Apprentissage et adaptation pour la modélisation stochastique de systèmes dynamiques réels

Laurent Jeanpierre

FACULTE DES SCIENCES & TECHNIQUES

U.F.R. Sciences & Techniques Mathématiques, Informatique et Automatique
Ecole Doctorale IAE+M
Département de Formation Doctorale en Informatique

Thèse

présentée pour l'obtention du titre de

Docteur de l'Université Henri Poincaré, Nancy-I

en Informatique

par **Laurent JEANPIERRE**

Apprentissage et adaptation pour la modélisation stochastique de systèmes dynamiques réels

Soutenance publique : le 03 Décembre 2002

Membres du jury :

Président de jury :	M. Claude MEISTELMAN	Professeur Universitaire, Professeur des Hôpitaux
Rapporteur Interne :	M. Dominique COLNET	Professeur, Université Nancy 2
Rapporteurs :	M. Jacques DEMONGEOT	Professeur Universitaire, Professeur des Hôpitaux Membre de l' Institut Universitaire de France
	M. Paul RUBEL	Professeur INSA, Lyon
Directeur de thèse :	M. François CHARPILLET	Directeur de recherche INRIA
Examinateur :	M. Jean-Paul HATON	Professeur U.H.P., Nancy I Membre de l' Institut Universitaire de France
Membre Invité :	M. Pierre-Yves DURAND	Néphrologue à l'ALTIR

Laboratoire lorrain de recherche en informatique et ses applications
Campus Scientifique, 54506, Vandœuvre-lès-Nancy

Remerciements

Une thèse est un travail passionnant, mais long. Tout au long de ces trois années, nombreuses sont les personnes qui me sont venues en aide, que ce soit régulièrement, ou ponctuellement. Je dédie donc ce travail à tous ceux qui m'ont épaulé et qui ont cru en moi.

Plus particulièrement, je désire remercier mes parents, qui m'ont hébergé, nourri et qui ont réussi à me supporter depuis ma plus tendre enfance. Merci à leur soutien, à leur aide, et à leurs reproches quant à mes horaires irréguliers.

Au-delà de ma famille, je souhaite remercier tous ceux qui m'ont encadré dans le cadre de mes recherches :

- Odile Mella, que je ne remercierai jamais assez pour m'avoir introduit dans le monde de la recherche, en 1996, à l'occasion d'un stage d'été.
- François Charpillet, mon directeur de thèse, qui m'a guidé et soutenu durant toutes ces années. Je regrette de lui avoir reproché de me pousser à publier ; je comprends maintenant qu'il s'agit d'un élément important de la vie d'un chercheur.
- Anne Boyer et Laurent Vigneron qui m'ont permis d'enseigner à mon premier groupe d'étudiants ; depuis lors, l'envie d'enseigner ne m'a pas quitté.
- Brigitte Jaray, qui m'a encadré tout au long de mes enseignements en tant que moniteur.
- Pierre-Yves Durand et Etienne Junke, avec qui j'ai régulièrement collaboré au sein de projets médicaux. (Respectivement, Diatelic et l'assistance à l'anesthésie)
- Alain Dutech qui, plus que tout autre, m'a aidé dans le processus de correction de mon manuscrit en m'indiquant les problèmes et les incohérences, mais aussi en proposant des suggestions, voir même des ébauches de solutions.
- Martine Kuhlmann, et Nadine Beurné, qui se sont échinées pour remplir (et faire remplir) les divers documents administratifs qu'on leur rend toujours en retard, et qui, malgré leur charge de travail, ne refusent jamais de nous conseiller et de nous aider.
- Franck, mais aussi Alain, Matthieu et Renato, avec qui j'ai pu construire Simplet, notre robot mobile, et sans qui ce projet n'aurait certainement jamais abouti.
- Arnaud (qui nous a abandonnés pour aller se cacher au Canada), David, Jack, Olivier, et bien sûr Alain et Dominique. Grâce à vous j'ai passé d'excellentes soirées, certes mouvementées mais au combien appréciées. Je regrette d'autant plus leur raréfaction.
- Mes collègues et amis qui ont désespérément tenté de me socialiser : Alain, Armelle, Eric, Franck, Iadine, Loïc, Olivier, Régis, Vincent (Le Schniouf, et le Profond), et Romaric principalement, mais aussi tous les autres que j'oublie.
- Grom. pour la question Java du jour, qui revenait plus ou moins régulièrement selon les périodes. Merci aussi pour tous les coups de main que tu as pu m'apporter et que je n'ai pu te rendre. A charge de revanche.

