993* - 220 p.

[BAR 83] BARTO, A.G., SUTTON, R.S., ANDERSON, C.W. - «Neuronlike Adaptive Elements that can Solve Difficult Learning Control Problems » - *IEEE Transactions on Systems, Man, and Cybernetics* - 1983, Vol. SMC-13, N°. 5 - p. 834-846

[BAR 89a] BARES, J., HEBERT, M., KANADE, T., KROTKOV, E., MITCHELL, T., SIMMONS, R., WHITTAKER, W. - «Ambler - An Autonomous Rover for Planetary Exploration » - *IEEE Computer Magazine* - juin 1989 - p. 18-26

[BAR 89b] BARAM, Y., - «Associative Memory in Fractal Neural Networks » - *IEEE Transactions on Systems, Man, and Cybernetics* - 1989, Vol. SMC-19, N°. 5 - p. 1133-1141

[BEA 94] BEAUFRÊRE, B. - *Application de la logique floue à la planification de trajectoires de robots mobiles dans des environnements inconnus* - Thèse de troisième cycle - Université de Poitiers, 1994 - 183 p.

[BEE 90] BEER, R.D., CHIEL, H.J., STERLING, L.S. - «A Biological Perspective on Autonomous Agent Design » - *In Maes, P. (Eds) - Designing Autonomous Agents, Theory and Practice from Biology to Engineering and Back* - Amsterdam: Elsevier Sciences Publishers, 1990 - p. 169-186.

[BEE 92] BEER, R.D., CHIEL, H., QUINN, R., ESPENSHIELD, K., LARSSON, P. - «A Distributed Neural Network Architecture for Hexapod Robot Locomotion » - *Neural Computation* - 1992, Vol. 4 - p. 356-365

[BES 92a] BESLON, G. - *Des Modèles Connexionnistes pour le Pilotage des Systèmes de Production* - Mémoire de DEA - Institut National des Sciences Appliquées de Lyon, Laboratoire d'Informatique des Systèmes de Production Industriels, 1992 - 62 p.

[BES 92b] BESSIERE, P. - «Genetic Algorithms Applied to Formal Neural Networks » - *In : Varela, F.J., Bourgine, P. (Eds), Toward a Practice of Autonomous Systems* - Cambridge: MIT Press, 1992 - p. 310-313

[BES 95] BESLON, G., BIENNIER, F., FAVREL, J. - «Flexible Conveying System Based on Autonomous Vehicles » - *Actes de CARs & FoF'95, Pereira, 28-30 Août 1995* - Vol. 1 - Pereira: Universidad Technologia de Pereira, 1994 - p. 115-120

[BID 94] BIDEAUX, E., BAPTISTE, P., PELLET, C. - «Localisation par Réseaux Neuromimétiques » - *Automatique Productique Informatique Industrielle (APII)* - 1994, Vol. 28, N°. 5 - p. 445-453

[BIE 90a] BIENNIER, F. - *Modélisation d'une Base d'Hyperdocuments et Méthodes Connexionnistes d'aide à la navigation* - Thèse de troisième cycle - Institut National des Sciences Appliquées de Lyon, 1990 - 256 p.

[BIE 90b] BIENNIER, F., PINON, J.M., GUIVARCH, M. - «A Connectionist Method to Retrieve Information in Hyperdocuments » - *Actes de la conférence INNC'90, Paris, 9-13 juillet 1990* - Vol. 1 - Dordrecht: Kluwer Academic Publishers, 1990 - p. 444-448

[BIE 92a] BIENNIER, F., BESLON, G., FAVREL, J., - «Utilisation de Réseaux Neuronaux pour le Pilotage d'Ateliers Flexibles » - *Actes de NeuroNîmes'92, Nîmes, 2-6 Nov. 1992* - p. 465-476

[BIE 92b] BIENNIER, F., FAVREL, J., GUIVARCH, M. - «Workshop Layout Guided By Neural Networks » - *Actes de CARs & FOF 92*, *Metz, Août 1992* - p. 709-723

[BIE 93] BIENNIER, F., FAVREL, J., BESLON, G. - «Integration of Information and Knowledge from the Engineering Activity to the Workshop Control » - *Actes de IEEE International Conférence on Robotics and Automation, Atlanta, Mai 1993* - Los Alamitos : IEEE Press, 1993 - Vol. 1, p. 840-845

[BIE 95a] BIENNIER, F., FAVREL, J. - «Technologie de Groupe Assistée par Réseaux Neuronaux » - - *Automatique Productique Informatique Industrielle (APII)* - 1995, Vol. 29, N°. 3 - p. 285-305

[BIE 95b] BIENNIER, F., COQUARD, P., FAVREL, J. - «An Hypermedia Based Meta-Model for Concurrent Engineering » - *Concurrent Engineering, Research and Applications* - 1995, Vol. 3, N°. 1 - p. 3-11

[BOM 94] BOMBARDIER, V., SIMON, C., GANDIN, P., DUBOIS, G., BREMONT, J. - «Reconnaissance Couleur par Logique Floue » - *Automatique Productique Informatique Industrielle (APII)* - 1994, Vol. 28, N°. 5 - p. 479-486

[BON 87] BONETTO, R. - *Les ateliers flexibles de production (2nd Edition)* - Paris : Hermes, 1987 - 250 p.

[BON 94] BONABEAU, E., THERAULAZ, G. (coordonnateurs) - *Intelligence Collective* - Paris: Hermes, 1994 - 288 p.

[BRA 84] BRAITENBERG, V. - *Vehicles : Experiments in Synthetic Psychology* - Cambridge: MIT Press, 1984 - 320 p.

[BRA 91] BRAITENBERG, V. - «Information from Structure : A Sketch of Neuroanatomy » - *In Domany, Van Hemmen et Schulten's (Eds.) «Models of Neural Networks »* - Berlin : Springer-Verlag, 1991 - p. 107-120

[BRO 86] BROOKS, R.A. - «A Robust Layered Control System for a Mobile Robot » - *IEEE Journal of Robotics and Automation* - 1986, Vol. RA-2, N°. 1 - p. 14-23

[BRO 89] BROOKS, R.A. - «A Robot That Walk ; Emergent Behavior from a Carefully Evolved Network » - *Neural Computation* - 1989, Vol. 1 - p. 253-262.

[BRO 90a] BROOKS, R.A. - «Elephants Don't Play Chess » - *In Maes, P. (Eds), Designing Autonomous Agents, Theory and Practice from Biology to Engineering and Back* - Amsterdam: Elsevier Sciences Publishers, 1990 - p. 3-15

[BRO 90b] BROOKS, R.A. - «Network Based Autonomous Robot Motor Control: From Hormones to Learning » - *In : Eckmiller, R. (Eds) , Advanced Neural Computers* - Amsterdam : Elsevier Science Publishers B.V., 1990 - p. 341-348.

[BRO 91] BROOKS, R.A. - «Intelligence Without Reason » - *Actes de la Twelfth International Joint Conference on Artificial Intelligence* - Boston: Morgan-Kaufmann Publishers, 1991 - p. 569-595

[BRU 93] BRUN-PICARD, D., FERRARINI, A., NERVI, M.L., COUVREUR, L. - «Proposition d'une Architecture Décentralisée Autonome et Ouverte sur le Pilotage d'Atelier Flexible de Fabrication » - *Actes du 4ème Congrès International de Génie Industriel, Marseille, 15-17 Décembre 1993* - Vol. 2 - p. 395-403

[BUL 92] BULLIER, J., SALIN, P., GIRARD, P. - «Le Cerveau en Temps Réel » - *La recherche* - 1992, Vol. 23, N°. 246 - p. 974-981

[BUR 88] BURNOD, Y., - *An Adaptative Neural Network : The Cerebral Cortex* - Paris : Masson, 1988 - 376 p.

[BUR 90] BURNOD, Y., GRANDGUILLAUME, P., CAMINITI, R., JOHNSON, P. «Model of the Visuo-Motor Transformation Produced by the Cerebral Cortex for 3D Arm Movements » - *Actes de la conférence INNC'90, Paris, 9-13 juillet 1990* - Vol. 1 - Dordrecht: Kluwer Academic Publishers, 1990 - p. 463-466

[CAM 88] CAMPAM, R., BEUGNON, G., LAMBIN, M. - «De l'Information Visuelle à la Représentation de l'Espace chez les Insectes » - *Annales de la société entomologique de France* - N°. 4 - Oct./Dec. 1988 - p. 433-438.

[CAM 95] CAMPAGNE, J.P., FAVREL, J., «A Framework to Specify a Reactive and Proactive Management System » - *A Paraître dans les actes de ETFA'95, Paris, 10-13 octobre 1995 - 6 p.*

[CAR 86] CARPENTER, G., GROSSBERG, S. - «Adaptive Resonnance Theory : Stable self-organization of neural recognition code in response to arbitrary list of input patterns » - *Actes de Annual Conference of the cognitive Science Society, Hillsdale (USA)*, 1986 - p. 45-62.

[CAR 87] Carpenter, G., Grossberg, S. - «ART2 : Self-organization of stable category recognition codes for analog input patterns » - *Actes de IEEE International Conference on Neural Networks, San Diego (USA), 1987* - Vol. 2 - Los Alamitos : IEEE Press, 1987 - p. 727-736.

[CAR 91] Carpenter, G., Grossberg, S., Reynolds, J. - «ARTMAP : Supervised Real-Time Learning and Classification of Nonstationary Data by a Self-Organizing Neural Network » - *Neural Networks* - 1991, Vol. 4 - p. 543-564

[CHE 88] Cheung, J., Omidvar, M. - «Mathematical Analysis of Learning Behavior of Neuronal Models » - *Actes de 1987 IEEE Conference on Neural Information Processing Systems - Natural and Synthetic, New-York, 1987* - Los Alamitos: IEEE Press Eds., 1988 - p. 174-183.

[CHE 91] CHEN, S., COWAN, C.F.N., GRANT, P.M. - «Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks » - *IEEE Transactions on Neural Networks* - 1991, Vol. 2, N°. 2 - p. 302-309

[CHE 92] CHERIAN, S., TROXELL, W.O. - «A Neural Network Based Behavior Hierarchy for Locomotion Control » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge: MIT Press, 1993 - p. 61-70

[COC 88] COCHET, Y. PAGET, G. - «Réseaux Neuronaux pour Base d'Image » - *Actes du congrès NeuroNîmes'88, Nîmes, 15-17 Novembre 1988* - Nanterre : EC2 Eds, 1988 - p. 283-293

[COH 83] COHEN, M., GROSSBERG, S. - «Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks » - *IEEE Transactions on Systems, Men and Cybernetics* - 1983, Vol. SMC-13, N°. 5 - p. 815-825.

[COH 92] COHENDET, P., LLERENA, P., MUTEL, B. - «Flexibilités et mise en cohérence des données de production » - *In : De Tressac, G., Dubois, P. - Les nouvelles rationalisations de la production* - Toulouse : Cépaduès-Editions, 1992 - p. 25-42.

[COI 93] COIFFET, P. - *Robot Habilis Robot Sapiens* - Paris : Hermes, 1993 - 425 p.

[COL 75] COLLINS, A.M., LOFTUS, E.F., - «A Speading Activation Theory of Semantic Processing » - *Psychological Review* - 1975, N°. 82 - p. 407-428

[COM 94] COMBES, D., SIMMERS, J., MEYRAND, P., MOULINS, M. - «Mécanismes Cellulaires et Synaptiques Responsable de Effet Multiples d'un Unique Neurone Sensoriel Primaire sur un Générateur Central de Rythmes Moteurs » - *Actes des journées NSI'94, Chamonix, 2-5 Mai 1994* - p. 87-89

[COU 94] COUTURIER, P., JOHANNET, A., SARDA, I., TOUZET, C. - «Robot Hexapode: Réalisation et Apprentissage de la Marche » - *Actes des journées NSI'94, Chamonix, 2-5 Mai 1994* - p. 59-62

[COX 92] COX, C.J., LOTHERS, M., PAP, R.M., THOMAS, C.R. - «A Decentralized Adaptive Joint Neurocontroller » - *Actes de IJCNN'92, Baltimore, 7-11 juin 1992* - Vol. 2 - Los Alamitos: IEEE Press Eds. - p. 869-873

[CRA 89] CRAYE, E., GENTINA, J.C. - «Distributed Implementation and Hierarchical Level for Control of Flexible Manufacturing Systems » - *Actes de AIPAC'89* - Vol. 2 - 1989 - p. 190-193

[CRU 92] CRUSE, H., MÜLLER-WILM, U., DEAN, J. - «Artificial Neural Nets for Controlling a 6-legged walking system » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge: MIT Press, 1993 - p. 52-60

[DEL 94] DELACOUR, J. - *Biologie de la Conscience* - Paris : Presses Universitaires de France, 1994 - 128 p.

[DEM 92] DEMUTH, H., BEALE, M. - *Neural Network Toolbox* - Natick: The MathWorks Inc. 1992, 200 p.

[DEW 87] DEWDNEY, A. - «Récréations Informatiques » - *Pour la science* - Mai 1987, Vol. 167 - p. 18-22

[DIN 90] DINSE, H.R., RECANZONE, G.H., MERZENICH, M. - «Direct Observation of Neural Assembly Interaction During Cortical Representational Reorganization - *Actes de la conférence INNC'90, Paris, 9-13 juillet 1990* - Vol. 1 - Dordrecht: Kluwer Academic Publishers, 1990 - p. 533-536

[DOR 92] DORIZZI, B., DUVAL, J.M., DEBAR, H. - «Utilisation de Réseaux Récurrents pour la Prévision de la Consommation Electrique » - *Actes de Neuro-Nîmes'92, Nîmes, 2-6 Novembre 1992* - Nanterre: EC2 Eds, 1992 - p. 141-150

[DOR 94] DORIZZI, B. - «Réseaux de Neurones et Reconnaissance de Formes » - *Actes du séminaire COMETT II, Lyon, 1994* - INSA de Lyon, 1994 - p. 37-50

[DOR 96] DORIGO, M., MANIEZZO, V., COLORNI, A. - «The Ant System: Optimization by a Colony of Cooperating Agents » - *A Paraître dans IEEE Transactions on systems, man, and cybernetics - 1996, Vol. 26, N°. 2*

[DOT 93] DOTY, K.L., VAN AKEN, E. - «Swarm Robot Material Handling Paradigm For a Manufacturing Workcell » - *Actes de 1993 IEEE International Conference on Robotics and Automation, Atlanta, 2-6 Mai 1993* - Vol. 1 - Los Alamitos: IEEE Press Eds, 1993 - p. 778-782

[DRO 92] DROGOUL, A., FERBER, J. - «From Tom Thumb to the Dockers: Some Experiments with Foraging Robots » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge: MIT Press, 1993 - p. 451.459

[DUB 90] DUBOIS, D. - *Le Labyrinthe de l'Intelligence* - Paris: InterEditions, 1990 - 331 p.

[DUR 94] DURAND, D. - *La Systémique* - Paris: PUF, 1994 - 126 p.

[ERC 91] ERCEAU, J., FERBER, J. - «L'intelligence artificielle distribuée » - *La Recherche* - 1991, Vol. 22, N° 233 - p. 750-758

[FEC 93] FECHNER, T., HINZE, A. - «Delta Analysis : A Method for the Determination of Input Feature Significance in Neural Networks » - *Actes de NeuroNîmes'93, Nîmes, 25-29 Octobre 1993* - Nanterre: EC2 Eds., 1993 - p. 393-398

[FER 94] FERBER, J. - «Coopération Réactive et Emergence » - *Intellectica* - 1994, Vol. 19, N°. 2 - p. 19-52.

[FRA 92] FRANCESCHINI, N. - «L'Intelligence Visuo-Motrice » - *Le Courrier du CNRS* - 1992, N°. 79 - p. 47

[FRE 91] FRENCH, R.M. - «Using Semi-Distributed Representation to Overcome Catastrophic Forgetting in Connectionist Networks » - *Rapport Technique 51-1991 du CRCC* - Floomington : Center for Research on Concepts and Cognition - 1991 - 8 p.

[FUN 89] FUNAHASHI, K.I. - «On the Approximate Realization of Continuous Mapping by Neural Networks » - *Neural Networks* - 1989, Vol. 2 - p.183-192

[GAL 92] GALLAGHER, J.C., BEER, R.D. - «A Qualitative Dynamical Analysis of Evolved Locomotion Controllers » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge:: MIT Press, 1993 - p. 71.80

[GEV 93] GEVA, S., SITTE, J. - «A Cartpole Experiment Benchmark for Trainable Controllers » - *IEEE Control Systems* - 1993, Vol. 13, N°. 5 - p. 40-51

[GIA 94] GIARD, V. - *Gestion de la Production (2nd Edition)* - Paris : Economica Eds, 1994 - 1068 p.

[GIR 84] GIRALT, G., CHATILA, R., VAISSET, M. - «An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots » - *Actes de International Symposium on Robotic Research* - Cambridge : MIT Press, 1984 - p. 191-214

[GIR 93] GIRALT, G. - «Robots d'intervention : Autonomie, Réactivité, Programmation » - *Actes du séminaire international LAAS-CNRS, Toulouse, 24-28 Mai 1993* - Toulouse : Cépaduès Eds., 1993 - p. 95-111.

[GOL 94] GOLDBERG, D.E. - *Algorithmes Génétiques* - Paris : Addison-Wesley France, 1994 - 417 p.

[GRA 89] GRAY, C.M., KÖNIG, P., ENGEL, A.K., SINGER, W. - «Oscillatory Responses in Cat Visual Cortex Exhibit Inter-columnar Synchronization which Reflects Global Stimulus Properties » - *Nature* - 1989, Vol. 338, N° 6213 - p. 334-337

[GRO 73] GROSSBERG, S. - «Contour Enhancement, Short Term Memory, and Constancies in Reverberating Networks » - *Studies in Applied Mathematics* - 1973, N°. 52 - p. 217-257

[GRO 76] GROSSBERG, S. «Adaptive Pattern Classification and Universal Recoding. I: Parallel Development and Coding of Neural Feature Detectors » - *Biological Cybernetics* - 1976, N°. 23 - p. 121-134

[GRO 87] GROSSBERG, S., LEVINE, D.S. - «Neural Dynamics of Attentionally Modulated Pavlovian Conditioning: Blocking, Interstimulus Interval, and Secondary Reinforcement » - *Applied Optics* - 1987, Vol. 26, N°. 23 - p. 5015-5030

[GRO 91a] GROSSBERG, S., SOMERS, D. - «Synchronized Oscillations During Cooperative Feature Linking in Visual Cortex » - *Actes de IJCNN'91, Seattle, 8-12 juillet 1991* - Vol. 2 - Los Alamitos : IEEE Press, 1991 - p. 249-254

[GRO 91b] GROSSBERG, S. - «Unsupervised Learning (tutorial notes) » - *Actes de IJCNN'91, Seattle, 8-12 juillet 1991* - Vol. 2 - Los Alamitos : IEEE Press, 1991 - 102 p.

[GUI 37] GUILLAUME, P. - *La Psychologie de la Forme* - Paris : Flammarion, 1979 (première édition : 1937) - 258 p.

[GUL 90] GULLAPALLI, V. - «A Stochastic Reinforcement Learning Algorithm for Learning Real-Valued Functions » - *Neural Networks* - 1990, Vol. 3 - p. 671-692

[GUT 94] GUTSCHE, R., LALONI, C., WAHL, F.M. - «Object Flow Fields - A New Paradigm for Path Planning for AGVs Within Production Environments » - *Actes de DKSME'94, Hong-Kong, 2-4 Mai* 1994 - Vol. 1 - Hong-Kong: Chinese University of Hong-Kong, 1994 - p. 216-221

[HAM 92] HAMPSHIRE II, J.B., WAIBEL, A. - «The Meta-Pi Network: Building Distributed Knowledge Representations for Robust Multisource Pattern Recognition » - *IEEE Transactions on Pattern Analysis and Machine Intelligence* - 192, Vol. 14, N°. 7 - p. 751-769

[HAM 95] HAMILTON, P. - «An Approach to Transfer Neurobiological Concepts into Artificial Neural Nets » - *Actes de la conférence «Neural Parallel and Scientific Computations, Altanta, 28-31 Mai 1995* - Atlanta : Dynamic Publishers Inc., 1995 - p. 193-196

[HAN 95] HANSEL, D., MATO, G., MEUNIER, C. - «Synchrony in Axcitatory Neural Networks » - *Neural Computation* - 1995, Vol. 7, N°. 2 - Cambridge: MIT Press, 1995 - p. 307-337

[HAR 92] HARVEY, I., HUSBANDS, P., CLIFF, D. - «Issues in Evolutionary Robotics » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge: MIT Press, 1993 - p. 364-373

[HAT 92] ATCHUEL, A., SARDAS, J.C. - «Les grandes transitions contemporaines des systèmes de production, une démarche typologique » - *In : De Tressac, G, Dubois, P. - Les nouvelles rationalisations de la production* - Toulouse : Cépaduès-Editions, 1992- p. 1-24.

[HEB 49] HEBB, D.O. - *The Organization of Behaviour* - New-York : J. Wiley & sons Eds, 1949 - 130 p.

[HIN 86] HINTON, G.E., SEJNOWSKI, T.J. «Learning and Relearning in Boltzmann Machines » - *In* : *Rumelhart, D.E., McClelland, J.L. (Eds), Parallel Distributed Processing, Exploring the Micro-Structure of Cognition, Vol. 1 : Foundations* - Cambridge : MIT Press, 1986 p. 282-317

[HOF 92] HOFFMEISTER, F., BÄCK, T. - «Genetic Self-Learning » - *In : Varela, F.J., Bourgine, P. (Eds), Toward a Practice of Autonomous Systems* - Cambridge: MIT Press, 1992 - p. 227-235

[HOL 75] HOLLAND, J. - *Adaptation in Natural and Artificial Systems* - Ann Arbor: University of Michigan Press, 1975

[HOO 89] HOOPER, S.L., MOULINS, M. - «Switching of a Neuron from One Network to Another by Sensory-Induced Changes in Membrane Properties » - *Science* - 1989, Vol. 244 - p. 1587-1589

[HOP 82] HOPFIELD, J.J. - «Neural Networks and Physical Systems with Emergent Collective Computational Abilities » - *Proceedings of the National Academy of Sciences of the USA* - 1982, Vol. 72 - p. 2554-2558

[HOP 84] HOPFIELD, J.J. - «Neuron with graded response have collective computational abilities » - *Proceedings of the National Academy of Sciences of the USA* - 1984, Vol. 81 - p. 3088-3092

[HOP 85] HOPFIELD, J.J., TANK D. - «Neural Computations of décision in optimization problems » - *Biological Cybernetics* - 1985, N°. 52 - p. 141-152

[HOR 89] HORNIK, K., STINCHCOMBE, M., WHITE, H. - «Multilayer Feedforward Networks are Universal Approximators » - *Neural Networks* - 1989, Vol. 2, N°. 5 - p. 359-366

[HOR 91] HORN, D., USHER, M. - «Segmentation and Binding in an Oscillatory Neural Network » -*Actes de IJCNN'91, Seattle, 8-12 Juillet 1991* - Vol. 2 - Los Alamitos: IEEE Press Eds, 1991 - p. 243-248

[HUN 92] HUNT, K.J., SBARBARO, D., SBIKOWSKI, R., GAWTHROP, P.J. - «Neural Networks for Control - A Survey » - *Automatica* - 1992, Vol. 28, N°. 6 - p. 1083-1112.

[ICH 92] ICHIKAWA, Y., SAWA, T. - «Neural Network Application for Direct Feedback Controllers » - *IEEE Transactions on Neural Networks* - 1992, Vol. 3, N°. 2 - p. 224-231

[IYE 91] IYENGAR, S.S., ELFES, A. - *Autonomous Mobile Robots, Control, Planning, and Architecture* - Los Alamitos : IEEE Press, 1991 - 527 p.

[JAC 93] JACOBS, R.A., JORDAN, M.I. - «Learning Piecewise Control Strategies in a Modular Neural Network Architecture » - *IEEE Transactions on Systems, Man, and Cybernetics* - 1993, Vol. SMC-23, N°. 2 - p. 337-345

[JAG 93] JAGOU, P. - *Concurrent Engineering* - Paris : Hermes Eds, 1993 - 140 p.

[JOR 92] JORDAN, M.I., RUMELHART, D.E. - «Forward Models: Supervised Learning with a Distal Teacher » - *Cognitive Science* - 1992, Vol. 16, N°. 3 - p. 307-354

[JOR 94] JORDAN, M.I., JACOBS, R.A. - «Hierarchical Mixtures of Experts and the EM Algorithm » - *Neural Computation* - 1994, Vol. 6 - Cambridge: MIT Press, 1994 - p. 181-214

[KAE 90] KAELBLING, L.P., ROSENSCHEIN, S.J. - «Action and Planning in Embedded Agents » - *In Maes, P. (Eds), Designing Autonomous Agents, Theory and Practice from Biology to Engineering and Back* - Amsterdam: Elsevier Sciences Publishers, 1990 - p. 35-48

[KAL 85] KALLEL, G., PELLET, X., BINDER, Z. - «Conduite Décentralisée Coordonnée d'Atelier » - *Revue d'Automatique, de Productique et d'Informatique Industrielle (APII)* - 1985, Vol. 19, N°. 4 - p. 371-387

[KAM 90] KAMP, Y., HASLER, M. - *Réseaux de Neurones Récursifs pour Mémoires Associatives* - Lausanne : Presses Polytechniques et Universitaires Romandes, 1990 - 237p.

[KAN 88] KANDEL, E. - «Les Petits Systèmes de Neurone » - *In : Hubel, P. et al. (Eds.)*, *Le cerveau* - Paris : Belin, 1988 - p. 29-40

[KIR 83] KIRKPATTRICK, S., GELATT, C.D., VECCHI, M.P. - «Optimization by Simulated Annealing » - *Science* - 1983, Vol. 220 - p. 671-680

[KLO 86] KLOPF, A.H. - «Drive-Reinforcement Model of Single Neuron Function : An Alternative to Hebbian Neuronal Model » - *Actes de AIP Conference 151 : Neural Networks for Computing, New-York, 1986* - New-York: American Institut of Physics, 1986 - p. 265-270.

[KLO 90] KLOPF, A.H. - «The Role of Time in Natural Intelligence: Implications for Neural Network and Artificial Intelligence Research » - *In : Eckmiller, R.(Eds) , Advanced Neural Computers* - Amsterdam : Elsevier Science Publishers B.V., 1990 - p. 201-207.

[KLO 92] KLOPF, A.H., MORGAN, J.S., WEAVER, S.E. - «Modeling Nervous System Function with a Hierarchical Network of Control Systems that Learn » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge: MIT Press, 1993 - p. 254-261

[KOH 77] KOHONEN, T. - *Associative Memory* - Berlin : Springer-Verlag, 1977 - 256 p.

[KOH 84] KOHONEN, T. - *Self-Organization and Associative Memory* - Berlin : Springer Verlag, 1984 - 312 p.

[KÖN 92] KÖNIG, P., JANOSCH, B., SCHILLEN, T.B. - «Stimulus-Dependent Assembly Formation of Oscillatory Responses: III. Learning » - *Neural Computation* - 1992, Vol. 4, N°. 5 - Cambridge: MIT Press, 1992 - p. 666-681

[KOS 86a] KOSKO, B. - «Fuzzy Cognitive Maps » - *International Journal of Man/Machine Studies - 1986, Vol. 24* - p. 65-75

[KOS 86b] KOSKO, B. - «Differential Hebbian Learning » - *Actes de AIP Conference 151 : Neural Networks for Computing, New-York, 1986* - New-York: American Institute of Physics, 1986 - p. 277-282.

[KOS 88] KOSKO, B. - «Feedback Stability and Unsupervised Learning » - *Actes de International Conference on Neural Networks (ICNN'88), juillet 1988* - Vol. 1 - p. 141-152

[KOU 92] KOUAM, A., BADRAN, F., THIRIA, S. - «Approche Méthodologique pour l'Etude de la Prévision à l'Aide de Réseaux de Neurones » - *Actes de Neuro-Nîmes'92, Nîmes, 2-6 Novembre 1992* - Nanterre: EC2 Eds, 1992 - p. 117-127

[KUB 92] KUBE, C.R., ZHANG, H. - «Collective Robotic Intelligence » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge: MIT Press, 1993 - p. 460.468

[LAN 92] LANGLOIS, T. - *Algorithmes d'Apprentissage par Renforcement pour la Commande Adaptative* - Thèse de troisième cycle - Université de Technologie de Compiègne, 1992 - 125 p.

[LEC 85] LE CUN, Y. - «Une Procédure d'Apprentissage pour Réseaux à Seuil Asymétrique » - *Actes de Cognitiva'85, Paris, Juin 1985* - p. 599-604.

[LEG 92] LEGAIT, A. - *L'intégration : Quels Impacts sur la Spécification d'un Système de Gestion de Production Assistée par Ordinateur* - Thèse de troisième cycle - Institut National des Sciences Appliquées de Lyon, 1992 - 226 p.

[LEN 94] LENAY, C. - «Organisation Emergente dans le Populations : Biologie, Ethologie, Systèmes Artificiels » - *Intellectica* - 1994, Vol. 19, N°. 2 - p. 9-17

[LIN 91] LIN, C.S., KIM, H. - «CMAC-Based Adaptive Critic Self-Learning Control » - *IEEE Transactions on Neural Networks* - 1991, Vol. 2, N°. 5 - p. 530-533

[LIN 92] LIN, L.J. - «Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching » - *Machine Learning* - 1992, Vol. 8 - Boston: Kluwer Academic Publishers, 1992 - p. 293-321

[LOR 91] LORQUET, V., PUGET, P., NIEZ, J.J. - «Réseaux Multicouches à Accès Semi-distribué » - *Actes de NeuroNîmes'91, Nîmes, 4-8 Novembre 1991* - Nanterre: EC2 Eds, 1991 - p. 397-410

[MAE 90a] MAES, P. (Eds) - *Designing Autonomous Agents, Theory and Practice from Biology to Engineering and Back* - Amsterdam: Elsevier Sciences Publishers, 1990 - 193 p.

[MAE 90b] MAES, P. - «Situated Agents Can Have Goals » - *In Maes, P. (Eds) - Designing Autonomous Agents, Theory and Practice from Biology to Engineering and Back* - Amsterdam: Elsevier Sciences Publishers, 1990 - p. 49-70

[MAE 90c] MAES, P., BROOKS, R.A. «Learning to Coordinate Behaviors » - *Actes de AAAI'90, Boston, 1990* - Boston: Morgan-Kaufmann Publishers, 1990 - p. 796-702

[MAE 92a] MAES, P. - «Learning Networks from Experience » - *In : Varela, F.J., Bourgine, P. (Eds), Toward a Practice of Autonomous Systems* - Cambridge: MIT Press, 1992 - p. 48-57

[MAE 92b] MAES, P. - «Behavior-Based Artificial Intelligence » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge: MIT Press, 1993 - p. 2-10

[MAL 90a] MALCOM, C., SMITHERS, T. - «Symbol Grounding via a Hybrid Architecture in an Autonomous Assembly System » - *In Maes, P. (Eds), Designing Autonomous Agents, Theory and Practice from Biology to Engineering and Back* - Amsterdam: Elsevier Sciences Publishers, 1990 - p. 123-144

[MAL 90b] MALSBURG, VON DER C. - «Considerations for a Visual Architecture » - *In : Eckmiller, R. (Eds) , Advanced Neural Computers* - Amsterdam : Elsevier Science Publishers B.V., 1990 - p. 303-313

[MAR 90a] MARGARITA, S. - «Recognition of European Car Plates with Modular Neural Networks » - *Actes de la conférence INNC'90, Paris, 9-13 juillet 1990* - Vol. 1 - Dordrecht: Kluwer Academic Publishers, 1990 - p. 408-411

[MAR 90b] MARTINETZ, T., SCHULTEN, K. - «Hierarchical Neural Net for Learning Control of a Robot's Arm and Gripper » - *Actes de IJCNN'90, San Diego, 17-21 juin 1990* - Vol. 2 - p. 747-752

[MAR 94] MARTIN, N., FRANCESCHINI, N. - «Contrôle Automatique de la Vitesse et de l'Orientation d'un Agent Mobile » - *Actes des journées NSI'94, Chamonix, 2-5 Mai 1994* - p. 133-138

[MAT 92] MATARIC, M.J. - «Designing Emergent Behaviors: From Local Interactions to Collective Intelligence » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge: MIT Press, 1993 - p. 432-441

[McC 43] MCCULLOCH, W., PITTS, W. - «A Logical Calculus for the Ideas Immanent in Nervous Activity » - *Bulletin Mathematics and Biophysics* - 1943, N°. 5 - p. 115-133

[McC 47] MCCULLOCH, W., PITTS, W. - «How We Know Universal: The Perception of Auditory and Visual Forms » - *Bulletin Mathematics and Biophysics* - 1943, N°. 9 - p. 127-147

[MEY 91] MEYRAND, P., SIMMERS, J., MOULINS, M. - «Construction of a Pattern Generating Circuit with Neurons of Different Networks » - *Nature* - 1991, Vol. 351, N°6321 - p. 60-63

[MIC 68] MICHIE, D., CHAMBERS, R.A. - «BOXES: An Experiment in Adaptive Control » - *In : Dale, E., Michie, D. (Eds) Machine Intelligence 2* - Edimburgh: Oliver and Boyd Eds, 1968 - p. 137-152

[MID 91] MIDENET, S. - *Modèle connexionniste d'apprentissage d'associations par carte auto-organisatrice. Contribution à l'étude d'une mise en correspondance des représentations connexionnistes et symboliques* - Thèse de troisième cycle - Ecole Nationale Supérieure des Télécommunications, 1991 - 231 p.

