5. APPLICATIONS

Le système INSS a été testé sur 4 problèmes différents. Nous avons d'abord choisi d'appliquer INSS à un problème académique (*toy problem*), les problèmes du Moine [THR 91], afin de pouvoir mieux mesurer les résultats de son utilisation et de comparer sa performance par rapport à d'autres systèmes d'apprentissage automatique. Le fait d'avoir choisi un problème artificiel, nous a permis de mieux exploiter l'application, car le problème était bien connu et nous avions le contrôle sur toutes les données du problème. Ensuite, nous sommes passés à un autre problème plus complexe et réel, l'aide au diagnostic médical des comas toxiques [AMY 97]. Ce problème est beaucoup plus difficile à traiter, et d'ailleurs il n'a pas toujours été résolu de façon très satisfaisante. C'est sa difficulté qui faisait son intérêt.

Les autres applications sont très différentes. Les deux premières applications étaient des problèmes de classification 'classiques', tandis que les deux autres applications abordent des sujets plus spécifiques. La troisième application, le problème de la balance, a été étudié par des chercheurs afin de modéliser le développement cognitif chez l'enfant [SCH 94a]. Pour nous, l'intérêt de ce problème était plutôt lié au type de règles symboliques (règles de haut niveau) utilisées pour coder les connaissances sur le sujet. Finalement, la dernière application a été l'utilisation d'INSS dans un problème de contrôle d'un robot autonome, le Khepera [MON 93]. Cette application nous a permis d'étudier certains aspects liés au contrôle sensori-moteur d'un robot, d'abord sur un simulateur, puis sur un vrai robot.

En utilisant INSS sur ces quatre applications, nous avons essayé de mettre en valeur les propriétés de notre système, ainsi que de valider l'implémentation qui a été faite.

Dans les sections qui suivent, nous allons décrire plus en détails ces applications et les résultats obtenus en utilisant INSS. Avant de passer aux applications, nous voulons souligner d'une part que l'évaluation des systèmes connexionnistes est une tâche difficile étant donné le manque d'une méthodologie adaptée[PRE 94a] ; d'autre part que l'évaluation des systèmes hybrides comme INSS est encore plus difficile, car il faut trouver des problèmes comportant une base de règles et une base d'exemples. Les quatre applications sur lesquelles nous avons travaillé, sont composées de problèmes qui englobent des connaissances théoriques (qui peuvent êtres exprimées par des règles du type accepté par INSS) et des connaissances empiriques (base d'exemples).

5.1 LES PROBLEMES DU MOINE (THE MONK'S PROBLEMS)

Nous avons choisi d'utiliser un problème très connu dans le domaine de l'apprentissage automatique, car il a été conçu pour l'évaluation de différents algorithmes. Les problèmes du Moine (*Monk's problems*) ont été proposés dans la 2ème école d'été sur l'apprentissage automatique, afin de comparer différentes méthodes, telles que : les arbres de décision (ID3, ID5R, TDIDT, Assistant Professional, PRISM), les algorithmes de la famille AQ (AQ15, AQ17, AQR, CN2), les méthodes de programmation logique inductive (ILP - mFOIL), les méthodes de construction de prototypes (CLASSWEB, COBWEB), et les réseaux de neurones artificiels (Rétro-Propagation, Cascade-Correlation) [THR 91]. Les "problèmes du Moine" sont ainsi appelés simplement parce que le groupe de travail qui a travaillé sur ces problèmes s'est réuni dans un monastère pendant le déroulement de l'école.

5.1.1 Description du Problème

Ce problème est constitué par trois sous-problèmes de classification de 'robots artificiels' : monk1, monk2, monk3. Etant donné une base d'apprentissage, nous essayons d'apprendre la fonction qui nous permettra de classifier correctement les exemples dans deux catégories : ceux qui appartiennent à la classe Monk (c'est un robot) et ceux qui n'appartiennent pas à cette classe (ce n'est pas un robot). La Figure 57 donne une description des six variables d'entrées applicables aux trois sous-problèmes.

Variables d'entrée:

```
$Features:
        HEAD_SHAPE  :  nominal # 3  : Octagon, Square, Round;
        BODY_SHAPE  :  nominal # 3  : Octagon, Square, Round;
        IS_SMILING    :  nominal # 2  :  No, Yes;
        HOLDING        :  nominal # 3  : Flag, Balloon, Sword;
        JACKET_COLOUR  :  nominal # 4  : Blue, Green, Yellow, Red;
        HAS_TIE          :  nominal # 2  :  No, Yes.
$End_Feat.
```

Règles Symboliques - Monk1:

```
$Rules:
  Monk1 <- HEAD_SHAPE(Round), BODY_SHAPE(Round);                 Règle I
  Monk1 <- HEAD_SHAPE(Square), BODY_SHAPE(Square);               Règle II
  Monk1 <- HEAD_SHAPE(Octagon), BODY_SHAPE(Octagon);             Règle III
  Monk1 <- JACKET_COLOUR(Red).                                   Règle IV
$End_rules.
```

Figure 57 - Description du problème du Moine: "Monk's Problem 1"

Le problème Monk1 est constitué par une base d'apprentissage de 124 exemples. Cette base est obtenue à partir de la base complète qui comporte 432 exemples, c'est-à-dire toutes les possibilités de combinaison des valeurs des six entrées. Dans le Monk1, il n'y a pas d'exemples classés incorrectement, et la règle que nous devons apprendre est (voir aussi Figure 57) :

Si Head_Shape = Body_Shape ***ou*** Jacket_Color = Red

Alors Monk1 est vrai (c'est un robot)

Le problème Monk2 est constitué par une base d'apprentissage de 169 exemples, sans exemples mal classés. La règle codée dans les exemples est :

Si deux et seulement deux des attributs d'entrée prennent leurs premières valeurs

Alors Monk2 est vrai (c'est un robot)

Le problème Monk3 est constitué par une base d'apprentissage de 122 exemples, dont 5% des cas sont des exemples mal classés (la classe qui leur est attribuée est incorrecte). La règle codée dans les 'bons' exemples est :

Si (Jacket_Color = Green ***et*** Holding = Sword) ***ou***

(Jacket_Colort ≠ Blue ***et*** Body_Shape ≠ Octagon)

Alors Monk3 est vrai (c'est un robot)

5.1.2 Les Résultats Expérimentaux obtenus avec INSS

Dans nos expériences, nous nous sommes concentrés sur le premier ensemble de règles et d'exemples proposés dans les problèmes du Moine - le problème Monk1. La Figure 57 montre la définition symbolique complète de ce problème dans le langage accepté par NeuComp. Les deux autres sous-problèmes, le Monk2 et Monk3, ont été aussi étudiés, mais cela a été fait plutôt afin de tester le bon fonctionnement des algorithmes que nous avons réalisé (Nous avons aussi utilisé d'autres problèmes d'apprentissage tels que le 'Xor' et la 'double spirale' [FAH 90]). Les résultats que nous avons obtenus avec le problème Monk1 servent à montrer les principales caractéristiques de notre système. Nous avons réalise deux groupes principaux d'expériences avec ce problème.

Premièrement, nous avons utilisé INSS afin de mettre en valeur deux de ses propriétés principales : la possibilité d'utiliser et d'intégrer des connaissances théoriques et empiriques ; et la capacité de profiter des connaissances théoriques de départ afin d'accélérer le processus d'apprentissage. Pour faire cela, nous avons pris l'ensemble d'apprentissage composé de 124 exemples (ensemble d'exemples originaux) et l'ensemble de tests de généralisation composé de 432 exemples (ensemble d'exemples originaux qui couvrent tout l'espace d'entrées). Nous avons commencé par la réalisation de l'apprentissage à partir des exemples, en utilisant toute la base d'apprentissage (124 exemples) et après en réduisant cette base à 75% de sa taille originale (93 exemples) et à 50% de sa taille originale (62 exemples). Ensuite, nous avons testé la capacité de généralisation du système en réalisant la compilation de règles sous la forme d'un réseau, et en faisant son activation pour tous les 432 exemples de la base de test. Dans ces expériences, nous avons utilisé tout d'abord 100% des règles (les 4 règles), après 75% des règles (3 règles), et finalement seulement 50% des règles (2 règles). Finalement, nous avons utilisé des combinaisons des partitions de la base d'exemples et de la base de règles, afin de voir le comportement de notre système en présence d'un ensemble incomplet de connaissances théoriques et empiriques. Dans ces dernières expériences, nous avons tout d'abord inséré la base de règles disponible dans un réseau, et ensuite nous avons raffiné les connaissances par apprentissage à partir des exemples disponibles. Les résultats obtenus dans ces expériences, sont présentés dans le Tableau 7, où nous présentons les taux de généralisation (mesurés par l'activation du réseau en utilisant tous les 432 exemples de la base de test) et les nombres d'époques nécessaires pour arriver à ces résultats ('BestEpoch').

Portion des Règles	Portion des Exemples	Généralisation avec les règles (MS)	Généralisation* avec les exemples (MC)	Généralisation* avec INSS (Syst. Hybride)
-	100%	-	100% - 108.2 époques	**100% - 108.2 époques**
-	75%	-	89.21% - 383.4 époques	**89.21% - 383.4 époques**
-	50%	-	70.92% - 303.8 époques	**70.92% - 303.8 époques**
100%	-	100%	-	**100%**
75%	-	83.33%	-	**83.33%**
50%	-	72.22%	-	**72.22%**
75%	100%	83.33%	100% - 108.2 époques	**100% - 59.4 époques**
50%	100%	72.22%	100% - 108.2 époques	**100% - 57.6 époques**
75%	75%	83.33%	89.21% - 383.4 époques	**100% - 57.4 époques**
75%	50%	83.33%	70.92% - 303.8 époques	**100% - 61.2 époques**
50%	75%	72.22%	89.21% - 383.4 époques	**100% - 61.0 époques**
50%	50%	72.22%	70.92% - 303.8 époques	**89.86% - 62.0 époques**

* Les taux de généralisation représentent la moyenne obtenue à partir de 5 tests différents (poids initiaux différents)

Tableau 7 - Problème Monk1 : Utilisations simultanées de règles et d'exemples pour améliorer le taux de généralisation final et la vitesse d'apprentissage

En se basant sur les résultats présentés dans le Tableau 7, nous pouvons affirmer que le système INSS a bien su exploiter la complémentarité entre les connaissances des bases d'exemples et de règles. La Figure 58 montre plus nettement *l'avantage de l'utilisation de notre méthode hybride sur des méthodes isolées (MS ou MC).* L'INSS surpasse toujours les résultats de généralisation obtenus par l'utilisation d'un seul type d'approche. De plus, on peut remarquer dans le Tableau 7 le faible nombre d'époques nécessaires pour l'apprentissage avec INSS. On remarque que *l'apprentissage est toujours plus rapide quand on utilise des connaissances a priori.*

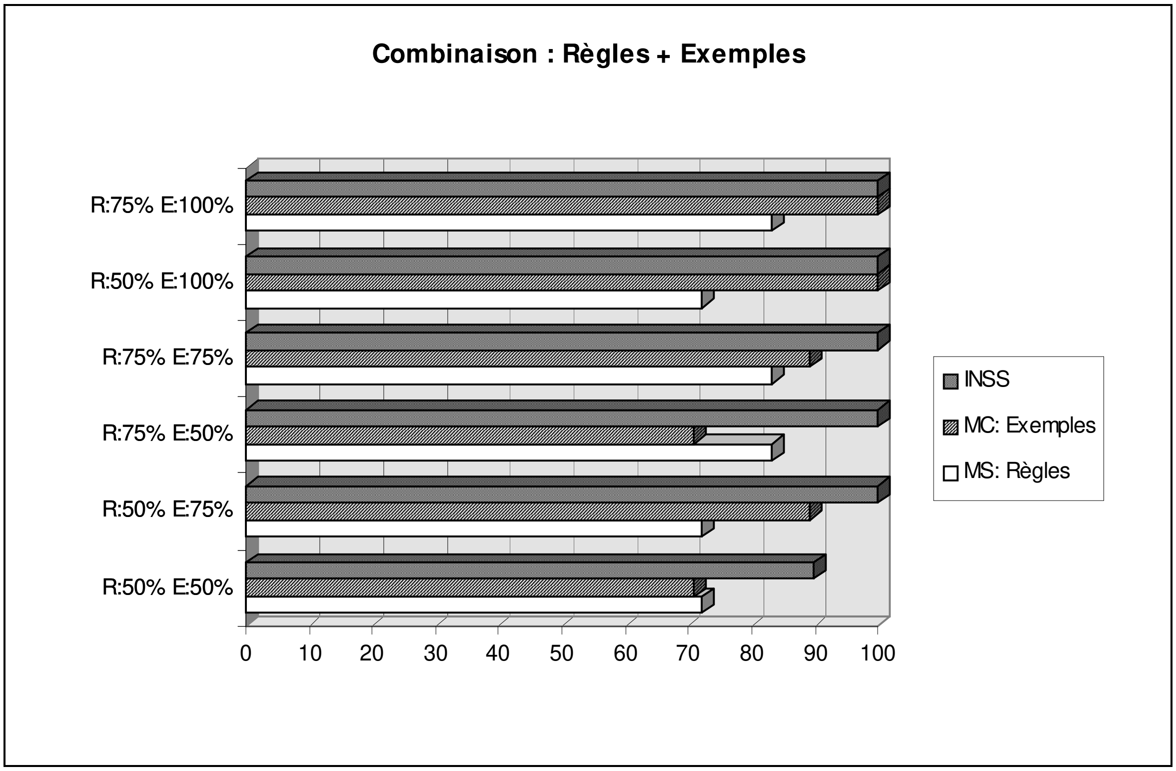


Figure 58 - Monk1 : Taux de généralisations du MS, MC et Système Hybride

Notre deuxième groupe d'expériences avec le problème Monk1 portait sur l'extraction de règles. Nous avons fait des expériences pour savoir si le système était capable de retrouver l'ensemble complet des règles à partir des connaissances incomplètes raffinées par l'apprentissage d'une base d'exemples complète. Notre première expérience a consisté à créer un réseau avec la compilation de 75% des règles (3 parmi les 4 règles disponibles), puis à appliquer la méthode d'explicitation de règles. Le processus d'extraction a été appliqué seulement sur deux unités, la sortie et une unité cachée, car seulement une unité a été ajoutée au réseau pendant l'apprentissage. Nous avons répété ce test pour toutes les différentes configurations des ensembles incomplets de règles : une règle éliminée parmi les quatre disponibles. Dans tous les cas testés*, la méthode d'extraction nous a permis de retrouver la bonne règle qui avait été éliminée de l'ensemble d'origine.* Cette règle éliminée se trouvait bien représentée dans les nouvelles connaissances acquises par le réseau. La Figure 59 montre un exemple des règles obtenues dans une des expériences que nous avons réalisées. Il faut remarquer que les règles extraites sont en fait des règles équivalentes aux règles recherchées.

Règle éliminée : règle III

Monk1 <- Head_Shape (Octagon), Body_Shape (Octagon);

Règles extraites :

New_Unit <- Body_Shape (Round);
New_Unit <- Body_Shape (Square);
New_Unit <- Head_Shape (Round);
New_Unit <- Head_Shape (Square);
New_Unit <- not (Head_Shape (Octagon)), not (Body_Shape(Octagon));

Final_Monk1 <- Monk1_75%;
Final_Monk1 <- not (NewUnit);

Règles après manipulation : simplification de la négations final

New_Unit <- Head_Shape (Octagon)),
not (Head_Shape(Round)), not (Head_Shape (Square)),
not (Body_Shape(Round)), not (Body_Shape (Square));
New_Unit <- Body_Shape (Octagon)),
not (Head_Shape(Round)), not (Head_Shape (Square)),
not (Body_Shape(Round)), not (Body_Shape (Square));

Final_Monk1 <- Monk1_75%;
Final_Monk1 <- NewUnit;

Règles finales :

Nous savons que :

Si Head_Shape (Octagon)
Alors not (Head_Shape (Round)) et not (Head_Shape (Square))

et

Si Body_Shape (Octagon)
Alors not (Body_Shape (Round)) et not (Body_Shape (Square))

Donc, les règles peuvent être simplifiées encore plus :

New_Unit <- Head_Shape (Octagon)), Body_Shape (Octagon)),
Final_Monk1 <- Monk1_75%;
Final_Monk1 <- NewUnit;

Figure 59 - Problème Monk1 : Règles Extraites

Dans une autre expérience, nous avons éliminé 50% des règles (règles III et IV). Après l'apprentissage des exemples (la base d'apprentissage utilisée est toujours la base d'origine composée de 124 exemples), nous avons pu retrouver les deux règles éliminées à travers l'utilisation du module Extract du système INSS.

En conclusion, nous avons réussi a montrer avec ce dernier groupe d'expériences que le système INSS est capable de prendre une base de connaissances initiales (règles), de la convertir dans un réseau, d'ajouter de connaissances par apprentissage, et enfin d'expliciter les nouvelles connaissances acquises.