Et pour finir, je souhaite remercier mon jury de thèse qui a su apprécier mon travail, qui m'a proposé des améliorations possibles, et surtout qui a jugé de la qualité de mon travail de façon bien plus généreuse que je ne l'aurais fait moi-même.

Table des Matières

Table des Figures

1 Introduction

L'intelligence artificielle est une branche de l'informatique qui vise à la résolution de problèmes qui nécessiteraient de l'intelligence s'ils étaient résolus par un être humain. Au moment où ce nouveau courant est apparu, les machines se bornaient à imiter le raisonnement humain. En particulier, les systèmes à base de règles permettent l'application automatique de règles d'inférence fournies par un expert du domaine. Depuis, de nombreux progrès ont été faits, et principalement dans le domaine de l'apprentissage. Les méthodes d'apprentissage visent plus spécifiquement à construire un modèle de raisonnement de façon automatique, déchargeant le programmeur de cette tâche. Pour y arriver, on établit en général un modèle du problème à résoudre. Ce modèle est une abstraction du problème à résoudre, généralement formelle, et qui permet de conduire un raisonnement adapté.

Deux grandes classes de problématiques peuvent être dérivées du terme apprentissage. On peut tout d'abord essayer de déduire un modèle du système étudié en modélisant le fonctionnement de ce dernier aussi finement que possible. L'autre facette majeure de l'apprentissage consiste à construire un modèle de comportement permettant d'accomplir une tâche donnée. Cette dernière approche peut être basée sur un modèle explicite du système ou non. Ce qui importe, ce n'est pas de modéliser le fonctionnement de ce dernier, mais bien de fournir une décision pertinente.

L'introduction de l'intelligence artificielle en médecine est un fait relativement nouveau, puisque ses débuts datent des années 80. En effet, la maturation de cette science jeune qu'est l'informatique tend à démocratiser cette nouvelle ressource, aussi bien en terme de matériel qu'en terme de connaissance. L'explosion des technologies des réseaux n'a fait qu'amplifier cette tendance, offrant des possibilités auxquelles nul ne songeait 20 ans auparavant. L'apprentissage prend dans ce domaine un sens tout particulier, car on ignore encore en grande partie le fonctionnement réel des pathologies étudiées. Dans le meilleur des cas, les médecins possèdent une théorie assez élaborée, basée sur de nombreuses observations cliniques. Cependant, il est extrêmement difficile pour eux de valider une telle connaissance. De plus, les patients présentent une variabilité importante, ce qui ne fait que rendre leur travail encore plus compliqué s'ils veulent comprendre finement le processus sous-jacent.

Pour ma part, l'aventure a commencé lors d'un stage effectué durant l'été, en 1997. Le but de ce dernier était de formaliser les règles de diagnostic utilisées par les médecins de l'ALTIR pour le suivi de leurs patients dialysés à domicile. L'ALTIR est l'association lorraine pour le traitement de l'insuffisance rénale, basée à Brabois, à proximité du Centre Hospitalier Universitaire. Elle regroupe principalement des médecins néphrologues (spécialistes des problèmes rénaux) et le personnel médical nécessaire au fonctionnement d'un tel centre. Ce stage a été un véritable tournant pour moi. Ce projet m'a en effet convaincu qu'une machine pouvait aider les humains dans des tâches importantes, fussent-elles aussi complexes que la surveillance de personnes malades, en danger de mort permanent.