[MIL 87] MILLER III, W.T. - «Sensor-Based Control of Robotic Manipulators Using a General Learning Algorithm » - *IEEE Journal of Robotics and Automation* - 1987, Vol. RA-3, N°. 2 - p. 157-165

[MIL 90] MILLER, W.T. III, GLANZ, F.H., KRAFT, L.G. III - «CMAC: An Associative Neural Network Alternative to Backpropagation » - *Proceedings of the IEEE* - 1990, Vol. 78, N°. 10 - p. 1561-1567

[MIN 68] MINSKY, M., PAPPERT, S. - *Perceptron* - Cambridge : MIT Press, 1968 - 275 p.

[MIR 94] MIRANKER, W.L., WILLNER, B.E. - «A Neural System for the Animat » - *Neural, Parallel & Scientific Computations* - 1994, Vol. 2, N°. 4 - Atlanta : Dynamic Publishers, 1994 - p. 517-532

[MIT 92] MITCHELL, M., FORREST, S., HOLLAND, J.H. - «The Royal Road for Genetic Algorithms: Fitness Landscape and GA Performance » - *In : Varela, F.J., Bourgine, P. (Eds), Toward a Practice of Autonomous Systems* - Cambridge: MIT Press, 1992 - p. 245-254

[MON 94] MONDADA, F., FLOREANO, D. - «Conception Evolutionniste de Réseaux de Neurones pour le Contrôle de Robots Mobiles » - *Actes des journées NSI'94, Chamonix, 2-5 Mai 1994* - p. 95-98

[MOR 95] MORIN, E. - «Vers un Nouveau Paradigme » - *Sciences Humaines* - février 1995, N°. 47 - p. 20-23

[MOZ 92] MOZER, M.C., ZEMEL, R.S., BEHRMANN, M., WILLIAMS, C.K.I. - «Learning to Segment Images Using Dynamic Feature Binding » - *Neural Computation* - 1992, Vol. 4, N°. 5 - Cambridge : MIT Press, 1992 - p. 650-665

[MUR 94] MURA, F., FRANCESCHINI, N. - «Vision de Mouvement pour le Contrôle de l'altitude et de la vitesse d'un aéronef » - *Actes des journées NSI'94, Chamonix, 2-5 Mai 1994* - p. 143-148

[NEU 48] VON NEUMANN, J. - «The General and Logical Theory of Automata » - *In : Lloyd A. J. (Eds), Cerebral Mechanism in Behavior : The Hixon Symposium* - New-York: John Wiley and Sons, 1951

[NEW 93] NEWTON, R.T., XU, Y. - «Real-Time Implementation of Neural Network Learning Control of a Flexible Space Manipulator » - *Actes de 1993 IEEE International Conference on Robotics and Automation, Atlanta, 2-6 Mai 1993* - Vol. 1 - Los Alamitos: IEEE Press Eds, 1993 - p. 135-141

[NGU 90] NGUYEN, D., WIDROW, B. - «The Truck Backer-Upper » - *Actes de la conférence INNC'90, Paris, 9-13 juillet 1990* - Vol. 1 - Dordrecht: Kluwer Academic Publishers, 1990 - p. 399-407

[NIL 69] NILSSON, N.J. - «A Mobile Automaton: An Application of Artificial Intelligence Techniques » - *Actes de International Joint Conference on Artificial Intelligence, 1969* - San-mateo : Morgan Kauffman Publishers, 1969 - 12 p.

[NUM 92] NUMAOKA, C., TAKEUCHI, N. - «Collective Choice of Strategic Type » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge: MIT Press, 1993 - p. 469-477

[OPI 90a] OPITZ, R. - «Neural Network Application for Autonomous Mobile Robots (Lernfahrzeug) » - *Actes de la conférence INNC'90, Paris, 9-13 juillet 1990* - Vol. 1 - Dordrecht: Kluwer Academic Publishers, 1990 - p. 225-228

[OPI 90b] OPITZ, R. - «Das Lernfahrzeug Neural Network Application for Autonomous Mobile Robots » - *In : Eckmiller, R. (Eds) , Advanced Neural Computers* - Amsterdam : Elsevier Science Publishers B.V., 1990 - p. 373-379

[PAR 92a] PARKER, L.E. - «Adaptive Action Selection for Cooperative Agent Teams » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge: MIT Press, 1993 - p. 442-450

[PAR 92b] PARISI, D., NOLFI, S., CECCONI, F. - «Learning, Behavior, and Evolution » - *In : Varela, F.J., Bourgine, P. (Eds), Toward a Practice of Autonomous Systems* - Cambridge: MIT Press, 1992 - p. 207-216

[PAY 90] PAYTON, D.W. - «Internalized Plans: A Representation for Action Resources » - *In Maes*, *P. (Eds) - Designing Autonomous Agents, Theory and Practice from Biology to Engineering and Back* - Amsterdam: Elsevier Sciences Publishers, 1990 - p. 89-103

[PEA 89] PEARLMUTTER, B.A. - «Learning State Space Trajectories in Recurrent Neural Networks » - *Neural Computation* - 1989, Vol. 1 - Cambridge: MIT Press, 1989 - p. 263-269

[PEL 92] PELLIONISZ, A.J., JÖRGENSEN, C.C., WERBOS, P.J. - «Cerebellar Neurocontroller Project, for Aerospace Applications, in a Civilian Neurocomputing Initiative in the «Decade of the Brain» » - *Actes de IJCNN'92, Baltimore, 7-11 Juin 1992* - Vol. 3 - Los Alamitos: IEEE Press Eds, 1992 - p. 379-384

[PEL 95] PELISSIER, A., TETE, A. - *Sciences Cognitives, Textes fondateurs (1943-1950)*- Paris: PUF, 1995 - 313 p.

[PIA 64] PIAGET, J. - *Six Etudes de Psychologie* - Paris: Denoël, 1964 - 217 p.

[POG 90] POGGIO, T., GIROSI, F. - « Networks for Approximation and Learning » - *Proceedings of the IEEE* - 1990, Vol. 778, N°. 9 - p. 1481-1497

[POM 93] POMERLEAU, D.A. - *Neural Network Perception for Mobile Robot Guidance* - Boston : Kluwer Academic Publishers, 1993 - 191 p.

[PRA 91] PRAJOUX, R., GIRALT, G. - «Issues and Trends in Non-Manufacturing Robotics » - *Revue d'Automatique et de Productique Appliquées (RAPA)* - 1991, Vol. 4, N°. 1 - p. 9-35

[PSA 88] PSALTIS, D., SIDERIS, A., YAMAMURA, A.A. - «A Multilayered Neural Network Controller » - *IEEE Control System Magazine* - 1988, Vol. 8, N°. 2 - Los Alamitos: IEEE Press Eds., 1988 - p. 17-21

[RAB 91] RABELO, L.C., AVULA, X.J.R. - «Intelligent Control of a Robotic Arm Using Hierarchical Neural Network Systems » - *Actes de IJCNN'91, Seattle, 8-12 Juillet 1992* - Vol. II - Los Alamitos: IEEE Press Eds, 1991 - p. 747-751

[RAB 92] RABELO, L.C., AVULA, X.J.R. - «Hierarchical Neurocontroller Architecture for Robotic Manipulation » - *IEEE Control System Magazine* - Avril 1992 - p. 37-41

[RAM 93] RAM, A., SANTAMARIA, J.C. - «Multistrategy Learning in Reactive Control Systems for Autonomous Robotic Navigation » - *Informatica* - 1993, Vol. 17, N°. 4 - p. 347-369

[RAS 82] RASMUSSEN, J., LIND, M. - «A Model of Human Decision Making in Complex Systems and its Use for Design of System Control Strategies » - *American Control Conference, Arlington, Juin 1982 - p. 270-276*

[RIV 93] RIVALS, I., PERSONNAZ, L., DREYFUS, G., CANAS, D. «Real-Time Control of an Autonomous Vehicle : A Neural Approach to the Path Following Problem » - *Actes de NeuroNîmes'93, Nîmes, 25-29 Octobre 1993* - Nanterre: EC2 Eds, 1993 - p. 219-229

[ROC 94] ROCHER, E., VAUCHER, G. - *Perceptron Multi-couches Etendu aux Corps des Complexes* - Rapport interne de l'Ecole Supérieure d'Electricité N°. 04038 - Rennes : SICReS Eds, 1994 - 20 p.

[RON 94] RONDEAU, L., DUBOIS, G., LEVRAT, E., LAMOTTE, M. - «Résolution d'un système d'équations de relation floue par les algorithmes génétiques » - *Actes de IPMU'94, Paris, 4-8 juillet 1994* - Vol. II - Paris : AFCET Eds, 1994 - p. 1093-1098

[RON 95] RONCO, E., GAWTHROP, P. - «Modular Neural Networks: a State of the Art » - *Rapport Interne du «Center for Systems and Control, faculty of Engineering», université de Glasgow (UK)* - Tech. Rep. #CSC-95026 (soumis à Neural Networks) - 1995 - 20 p.

[ROS 58] ROSENBLATT, F. - «The Perceptron : A Probabilistic Model for Information Storage and Organization in the Brain » - *Psychological Review* - 1958, N°. 65, p. 386-404.

[RUM 86a] RUMELHART, D.E., McCLELLAND, J.L. (Eds) - *Parallel Distributed Processing, Exploring the Micro-Structure of Cognition, Volume 1 : Foundations* - Cambridge : MIT Press, 1986 - 576 p.

[RUM 86b] RUMELHART, D.E., HINTON, G.E., WILLIAMS, R.J. - «Learning Internal Representations by Error Propagation » - *In : Rumelhart, D.E., McClelland, J.L. (Eds), Parallel Distributed Processing, Exploring the Micro-Structure of Cognition, Vol. 1 : Foundations* - Cambridge : MIT Press, 1986 - p. 318-362.

[SAE 90] SAERENS, M., SOQUET, A. - «Neural Controller Based on Back-propagation Algorithm » - *IEE Proceedings - F* - IEE Eds, 1991 - Vol. 138 - N°. 1 - p. 55-62

[SAG 90] SAGA, K., SUGASAKA, T., NAGATA, S. - «Adaptive Behavior Control for a Robot by Neural Networks Using Self-Supervised Learning » - *Actes de NeuroNîmes'90, Nîmes, 12-16 Novembre 1990* - Nanterre : EC2 Eds, 1990 - p. 433-441.

[SCH 91] SCHALCHLI, L. - «Un Système Nerveux très Plastique » - *La Recherche* - 1991, Vol. 22, N°. 238 - p. 1506-1507

[SCH 93] SCHOENAUER, M., RONALD, E., DAMOUR, S. - «Evolving Nets for Control » - *Actes de NeuroNîmes'93, Nîmes, 25-29 Octobre 1993* - Nanterre : EC2 Eds, 1993 - p. 271-278

[SCH 95] *Sciences humaines* - numéro spécial «l'acteur et ses logiques » - 1995, N°. 9 - 49 p.

[SEH 94] SEHAD, S., TOUZET, C. - «Apprentissage par Renforcement pour l'Acquisition de Comportements en Robotique » - *Actes des journées NSI'94, Chamonix, 2-5 Mai 1994* - p. 67-70

[SEL 83] SELVERSTON, A.I., MILLER, J.P., WADEPUHL, M. - «Neural Mechanisms for Production of Cyclic Motor Patterns » - *IEEE Transactions on Systems, Man, and Cybernetics* - 1983, Vol. SMC-13, N°. 5 - Los Alamitos: IEEE Press Eds, 1983 - p. 749-757

[SHA 50] SHANNON, C.E. - «A Chess-Playing Machine » - *Scientific American* - 1950, Vol. 182 - p. 48-51

[SHA 92] SHATZ, C. - «La Maturation du Cerveau » - *Pour la Science* - 1992, N°. 181 - p. 50-59.

[SIM 92] SIMON, N., CORPORAAL, H., KERCKHOFFS, E. - «Variations on the Cascade-Correlation Learning Architecture for Fast Convergence in Robot Control » - *Actes de Neuro-Nîmes'92, Nîmes, 2-6 Novembre 1992* - Nanterre: EC2 Eds, 1992 - p. 455-464

[SIN 93] SINGER, W. - «Synchronization of Cortical Activity and its Putative Role in Information Processing and Learning » - *Annual Review of Physiology* - 1993, Vol. 55 - p. 349-374.

[SPI 92] SPIESSENS, P., TORREELE, J. - «Massively Parallel Evolution of Recurrent Networks: An Approach to Temporal Processing » - *In : Varela, F.J., Bourgine, P. (Eds), Toward a Practice of Autonomous Systems* - Cambridge: MIT Press, 1992 - p. 70-77

[STE 88] STEVENS, C. - «Le neurone » - *In : Hubel, P. et al. (Eds.)*, *Le cerveau* - Paris : Belin, 1988 - p.17-28

[SUT 88] SUTTON, R. - «Learning to Predict by the Method of Temporal Differences » - *Machine Learning* - 1988, Vol. 3 - Boston : Kluwer Academic Publishers, 1988 - p. 9-44

[THO 88a] THORPE, C., HEBERT, M.H., KANADE, T., SHAFER, S.A. - «Vision and Navigation for the Carnegie Mellon Navlab » - *IEEE Transactions on Pattern Analysis and Machine Intelligence* - 1988, Vol. 10, N°. 3 - 11 p.

[THO 88b] THORPE, S.J. - «Traitement d'Images chez l'Homme » - *Technique et Science Informatiques* - 1988, Vol. 27, N°. 6 - p. 517-525

[THO 94] THORPE, S.J., GAUTRAIS, J., HOLZHEY, C. - «Traitement dans le Système Visuel avec une seule Impulsion par Neurone » - *Actes des journées Neurosciences et Sciences de l'Ingénieur, Chamonix, 2-5 Mai 1994* - p. 11-13

[THU 91] THURN, S.B., LINDEN, A., MÖLLER, K. - «Adaptive Look-ahead Planning » - *Actes des journées NeuroNîmes, Nîmes, 12-16 Novembre 1990* - Nanterre : EC2 Eds, 1990 - p. 405-417

[TRO 92a] TROTTER, Y., BURNOD, Y. - «La Représentation Corticale de l'Espace 3D » - *Le Courrier du CNRS* - 1992, N°. 79 - p. 49

[TRO 92b] TROTTER, Y., CELEBRINI, S., STRICANNE, B., THORPE, S., IMBERT, M. - «Modulation of Neural Stereoscopic Processing in Primate Area V1 by Viewing Distance » - *Science* - 1992, N°. 257 - p. 1279-1281

[TUN 94] TUNALI, E.T., SLUZEK, A., TAN, K. - «A Vision Guided Robot System for Picking Objects from a Moving Conveyor Belt » - *Actes de DKSME'94, Hong-Kong, 2-4 Mai* 1994 - Vol. 1 - Hong-Kong: Chinese University of Hong-Kong, 1994 - p. 210-215

[TUR 50] TURING, A.M. - «Computing Machinery and Intelligence » - *Mind* - 1950, Vol. 59, N°. 235

[TUR 88] TURK, M.A., MORGENTHALER, D.G., GREMBAN, K.D., MARRA, M. - «VITS - A Vision System for Autonomous Land Vehicle Navigation » - *IEEE Transactions on Pattern Analysis and Machine Intelligence* - 1988, Vol. 10, N°. 3 - p. 342-361

[WAT 92] WATKINS, C.J.C.H., DAYAN, P. - «Q-Learning » - *Machine Learning* - 1992, Vol. 8 - Boston: Kluwer Academic Publishers, 1992 - p. 279-292

[WEI 92] WEIβ, W. - «Action Selection and Learning in Multi-Agent Environnements » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge: MIT Press, 1993 - p. 451.459

[WEI 95] WEINBERG, A. - «Les Jeux de l'Ordre et du Désordre » - *Sciences Humaines* - 1995, N°. 32 - p. 16-19

[WER 74] WERBOS, P.J. - *Beyong Regression : New Tools for Prediction and analysis in the Behavioral Sciences* - Thèse de PhD - Harvard University, 1974

[WER 87a] WERBOS, P.J. - «Learning How the World Works : Specifications for a Predictive Networks in Robots and Brains » - *Actes de IEEE International Conference on Systems, Man and Cybernetics, 1987* - New-York : IEEE Eds - p. 302-310.

[WER 87b] WERBOS, P.J. - «Building and Understanding Adaptive Systems : A Statistical/Numerical Approach to Factory Automation and Brain Research » - *IEEE Transactions on Systems, Man and Cybernetics* -1987, Vol. SMC-17, N°. 1 - p. 7-20.

[WER 89] WERBOS, P.J. - «Backpropagation and Neurocontrol: A Review and Prospectus » - *Actes de IJCNN'89* - Vol. 1 - Los Alamitos: IEEE Press Eds, 1989 - p. 209-216

[WER 90] WERBOS, P.J. - «Backpropagation Through Time: What It Does and How to Do It » - *Proceedings of the IEEE* - 1990, Vol. 78, N°. 10 - Los Alamitos: IEEE Press Eds, 1990 - p. 1550-1560

[WER 92] WERBOS, P.J., PELLIONISZ, A.J. - «Neurocontrol and Neurobiology: New Developments and Connections » - *Actes de IJCNN'92, Baltimore* - Los Alamitos: IEEE Press Eds., 1992 - p. 373-378

[WID 60] WIDROW, B., HOFF, M.E. - «Adaptative Switching Circuits » - *Actes de l'IRE WESCON Convention* - 4ème partie, 1960, p. 96-104

[WID 73] WIDROW, B., GUPTA, N.K., MAITRA, S. - «Punish/Reward: Learning with a Critic in Adaptive Threshold Systems » - *IEEE Transactions on Systems, Man, and Cybernetics* - 1973, Vol. SMC-3, N°. 5 - Los Alamitos: IEEE Press Eds, 1973 - p. 455-465

[WIE 48] WIENER, N. - *Cybernetics: or Control and Communication in the Animal and Machine* - Cambridge: MIT Press, 1961 (2° éd.)

[WIL 89] WILLIAMS, R.J., ZIPSER, D. - «A Learning Algorithm for Continually Running Fully Recurrent Neural Networks » - *Neural Computation* - 1989, Vol. 1 - Cambridge : MIT Press, 1989 - p. 270-280

[YAN 92] YANCO, H., STEIN, L.A. - «An Adaptative Communication Protocol for Cooperating Mobile Robots » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge: MIT Press, 1993 - p. 478-485

[ZAP 92] ZAPATA, R., LEPINAY, R., NOVALES, C., DEPLANQUES, P. - «Reactive Behaviors of Fast Mobile Robots in Unstructured Environments: Sensor-Based Control and Neural Networks » - *Actes de From Animals to Animats: Second Conference on Simulation of Adaptive Behavior, Hawaii, 1992* - Cambridge: MIT Press, 1993 - p. 108-115

[ZEL 94] ZELL, A., MAMIER, G., VOGT, M., MACHE, N., HÜBNER, R., HERRMANN, K.U., SOYEZ, T., SCHMALZL, M., SOMMER, T., HATZIGEORGIOU, A., DÖRING, S., POSSELT, D. - *SNNS User Manual Version 3.2* - Rapport interne de l'Université de Stuttgart N°. 3/94 - Stuttgart : Institute for Parallel and Distributed High Performance Systems, 1994 - 275 p.

ANNEXES :

Annexe A1 **:** *Résultats de la simulation du système de transport*

Annexe A2 **:** *Les principaux modèles connexionnistes*

Annexe A3 **:** *Apprentissage du OU Exclusif*

Annexe A4 **:** *Classification : la reconnaissance de couleurs*

Annexe A5 **:** *Apprentissage de séquences booléennes*

Annexe A6 **:** *Neurocontrôle : le pendule inverse*

ANNEXE A1 :

RESULTATS DE LA SIMULATION DU SYSTEME DE TRANSPORT

ANNEXE A1 : Résultats de la simulation du système de transport

Nous présentons ici les résultats des différentes campagnes de tests menées sur la simulation (voir section II.3.5). Chaque tableau donne le nombre de pièces produites dans l'atelier en 20 000 unités de temps (UT) en fonction de différents paramètres. Pour chaque valeur des paramètres, dix essais ont été effectués. Nous en donnons ici la valeur moyenne, l'écart type ainsi que les valeurs maximales et minimales. Pour une analyse de ces résultats on se rapportera à la section II.3.5.

A1.1. Choix de la loi de gestion

Deux campagnes de tests ont été effectuées afin de déterminer la loi d'échange la plus performante. L'atelier comporte 5 robots et le temps de process maximum varie entre 100 et 700 UT. La productivité de l'atelier est mesurée par le nombre total de pièces produites (c'est-à-dire sorties de l'atelier).

A1.1.1. Loi d'échange

Temps Process Maximum	100	200	300	400	500	600	700
Moyenne	86.4	71.5	62.2	53.4	48.6	37.7	40.6
Ecart Type	3.6	4.7	4.4	6.48	7.6	5.56	7.2
Max.	94	81	71	62	61	49	57
Min.	80	63	56	41	31	30	28

A1.1.2. Loi «Request For Service» (RFS-Rule)

Temps Process Maximum	100	200	300	400	500	600	700
Moyenne	70.7	68.8	59.6	57.3	44.9	43.6	39.7
Ecart Type	6.42	4	3.2	3.1	5.66	5.2	7.1
Max.	81	80	65	63	58	54	52
Min.	52	64	54	51	30	34	18

A1.2. Influence du nombre de robots

Le temps de process maximum est fixé à 100 UT et le nombre de chariots varie entre 1 et 13. La loi de gestion utilisée est la loi d'échange. La productivité est donnée par le nombre de pièces produites.

Nombre de Robots	1	2	3	4	5	6	7
Moyenne	19.1	36.7	54.8	70.9	84.4	98.9	101.2
Ecart Type	2.12	2.3	5.6	5.32	3.6	4.3	4.24
Max.	15	44	66	78	92	111	110
Min.	23	34	46	61	78	87	94

Nombre de Robots	8	9	10	11	12	13
Moyenne	109	111.3	111.5	115.1	99.1	98.4
Ecart Type	6.6	2.64	15.8	7.5	13.26	12.12
Max.	118	118	124	127	125	122
Min.	95	106	38	104	69	68

A1.3. Influences relatives du nombre de robots et du temps de process maximum

Pour 1, 3, 5, 7, 9 et 11 robots, la production a été mesurée (par le nombre de pièces produites) pour chacun des temps de process maximum. La loi de gestion utilisée est la loi d'échange.

A1.3.1. Un robot

Temps Process Maximum	100	200	300	400	500	600	700
Moyenne	19.1	16.7	16.4	16.8	14.4	14.4	14.5
Ecart Type	2.12	2.16	3.48	2.4	2.68	2	3.1
Max.	15	21	23	24	20	19	20
Min.	23	13	10	12	11	10	10

A1.3.2. Trois robots

Temps Process Maximum	100	200	300	400	500	600	700
Moyenne	54.8	51.7	46.3	41.8	36	34.5	31.2
Ecart Type	5.6	4.44	4.3	3.4	7	2.1	5.04
Max.	66	64	52	48	46	40	47
Min.	46	44	38	36	23	31	19

A1.3.3. Cinq robots

Temps Process Maximum	100	200	300	400	500	600	700
Moyenne	86.4	71.5	62.2	53.4	48.6	37.7	40.6
Ecart Type	3.6	4.7	4.4	6.48	7.6	5.56	7.2
Max.	94	81	71	62	61	49	57
Min.	80	63	56	41	31	30	28

A1.3.4. Sept robots

Temps Process Maximum	100	200	300	400	500	600	700
Moyenne	101.2	94.7	74.6	69.7	55.9	49.1	40.7
Ecart Type	4.24	4.7	5.08	7.7	5.12	6.92	4.24
Max.	110	101	82	85	68	65	48
Min.	94	85	64	55	44	39	35

A1.3.5. Neuf robots

Temps Process Maximum	100	200	300	400	500	600	700
Moyenne	111.3	106.9	79.8	71.6	57.5	51.2	45.8
Ecart Type	2.64	5.88	7.04	12.08	4.6	6	3
Max.	118	120	102	89	64	64	54
Min.	106	95	73	39	46	36	39

A1.3.6. Onze robots

Temps Process Maximum	100	200	300	400	500	600	700
Moyenne	115.1	100.7	88.6	70.4	58.7	48.2	46.8
Ecart Type	7.5	5.16	8.28	6.8	9.1	5.8	11.2
Max.	127	109	104	80	75	59	69
Min.	104	86	71	57	43	36	24

A1.4. Production d'un atelier théorique chargé à 100%

Pour calculer le rendement du système de transport, nous avons comparé la productivité de l'atelier, pour chaque configuration, avec un atelier théorique fonctionnant en flux tendu. Dans cet atelier chaque pièce passe systématiquement par trois usinages, chacun d'eux durant «*Temps de process max. / 2*» (un usinage par machine). La productivité de l'atelier est alors égale à celle de chacune des machines, soit, en 20 000 unités de temps :

*20 000*2 / Temps de process max.*

soit :

Temps de Process Maximum	100	200	300	400	500	600	600
Productivité Théorique	400	200	133.33	100	80	66.66	57.14

ANNEXE A2 :

LES PRICIPAUX MODELES CONNEXIONNISTES

ANNEXE A2 : Les principaux Modèles Connexionnistes

A partir des principes connexionnistes généraux (énoncés chapitre IV, section 3), de nombreux modèles de réseaux neuromimétiques ont été développés. Dans cette annexe, nous énumérons les plus marquants dans l'ordre chronologique de leur première utilisation.

A2.1. Les réseaux «primitifs»

Les premiers modèles de réseaux de neurones sont nés d'un compromis entre les principes théoriques énoncés par Mc Culloch et Pitts et les capacités des machines et des algorithmes de l'époque. Les premières architectures utilisées étaient donc monocellulaires, basées sur une cellule booléenne dont la fonction de décision était la fonction seuil. Il s'agit du Perceptron [ROS 58] et de l'ADALINE (pour «ADAptive LInear NEtwork») [WID 60] qui diffèrent essentiellement par leur mode d'apprentissage (supervisé dans les deux cas). Ces deux réseaux ont été conçus pour des tâches de classification : la réponse de l'unique neurone , fonction des pondérations associées à chacun des paramètres d'entrée détermine l'appartenance à l'une ou l'autre des classes. L'utilisation en parallèle de plusieurs de ces réseaux permet d'étendre la méthode à des problèmes de séparation entre plus de deux classes.

L'algorithme d'apprentissage utilisé pour le Perceptron est très simple : en cas d'erreur en sortie le vecteur d'entrée correspondant est ajouté ou retranché au vecteur des poids synaptiques.

L'ADALINE, par contre, utilise une règle plus complète car l'erreur est calculée non plus à partir de l'état binaire de la cellule mais à partir de la stimulation que la cellule aurait due recevoir en entrée (figure A2-1). De plus cette méthode (dite « Apprentissage de Widrow-Hoff » ou « règle Δ ») permet une réduction du gradient d'erreur à partir de la méthode des moindres carrés.

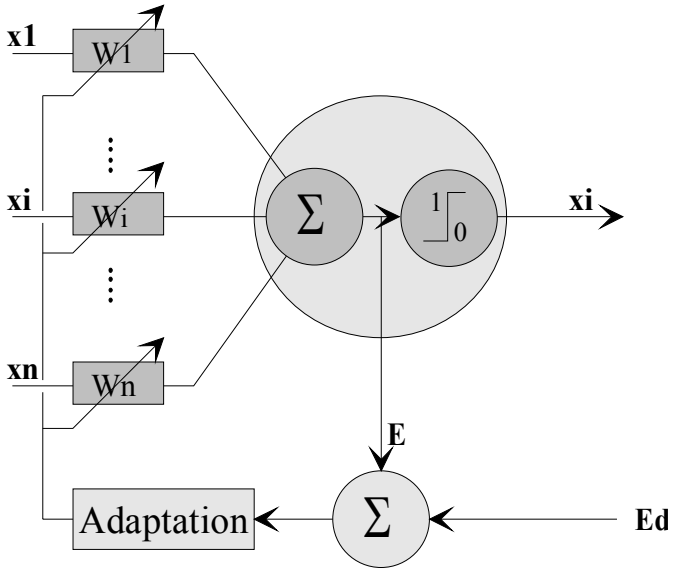


Figure A2-1 : *L'ADALINE de Widrow-Hoff*

E : Somme pondérée des entrées

Ed : Valeur désirée pour E

La règle d'apprentissage utilisée pour l'ADALINE est la suivante (A2-1) :

Si on a : $E = \sum_{i=1}^{n} W_i x_i$

et si E_d est la valeur analogique désirée pour E, alors on a, en dérivant le gradient d'erreur par rapport à W :

$$\Delta W_i = -\alpha \nabla C = \alpha (E_d - E) X_i \qquad \textbf{(A2-1)}$$

où ∇C est le gradient de l'erreur quadratique $C = \frac{1}{2}(E - E_d)^2$

La principale limitation de ces deux réseaux provient directement de leur structure : en 1968 Minsky et Paperts ont montré, dans le livre «Perceptron» [MIN 68], que les modèles de réseaux à une couche de connexions et à fonction d'activation linéaire sont incapables de distinguer deux classes si celles-ci ne sont pas linéairement séparables. Devant les problèmes de dynamique posés par les modèles récurrents et, en l'absence d'algorithme d'apprentissage adaptés aux réseaux multicouches, cette contrainte a plongé les réseaux de neurones dans l'oubli pendant près de vingt ans, jusqu'à l'exploitation pratique des modèles récurrents puis, surtout, jusqu'à l'utilisation de l'algorithme de la rétropropagation du gradient pour l'apprentissage dans les réseaux en couches.

A2.2. Les mémoires associatives

Au début des années soixante-dix, on a cherché à utiliser les réseaux de neurones non plus comme classificateurs mais comme mémoires auto- ou hétéro-associatives. Cette approche a conduit plusieurs chercheurs, dont S. Amari et T. Kohonen, à construire des réseaux à deux couches reliées par des connexions symétriques (figure A2-2)[AMA 72] [KOH 77]. Cette architecture permet au réseau de mettre en correspondance les images d'entrée (c'est-à-dire l'ensemble des paramètres d'entrée) et les images de sortie. Dans le cas des mémoires auto-associatives, ces images sont identiques mais l'image d'entrée peut être incomplète ou bruitée. La mise en correspondance est effectuée en utilisant des relations numériques (les poids) entre les différents paramètres composant ces images.

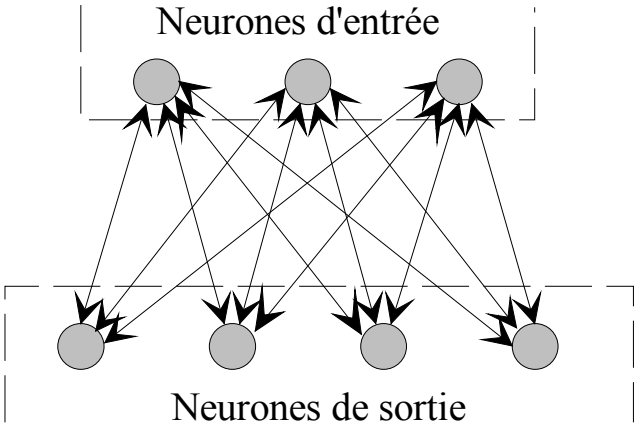


Figure A2-2 : *Exemple de mémoire associative*

Les mémoires associatives présentent une structure relativement simple qui a permis d'importantes avancées dans la théorie mathématique des réseaux connexionnistes [KAM 90]. Leur convergence en particulier a pu être vérifiée (les mémoires associatives correspondent en effet aux critères du théorème de Cohen-Grossberg-Kosko) et les algorithmes d'apprentissage utilisés sont désormais bien connus.