5.1.3 Discussion Finale sur l'étude des Problèmes du Moine

Nous avons montré avec les expériences réalisées sur les problèmes du Moine que :

- Le système INSS donne des très bons résultats quand de sont utilisation avec ces problèmes. Nous pouvons comparer nos résultats avec ceux qui ont été obtenus dans l'école européenne d'apprentissage automatique en utilisant d'autres méthodes [THR 91]. Notre système est au moins tout aussi performant que les autres systèmes étudiés dans cette école (il arrive à dépasser plusieurs méthodes). De plus, il offre tous les avantages de l'utilisation d'une approche hybride neuro-symbolique;

- Nous avons montré ici que la combinaison des connaissances théoriques (règles) et des connaissances empiriques (exemples), en utilisant une approche hybride, nous permet d'améliorer les performances du système. Le système INSS est capable d'utiliser les différentes connaissances disponibles sur le problème de façon à accélérer le processus d'apprentissage et à augmenter sa capacité de généralisation par rapport à l'utilisation d'une seule source de connaissances;

- Nous avons montré que notre système est capable d'expliciter les nouvelles connaissances acquises à travers l'extraction de règles. INSS est capable de recevoir des connaissances initiales théoriques décrites par des règles, de les raffiner par apprentissage et de trouver les nouvelles connaissances acquises. Dans nos expériences, nous avons montré qu'il est possible de retrouver les règles que nous avons intentionnellement effacées de l'ensemble complet de connaissances sur le problème;

- Nos expériences montrent que le système INSS est un outil qui possède les principales caractéristiques nécessaires à un système d'acquisition constructive de connaissances : l'insertion de connaissances a priori sur le problème, le raffinement des connaissance avec la possibilité de faire évoluer la structure des réseaux, et finalement l'extraction des nouvelles connaissances acquises.

Une description complémentaire des expériences que nous avons réalisée sur l'apprentissage et l'extraction de règles dans les problèmes du Moine, peut être obtenue dans d'autres publications corrélées avec cette thèse [OSO 95a, DEC 95, DEC 96].

5.2 DIAGNOSTIC MEDICAL - COMAS TOXIQUES

Le système INSS a été testé sur l'application médicale : aide au diagnostic des comas toxiques induits par psychotropes. Après avoir décrit le problème, nous présenterons les résultats obtenus avec l'utilisation de notre système.

5.2.1 Description du Problème

Quand un patient comateux est admis à l'unité de traitement intensif, le médecin fait une tentative de diagnostic préalable basé sur certains paramètres cliniques et biologiques. Ce diagnostic permet d'appliquer des soins initiaux urgents au patient, en attendant la confirmation (ou l'infirmation) postérieure du diagnostic à l'aide d'examens toxicologiques. Etant donné l'état critique dans lequel peuvent se trouver les patients, et compte tenu du fait que les examens d'analyse toxicologique peuvent tarder à être obtenus, ce diagnostic préalable doit impérativement être fait très rapidement. L'utilisation d'un système expert d'aide au diagnostic peut être très utile dans ce type de situation.

En relation avec ce problème, il faut souligner que :

- il n'existe pas aujourd'hui de modèle complet décrivant à l'aide de règles les connaissances sur ce domaine de diagnostic;

- Le problème consiste en une tâche de classification. On donne en entrée les symptômes du patient, en sortie on essaie d'obtenir une liste des substances qui ont été probablement ingérées par la personne;

- Nous pouvons disposer d'exemples obtenus à partir des cas préalablement traités, et dans lesquels le diagnostic a été confirmé ou invalidé par des examens toxicologiques (base de données supervisées);

- Notre but n'est pas de remplacer le médecin (celui-ci ne pourra jamais être remplacé par un système automatisé), mais de créer un outil *d'aide* au diagnostic. Les systèmes experts représentent l'image d'un "conseiller virtuel".

5.2.2 Description des données disponibles sur le problème

La base de cas a été conçue par le docteur Vincent Danel et Philippe Cony à l'unité de toxicologie clinique dans le Centre Hospitalier Universitaire de Grenoble - CHU Grenoble. Les études réalisées par le docteur Danel ont été faites en partie dans le cadre de sa thèse [DAN 97]. La base de cas qu'il a mise à notre disposition est une base de 505 cas bien analysés. Un cas est constitué de deux parties :

- La description du problème : le problème est décrit par 13 paramètres ou symptômes qui ont été sélectionnés par les médecins comme des attributs significatifs par rapport au diagnostic à effectuer (paramètres médicalement pertinents, et si possible discriminants). De plus, tous ces paramètres peuvent être obtenus immédiatement dès l'admission du patient. Nous présentons la liste de définition des 13 attributs dans le Tableau 8;

- Le psychotrope ou la combinaison de psychotropes étant absorbés par le patient : il existe 28 combinaisons différentes des 7 substances toxiques que nous nous proposons d'identifier. La liste des 7 substances (psychotropes + alcool) et de leurs combinaisons est donnée dans le Tableau 9. Dans cette figure, nous fournissons les abréviations utilisées (nomenclature utilisée dans la base de données), ainsi que le nombre de cas de la base où chaque substance a été détectée.

	Symptômes	***généraux***	
	Variable	**Nature**	**Valeurs**
1	Sexe - SEXE	Discrète	{féminin, masculin} - 0/1
2	Température - TEMP	Continue	{ ... }
	Temp_Low	Discrète	0/1
	Temp_Normal	Discrète	0/1
	Temp_High	Discrète	0/1
	Symptômes	*neurologiques*	
3	Etat - CALME	Discrète	{calme, agité} - 0/1
4	Photomoteurs - PHOTO (réaction des pupilles)	Discrète	{présent, absent} - 0/1
5	Regard - REGARD	Discrète	{normal, errance} - 0/1
6	Etat des Pupilles - PUPIL	Discrète	{myosis, intermédiaire, mydriase}
	Pupil_myosis	Discrète	0/1
	Pupil_intermed	Discrète	0/1
	Pupil_mydriase	Discrète	0/1
7	Tonus musculaire - TONUS	Discrète	{hypertonie, hypotonie}
8	Réflexe des tendons - ROT	Discrète	{vifs, diminués}
	Symptômes	*cardiologiques*	
9	Pression Artérielle - PAS	Continue	{...}
	PAS_Low	Discrète	0/1
	PAS_Normal	Discrète	0/1
	PAS_High	Discrète	0/1
10	Fréquence Cardiaque - FC	Continue	{...}
	FC_Low	Discrète	0/1
	FC_Normal	Discrète	0/1
	FC_High	Discrète	0/1
11	Mesure ECG - QRS	Continue	{...}
	QRS_Normal	Discrète	0/1
	QRS_Prolonged	Discrète	0/1
12	Intervalle QT - QT	Discrète	{normal, allongé} - 0/1
		Urine	
13	Globe vésical - GLOBE	Discrète	{oui, non} - 0/1

Tableau 8 - Les 13 paramètres d'entrée décrivant un cas d'intoxication

	Abréviations	*Subst. Toxique*	*Occurrence*
1	adt, a	antidépresseurs tri-cycliques	265
2	B	barbituriques	86
3	ben, b	benzodiazépine	414
4	c	carbamate	68
5	p	phénothiazine	130
6	m	morphine	13
7	E	alcool	137
Combinaisons			
a (25), aBb (12), aBbp (9), ab (104), abc (11), abm (5), abp (38), ap (5), B (8), Bb (23), Bbcp (5), Bcp (13), ben (37), bc (24), bm (8), bp (31), c (5), p (5), Ea (12), Eab (36), Eabp (8), E (12), EB (6), EBb (5), EBbp (5), Eb (32), Ebc (10), Ebp (11)			

Tableau 9 - Les différentes substances toxiques avec les nombres respectifs d'occurrences dans la base

La base de données que nous avons utilisée initialement comptait 13 attributs, dont 8 sont des attributs binaires, 1 est un attribut ternaire et 4 autres sont des attributs a valeurs continues. Une autre base de données a été obtenue par discrétisation des attributs. On obtient ainsi 21 attributs tous binaires. La base de 13 attributs d'entrée est la base *Coma13* , la base avec 21 attributs d'entrée est la base *Coma21*.

Les données sur ce problème ont été étudiées au cours des recherches menées par :

- Les équipes intégrant le projet ESPRIT MIX [AMY 97, HIL 94a], dont l'équipe Réseaux d'Automates du laboratoire Leibniz;
- Le Docteur V. Danel, dans le cadre de sa thèse [DAN 97];
- Maria Malek, dans le cadre de sa thèse sur les systèmes hybrides (système PROBIS : CBR+ANN);
- Les personnes attachées au projet SYMEDIA (projet en cours avec la participation des Laboratoires Leibniz et TIMC de l'IMAG, du C.E.A de Grenoble et du C.H.U. de Grenoble). F. Osório fait partie ce groupe de recherche.

Plusieurs techniques d'apprentissage automatique ont été appliquées sur ces données dont : les graphes de décision et le sytème SIPINA (V.Danel), le système hybride Probis (M.Malek) et le système hybride INSS (F.Osório). Initialement, nous avons suivi le même type de procédure adopté par M. Malek pour traiter ce problème. Dans sa thèse, M. Malek démontre que la base de données qui comporte 28 classes différentes en sortie, n'est pas traitable telle qu'elle. Les relations entre le nombre de cas, le nombre d'attributs d'entrée et le nombre de classes ne permettent pas de bien discerner les classes, car "l'ensemble des cas dont nous disposons ne couvre pas le domaine, et n'est pas suffisant pour définir les frontières entre les 28 classes différentes" [MAL 96b]. M. Malek a proposé de décomposer le problème en plusieurs sous-tâches, afin de faciliter sa résolution. Donc, au lieu d'essayer de classer les 505 cas dans 28 classes, nous avons essayé d'identifier, pour chacune des 7 substances toxiques, si elle a été absorbée ou non par le patient. Nous avons ainsi transformé un problème de classification à 28 classes en 7 sous-problèmes, chacun contenant une sortie unique qui indique la présence ou l'absence d'une des 7 substances toxiques.

5.2.3 Les Résultats Expérimentaux obtenus avec INSS

Nous avons réalisé différentes expériences avec les données disponibles sur ce problème. Notre première expérience a été faite avec la base *Coma13*, afin de comparer les performances (capacité de généralisation) du système INSS avec celles des autres systèmes. Nous avons repris les résultats publiés par Malek dans sa thèse [MAL 96b], et nous avons reproduit les mêmes expériences avec INSS. Le Tableau 10 présente les résultats d'apprentissage de la base Coma13 par rapport au taux de généralisation obtenu pour chacune des 7 substances toxiques. Le système INSS est comparé avec les méthodes K-PPV (K-plus-proches-voisins), C4.5 (arbres de décision) et Probis (système hybride CBR+ANN).

Classe \ Méthode	K-PPV	C4.5	ProBIS	INSS
Alcool -E	66.56%	65.40%	68.94%	**74.50%**
ADT - a	55.39%	55.26%	57.63%	**60.79%**
Barbituriques - B	65.65%	63.32%	64.60%	**82.45%**
Benzodiazepines - b	62.37%	64.34%	63.95%	**83.37%**
Carbamates - c	81.58%	**87.64%**	84.87%	87.28%
Morphine - m	97.23%	97.50%	**97.97%**	97.88%
Phenothiazines - p	66.45%	71.26%	68.95%	**75.36%**

Tableau 10 - Comparaison des taux de généralisation après apprentissage

Dans ces expériences, nous avons utilisé les données et procédures suivantes :

- La base Coma13 est composé de 505 cas décrits à l'aide de 13 attributs d'entrée et un seul de sortie. Les 505 cas sont dupliqués dans 7 fichiers, chacun contenant une sortie indiquant le diagnostic relatif à une des 7 substances toxiques à identifier;

- Nous avons divisé aléatoirement les 505 cas disponibles en deux parties : un fichier d'apprentissage avec 70% des cas (354 exemples) et un fichier de test de généralisation avec les 30% restants des cas (151 exemples);

- Les résultats obtenus avec INSS représentent une moyenne de 10 tests différents. Chaque test a été réalisé à partir de la création d'un réseau initialisé avec des valeurs des poids choisies aléatoirement (différents initialisations pour chacun des tests);

- Les résultats indiqués dans les colonnes K-PPV, C4.5 et Probis sont une reproduction des meilleurs résultats obtenus pour chacune des méthodes, tels que nous les trouvons publiés dans la thèse de M.Malek [MAL 96b]. Ces résultats ont été obtenus en utilisant des fichiers d'apprentissage et de test réalisés à partir de la même base de données d'origine (Coma13) et repartis de la même façon que dans nos tests (70% apprentissage - 30% test). La seule différence est que les résultats finaux représentent une moyenne de seulement 5 tests différents;

- Dans tous nos tests avec INSS, nous avons utilisé la méthode d'apprentissage CasCor, sans insertion de connaissances a priori. Le critère d'arrêt de l'apprentissage était : soit après 1200 époques, soit avec une erreur de sortie inférieure à 0.05 (StopErr). Nous avons réalisé une époque de test après chaque époque d'apprentissage afin de trouver quelle a été la meilleure époque (BestEpoch) et le meilleur taux de généralisation obtenu pendant l'apprentissage.

Une description plus détaillée des résultats obtenus avec INSS dans cette expérience peut être trouvée dans le rapport MIX sur l'application médicale [AMY 97]. La première conclusion sur ces expériences est que le système INSS a une performance très convenable par rapport à ce problème. La performance d'INSS est remarquable vis-à-vis des autres techniques. On peut constater aussi que cette application est assez complexe, puisque pour certaines classes toutes les méthodes appliquées n'ont pas réussi a avoir des résultats plus satisfaisantes. Ainsi pensons-nous avoir réussi à montrer que le système INSS peut être appliqué à des problèmes réels complexes tout aussi bien qu'à des problèmes académiques.

Toutefois, les résultats que nous venons de présenter ne permettent pas de mettre en valeur les autres propriétés du système, qui font la différence entre notre approche et les autres techniques classiques d'apprentissage automatique. C'est pourquoi nous avons essayé d'approfondir nos recherches sur ce problème. Notre but était aussi de mieux comprendre pourquoi les résultats obtenus n'étaient pas très satisfaisants dans certains cas. Notre démarche a été la suivante : faire l'extraction de règles après l'apprentissage, afin de pouvoir valider les connaissances acquises par le réseau auprès d'un expert du domaine.

Comme l'extraction de règles faite par INSS, avec la version actuelle du système, ne supporte que des règles d'ordre 0, nous avons été obligés d'utiliser la base de données *Coma21*. Dans cette base, tous les attributs d'entrée ont été discrétisés (entrées binaires). La base Coma21 est la base de données qui a été choisie pour être testé dans le projet MIX.

Nous avons commencé nos tests avec les données relatives à la classe de psychotrope qui a obtenu le meilleur taux de généralisation - la morphine avec 97.88% de bonnes réponses. De plus, nous savions que les cas de diagnostic positif de cette classe étaient très peu présents dans la base de données (seulement 13 cas parmi les 505). Nous avions des doutes sur la qualité des réponses du réseau relatives à l'identification de l'absorption de morphine, qui ont été vite confirmés par l'extraction des règles. Voici la règle extraite à partir du réseau préalablement entraîné avec la base Coma21 dans le cas de d'identification de la classe morphine - 'm' :

```
$RULES:
$END_RULES.
```

Elle montre qu'il n'y a pas de règles extraites, c'est-à-dire que le réseau ne va jamais identifier un cas comme appartenant à la classe 'm'. Les réponses du réseau analysé sont toujours négatives par rapport à l'identification de la présence de la morphine. Cela n'a rien d'étonnant, car le petit nombre de cas de la classe 'm' présents dans la base d'apprentissage ne nous permet pas d'arriver à apprendre vraiment à bien les classer. En ce qui concerne le taux de 97.88% de bonnes réponses, il est assez proche du taux de bonnes réponses obtenu si l'on ne donne que des réponses négatives, car la base est composée par 97.43% de cas négatifs.

Ensuite, nous sommes passés à l'application des outils de validation de connaissances disponibles dans INSS. Nous avons donc sorti le nombre de cas 'faux_0' et de 'faux_1' obtenus en activant le réseau après l'apprentissage avec toute la base de 505 cas. Le résultat était évident : toutes les erreurs produits par le réseau étaient d'un seule type - 13 cas de 'faux_1'. Le fichier 'data.err' créé par INSS ne contenait que des cas positifs de la classe 'm'. Le réseau a toujours répondu avec une sortie négative, même pour les 13 cas positifs existants dans la base de données complète. Il n'y a pas de doute, le meilleur taux de bonnes réponses qu'on peut avoir pour la classe 'm' avec l'apprentissage de la base Coma21 est celui d'une réponse unique toujours négative.