L'ajout d'un système informatique permet en effet d'assurer le suivi constant de ces patients, alors qu'ils demeurent à domicile. Cette surveillance permanente devrait donc améliorer leur sécurité. Ajouter un module d'intelligence artificielle à ce système permet alors de trier les informations, de les recouper, et d'en déduire une série de conseils et d'alertes permettant d'aider les médecins dans leur tâche.

En particulier, en attirant l'attention des néphrologues sur les patients qui montrent des symptômes alarmants, le système permet aux médecins de passer moins de temps à surveiller de nombreux patients bien portants. Ce temps économisé peut alors être employé à résoudre les problèmes des malades qui en ont besoin.

Par la suite, je me suis rendu compte que ce système pouvait également enseigner aux médecins des choses intéressantes, faisant progresser la médecine en même temps que l'informatique. Cette étape intervient plus particulièrement lorsqu'un évènement inattendu se trouve prédit par le système, avant que l'équipe médicale n'en prenne conscience. En effet, en permettant le recoupement de nombreuses données, cette analyse peut mettre en évidence des relations qui, autrement, seraient plus difficiles à découvrir.

Alternativement, la non–prédiction d'un élément avéré permet a posteriori d'affiner les règles de diagnostic du système de façon à être capable de prédire un tel événement s'il devait se reproduire. Ces règles affinées permettent en retour de faire progresser les connaissances des médecins en suggérant de nouvelles approches, ou en modérant des lois tenues pour acquises.

Mon expérience au sein de Diatelic a rendu les exercices jouets auxquels l'intelligence artificielle s'attaque habituellement complètement insipides. C'est pourquoi je me suis fermement attaché à ce que les algorithmes développés restent applicables sur des cas réels. La médecine, de par ses profondes implications dans la vie courante, en est un exemple particulièrement intéressant. Cependant, ce type d'application pose de nouveaux problèmes, tant au niveau moral qu'au niveau informatique. Ainsi, l'éthique et la responsabilité qu'impliquent des décisions thérapeutiques doivent être évaluées sérieusement avant toute expérimentation.

Au niveau de l'informatique, cela pose de sérieuses restrictions sur les algorithmes utilisables. Ces deux classes de contraintes sont étroitement liées. Par exemple, il semble évident que le système n'a pas droit à l'erreur; cela implique donc un rejet a priori des politiques de recherche de solution par essais et erreurs. Par extension, cela inhibe également la plupart des politiques d'exploration, du moins tant qu'un patient est connecté au système. Néanmoins, je montrerai qu'il est possible d'envisager de tel les approches, sous réserve d'imposer des garde-fous au système.

J'ai ensuite considéré l'application de la structure élaborée dans le cadre de Diatelic à l'assistance à l'anesthésie. Cette évolution m'a permis de souligner certains points particulièrement difficiles, mais aussi de souligner l'adéquation de l'architecture que nous proposons à ce type de suivi. En effet, dans le cadre de la surveillance d'une anesthésie, les contraintes temporelles sont beaucoup plus fortes, ce qui est à la fois un avantage et un inconvénient. En effet, si l'on dispose de nettement moins de temps pour prendre une décision, les perturbations de l'état du patient sont aussi plus limitées. Son évolution est donc mieux contrôlée, et plus prévisible. De plus, il est beaucoup plus facile d'obtenir des données en grand nombre, car le patient se trouve dans un environnement contrôlé, et doté d'instruments de mesure performants.

Avec ces deux applications j'ai pu mettre en évidence que la tâche est ardue, mais pas impossible. Un effort de généralisation de l'architecture de ces systèmes m'a alors permis de clarifier énormément de points, tout en permettant son application à d'autres problèmes, parfois très éloignés. Je montrerai donc que, sans nécessiter de profondes modifications, le modèle de diagnostic et d'apprentissage que je présente peut être appliqué à des problèmes venant d'horizons très différents.