Deux approches différentes peuvent être utilisées pour l'entraînement des mémoires associatives : les poids peuvent être calculés analytiquement à partir de la base d'apprentissage ou appris grâce à une utilisation supervisée de la règle de Hebb . Dans le second cas, les images d'entrée et de sortie sont présentées simultanément au réseau et les poids sont renforcés en fonction des corrélations d'activité. Malheureusement la convergence de cet apprentissage reste soumise à une contrainte d'orthogonalité entre les différentes entrées.

De plus, les mémoires associatives ont été à l'origine de nombreuses passerelles entre les réseaux de neurones et d'autres systèmes complexes, plus particulièrement la physique des particules (verres de spin) et la thermodynamique (approche énergétique, recuit simulé).

A2.3. Le modèle de Hopfield et la machine de Boltzmann

L'analogie entre les réseaux de neurones récurrents et la physique des particules, utilisée initialement sur les mémoires associatives, a été plus concrètement utilisée par John Hopfield pour définir en 1982 un modèle de réseau symétrique totalement connecté, à l'exception des autoconnexions [HOP 82]. Ce réseau est construit à partir d'un maillage de neurones binaires, les poids des connexions pouvant, comme pour le modèle précédent, être calculés analytiquement (en particulier à partir de contraintes [HOP 85]) où appris par une règle hebbienne. L'analogie avec les systèmes physiques permet de définir une énergie globale pour le réseau (somme des contributions locales de chacune des cellules). Hopfield a montré que l'énergie (qui est une grandeur positive) du réseau est toujours diminuée lorsqu'un neurone change d'état suivant sa fonction de transfert. La fonction d'énergie utilisée étant une fonction de Lyapunov, on en déduit que le réseau converge en un nombre fini d'itérations vers un état stable correspondant à un minimum d'énergie. On notera que ce type de réseau, dont l'espace d'état est fini dénombrable, ne peut pas présenter d'attracteurs chaotiques.

Le réseau de Hopfield a été utilisé pour une grande variété de problèmes mais les applications les plus fréquentes sont l'optimisation multicritère (en particulier pour les problèmes NP-Complets tels que le voyageur de commerce) et l'utilisation comme mémoire adressable par le contenu.

Dans le premier cas, chaque cellule représente un élément de réponse (pour le voyageur de commerce, la date à laquelle il traverse une ville) ; les poids sont calculés analytiquement en fonction des contraintes du problème et des états illégaux, considérés comme des contraintes particulières. Le réseau est alors soumis à une excitation qui le conduit à se stabiliser dans un minimum local d'énergie, correspondant à une solution admissible.

L'utilisation du réseau comme mémoire adressable par le contenu, au contraire, fait appel à un processus d'apprentissage hebbien qui permet de renforcer les corrélations entre les différents éléments composant «l'image» mémorisée. L'utilisation du modèle de Hopfield est cependant limitée par de nombreuses imperfections : la loi de Hebb entraîne des contraintes d'orthogonalité entre les différentes «images» apprises, sous peine de créer des minima locaux ne correspondant pas aux états appris. D'autre part, les capacités de mémorisation du modèle de

Hopfield (c'est-à-dire le rapport nombre d'images apprises sur nombre de cellules) sont assez médiocres, surtout lorsque le nombre de cellules augmente. Enfin ce modèle demande d'importants moyens de calcul, lors de l'apprentissage puis pour son utilisation. Il reste donc limité à des problèmes restreints en nombre de contraintes ou en nombre de motifs à mémoriser.

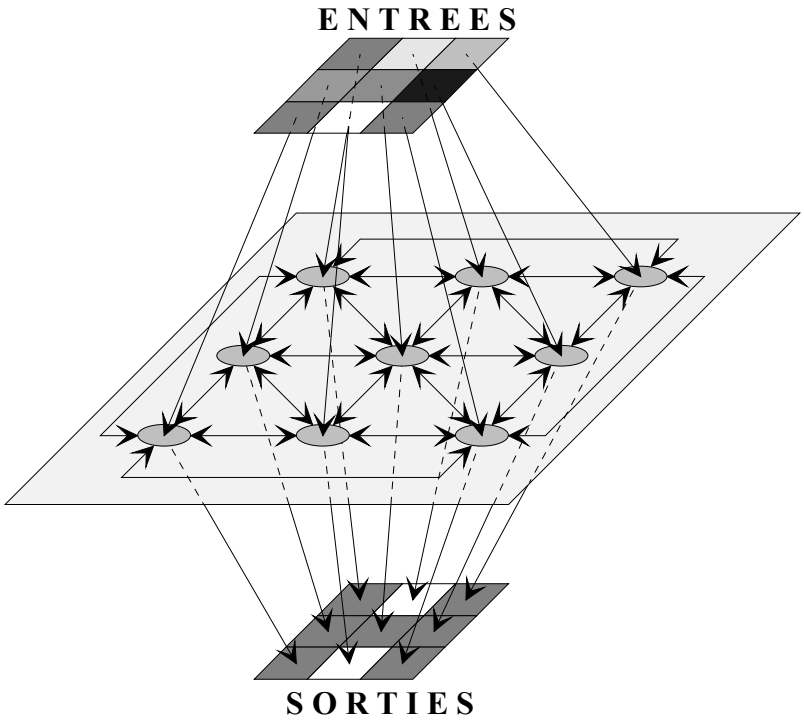


Figure A2-3 : *Le modèle de Hopfield utilisé comme mémoire adressable par le contenu.*

Pour des raisons de commodités de lecture, toutes les connexions n'ont pas été représentées sur la figure

De nombreuses variantes ou extensions de ce modèle ont été proposées dans la littérature. On citera en particulier les réseaux d'ordre supérieur (qui utilisent des corrélations d'activités entre trois neurones ou plus suivant l'ordre d'utilisé), l'extension aux cellules à valeurs réelles [COH 83], le calcul en temps continu [HOP 84], et ,surtout, l'introduction des principes du recuit simulé [KIR 83] dans les réseaux de neurones. Pour cela on ajoute, à la fonction de décision des neurones, un processus stochastique soumis à un paramètre global (la température). Des accroissements locaux d'énergie sont alors possibles ce qui permet au réseau d'éviter les minima locaux d'énergie et donc de converger vers de meilleures solutions que dans la version initiale de l'algorithme.

L'union de l'architecture du réseau de Hopfield et des principes du recuit simulé a permis à Hinton et Sejnowski [HIN 86] de proposer un modèle de réseau : la machine de Boltzmann. Comme le réseau de Hopfield, la machine de Boltzmann est un réseau totalement connecté symétrique mais dont les unités ont une fonction de transfert stochastique fonction d'un paramètre de *température T :* lorsque la température est nulle les fonctions de transfert sont totalement déterministes : l'influence du facteur stochastique croît avec la température. Lors de l'utilisation du réseau, la température est initialement fixée à un niveau élevé (ce qui permet au réseau d'éviter les minima locaux) puis décroît pour permettre au réseau de converger vers une solution stable.

La machine de Boltzmann utilise un algorithme d'apprentissage supervisé inspiré de la règle de Hebb (renforcement des corrélations d'activité) et de la théorie de l'information (différence entre l'information présente dans le réseau et l'information souhaitée, l'information étant représentée par les corrélations d'activité entre les neurones du réseau). Deux phases sont nécessaires à l'apprentissage : dans un premier temps les neurones d'entrée et de sortie sont fixés à leur valeur désirée et on laisse le réseau se stabiliser ; dans un second temps, seul l'état des cellules d'entrée est fixé avant calcul des activations et stabilisation du réseau. Lors de chacune de ces deux phases les corrélations d'activité sont mémorisées (respectivement p_{ij}^{+} et p_{ij}^{-}) et leur comparaison permet ensuite de modifier les poids synaptiques selon l'équation suivante :

$$dW_{ij} = \frac{\left(p_{ij}^{+} - p_{ij}^{-}\right)}{T} \qquad \textbf{(A2-2)}$$

La machine de Boltzmann est un réseau relativement difficile à mettre en oeuvre, en particulier en raison des temps d'apprentissage (deux phases de relaxation par itération avec mémorisation de toutes les corrélations d'activité) mais l'apport du recuit simulé permet de limiter l'influence des minima locaux ce qui le rend particulièrement intéressant comme mémoire auto-associative et pour des applications d'optimisation.

A2.4. Les cartes topologiques de Kohonen

Alors que les modèles précédents, bien que d'inspiration neurobiologique, étaient fortement influencés par l'analogie avec la physique des particules, Tuevo Kohonen a proposé une approche directement issue des recherches sur le système nerveux. Les cartes topologiques [KOH 84] utilisent, en effet, plusieurs structures caractéristiques des systèmes nerveux biologiques, en particulier des aires visuelles. Comme celles-ci, le modèle de Kohonen permet d'associer par auto-organisation, différentes «images» (au sens large, c'est-à-dire des ensembles de paramètres topologiquement interdépendants) en fonction de leurs similarités. Cette capacité leur permet de résoudre assez simplement des problèmes de reconnaissance de forme (en isolant les motifs similaires) ou d'optimisation en superposant deux espaces topologiques différents (par exemple, pour le voyageur de commerce, l'espace des coordonnées cartésiennes des villes traversées et la carte topologique ordonnant les villes).

Le principe utilisé par Kohonen pour la définition de son réseau est relativement simple : les neurones (binaires) sont regroupés en deux couches (neurones d'entrée et de sortie). Les seules connexions utilisées sont celles reliant la couche d'entrée à la couche de sortie mais des contraintes topologiques sont introduites entre les différentes cellules de sortie, d'où la notion de carte (figure A2-4). Ces contraintes peuvent être considérées comme des connexions de valeur fixe entre les cellules de la couche de sortie mais elles sont généralement intégrées, lors de l'apprentissage, dans les connexions reliant les deux couches du réseau. Leur valeur est déduite de la distance séparant les neurones grâce à une fonction dite «du chapeau mexicain» (figure
A2-5).

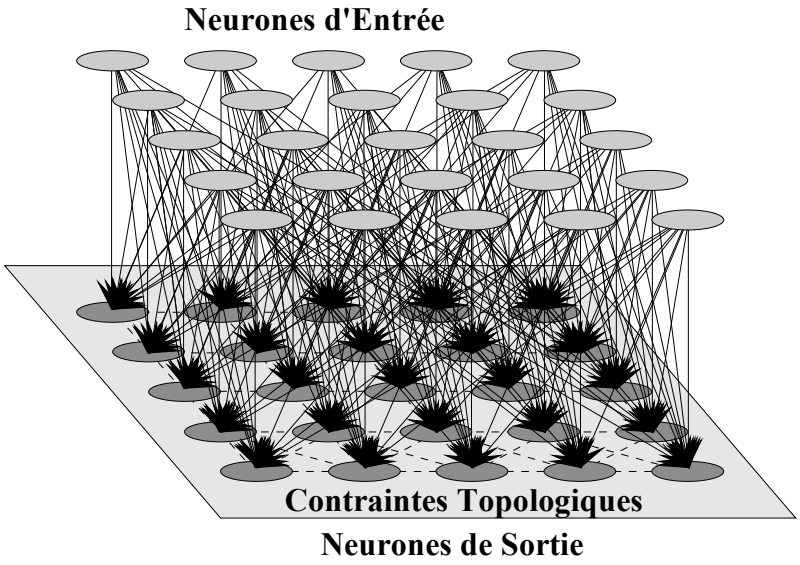


Figure A2-4 : *connectivité dans les cartes topologiques de Kohonen*

Chaque cellule de la couche supérieure est reliée à l'ensemble des cellules de la couche inférieure

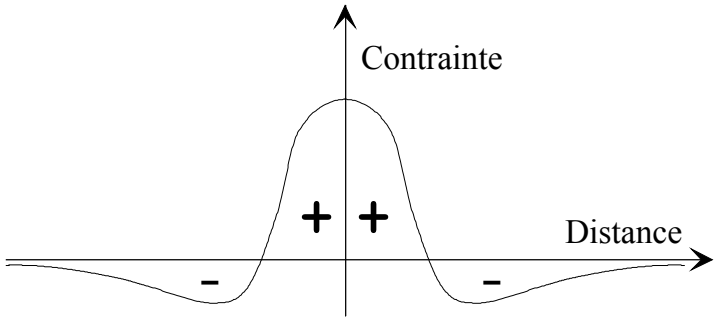


Figure A2-5 : *La fonction du «chapeau mexicain»*

Le principe de fonctionnement de ce réseau, bien que très simple, sort du cadre général énoncé ci-dessus. Ici, les pondérations des connexions aboutissant à un neurone représentent l'ensemble des valeurs d'entrée provoquant l'activation de ce neurone (auxquelles s'ajoutent les contraintes topologiques latérales afin de regrouper les entrées similaires et de dissocier les entrées appartenant à des classes différentes). On en déduit la loi d'activation des neurones puisqu'il suffit de rechercher les similitudes entre le vecteur d'entrée $\vec{e}$ et les vecteurs «poids» $\vec{W}_i$ de chacun des neurones i (A2-3),

$$x_i = \sum_{k=1}^{m} W_{ik} e_k \quad \text{, avec } m \text{ la dimension du vecteur d'entrée} \qquad \textbf{(A2-3)}$$

L'algorithme d'apprentissage de ce réseau est alors obtenu très simplement : on renforce la réceptivité du neurone «vainqueur» et des neurones voisins au vecteur d'entrée correspondant, (A2-4) :

$$W_{ik}(t+1) = W_{ik}(t) + \alpha(t)\left[e_i - W_{ik}(t)\right] \qquad \textbf{(A2-4)}$$

Avec $\alpha(t)$ le taux de correction au temps t .

Les neurones voisins du neurone i voient les pondérations de leurs connexions entrantes modifiées suivant ΔW_{ik} et $F(d_{ij})$ où $F()$ est une fonction de la distance entre les neurones, généralement la fonction du «chapeau mexicain» présentée figure A2-5 (d_{ij} est la distance euclidienne séparant les deux neurones). Le taux de correction appliqué au réseau diminue progressivement au cours de l'apprentissage pour passer d'une valeur «forte» (supérieure à 0.5 pour permettre de séparer les zones de réponse) à une valeur de l'ordre du centième permettant d'affiner l'apprentissage sans remettre en question les phases précédentes. Ce type d'apprentissage nécessite donc de présenter chaque entrée plusieurs fois afin de localiser la zone de réponse associée (classification par auto-organisation) puis d'organiser l'espace autour des différentes zones de réponse (recouvrements, exclusions...). L'apprentissage est donc souvent très lent (plusieurs milliers d'itérations).

A2.5. Les cartes cognitives de Bart Kosko

Les cartes cognitives ont déjà été présentées comme un modèle symbolico-connexionniste (voir chapitre IV, section 3.4.1). En effet, ce type de réseau manipule directement l'information symbolique codée localement dans la structure : à chaque cellule est associée un concept et les connexions représentent les liens de cause à effet identifiés entre ces concepts. Les réseaux correspondants peuvent donc présenter des structures très variables, le plus souvent récurrentes (sans autoconnexion) et non symétriques (figure A2-6). La carte cognitive permet donc de coder un ensemble de connaissances sous une forme distribuée puis, en utilisant les principes connexionnistes d'activation, d'exploiter ce codage pour valider ou compléter une information incomplète ou incertaine.

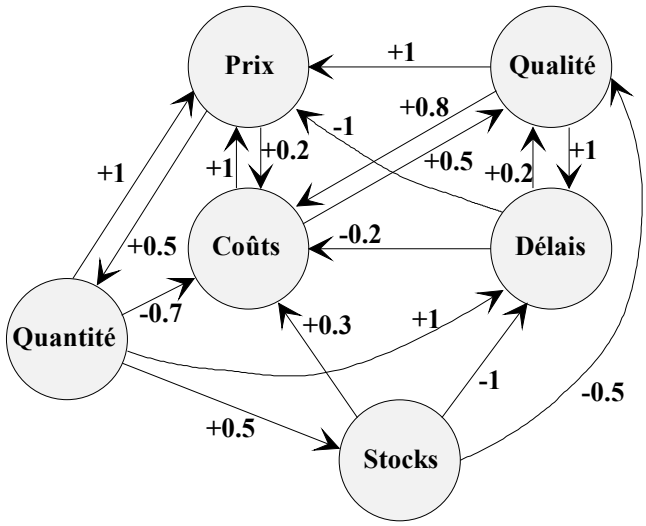


Figure A2-6 : *Exemple de carte cognitive floue*

Parce qu'elles manipulent directement des concepts et des règles explicites, les cartes cognitives sont souvent qualifiées de «système expert neuronal». Cependant cette comparaison doit être rejetée pour deux raisons principales : d'une part l'activation des concepts et les liens de cause à effet sont représentés par des valeurs numériques[92] (d'où l'appellation anglaise de «fuzzy cognitive maps», *cartes cognitives floues*) et d'autre part, alors que les moteurs d'inférence utilisés par les systèmes experts utilisent des heuristiques de parcours d'arbre, les cartes

[92] Les premiers modèles de cartes cognitives utilisaient des connexions signées (activation ou inhibition) mais non pondérées ce qui les rapproche des règles booléennes utilisées dans les systèmes experts.

cognitives utilisent les modes d'activation classiques des réseaux de neurones. Or, en raison de leur structure, les cartes cognitives présentent souvent des cycles limités ou des états oscillatoires qui interdisent l'interprétation d'une partie au moins des concepts décrits. Les cartes cognitives présentent malgré cela de nombreux intérêts, en particulier parce qu'elles permettent d'exploiter directement des bases de connaissances et de modifier, par une loi d'adaptation dérivée de la loi de Hebb, les relations entre les différents concepts identifiés. De plus les cartes cognitives, comme les mémoires associatives [BIE 90a] [BIE 90b] [COC 88], ont été utilisées comme modèle pour différents algorithmes de construction épigénétiques (ajout/retrait de cellules au cours de l'apprentissage [BIE 92a]) et permettent aux algorithmes connexionnistes de faire leurs premiers pas vers une exploitation pratique des états attracteurs cycliques ou chaotiques.

A2.6. Les réseaux en couches et la rétropropagation du gradient

Après le coup d'arrêt marqué en 1968 par le livre Minsky et Pappert, après une redécouverte partielle par J. Hopfield, T. Kohonen et quelques autres, c'est en 1986 que David Rumelhart, Geoffrey Hinton et Ronald William, dans un autre livre «*Parallel Distributed Processing : exploration in the microstructures of cognition*» [RUM 86a], ont offert au neuromimétisme une nouvelle jeunesse. Ce livre, issu des travaux de l'ensemble du «Parallel Distributed Processing research group», présente, en effet, un important travail théorique autour des architectures connexionnistes. David Rumelhart, en particulier, y propose, suivant les travaux de Paul J. Werbos [WER 74] puis de Yann Le Cun [LEC 85], un algorithme d'apprentissage pour les réseaux multicouches ce qui permet de dépasser les limites posées par Minsky et Papperts [RUM 86b].

Dans un réseau multicouches (souvent appelé «Perceptron Multi-Couches» ou «MLP», acronyme de « Multi-Layered Perceptron ») la propagation de l'activation de l'entrée vers les couches cachées puis vers la sortie permet de déterminer la réponse du réseau (correspondant à l'état des neurones de la couche de sortie). On peut alors déduire, par comparaison avec la réponse souhaitée, l'erreur commise par le réseau.

Alors que, dans un réseau mono-couche, l'erreur relative due à chacune des connexions peut être directement calculée (par exemple en utilisant la règle de Widrow-Hoff), dans un réseau multi-couches le problème principal est de répartir l'erreur sur l'ensemble des connexions sachant que chaque élément du vecteur d'entrée peut influencer les sorties par une multitude de chemins de longueur deux ou plus selon le nombre de couches cachées. L'algorithme de la rétropropagation du gradient permet alors, après avoir exprimé l'erreur sous une forme quadratique, de calculer les dérivées partielles de cette erreur par rapport aux poids du réseau. Ce calcul est effectué progressivement, de la sortie vers l'entrée du réseau, d'où le nom de l'algorithme : la rétropropagation du gradient. Il impose donc d'utiliser des cellules à état réel, dont la fonction de transfert est continue et dérivable (ce qui est le cas, en particulier, pour les fonctions sigmoïdales et gaussiennes).

L'algorithme de rétropropagation présenté ici est généralement considéré comme le modèle «standard» en regard des nombreuses variantes existantes («Batch-Backprop», «Quickprop», «Backpercolation», ...). Il est parfois qualifié de «on-line Backpropagation» car l'algorithme d'apprentissage est appliqué après chaque présentation. Cette appellation constitue cependant un abus de langage car on distingue clairement, dans l'utilisation de la rétropropagation, la phase d'apprentissage de la phase d'exploitation. Il est appliqué ici à un réseau à trois couches (couche d'entrée, couche cachée et couche de sortie, figure A2-7), ce qui conduit à l'algorithme suivant :

RETROPROPAGATION DU GRADIENT :

Notations :

Soit un réseau à trois couches (couche d'entrée, couche cachée et couche de sortie) comptant respectivement m , n et p cellules d'état x_i , y_j et z_k (où i , j et k représentent les indices successifs des cellules des trois couches, $i = 1..m$, $j = 1..n$, $k = 1..p$). On note v_{ji} et w_{kj} les pondérations appliquées respectivement aux connexions de la couche d'entrée à la couche cachée et de la couche cachée à la couche de sortie.

On note, d'autre part, Θ_j le seuil des neurones cachés et Γ_k le seuil des neurones de sortie.

On note enfin $f(\ )$ la fonction sigmoïde : $f(x) = (1 + e^{-x})^{-1}$

Algorithme :

(1) : toutes les connexions (v_{ji} et w_{kj}) et les seuils (Θ_j et Γ_k) sont initialisées aléatoirement dans l'intervalle [-1,1].

(2) : pour chaque paire d'entrée/sortie $(\vec{e}, \vec{s})$ à apprendre, faire :

(a) : Propager l'activation des cellules d'entrée aux cellules de sortie.

Soit : $x_i = e_i$

$$y_j = f\left(\sum_{i=1}^{m} x_i v_{ji} + \Theta_i\right) \qquad \textbf{(A2-5)}$$

$$z_k = f\left(\sum_{j=1}^{n} y_j w_{kj} + \Gamma_k\right) \qquad \textbf{(A2-6)}$$

(b) : calculer l'erreur commise par le réseau sur chacune des cellules de sortie :

$$E_k = z_k(1 - z_k)(s_k - z_k) \qquad \textbf{(A2-7)}$$

(c) : calculer l'erreur commise par le réseau sur chacune des cellules cachées :

$$F_j = y_j(1 - y_j)\sum_{k=1}^{p} w_{kj} E_k \qquad \textbf{(A2-8)}$$

(d) : modifier les connexions w_{kj} et les seuils Γ_k :

$$\Delta w_{kj} = \alpha y_j E_k \qquad \textbf{(A2-9)}$$

$$\Delta \Gamma_k = \alpha E_k \qquad \textbf{(A2-10)}$$

avec α : facteur d'apprentissage de la couche de sortie ($\alpha > 0$).

(e) : modifier les connexions v_{ji} et les seuils Θ_j :

$$\Delta v_{ji} = \beta x_i F_j \qquad \textbf{(A2-11)}$$

$$\Delta \Theta_j = \beta F_j \qquad \textbf{(A2-12)}$$

avec β : facteur d'apprentissage de la couche cachée ($\beta > 0$)

(3) : répéter l'opération (2) jusqu'à ce que l'erreur résiduelle E_k soit inférieure, pour chaque neurone de sortie, au seuil de tolérance.

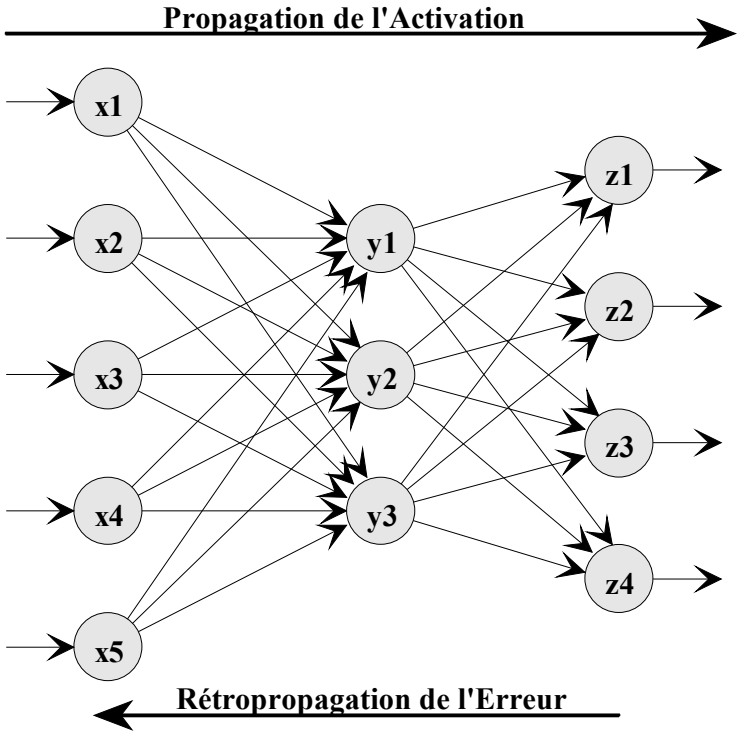


Figure A2-7 : *Exemple de réseau à trois couches*

La combinaison «réseau en couches»/»rétropropagation du gradient» est sans conteste le modèle connexionniste le plus utilisé actuellement, y compris en milieu industriel. Les raisons principales en sont probablement la fiabilité de cet algorithme (convergence de l'apprentissage et stabilité assurée) ainsi que l'ampleur des études mathématiques qui lui ont été consacrées. Il a ainsi été prouvé qu'un réseau à trois couches est capable d'approximer avec une précision quelconque (pour peu que le nombre de cellules soit suffisant) toute fonction mathématique dérivable (cette condition n'est plus nécessaire si on rajoute une deuxième couche cachée). D'autre part, des algorithmes permettant de dimensionner le réseau ou de choisir les facteurs d'apprentissage ont été mis au point. Ils permettent aux utilisateurs de s'affranchir d'une grande partie du travail de développement qui consistait en un ajustement parfois difficile de l'ensemble des paramètres.

De plus, on peut ajouter la vitesse au crédit de ce modèle. En effet, le nombre de passes nécessaire à l'obtention d'une réponse étant constant (à l'inverse des réseaux récurrents), il est possible de garantir un temps de réponse et donc, sous certaines conditions, de travailler en temps réel.

Par contre, les principes forts qui font les avantages de la rétropropagation du gradient peuvent aussi, dans certains cas, tourner à son désavantage. Ainsi la décomposition apprentissage/utilisation interdit souvent à ce modèle de travailler dans un environnement non-stationnaire sous peine de reprendre à zéro la phase d'apprentissage (dans certains cas, on peut observer un phénomène «d'oubli catastrophique» : tout ou partie de l'apprentissage est oublié). De même, la présentation des couples d'entrée/sortie, lors de l'apprentissage, doit suivre des règles très strictes et le réseau risque, si elles ne sont pas respectées, de ne pas pouvoir apprendre l'ensemble des couples.

Enfin, les restrictions évoquées chapitre IV (section 3.4.2) concernant les réseaux non-récurrents s'appliquent directement aux réseaux en couches. A moins de recevoir explicitement les entrées adéquates (c'est-à-dire une suite temporelle de paramètres), ces réseaux ne sont donc pas capables de traiter des séries temporelles et, en particulier, des suites bouclées. En d'autres termes, ils ne peuvent pas inclure la notion de contexte dans leur fonction de décision.

A2.7. Les «Radial Basis Functions»

Les «Radial Basis Functions[93]« (ou «RBF-Nets») sont, comme les réseaux que nous venons de présenter, des réseaux « feed-forward » composés de trois couches de neurones. Cependant, ils reposent sur un principe d'organisation de l'information totalement différent. C'est pourquoi nous avons choisi de les présenter séparément.

A l'inverse de la plupart des réseaux de neurones présentés dans cette annexe, les «Radial Basis Functions» ne sont pas issus de l'analogie connexionniste mais de la transposition vers les réseaux de neurones de modèles mathématiques pour l'approximation de fonctions [POG 90].

Soit N paires $(\vec{x}_i, y_i)\left(\vec{x} \in \Re^n, y \in \Re\right)$ on cherche une fonction $f(\ )$ de la forme :

$$f(\vec{x}) = \sum_{i=1}^{K} c_i h\left(\left\|\vec{x} - \vec{t}_i\right\|\right) \qquad \textbf{(A2-13)}$$

où, $h(\ )$ est la «Radial-Basis-Function» et où les $\vec{t}_i$ sont les K centres (ou Noyaux, Kernels) devant être choisis dans $\Re^n$. Les noyaux, ainsi que les coefficients c_i, sont à déterminer à partir de l'ensemble des couples $(\vec{x}_i, y_i)$ de sorte que $f(\ )$ minimise une fonction de coût composée de l'erreur quadratique et d'un terme stabilisateur destiné à lisser la fonction :

$$C[f] = \sum_{i=1}^{n} (y_i - f(\vec{x}_i))^2 + \lambda \|Pf\|^2 \qquad \textbf{(A2-14)}$$

La fonction $h(\ )$ est appliquée à la distance euclidienne entre chaque centre $\vec{t}_i$ et chaque élément $\vec{x}$. On choisit généralement une fonction ayant son maximum en zéro (distance nulle), la fonction la plus couramment utilisée est une gaussienne (équation A2-15 mais d'autre fonctions sont utilisables et plusieurs études montrent que la fonction choisie n'est pas déterminante pour les performances de l'algorithme [CHE 91].

$$h\left(\left\|\vec{x} - \vec{t}_i\right\|\right) = e^{-\theta\left\|\vec{x} - \vec{t}_i\right\|^2} \qquad \textbf{(A2-15)}$$

L'algorithme des «RBF» consiste alors à choisir les K centres dans l'espace de départ (généralement les centres sont choisis dans l'ensemble des éléments $\vec{x}$) puis à déterminer analytiquement le vecteur $\vec{c} = (c_1, \cdots, c_K)^T$ minimisant la fonction de coût (équation A2-16).

---

[93] N'ayant pas trouvé dans la littérature de traduction française, nous utiliserons, dans l'ensemble du texte, l'expression anglo-saxonne ou l'abréviation « RBF-Nets ».

Si G est une matrice (n, K) telle que :

$$G = \begin{pmatrix} h(\vec{x}_1 - \vec{t}_1) \cdots h(\vec{x}_1 - \vec{t}_K) \\ \vdots \ddots \vdots \\ h(\vec{x}_n - \vec{t}_1) \cdots h(\vec{x}_n - \vec{t}_K) \end{pmatrix}$$

et si H est une matrice (K, K) telle que :

$$H = \begin{pmatrix} h(\vec{t}_1 - \vec{t}_1) \cdots h(\vec{t}_1 - \vec{t}_K) \\ \vdots \ddots \vdots \\ h(\vec{t}_K - \vec{t}_1) \cdots h(\vec{t}_K - \vec{t}_K) \end{pmatrix}$$

Alors, si $\vec{y} = (y_1, \cdots, y_n)^T$, on a :

$$\vec{c} = \left(G^T . G + \lambda H\right)^{-1} . G^T . \vec{y} \qquad \textbf{(A2-16)}$$

Hormis le terme stabilisateur λH, l'équation (A2-16) est identique à l'inversion matricielle de Moore-Penrose utilisée en interpolation linéaire.

La fonction $f(\ )$ peut être directement représentée sous la forme d'un réseau de neurones à trois couches comptant n neurones sur la couche d'entrée, K neurones sur la couche cachée et un seul neurone de sortie. Les connexions de la couche cachée sont pondérées par les différentes composantes des vecteurs noyaux et les connexions de la couche de sortie reçoivent une pondération égale à la valeur des coefficients c_i (figure A2-8). La fonction de transfert des neurones cachés est la fonction «RBF» appliquée à la distance euclidienne entre le vecteur d'entrée et le vecteur de poids de l'unité (c'est-à-dire, pour le neurone i entre $\vec{x}$ et $\vec{t}_i$). Dans le cas d'une fonction gaussienne, le facteur θ (équation A2-15) correspond alors au seuil des cellules cachées.