Classe	Cas Positifs	Cas Négatifs	X	Non(X)
Alcool - E	137	368	27.12%	**72.88%**
ADT - a	265	240	**52.47%**	47.53%
Barbi - B	86	419	17.02%	**82.98%**
Benzo - b	414	91	**81.98%**	18.02%
Carba - c	68	437	13.46%	**86.54%**
Morph - m	13	492	2.57%	**97.43%**
Pheno - p	130	375	25.74%	**74.26%**

Tableau 11 - Distribution des cas dans la base de données (Coma13/Coma21)

Nous avons voulu aller encore plus loin dans les expériences faites à partir de cette base. Tout d'abord, nous avons établi une liste du nombre de cas positifs et négatifs de chaque classe (voir Tableau 11). Ensuite, nous avons constaté que le taux de bonnes réponses de INSS pour la classe 'b' (83.37%) était assez proche du pourcentage de cas positifs liés à cette classe ('benzo' = 81.98%). Nous avons donc fait l'extraction de règles à partir du réseau Coma21-Benzo. La règle obtenue est la suivante :

```
$RULES:
        Benzo <- .
$END_RULES.
```

Elle indique que la conclusion 'Benzo' va être toujours positive, car il n'y a pas de conditions qui s'imposent pour que la sortie soit activée. Tous les cas seront classés comme des cas appartenant à la classe 'b' (benzo positif). On peut constater dans le Tableau 11 que cette classe possède une majorité d'exemples de cas positifs, par opposition à la classe 'm' que nous avons analysée précédemment. Comme prévu, le réseau entraîné avec la base Coma21-Benzo ne présente que des sorties du type 'Faux_0' pour tous les 505 cas de la base.

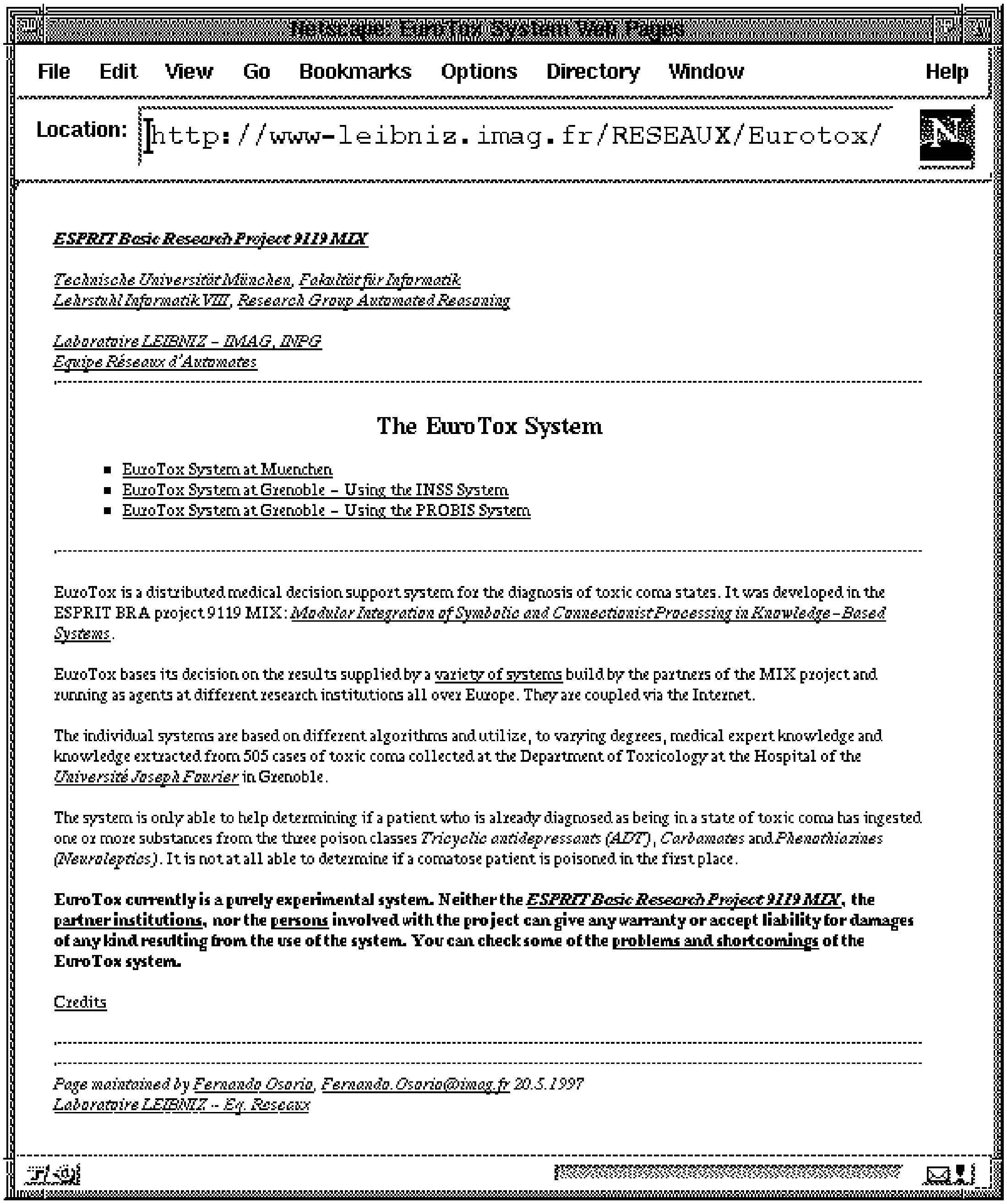
Pour conclure nos expériences sur l'extraction de règles dans cette application, nous avons choisi de travailler sur un réseau qui a appris la base Coma21-Adt (un des trois psychotropes choisis pour être étudié plus en détails dans le projet MIX [AMY 97]). Après application des méthodes de simplification du réseau, d'extraction de règles, et de sélection des règles obtenues, nous avons obtenu l'ensemble de règles présenté ci-dessous :

```
%%
%% Extracted rules
%% Score over 505 examples: 64.35% correct answers [19 rules]
%%
$Rules:

ADT <- GLOBE, PUPIL_INTERMED, not(FC_LOW), not(PUPIL_MYOSIS), not(REGARD), not(TEMP_HIGH),
       not(SEXE) ;
ADT <- QTC, PUPIL_INTERMED, not(FC_LOW), not(PUPIL_MYOSIS), not(REGARD), not(TEMP_HIGH) ;
ADT <- GLOBE, TEMP_LOW, not(PUPIL_MYOSIS), not(REGARD), not(TEMP_HIGH) ;
ADT <- QTC, TEMP_LOW, not(FC_LOW), not(REGARD), not(TEMP_HIGH), not(SEXE) ;
ADT <- QTC, TEMP_LOW, not(REGARD), not(TEMP_HIGH), not(TEMP_NORMAL), not(SEXE);
ADT <- QTC, TEMP_LOW, not(FC_LOW), not(PUPIL_MYOSIS), not(REGARD) ;
ADT <- QTC, TEMP_LOW, not(PUPIL_MYOSIS), not(REGARD), not(TEMP_HIGH);
ADT <- QTC, TEMP_LOW, not(PUPIL_MYOSIS), not(REGARD), not(TEMP_NORMAL);
ADT <- PUPIL_INTERMED, TEMP_LOW, not(FC_LOW), not(PUPIL_MYOSIS), not(REGARD), not(TEMP_NORMAL),
       not(SEXE);
ADT <- PUPIL_INTERMED, TEMP_LOW, not(PUPIL_MYOSIS), not(REGARD), not(TEMP_HIGH) ;
ADT <- PUPIL_INTERMED, TEMP_LOW, not(PUPIL_MYOSIS), not(REGARD), not(SEXE) ;
ADT <- QTC, not(FC_LOW), not(PUPIL_MYOSIS), not(REGARD), not(TEMP_HIGH), not(TEMP_NORMAL) ;
ADT <- QTC, not(PUPIL_MYOSIS), not(REGARD), not(TEMP_HIGH), not(SEXE) ;
ADT <- PUPIL_MYDRIASE, not(FC_LOW), not(PUPIL_MYOSIS), not(REGARD), not(TEMP_HIGH) ;
ADT <- PUPIL_INTERMED, not(PUPIL_MYOSIS), not(REGARD), not(TEMP_HIGH), not(TEMP_NORMAL),
       not(SEXE) ;
ADT <- TEMP_LOW, not(FC_LOW), not(PUPIL_MYOSIS), not(REGARD), not(TEMP_HIGH) ;
ADT <- TEMP_LOW, not(PUPIL_MYOSIS), not(REGARD), not(TEMP_HIGH), not(TEMP_NORMAL) ;
ADT <- TEMP_LOW, not(PUPIL_MYOSIS), not(REGARD), not(TEMP_HIGH), not(SEXE) ;
ADT <- TEMP_LOW, not(PUPIL_MYOSIS), not(REGARD), not(TEMP_NORMAL), not(SEXE).

$End_rules.
```

Figure 60 - Règles obtenues à partir d'un réseau (apprentissage de la base Coma21-adt)

Les règles obtenues à partir du réseau entraîné avec la base Coma21-Adt ont été présentées à un expert du domaine. Selon lui, ces règles peuvent être considérées comme un ensemble de connaissances valables capturant certaines relations implicites entre les symptômes du patient et le psychotrope en question. De plus, nous avons compilé les règles présentées sur la Figure 60 sous la forme d'un réseau, et ensuite nous avons activé ce réseau avec la base de 505 cas, obtenant un taux de 64.35% de bonnes réponses (résultat supérieur à celui obtenu avec les autres méthodes d'apprentissage automatique, cf. Tableau 10).

Une fois terminées les expériences sur l'extraction de règles, nous avons fait une dernière expérience avec la base de données disponible. Les résultats antérieurs nous ont amenés à mettre en question les attributs d'entrée choisis (sont-ils suffisants ?) et la division du problème en 7 sous-problèmes (sont-ils vraiment indépendants ?). Nous avons fait une expérience simple : prendre la base de données Coma21-Adt et ajouter les 6 autres sorties des autres classes comme des entrées du réseau. Ainsi, nous avons obtenu un réseau avec 27 entrées (les 21 symptômes + les 6 sorties : E, B, b, c, m, p) et une seule sortie (Adt). Notre but était de vérifier si les 6 autres sorties ont une influence significative sur la septième et dernière classe. Les résultats se sont avérés positifs, puisque nous sommes passés d'un taux moyen de réussite de 60.79% à 73.50% (avec le meilleur résultat obtenu égal à 79.40% de bonnes réponses). Les résultats obtenus sont non négligeables, car si l'on ajoute des nouvelles entrées totalement non corrélées à un problème, l'apprentissage du réseau doit se faire avec plus de difficulté. Nos résultats montrent exactement le contraire. Notre conclusion est : soit le réseau a augmenté sa performance à cause des interrelations entre les 7 classes de sortie (sorties non-indépendantes), soit il existe des variables d'entrée cachées qui n'ont pas été explicitées et qui ont été prises en compte pour le diagnostic des 6 autres classes (maintenant fournies comme entrées). Bien que nous n'ayons pas approfondi nos recherches sur ce sujet, les résultats décrits ci-dessus apportent une contribution importante à l'étude du problème et donnent des nouvelles pistes à suivre.


Netscape: EuroTox System Web Pages

File Edit View Go Bookmarks Options Directory Window Help

Location: http://www-leibniz.imag.fr/RESEAUX/Eurotox/

ESPRIT Basic Research Project 9119 MIX

Technische Universität München, Fakultät für Informatik
Lehrstuhl Informatik VIII, Research Group Automated Reasoning

Laboratoire LEIBNIZ - IMAG, INPG
Equipe Réseaux d'Automates

The EuroTox System

- EuroTox System at Muenchen
- EuroTox System at Grenoble - Using the INSS System
- EuroTox System at Grenoble - Using the PROBIS System

EuroTox is a distributed medical decision support system for the diagnosis of toxic coma states. It was developed in the ESPRIT BRA project 9119 MIX: *Modular Integration of Symbolic and Connectionist Processing in Knowledge-Based Systems*.

EuroTox bases its decision on the results supplied by a variety of systems build by the partners of the MIX project and running as agents at different research institutions all over Europe. They are coupled via the Internet.

The individual systems are based on different algorithms and utilize, to varying degrees, medical expert knowledge and knowledge extracted from 505 cases of toxic coma collected at the Department of Toxicology at the Hospital of the *Université Joseph Fourier* in Grenoble.

The system is only able to help determining if a patient who is already diagnosed as being in a state of toxic coma has ingested one or more substances from the three poison classes *Tricyclic antidepressants (ADT)*, *Carbamates* and *Phenothiazines (Neuroleptics)*. It is not at all able to determine if a comatose patient is poisoned in the first place.

EuroTox currently is a purely experimental system. Neither the *ESPRIT Basic Research Project 9119 MIX*, the partner institutions, nor the persons involved with the project can give any warranty or accept liability for damages of any kind resulting from the use of the system. You can check some of the problems and shortcomings of the EuroTox system.

Credits

Page maintained by Fernando Osorio, Fernando.Osorio@imag.fr 20.5.1997
Laboratoire LEIBNIZ -- Eq. Reseaux

Figure 61 - Le Système Eurotox-INSS, Projet MIX

Finalement, nos recherches ont abouti au développement d'un outil expérimental de consultation de notre système à travers le WWW (World-Wide-Web). L'interface avec l'utilisateur été conçue dans le cadre du projet MIX (voir Figure 61). Elle consiste en un formulaire électronique qui permet à l'utilisateur de saisir les symptômes du patient et d'interroger plusieurs systèmes d'aide au diagnostic à travers le réseau Internet. Dans la Figure 62, nous présentons un exemple de l'écran d'interface du système avec le formulaire. On peut accéder à cette interface à l'adresse suivante : *http://www-leibniz.imag.fr/RESEAUX/Eurotox/* .

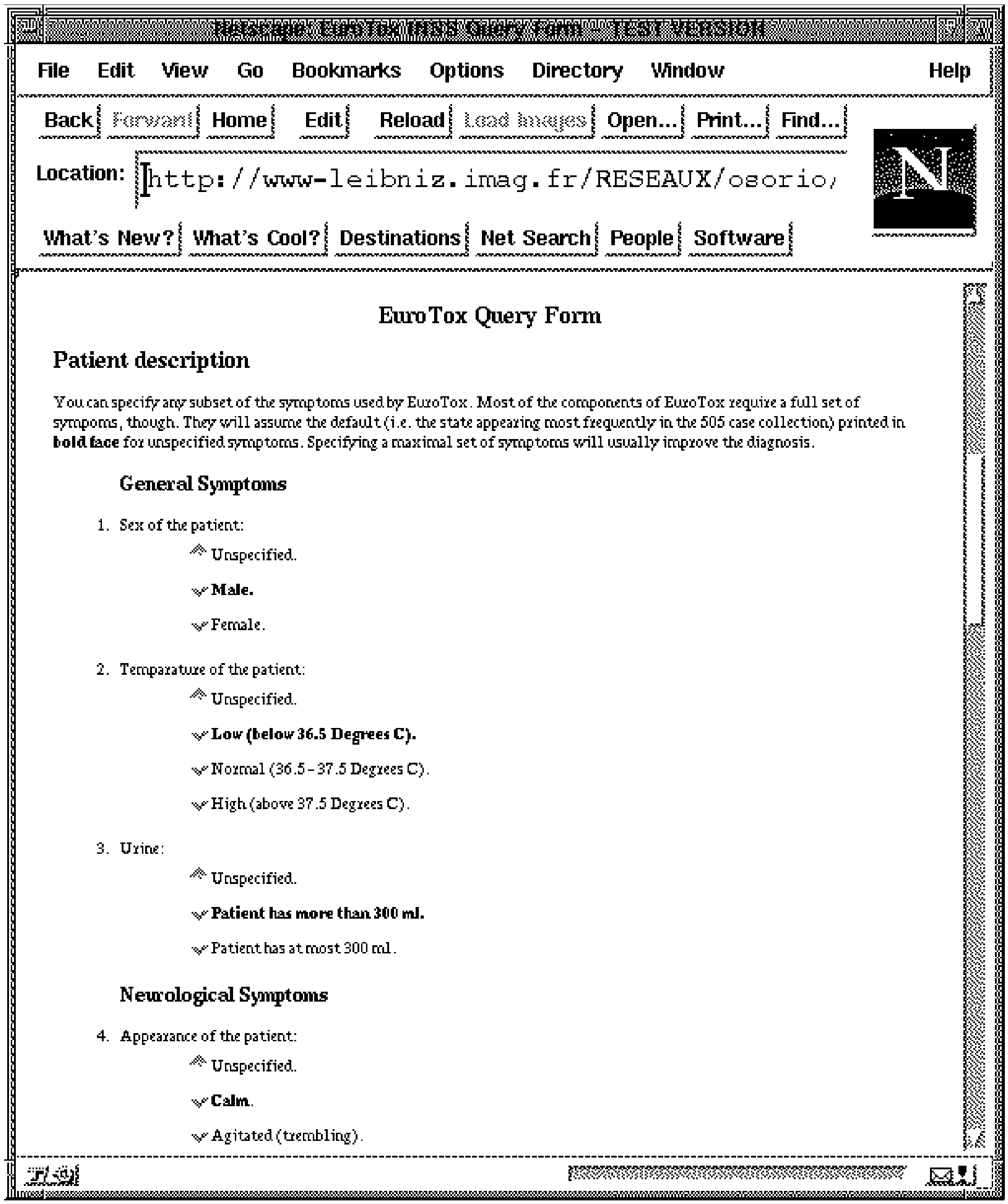


Figure 62 - Formulaire électronique pour la consultation du système Eurotox-INSS

L'intégration des outils du système INSS à cette interface a été faite très rapidement. Cela nous montre que le système INSS est un outil de conception modulaire et qu'il est assez simple de l'intégrer dans différentes applications.