Tout cela a conduit à la réalisation d'une bibliothèque logicielle générique, associée par la suite à une interface de développement visuel qui facilite sa mise en œuvre. L'application de cette bibliothèque à la robotique mobile permet de mettre en relief le fait que cette architecture logicielle n'est pas limitée au diagnostic médical, mais qu'elle recouvre une classe de problèmes très généraux.

Ce nouveau cadre applicatif m'a permis de souligner des problèmes que j'avais seulement entrevus dans les problématiques médicales. En effet, je suis plus familier avec le mouvement d'un robot qu'avec l'évolution d'un patient. Cette dernière application a donc non seulement élargi ma vision des choses en offrant des techniques comme la planification, mais elle m'a surtout permis de comprendre certains comportements observés auparavant.

Je présenterai donc tout d'abord le cadre applicatif de mon travail, de façon à expliciter les aspects non directement liés à l'informatique des applications. Cette description débouche naturellement sur une palette de problèmes à résoudre. Parmi ceux-ci, les plus importants sont clairement l'interaction avec les médecins et la prise en compte de notre ignorance dans les règles d'évolution des pathologies médicales.

Après cette digression sur le cadre général des applications envisagées, je montrerai comment divers modèles de diagnostic peuvent être employés, et surtout comment les faiblesses des uns sont compensées par les avantages des autres. Ainsi, j'expliquerai tout d'abord une approche basée sur des systèmes à base de règles, méthode qui permet de traduire les règles de diagnostic utilisées par les médecins en une forme exécutable par une machine. Les difficultés que rencontre ce type de modèles, et plus particulièrement les problèmes que pose la variabilité des patients, nous conduisent naturellement à des modèles stochastiques.

En particulier, les processus Markoviens partiellement observables permettent la prise en compte de notre relative ignorance de la physique des pathologies. En effet, les médecins cherchent toujours à tirer des règles des observations cliniques auxquelles ils procèdent. En première approche, ces modèles sont toujours basés sur des moyennes, des écart-types, en bref, des statistiques. Utiliser des modèles informatiques basés sur ces mêmes principes mathématiques permet donc à nouveau de traduire leurs observations sous une forme exécutable par la machine. Seulement, ces modèles tiennent compte du fait qu'ils sont imparfaits, et que des divergences sont possibles.

J'ai pu mettre en évidence un autre avantage de ces modèles : grâce au choix d'un modèle adapté à la collaboration avec des médecins, le fonctionnement intrinsèque du processus peut être rendu compréhensible de façon triviale par les médecins. Leur interaction avec le modèle étant rendue plus facile, il leur est alors possible de créer un modèle par patient, ce qui aboutit naturellement à la notion de profil. Grâce à l'ajout de ces profils personnalisés, le modèle générique peut être spécifié et adapté à chaque patient. Ces modèles spécifiques permettent une analyse plus fine de chaque patient, ce qui mène à un suivi plus précis, et moins d'alertes erronées.

L'adaptation de ces profils par le médecin nous a conduits naturellement à envisager une adaptation automatique par la machine. J'expliquerai donc les techniques d'apprentissage qui peuvent être employées avec ces modèles, en tenant compte des contraintes imposées par notre cadre applicatif. En particulier, je montrerai comment une descente de gradient permet au système d'adapter son modèle pour prendre en compte la modification de son diagnostic par le médecin.

Cette approche autorise donc le personnel médical responsable des patients à guider le modèle de diagnostic, sans pour autant nécessiter la compréhension complète de l'implication de chaque paramètre dans le diagnostic. Outre une plus grande facilité d'emploi, cela permet naturellement une plus grande sécurité, car je montrerai que des paramètres tels que la stabilité numérique du modèle peuvent être pris en compte aisément. Ainsi, le modèle obtenu est plus robuste, tout en garantissant que la sémantique du diagnostic produit sera compatible avec celle des médecins.