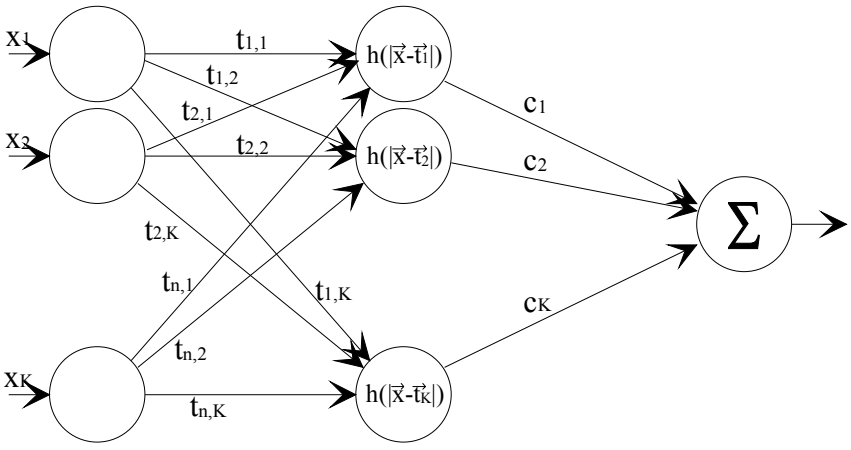


Figure A2-8 : *Architecture des «RBF-Nets»*

L'analogie avec les réseaux de neurones a conduit à l'extension des «Radial Basis Functions». La fonction de transfert de la cellule de sortie, en particulier, peut être plus complexe qu'un simple sommation des influx entrants : l'utilisation d'un seuil et d'une fonction

non-linéaire (par exemple une fonction sigmoïde) permet d'améliorer les capacités d'approximation. En outre, il est toujours possible d'augmenter le nombre de neurones sur la couche de sortie et d'approximer plusieurs fonctions à partir d'un même ensemble de noyaux.

Les «RBF-Nets» présentent de nombreux intérêts dont, en premier lieu, la possibilité de calculer analytiquement les poids à partir de la base d'apprentissage ce qui permet de s'affranchir de la phase d'entraînement du réseau. Cela suppose cependant que l'on ait préalablement placé les noyaux de façon optimale ce qui n'est pas toujours possible. Des algorithmes d'apprentissage (basés par exemple sur les moindres carrés [CHE 91]) permettent alors d'adapter la position des noyaux dans l'espace d'entrée.

Pour notre propos les «RBF-Nets» présentent un intérêt d'un autre type. En effet, dans ce type de réseau, l'information n'est pas totalement distribuée sur l'ensemble des connexions : la distribution de l'information est directement dépendante du nombre de noyaux et de leur position, chaque connexion supportant un élément d'information parfaitement identifiable. On peut donc considérer que les «RBF-Nets» constituent une forme particulière de réseau à information semi-distribuée, voire de réseau modulaire [RON 95].

A2.8. L'«Adaptive Resonance Theory»

Il n'est pas possible de clore cette annexe concernant les modèles de réseaux neuromimétiques sans évoquer le travail effectué par Stephen Grossberg et le «Center for Adaptive Systems» depuis plus de trente ans. A partir d'un réseau totalement connecté, le «Shunting Grossberg» capable, en utilisant un algorithme d'apprentissage hebbien, de classer des entrées quelconques, Grossberg et Carpenter ont développé le principe de l'«Adaptive Resonance» [GRO 76] puis, sur cette base, les réseaux «ART1» [CAR 86], «ART2» [CAR 87] et «ARTMAP» [CAR 91].

Le principe du «Shunting Grossberg» [GRO 73] est relativement simple : un réseau totalement connecté comportant des neurones à valeurs réelles présente, si les connexions inter-neurones sont inhibitrices et les autoconnexions activatrices, un comportement compétitif type «Winner-Takes-All». La particularité du réseau définit par Grossberg par rapport aux réseaux compétitifs classiques est la dissociation des entrées positives et des entrées négatives dans la fonction de décision des cellules. Les premières sont en effet additionnées à l'auto-activation des neurones tandis que les secondes s'ajoutent aux inhibitions latérales du réseau (de plus Grossberg rajoute dans cette fonction de décision un facteur de gain qui permet de limiter les effets du bruit et de normaliser automatiquement les entrées). Si $\vec{e}^{+}$ et $\vec{e}^{-}$ sont, respectivement, les entrées activatrices et inhibitrices alors l'équation dynamique des cellules du réseau est la suivante (A2-17) :

$$\Delta x_i = -\alpha x_i + (\beta - x_i)\left[f(x_i) + e_i^{+}\right] - (x_i + \mu)\left[\sum_{j=1}^{n} f(x_j) W_{ji} + e_i^{-}\right] \qquad \textbf{(A2-17)}$$

où $f(\ )$ représente la fonction sigmoïde et $(\beta - x_i)$ et $(x_i + \mu)$ représentent les facteurs de gain associés aux activations positives et négatives.

On peut prouver qu'avec cette fonction d'activation le «Shunting Grossberg» répond à l'ensemble des conditions du théorème de Cohen-Grossberg (chapitre IV, section 3.5.2). Ce type de réseau converge donc systématiquement vers des attracteurs stables.

A partir de ce modèle Stephen Grossberg a étudié le problème de la convergence des algorithmes d'apprentissage. En effet, la plupart d'entre eux ne convergent vers des valeurs fixes que parce que le facteur d'apprentissage décroît progressivement jusqu'à atteindre une valeur proche de zéro (convergence en valeur). Pour résoudre le dilemme de la «stabilité/plasticité» [GRO 91b] (trop de stabilité conduit à un apprentissage rigide tandis que trop de plasticité produit des états chaotiques), Grossberg a proposé en 1976 [GRO 76] l'«Adaptive Resonance Theory» qui différencie deux fonctions de mémorisation : l'attention qui mémorise les événements «familiers» et l'orientation qui mémorise les événements nouveaux. Ce principe a permis la création d'une famille de réseaux (ART1, ART2, ARTMAP, ...). Nous présentons ici le premier de ces réseaux, l'ART1, qui classe par auto-apprentissage un nombre quelconque de motifs binaires (l'ART2 en est une version analogique et ARTMAP une version supervisée).

L'ART1 est un réseau à deux couches : une couche d'entrée reçoit les motifs d'entrée et les transmet à une couche de sortie composée d'un réseau du type «Shunting Grossberg». Ces deux couches sont totalement interconnectées, on nomme w_{ij} les connexions reliant les cellules de la couche d'entrée aux cellules de la couche de sortie. Les connexions inverses sont notées v_{ji} , elles sont utilisées pour déterminer l'attention du réseau. Le fonctionnement de ce réseau est alors assez proche de celui du «Shunting Grossberg» simple : lorsqu'une entrée est présentée à la première couche, elle est transmise, via les connexions w_{ij} à la deuxième couche qui, en raison de sa dynamique compétitive, choisit une classe d'appartenance (c'est-à-dire une cellule). Les connexions récurrentes v_{ji} sont alors utilisées pour comparer la classe choisie avec le motif d'entrée. Si la différence est inférieure à un paramètre de vigilance alors la classification est acceptée et apprise. Sinon le réseau inhibe la classe correspondante et renouvelle le processus jusqu'à trouver une classe acceptable. Si aucune classe acceptable n'est trouvée, alors une cellule supplémentaire est ajoutée à la couche de sortie. Cette procédure permet d'ajouter une classe au réseau.

A2.9. D'autres modèles...

Nous avons essayé, au cours des sections précédentes, de fournir au lecteur un large aperçu sur les différents modèles de réseaux neuromimétiques. Devant la multiplicité des modèles connexionnistes créés au cours de la dernière décennie cet aperçu ne peut en aucun cas être considéré comme exhaustif. Nous l'espérons simplement représentatif des tendances principales des recherches actuelles sur les réseaux de neurones.

ANNEXE A3 :

APPRENTISSAGE DU OU EXCLUSIF

ANNEXE A3 : Apprentissage du OU EXclusif

Cette annexe reprend et développe les résultats présentés section VI.4.3. Ceux-ci seront, en particulier, analysés sous l'angle de la représentation des connaissances dans la matrice de poids. Nous montrerons ainsi que les réseaux utilisant le mode ordonnancé développent bien, durant la phase d'apprentissage, une représentation semi-distribuée des connaissances acquises.

L'apprentissage des opérateurs booléens élémentaires n'est utilisé que pour l'étude du comportement du réseau sur des problèmes simples. En outre, l'apprentissage de la fonction «OU EXclusif» a été testé sur différentes architectures connexionnistes. Ce cas permet, en effet, de vérifier que le réseau est capable de séparer des classes non-linéairement séparables. Nous avons testé le mode ordonnancé sur différentes fonctions booléennes (ET, OU, OU Exclusif ou XOR, ...) mais nous ne détaillons ici que les résultats obtenus pour le OU Exclusif. Les autres fonctions étant linéairement séparables, elles présentent en effet peu d'intérêt.

x	*y*	*x* xor *y*
VRAI	VRAI	FAUX
VRAI	FAUX	VRAI
FAUX	VRAI	VRAI
FAUX	FAUX	FAUX

Tableau A3-1 : *Table de vérité du OU EXclusif*

A3.1. Architecture

Topologie :

Le réseau utilisé pour l'apprentissage du OU Exclusif possède les caractéristiques suivantes :

- 4 Neurones d'entrée (deux neurones d'entrée par valeur booléenne)

- 2 Neurones de sortie (codant respectivement les valeurs «VRAI» et «FAUX»)

- entre 2 et 100 neurones d'afférence (la limitation supérieure est due aux temps de calcul induits par l'augmentation de la taille du réseau).

- Tous les poids synaptiques sont initialisés aléatoirement dans [-0.8 +0.8]

x	*y*	ENTREE	SORTIE	*x* xor *y*
VRAI	VRAI	0-1-0-1	1-0	FAUX
VRAI	FAUX	0-1-1-0	0-1	VRAI
FAUX	VRAI	1-0-0-1	0-1	VRAI
FAUX	FAUX	0-1-0-1	1-0	FAUX

Tableau A3-2 : *Table d'entrée/sortie du réseau pour le OU EXclusif*

Apprentissage :

Nous avons testé les différents algorithmes d'apprentissage présentés section VI.3.4.2, (équations VII-20 à VII-24). Le signal de renforcement est de +0.005 en cas de réponse correcte et de -0.05 en cas de réponse erronée. Ce signal est utilisé comme facteur d'apprentissage variable (α dans les équations VII-17 à VII-24) dans les algorithmes de modification des poids. Les états non répondants sont supprimés par une diminution des seuils de 0.1 en cas de stabilité précoce ou en cas de cycle bloquant. Nous n'avons pas utilisé de procédure spécifique pour les cycles bloquants en raison de la faible taille des réseaux utilisés. De fait, nous n'avons observé qu'une faible proportion de cycles bloquants comparativement aux stabilités précoces. Cependant, cette caractéristique particulière des réseaux utilisés est probablement la cause des faibles performances observées lorsque la taille du réseau augmente. Pour des réseaux de plus de 15 neurones on observe, en effet, un nombre plus important de cycles bloquants que la diminution des seuils ne permet pas de contrer efficacement.

Organisation de l'apprentissage :

Chaque session d'apprentissage comprend entre 100 et 500 couples entrée/sortie choisis aléatoirement dans la table d'entrée/sortie du réseau pour le OU EXclusif (tableau A3-2). Trois seuils d'apprentissage ont été utilisés :

- 25 couples entrée/sortie présentés et reconnus sans erreur (APP25)

- 50 couples entrée/sortie présentés et reconnus sans erreur (APP50)

- 100 couples entrée/sortie présentés et reconnus sans erreur (APP100)

L'apprentissage est arrêté lorsque le troisième seuil est atteint ou, à défaut, après 500 itérations.

A3.2. Résultats et analyse

Résultats généraux :

Dans l'ensemble, les résultats obtenus sont bons : avec les algorithmes d'apprentissage VII-20, VII-21 et VII-22, le réseau parvient à apprendre le OU EXclusif en quelques dizaines d'itérations quelle que soit la taille du réseau utilisé (2, 3, 4, 5, 8, 10, 15, 20 et 30 cellules cachées, des tests manuels ont été effectués pour 50 et 100 neurones). Cependant le temps d'apprentissage est très variable et fortement dépendant des conditions initiales, en particulier lorsque le réseau est sous-dimensionné : si le réseau compte moins de 5 neurones cachés l'apprentissage n'est pas assuré. De plus, les états non-répondants perturbent fortement l'apprentissage.

Par ailleurs, on observe que l'apprentissage n'a pas une convergence régulière : lorsque le réseau a convergé vers la bonne réponse (pour tous les cas de la table de vérité), le fait de prolonger l'apprentissage peut conduire à des erreurs de classification (l'apprentissage est dit « instable »). Cela se traduit par un temps d'apprentissage souvent plus important pour les critères APP100 et APP50 que pour le critère APP25. Dans certains cas, l'apprentissage ne s'est pas stabilisé avant la limite des 500 itérations.

Influence de l'algorithme d'apprentissage :

Nous avons utilisé le problème du OU EXclusif pour comparer les différents algorithmes d'apprentissage proposés pour le mode ordonnancé, soit :

- **Apprentissage hebbien orienté** (équation VI-20),

- **Differential Hebbian Learning orienté** (équation VI-21),

- **MRT Learning orienté** (équation VI-22),

- **MRT Learning orienté avec terme modérateur** (équation VI-23),

- **MRT Learning orienté avec terme modérateur orienté** (équation VI-24).

Seuls les trois premiers algorithmes permettent l'apprentissage du OU EXclusif (tests effectués sur des réseaux de 4 et 10 neurones) : l'utilisation d'un terme modérateur compromet généralement l'apprentissage. En outre, le terme modérateur orienté amplifie l'instabilité lors de l'apprentissage. Ce comportement est un effet indirect de l'utilisation d'un terme modérateur. En effet, celui-ci provoque la normalisation des poids entre +1 et -1 et toutes les connexions d'un chemin d'activation convergent asymptotiquement vers l'une ou l'autre de ces limites. La normalisation supprime donc progressivement, au cours de l'apprentissage, les différences de pondération existant entre les connexions. Or ces différences de pondération sont importantes, en particulier lorsque des neurones appartiennent à plusieurs chemins différents (figure A3-1). Les chemins appris sont donc régulièrement remis en cause lors de leur normalisation ce qui génère des instabilités interdisant l'apprentissage. Dans le cas de la modération orientée, ces instabilités sont particulièrement fortes car elles sont amplifiées par le terme modérateur.

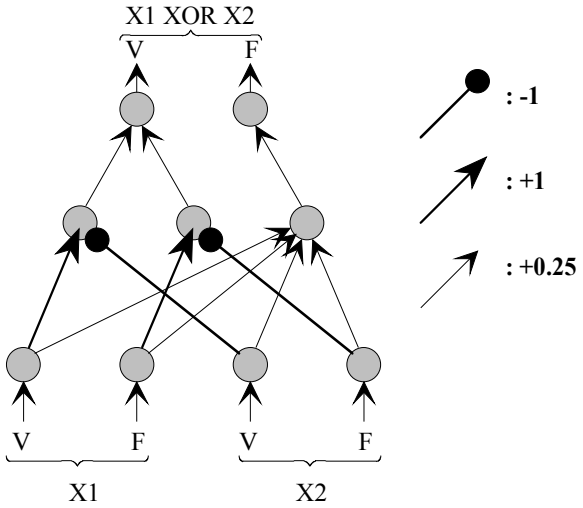


Figure A3-1 : *Remise en cause d'un chemin d'activation par la normalisation.*

Lorsque les poids du réseau sont normalisés, toutes les connexions pondérées à +0.25 atteignent progressivement la valeur +1. Le troisième neurone de la couche cachée reçoit alors un influx permanent de +2 quelle que soit l'entrée proposée ce qui conduit le réseau à répondre systématiquement «F».

Les résultats obtenus pour chacun des trois premiers algorithmes d'apprentissage (apprentissage hebbien orienté, differential hebbian learning orienté et MRT learning orienté) sont similaires ce qui peut être expliqué très simplement : les neurones du réseau étant initialisés à l'état 0, les changements d'état dans le réseau sont presque exclusivement positifs. On a donc généralement $x = \Delta x$ et les trois algorithmes entraînent la même variation des poids synaptiques. Tous les résultats présentés ci-après ont été obtenus avec le MRT Learning orienté.

Dimensionnement du réseau :

Nous avons étudié la convergence de l'algorithme d'apprentissage pour plusieurs tailles de réseau (2, 3, 4, 5, 8, 10, 15, 20 ou 30 neurones sur la couche cachée). Les résultats obtenus pour chacun des critères d'apprentissage sont présentés figure A3-2.

D'après cette figure, le réseau parvient à apprendre le OU EXclusif mais avec une forte dépendance aux conditions initiales lorsqu'il est faiblement dimensionné. Lorsque le nombre de neurones sur la couche cachée devient important (20 et plus), le critère APP100 n'est atteint que dans trois réseaux sur dix. Ce mauvais résultat est en partie dû à la limitation de l'apprentissage à 500 itérations qui oblige le réseau à atteindre le critère APP100 avant 400 itérations (les critères APP25 et APP50 étant eux-mêmes atteints tardivement). De même, pour une couche cachée de 10 neurones, les deux cas de non convergence au critère APP100 correspondent à des convergences APP25 pour 442 itérations (ce qui correspond aussi à une convergence APP50) et 457 itérations.

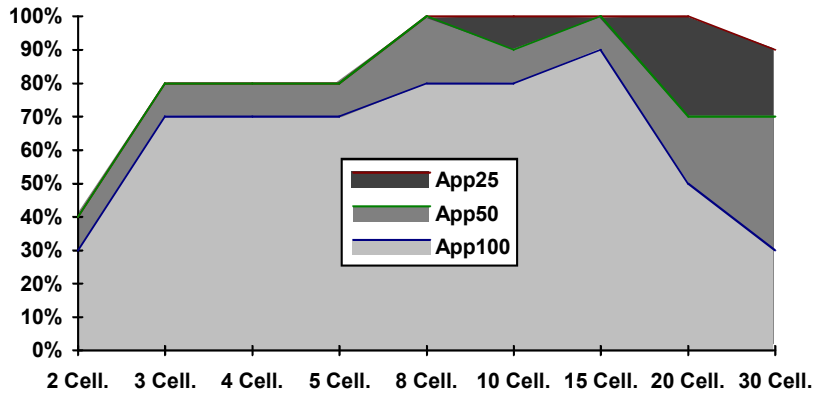


Figure A3-2 : *Convergence de l'apprentissage en fonction de la taille du réseau*

Chaque valeur correspond au pourcentage de réseaux ayant convergé en moins de 500 itérations pour chacun des trois critères (10 essais pour chaque configuration).

Le nombre d'itérations nécessaire à l'apprentissage du OU EXclusif, de l'ordre de quelques dizaines d'itérations, est similaire voire inférieur à celui obtenu pour d'autres types de réseaux (d'après Pearlmuter [PEA 89], Gullapalli [GUL 90] et Ichikawa [ICH 92][94]). La figure A3-3 présente les délais moyens de convergence pour chacun des trois critères. Le nombre de cellules sur la couche cachée apparaît comme un facteur important pour le temps d'apprentissage : trop faible, il limite le nombre de chemins possibles et l'apprentissage est relativement long, trop important, le réseau devient instable et les états non répondants ralentissent considérablement l'apprentissage.

Les problèmes d'apprentissage inhérents au sous-dimensionnement du réseau sont indirectement dus à l'utilisation d'un apprentissage par renforcement. En effet, lorsque le réseau est sous-dimensionné, le nombre de chemins d'activation potentiel est limité (un réseau comportant deux neurones ne compte que 14 connexions). Or l'utilisation d'un apprentissage hebbien avec un signal de renforcement ne permet de créer des chemins d'activation vers une cellule de sortie que si celle-ci est activée et que son activation conduit à un signal de renforcement positif. Or, si aucune connexion ne permet d'obtenir ce résultat (ce qui est très probable lorsque le réseau ne possède que quatre ou six connexions sur sa couche de sortie) le réseau ne dispose d'aucun algorithme pour générer une connexion inexistante. La fonction recherchée ne peut alors être apprise qu'au moyen d'un abaissement général des seuils qui provoque l'activation spontanée des cellules de sortie impossibles à atteindre par la voie synaptique[95]. Dans le cas d'un apprentissage supervisé cette contrainte n'existe pas puisque la consigne permet de créer de nouveaux chemins d'activation.

---

[94] Pearlmuter propose un modèle de réseau récurrent (avec autoconnexions), dont la topologie est fixée par l'utilisateur, apprenant le OU EXclusif en 100 itérations (en moyenne) [PEA 89]. D'après l'auteur ce résultat est comparable aux résultats obtenus dans un réseau en couches utilisant la rétropropagation. Ichikawa utilise un algorithme génétique pour fixer les poids d'un réseau en couches en vue d'apprendre le OU EXclusif. La convergence est obtenue après 30 générations de 20 réseaux [ICH 92]. Lors des tests des SRV-Units, V. Gullapalli n'obtient une convergence qu'après plusieurs milliers d'itérations [GUL 90]

[95] L'étude des matrices de poids obtenues pour les réseaux de deux neurones montre que, généralement, c'est l'activation spontanée d'une des cellules de sortie qui permet l'apprentissage. On observe alors un effet pervers de l'algorithme de descente des seuils : lorsque plusieurs déséquilibres sont identiques (ce qui est le cas pour les deux cellules de sortie en cas d'activation spontanée) la recherche du déséquilibre maximum retient systématiquement le premier (réponse « NON »). Seule cette réponse peut donc être obtenue par activation spontanée ce qui explique en partie la proportion d'erreurs pour n=2.

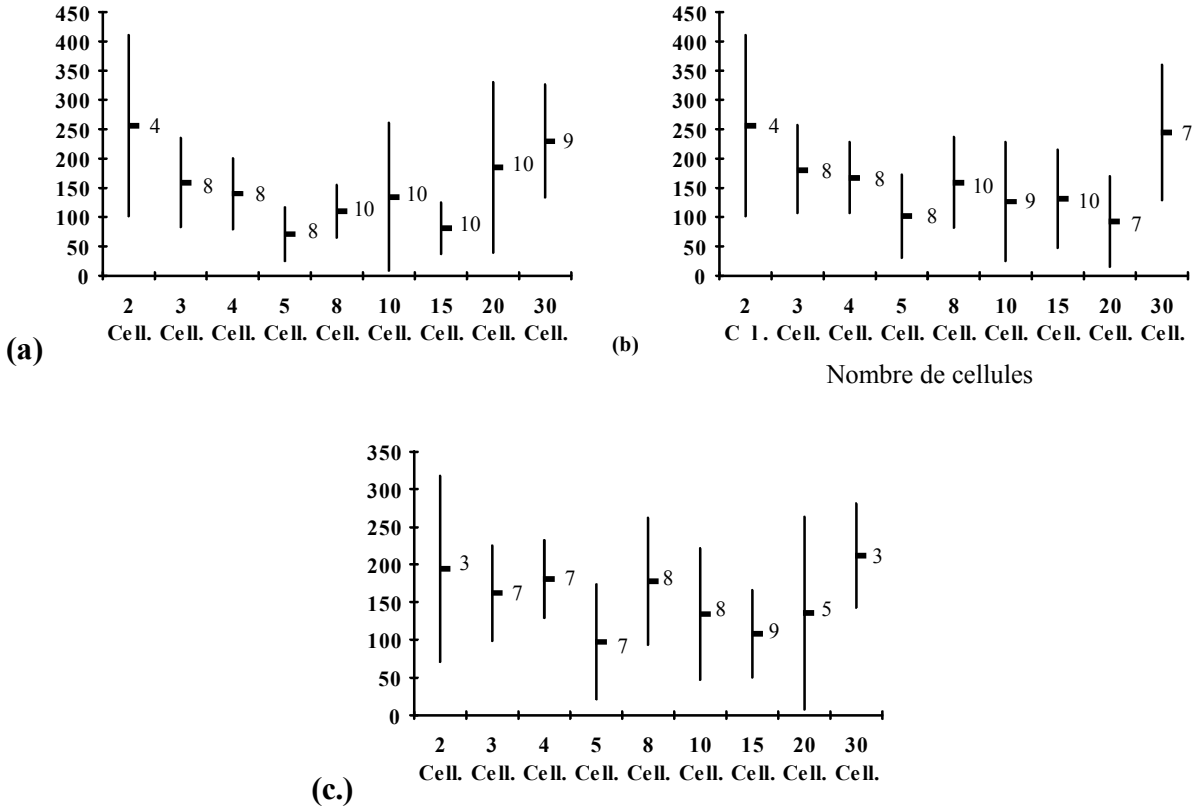


Figure A3-3 : *Délais moyens de convergence des réseaux Moyenne et écart type sur les réseaux ayant convergé en moins de 500 itérations (pour chaque configuration, le nombre de réseaux ayant convergé est indiqué sur la figure).*

(a.) : Critère APP25 (b.) : Critère APP50 (c.) : Critère APP100

Les états non-répondants :

Les états non-répondants apparaissent comme le principal facteur limitant l'apprentissage (figures A3-4 et A3-5). De plus, la procédure utilisée ne permet pas toujours de les éliminer efficacement (diminution globale des seuils de 0.1 par itération correspondant à un état non-répondant).

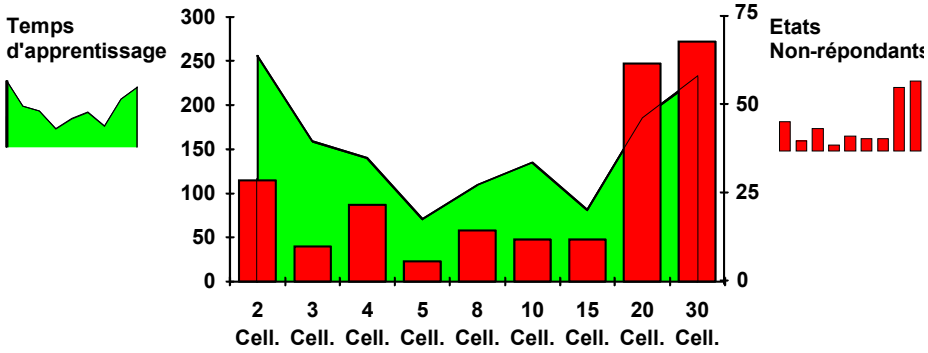


Figure A3-4 : *Comparaison entre le temps moyen d'apprentissage (APP25) et le nombre moyen d'états non-répondants observés durant l'apprentissage.*

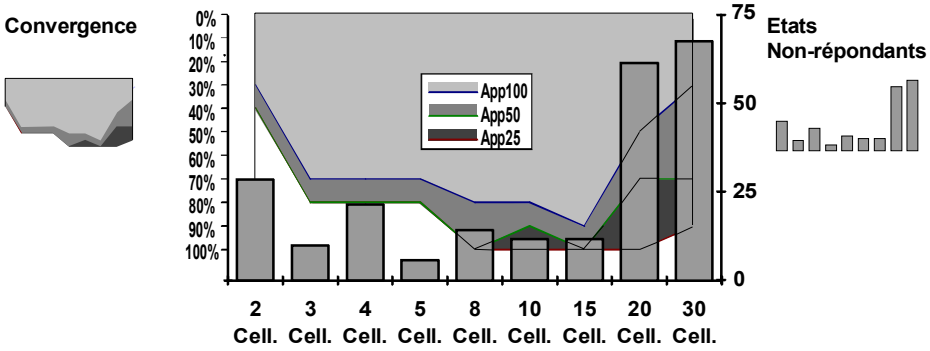


Figure A3-5 : *Comparaison entre le pourcentage de convergence et le nombre moyen d'états non-répondants observés durant l'apprentissage.*

D'après les figures A3-4 et A3-5, les états non-répondants sont particulièrement préjudiciables au bon fonctionnement du réseau lorsque celui-ci est sur-dimensionné : lorsque des états non-répondants apparaissent, ils persistent souvent pendant une centaine d'itérations et ralentissent fortement l'apprentissage. C'est pourquoi nous avons mené des tests, pour les réseaux comptant 20 et 30 cellules cachées, en diminuant le seuil de 0.25 pour chaque état non-répondant. Les résultats montrent que l'apparition des états non-répondants est fortement liée aux paramètres du réseau et que les performances sont accrues :

Pour 20 cellules cachées (10 essais) :

- 100% des réseaux ont atteint le critère APP25 (contre 100% précédemment)
  en un nombre moyen d'itérations : 86.0, écart type : 58.4
  (contre 184.5/145.7 précédemment)
- 100% des réseaux ont atteint le critère APP50 (contre 70% précédemment)
  en un nombre moyen d'itérations : 123.0, écart type : 95.6
  (contre 92.8/77.2 précédemment)
- 70% des réseaux ont atteint le critère APP100 (contre 40% précédemment)
  en un nombre moyen d'itérations : 170.8, écart type : 109.3
  (contre 135.7/128.1 précédemment)
- Nombre moyen d'états non-répondants apparus dans les réseaux avant arrêt de l'apprentissage : 7.5 (contre 61.8)

Pour 30 cellules cachées (10 essais) :

- 80% des réseaux ont atteint le critère APP25 (contre 90% précédemment)
  en un nombre moyen d'itérations de : 65.6, écart type : 33.7
  (contre 230.8/96.7 précédemment)
- 80% des réseaux ont atteint le critère APP50 (contre 70% précédemment)
  en un nombre moyen d'itérations de : 98.2, écart type : 33.0
  (contre 244.1/115.5 précédemment)
- 70% des réseaux ont atteint le critère APP100 (contre 30% précédemment)
  en un nombre moyen d'itérations de : 124.0, écart type : 43.1
  (contre 212.0/69.3 précédemment)
- Nombre moyen d'états non-répondants apparus dans les réseaux avant arrêt de l'apprentissage : 24.9 (contre 68.0)

Influence de l'initialisation des matrices de poids synaptiques :

Nous avons vu que l'apprentissage par renforcement ne permet pas de créer de nouveaux chemins d'activation. Or les valeurs utilisées pour l'initialisation des poids synaptiques ne permettent pas d'assurer l'existence d'un chemin reliant les entrées aux sorties. Cette caractéristique est confirmée par l'étude des courbes d'apprentissage qui montre que les états non-répondants (essentiellement des stabilités précoces) apparaissent dès les premières itérations : pour plus de 40% des apprentissages observés (tous réseaux confondus), les états non-répondants sont apparus au cours des quatre premières itérations de l'apprentissage et pour plus de 55% au cours des dix premières itérations. De plus, on constate (figure A3-6) que les temps d'apprentissage sont fortement corrélés avec la date d'apparition du premier état non-répondant. Il semble donc important d'initialiser les poids synaptiques de façon à limiter l'apparition d'états non-répondants.

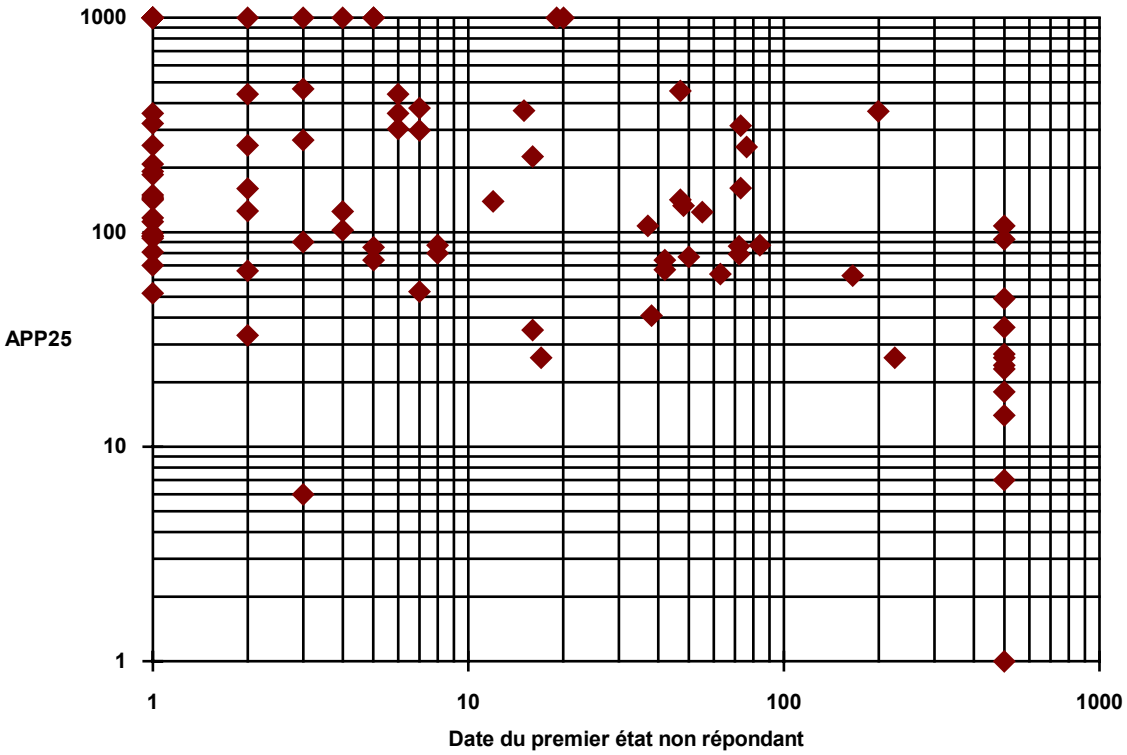


Figure A3-6 : *Vitesse de convergence (APP25) en fonction de la date du premier état non répondant pour les 90 tests effectués.*

La position des groupes de points montre que l'apparition d'états non-répondants aux premiers stades de l'apprentissage compromet fortement la convergence. Les réseaux ne présentant pas d'états non-répondants sont représentés à l'abscisse 500 et les réseaux non convergents à l'ordonnée 1 000 On notera que les points suivants sont associés à plusieurs réseaux : 6 pour le point (1,1 000) et 2 pour le point (5,1 000).