5.2.4 Discussion Finale sur l'Application Médicale

Avec l'utilisation d'INSS, nous avons pu constater que la taille et la nature même de la base de données disponible ne permet pas de faire le diagnostic des 7 classes de substances toxiques d'une façon indépendante. Les outils de validation de connaissance nous ont permis de détecter et d'expliciter certains problèmes relatifs à l'apprentissage de la base disponible. Dans certaines situations, les autres méthodes d'apprentissage automatique ne permettent pas de déceler ces problèmes, qui passent alors inaperçus. L'exemple le plus évident des problèmes rencontrés est le cas où les outils d'aide au diagnostic d'une classe spécifique ont donné toujours une seule et unique réponse. Les taux de bonnes réponses étant satisfaisants, les utilisateurs des systèmes d'apprentissage avaient une tendance à négliger les signes montrant de que ce taux n'était pas une bonne mesure de la qualité du système. Les outils disponibles dans le système INSS, ainsi que la façon de présenter les résultats, amènent l'utilisateur à détecter plus facilement un problème comme celui-ci.

Nous avons montré que le système INSS a une performance très satisfaisante par rapport à d'autres outils d'apprentissage automatique. De plus, notre système s'est montré capable d'exploiter plus profondément les informations disponibles, grâce en particulier à la validation des connaissances et à l'extraction de règles. En utilisant INSS, nous avons pu aussi mettre en question certains choix qui ont été faits par les chercheurs impliqués dans l'étude de cette application. Les résultats obtenus avec INSS montrent qu'on doit étudier ce problème plus en détail, en réalisant une analyse plus approfondie des symptômes choisis en entrées et de la dépendance qui peut exister entre les classes de sortie.

En conclusion, notre approche de l'acquisition constructive de connaissances semble être très intéressant quand on est confronté à des applications réelles. La constante mise en question des connaissances disponibles, leur validation et leur évolution, sont des points très importants dans un système d'acquisition de connaissances. Le problème d'aide au diagnostic médical des comas toxiques est un exemple de problème qui requiert l'utilisation d'une approche comme la nôtre. Il est en effet très complexe, et il présente des aspects qui peuvent être mieux traités en utilisant une approche d'acquisition constructive de connaissances.

5.3 PROBLEME DE LA BALANCE

Le système INSS a été utilisé dans un problème d'apprentissage ayant servi à modéliser le développement cognitif chez les enfants. Il s'agit du problème de la prévision de l'état d'une balance (*balance scale problem*) [SIE 76, MCC 89, SHU 94a]. Nous nous sommes intéressés à ce problème pour les raisons suivantes :

- Il porte sur une tâche de modélisation du développement cognitif chez l'homme (acquisition constructive de connaissances), cette modélisation étant basée sur des données réelles obtenus à partir d'études pratiques avec des enfants Les premières expériences sur le sujet ont été réalisées par des psychologues;

- Le problème d'apprentissage de la prévision de l'état de la balance a été en partie déjà modélisé en utilisant des techniques d'apprentissage automatique (arbres de décision et réseau de neurones);

- Les connaissances sur le domaine sont exprimées à l'aide d'une théorie (ensemble de règles) et de connaissances empiriques (ensemble de cas pratiques);

- Les résultats obtenus avec l'utilisation de l'algorithme CasCor ont attiré l'attention de la communauté des chercheurs du domaine de l'apprentissage automatique. Ces expériences, menées par l'équipe de Thomas Shultz, ont été publiées dans une des revues le plus prestigieuses du domaine de l'apprentissage automatique [SHU 94a].

Notre intérêt par ce problème et les premiers résultats obtenus dans nos expériences, nous ont conduits à proposer un sujet de stage de D.E.A. sur ce thème. L'étude de l'apprentissage du problème de la balance a été réalisée par Gerardo Reyes Salgado dans le cadre de sont stage de

D.E.A. [REY 97] réalisé sous la supervision de F. Osório. G. Reyes a étudié l'apprentissage du problème de la balance, ainsi que le problème de la représentation et de l'extraction de règles de haut niveau, un sujet fortement lié au problème en question.

5.3.1 Description du Problème

Le *problème de la balance équilibrée* (Balance Scale Problem) [SIE 76], [MCC 89], [SHU 94a] trouve son origine dans l'étude psychologique du développement cognitif chez l'homme. Dans les expériences sur lesquelles sont basée cette étude, une balance sur laquelle différents poids sont disposés à différentes distances à gauche et à droite d'un pivot est présentée à des enfants. La tâche est de prédire vers quel côté la balance va pencher quand les appuis qui la maintiennent en équilibre sont retirés. Normalement, de chaque côté la totalité des poids sont placés sur une seule position (voir la Figure 63, où nous montrons une balance de 5 positions et 5 poids maximum).

Siegler a trouvé à propos de cette balance six sous-problèmes différents [SIE 76] :

1. Le problème de la balance *"équilibrée"* : la même quantité de poids est placée à égale distance de chaque côté ;

2. Le problème des *"poids"* : les distances étant égales, la balance penche du côté doté du poids le plus élevé ;

3. Le problème de la *"distance"* : les poids étant égaux, la balance penche du côté où la distance est la plus grande.

4. Le *"conflit-poids"* : on a plus de poids d'un côté, une distance plus grande de l'autre côté, et le côté qui penche est celui qui a le plus grand poids. Les deux autres problèmes, appelés *"conflits",* ont aussi plus de poids d'un côté et une distance plus grande de l'autre côté;

5. Le *"conflit-distance"* : on a un *"conflit",* et de plus le côté qui penche est celui qui a la plus grande distance.

6. Finalement, le *"conflit-équilibrée"* : on a un *"conflit",* et de plus la balance reste équilibrée, c.-à-d. que la balance est en équilibre même si les poids et les distances sont différents de chaque côté.

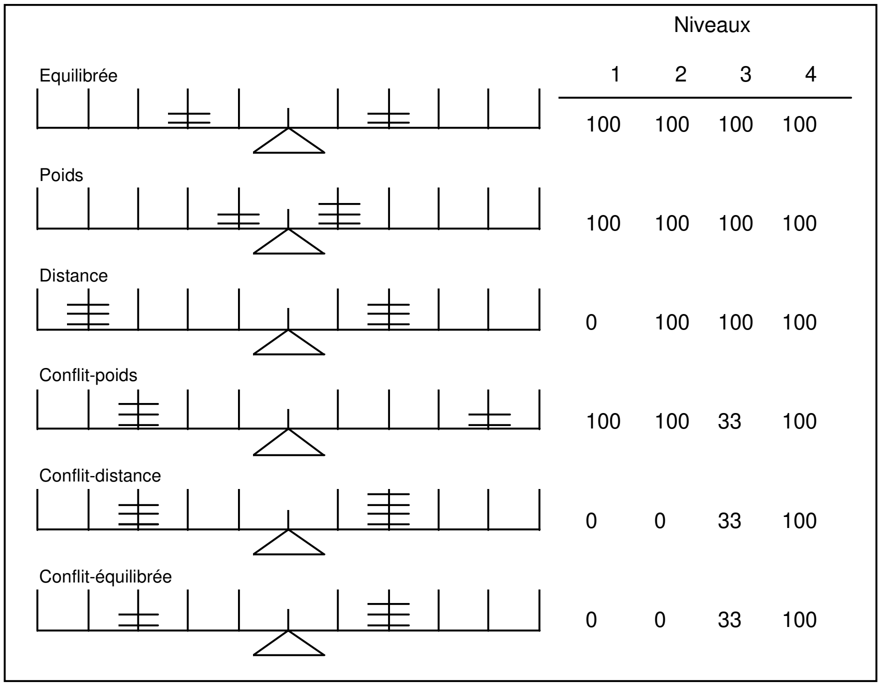


Figure 63 - Les sous-problèmes de la balance et les taux de réussite (en %) pour chacun des quatre niveaux.

Siegler a montré aussi que le développement cognitif chez les enfants se fait à travers 4 niveaux différents (voir Figure 63) basés sur les règles suivantes[2] :

• ***Niveau 1*** : l'enfant utilise seulement les poids pour savoir si la balance est équilibrée : si les poids sont égaux, alors la balance est en équilibre;

Si Pg = Pd, alors la balance est équilibrée ;
Si Pg > Pd, alors la balance penche à gauche ;
Si Pg < Pd, alors la balance penche à droite

---

2 Pg = Poids à gauche ; Pd = Poids à droite ; Dg = Distance à gauche ; Dd = Distance à droite

• ***Niveau 2*** : l'enfant donne priorité au poids, mais il prend en considération la distance quand les poids des deux côtés sont égaux;

Si Pg > Pd, alors la balance penche à gauche ;
Si Pg < Pd, alors la balance penche à droite ;
Si Pg = Pd et Dg > Dd, alors la balance penche à gauche ;
Si Pg = Pd et Dg < Dd, alors la balance penche à droite ;
Si Pg = Pd et Dg = Dd, alors la balance est équilibrée.

• ***Niveau 3*** : l'enfant prend en considération poids et distance, mais il y a confusion quand un côté a un poids plus grand et l'autre côté une distance plus grande;

Si Pg = Pd et Dg = Dd, alors balance est équilibrée ;
Si Pg > Pd et Dg > Dd, alors la balance penche à gauche ;
Si Pg < Pd et Dg < Dd, alors la balance penche à droite ;
Si Pg > Pd et Dg < Dd, alors il y a confusion ;
Si Pg < Pd et Dg > Dd, alors il y a confusion ;

• ***Niveau 4*** : l'enfant multiplie distance par poids de chaque côté et compare les produits : si les produits sont égaux alors la balance est en équilibre, sinon, la balance penche du côté où la valeur du produit est la plus grande.

Si Pg*Dg = Pd*Dd, alors la balance est équilibrée ;
Si Pg*Dg > Pd*Dd, alors la balance penche à gauche ;
Si Pg*Dg < Pd*Dd alors la balance penche à droite.

Chacun de ces groupes de règles représente un des 4 niveaux cognitifs associés au problème. La Figure 63 montre les taux de prédictions associés à chacun des niveaux et par conséquence à chacun des ensembles de règles. Cette expérience, à l'origine conçue pour être utilisée dans l'étude du comportement humain, est devenue une source d'inspiration pour l'étude de la modélisation par ordinateur du développement cognitif humain. Les travaux connexionnistes le plus connus ont été développés par McClelland et par Shultz [MCC 89, SHU 94a].

Les études développées dans le domaine de l'apprentissage automatique ont utilisé des bases d'exemples obtenues généralement à partir d'une base complète de 625 cas (toutes les combinaisons possibles des entrées) décrits de la façon suivante :

Entrées : Pg, Dg, Pd, Dd (4 entrées qui peuvent assumer les valeurs 1,2,3,4 ou 5) où

Pg, poids à gauche ; Dg, distance à gauche ;

Pd, poids à droite ; Dd, distance à droite ;

Sorties : état de la balance (2 sorties qui peuvent assumer les valeurs -0.5, 0.0 ou +0.5)

état 1 = +0.5, -0.5, quand la balance penche à gauche ;

état 2 = 0.0, 0.0, quand la balance est équilibrée, et

état 3 = -0.5, +0.5, quand la balance penche à droite.

Par exemple, une balance qui a un poids égal à 2 unités à la position 3 du côté gauche et un poids égal à 4 unités à la position 2 du côté droite et qui par conséquence penche du côté droit, est représentée par : *2, 3, 4, 2, -0.5, +0.5*

Il faut souligner que les différentes études de modélisation de l'apprentissage du problème de la balance [SCH 96, SHU 94a, MCC 89] auxquels nous faisons référence ici, ont pour principal but la modélisation du développement cognitif, c'est-à-dire l'étude du passage d'un niveau à l'autre pendant l'apprentissage de la tâche (voir les 4 niveaux de Siegler dans la Figure 63).

5.3.2 Les Résultats Expérimentaux Obtenus avec INSS

Nous avons commencé notre étude sur le problème de la balance par une étude du type de règles utilisées dans la définition de chacun des 4 niveaux de Siegler. Nous avons constaté que les règles utilisées ne sont pas de simples règles d'ordre 0 du type utilisé par KBANN. Les règles du problème de la balance sont des règles qui mettent en relation deux entrées, avec des opérateurs de comparaison comme : plus grand, plus petit et égal. Nous avons appelé ce type de règles qui présentent une forte relation entre leurs entrées *règles de haut niveau* [REY 97, REY 97a]. Les règles de haut niveau peuvent présenter des 'opérateurs continus' (e.g. plus grand, plus petit) entre deux entrées, alors que les règles simples d'ordre 0 n'utilisent que des 'opérateurs binaires' entre deux entrées (e.g. et, ou). Le développement des recherches afin d'introduire des règles d'ordre 0^+ (haut niveau) dans notre compilateur NeuComp (voir Section 4.2.1), a été fortement influencé par nos études et expériences sur le problème de la balance et aussi sur la robotique autonome.

Les résultats que nous avons obtenus dans nos tests préliminaires ont indiqué que les réseaux avaient des grosses difficultés pour apprendre parfaitement à résoudre le problème de la balance. Cela nous a amené à diriger nos recherches sur l'apprentissage parfait du problème, c'est-à-dire l'apprentissage du niveau 4 (100% de bonnes réponses dans les six sous-problèmes). Nous étions convaincus qu'il fallait tout d'abord comprendre d'où venait la difficulté de l'apprentissage parfait du niveau 4, avant de passer à d'autres études comme celui du passage d'un niveau à l'autre. Nos premières expériences ont portées sur la capacité d'un réseau à bien apprendre le problème complet. Nous avons voulu vérifier si les réseaux étaient bien capables de représenter des règles de haut niveau, comme celles du niveau 4.

Pour commencer, nous avons essayé de reproduire les expériences réalisées par Shultz avec le problème de la balance [SHU 94]. Shultz a utilisé une base d'exemples d'apprentissage initiale de 100 exemples (choisis parmi les 625 exemples) et il a ajouté un nouvel exemple dans la base à chaque époque d'apprentissage. L'ajout progressif d'exemples a été fait avec le seul but de garantir le passage progressif d'un niveau cognitif à l'autre. Après l'apprentissage, il a fait un test de généralisation avec une base de 24 exemples : ces 24 exemples sont choisis en prenant au hasard 4 exemples de chacun des 6 sous-problèmes différents. L'auteur signale que l'apprentissage se termine après 300 époques parce que le réseau a appris la 4ème règle, en ajoutant en moyenne 2 unités cachées. En résumé, il utilise 400 exemples d'apprentissage, 24 de test et il a besoin de 300 époques d'apprentissage.

Avant de continuer, il faut dire que pour les *règles du niveau 4*, l'apprentissage a été considéré comme acceptable par Shultz à condition d'obtenir *un minimum de 20 bonnes réponses* sur la base de test de 24 exemples ; alors que pour nous, l'intérêt est de trouver un réseau qui va apprendre parfaitement les règles du niveau 4 (100 % de bonnes réponses). C'est pourquoi nous parlerons ici de *"4ème niveau Shultz"* et *"4ème niveau parfait"*.

Pour répéter les expériences de Shultz, nous avons utilisé le système INSS avec l'algorithme d'apprentissage CasCor sans introduire de connaissances a priori. En ce qui concerne l'apprentissage, nous avons choisi au hasard 400 exemples dès le départ (la seule différence par rapport aux expériences de Shultz) et comme base de test 24 exemples, choisis exactement comme dans le travail de Shultz. Nous avons commencé par la réalisation de trois expériences. Premièrement, faire apprendre au réseau le *"4ème niveau parfait"* ; deuxièmement, arrêter l'apprentissage à 300 époques ; troisièmement, réaliser l'apprentissage avec toute la base d'exemples disponible sur le problème (625 exemples). Pour chaque expérience, nous avons répété l'apprentissage 5 fois en prenant des poids initiaux différents. Les résultats sont les suivants :

• Expérience 1 : pour apprendre le *4ème niveau parfait*, il a fallu attendre en moyenne 1544 époques et on a ajouté 14 unités cachées au réseau pendant l'apprentissage. Toutes les simulations ont requis plus de 1000 époques d'apprentissage et plus de 10 unités cachées ont toujours été ajoutées au réseau.

• Expérience 2 : quand le réseau est arrêté à 300 époques (*4ème niveau Shultz*), on obtient une moyenne de 90.84% de bonnes réponses sur les 24 exemples de test, usuellement sans aucune unité ajoutée. Le meilleur taux de généralisation a été obtenu à peu près à la 35ème époque. Dans aucune des simulations nous ne sommes arrivé à un taux de généralisation égal à 100%.

• Expérience 3 : les résultats n'ont pas beaucoup changé par rapport à l'expérience 1. En moyenne, nous avons ajouté 18 unités au bout de 1871 époques, pour pouvoir arriver à un taux de 100% de réponses correctes.