Afin d'unifier ces diverses composantes, je présenterai alors comment cette approche peut être généralisée afin de tenir compte des différentes contraintes des divers problèmes possibles. De plus, des considérations bassement matérielles montrent que les chercheurs désirant appliquer leurs algorithmes à des cas concrets passent finalement plus de temps à écrire et réécrire du code faisant toujours la même chose. Pour contrer cette tendance, j'ai réalisé une bibliothèque logicielle permettant de mettre en commun une grande variété d'algorithmes réutilisables facilement.

Naturellement, cette approche conduit à écrire un code permettant de créer des instances de modèles adaptées au problème à résoudre, et à les relier les unes aux autres. Afin de soulager le chercheur de cette nouvelle phase très mécanique, je présenterai une interface de développement visuel permettant de créer un module mettant en œuvre diverses parties de la bibliothèque de façon intuitive. Toute cette conception se faisant sans souci de langage de programmation, l'utilisateur peut alors se consacrer à l'évaluation des modèles nécessaires, et au choix de leurs paramètres.

Finalement, j'espère montrer que l'application de l'intelligence artificielle à des exemples réalistes est intéressante, non seulement par son ancrage dans la vie courante, mais surtout parce qu'elle permet en même temps de faire progresser la recherche fondamentale en soumettant de nouveaux problèmes à résoudre, et de nouveaux écueils à franchir.

2 Un cadre applicatif

2.1 Motivation

Dans le cadre de la recherche en intelligence artificielle, nous sommes souvent confrontés à des problèmes particulièrement compliqués. En effet, les problèmes réels auxquels nous avons envie de trouver une solution informatique intelligente sont souvent complexes et imposants. Cette difficulté inclut aussi bien la difficulté à abstraire le problème pour en extraire une définition claire et formelle, que la difficulté à mettre en œuvre les algorithmes susceptibles de les résoudre. Pour contourner cet écueil, on utilise alors généralement des problèmes dits *académiques*. Ces problèmes sont des cas simplifiés et bien définis sur lesquels on peut tester des algorithmes, et surtout les comparer aux travaux que d'autres équipes mènent de par le monde. Une fois que les algorithmes ont été soigneusement étudiés sur ces problèmes-jouets, on peut espérer les transposer sur des applications réelles.

Aujourd'hui, de nombreux algorithmes ont fait leurs preuves sur ces problèmes, mais restent plus ou moins cantonnés à ces derniers. En effet, les applications réelles posent des problèmes d'un ordre différent de la recherche fondamentale, et sont souvent délaissées par les chercheurs. Ainsi, afin de pouvoir passer d'un algorithme qui fonctionne sur un problème de taille réduite, à une version fonctionnelle sur un cas concret, il est souvent nécessaire d'écrire un grand nombre de lignes de programme. Une fois que ce code est écrit, testé et corrigé, il est souvent nécessaire de l'optimiser afin qu'il s'exécute en un temps raisonnable. Enfin, les applications réelles sont souvent plus complexes que les exemples académiques. Ainsi, bien que l'algorithme fonctionne dans 90% des cas, il suffit d'un exemple qui sorte un peu du cadre formel du projet pour empêcher l'application de fonctionner. Le problème vient alors du fait que ces 10% manquants sont bien plus difficiles à obtenir que les autres. Il faut tenir compte de nombreux cas particuliers, ce qui prend évidemment beaucoup de temps.

Peu de chercheurs sont décidés à prendre tout ce temps nécessaire à parfaire leurs algorithmes. Ils préfèrent souvent passer à l'étude de nouvelles variantes, de nouvelles options, voire même à d'autres algorithmes afin d'obtenir de meilleurs résultats sur les mêmes 90% des cas qui fonctionnent. Je pense pourtant que l'application de ces algorithmes à des cas concrets pose de nouveaux problèmes, au moins aussi intéressants pour l'informatique fondamentale que l'accroissement des performances évaluées sur les problèmes–jouets habituels. C'est pourquoi je me suis concentré sur des méthodes permettant de porter ces algorithmes vers des applications intéressantes telles que la médecine.