Dans tous les réseaux utilisés pour l'apprentissage du OU EXclusif les poids étaient initialisés aléatoirement entre -0.8 et +0.8. La moyenne des poids étant nulle, il y avait de fortes chances pour que l'influx ne puisse pas être transmis jusqu'aux neurones de sortie (surtout si le réseau ne compte que peu de connexions). De plus, lorsque le réseau recevait un signal de renforcement négatif la moyenne des poids devenait elle-même négative, d'où une aggravation du risque d'apparition de stabilités précoces. Par la suite nous avons systématiquement utilisé des

valeurs d'initialisation permettant de créer une majorité de connexions activatrices (en particulier pour le fuseau de sortie) ce qui permet d'assurer l'existence de chemins d'activation.

Représentation des connaissances dans le réseau :

Le mode ordonnancé a été conçu pour autoriser l'émergence d'assemblées de neurones dans un réseau connexionniste, c'est-à-dire pour permettre une représentation semi-localisée des connaissances dans le réseau.

L'analyse des poids synaptiques après convergence de l'apprentissage montre que le réseau possède deux types de connexions : les matrices FUS_ENTR, CONNEXIO et FUS_SORT (correspondant respectivement au fuseau d'entrée, aux connexions afférentes et au fuseau de sortie) contiennent un faible nombre de connexions fortement pondérées (activatrices ou inhibitrices) tandis que le reste des matrices est composé de poids non négligeables mais d'un ordre de grandeur inférieur. Les figures A3-7 et A3-8 représentent les poids observés pour un réseau de 15 neurones à l'issue d'une phase d'apprentissage de 2 000 itérations (le critère APP100 ayant été obtenu après 29 itérations[96]). Ces poids de plus faible importance sont généralement regroupés en lignes et en colonnes dans les trois matrices (figure A3-8), ils correspondent à des neurones non-impliqués dans l'apprentissage. Cette organisation est caractéristique de l'émergence d'un sous-réseau dans le réseau global.

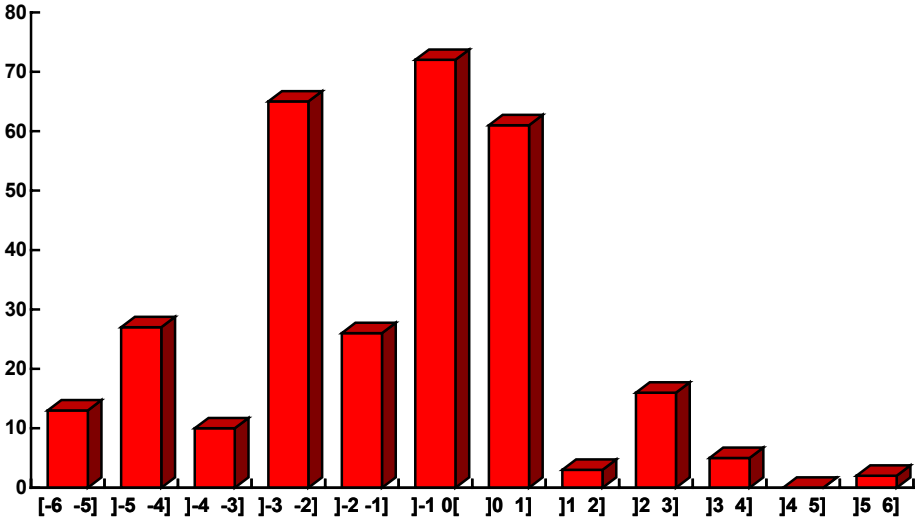


Figure A3-7 : *Répartition des poids synaptiques après apprentissage du OU Exclusif (le réseau compte 15 neurones cachés, 4 neurones d'entrée et 2 neurones de sortie, soit un total de 330 connexions).*

On distingue trois familles de connexions. Seules les deux premières ont un rôle fonctionnel important, elles appartiennent au sous-réseau «OU EXclusif» :

- *Fortement activatrices ([+2 +6], 34 connexions)*
- *Fortement inhibitrices ([-6 -3], 50 connexions)*
- *Connexions de moindre influence (]-3 +2[, 216 connexions)*

[96] Nous avons prolongé l'apprentissage afin de bien différencier les poids utilisés du reste de la matrice W.

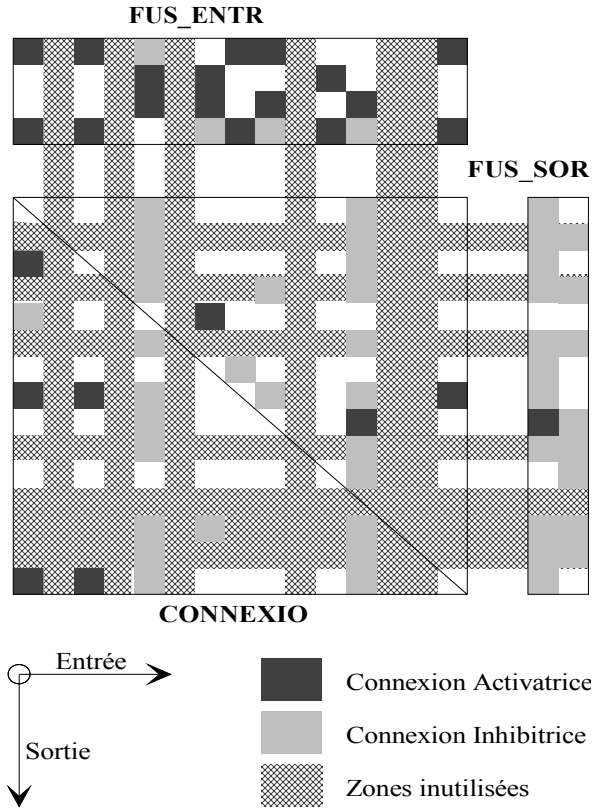


Figure A3-8 : *Représentation matricielle du fuseau d'entrée, des connexions afférentes et du fuseau de sortie*

Les connexions inutilisées sont regroupées en lignes et en colonnes (zones hachurées) correspondant à des cellules inutilisées.

Outre la décomposition entre «neurones utilisés» et «neurones inutilisés», on observe aussi, dans certains réseaux, des neurones (utilisés) qui passent systématiquement à l'état actif lorsqu'une entrée est présentée au réseau. Ces neurones sont reliés à l'ensemble des cellules «utiles» de la couche cachée et de la couche de sortie par des connexions fortes mais non discriminantes (parce qu'ayant toutes des poids sensiblement identiques). Ces neurones réalisent donc une fonction de seuillage de l'activité des neurones appartenant au sous-réseau et facilitent ainsi leur activation. L'intérêt essentiel de cette observation est qu'elle permet de faire un lien entre ces «neurones-seuils» et les «hyper-neurones» utilisés dans l'approche épigénétique (section V.2.4.3). Le comportement d'un réseau utilisant le mode ordonnancé peut donc être rapproché d'un modèle construit pour avoir une représentation semi-distribuée des connaissances, à savoir les Hypercartes Cognitives Epigénétiques (figure A3-9).

L'analyse des matrices de poids peut être complétée par l'étude du comportement dynamique du réseau en phase de reconnaissance (c'est-à-dire l'étude des séquences de mise à jour pour chacun des « patterns » d'entrée). Celle-ci permet d'identifier les neurones participant réellement à la tâche de reconnaissance. On observe généralement que seul un sous-ensemble des neurones du réseau est utilisé pour la reconnaissance du OU EXclusif (dans le cas du réseau présenté ci-dessus 10 neurones sur 15 sont utilisés mais le nombre de cellules utilisées varie généralement entre 5 et 10). Le réseau génère donc bien, lors de l'apprentissage, une assemblée de neurones spécifique.

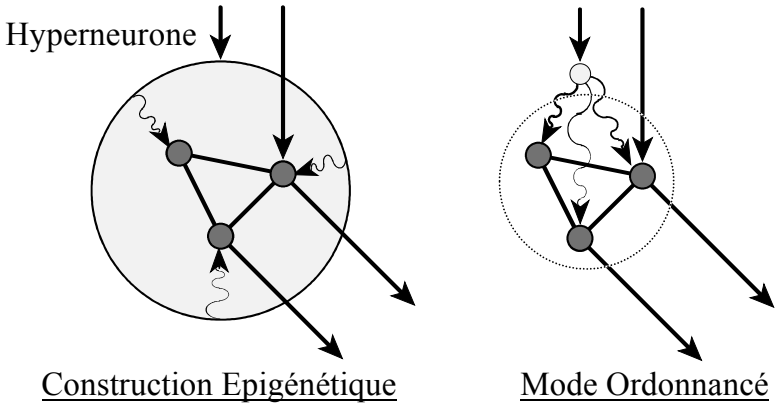


Figure A3-9 : *Analogie entre les Hyperneurones et les Neurones Seuils*

L'activation d'un hyperneurone favorise l'activation des neurones inclus. De même, dans les réseaux utilisant le mode ordonnancé, l'activation d'un neurone seuil favorise la décharge de l'ensemble des cellules vers lesquelles il réémet l'influx nerveux.

Dans le cas du OU EXclusif (et pour la reconnaissance de formes en général), l'émergence d'une seule assemblée de neurones ne présente qu'un intérêt limité. Cependant ce comportement autorise le réseau à «s'auto-dimensionner» en fonction du problème qui lui est soumis : quelle que soit la taille initiale du réseau, l'apprentissage du OU EXclusif conduit en général à l'utilisation de 5 à 10 neurones. Les neurones inutilisés restent alors disponibles pour un apprentissage ultérieur (ce qui permet un apprentissage «par partie», cf. annexe A5) ou peuvent être supprimés. Ceci permet donc de pouvoir dimensionner le réseau automatiquement : le nombre initial de neurones est surévalué et les neurones inutiles sont supprimés après l'apprentissage.

A3.3 Conclusion

Cette première session de tests, particulièrement riche en informations, nous a permis de mieux appréhender le fonctionnement des réseaux utilisant le mode ordonnancé, en particulier sur le plan de la dynamique des séquences d'activation. Les performances obtenues nous permettent de passer à l'étape suivante, c'est-à-dire d'utiliser le mode ordonnancé pour réaliser des tâches plus complexes que le OU EXclusif, d'autant plus que l'analyse des résultats obtenus nous montre que le réseau a bien développé une représentation semi-distribuée des informations apprises. Par contre, cette première approche nous a aussi conduit à remettre en question quelques-uns des principes théoriques retenus pour le mode ordonnancé. On retiendra principalement l'initialisation des poids synaptiques (la valeur moyenne des poids doit être positive afin de créer des chemins d'activation potentiels) et l'élimination des stabilités précoces par diminution des seuils, qui est efficace mais qui s'accompagne d'effets pervers (activation spontanée de la première unité de la couche de sortie).

Par ailleurs, l'apprentissage du OU EXclusif nous a permis d'effectuer une première comparaison des algorithmes d'apprentissage proposés précédemment (section VI.3.4.2). Ces comparaisons nous amènent à des conclusions partiellement opposées aux prévisions théoriques : alors qu'un terme modérateur semblait favoriser la convergence en valeur, nous avons observé que celui-ci est fortement générateur de perturbations lors de l'apprentissage en raison de la normalisation des poids synaptiques entraînée par cet algorithme. Nous avons donc retenu, pour l'apprentissage du OU EXclusif, un apprentissage de type MRT-Learning orienté.

ANNEXE A4 :

CLASSIFICATION :
LA RECONNAISSANCE DE COULEURS

ANNEXE A4 : Classification : La Reconnaissance de Couleurs

A partir des connaissances acquises sur l'apprentissage du OU EXclusif, nous avons cherché à appliquer le mode ordonnancé à un problème concret en comparant nos résultats avec ceux fournis par d'autres méthodes. Ces résultats ont été présentés succinctement dans la section VI.4.4. La présente annexe complète la présentation du problème (en particulier sur le plan des pré-traitements) et détaille les résultats obtenus.

Dans le cadre d'une collaboration avec le Centre de Recherche en Automatique de Nancy (équipe Electricité, Electronique et Automatique), nous avons utilisé le mode ordonnancé pour une application de reconnaissance de couleurs dans un contexte de vision artificielle. Le but de cette application est l'identification robuste d'objets de forme identique, donc uniquement différenciables par leur couleur. Cette identification doit être effectuée dans des conditions d'éclairage variables, en particulier en intensité. Cette application a fait l'objet d'une étude approfondie dans le cadre de l'Atelier Inter-établissement de Productique de Lorraine et plusieurs méthodes de classification ont été utilisées (méthodes statistiques, réseaux de neurones, classification floue, algorithmes génétiques) [BOM 94]. Il est donc possible de comparer les résultats obtenus avec le mode ordonnancé sur les mêmes lots de données que ceux utilisés précédemment. De plus, ce problème permet de tester les capacités de généralisation de l'algorithme, ce qui n'était pas possible avec des fonctions booléennes.

A4.1. Présentation du problème

L'application développée à l'AIP Lorraine comprend un bras robot, un tapis roulant et un système de vision couleur par caméra CCD. Le problème posé est la reconnaissance des pièces posées sur le tapis roulant. Le système doit être capable de reconnaître les différents objets en présence, c'est-à-dire les pièces (rouges, bleues ou jaunes) et le tapis roulant (gris), à partir du signal RVB[97] fourni par la caméra et d'un signal blanc de référence.

L'espace RVB étant difficilement exploitable (en raison des non-linéarités provoquées par l'information luminance), le signal est prétraité de façon à passer dans le repère CIE LUV[98]. Celui-ci sépare, en effet, l'information luminance (L^*) des informations chrominance (u^* et v^*) ce qui permet de réduire la dimension du problème de classification en passant d'un espace tridimensionnel (R, V, B) à un espace bidimensionnel (u^* et v^*, l'information luminance pouvant être éliminée car elle ne contient pas l'information «couleur»).

Le passage au repère CIE LUV s'effectue à partir des informations chrominance et luminance du signal considéré et d'un blanc de référence, exprimées dans le repère CIE 1931 XYZ. Lui-même est obtenu à partir du repère RVB au moyen d'une matrice de transfert (Equation A4-1) :

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.6069 & 0.1735 & 0.2003 \\ 0.2989 & 0.5866 & 0.1144 \\ 0.0000 & 0.0661 & 1.1157 \end{pmatrix} \begin{pmatrix} R \\ V \\ B \end{pmatrix} \qquad \textbf{(A4-1)}$$

---

[97] Rouge, Vert, Bleu.

[98] Le repère CIE LUV est un repère standard de la Commission Internationale de l'Eclairage,

A partir des valeurs X, Y, Z, de l'objet à identifier, et des valeurs X_n, Y_n, Z_n, correspondant au blanc de référence, on obtient les valeurs u', v', u'_n et v'_n (équation A4-2) :

$$u' = \frac{4X}{4X + 15Y + 3Z}$$
$$v' = \frac{9Y}{X + 15Y + 3Z}$$
$$u'_n = \frac{4X_n}{4X_n + 15Y_n + 3Z_n} \qquad \textbf{(A4-2)}$$
$$v'_n = \frac{9Y_n}{X_n + 15Y_n + 3Z_n}$$

On obtient ensuite l'information luminance L^* puis les coordonnées chromatiques de l'objet (u^* et v^*) :

$$L^* = 116\left(Y/Y_n\right)^{1/3} - 16$$
$$u^* = 13L^*\left(u' - u'_n\right) \qquad \textbf{(A4-3)}$$
$$v^* = 13L^*\left(v' - v'_n\right)$$

Les données utilisées pour l'apprentissage sont issues de 6 phases d'acquisition (correspondant à six valeurs d'éclairement différentes) au cours desquelles 400 mesures ont été effectuées pour chaque couleur, soit un total de 9 600 points (figure A4-1). Ceux-ci ont ensuite été répartis en deux lots de données de taille identique (un lot de mémorisation et un lot de généralisation).

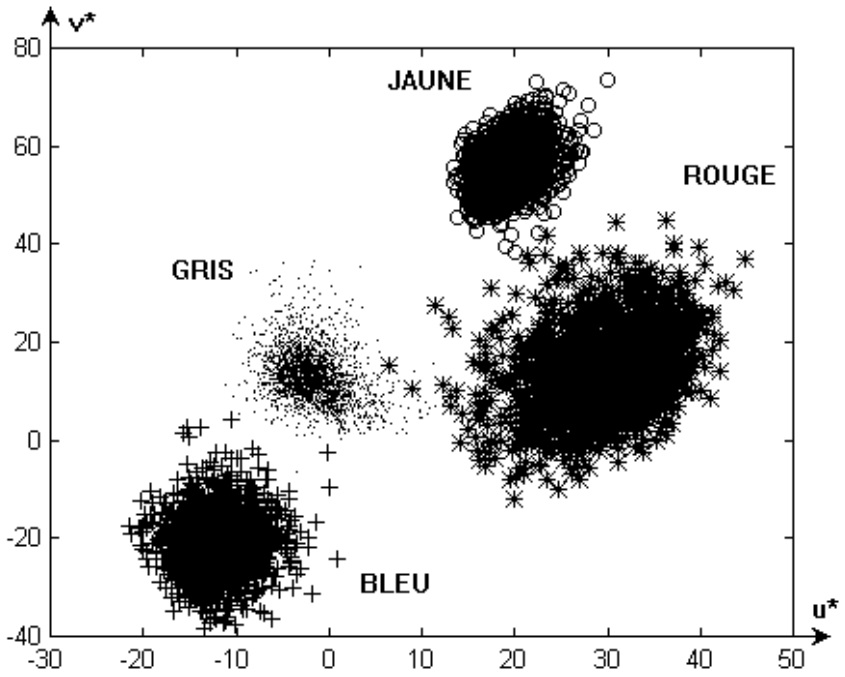


Figure A4-1 : *Représentation graphique des lots de données.*

Outre un réseau à mode ordonnancé, plusieurs autres méthodes de classification ont été utilisées [BOM 94]:

- Un classificateur Bayesien

- Un réseau de neurones type LVQ (Learning Vector Quantization)

- Des classificateurs flous en 6, 7 et 9 termes linguistiques avec éventuellement ajustement manuel des fonctions d'appartenance (7 termes) ou ajustement par algorithmes génétiques (6 termes) [RON 94].

Les résultats obtenus sont les suivants (Tableau A4-1) :

	Class. Flou, 7 termes non ajustés	Class. Flou, 7 termes ajustés manuel.	Class. Flou, 9 termes non ajustés	Class. Flou, 6 termes ajustés génétique	LVQ1	Class. Bayesien
Erreur de mémorisation	0.60%	0.16%	0.14%	0.06%	0.42%	0.06%
Erreur de généralisation	0.49%	0.28%	0.18%	0.08%	0.24%	0.11%

Tableau A4-1 : *Performances comparées des différentes méthodes de classification appliquées au problème de la reconnaissance de couleurs, d'après [BOM 94] et des communications personnelles de Laurent Rondeau (fuzzification en 6 termes avec ajustement des règles par algorithmes génétiques). Les ensembles de mémorisation et de généralisation contiennent 4800 points chacun.*

Etant données les caractéristiques particulières du problème (séparation de quatre classes dans un espace réel à deux dimensions), nous avons appliqué le mode ordonnancé de deux manières différentes :

- Avec un réseau d'unités à valeur binaire, en discrétisant l'espace d'entrée.

- Avec un réseau dont seules les cellules d'entrée sont à valeurs réelles ($x_i \in [0\,1]$), les cellules de la couche cachée et de la couche de sortie étant à valeurs binaires. Les entrées sont représentées par un codage semi-distribué des entrées [LOR 91].

Les sections A4.2 et A4.3 présentent les architectures respectives de chacun de ces réseaux ainsi que les résultats obtenus en apprentissage et en généralisation. Ces résultats seront ensuite analysés dans la section A4.4.

A4.2. Réseau à valeurs booléennes

Dans un premier temps nous avons utilisé un réseau identique à celui proposé pour le OU EXclusif mais comptant 40 neurones d'entrée (discrétisation de l'espace d'entrée en deux fois 20 termes), 20 neurones cachés et 4 neurones de sortie (Bleu, Rouge, Gris et Jaune). Nous avons tenu compte, pour ce réseau, des problèmes de stabilité précoce évoqués pour le OU EXclusif. Les poids synaptiques sont donc initialisés aléatoirement entre 0 et 0.2 pour le fuseau d'entrée et les connexions afférentes et entre 0.1 et 0.3 pour le fuseau de sortie. Tous les seuils sont fixés à zéro. Cette initialisation est suffisante pour éviter les états non-répondants. Nous n'avons donc

pas utilisé d'algorithme spécifique pour les éviter (un état non répondant est considéré comme une erreur du réseau).

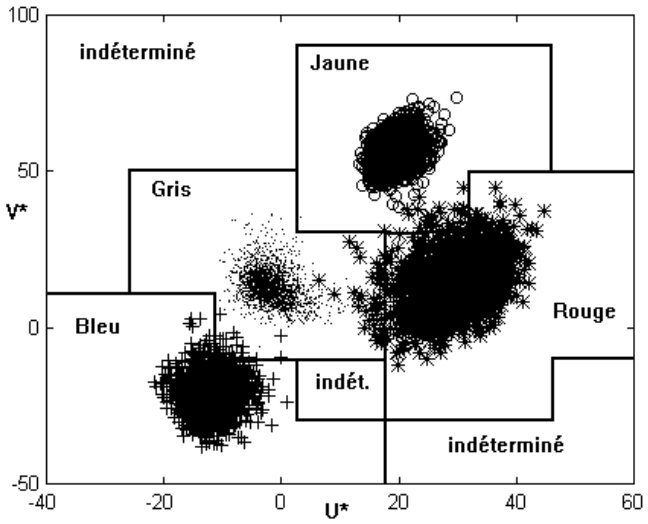
Les entrées du réseau (*u** et *v**) sont discrétisées en 20 termes chacune. *u** prend ses valeurs entre -25 à +45 par pas de 3,5. *v** prend ses valeurs entre -40 à +80 par pas de 6. Les exemples de l'ensemble d'apprentissage sont présentés dans un ordre totalement aléatoire. Le signal de renforcement est de +0.05 en cas de succès, -0.05 en cas d'erreur et -0.1 en cas d'état non répondant. L'algorithme d'apprentissage utilisé est le MRT Learning orienté avec un terme modérateur orienté (contrairement aux résultats obtenus avec le OU EXclusif, cet algorithme est, ici, sensiblement plus performant que les algorithmes n'utilisant pas de terme modérateur).

Résultats :

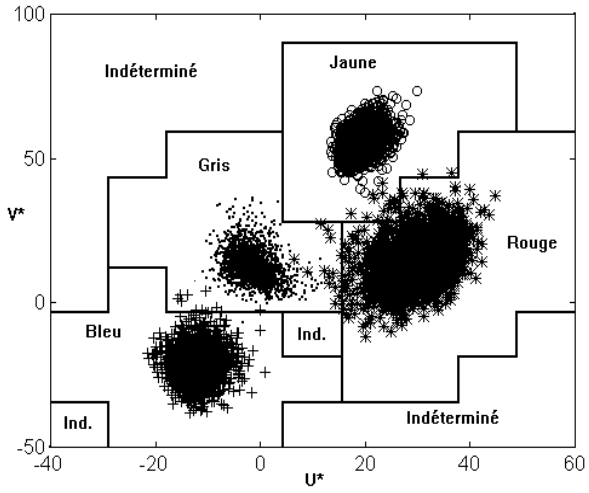
- Après une présentation du fichier d'apprentissage (4 494 points) :
  Erreur de mémorisation : 0.2225 % (10 erreurs de mémorisation sur 4 494)
  Erreur de généralisation : 0.2225 % (10 erreurs de généralisation sur 4 600)
- Après deux présentations du fichier d'apprentissage (soit 8 988 points présentés) :
  Erreur de mémorisation : 0.2225 % (10 erreurs de mémorisation sur 4 494)
  Erreur de généralisation : 0.1557 % (7 erreurs de généralisation sur 4 600)
- Après trois présentations du fichier d'apprentissage (soit 13 482 points présentés) :
  Erreur de mémorisation : 0.1335 % (6 erreurs de mémorisation sur 4 494)
  Erreur de généralisation : 0.1780 % (8 erreurs de généralisation sur 4 600)
- Après 20 000 itérations :
  Erreur de mémorisation : 0.089 % (4 erreurs de mémorisation sur 4 494)
  Erreur de généralisation : 0.1335 % (6 erreurs de généralisation sur 4 800)
- Après 40 000 itérations :
  Erreur de mémorisation : 0.11 % (5 erreurs de mémorisation sur 4 494)
  Erreur de généralisation : 0.11 % (5 erreurs de généralisation sur 4 800)

Analyse :

Les résultats obtenus avec le mode ordonnancé sont légèrement supérieurs à ceux obtenus par les méthodes de fuzzification «classiques» en 7 et 9 termes et nettement meilleurs que les résultats obtenus avec le LVQ1. Par contre, notre réseau a des performances bien moindres que celles fournies par les méthodes de fuzzification avec ajustement de fonctions d'appartenance par algorithme génétique ou par les classificateurs bayesiens. En fait, ces performances sont essentiellement dépendantes de la discrétisation effectuée sur les entrées : le réseau ne fait qu'apprendre à associer une couleur à chacun des éléments de l'espace (des rectangles) contenant un ou plusieurs exemples d'apprentissage (éventuellement il gère les conflits). Il généralise ensuite ces règles sur les rectangles inconnus. Le choix d'une discrétisation plutôt qu'une autre peut donc remettre en question les performances tant en mémorisation qu'en généralisation (chaque cas de conflit se traduisant par, au moins, une erreur dans le lot correspondant). Ce problème explique probablement en partie l'important gain de performances entre une fuzzification en 7 termes (figure A4-2a) et une fuzzification en 9 termes (figure A4-2b) ainsi que l'intérêt de l'ajustement des règles floues. La méthode de discrétisation utilisée ici étant - fortuitement - relativement performante, le réseau peut afficher des performances correctes mais il ne peut en aucun cas les dépasser (figure A4-3).



(Fig. A4-2a.)



(Fig. A4-2b.)

Figure A4-2 : *Règles floues pour des fuzzifications en 7 et 9 termes.*
(a.) : Fuzzification en 7 termes. De nombreux conflits limitent l'apprentissage

(b.) : Fuzzification en 9 termes. La diminution du maillage réduit fortement le nombre de conflits et autorise un important gain de performance.

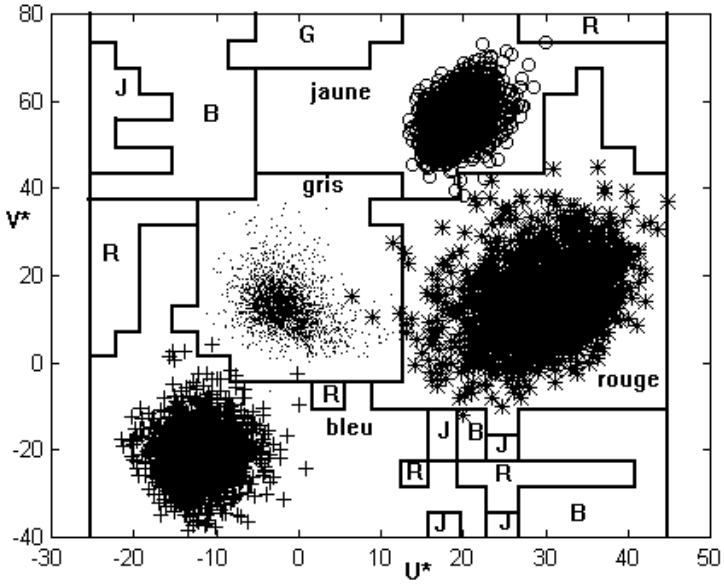


Figure A4-3 : *Fonction de décision du réseau de neurones.*

La discrétisation en 20 termes limite l'apparition de conflits territoriaux. Hormis les quatre zones principales, on observe des zones «indéterminées» pour lesquelles le réseau ne généralise que sur une seule de ses entrées (u ou v*).*

A4.3. Réseau à entrées réelles

Dans un deuxième temps, afin de limiter l'effet du choix du maillage, nous avons utilisé un réseau possédant quarante deux cellules d'entrée à état réel dont l'état est déterminé par des fonctions triangulaires de *u** et *v**. Cette représentation de l'espace d'entrée est dite «semi-distribuée» [LOR 91] car chacune des deux valeurs est représentée par plusieurs neurones (figure A4-4). En dehors du codage des entrées, le réseau est similaire à celui utilisé précédemment.

Résultats :

Le codage semi-distribué des entrées permet un important gain de performances car les frontières séparant les classes peuvent être fixées avec plus de précision alors que, dans le cas des entrées booléennes, les frontières potentielles sont fixées par la discrétisation (figure A4-5). Le réseau de neurones est alors capable de classer les quatre couleurs avec des performances similaires aux meilleurs classificateurs utilisés jusqu'à présent :

- Après une présentation de la moitié du fichier d'apprentissage, soit environ 2 300 points, le réseau présente déjà de bons résultats :
  Erreur sur l'ensemble de la base de mémorisation : 0.16 % (10 erreurs)
  Erreur sur la base de généralisation : 0.29 % (10 erreurs)

- Après une présentation du fichier d'apprentissage (4 494 points) :
  Erreur de mémorisation : 0.0667 % (3 erreurs sur 4 494 points)
  Erreur de généralisation : 0.0890 % (4 erreurs sur 4 600 points)

Par contre, de même que dans le cas du OU EXclusif, l'apprentissage n'est pas stable dans le temps : après deux présentations du fichier d'apprentissage, les performances retombent à 0.1112 % / 0.1335 % pour se stabiliser ensuite à cette valeur. Ce comportement est directement dû à l'algorithme d'apprentissage utilisé : le terme modérateur orienté permet au réseau d'apprendre rapidement les cas isolés et donc d'atteindre de très bonnes performances après un apprentissage court. Par contre, il entraîne une convergence de l'ensemble des poids utilisés vers +/-1 ce qui restreint fortement le choix des frontières potentielles et provoque, à terme, une baisse de performances.