Etant donné ces résultats, nous avons conclut qu'il n'est pas possible d'arriver à l'apprentissage du 4ème niveau parfait sans ajouter plusieurs unités et seulement après un très grand nombre d'époques d'apprentissage. Cela montre bien que les réseaux de type PMC ont des problèmes pour apprendre cette tâche. Ces réseaux n'arrivent pas à bien représenter les règles de niveau 4 étant donné le grand nombre d'unités cachées qu'ils sont obligés d'ajouter au réseau. Il semble que le réseau, capable de représenter les règles plus simples, ne parvient pas à bien apprendre les règles multiplicatives du niveau 4 (Pg*Dg comparé à Pd*Dd). Il est obligé de réaliser un apprentissage par coeur en ajoutant beaucoup d'unités. Nous avons constaté aussi que Shultz n'est probablement jamais arrivé au niveau 4 parfait.

Cette question sur la capacité des réseaux de représenter de façon interne les règles du niveau 4 est-elle si importante ? Nous pensons que oui. Nous nous sommes donc intéressés aux règles de haut niveau, à leur représentation interne dans les réseaux (compilation de règles), et à leur extraction à partir d'un réseau. L'étude sur la capacité de représenter les connaissances décrites par les règles du niveau 4 est très intéressante d'autant plus qu'on n'a jamais compris comment un arbre de décision (algorithme du type C4.5) peut représenter ce type de règles. Usuellement les arbres de décision ne prennent en considération qu'un seul attribut d'entrée à la fois, et pourtant on trouve des travaux sur l'apprentissage du problème de la balance qui s'utilisent de cette méthode [SCH 96]. Notre but n'était pas de vérifier si les arbres de décision peuvent ou non apprendre cette tâche, mais il fallait quand même vérifier si les réseaux que nous utilisons étaient capables d'apprendre les règles du niveau 4.

Dans la suite de nos expériences, nous avons essayé de mieux comprendre les connaissances acquises par les réseaux dans la tâche de la balance, afin de pouvoir proposer des alternatives pour améliorer l'apprentissage des règles du niveau 4. Une fois que nous n'avons pas encore des outils pour extraire des règles de ce type à partir des réseaux, nous avons décidé d'analyser plus en détails les réponses obtenues et de valider le comportement du réseau. Dans le cas de l'expérience 2, nous avons constaté qu'on arrive au 4ème niveau Shultz avec une moyenne de 2 ou 3 exemples (sur les 24) mal classés, ces exemples étant toujours du type "conflit". Avec l'expérience 3, nous avons constaté que le taux de bonnes réponses, après avoir augmenté très rapidement jusqu'à plus de 90% de bonnes réponses, se met ensuite à augmenter très lentement au fur et à mesure qu'on ajoute des nouvelles unités au réseau (voir Figure 64).

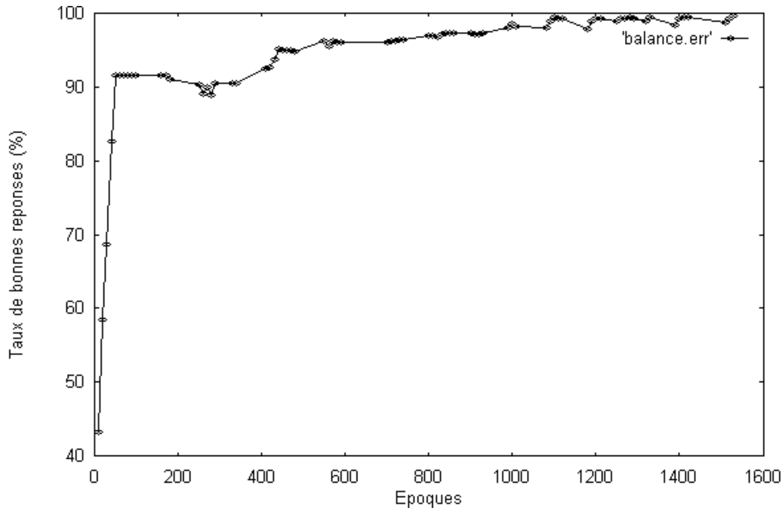


Figure 64 - Exemple de l'évolution du taux de bonnes réponses dans l'apprentissage du problème de la balance

Nous avons conclu que le réseau peut apprend sans problèmes des règles de niveau 1, 2, 3 ; mais il n'arrive pas au vrai niveau 4, car il n'est pas capable de représenter parfaitement la relation multiplicative existante entre les entrées. Nous pensons que le réseau arrive en réalité à un niveau 3.5, c'est-à-dire à un niveau intermédiaire entre les niveaux 3 et 4. Nous supposons que pour arriver à un taux de 100% de bonnes réponses, le réseau va arriver tout d'abord jusqu'au niveau 3.5, puis qu'il réalise un apprentissage par coeur des "cas problème". Des études réalisées avec la participation d'une autre personne de notre équipe, Benjamin Deuker, ont permis de trouver un nouvel ensemble de règles qui semblent représenter ce niveau 3.5. Les règles ont été trouvées par une analyse humaine, car le module d'extraction de règles n'est pas capable d'extraire ce type de règles à partir d'un réseau. Cependant, le système INSS a eu un rôle très important dans l'obtention de ce nouvel ensemble de règles, car il a permis l'analyse détaillée des réponses des réseaux. Les règles du niveau 3.5 sont les suivantes :

Si Pg = Pd et Dg = Dd, alors la balance est équilibrée ; Si Pg = Dd et Pd = Dg, alors la balance équilibrée ; Si Pg > Pd et Dg > Dd, alors la balance penche à gauche ; Si Pg > Dd et Dg > Pd, alors la balance penche à gauche ; Si Pg < Pd et Dg < Dd, alors la balance penche à droite ; Si Pg < Dd et Dg < Pd, alors la balance penche à droite.
* Cette base de règles peut être compilé dans un réseau avec 3 sorties binaires (0/1)

On peut remarquer que ces règles utilisent des comparaisons directes entre poids et distances, ce qui n'a pas de sens d'un point de vue de la physique. Toutefois, cet ensemble de règles permet d'obtenir un taux de bonnes réponses de 87.20% sur la base de 625 cas, et de 91.66% sur une base de 24 cas (seulement deux exemples sont mal classés). Ces résultats sont tout à fait compatibles avec les résultats obtenus par Shultz. Nous n'avons pas encore les moyens pour prouver avec certitude que les règles du niveau 3.5 sont réellement les règles que les réseaux apprennent à la place des règles du niveau 4, mais les expériences que nous avons réalisées nous amènent à considérer sérieusement cette possibilité. Des recherches en cours pourront nous apporter de nouvelles données sur ce sujet.

En ce qui concerne le problème de l'apprentissage des relations multiplicatives comme celles du niveau 4, nous avons conclu que les réseaux de type PMC n'ont pas une capacité de représentation interne de connaissances adéquate pour bien généraliser dans un cas pareil. D'autres expériences que nous avons faites ont montré qu'il est très difficile de réussir l'apprentissage d'une relation multiplicative entre deux entrées avec des réseaux de type PMC. Dans le cas du problème de la balance, cette difficulté d'apprentissage augmente avec le nombre de poids et des distances utilisées pour placer les poids (l'apprentissage par coeur devient de plus en plus nécessaire). Nous avons proposé pour ce problème une solution qui utilise un autre type de réseaux, les *réseaux Sigma-Pi* [RUM 86b, REY 97].

Type du Réseau	nombre d'exemples de apprentissage	nombre d'exemples de test	nombre maximum d'époques	Moyenne sur 5 simulations		
				Test de généralisation	Meilleure époque	CasCor Unités ajoutées
Shultz PMC	400	24	≤ 300	90.84%	35	0
Parfait PMC	400	24	sans limite	**100.00%**	**1544**	**14**
Shultz Sigma-Pi	400	24	≤ 300	**100.00%**	**157**	**0**
Shultz* PMC	400	625	≤ 300	92.16%	206	2
Parfait* PMC	400	625	sans limite	**100.00%**	**1500**	**13**
Parfait* Sigma-Pi	400	625	sans limite	**100.00%**	**396**	**1**
Complet PMC	625	625	sans limite	**100.00%**	**1871**	**18**
Complet Sigma-Pi	625	625	sans limite	**100.00%**	**91**	**0**

* Utilise tous les exemples disponibles dans la base de test de généralisation

Tableau 12 - Résultats des simulations du problème de la balance obtenus avec différents ensembles de données et différents types de réseaux

Nous avons utilisé le système INSS pour simuler ce type de réseaux, et les résultats obtenus sont encourageants. Le Tableau 12 montre les résultats de plusieurs expériences que nous avons faites avec la base de données du problème de la balance. On observe que les résultats des expériences avec la simulation des réseaux Sigma-Pi sont nettement supérieurs à ceux obtenus avec des réseaux de type PMC.

5.3.3 Discussion Finale sur le Problème de la Balance

Nous avons utilisé le système INSS pour étudier le problème de la balance présenté par Th. Shultz. Notre système a permis d'identifier des problèmes liés à l'apprentissage des règles qui composent le 4ème niveau du problème. Nous avons fini par nous concentrer spécifiquement sur cette étude de l'apprentissage des connaissances qui décrivent le 4ème niveau du problème de la balance. Notre intérêt pour cet aspect du problème est dû au fait que nous voulions mieux comprendre comment sont acquises et codées les connaissances de haut niveau (règles de haut niveau) dans les réseaux. En utilisant nos outils, nous avons pu constater que probablement Shultz [SHU 94] n'est jamais vraiment arrivé au 4ème niveau parfait. Avec l'aide de INSS et de l'interaction avec un expert humain, nous avons trouvé un ensemble de règles d'un niveau intermédiaire entre les niveaux 3 et 4 - le niveau 3.5. Les tests réalisés avec INSS montrent que cet ensemble de règles, plus simple que les règles du niveau 4, peut en réalité représenter le niveau final atteint par les autres chercheurs dans ses expériences sur l'apprentissage du problème de la balance.

Ensuite, nous avons étudié des solutions pour résoudre le problème de l'apprentissage des règles multiplicatives du niveau 4. Nous avons proposé une solution basée sur l'utilisation des réseaux de type Sigma-Pi. Nous avons pu tester l'apprentissage de ce type de réseau à travers des simulations réalisées en utilisant le système INSS. Les résultats obtenus montrent que ce type de réseau permet d'apprendre les règles du niveau 4 de façon plus rapide et avec une meilleure généralisation (utilise moins d'unités) que les réseaux de type PMC.

En conclusion, l'utilisation d'INSS pour traiter le problème de la balance a apporté deux contributions majeures : l'identification des problèmes liés à l'apprentissage des règles du 4ème niveau et l'aide à l'expert afin d'identifier les règles du niveau 3.5. De plus, nous avons proposé une alternative qui permet aux réseaux d'apprendre parfaitement les règles multiplicatives de haut niveau présentes dans le problème de la balance. Finalement, en ce qui concerne le problème de l'extraction de règles de haut niveau à partir des réseaux, on trouve dans les travaux réalisés par Reyes [REY 97] une proposition d'un système d'extraction qui pourra être capable d'extraire ce type de règles (basée sur des algorithmes génétiques comme ceux cités dans la Section 2.2.2.2).

5.4 ROBOTIQUE AUTONOME

Pour cette dernière application, nous avons choisi d'utiliser notre système dans une tâche de contrôle d'un robot mobile autonome. Des études avaient déjà été réalisées par des chercheurs de notre équipe dans le cadre d'un projet financé par la DRET [FES 97, FES 96]. Ces travaux portaient sur le contrôle d'un robot autonome, le robot Robuter, avec utilisation des réseaux de neurones.

La réalisation d'un système de contrôle pour la robotique autonome utilisant des réseaux de neurones est un sujet de recherche intéressant, car il permet d'aborder des questions telles que :

- Les problèmes liés au contrôle sensori-moteur;
- La réalisation de tâches de bas niveau (e.g. évitement d'obstacles) ainsi que de tâches de haut niveau (e.g. planification);
- Les interactions possibles entre un module connexionniste et un module symbolique;
- Les informations du niveau sensoriel sont représentées usuellement par des valeurs continues (signaux obtenus à partir des capteurs), ce qui pose des problèmes pour réaliser des traitements purement symboliques;

- La possibilité d'implémentation de plusieurs types de comportements différents, tels que : l'évitement d'obstacles, le suivi d'un mur, le suivi d'un couloir, le passage d'une porte, le déplacement vers une direction spécifiée;

- Les possibilités d'application d'approches modulaires (combinaison de multiples comportements) ou constructives (ajout incrémental de nouvelles compétences).

De plus, la robotique autonome est en plein développement actuellement, car les applications de ce type de système peuvent être multiples, à savoir : l'exploration d'environnements hostiles comme la planète Mars et les volcans, le pilotage automatique d'engins industriels, le pilotage automatique de voitures. Il faut bien reconnaître que les avances dans ce domaine ne nous permettent pas encore d'avoir des systèmes vraiment autonomes très performants (e.g. le robot *Sojourner* envoyé sur la planète Mars était en réalité télécommandé la plupart du temps). Ainsi, la robotique autonome reste-elle un sujet de recherche très intéressant et ouvert à de nouvelles études.

Nous avons choisi d'utiliser le robot Khepera [MON 93], le laboratoire Leibniz ayant à disposition un simulateur de ce robot, ainsi que le robot lui-même. Notre but était de tester INSS dans le cadre d'une nouvelle application réelle basée sur l'utilisation de valeurs d'entrée continues. Cette application est aussi parfaitement adaptée à l'étude et à la validation de notre système, car :

• Nous pouvons obtenir des exemples d'apprentissage et de test plus facilement, sans avoir des limitations comme c'était le cas pour la base de données des comas toxiques limitée, elle, à un certain nombre de cas connus;

• Nous pouvons générer de nouveaux exemples pour les situations qui posent des problèmes à l'apprentissage d'une tâche;

• Nous pouvons "jouer le rôle de l'expert" sur le sujet traité, car les connaissances sur des tâches comme l'évitement d'obstacles ne sont pas très complexes. Ainsi, nous pouvons facilement définir des bases de règles sur le problème;

• Nous avons besoin d'un système capable de traiter les valeurs sensorielles des capteurs (valeurs continues). Nous avons besoin d'un système capable de représenter les connaissances théoriques sur le problème à travers des règles d'ordre 0^+;

Nous allons présenter ici seulement les résultats dont nous pensons qu'ils sont les plus significatifs et qu'ils complétent le mieux les résultats déjà obtenus dans les autres applications.

5.4.1 Description de l'Environnement de Tests Utilisé

Nous avons utilisé le robot Khepera [MON 93] dans nos expériences en robotique autonome. Initialement, les expériences ont été développées avec un simulateur du robot, le *Khepera Simulator 2.0* développé par Olivier Michel à l'université de Nice Sophia-Antipolis [MIC 96]. Ce logiciel '*freeware*' peut être téléchargé à partir du WWW (Internet) à l'adresse : *http://wwwi3s.unice.fr/~om/khep-sim.html*.

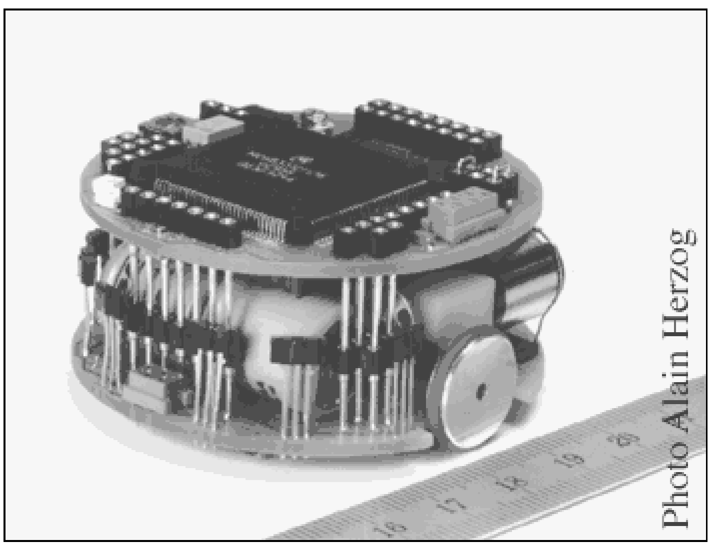


Figure 65 - Le robot Khepera

5.4.1.1 Le Robot Khepera

Khepera est un robot miniature de forme circulaire destiné à la recherche (voir Figure 65). D'un diamètre de 55mm, d'une hauteur de 30mm, Khepera pèse 86g. Il a été développé afin de tester les algorithmes de contrôle en robotique dans le monde réel. Il est constitué de deux roues et 8 capteurs infrarouges (IR). Ces derniers permettent la détection des obstacles (émission/réception IR) et des sources lumineuses (réception IR). Six capteurs sont placés à l'avant du robot et les deux autres à l'arrière (voir Figure 66). Pour la mesure de distance, les capteurs renvoient des valeurs numériques comprises entre 0 (lorsque le robot est à une distance supérieure à 5cm de l'obstacle) et 1023 (lorsque le robot est à une distance inférieure à 2cm de l'obstacle). Pour la mesure d'intensité lumineuse, l'intervalle est de 50 (intensité forte) à 520 (intensité nulle).