Je vais donc présenter dans ce chapitre les points caractéristiques de trois applications différentes, en montrant bien leurs buts et leurs implications. Tous ces problèmes sont centrés sur la notion de diagnostic, mais je vais montrer que cette notion est plus générale qu'il ne le semble. En effet, je présenterai tout d'abord deux applications portant sur la médecine, avant de m'intéresser à la navigation d'un robot mobile. Chacun de ces cas présente des contraintes spécifiques, tant au niveau de leur contexte et des données disponibles que dans les objectifs qu'ils convoitent. Cependant, je montrerai comment on peut généraliser cette description afin d'obtenir un modèle global permettant d'unifier toute une classe de problèmes. Ce modèle servira par la suite de fil conducteur tout au long des chapitres du mémoire. Je me concentrerai alors sur l'application à chacun des problèmes, en insistant sur les spécificités de chacun et sur les problèmes qu'elles posent.

2.2 Présentation spécifique de chaque application

2.2.1 Le projet DIATELIC

Ce projet est situé dans le cadre de la télémédecine et, de manière plus spécifique, de la surveillance de patients dialysés traités à leur domicile par une dialyse péritonéale continue ambulatoire. Je vais donc commencer par expliquer la nature des troubles médicaux dont souffrent les patients avant de présenter les objectifs de ce projet et les moyens techniques que cela implique.

L'insuffisance rénale conduit à une perte graduelle des facultés d'épuration de l'organisme et à une atteinte de la régulation hydrique. Cette pathologie nécessite la mise en œuvre de soins palliatifs au dysfonctionnement des reins. Pour ces soins, deux grandes approches sont envisageables : la greffe d'un ou plusieurs reins, et la dialyse. La première nécessite évidemment la disponibilité d'un donneur compatible, ce qui reste un cas assez rare. La seconde est donc souvent utilisée dans l'attente d'un donneur éventuel.

2.2.1.1 Cadre médical

La dialyse est un processus qui consiste à filtrer le sang du patient, complétant ou remplaçant le travail fait normalement par les reins. L'objectif de ce filtrage est multiple, car le rein a de nombreuses fonctions. Ainsi, il est chargé de la régulation de l'hydratation du corps, et de l'élimination de différentes toxines. Actuellement, trois grandes méthodes de dialyses sont utilisées régulièrement : l'hémodialyse, la dialyse péritonéale automatisée (DPA), et la dialyse péritonéale continue ambulatoire (DPCA).

a) L'évolution de la dialyse au cours du temps

L'hémodialyse est l'approche la plus ancienne, puisqu'elle a été étudiée chez l'animal dès 1912 par Abel, Rowntree et Turner. Cependant, la première dialyse d'un être humain réalisée avec succès n'a été obtenue qu'en 1945. A partir de 1960, grâce aux progrès réalisés sur les reins artificiels, cette méthode s'est progressivement généralisée. Elle peut en effet s'appliquer à la plupart des patients, avec très peu de restrictions. Cette méthode consiste à créer une circulation sanguine extracorporelle. On fait donc passer une partie du flux sanguin du patient à travers une machine qui filtre les résidus toxiques et régule la quantité d'eau présente dans le sang. En général, chacune de ces séances dure plusieurs heures, et deux à trois séances sont nécessaires par semaine. Il s'agit donc d'un traitement lourd et difficile à supporter par les patients, puisqu'il nécessite leur présence dans un centre de dialyse, plusieurs nuits par semaine.