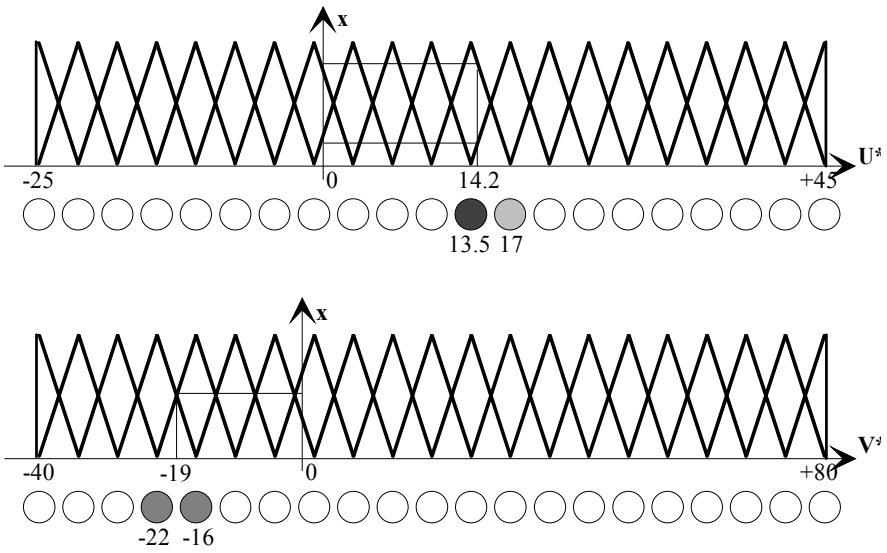


Figure A4-4 : *Fonctions triangulaires d'entrée.*

21 neurones sont utilisés pour le codage de chacune des valeurs d'entrée. A chaque neurone d'entrée correspond une fonction en triangle permettant de calculer son état en fonction de la variable à laquelle il est associé (u ou v*). Les triangles sont symétriques et ont une base de 7 pour u* et 12 pour v*.*

On a ici u=14.2 et v*=-19, ce qui conduit à l'activation des neurones N12, N13, N25 et N26 (dont les fonctions triangles encadrent les valeurs de u* et v*, N12 est centré sur u*=13.5, N13 sur u*=17, N25 sur v*=-22 et N26 sur v*=-16) avec :*

$$x_{12} = 1 - \frac{|14.2 - 13.5|}{3.5} = 0.8 \qquad x_{13} = 1 - \frac{|14.2 - 17|}{3.5} = 0.2$$

$$x_{25} = 1 - \frac{|-19 + 22|}{6} = 0.5 \qquad x_{26} = 1 - \frac{|-19 + 16|}{6} = 0.5$$

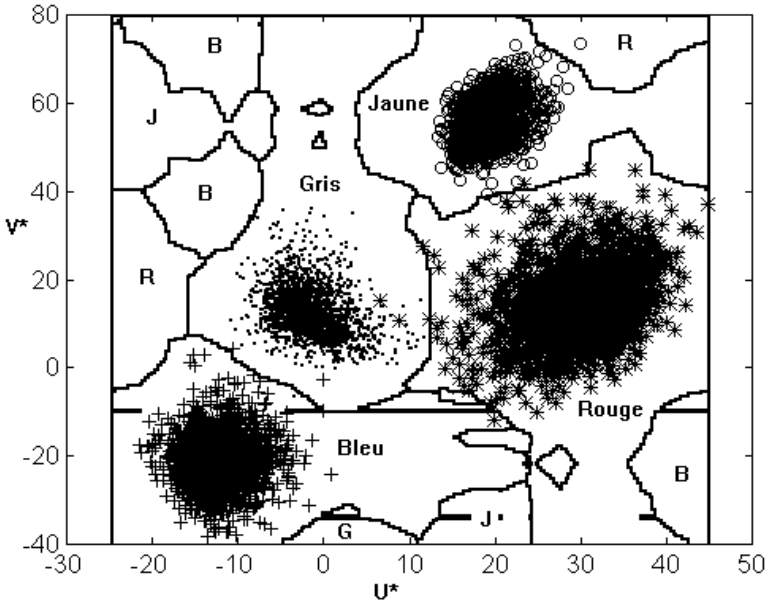


Figure A4-5 : *Classes apprises par le réseau à entrées semi-distribuées.*

A4.4. Conclusion

L'application à la reconnaissance des couleurs nous a permis de mettre en évidence les forces et les faiblesses des réseaux utilisant la mise à jour ordonnancée par rapport à d'autres algorithmes de classification et d'apprentissage.

La faiblesse principale de notre algorithme est l'instabilité de l'apprentissage : celui-ci entraîne souvent des baisses de performances d'autant plus qu'il est en général impossible de savoir à quel moment arrêter l'apprentissage pour assurer les meilleurs résultats. Signalons cependant que, même en cas d'instabilité lors de l'apprentissage, les performances obtenues sont toujours bonnes et comparables à la plupart des autres algorithmes proposés.

Par contre, par rapport à l'ensemble des classificateurs utilisés, le mode ordonnancé présente des avantages structuraux indéniables dus à la représentation semi-distribuée des informations dans le réseau :

- sa capacité à s'auto-dimensionner (après apprentissage, le réseau à entrées semi-distribuées n'utilise que huit neurones sur vingt) permet de limiter les temps de calcul. D'après V. Bombardier [BOM 94] les temps de calcul[99] s'échelonne entre 48h pour le LVQ1 et quelques minutes pour l'approche floue (mais celle-ci demande ensuite un ajustement manuel pour donner de bons résultats, l'ajustement par un algorithme génétique demande plus de 24 heures de calcul). En utilisant le mode ordonnancé il faut moins de 15 minutes pour 4 600 itérations (c'est-à-dire une présentation complète de la base d'apprentissage). En 7 minutes les performances sont donc déjà de 0.16 % / 0.29 % pour le réseau à entrées semi-distribuées et les performances maximum sont obtenues en un quart d'heure. Avec le réseau à entrées booléennes, les calculs sont plus longs

[99] Tous les algorithmes ont été testés sur la même machine (compatible PC type 486/66 avec 32 Mo de mémoire vive) et sous le même environnement : Matlab/Windows

(environ une heure pour des performances de 0.089 % / 0.1335 %) mais, à performance égale, ils restent largement inférieurs à la plupart des autres classificateurs.

- La représentation semi-distribuée permet au réseau de tenir compte d'éléments peu représentés dans la base d'apprentissage : pour chaque couleur, la grande majorité des points de la base est regroupée à proximité immédiate du centre de la classe (ils correspondent aux conditions optimales d'éclairement). Il est probable que cette répartition des exemples d'apprentissage soit à l'origine des faibles performances affichées par le LVQ1 (la base d'apprentissage étant très mal répartie, les zones faiblement représentées sont difficilement apprises). Par contre le mode ordonnancé permet de tenir compte de ces points particuliers en générant des chemins spécifiques dont l'apprentissage est accéléré par l'utilisation d'un terme modérateur orienté.

- Le mode ordonnancé est simple à mettre en oeuvre : l'auto-dimensionnement permet de simplifier le paramètrage du réseau : seules les valeurs d'initialisation des poids et les facteurs d'apprentissage sont déterminants pour le fonctionnement du réseau. On notera que l'initialisation des poids a une influence importante sur l'apparition d'états non-répondants (contrairement au cas du OU EXclusif, l'apprentissage n'a pas été accompagné, ici, par l'apparition d'états non-répondants).

Le problème de la reconnaissance des couleurs nous conforte donc dans les conclusions proposées au terme de la section précédente : grâce à la semi-distribution des connaissances dans le réseau et à la création d'assemblées de neurones, le mode ordonnancé offre des potentialités intéressantes telles que l'auto-dimensionnement et la génération de chemins d'activation spécifiques. Par contre, les algorithmes d'apprentissage utilisés (d'origine hebbienne), s'ils permettent l'apprentissage des comportements souhaités, ne permettent pas la convergence en loi des poids synaptiques. De plus, l'utilisation d'algorithmes d'apprentissage spécifiques permettant la convergence en valeur ne permet pas de résoudre efficacement le problème car ils entraînent une uniformisation des poids qui est souvent nuisible au bon fonctionnement du réseau (en particulier lorsque les entrées sont à valeurs réelles).

ANNEXE A5 :

APPRENTISSAGE DE SEQUENCES BOOLEENNES

ANNEXE A5 : Apprentissage de séquences booléennes

Que ce soit dans la définition générale d'un contrôleur Neuro-Réactif (section V.4), dans l'étude des systèmes multi-agents (section III.4) ou dans l'étude d'un système de transport totalement décentralisé (section II.3.5), nous avons vu que le contrôleur utilisé doit être capable de générer des séquences comportementales. C'est pourquoi nous avons également testé le mode ordonnancé pour l'apprentissage de séquences booléennes bouclées. Cette annexe reprend et complète les résultats présentés section VI.4.5. Elle détaille, en particulier, l'analyse des matrices de poids après apprentissage.

Présentation :

L'apprentissage par renforcement de séquences bouclées est un problème complexe car les différents états de la séquence ne sont pas différenciables par les entrées du réseau. Celui-ci doit donc apprendre à différencier deux états qui, bien que possédant les mêmes entrées, ne conduisent pas à la même sortie. Il doit donc disposer d'une mémoire à long terme (les connexions synaptiques) mais aussi d'une mémoire à court terme lui permettant de suivre le déroulement de la séquence (les activités neuronales).

La séquence utilisée lors des tests comporte quatre états A, B, C, D mais le passage d'un état au suivant dépend de l'avancement dans la séquence (figure A5-1). L'état courant est présenté en entrée et le réseau doit produire l'état suivant. Celui-ci sera ensuite présenté en entrée au pas de temps suivant.

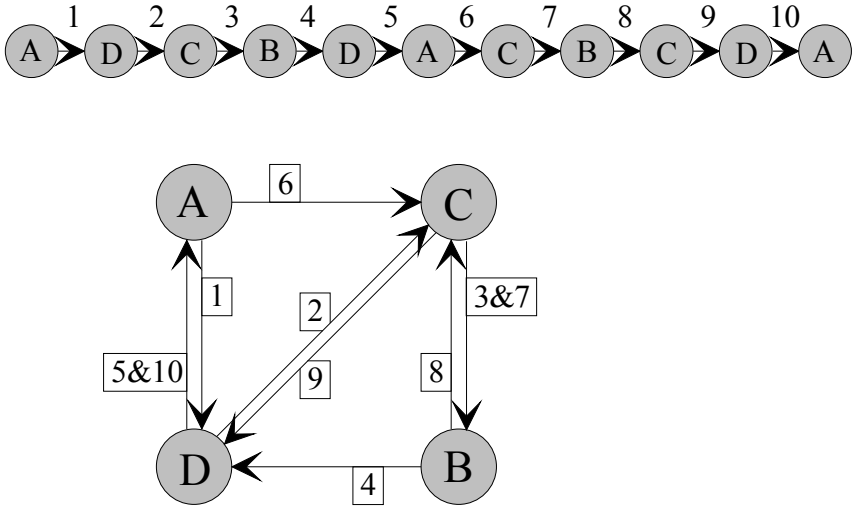


Figure A5-1 : *La séquence utilisée pour les tests*

La séquence est d'ordre 2 (les états T-1 et T-2 sont nécessaires pour déterminer l'état T, voir section IV.2.1, équation IV-5) et compte 11 pas pour quatre états différents.

Architecture :

Le réseau utilisé pour l'apprentissage de cette séquence compte 4 neurones d'entrée, 4 neurones de sortie et 50 neurones cachés. Contrairement aux applications de reconnaissance de formes et de classification, les neurones d'afférence et de sortie ne sont pas remis à zéro lorsqu'un nouveau vecteur d'entrée est présenté au réseau. On considère que le réseau fournit une

réponse lorsqu'un neurone de sortie est mis à jour et que cela conduit la couche de sortie à ne posséder qu'un seul élément actif. Cette connaissance a priori du fonctionnement du système est mise en oeuvre par des connexions réflexes (donc non apprises) inhibitrices (de poids -1) entre les différents neurones de la couche de sortie.

Dans un premier temps, nous avons utilisé, pour l'apprentissage de séquences booléennes, différents modèles de réseaux similaires à ceux proposés pour le OU EXclusif et la reconnaissance des couleurs. Ces premiers travaux nous ont montré que le réseau parvient à apprendre, par renforcement, des séquences de faible longueur (ne comportant qu'une ou deux boucles d'ordre 1). Toutefois, l'apprentissage de la séquence complète (10 pas, ordre 2) demande de guider au moins partiellement l'apprentissage. Nous avons donc modifié l'algorithme d'apprentissage pour utiliser les principes de «sortie forcée[100] » [WIL 89].

Le principe de la sortie forcée est extrêmement simple : l'apprentissage étant fonction des corrélations d'activités entre les neurones du réseau, il suffit de forcer le neurone correspondant à la sortie désirée à l'état actif pour faciliter le développement des chemins d'activation vers cette cellule particulière. En outre, l'utilisation de la sortie forcée permet d'éliminer les états non-répondants de façon plus efficace que les méthodes générales proposées jusqu'à maintenant : un état non-répondant est caractérisé par un chemin n'aboutissant pas aux cellules de sortie. Si la sortie désirée est active, une simple loi hebbienne permet de prolonger les chemins d'activation jusqu'à la sortie (dans le cas d'une stabilité précoce) ou de renforcer les déséquilibres externes par rapport aux déséquilibres internes du cycle bloquant (dans ce cas, la loi hebbienne est aussi utilisée pour réduire les déséquilibres internes).

Le réseau utilisant la sortie forcée possède les mêmes caractéristiques que les réseaux décrits ci-dessus (en termes de taille et de connexions réflexes). Les poids sont initialisés aléatoirement entre -0.1 et +0.1 pour le fuseau d'entrée et les connexions afférentes et entre 0 et +0.2 pour le fuseau de sortie. Les signaux de renforcement utilisés sont de +0.005 en cas de réussite à un pas de la séquence et de -0.005 en cas d'échec. La séquence est présentée par éléments successifs tant que le réseau répond correctement (apprentissage par partie). En cas d'échec les neurones sont ré-initialisés et la séquence est reprise au début. Ce type d'apprentissage «par partie» (les éléments successifs de la séquence sont appris les uns après les autres) est rendu possible grâce au mode ordonnancé : les informations étant localisées sur un petit nombre de connexions, de nouveaux pas de la séquence peuvent être appris sans risquer un oubli catastrophique.

Résultats :

Le réseau utilisant la sortie forcée a réussi à apprendre la séquence complète dans neuf cas sur dix. Le temps d'apprentissage moyen est de 254.3 itérations (écart type : 127). Le réseau ayant convergé le plus rapidement a appris la séquence complète en 86 itérations et le plus lent en 473 itérations. Le réseau n'ayant pas convergé est parvenu à apprendre les dix premiers pas de la séquence en 120 itérations (dans ce cas, l'apprentissage a été stoppé après 1 000 itérations). Les courbes d'apprentissage montrent que le réseau apprend très rapidement les premiers pas de la séquence (pour lesquels il n'y a pas de conflits dus aux bouclages), après quoi l'apprentissage

[100] « Teacher-Forced Learning »

ralentit car le réseau doit parfois remettre en cause les phases d'apprentissage antérieures (figure A5-2).

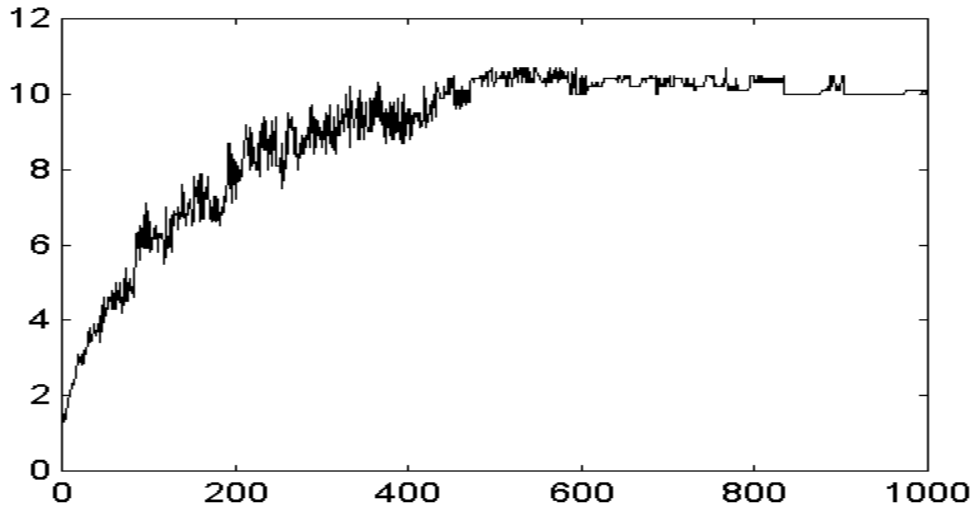


Figure A5-2 : *évolution de l'apprentissage en fonction du temps.*

La courbe représentée est une moyenne sur les 10 sessions d'apprentissage. Dans un premier temps (itérations 0 à 100), l'apprentissage est très rapide et la courbe présente peu d'oscillations car le réseau n'a pas encore dû apprendre les boucles. De 100 à 450 itérations environ, la courbe d'apprentissage oscille fortement car la progression dans la séquence entraîne des conflits avec les premiers pas appris. De 450 à 1 000 itérations, 9 des 10 sessions ont appris la séquence.

Nous avons testé la robustesse de l'apprentissage de la séquence. Pour cela certaines entrées du réseau ont été arbitrairement supprimées et le réseau ne reçoit pas d'entrées pendant un ou plusieurs pas de la séquence (l'état initial «A» est toujours présenté au réseau). La robustesse du réseau est caractérisée par sa capacité à retrouver les états «manquants». Les résultats obtenus sont présentés dans le tableau A5-1 (moyenne sur 25 essais pour le réseau étudié figure A5-3) :

Proportion d'entrées supprimées	100%	50%	25%	12.5%	6.25%
Date moyenne de la première erreur	6	8.24	10.44	8.68	10.24

Tableau A5-1 : *Robustesse de l'apprentissage de séquence.*

(Moyenne sur 25 présentations)

Les résultats montrent que le réseau présente une assez bonne robustesse puisque, en l'absence de toute entrée, les six premiers pas de la séquence sont retrouvés. En fait, la suppression de certains états particuliers du réseau provoque systématiquement une erreur due à l'impossibilité de différencier deux boucles dans la séquence.

Représentation de l'information dans le réseau :

Il est difficile d'étudier les représentations internes de l'information dans les réseaux obtenus : ceux-ci contiennent, en effet, près de 2 800 poids synaptiques et l'étude statique des réseaux n'est pas envisageable. Par contre, il est possible d'utiliser les séquences de mise à jour lors du déroulement de la séquence pour déterminer, à partir des corrélations d'activité, la fonction des différents neurones d'afférence (figure A5-3).

L'analyse des séquences d'activation dans le réseau montre tout d'abord les capacités d'auto-dimensionnement du réseau : seul un sous-ensemble des neurones d'afférence est utilisé pour l'apprentissage de la séquence (dans l'exemple présenté figure A5-3, seuls 14 neurones sur 50 sont utilisés). De plus, l'observation des corrélations d'activité permet de montrer les relations de causalité entre différentes cellules et, plus particulièrement, entre les cellules d'entrée et les cellules d'afférence.

On observe ainsi des liens de cause à effet entre les cellules d'entrée et les neurones d'afférence (de l'entrée A vers le neurone 43, de l'entrée B vers le neurone 48, de l'entrée C vers le neurone 27 ou de l'entrée D vers le neurone 3). Ces liens correspondent à des activations sensori-motrices réflexes, et permettent au réseau de suivre le déroulement de la séquence tant que celle-ci ne comporte pas de boucle interne. Durant les quatre premiers pas de la séquence, seul ce type de relation est utilisé (pas 1 à 5).

Pour apprendre les pas 5 à 11, le réseau doit développer des capacités de mémorisation à court terme. On observe donc des cellules dont l'activité n'est plus directement liée à l'entrée du réseau mais au déroulement de la séquence. Ces cellules, en devenant actives, permettent au réseau de mémoriser l'avancée dans la séquence (neurones 16, 19, 25, 45).

Certains neurones ont des caractéristiques doubles et leur état correspond à l'identification d'une entrée dans un contexte particulier. Les neurones 37 et 42, par exemple, caractérisent le passage par la boucle «AC» par opposition à «AD» : leur activation lors du pas de temps précédent (pas 5, «D», pour la sous-séquence «AC») permet ensuite au réseau de répondre «C» à la présentation de l'entrée «A». De même, le neurone «39» permet au réseau de répondre «C» après présentation de la sous-séquence «CB» alors qu'en son absence, «CB» provoque «D».

Enfin, deux autres types de neurones peuvent être observés dans le réseau : les neurones caractérisant des sous-séquences n'intervenant pas dans la détermination entre deux boucles (50 et 41) et les inter-neurones dont l'activation ne semble pas avoir de rôle fonctionnel précis mais qui appartiennent à l'un ou l'autre des chemins d'activation (8).

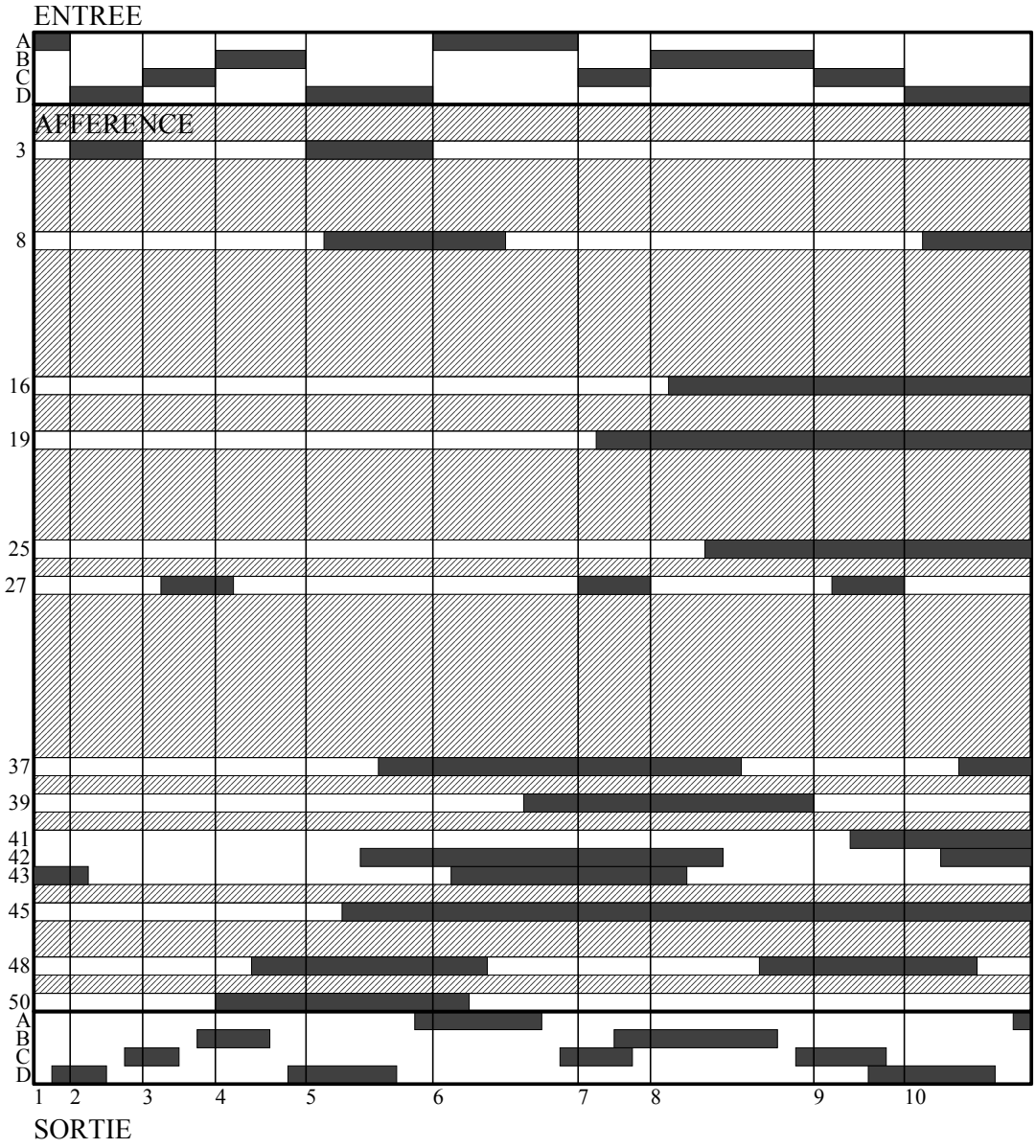


Figure A5-3 : *Etat des neurones d'entrée, d'afférence et de sortie pendant le déroulement de la séquence (séquence apprise en 355 itérations).*

On a porté en abscisse le temps interne du réseau (un neurone mis à jour par pas de temps) et en ordonnée les identifications des différents neurones (ABCD pour les quatre cellules d'entrée, 1 à 50 pour les cellules cachées, et ABCD pour les quatre cellules de sortie). Les neurones à l'état actif sont représentés en noir et les neurones inactifs en blanc. Les zones hachurées indiquent les neurones inutilisés (le réseau n'utilise que 14 neurones pour reconnaître la séquence).

Conclusion :

Le réseau réussit parfaitement à apprendre une séquence booléenne comportant plusieurs bouclages d'ordres 1 et 2. De plus, après apprentissage, le réseau est relativement robuste et il

parvient à retrouver les premiers éléments de la séquence, c'est-à-dire à générer en interne des oscillations lui permettant de produire, sur les cellules de sortie, un cycle prédéterminé[101]. Ces résultats sont encourageants car ils permettent d'envisager des réseaux qui, grâce au mode ordonnancé, généreraient des séquences motrices complexes à partir d'un unique stimulus déclenchant. De plus, l'analyse des séquences d'activation internes au réseau nous montre que le réseau a bien développé les deux types de mémorisation nécessaires. Une mémoire à long terme, codée sur les connexions, permet par des activations type «réflexes» de suivre l'avancée de la séquence tandis qu'une mémoire à court terme, codée par l'état de neurones particuliers, permet de stocker la position dans la séquence et donc d'éliminer les conflits dus aux boucles internes.

Comparé à d'autres modèles connexionnistes d'apprentissage de séquences, le mode ordonnancé ne permet pas d'augmenter la capacité d'apprentissage ni d'apprendre des séquences bouclées d'un ordre supérieur. Ses performances sont même inférieures à celles du modèle d'apprentissage de séquences motrices proposé par Bernard Ans [ANS 94]. Cependant nous n'avons pas relevé, dans la littérature, de modèles connexionnistes capables d'apprendre ce type de séquences sans représentation explicite des boucles internes : dans le réseau proposé par Bernard Ans, certaines cellules sont explicitement réservées à la mémorisation des différentes boucles internes. D'autres modèles, comme celui proposé par W. Miranker [MIR 94], codent explicitement la séquence par l'activation d'une série de neurones identifiant les pas successifs et reliés directement aux cellules de sortie appropriées. Au contraire, le mode ordonnancé permet d'apprendre une séquence de façon implicite, sans utiliser de cellules spécifiques : la représentation semi-localisée de l'information et l'émergence d'assemblées de neurones permettent à un réseau totalement connecté de se spécialiser «par parties» et d'acquérir spontanément une structure interne semblable à celle mise en place explicitement, par l'homme, dans un réseau de neurones devant résoudre un problème similaire.

---

[101] Cette caractéristique est comparable à la définition données par D.O. Hebb pour les assemblées de neurones :

« Any frequently repeated, particular stimulation will lead to the slow development of a « cell-assembly », a diffuse structure comprising cells in the cortex and diencephalon (an also, perhaps, in the basal ganglia of the cerebelum), capable of acting briefly as a closed system, delivering facilitation to other such systems and usually having a specific motor facilitation. » ([HEB 49], Introduction, p. XIX)

« It is proposed first that a repeated stimulation of specific receptors will lead slowly to the formation of an « assembly » of association-area cells which can act briefly as a closed system after stimulation has ceased; this prolongs the time during which the structural changes of learning can occur and constitutes the simplest instance of a representative process (image or idea). » ([HEB 49], chapitre 4., p. 60)

ANNEXE A6 :

NEUROCONTROLE :
LE PENDULE INVERSE

ANNEXE A6 : Neurocontrôle : le pendule inverse

Les trois phases de test décrites dans les annexes précédentes étaient essentiellement destinées à tester les potentialités du mode ordonnancé et à en étudier le comportement avant de l'utiliser pour des applications plus complexes. Nous avons ensuite appliqué le mode ordonnancé à un premier problème de neurocontrôle : l'apprentissage du contrôle d'un pendule inverse. Ce test a déjà été abordé au cours de la section VI.4.6. Cependant nous n'avions pas développé, alors, les différents résultats obtenus. Cette annexe détaille l'architecture utilisée pour l'apprentissage du contrôle d'un pendule inverse (en particulier le principe de l'Adaptive Critic Element) et détaille les résultats obtenus.

A6.1. Présentation du problème

Le problème du pendule inverse[102] a été très largement utilisé au cours des vingt dernières années pour tester les algorithmes de contrôle non-linéaire et en particulier les algorithmes d'apprentissage appliqués au contrôle de systèmes. Cette application fait maintenant figure de «Benchmark» dans le domaine du contrôle [GEV 93].

Le pendule inverse est constitué d'un chariot mobile se déplaçant sur une piste linéaire et portant un balancier pouvant pivoter autour d'un axe fixé sur le chariot (figure A6-1). Le système est simplement commandé en appliquant au chariot une force longitudinale F, le but étant de maintenir le balancier en équilibre sans que le chariot ne sorte de la piste.

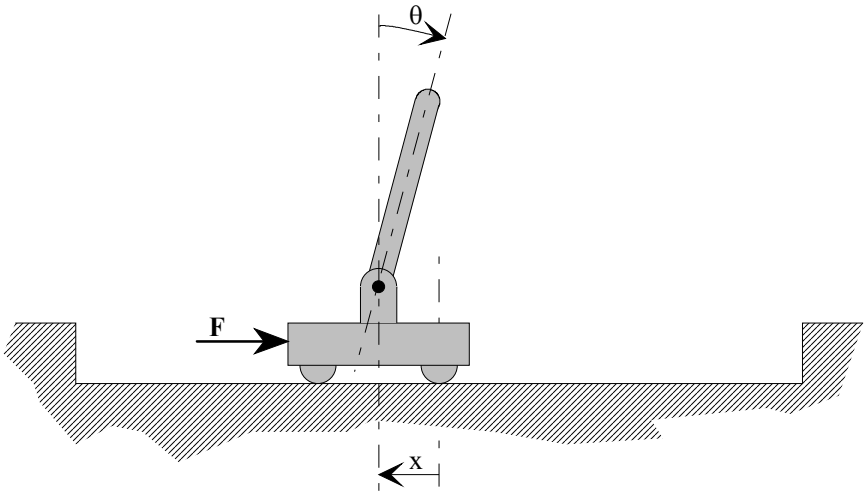


Figure A6-1 : *Le pendule inverse*

L'état du système est donné par quatre variables : l'angle du balancier avec la verticale θ, la vitesse angulaire du balancier $\dot{\theta}$, la position de l'axe par rapport au centre de la piste x et la vitesse du chariot $\dot{x}$. Le domaine de viabilité du système est défini par rapport à l'angle du

[102] « Cartpole problem » ou « inverted pendulum »

balancier (chute du balancier) et à la position de l'axe (le chariot sort de la piste). Les valeurs limites standardisées sont[103] :

$$x \in [-2.4m \quad 2.4m]$$
$$\theta \in [-12° \quad 12°]$$

Outre les limites du domaine de viabilité, les paramètres du système sont :

- L'accélération de la pesanteur : $g = 9.8m.s^{-2}$
- La demi-longueur du balancier : $l = 0.5m$
- La masse du chariot : $m_c = 1kg$
- La masse du balancier : $m_b = 0.1kg$
- La force appliquée : $F = \pm 10N$
- Le temps d'intégration de la commande : $\Delta t = 0.02s$

L'ensemble du système est supposé exempt de frottements. Dans ce cas le système est régi par les équations dynamiques suivantes (d'après [GEV 93]) :

$$\frac{d^2\theta}{dt^2} = \frac{g\sin\theta - a\cos\theta - \mu_b\dot{\theta}^2 l\cos\theta\sin\theta}{l\left(\frac{4}{3} - 3\mu_b\cos^2\theta\right)}$$
$$\frac{d^2x}{dt^2} = \frac{\frac{4}{3}a + \left(\frac{4}{3}\dot{\theta}^2 l - g\cos\theta\right)\mu_b\sin\theta}{\frac{4}{3} - 3\mu_b\cos^2\theta} \qquad \textbf{(A6-1)}$$

avec :

$$\mu_b = \frac{m_b}{m_b + m_c} \qquad \text{et,} \qquad a = \frac{F}{m_b + m_c}$$

Bien que le pendule inverse soit maintenant considéré comme un «Benchmark» utilisé par de nombreux auteurs (voir section suivante), les caractéristiques des systèmes utilisés diffèrent souvent d'un auteur à un autre et il est souvent difficile, voire impossible, de comparer les résultats obtenus. Hormis les variations dues aux valeurs des paramètres «physiques», les différents travaux diffèrent de par les buts poursuivis (éviter la chute ou maintenir le balancier proche de la verticale, centrer le chariot ou éviter de sortir de la piste, ...), par l'information transmise au contrôleur (valeurs continues des quatre variables d'état ou discrétisation), par le type de commande appliquée au chariot ou par le temps d'intégration de la commande.

Ces deux derniers points sont particulièrement cruciaux : le problème du pendule inverse est utilisé en raison de sa non-linéarité. Cependant, au voisinage de l'origine ($\theta \approx 0$ et $x \approx 0$), les lois dynamiques peuvent être linéarisées et le pendule peut être contrôlé par une loi linéaire. Or,

---

[103] Conformément à la standardisation proposée par [GEV 93] nous utilisons les paramètres de [BAR 83].

l'utilisation d'une commande à valeurs réelles (choix de F dans [-10 +10]) permet, en n'appliquant au système qu'une force minimale, de limiter les évolutions du système à la zone linéaire alors qu'un contrôle «*Bang-Bang*» ($F \in \{-10,+10\}$) entraîne des mouvement brusques et génère des instabilités qui compliquent la tâche de contrôle. De même, la diminution de l'intervalle de commande permet d'agir plus vite sur le système et donc de limiter les mouvements du balancier. En fait, tout ce qui concourt à limiter les déplacements du balancier et/ou du chariot (y compris l'utilisation de termes de «frottement») conduit à simplifier le problème en autorisant le pendule à rester plus longtemps dans la zone de contrôle linéaire. Le problème se pose dans les mêmes termes en ce qui concerne les variables d'état utilisées : alors que dans [BAR 83], l'espace d'état est discrétisé (ce qui conduit à autoriser de forts déplacements puisque ceux-ci ne sont pas perçus par le contrôleur), dans de nombreux autres travaux l'espace d'état est composé de quatre variables continues. Le contrôleur peut ainsi réagir immédiatement dès l'amorce d'un déséquilibre. L'amplitude des déplacements est alors directement liée à la force appliquée et à l'intervalle de commande.