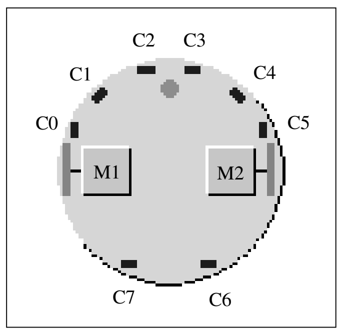


Figure 66 - Emplacement des capteurs sur Khepera

La commande des moteurs entraînant les roues est donnée par des valeurs comprises dans un intervalle entre -10 (tout vitesse vers l'arrière) et +10 (tout vitesse vers l'avant). Les deux moteurs sont indépendants, ce qui permet de faire des rotations par l'intermédiaire d'une commande utilisant deux valeurs opposées (e.g. +5,-5 = réaliser une rotation en vitesse moyenne dans le sens horaire). Khepera peut être utilisé sur un bureau, connecté à une station de travail par un câble de liaison série du type RS232. Cette liaison permet de lire les valeurs des capteurs et d'envoyer des commandes au robot en temps réel. On peut aussi télécharger un programme de contrôle dans sa mémoire (Khepera est doté d'un processeur Motorola 68331) par la liaison série, puis de le déconnecter du câble série afin de pouvoir rester complètement autonome.

5.4.1.2 Le Simulateur de Khepera

Le *Khepera Simulator 2.0* a été développé par Olivier Michel [MIC 96] à l'université de Nice Sophia-Antipolis sous l'environnement UNIX avec le langage C. Le simulateur est doté d'une interface graphique conviviale qui permet de créer des environnements personnalisés et d'y déplacer le robot très facilement. L'interface graphique est composée de deux parties : une partie correspondant à l'environnement simulé et une partie permettant à l'utilisateur de commander et d'observer le comportement du robot (contrôle des capteurs IR et des moteurs). La Figure 67 présente l'interface graphique du simulateur de Khepera.

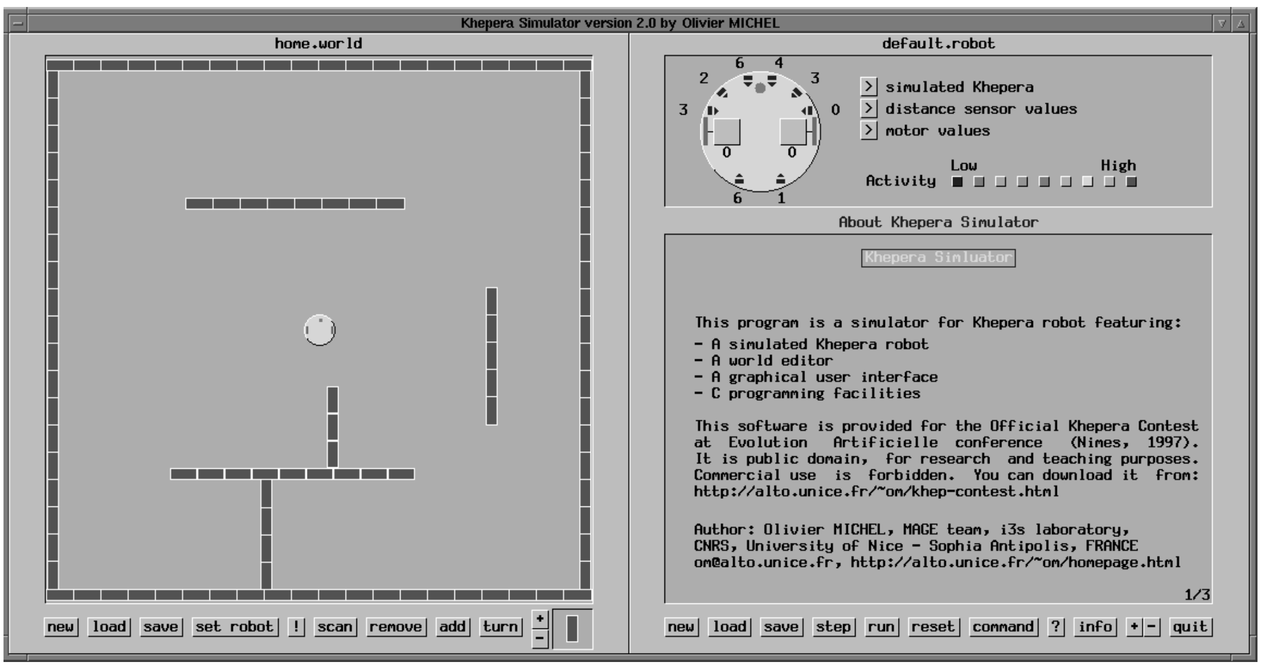


Figure 67 - Interface graphique du simulateur de Khepera

L'environnement simulé est équivalent à un espace plan de 1m sur 1m. Il est composé de trois types d'objets différents : des briques à 24 orientation possibles, des barreaux et des sources lumineuses. Ces éléments peuvent être placés à un endroit quelconque de l'environnement. Les briques permettent de créer des murs, et les barreaux représentent des obstacles ponctuels qui n'activent qu'un seul capteur à la fois. Quant aux sources de lumière, ces sont des obstacles équivalents aux barreaux mais qui activent les capteurs de luminosité. De nombreux "mondes" (configurations de l'environnement) sont déjà définis dans le simulateur.

La commande du robot, l'affichage graphique de textes et de dessins sont entièrement paramètrables et programmables en langage C. C'est grâce à ces fonctions programmables que l'on peut développer son propre contrôleur du robot mobile Khepera. Toutes les informations sur l'état du robot sont aussi facilement accessibles par ces fonctions (e.g. valeurs des capteurs de distance, valeurs de la commande passée aux moteurs). Le manuel du simulateur possède une description détaillé des fonctions disponibles et de son mode d'emploi [MIC 96].

Le modèle choisi pour simuler les moteurs du robot est très simple. Le robot se déplace suivant la commande envoyée par l'utilisateur. Un bruit aléatoire de plus ou moins 10% est ajouté à la vitesse du moteur et un bruit de plus ou moins 5% est ajouté au calcul de la position résultante du robot obtenue en fonction du déplacement et de la direction indiquée par la commande précédente. Pour simuler un capteur, on considère un ensemble de 15 points disposés dans un triangle situé devant le capteur. La distance à l'obstacle est obtenue en fonction de la présence ou non d'obstacles en chacun des points considérés. A cela est rajouté un bruit de plus ou moins 10% de l'amplitude de la distance précédemment calculée.

Le modèle de simulation utilisé donne des résultats suffisamment proches de la réalité pour qu'on puisse ensuite passer au robot réel. Nous n'avons pas besoin de réaliser des changements significatifs sur les algorithmes de contrôle développés sur le simulateur afin de les utiliser sur le robot réel. De plus, nous pouvons utiliser la même interface du simulateur lorsque nous avons à choisir entre deux modes d'opération : mode de simulation ou mode de contrôle en temps réel du robot (les commandes et les valeurs des capteurs sont obtenues à partir de la liaison série).

Nous avons introduit quelques modifications sur le simulateur de Khepera afin de pouvoir mieux exploiter ses fonctionnalités :

1. Nous avons introduit la possibilité de commander le robot simulé par le clavier. L'utilisateur peut remplacer l'algorithme de contrôle et générer lui-même quelques commandes à envoyer au robot (e.g. avancer, tourner à gauche, tourner à droite).

2. Nous avons créé des fonctions qui permettent d'enregistrer sur des fichiers toutes les commandes envoyées au robot, ainsi que sont état à chaque pas (capteurs, position et orientation). De cette façon, nous pouvons créer des fichiers d'apprentissage supervisé (voir Figure 68) en réalisant le contrôle du robot par le clavier. L'utilisateur peut accompagner l'état des capteurs du robot, qui sont affichés à l'écran, et ensuite indiquer la commande la plus appropriée à réaliser.

3. Nous avons introduit une version simplifiée du simulateur NeuSim dans le simulateur de Khepera. Ainsi, nous avons la possibilité d'utiliser les réseaux créés par INSS pour contrôler le petit robot. Cette version simplifiée de NeuSim permet de charger un réseau et de l'activer, mais elle ne permet pas de réaliser l'apprentissage. Il faut utiliser la version complète de NeuSim pour pouvoir obtenir un réseau prêt à être employé avec le simulateur du robot.

4. Nous pouvons aussi enregistrer le comportement du robot quand il est contrôlé par le simulateur des réseaux de neurones. Cela nous permet se suivre précisément les réponses données par le système et, si l'on veut, de les valider postérieurement. De plus, nous pouvons utiliser ce fichier afin d'afficher la trajectoire complète parcourue par le robot pendant une expérience.

Fichier : robot.learn (base d'apprentissage - 8 entrées, 3 sorties)

```
8    3    1028  0
4    1    3     4    6    3    0    4      0 1 0
1    3    6     3    6    6    6    6      0 1 0
6    3    104   234  772  96   3    6      1 0 0
2    6    104   229  724  107  3    4      1 0 0
563  1023 57    6    6    0    3    1      0 0 1
544  1023 1     3    4    0    5    1      0 0 1
...
C0   C1   C2    C3   C4   C5   C6   C7     G A D
```

C0 - C7 : Capteurs 0 à 5 (devant) + 6 et 7 (arrière)
G: Tourner à gauche
A: Avancer
L: Tourner à droite

Figure 68 - Exemple d'un fichier d'apprentissage utilisé par NeuSim

Les expériences que nous avons développées avec le simulateur de Khepera, ainsi que les modifications apportées au simulateur, ont suscité d'autres travaux postérieurs développés au sein de notre équipe [PLE 97, MEK 97, PAR 96].

5.4.2 Les Résultats Expérimentaux Obtenus avec INSS

Nous avons commencé nos expériences par l'apprentissage d'un comportement d'évitement d'obstacles. Cette expérience a consisté à créer un fichier d'apprentissage contenant des situations caractéristiques d'évitement d'obstacle, indiquées dans la Figure 69. Nous avons créé un fichier contenant une base d'apprentissage supervisé (robot télécommandé) et ensuite nous avons fait apprendre au réseau cette base. Le comportement que nous avons voulu apprendre consistait à s'éloigner des obstacles (murs) à chaque fois que le robot les rencontrait, c'est-à-dire à tourner vers un côté jusqu'à ne plus avoir un obstacle en face.

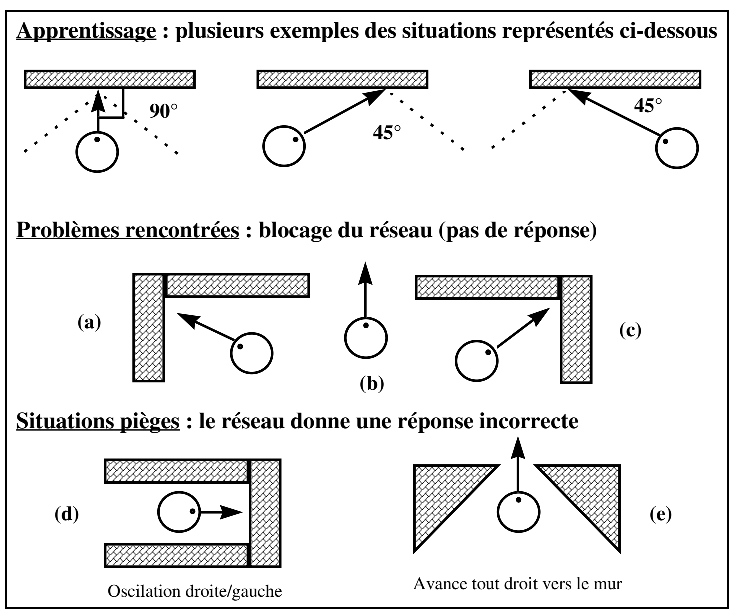
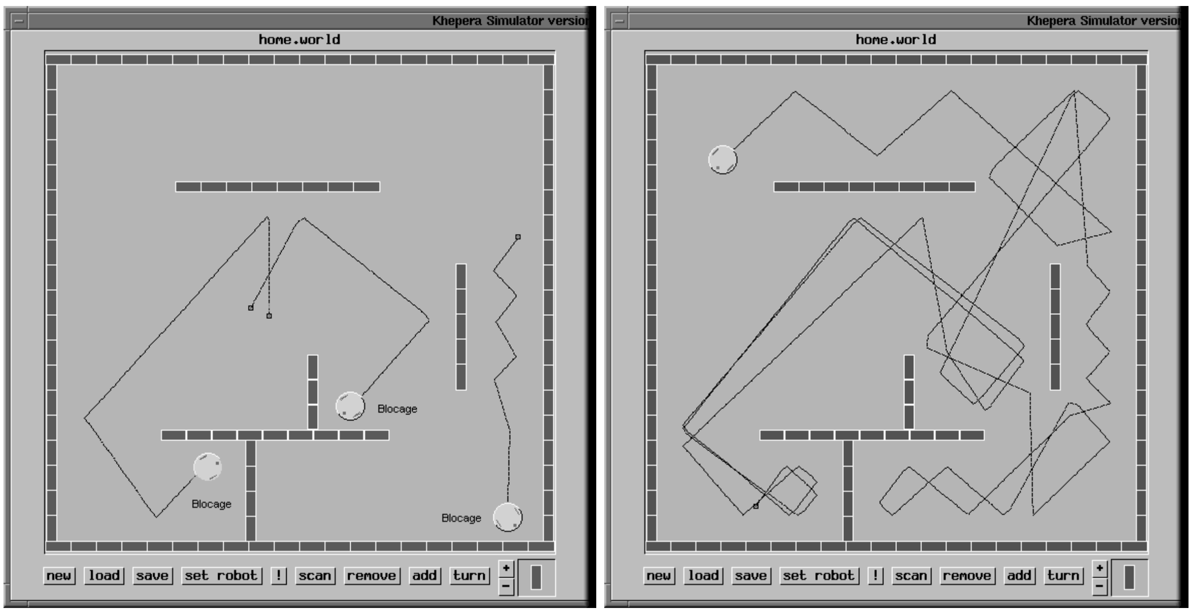


Figure 69 - Situations typiques d'évitement d'obstacle

Dans nos expériences, nous avons obtenu un taux proche de 95% de bonnes réponses sur la base d'apprentissage utilisée. Nous avons constaté que le réseau a bien appris à éviter les murs qui se présentaient en face du robot (comportement basé sur les 3 types de cas d'apprentissage - Figure 69). Nous avons obtenu une bonne généralisation pour des situations différentes de celles apprises, où le contact avec le mur s'est produit avec un angle différent de ceux utilisés dans la base d'apprentissage.



(a) Apprentissage de situations simples *(b) Après l'ajout des nouveaux exemples*

Figure 70 - Evitement d'obstacles : trajectoire du robot après l'apprentissage

Cependant, le robot qui a fait preuve d'un comportement parfait en présence d'un mur unique et sans coins, n'avait pas un bon comportement en face de situations comme celles que nous décrivons dans la Figure 69 (problèmes et pièges). Dans des situation de type 'a', 'b' ou 'c' le réseau donne une réponse avec toutes les sorties proches de zéro, c'est-à-dire il ne prend aucune décision et s'arrête. Cela se produit car le réseau est face à une situation (configuration des capteurs) complètement inconnue. Nous avons ajouté des exemples des situations 'a' et 'c' dans le fichier d'apprentissage et le réseau ainsi obtenu n'a plus présenté de problèmes par rapport à ce type de cas (voir Figure 70). Par contre, nous avons aussi ajouté des exemples de cas de la situation de type 'b' (passage d'une porte) qui n'ont pas beaucoup contribué à améliorer le comportement du robot contrôlé par notre réseau. Avec ces nouveaux exemples appris, un nouveau problème est apparu : le robot, qui est maintenant devenu capable de passer par une configuration de type 'b', va tout droit dans une situation de type 'e', jusqu'au point de heurter le mur. Nos essais ont montré que les capteurs du robot n'ont pas une précision suffisante pour distinguer entre une situation 'b' ou 'e' (capteurs latéraux saturés et capteurs frontaux avec une faible activation).

Un dernier problème constaté lors de nos essais sur l'évitement d'obstacles, est le cas du blocage du robot dans une situation de type 'd' (couloir sans issue). Le robot rencontre le fond du couloir. Il commence à tourner vers l'un des deux côtés. Il rencontre un mur et commence à tourner vers l'autre côté où il rencontre de nouveau un mur. Il se met alors à osciller entre un côté et l'autre indéfiniment. Ce type de blocage peut aussi arriver dans certains cas bien précis de situations de type 'a' ou 'c'. Nous avons essayé d'ajouter des exemples sur les situations de blocage, mais cela n'a pas marché. L'utilisation de notre système a permis d'identifier la cause exacte du problème : le robot se met à osciller entre tourner à gauche et tourner à droite, parce que chaque commande provoque un changement de l'état des capteurs qui amène à la commande inverse dans le pas de temps suivant.

Nous avons conclu qu'il faut prendre en compte le temps (contexte de la situation) afin de pouvoir sortir de cette situation de blocage. Une fois que l'on a commencé à tourner à droite, il faut poursuivre cette opération jusqu'à ce que le chemin soit libre. Afin de résoudre ce problème, nous avons envisagé d'utiliser une fenêtre temporelle : utiliser l'état des capteurs dans les pas de temps antérieurs afin de fournir un contexte au réseau, et ainsi pouvoir prendre une décision plus juste.