Ganter, dès 1923, a proposé une nouvelle méthode de dialyse : la dialyse péritonéale. Cette approche consiste à utiliser le péritoine comme filtre naturel. Elle a donc été beaucoup plus facile à mettre en œuvre que l'hémodialyse, puisqu'elle ne nécessite pas de circulation sanguine extracorporelle. En particulier, cette méthode ne souffre d'aucun problème lié à la coagulation du sang dans les tubulures. Elle est donc particulièrement adaptée aux patients tolérant mal les traitements anticoagulants. Néanmoins, avec les améliorations successives des reins artificiels, de plus en plus faciles à mettre en œuvre et de plus en plus sûrs, l'utilisation de cette technique a beaucoup diminué depuis 1960. Les différents types de dialyse péritonéale sont discutés dans (18), avec leurs avantages et leurs inconvénients.

b) La technique de la dialyse péritonéale

Le péritoine est une poche naturelle, située dans l'abdomen, qui sépare en particulier les organes vitaux des côtes. La paroi de cette poche possède de très nombreux vaisseaux sanguins de petite taille, les capillaires, qui irriguent toute sa surface pariétale. Cela fait du péritoine un site idéal de transfert osmotique. L'osmose est un phénomène physique qui permet à une solution de traverser une paroi semi-perméable : lorsqu'une paroi semi-perméable sépare deux liquides dans lesquels sont dissous des éléments, on peut observer que la quantité de ces éléments de chaque côté de la paroi (la concentration de chacune des solutions) s'équilibre progressivement. La paroi des capillaires étant suffisamment fine pour permettre cet échange, il est possible d'utiliser le péritoine comme filtre osmotique naturel.

Le patient est donc opéré afin d'ajouter un cathéter au bas du péritoine, en ménageant une sortie à travers la paroi abdominale. Cette *émergence*, comme l'appellent les néphrologues, permet alors de remplir le péritoine avec environ deux litres d'une solution de sérum physiologique. Cette solution contient des additifs qui permettent de faire varier l'effet de ce rein artificiel. Par exemple, en augmentant sa concentration, l'osmose va drainer plus d'eau vers le péritoine de façon à diminuer la concentration de la solution qui s'y trouve. Inversement, si l'on injecte une solution dont la concentration est inférieure à celle du sang, l'eau contenue dans le péritoine va être transférée vers le flux sanguin de façon à équilibrer les concentrations. En fonction de la nature de l'additif, selon qu'il est absorbé ou non par le flux sanguin, le transfert sera plus ou moins régulier tout au long de la stase. Les principaux additifs utilisés sont le glucose, qui est absorbé par l'organisme, et l'icodextrine qui ne l'est pas.

De façon évidente, il convient de surveiller régulièrement l'état de l'émergence, car elle est très sujette à inflammation, et une infection est à craindre à chaque manipulation. Or une dialyse péritonéale peut demander de nombreuses manipulations à chaque changement de la solution. C'est donc une source possible de troubles liés à cette méthode. Néanmoins, cette dernière ne nécessite pas l'emploi de substances anticoagulantes, puisqu'aucune circulation extracorporelle n'est employée.

La dialyse péritonéale automatisée utilise ce principe : le patient se rend dans un centre de dialyse, où son émergence est reliée à une machine qui va successivement préparer, injecter, laisser reposer et retirer plusieurs litres de sérum physiologique. A chaque transfert, le poids du fluide est mesuré avant et après la dialyse, et le différentiel correspond à peu près à la quantité d'eau qui a été drainée. Le médecin responsable du traitement peut donc paramétrer l'opération de façon à maintenir le patient dans un état de santé correct. En effet, la machine propose d'ajuster la concentration de la solution à injecter en mélangeant divers produits selon les directives du médecin. De même que pour la dialyse standard, c'est un traitement lourd, qui dure plusieurs heures, et qui est répété plusieurs fois par semaine.

Finalement, la dialyse péritonéale continue ambulatoire constitue l'une des dernières avancées de la dialyse (55). Son principal avantage est de pouvoir s'effectuer au domicile des patients, et non en centre spécialisé. Pour arriver à ce résultat, le patient doit remplir lui-même son péritoine avec des poches de sérum étalonnées. Trois principaux types de poches, de concentrations différentes, sont utilisés : les isotoniques, les hypertoniques, et les medium, de concentration intermédiaire. Les poches isotoniques ont une concentration approximativement égale à celle de l'organisme, alors que les poches hypertoniques sont plus concentrées que les deux autres.