Il convient donc d'être prudent quant à l'utilisation du pendule inverse comme support de test pour un algorithme d'apprentissage. Thibault Langlois [LAN 92] propose de soumettre le contrôleur à un environnement bruité ou perturbé ce qui entraîne des déviations par rapport à l'origine et permet de tester plus profondément les performances du contrôleur. Schlomo Geva et Joaquin Sitte [GEV 93] proposent de lancer le pendule dans une position éloignée de l'origine et avec des vitesses non-nulles. Le contrôleur doit alors non seulement être capable de maintenir le pendule en équilibre mais aussi de rejoindre cet équilibre à partir d'une large zone de l'espace d'état. Nous utiliserons ici une approche en tous points similaire à celle de Andrew G. Barto [BAR 83], c'est-à-dire un contrôle «*Bang-Bang*» et une discrétisation de l'espace d'état en 162 zones. Cette approche entraîne de fortes oscillations du système tant en position angulaire qu'en position sur la piste. Le système sort donc systématiquement des zones de contrôle linéaire. En outre, comme le propose Schlomo Geva [GEV 93], nous avons testé le système a partir de positions éloignées de l'origine. Malheureusement, en raison des temps de calculs nécessaires aux tests, ces essais sont restés limités. Ils ont cependant montré de bonnes capacités d'apprentissage et de robustesse.

A6.2. Travaux antérieurs

Le pendule inverse a été très largement utilisé comme support de test pour des algorithmes d'apprentissage connexionnistes ou non. Parmi les approches connexionnistes utilisant un contrôle *«Bang-Bang»* on peut citer, entre autres :

- Andrew G. Barto, Richard S. Sutton et Charles W. Anderson ([BAR 83]) qui utilisent le problème du pendule inverse comme support de test pour une architecture de critique adaptative (ASE/ACE, voir section IV.4.3.1). Les variables d'état sont discrètes. Cette architecture utilise un contrôleur formé d'une seule cellule neuronale et supprime le problème des non-linéarités grâce à l'approche «Boîtes» de Michie et Chambers. Avec ce système, le pendule est maintenu en équilibre pendant 100 000 itérations (environ 30 mn) après moins de 100 essais.

- Charles W. Anderson ([AND 89]) a utilisé une architecture similaire mais basée sur des réseaux à deux couches. Les variables d'état sont continues et le pendule est maintenu en équilibre après 5 000 essais.

- M. Saerens et A. Soquet ([SAE 90]) utilisent un algorithme d'apprentissage par renforcement avec un réseau en couches. Le signal de renforcement est calculé à chaque pas de temps grâce à une fonction de coût quadratique. L'apprentissage est extrêmement rapide (45 000 itérations en moins de 35 essais) mais le système est simplifié par l'utilisation d'une commande «*Bang-Zéro-Bang*» ($F \in \{-10,0,+10\}$) et de variables d'état continues. En outre, M. Saerens considère que le pendule chute lorsque l'angle avec la verticale dépasse 45° (contre 12° pour les autres approches décrites ici).

- Chun-Shin Lin et Hyongsuk Kim ([LIN 91]) ont utilisé la même architecture que [BAR 83] mais en remplaçant l'ASE et l'ACE par des réseaux de type CMAC [MIL 90]. Cette approche permet d'accélérer l'apprentissage grâce à une généralisation du contenu des «Boîtes». Cependant la discrétisation utilisée est plus fine. En outre, comme pour A.G. Barto [BAR 83], l'approche retenue linéarise le problème.

- Sebastien B. Thurn ([THU 91]) utilise l'approche « adaptive-look-ahead planning[104] » pour le contrôle du pendule inverse. Cependant, d'après l'auteur, cette approche n'a pas permis de maintenir le pendule en équilibre plus d'une trentaine de secondes.

- Thibault Langlois ([LAN 92]) utilise une architecture type adaptive critique similaire à celle de Barto [BAR 83] mais basée sur des réseaux à trois couches et en utilisant la rétropropagation du gradient (contrôle «*bang-bang*», variables d'état continues). Cette approche permet d'obtenir de très bonnes performances puisque après 2 000 essais seuls 7% des réseaux n'ont pas convergé alors que plus de 30 % ont convergé avant 200 essais. De plus T. Langlois a testé la sensibilité du contrôleur au bruit : avec un bruit gaussien de 1 % (sur les variables d'état), les résultats chutent fortement (sur 10 apprentissages seuls 6 ont convergé avant 30 000 essais, le plus rapide à converger ayant stabilisé le pendule après plus de 9 000 essais). Avec un bruit de 2 %, un contrôleur a été obtenu après 23 241 essais.

A6.3. Architecture utilisée et premiers tests

L'architecture retenue pour le pendule inverse est en tous points identique à celle proposée par [BAR 83] (section IV.4.3.1). Seul le réseau de contrôle (l'Adaptive Search Element) a été remplacé par un réseau utilisant le mode ordonnancé. En outre, ce réseau ne reçoit pas, comme dans [BAR 83], l'identification de la «Boîte» dans laquelle se trouve le système à l'instant courant, mais directement les quatre variables d'état. Ces variables sont discrètes, le maillage étant identique à celui utilisé pour générer les «Boîtes» de l'ACE (tableau A6-1). La figure A6-2 présente l'architecture générale retenue. L'apprentissage se déroule par essais successifs : au début de chaque essai le pendule est ré-initialisé à proximité de l'origine ($x_0 \in [-0.012\ \text{m} + 0.012\ \text{m}]$, $\theta_0 \in [-0.3° + 0.3°]$, $\dot{x}_0 \in [-0.001\ \text{m.s}^{-1} + 0.001\ \text{m.s}^{-1}]$ et $\dot{\theta}_0 \in [-0.03\ °.\text{s}^{-1} + 0.03\ °.\text{s}^{-1}]$) et l'essai dure jusqu'à ce que le balancier tombe ou que le chariot sorte de la piste.

---

[104] Cette méthode est proche de la rétropropagation à travers le temps mais le réseau « modèle » génère deux réponses différentes : l'état successif et la prédiction du signal de renforcement à venir [THU 91].

Variable	Nbre de termes	Intervalles
x	3	$[-2.4m\ -0.8m], [-0.8m\ 0.8m], [0.8m\ 2.4m]$
θ	6	$[-12°\ -6°], [-6°\ -1°], [-1°\ 0°],$ $[0°\ 1°], [1°\ 6°], [6°\ 12°]$
$\dot{x}$	3	$[-\infty\ -0.5m.s^{-1}], [-0.5m.s^{-1}\ 0.5m.s^{-1}], [0.5m.s^{-1}\ +\infty]$
$\dot{\theta}$	3	$[-\infty\ -50°.s^{-1}], [-50°.s^{-1}\ 50°.s^{-1}], [50°.s^{-1}\ +\infty]$

Tableau A6-1 : *Discrétisation des variables d'état.*

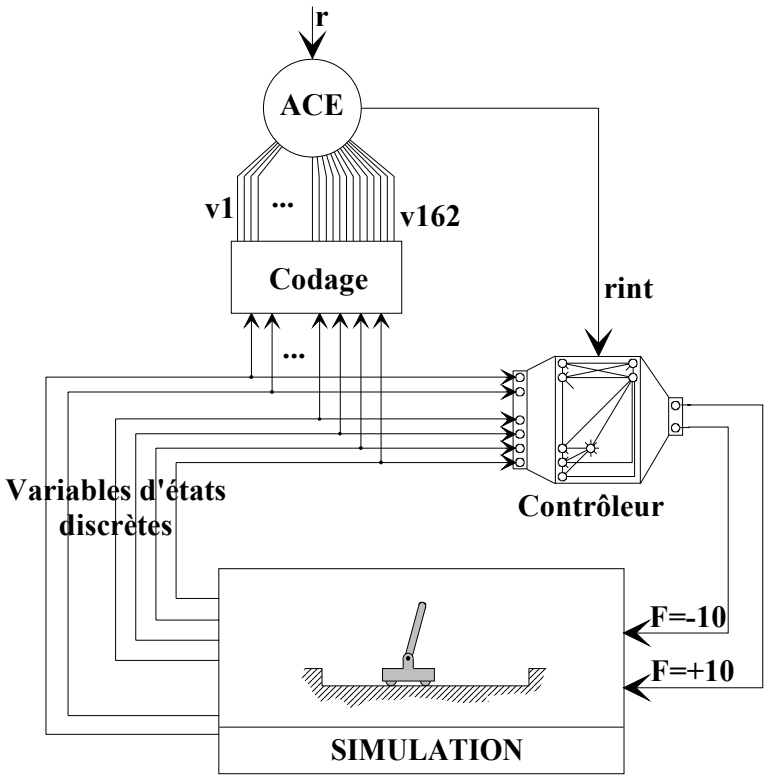


Figure A6-2 : *Architecture utilisée lors de l'apprentissage.*

Le simulateur fournit au contrôleur 15 variables binaires codant l'état du système. Le contrôleur est un réseau à mode ordonnancé qui génère, en sortie, l'action choisie. Les variables d'état sont par ailleurs codées sous forme de 162 «Boîtes» (v1 à v162) utilisées par une cellule ACE pour générer un signal de renforcement secondaire (rint) en prévision du renforcement primaire (r). Le renforcement secondaire est utilisé pour l'apprentissage du contrôleur.

Génération du signal de renforcement : l'ACE :

Comme le montre la figure A6-2, le signal de renforcement secondaire est généré par un *Adaptive Critic Element* (ACE). Cet élément utilise un algorithme d'apprentissage du même type que le TD-learning (section IV.4.3.4) mais restreint aux cellules binaires [BAR 83].

L'ACE est une cellule neuronale recevant en entrée 162 connexions pondérées (v_1 à v_{162}) correspondant chacune à une «Boîte» de l'espace d'état, la fonction de cette cellule étant de générer, en sortie, un signal de renforcement secondaire $\hat{r}$ en prédiction du signal de renforcement primaire r :

Si $\vec{x}(t) = x_1(t), x_2(t), \ldots, x_{162}(t)$ est le vecteur d'entrée de l'ACE, alors on a :

$$p(t) = \sum_{i=1}^{162} v_i(t) x_i(t) \qquad \textbf{(A6-2)}$$

$p(t)$ est une estimation de la qualité de l'état courant. Le signal de renforcement secondaire est alors calculé par la fonction :

$$\hat{r}(t) = r(t) + \gamma p(t) - p(t-1) \qquad \textbf{(A6-3)}$$

où γ est un facteur de décroissance de la prédiction provoquant la disparition progressive du signal de renforcement secondaire en l'absence de renforcement primaire ($0 < \gamma < 1$). Pour l'ACE, on a : $\gamma = 0.95$.

Les poids de l'ACE sont progressivement fixés par la règle d'apprentissage suivante (équation A6-4) :

$$v_i(t+1) = v_i(t) + \beta \hat{r}(t) \bar{x}_i(t) \qquad \textbf{(A6-4)}$$

où β est le facteur d'apprentissage de l'ACE ($\beta = 0.5$) et $\bar{x}_i(t)$ est la trace associée à l'entrée $x_i(t)$: $\bar{x}_i(t+1) = \lambda \bar{x}_i(t) + (1-\lambda) x_i(t)$ (avec $x_i(0) = 0$). λ est le facteur d'affaiblissement de la trace ($0 \leq \lambda < 1$). On a ici $\lambda = 0.8$.

Résultats :

A partir de cette architecture, plusieurs modèles de réseaux à mode ordonnancé ont été testés mais aucun d'entre eux n'est parvenu à stabiliser le pendule pendant plus de deux minutes[105] (6 000 itérations). Les différentes campagnes d'apprentissage ont en général porté sur

---

[105] Au cours de cette première phase, nous avons étendu le mode ordonnancé aux cellules à état réel. Cette version est parvenue à plusieurs reprises à contrôler le système pendant au moins 10000 itérations (point d'arrêt des essais). Cependant, on observe que les performances générales sont particulièrement faibles en regard de ce résultat. Il semble que les réseaux ayant réussi à contrôler le pendule aient plus été générés par le hasard des initialisations que par l'apprentissage lui-même. Ce résultat corrobore les travaux de [GEV 93] qui montrent qu'un contrôleur du pendule inverse (pour un contrôle *bang-bang* et pour des variables d'états continues) peut être rapidement obtenu par un processus stochastique : sur 10000 contrôleurs générés stochastiquement, 833 (soit 8.33%) sont capables de maintenir le système en équilibre pendant au moins 5 minutes (15000 itérations). Dans notre cas, sur l'ensemble des

400 essais. L'examen de ces premiers résultats nous montre que les réseaux apprennent assez rapidement à contrôler le pendule mais qu'ils ne parviennent pas à gérer les conflits entre les deux buts antagonistes (maintenir le balancier en équilibre et maintenir le chariot au centre de la piste). De plus, l'apprentissage reste instable ce qui ne permet pas au réseau de progresser vers la stabilité : les courbes d'apprentissage montrent qu'au cours de certaines périodes, le réseau progresse constamment mais ces périodes sont généralement suivies d'une brusque dégradation des performances.

A6.4. Mode ordonnancé et unités à valeurs stochastiques

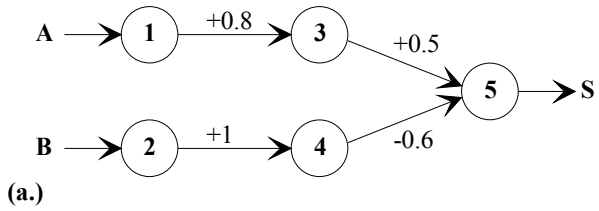
Les résultats obtenus lors de l'apprentissage du pendule inverse montrent, comme les trois tests précédents, que l'architecture connexionniste utilisée permet un apprentissage très rapide de boucles entrées/sorties simples (OU EXclusif, reconnaissance couleurs) ou complexes (apprentissage de séquences) mais que cet apprentissage n'est pas stable dans le temps. Or, dans le cas du pendule inverse, les sessions d'apprentissage sont suffisamment longues (jusqu'à 10 000 itérations par essais) pour que cette instabilité génère occasionnellement des comportements inadaptés et provoque ainsi la chute du pendule.

Les instabilités lors de l'apprentissage sont dues à la confrontation de deux comportements : d'une part, les algorithmes d'apprentissage ne permettent pas la convergence en valeurs (nous avons vu que l'utilisation d'un terme modérateur limite les capacités d'adaptation du système), d'autre part, le signal de renforcement est produit en permanence. La réunion de ses deux comportements provoque une évolution permanente des poids synaptiques et génère périodiquement des instabilités (figure A6-3).

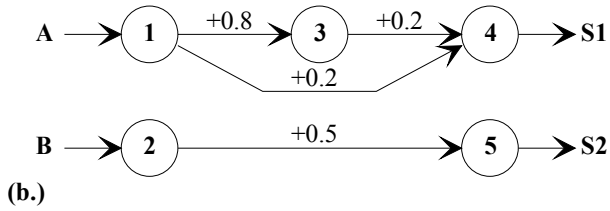
Les instabilités apparaissant lors de l'apprentissage sont donc dues à un comportement similaire à la saturation des sorties observée par Albus pour le réseau CMAC [ALB 75]. C'est pourquoi nous avons utilisé, pour résoudre ce problème, une approche similaire au «Stochastic Real Valued Control» proposé par Vijaykumar Gullapalli [GUL 90] : l'état des neurones est soumis à un processus stochastique dépendant d'une estimation de la qualité de l'état courant et l'apprentissage est lui-même soumis à cette estimation (voir section IV.4.3.2). Avant de décrire l'adaptation du «Stochastic Real Valued Control» au mode ordonnancé, nous allons rappeler brièvement le principe de l'architecture décrite par Vijaykumar Gullapalli.

---

contrôleurs testés en mode continu, environ 10% ont été capables de maintenir le pendule en équilibre pendant 10000 itérations.



A	B	S
0	0	0
0	1	0
1	0	1
1	1	0



A	B	S1	S2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	1

Figure A6-3 : *Exemple de perturbations générées lors de l'apprentissage*

(a.) Un premier type d'instabilité est directement provoqué par l'inégalité des connexions activatrices et inhibitrices face à l'apprentissage : lorsque la fonction entrée/sortie est apprise, le réseau ne reçoit plus que des renforcements positifs. Les connexions inhibitrices ne sont alors renforcées que si elles provoquent la désactivation d'une cellule ce qui n'est pas le cas, ici, de la connexion W_{54} qui est simplement bloquante. Par contre, la connexion W_{53} continue à être renforcée. Elle dépasse alors la valeur 0.6 ce qui génère une instabilité (A=1 et B=1 donnent S=1).

(b.) Les instabilités peuvent être provoquées par les mécanismes de compétition entre des chemins de longueur différente : lorsque l'entrée B est active, le chemin issu de A est bloqué au neurone 3 (car $d_5>d_4$). Cependant, lorsque l'apprentissage est prolongé, toutes les connexions en présence vont croître à la même vitesse ce qui provoque, à plus ou moins long terme, $W_{41}+W_{43}>W_{52}$. On a alors, lorsque A et B sont actifs, $d_5<d_4$ et le réseau active la sortie S1 ce qui provoque une instabilité. L'utilisation d'un terme modérateur dans l'apprentissage provoque une croissance plus rapide des poids les plus faibles ce qui amplifie ce processus.

Rappel : «Stochastic Real Valued Unit» (section IV.4.3.2) :

Les «Unités Stochastiques à Valeur Réelle» (SRV-Units) sont des unités à état réel dont la valeur est fixée par la fonction $\Psi(\ )$. Celle-ci suit une loi normale décrite par son espérance mathématique et sa variance. Chaque cellule possède deux séries de poids (w_i et v_i) utilisées respectivement pour calculer, à partir d'un vecteur d'entrée $x(t)$, l'espérance mathématique de l'état de la cellule $\mu(t)$ et un signal de renforcement interne $\hat{r}(t)$. Ce signal de renforcement est ensuite utilisé pour fixer, par la fonction $s()$, la variance $\sigma(t)$.

On a donc :

$$\mu(t) = \sum_{i=1}^{n} w_i(t) x_i(t)$$
$$\hat{r}(t) = \sum_{j=1}^{n} v_j(t) x_j(t) \qquad \textbf{(A6-5)}$$

La variance est ensuite calculée par :

$$\sigma(t) = s(\hat{r}(t)) = \max\left[\left(\frac{1-\hat{r}(t)}{5}\right),0\right] \qquad \textbf{(A6-6)}$$

Le facteur 5 utilisé au numérateur est un paramètre fixé par l'auteur.

L'état de l'unité est alors déterminé grâce à une loi normale $\psi()$:

$$A(t) = \psi(\mu(t),\sigma(t)) \qquad \textbf{(A6-7)}$$

Selon l'équation (A6-7), la variance est directement fonction du signal de renforcement interne calculé par l'équation (A6-6). Cela autorise la cellule à explorer plus rapidement l'espace de commandes lorsque le signal de renforcement est faible mais cela ne suffit pas à résoudre le problème de la convergence de l'apprentissage. V. Gullapalli propose donc d'utiliser la variance dans l'algorithme d'apprentissage au moyen d'un «bruit normalisé» $\Delta_w(t)$ (équation A6-8) utilisé dans une loi hebbienne pour calculer la variation des poids synaptiques (équation A6-9) :

$$\Delta_w(t) = (r(t)-\hat{r}(t))\left(\frac{A(t)-\mu(t)}{\sigma(t)}\right) \qquad \textbf{(A6-8)}$$

et,

$$w_i(t+1) = w_i(t) + \alpha\Delta_w(t)x_i(t) \qquad \textbf{(A6-9)}$$

Parallèlement les poids v_i sont modifiés grâce à un apprentissage supervisé de type Widrow-Hoff (règle Delta) [WID 60] pour que chacune des unités puisse calculer un signal de renforcement local (équations (A6-10) et (A6-11)) :

$$\Delta_v(t) = r(t)-\hat{r}(t) \qquad \textbf{(A6-10)}$$

$$v_i(t+1) = v_i(t) + \beta\Delta_v(t)x_i(t) \qquad \textbf{(A6-11)}$$

Signalons que Mark Sheppard de l'Université de Teeside (UK) a utilisé une unité SRVU pour le contrôle du pendule inverse avec, d'après l'auteur[106], des résultats concluants (contrôle du pendule pendant 2 millions d'itérations). Cependant l'auteur ne précise pas le nombre d'essais nécessaires ni le nombre de tests effectués : il ne fournit qu'un exemple au cours duquel le réseau parvient à contrôler le pendule après environ 750 itérations. D'une manière générale, l'absence d'informations quant aux paramètres utilisés pour la simulation ne permet pas d'exploiter efficacement ces résultats. Ainsi, le temps d'intégration de la commande n'est pas précisé ce qui ne permet pas d'exprimer les performances en terme de temps.

---

[106] Sheppard, M., Oswald, A., Valenzuela, C., Sullivan, G., Sotudeh, R. - « Reinforcement Learning in Control » - *Rapport interne de l'Université de Teeside, Middlesbrough, Cleveland, TSA 3BA, UK* - 10 pages - Non daté.

En outre, dans cette application, l'unité SRV reçoit un signal de renforcement externe calculé en fonction de $\dot{\theta}$ et de $\dot{x}$ ce qui simplifie l'apprentissage par rapport au problème initial de Barto. De plus, bien que l'auteur ne précise pas le type de commande utilisé (*Bang-Bang*, *Bang-Zéro-Bang*, Réelle, ...), l'amplitude des déviations provoquées par l'application d'une commande semble indiquer que la commande appliquée est à valeurs réelles[107]. Compte tenu de l'ensemble de ces restrictions et des résultats de [GEV 93] en génération aléatoire de contrôleurs, il n'est pas possible d'utiliser ces résultats. Hormis les tests proposés par V. Gullapalli, cette application constitue cependant la seule utilisation des SRV-Units dont nous ayons eu connaissance.

Application au mode ordonnancé :

Le cas du mode ordonnancé diffère des unités décrites par Gullapalli par plusieurs aspects :

- Les cellules utilisées sont à état binaire (la fonction stochastique n'est donc pas la même).
- L'information est stockée sous la forme de chemins d'activation (***path-coding***).
- Le signal de renforcement n'est pas produit localement par chaque cellule du réseau. Il est calculé par un «Adaptive Critic Element» et transmis à l'ensemble des neurones du réseau.

En conséquence, il n'est pas possible d'utiliser directement les équations (A6-5) à (A6-11) dans un réseau à mode ordonnancé. La notion de ***path-coding*** doit être prise en compte : la loi normale ne doit pas être appliquée directement à l'état des neurones mais à la valeur de leur déséquilibre. Ainsi, lorsque la variance est importante, le réseau est autorisé à explorer des chemins différents en modifiant les séquences d'activations dans le réseau (équations A6-12 à A6-14). Plutôt que de SRV-Units, nous parlerons donc ici de *chemins d'activation stochastiques*.

L'équation de calcul du déséquilibre instantané (VI-10) est utilisée ici pour calculer l'espérance mathématique du déséquilibre d'un neurone :

$$\mu_{d_i}(t) = (1 - 2x_i(t))\left[\sum_{j=1}^{n} W_{ij} x_j(t) - \theta_i\right] \qquad \textbf{(A6-12)}$$

Dans notre application, le signal de renforcement est calculé par la cellule ACE à partir de l'estimation de l'état courant $p(t)$ (équations A6-2 et A6-4) grâce à un mécanisme de

---

[107] Plusieurs autres indices semblent confirmer cette information :

- L'utilisation d'un unique neurone ne permet théoriquement pas de contrôler le pendule en raison des non-linéarités du système. L'utilisation d'une commande à valeurs réelles permet cependant de limiter les déplacements du pendule à la zone d'approximation linéaire.
- La courbe d'apprentissage montre que, dans certains cas, le pendule a chuté après seulement trois itérations. Or, lorsque le pendule est initialisé à proximité de l'origine, les valeurs standards ne permettent pas de chuter en moins de 0.15 secondes (8 itérations). Il semble donc que la commande soit à valeur réelle et qu'elle ait, dans certains cas, dépassé les 10 Newtons de la valeur standard.

différences temporelles. Dans les équations de V. Gullapalli, le signal de renforcement est égal à l'estimation de l'état. Compte tenu du signe de $p(t)$ ($-1 \le p(t) \le 0$), l'équation (A6-6) devient :

$$\sigma_d(t) = \frac{-p(t)}{5} \qquad \textbf{(A6-13)}$$

et,

$$d_i(t) = \Psi(\mu_{d_i}, \sigma_d) \qquad \textbf{(A6-14)}$$

Ce qui permet d'introduire un paramètre stochastique dans les séquences de mise à jour utilisées.

Il n'est pas possible d'utiliser, dans l'algorithme d'apprentissage, le «bruit normalisé» proposé par Gullapalli (équation A6-8) car l'état des neurones reste binaire. En outre, l'utilisation du mode ordonnancé nous empêche de relier directement l'activité d'une cellule avec le facteur stochastique appliqué à une connexion particulière[108]. C'est pourquoi nous avons «simplifié» la fonction d'apprentissage (du moins sa composante issue du calcul de la moyenne et de la variance) tout en conservant le principe initial : plus le terme stochastique a d'influence, plus l'apprentissage doit être poussé. L'algorithme hebbien utilisé est de type MRT-Learning avec un terme modérateur.

Soit :

$$\Delta W_{ij} = \alpha.\hat{r}.\sigma_d\left(1 - |W_{ij}|\right)H(t_i - t_j)x_j\Delta x_i \qquad \textbf{(A6-15)}$$

Les poids sont donc modifiés suivant une règle identique à l'adaptation de la règle MRT-Learning utilisée jusqu'à maintenant mais avec un signal de renforcement global $\hat{R}$ combinant le renforcement secondaire et l'évaluation de l'état. Le renforcement secondaire étant lui-même fonction de l'évaluation de l'état (d'après les règles de fonctionnement de l'ACE, cf. équation A6-3), on a, pour le renforcement global[109] :

$$\hat{R}(t) = \frac{-p(t)[r(t) + (\gamma p(t) - p(t-1))]}{5} \qquad \textbf{(A6-16)}$$

<u>*Résultats :*</u>

Le principe des *chemins d'activations stochastiques* a été appliqué à un réseau similaire composé de quinze neurones d'entrée, quarante neurones cachés et deux neurones de sortie. Les

---

[108] Il s'agit d'un problème de « *credit assignment* »

[109] On remarquera que le signal de renforcement obtenu est proche de l'erreur utilisée par T. Langlois pour le calcul de la fonction de coût du contrôleur : initialement calculée par (compte tenu des différences de notation) $E_c(t) = p(t) - p(t-1)$, cette erreur a été modifiée pour lier la vitesse d'apprentissage à l'évaluation de l'état. On obtient alors :

$$E_c(t) = |p(t)|[p(t) - p(t-1)]$$ (d'après T. Langlois, [LAN 92], chap. 6, p.91)

valeurs d'initialisation sont identiques à celles utilisées pour la reconnaissance des couleurs (annexe A4). Dix tests ont été effectués, on considère que la tâche est apprise lorsque le contrôleur parvient à maîtriser le pendule pendant 30 000 itérations (dix minutes)[110]. Au début de chaque essai le système est ré-initialisé autour de l'origine (voir section précédente). Les sessions d'apprentissage ont été limitées à 2 000 essais. Sur les dix contrôleurs testés, cinq sont parvenus à contrôler le pendule pendant 30 000 itérations au moins et trois sont parvenus à contrôler le pendule pendant plus de 12 000 itération (quatre minutes) (tableau A6-2). On notera que tous les contrôleurs qui sont parvenus à contrôler le pendule ont atteint ce but après moins de 300 essais : la moyenne[111] des temps d'apprentissage est de 181.4 essais (écart type : 54.08). La limite des 12 000 itérations est atteinte en moyenne en 250.6 itérations (écart type : 136.03).

Contrôleur	1	2	3	4	5	6	7	8	9	10
Temps de contrôle max.	30 000	29 176	1 927	30 000	30 000	30 000	26 589	12 422	30 000	2 840
Date de contrôle max.	123	342	429	106	284	214	683	271	180	330

Tableau A6-2 : *temps d'apprentissage pour chacun des dix contrôleurs testés.*

Analyse des résultats :

Les résultats obtenus en utilisant des chemins d'activation stochastiques sont donc très bons en ce qui concerne le nombre d'essais nécessaires à l'apprentissage de la loi de contrôle. Par contre, cette technique ne permet pas de résoudre totalement le problème des instabilités lors de l'apprentissage, ce qui n'est en rien étonnant puisque nous n'avons fait que réduire la modification des poids lorsque le réseau est dans une «bonne configuration» sans l'annuler pour autant. Il est d'ailleurs probable qu'en utilisant d'autres paramètres que ceux de Gullapalli pour le calcul de la variance des déséquilibres instantanés les résultats pourraient être encore améliorés. Etant donnés les temps nécessaires à l'apprentissage et la mobilisation des ressources informatiques que cela induit, il ne nous a pas été possible de tester efficacement d'autres configurations. On remarque aussi à la visualisation des courbes d'apprentissage pour les cas non convergents que, passé quelques centaines d'itérations, le réseau ne parvient plus à apprendre et qu'il reste bloqué dans des configurations « négatives » (figure A6-4). Il semble que l'origine de ce phénomène soit l'ACE dont les poids « saturent » après quelques centaines d'itérations. Le signal de renforcement fourni au contrôleur est alors dégradé ce qui limite les possibilités d'apprentissage.

---

[110] Le nombre de tests effectués est trop faible pour pouvoir effectuer une étude statistique poussée des résultats. En outre, la limitation à 30 000 itérations est inférieure à celle généralement utilisée (100 000 itérations). Ces limites sont dues aux temps de calcul induits par le grand nombre d'itérations : chaque essai demande entre 5 et 20 heures de calcul sur un PC 486/66. Il ne nous a pas été possible de mobiliser suffisamment de ressources pour permettre de multiplier les tests ou de les prolonger jusqu'à 100 000 itérations (ce qui représente environ 6 heures de calculs supplémentaires pour chaque essai).

[111] sur les cinq apprentissages ayant convergé avant 2 000 itérations.

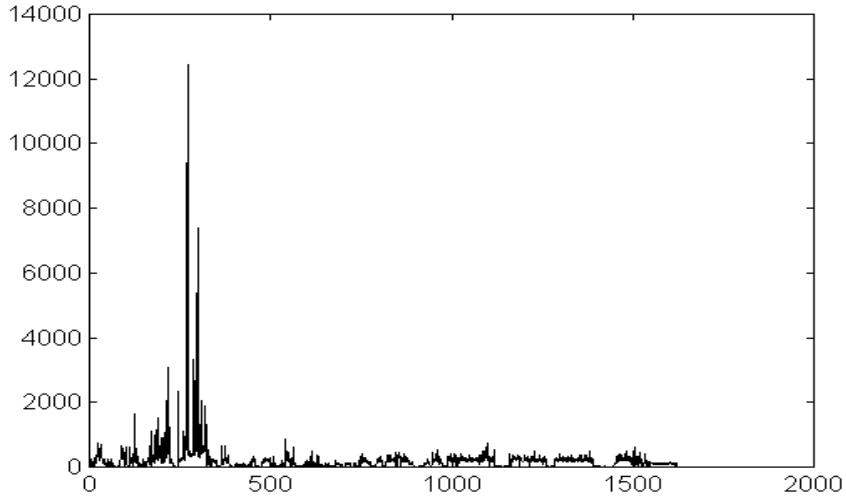


Figure A6-4 : *Courbe d'apprentissage (Contrôleur No. 8)*

La comparaison des performances obtenues par rapport à d'autres modèles de réseaux est rendue difficile par la grande diversité des conditions de test. Le seul modèle avec lequel la comparaison soit possible est l'architecture ASE/ACE de Barto, Sutton et Anderson. D'après [BAR 83] l'ASE/ACE permet d'obtenir un contrôleur efficace en moins de 100 essais. Hormis le problème de l'instabilité à l'apprentissage (et la saturation de l'ACE), les temps de convergence affichés par le mode ordonnancé sont du même ordre de grandeur alors que, dans le cas présent, le contrôleur reçoit directement les valeurs des quatre variables d'état (l'utilisation des «Boîtes» dans l'architecture ASE/ACE permet de convertir le problème du pendule inverse en un problème de contrôle linéaire).