Nous avons employé ce type d'approche dans une autre expérience qui nécessite aussi une information contextuelle (le suivi de contour de murs). Toutefois, pour éviter de tout recommencer à partir de zéro, nous avons adopté une solution plus simple : nous faisons un post-traitement des réponses du réseau. Si l'on trouve des séquences d'alternance entre les commandes tourner à gauche et tourner à droite, nous faisons ensuite un grand tour vers l'un des côtés.

Ce premier groupe d'expériences nous a permis de constater encore une fois l'importance d'avoir la capacité de faire évoluer les connaissances acquises sur le problème en question. L'utilisation de notre système a beaucoup aidé dans le traitement de cette tâche. Nous avons pu raffiner les connaissances du réseau en ajoutant de nouveaux exemples. Le système nous a aidés à trouver les 'points faibles' du comportement appris par le réseau et les problèmes de codage des informations d'entrée (nécessité d'informations temporelles). Le choix des exemples à ajouter dans la base d'apprentissage a été possible grâce à l'identification précise des problèmes rencontrés dans le comportement du robot. Nous n'avons pas fait de tests sur toutes les possibilités de problèmes que nous pourrons rencontrer dans une application comme celle-ci, et nous sommes certains qu'il existe encore d'autres problèmes différents de ceux que nous avons présentés ici. L'important pour nous n'est pas d'arriver à un système parfait (on peut se poser la question de savoir si un tel système existe bien) mais d'être capables de faire évoluer constamment les connaissances sur le problème étudié.

Notre deuxième série d'expériences a été composée par des tests sur la création et l'insertion de règles d'ordre 0^+ dans les réseaux. Avec l'expérience acquise dans notre première expérience, nous avons créé une base de règles symboliques qui puisse coder les connaissances sur le problème d'évitement d'obstacles. Nous avons obtenu l'ensemble de règles suivant :

```
#define DETECTWALL 900
#define SENSIB     100

% VERY SIMPLE Autonomous Robot Controller - by Osorio, Lab. Leibniz

$Features:
SSL : range : [0.0 , 1024.0];  % Sensor 0            2   3
SL  : range : [0.0 , 1024.0];  % Sensor 1       1     SFL SFR    4
SFL : range : [0.0 , 1024.0];  % Sensor 2         SL          SR
SFR : range : [0.0 , 1024.0];  % Sensor 3     SSL               SSR
SR  : range : [0.0 , 1024.0];  % Sensor 4      0                 5
SSR : range : [0.0 , 1024.0];  % Sensor 5
SBR : range : [0.0 , 1024.0];  % Sensor 6            SBL SBR
SBL : range : [0.0 , 1024.0].  % Sensor 7             7   6
$End_Feat.

$Rules:

right <= GT (SSL, DETECTWALL, SENSIB);
right <= GT (SL,  DETECTWALL, SENSIB);
right <= GT (SFL, DETECTWALL, SENSIB);

left  <= Not(right), GT (SFR, DETECTWALL, SENSIB);
left  <= Not(right), GT (SR,  DETECTWALL, SENSIB);
left  <= Not(right), GT (SSR, DETECTWALL, SENSIB);

forward <=  Not(right), Not(left).

$End_rules.

$End.
```

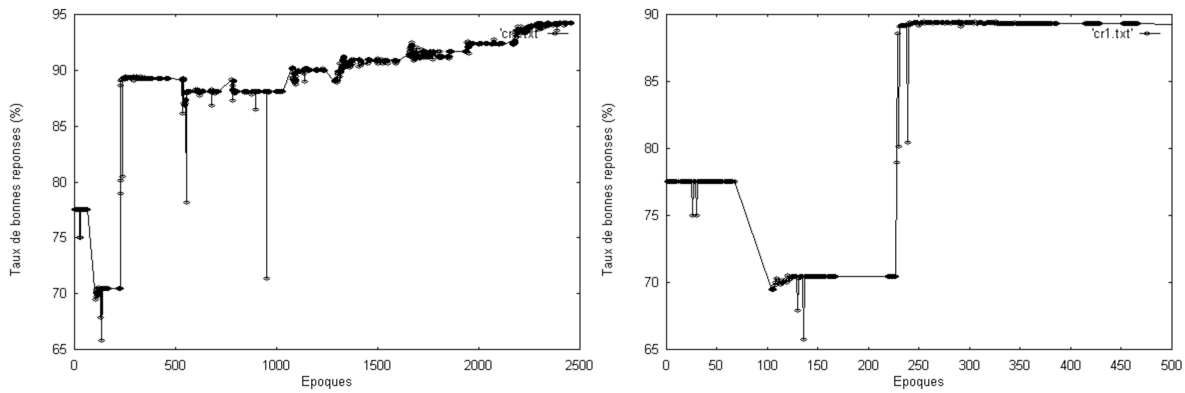
Figure 71 - Ensemble de règles symboliques pour l'évitement d'obstacles

Cette base de règles a été compilée sous la forme d'un réseau. Ensuite, nous avons testé ce réseau dans le simulateur du Khepera. Le comportement du robot obtenu avec cette base de règles est un peu différent de celui obtenu par l'apprentissage de la base d'exemples utilisée dans notre première série d'expériences. Le robot arrive à éviter les murs parfaitement, mais au lieu de s'éloigner toujours des murs, il tourne parfois de façon à pouvoir tout juste éviter les murs en continuant d'avancer. La Figure 73(a) montre des exemples du comportement du robot obtenu en utilisant la base de règles qui se trouve ci-dessus. Cette figure montre que, comme nous l'avons dit, le robot évite de trop s'approcher des murs, mais sans forcement s'en éloigner.

Il faut cependant remarquer que la façon dont nous avons avons conçu la base de règles symboliques a fini par introduire un biais dans le comportement du robot. Le robot va avoir en effet une tendance à tourner plutôt vers la droite. Ce type de comportement finit parfois par restreindre la capacité d'exploration de l'environnement du robot, puisqu'il il va toujours dans la même direction.

Enfin nous avons essayé d'améliorer les connaissances de départ du robot codées par les règles, à l'aide de l'apprentissage de la base d'exemples que nous avons utilisé précédemment avec succès (compilation de règles suivi de l'apprentissage). Notre première réaction a été de considérer cette expérience comme un échec complet. Tout de suite après le démarrage du processus d'apprentissage, nous nous sommes aperçus qu'au lieu d'améliorer la qualité des réponses du réseau, le processus d'apprentissage provoquait une décroissance de la quantité de bonnes réponses obtenues (voir la Figure 72 - les 200 premières époques).

Nous avons arrêté la simulation, et nous sommes d'abord demandé si le problème ne résultait pas d'une mauvaise implémentation de nos algorithmes d'apprentissage. Nous avons refait la simulation de façon à mieux comprendre ce qui s'était passé. Dans cette deuxième simulation, nous avons laissé le processus d'apprentissage poursuivre son exécution pendant un plus grand nombre d'époques. Nous avons été surpris par les résultats obtenus, et surtout par la façon dont s'est développé l'apprentissage. Le réseau a finalement réussi à apprendre la base avec un taux proche de 95% de bonnes réponses. Nous avons répété cette expérience plusieurs fois et le résultat a toujours été le même : une chute dans le taux de réussite dans les premières époques (autour de 300 époques), suivi d'une augmentation de ce taux jusqu'aux alentours de 95% de réussites. La Figure 72 montre un exemple typique de l'évolution du processus d'apprentissage dans une des expériences que nous avons réalisées.



(a) Apprentissage complet *(b) Vue en détail des premières 500 époques*

Figure 72 - Evolution du taux de réussite pendant l'apprentissage (règles + exemples)

Nous avons fait l'hypothèse que ce comportement tout à fait singulier du réseau est dû à l'introduction de règles symboliques avant la réalisation du processus d'apprentissage. Cette hypothèse a été renforcée par les faits suivants :

1. Les règles symboliques ont donné au robot un comportement (voir Figure 73(a)) différent de celui obtenu par l'apprentissage uniquement des exemples (voir Figure 70). Les connaissances théoriques divergent des connaissances pratiques disponibles sur cette application.

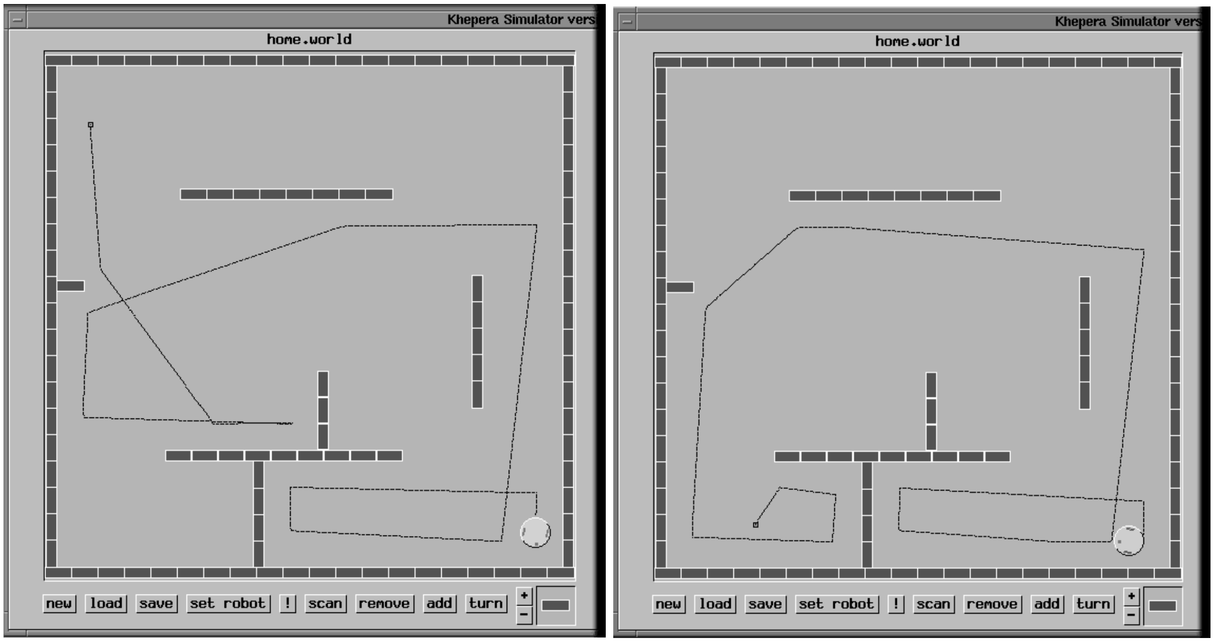
2. Le réseau avait besoin d'effacer (ou modifier fortement) ses connaissances de départ pour pouvoir arriver à bien apprendre la base d'exemples. Cela peut expliquer la décroissance initiale du taux de bonnes réponses, suivi d'une amélioration.

3. Le taux final de bonnes réponses (95%) est pratiquement le même que celui obtenu par l'apprentissage sans l'introduction des règles. Les règles n'ont pas aidé à améliorer la performance du réseau. De plus, il nous semble que les règles ont fini par être totalement remplacées par les nouvelles connaissances acquises.

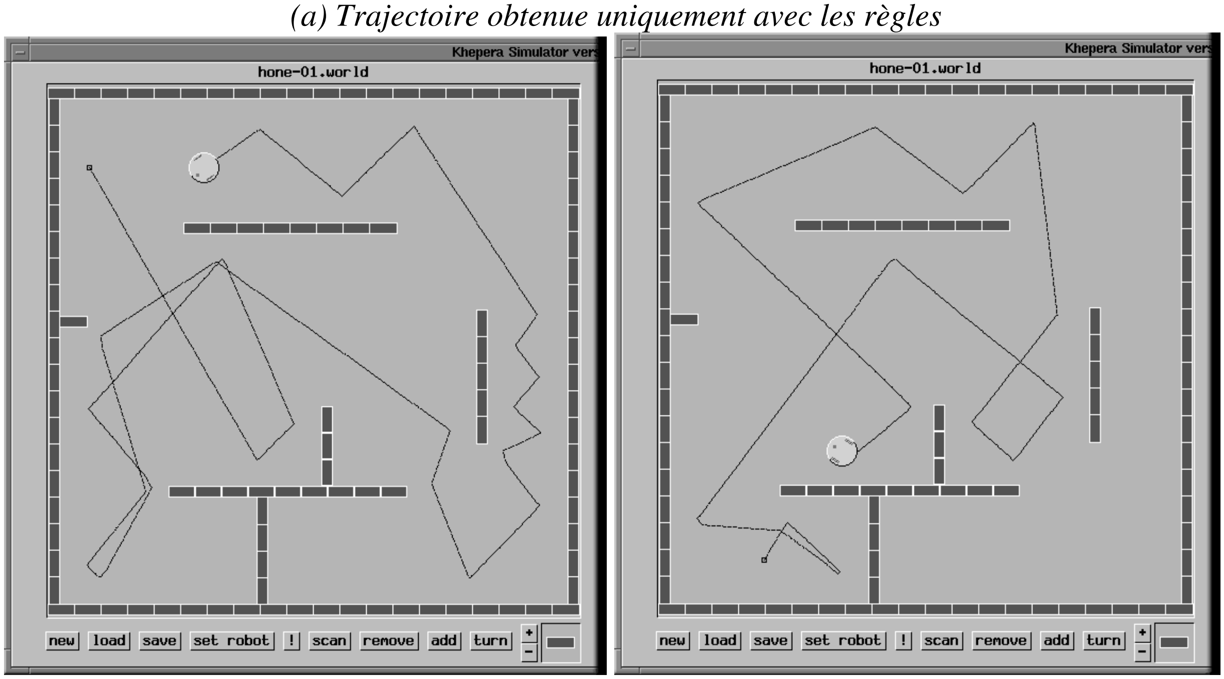
4. Le type de comportement initial du robot avant l'apprentissage (il utilise alors seulement les règles - voir Figure 73(a)) est différent de son comportement après l'apprentissage (voir Figure 73(b)). Le comportement initial est d'éviter les murs, sans avoir besoin de s'en éloigner. Le comportement final obtenu après l'apprentissage (règles + exemples) est très proche, sinon identique, de celui obtenu dans notre première expérience basée uniquement sur des exemples (voir la Figure 70 et Figure 73(b)).

Nous pensons donc que le comportement anormal du réseau est dû au fait que nous avons introduit des règles que ne sont pas en accord avec le type de comportement reproduit dans la base d'apprentissage. (Ce comportement du réseau peut être mis en relation avec le comportement du réseau étudié par D.E. Rumelhart et J.L. McLelland pour l'apprentissage du prétérit des verbes anglais ; voir le chapitre 18 du "PDP" [RUM 86a]). Nous avons réussi à apprendre les exemples, grâce à la capacité de nos réseaux d'ajouter de nouvelles connaissances (et de nouvelles unités), car un réseau statique ne serait pas capable de changer à ce point ses connaissances de départ. Cette expérience a prouvé que nos réseaux sont capables de manipuler des connaissances d'origines différentes (théoriques et pratiques), et ils fonctionnent bien même si les connaissances fournies sur le problème sont très divergentes.

Dernier point : le fait d'avoir réussi à imposer le comportement décrit par les exemples ne veut pas dire que l'utilisateur va être obligé de le considérer comme la meilleure solution. L'un des points forts du système INSS est cette possibilité de mettre face à face les différentes connaissances disponibles sur le problème, et ainsi de pouvoir mieux les intégrer, les exploiter et les valider.



(a) Trajectoire obtenue uniquement avec les règles



(b) Trajectoire obtenue après l'apprentissage (règles + exemples)

Figure 73 - Evitement d'obstacles : utilisation d'une base de règles

Le dernier groupe d'expériences que nous avons réalisées est composé par des tests sur l'apprentissage d'un nouveau type de comportement : le suivi de contour d'un mur. Ce type de tâche est plus complexe qu'un simple évitement d'obstacles, car le contexte de la situation dans laquelle nous nous trouvons joue un rôle essentiel. De plus, nous avons choisi de tester cette application sur le robot réel afin de valider les expériences réalisées au niveau du simulateur.

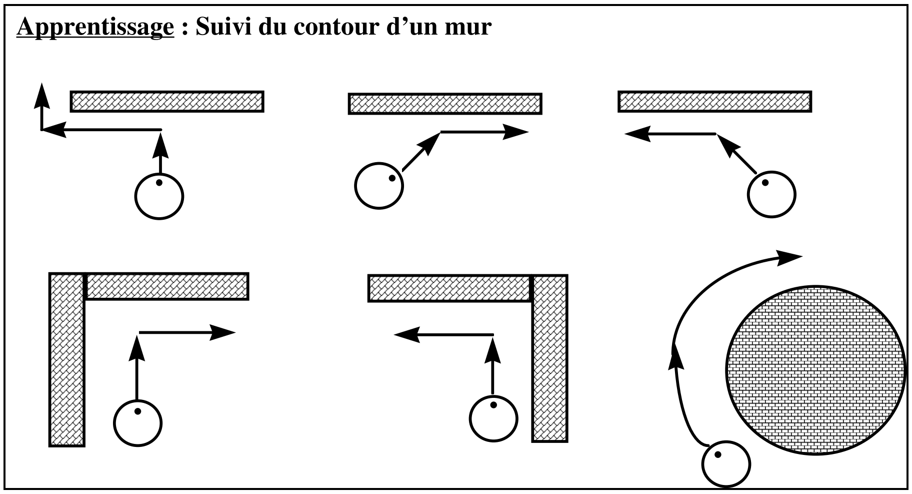


Figure 74 - Apprentissage d'un comportement de suivi du contour d'un mur

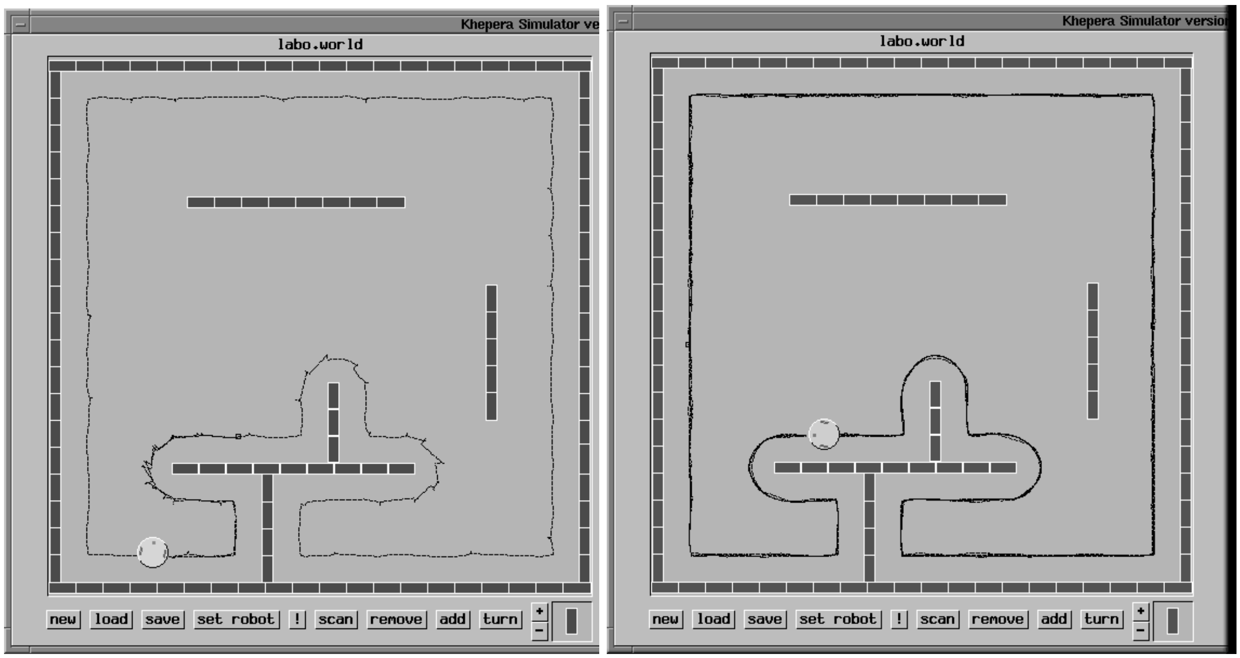
Nous avons commencé cette étude par l'apprentissage d'une base d'exemples. Cette base était composée de situations simples, comme celles représentées dans la Figure 74, sauf que nous avons obligé le robot à toujours tourner dans un seul sens, de façon à garder le mur toujours sur sa gauche (ceci évite les conflits et les exemples contradictoires). Chaque exemple de la base comporte les valeurs des capteurs dans le temps *'t'* (pas de temps présent) et dans le temps *'t-1'* (pas de temps précédent), plus la valeur de la sortie désirée (tourner à gauche, avancer ou tourner à droite). L'utilisation de l'état précédent des capteurs permet d'avoir une information contextuelle et ainsi d'éviter de décrocher du mur. Nos expériences pratiques ont montré que l'utilisation d'un seul pas de temps précédent était suffisante pour éviter le décrochage du mur.

Pour réaliser l'apprentissage du réseau, nous avons procédé de la façon suivante :

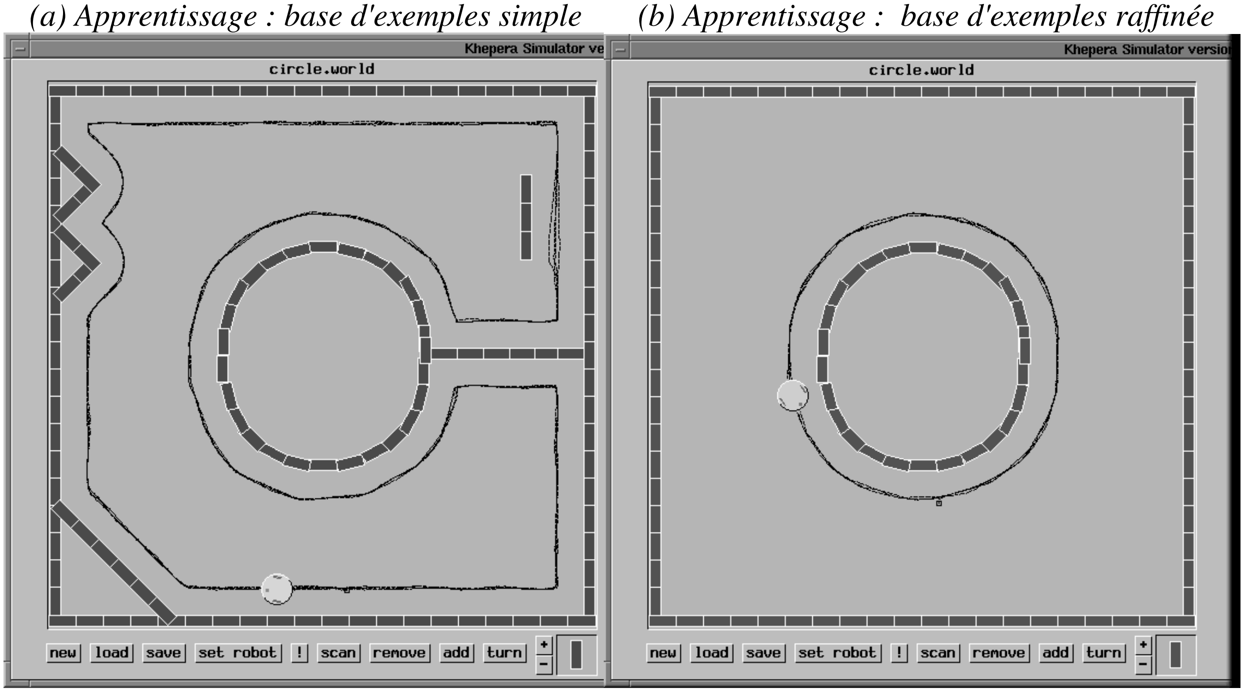
- Nous avons commencé l'apprentissage avec une base d'exemples simple, suivi d'un test du comportement du robot obtenu sur le simulateur;

- Ensuite, pour chaque situation de blocage (aucune réponse du réseau) ou de décrochage du mur, nous avons identifié les cas problèmes et nous avons ajouté des exemples complémentaires dans la base d'apprentissage afin de mieux apprendre à traiter ces cas;

- Nous avons répété ce processus jusqu'à ne plus avoir de problèmes de blocage ou de décrochage sur certains environnements de test préalablement choisis.



(a) Apprentissage : base d'exemples simple *(b) Apprentissage : base d'exemples raffinée*



(c) Trajectoire réalisée sans blocage, ni décrochage *(d) Trajectoire réalisée sans blocage, ni décrochage*

Figure 75 - Trajectoires du robot dans les environnements de test (suivi du contour)

La Figure 75 montre les trajectoires du robot dans les environnements de test du suivi de contour de murs. Dans la figure (a), on peut remarquer le comportement irrégulier du robot dans une phase initiale du processus d'apprentissage. Dans les figures (b), (c) et (d), on peut voir le comportement du robot après la fin du processus d'apprentissage (base d'exemples raffinée plusieurs fois). Nos tests ont montré qu'apparemment le robot simulé n'avait plus de problèmes de blocage ou de décrochage. L'étape suivante a été la réalisation d'un test de suivi de contour d'un mur avec le robot réel, en utilisant le même réseau testé sur le simulateur.

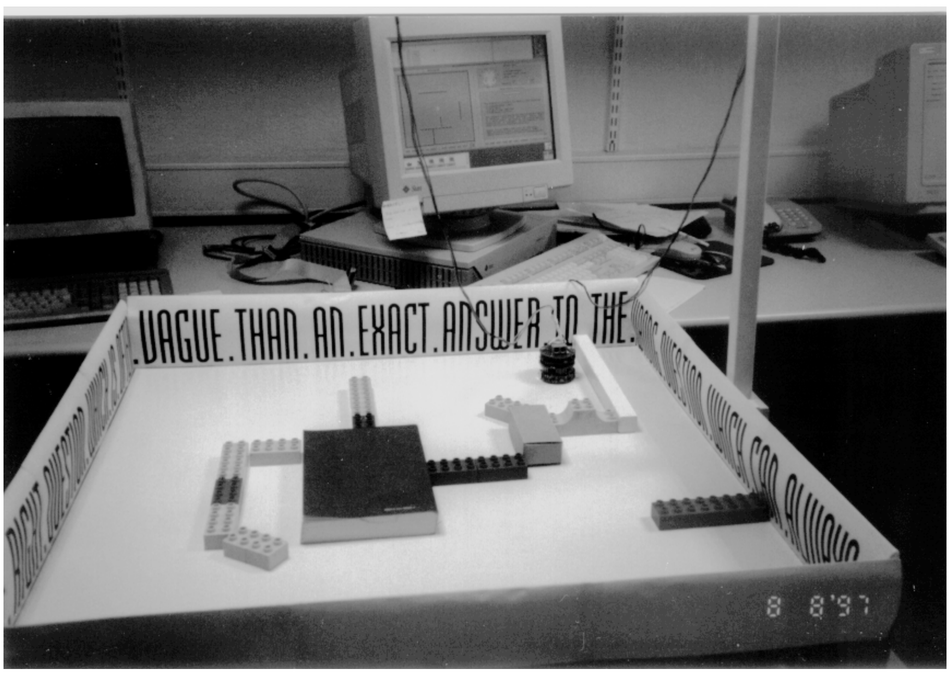


Figure 76 - Environnement de test avec le robot réel

Nous avons construit un environnement de test pour le robot, présenté dans la Figure 76. Comme le simulateur permet de commander le robot en utilisant la liaison série, nous n'avons rien eu à changer sur notre implémentation. La seule opération nécessaire à été de reconfigurer le simulateur de façon à passer du mode de contrôle du robot simulé au mode de contrôle du robot réel. La Figure 77 présente la trajectoire (elle est dessinée sur la photo) du robot réel obtenue en utilisant notre réseau pour la génération des commandes de contrôle. Nous avons répété cette expérience plusieurs fois. Dans aucune de ces expériences le robot n'a décroché, ni touché aux murs, ni s'est bloqué. Le comportement obtenu avec le robot réel a été parfait.

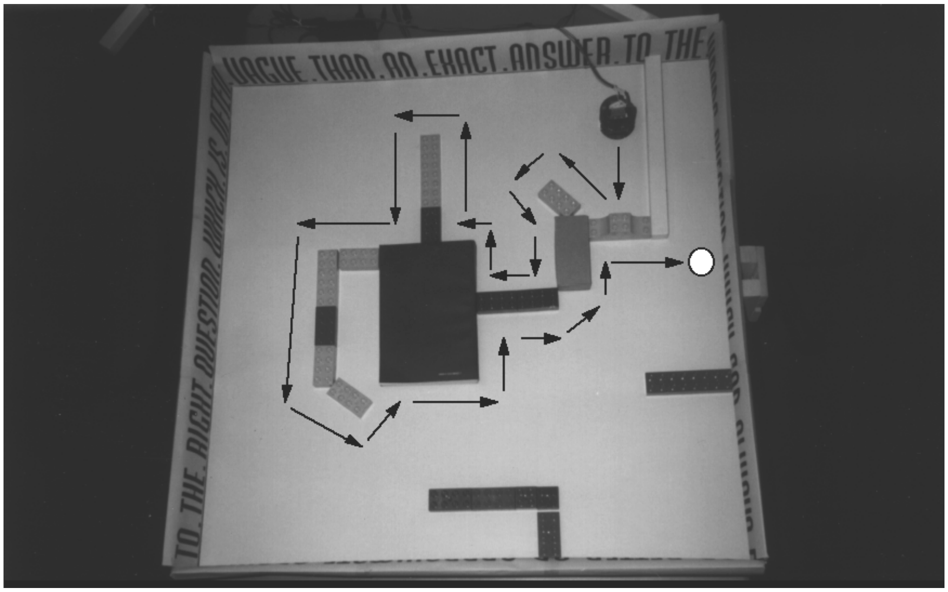


Figure 77 - Trajectoire du robot dans une tâche de suivi de contour des murs

Cette dernière expérience nous a permis de montrer que toutes les expériences réalisées au niveau du simulateur ne sont pas très différentes de ce qu'on peut faire avec le vrai robot. Le simulateur de Khepera implémente un modèle assez proche du comportement du vrai robot. Il faut reconnaître que cette facilité de passage du simulateur au robot réel est propre au Khepera et au type d'expérience que nous avons réalisées.

De plus, les réseaux que nous avons obtenus sont très robustes et généralisent bien dans des situations inconnues. Ainsi, notre système nous a-t-il permis de réussir l'implémentation du comportement de suivi de contour de murs d'une façon simple et rapide

5.4.3 Discussion Finale sur l'Application en Robotique Autonome

L'utilisation de notre système dans une application en robotique autonome nous a permis : de tester l'insertion de règles d'ordre 0^+, d'appliquer les techniques de raffinement de connaissances, et de montrer encore une fois que notre système peut être tout aussi performant dans des applications académiques que dans des applications réelles. Ce type d'application reste l'un des domaines de test des systèmes hybrides le plus intéressant que nous ayons eu l'occasion d'étudier. La possibilité de définir des bases de règles pour le contrôle du robot, ainsi que le fait de pouvoir générer des exemples (en quantité presque illimité), nous a permis d'approfondir nos études pratiques sur notre système.

Le système INSS a également été utilisé dans l'étude d'une autre application en robotique autonome à l'IFREMER. Il s'agit de mettre au point un système hybride de détection de pannes d'un robot sous-marin (Vortex). Des études préliminaires ont été développées par Christophe Perez dans le cadre d'un stage de D.E.S.S. en Informatique et Intelligence Artificielle [PER 96a, PER 96b] sous la supervision de M. Perrier. Ces travaux sont à l'origine des travaux menés actuellement par Benjamin Deuker dans le cadre de sa thèse, et aussi, d'un nouveau projet de coopération entre le laboratoire Leibniz et l'IFREMER (en cours d'élaboration).

Nous pensons que les études que nous avons développées jusqu'à présent sur les applications en robotique autonome doivent être poursuivies afin d'une part de mieux connaître toutes les potentialités et limites de notre système, d'autre part de confronter directement les résultats que nous avons obtenus à des résultats obtenus par d'autres chercheurs en robotique. Les systèmes de robotique autonome restent difficiles à évaluer avec précision. Nous n'avons pas les moyens d'obtenir une mesure absolue des performance du système, par opposition à certaines autres applications où cela est possible (e.g. diagnostic médical).

5.5 CONSIDERATIONS FINALES

Dans ce chapitre, nous avons présenté l'utilisation de notre système dans des applications académiques et dans des applications réelles. Les résultats obtenus montrent d'une façon pratique les divers points forts du système INSS :

• Le système a la capacité d'intégrer des connaissances théoriques (règles) et des connaissances empiriques (exemples) avec l'utilisation d'algorithmes très performantes. (Application utilisée : les problèmes du Moine);

• Le système a la capacité d'extraire seulement les nouvelles connaissances acquises par le réseau, sans avoir besoin d'extraire toutes les connaissances qui ont été introduites par compilation de règles. (Application utilisée : les problèmes du Moine);

• Le système est capable d'acquérir, d'extraire et de valider les connaissances disponibles sur un problème réel très complexe, avec une très bonne performance. (Application utilisée : aide au diagnostic des comas toxiques);

• Le système est capable de coder des règles de haut niveau (règles d'ordre 0+) dans les réseaux. Il peut traiter parfaitement les valeurs d'entrée discrètes ainsi que les valeurs continues. De plus, son module de validation permet d'identifier et de traiter des problèmes liés à l'étude des tâches sur lesquelles il fonctionne. (Applications utilisées : le problème de la balance et le contrôle en robotique autonome);

Nous pensons avoir réussi à montrer les différentes capacités du système INSS à travers ces expériences. Nous savons qu'il existe encore beaucoup d'études à réaliser sur notre système. Cependant, les expériences que nous avons réalisées montrent que ce nouveau système est très performant et qu'il peut être facilement appliqué à différents problèmes.