Comparés avec d'autres modèles d'apprentissage, les résultats semblent particulièrement performants : le contrôleur de T. Langlois est efficace après, en moyenne, 565 essais (chaque essai est stoppé après 6 000 itérations et la validation est faite sans apprentissage) et le taux de réseaux ayant convergé en 1 000 essais est similaire à celui obtenu avec le mode ordonnancé. Le contrôleur de C.W. Anderson converge après 6 000 essais, S. Thurn ne parvient à contrôler le pendule qu'une trentaine de secondes (1 755 itérations)... Cependant ces modèles doivent apprendre parallèlement une fonction de contrôle non-linéaire et une fonction d'apprentissage non-linéaire (alors que nous avons utilisé l'ACE qui apprend une fonction linéaire). En outre, ils utilisent tous des variables d'états à valeurs réelles... Autant de différences qui rendent caduques d'éventuelles comparaisons. En outre, l'instabilité de l'apprentissage reste un problème important pour le mode ordonnancé.

Analyse des contrôleurs obtenus :

D'après [GEV 93], l'apprentissage de la commande du pendule dans les conditions «optimales» n'est pas suffisant pour estimer la qualité d'un algorithme et on doit aussi tester la qualité du contrôleur obtenu. Nous avons donc étudié le comportement d'un des contrôleurs ayant convergé (le premier). Malheureusement les ressources informatiques disponibles ne nous ont pas permis de tester tous les contrôleurs obtenus.

Le premier test effectué a été une validation du contrôleur sur 180 000 itérations[112] (initialisation en 0,0,0,0). Ce test montre que le contrôleur est bien capable de contrôler le pendule pendant un temps considéré comme «infini». En outre, il nous a permis d'étudier les comportements du réseau et de visualiser les variations des paramètres. La figure A6-5 présente le comportement du réseau et du contrôleur sur les 120 premières itérations et la figure A6-6 montre l'évolution de x et de θ sur les 50 000 premières itérations. La courbe montre que la position du chariot reste dans les limites de perception du contrôleur ce qui lui permet de «concentrer son attention» sur l'évolution de θ. Par contre, lorsque le chariot se décale au-delà de la limite de perception ($x > 0.8$ ou $x < -0.8$) le contrôleur réagit fortement ce qui provoque de fortes oscillations en x et en θ, oscillations qu'il doit ensuite amortir. Les figures A6-7, A6-8 et A6-9 montrent le comportement du pendule durant cette phase (entre les temps 19 500 et 20 500).

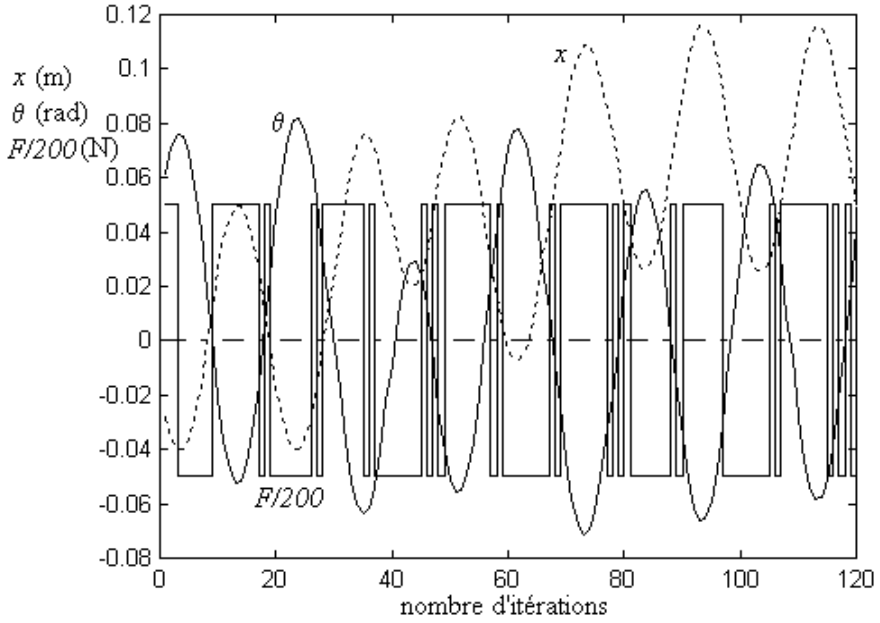


Figure A6-5 : *Comportement du contrôleur sur les 120 premières itérations.*

La courbe en pointillés représente la variable x, la courbe pleine la variable θ et les créneaux représentent le 1/200ème de la force appliquée.
En régime normal (c'est-à-dire quand x reste faible), le réseau développe deux types de comportement : de longs créneaux (6 ou 7 itérations) lui permettent de ramener le pendule vers la verticale tandis que des créneaux plus courts (une itération) permettent d'amortir le pendule quand la vitesse angulaire est trop importante.

[112] ces 180 000 itérations représentent une heure de temps de contrôle simulé. Sur un 486/66 ce calcul demande environ 4 heures sous Matlab. Le contrôleur, sous réserve d'être codé dans un langage compilé (type « C ») serait donc capable de contrôler le pendule est temps réel.

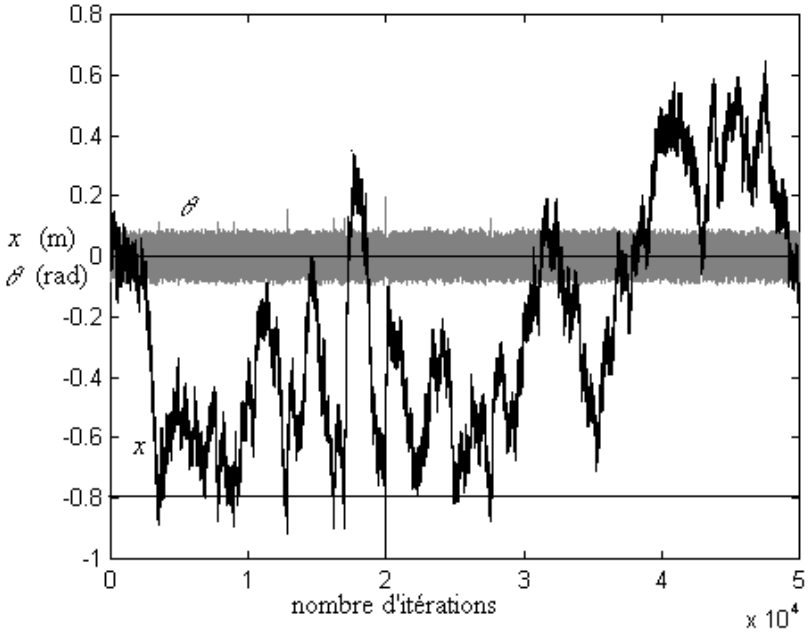


Figure A6-6 : *Evolution de x (noir) et de θ (gris) pendant 50 000 itérations.*

Tant que le pendule est en régime normal (-0.8<x<0.8), le réseau contrôle le système en fonction de l'angle du pendule mais lorsque x sort de ces valeurs, le pendule sort de son régime de fonctionnement normal et le contrôleur corrige brutalement la déviation de qui provoque des oscillations en x et en θ (par exemple après 400, 800, 900 itérations).

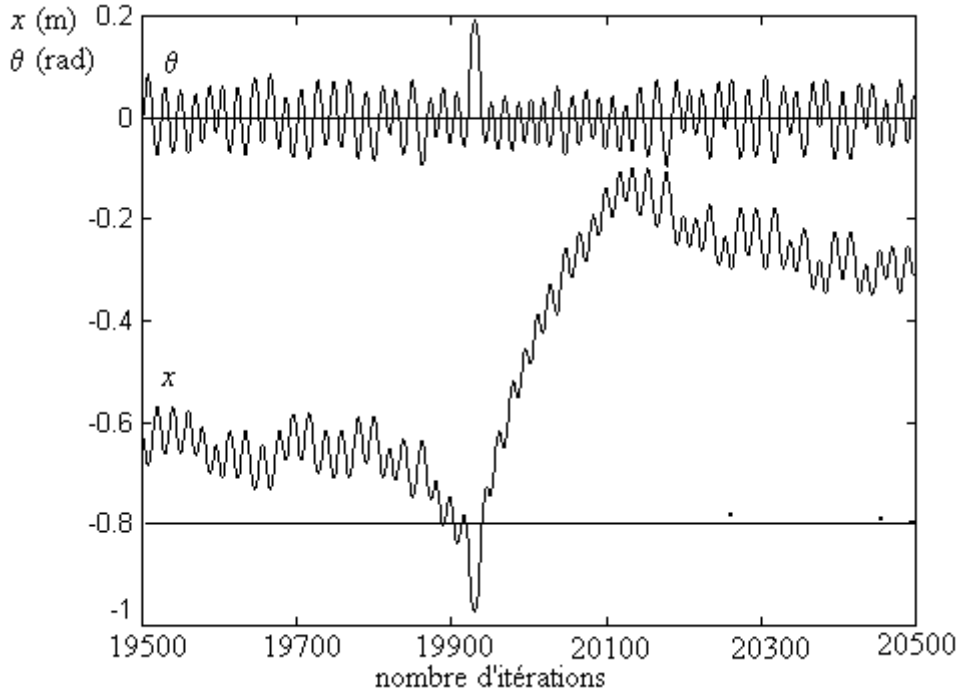


Figure A6-7 : *Evolution de x et de θ entre 19 500 et 20 500 itérations.*

Lorsque x dépasse -0.8, le contrôleur corrige fortement ce décalage ce qui génère de fortes oscillations en x et en θ.

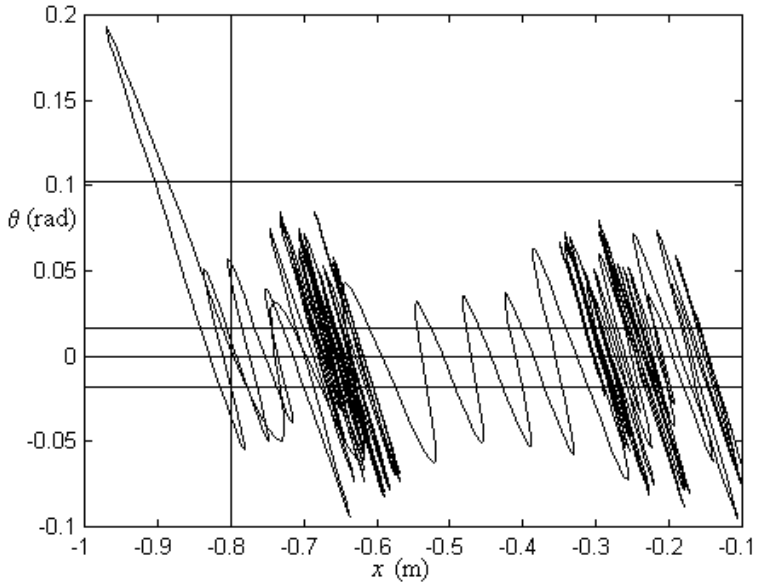


Figure A6-8 : *Trajectoire dans le plan de phase(x,θ) entre 19 500 et 20 500 itérations.*

Les lignes verticales et horizontales représentent les variations dans l'espace d'entrée du réseau

Lorsque le pendule sort du régime normal (initialement on a -0.75<x<-0.65), la correction apportée entraîne une forte déviation en θ ce qui provoque en retour une seconde correction. Le pendule va alors se décaler vers une nouvelle zone de fonctionnement (-0.3<x<-0,1).

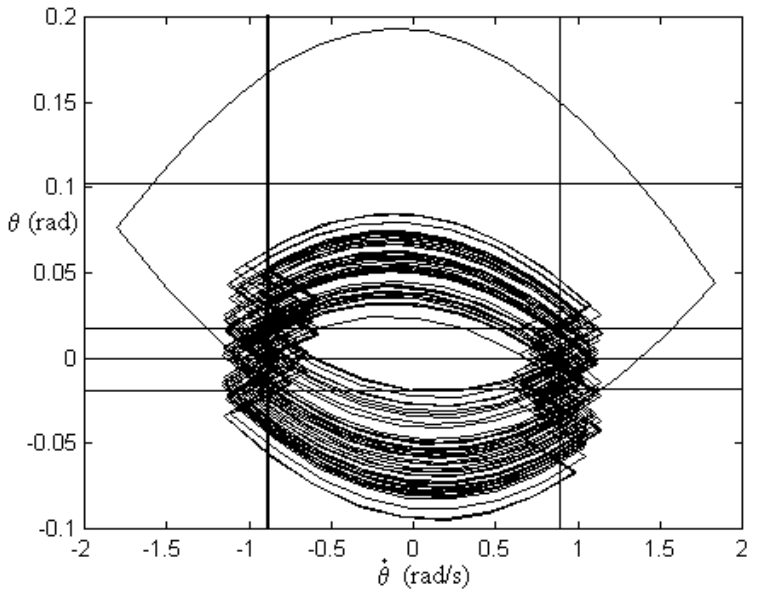


Figure A6-9 : *Trajectoire dans le plan de phase($\dot{\theta}$,θ) entre 19 500 et 20 500 itérations.*

Le régime normal est parfaitement bien identifié dans le plan ($\dot{\theta}$,θ) et ne dépend pas de la valeur de x. On remarquera en particulier les amortissements lorsque la vitesse angulaire sort de [-0.807, +0.807]. Par contre, la correction apportée provoque une forte oscillation que le contrôleur parvient à amortir en un cycle.

Dans un second temps, nous avons placé l'ensemble pendule/chariot dans une position initiale quelconque (vitesse nulle) afin de déterminer la zone de l'espace d'état depuis laquelle le contrôleur parvient à ramener le pendule dans le régime de fonctionnement normal[113]. Ce type de test est proposé par Schlomo Geva [GEV 93] pour tester la robustesse du contrôleur. [GEV 93] propose d'utiliser des vitesses initiales non nulles mais dans un espace d'état plus large (autorisant le pendule à basculer jusqu'à 45°). Les limites que nous avons utilisées ne permettent pas au système de se stabiliser s'il est initialisé avec une forte vitesse angulaire.

Les résultats obtenus lors de cette campagne de tests sont présentés figure A6-10. D'après ces résultats, le réseau parvient à contrôler le pendule quel que soit l'angle initial (entre +0.2 et -0.2 radian) sous réserve que la position du chariot reste dans la zone de fonctionnement normal (entre +0.8 et -0.8 m). En dehors de cette limite, les oscillations générées sur le pendule risque de le faire sortir de l'espace d'état. La chute est alors fonction de l'amplitude de l'oscillation, c'est-à-dire du temps que dure la correction. Or, deux conditions initiales proches peuvent provoquer des corrections de durées différentes en raison du temps d'intégration de la commande, 0.02 seconde, qui est relativement grand face à la période d'oscillation du pendule (environ 0.4 seconde). C'est pourquoi on observe des zones pour lesquelles la chute est possible mais non systématique. En fait, le réseau est capable de contrôler le pendule depuis l'ensemble des points de fonctionnement accessibles depuis la position initiale (0,0,0,0, zone grisée sur le figure A6-10) mais il n'a pas appris à corriger les oscillations angulaires générées lorsqu'on s'éloigne de cette zone. Pour obtenir des contrôleurs robustes en x et en θ, il aurait été nécessaire de modifier la phase d'apprentissage, soit en perturbant le système, soit en le réinitialisant dans une position aléatoire.

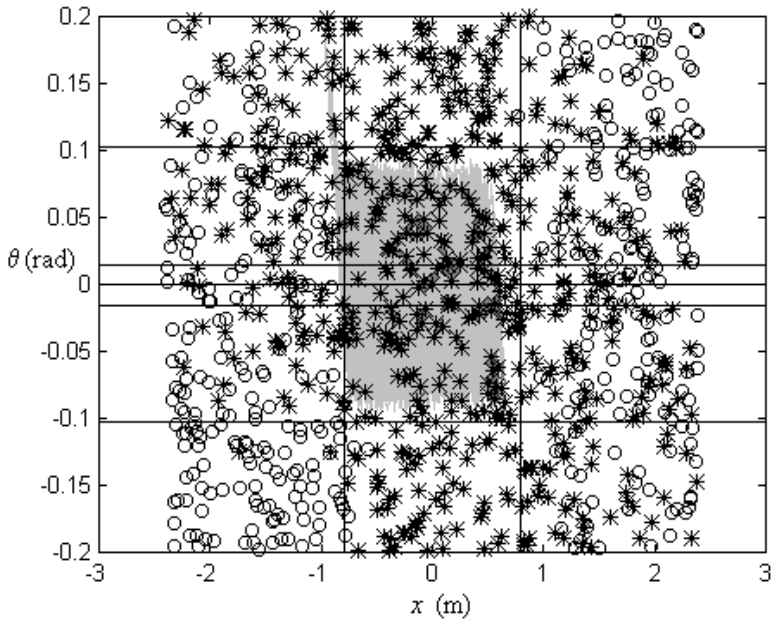


Figure A6-10 : *Comportement du système suivant les conditions initiales.*

******* *: pas d'erreur avant 1 000 itérations,* ***o*** *: erreur avant 1 000 itérations.*

[113] Pour limiter les temps de calcul nous avons considéré que le contrôleur parvient à maîtriser le pendule lorsqu'après 1000 itérations le système n'est pas sorti de l'espace d'état. Nous avons choisi cette solution après avoir constaté, sur une série de test prolongés jusqu'à 10000 itérations, que toutes les erreurs ont lieu dans les 500 premières itérations. De fait, sur les 1000 points essayés, nous avons observé 367 erreurs dont 1% entre 900 et 1000 et 3% entre 800 et 900.

Analyse du réseau obtenu :

Dans un second temps, nous avons étudié le réseau contrôleur obtenu (toujours pour le réseau numéro 1) afin de déterminer si des zones fonctionnelles ont effectivement émergé au cours de l'apprentissage.

La répartition des poids dans le réseau montre que, comme dans le cas du OU EXclusif, une minorité de poids s'est spécialisé (tableaux A6-3, A6-4 et A6-5).

Bornes	-0.2;-0.1	-0.1;0	**0;0.1**	**0.1;0.2**	0.2;0.3	0.3;0.4
Nombre de poids	6	71	362	132	22	7
Pourcentage	1%	11.8%	60.3%	22%	3.7%	1.2%

Tableau A6-3 : *Répartition des poids dans le fuseau d'entrée*

Les valeurs représentées en gras sont les valeurs d'initialisation.

Bornes	-0.05;0	**0;0.05**	**0.05;0.1**	**0.1;0.15**	**0.15;0.2**	0.2;0.25	0.25;0.3	0.3;0.35
Nombre de poids	3	58	856	568	54	15	4	2
Pourcentage	0.2%	3.7%	54.9%	36.4%	3.5%	1%	0.2%	0.1%

Tableau A6-4 : *Répartition des poids dans la couche cachée*

Les valeurs représentées en gras sont les valeurs d'initialisation.

Bornes	0;0.05	0.05;0.1	**0.1;0.15**	**0.15;0.2**	**0.2;0.25**	**0.25;0.3**	0.3;0.35	0.35;0.4
Nombre de poids	3	13	26	17	14	4	2	1
Pourcentage	3.7%	16.3%	32.5%	21.3%	17.5%	5%	2.5%	1.2%

Tableau A6-5 : *Répartition des poids dans le fuseau de sortie*

Les valeurs représentées en gras sont les valeurs d'initialisation.

Cette première information tend à montrer que des connexions spécifiques ont bien été apprises par le réseau. Nous avons donc étudié le comportement du réseau pour chacun des 162 points de l'espace d'entrée. Il apparaît que le réseau n'utilise que 26 neurones cachés sur les 40 que compte le réseau. Il y a donc bien eu auto-dimensionnement du réseau. Pour savoir si la

spécialisation des cellules est limitée à deux familles de neurones (utilisées/inutilisées), nous avons visualisé les neurones utilisés pour chaque point de l'espace d'état (figure A6-11) puis, afin de simplifier l'analyse, après projection sur chacun principaux plans de phases (x,θ), $(x,\dot{x})$ et $(\theta,\dot{\theta})$ (tableaux A6-6, A6-7 et A6-8).

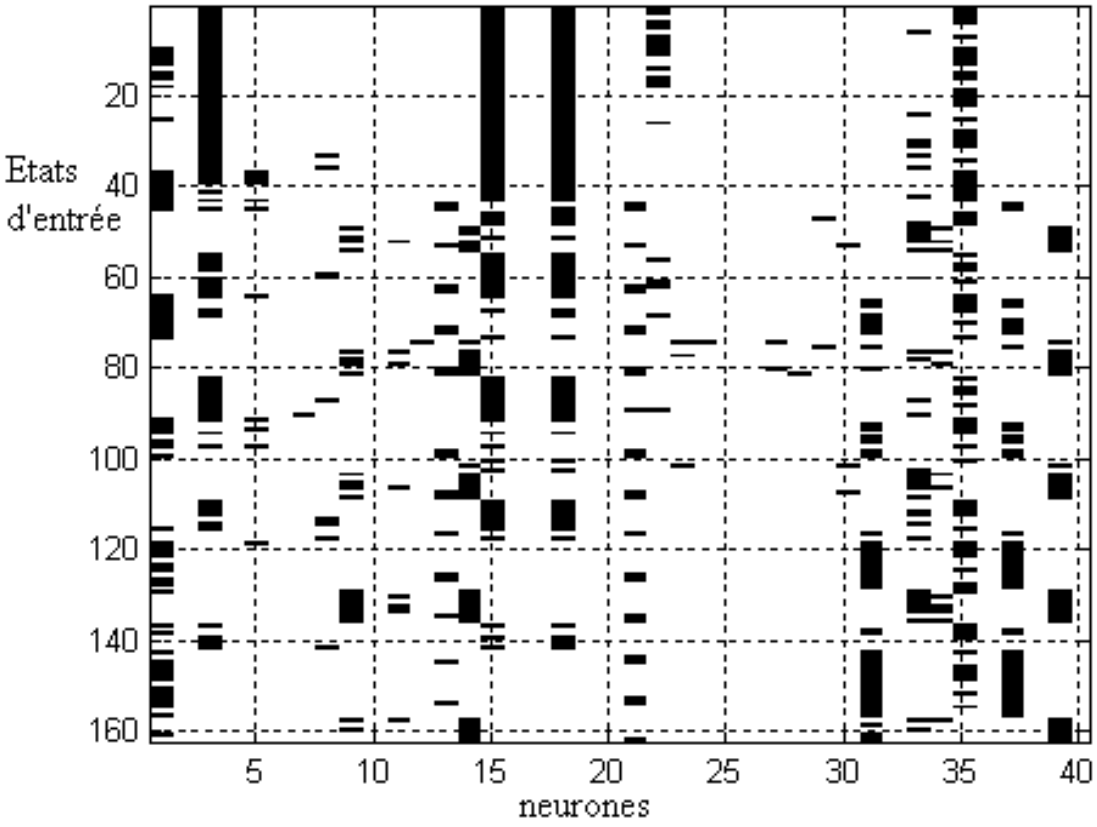


Figure A6-11 : *Activité des neurones pour chacun des points de l'espace d'entrée*

Les 162 points de l'espace d'entrée sont organisés comme suit :
- Pour chacune des variables par valeurs croissantes
- Par valeur de $\theta,\dot{\theta},x,\dot{x}$

	-12°<θ<-6°	**-6°<θ<-1°**	**-1°<θ<0°**	**0°<θ<1°**	**1°<θ<6°**	**6°<θ<12°**
-2.4<x<-0.8	3-15 18-22	3-5-8-9 11-13-14 15-18-21 30-32-33 37-39	3-9-11-13 14-15-18 21-22-27 28-31-34 37-39	3-5-7-9 11-13-14 15-18-21 22-30-31 33-34-37 39	3-8-9-11 13-14-15 18-21-31 33-34-37 39	13-14-21 31-37-39
-0.8<x<0.8	3-15-18 22-33	3-8-9 14-15-18- 33-34	3-8-9-11 14-15-18 22-23-31 33-34-37 39	3-8-9-14 15-18-31 33-34-37 39	3-8-9-14 15-18-31 33-34-37 39	3-8-9-11 14-15-31 33-34-37 39
0.8<x<2.4	3-15 18-22	3-5-15 18-29-33	3-5-12 14-15-18 22-23-24 27-29-31 37-39	3-5-14-15 18-23-30 31-33-37 39	3-9-14-15 18-31-33- 37-39	3-15-18 31-37

Tableau A6-6 : *Activation des neurones cachés après projection sur le plan de phase* (x,θ).

Les neurones 1 et 35 n'ont pas été représentés car il sont toujours actifs dans le plan (x,θ).

	$\dot{x}$<-0.5	**-0.5<$\dot{x}$<0.5**	**0.5<$\dot{x}$**
-2.4<x<-0.8	5-22-37	5-14-22 23-27-29 30-37	5-9-14 29-33-37 39
-0.8<x<0.8	9-11-14 22-33-34 37-39	8-9-14 22-23-33 37-39	8-9-11 14-33-34 37-39
0.8<x<2.4	5-9-11 14-22-33 34-37-39	9-13-14 21-22-27 30-39	5-7-8-9 13-14-21 22-28-33 39

Tableau A-7 : *Activation des neurones cachés après projection sur le plan de phase* $(x,\dot{x})$.

Les neurones 1, 3, 15, 18, 31 et 35 ne sont pas représentés car ils sont toujours actifs dans le plan $(x,\dot{x})$.

	-12°<θ<-6°	**-6°<θ<-1°**	**-1°<θ<0°**	**0°<θ<1°**	**1°<θ<6°**	**6°<θ<12°**
50<$\dot{\theta}$	1-3-15-18 22-33-	3-9-11-13 14-15-18 21-29-30 33-34-39	1-3-9-11 12-13-14 15-18-21 22-23-24 27-28-29 31-33-34 37-39	3-9-11-14 15-18-21 22-23-30 33-34-39	1-3-9-11 13-14-21 31-33-34 37-39	1-3-9-11 14-21-31 33-34-37 39
-50<$\dot{\theta}$<50	1-3-15-18 22-	1-3-5-13 15-18-21 33-37	1-3-5-13 15-18-21 22-31-37	1-3-5-13 15-18-21 31-37	1-5-13-21 31-37	1-13-21 31-37
$\dot{\theta}$<-50	3-15-18 22-33-	8-15-18 33-	8-13-15 18-21-33	7-8-13-15 18-21-33-	1-8-13-15 18-21-31 33-37	1-8-13-15 18-21-31 37

Tableau A6-8 : *Activation des neurones cachés après projection sur le plan de phase* $(\theta,\dot{\theta})$. *Le neurone 35 n'est pas représenté car il est toujours actif dans le plan* $(\theta,\dot{\theta})$

L'analyse des tableaux A6-6, A6-7 et A6-8 est longue et fastidieuse, aussi nous ne présenterons ici que les résultats principaux auxquels ils permettent d'aboutir et qui montrent que le réseau a bien développé une représentation semi-distribuée des connaissances (ces résultats ne sont pas issus de la seule étude des tableaux : nous nous sommes aussi basés sur les corrélations d'activité entre neurones du réseau).

La première information que nous donne cette étude est que certains neurones sont plus spécifiquement dépendants de certains plans de phase. C'est le cas, par exemple, pour le neurone 1 (N1) qui est toujours activé dans les deux premiers plans mais pas dans le troisième. On peut donc en conclure que N1 est plus spécifiquement dépendant de la vitesse angulaire et de ses relations avec l'angle du balancier. Il en va de même, chacun pour des variables spécifiques, pour les neurones N3, N15, N18, N31 et N35 (le fait que N35 soit toujours activé dans les trois plans de phase représentés ne signifie pas qu'il est toujours activé mais que son activité dépend de conditions non visualisées dans ces plans).

Le plan $(x,\dot{x})$ est de loin le plus simple à étudier, d'une part parce qu'une grande partie des neurones est toujours active (ou toujours inactive pour les neurones inutilisés), d'autre part, parce que des règles comportementales simples peuvent y être mises en évidence : les neurones N9 et N37 ne sont, en effet, activables qu'en dehors de deux zones « dangereuses » pour lesquelles la vitesse du chariot tend à l'éloigner du centre (N9 pour les vitesses négatives et N37 pour les vitesses positives). On constate d'ailleurs que le neurone N21 réalise le complément de N37 : il est activable exclusivement lorsque le système entre dans une zone dangereuse. D'autres règles comportementales peuvent être observées dans ce plan (par exemple, N27 n'est activable que lorsque le chariot est excentré et que la vitesse est moyenne) mais leur interprétation est moins évidente.

Le plan $(\theta,\dot{\theta})$, lui aussi, comporte des neurones spécifiques. Le plus caractéristique est probablement le neurone N5, actif exclusivement lorsque les deux variables sont moyennes. Les cellules N9 et N11, au contraire, sont actives lorsque la vitesse est fortement positive, sauf si cette vitesse permet de ramener le pendule vers la position verticale (c'est-à-dire si l'angle du pendule est fortement positif). De son côté, N3 réalise une fonction inverse (actif lorsque l'angle du pendule est négatif ou quand la vitesse angulaire est fortement négative). Schématiquement

on peut donc dire que N5 identifie les états «sécuritaires», N9 et N11 le risque de chute vers les angles positifs, et N3 le risque de chute vers les angles négatifs.

Le premier plan exposé (x, θ) est plus difficile à analyser, probablement parce que, contrairement aux deux autres, il est formé par deux variables non corrélées (à l'inverse des couples position/vitesse). On peut cependant mettre en évidence certains neurones spécifiques comme N13 et N21 qui sont actifs lorsque x est fortement négatif et que θ est positif ou faiblement négatif, c'est-à-dire lorsqu'il est opportun d'apporter une correction à la position (sans risquer de faire tomber le pendule). Le neurone N29 réalise approximativement la fonction symétrique.

L'étude manuelle des réseaux de neurones comportant plus d'une dizaine de cellules est difficile : il est possible de donner des ébauches d'interprétation mais il est hypothétique, dans la plupart des cas, de chercher à isoler les règles de décision associées à chaque cellule. Le but de cette étude était de vérifier que le réseau développe des représentations semi-distribuées de son environnement. Nous pensons que l'observation de cellules dont l'activation est corrélée avec les règles symboliques de contrôle du système (zones de danger, zones de sécurité, ne corriger la dérive en position que si le pendule penche du bon côté, ...) constitue un indice favorable mais elle ne constitue pas une preuve. Le développement de procédures automatiques de détection des corrélations temporelles entre neurones d'entrée, neurones cachés et neurones de sortie (par exemple basées sur l'analyse delta [FEC 93]) est probablement le seul moyen fiable qui permette d'affirmer l'émergence d'assemblées de neurones dans le réseau.

A6.5. Conclusion

L'utilisation, pour tester les applications du «mode ordonnancé» au neurocontrôle, d'un «Benchmark» comme le pendule inverse nous a permis de progresser plus rapidement, tant pour l'amélioration du contrôleur que pour son analyse. En effet, l'importance de la littérature portant sur ce système nous a permis de mieux identifier les causes des instabilités apparaissant lors de l'apprentissage et d'y remédier en adaptant pour le mode ordonnancé les solutions proposées pour les réseaux «classiques».

Les résultats des tests sont prometteurs puisque le mode ordonnancé apparaît comme une des méthodes d'apprentissage les plus rapides pour le problème du pendule inverse et que l'analyse des matrices de poids obtenues nous montre que le réseau a bien développé, en cours d'apprentissage, une représentation semi-distribuée des connaissances. Par contre, malgré les améliorations apportées par l'utilisation des «chemins d'activation stochastique», l'apprentissage reste instable et, dans la moitié des cas, le réseau n'a pas atteint les 30 000 itérations fixées comme limite à la phase d'apprentissage. Cependant nous avons vu que ce comportement n'est pas exclusivement le fait du mode ordonnancé mais aussi de la cellule ACE dont les poids sont rapidement saturés et qui, après quelques centaines d'itérations, ne fournit plus au contrôleur qu'un signal de renforcement dégradé. De plus, compte tenu des temps de calcul nécessaires à l'aboutissement d'une série de tests (plus de cinq jours de calcul en phase d'apprentissage sur un 486DX/66[114]), il ne nous a pas été possible d'optimiser les paramètres du réseau. Il est probable que des réglages appropriés permettraient d'améliorer les performances de l'apprentissage, en particulier quant au nombre de réseaux convergents.

---

[114] Le temps simulé total (durée de l'ensemble des essais pour les dix tests) est de 1 789 287 itérations, soit près de dix heures. Le temps d'apprentissage est donc dû avant tout au nombre d'essais effectués. Compte tenu des conditions dans lesquelles ces tests ont été effectués (utilisation d'un langage interprété), le rapport temps simulé/temps réel obtenu (environ un douzième) permet d'envisager un apprentissage en temps réel si le réseau est codé dans un langage compilé plus performant (type « C »).