Un patient utilise trois à quatre poches par jour. Une poche reste en stase pendant environ 4 heures dans le péritoine, temps pendant lequel le filtrage osmotique se déroule naturellement. Au bout de 4 heures, le système lymphatique qui draine en permanence le contenu du péritoine a absorbé suffisamment du sérum mis en stase pour que le transfert s'inverse. Conserver plus longtemps la poche n'apporte donc pas d'amélioration.

Lorsque le patient retire la poche, il la pèse et consigne cette valeur sur une fiche. Il doit également noter le niveau de turbidité de la poche en lisant un code situé au dos de l'étiquette de la poche, à travers le liquide qu'elle contient. Cette lecture permet donc de juger facilement si une poche est trouble, ou si elle est limpide. Une poche trouble est en effet un indice de la présence de fibres, ce qui peut révéler un début de pathologie ou de détérioration du péritoine.

c) Avantages et inconvénients de la méthode

A part l'injection quotidienne de leurs poches et la mesure de quelques paramètres médicaux, les patients peuvent vivre presque normalement. Cette méthode de soin est donc la plus agréable à vivre pour eux, puisqu'ils retrouvent une grande partie de leur autonomie. Elle pose cependant de graves problèmes quant à leur sécurité. En effet, puisque les patients opèrent leur dialyse tous seuls, à leur domicile, la surveillance qu'assurent les médecins est fortement amoindrie. Pour améliorer leur sécurité, les patients sont donc suivis par leur néphrologue à raison d'une visite mensuelle. Pour accroître encore cette sécurité, une infirmière peut passer voir le patient régulièrement. Cela lui permet de juger de l'état général du patient et de l'aider dans ses soins. Par exemple, elle peut prendre la tension, surveiller l'état de l'émergence et évaluer le niveau de turbidité des poches. Enfin, l'infirmière doit surtout avertir le médecin dans le cas où elle suspecte un problème. L'infirmière est donc plus particulièrement recommandée pour les patients qui ne sont pas totalement autonomes.

Les risques sont essentiellement liés aux erreurs de manipulations qui peuvent entraîner des infections graves telles que des péritonites, et à une mauvaise gestion du niveau d'hydratation du patient. Il convient donc de surveiller étroitement ce niveau. Cependant, comme les reins continuent un certain temps à éliminer une partie de l'eau superflue et que le patient continue à boire et à manger normalement, l'évolution de l'hydratation du patient est difficile à évaluer par avance. Cette évaluation se base principalement sur la surveillance du poids. C'est en effet un paramètre médical particulièrement facile à mesurer, tout en restant lié directement à l'hydratation du patient. Simplement, si le poids du patient augmente d'un kilogramme en une journée, il y a un risque non négligeable que ce soit un kilogramme d'eau stocké par le patient. Pour que cette estimation soit pertinente, en lissant les variations journalières du poids, une valeur de référence stable est nécessaire. Cette dernière est calculée à l'hôpital dans la période qui précède le retour du patient à son domicile. Ce poids de référence est nommé *poids-sec* par les médecins.

Une mauvaise estimation du poids-sec par le médecin peut entraîner des troubles tels que des hyperhydratations ou des déshydratations en faussant le jugement du niveau d'hydratation du patient par le médecin. Par exemple, un poids-sec trop élevé peut amener à considérer une situation normale comme une déshydratation, puisque le poids est au-dessous de sa valeur supposée idéale. Le médecin va alors vouloir réduire le traitement, de façon à permettre à l'organisme de retrouver son équilibre en accumulant plus d'eau. Comme le patient était en fait dans un état normal d'hydratation, il va tendre vers un état d'hyperhydratation imposé par le traitement. Il est donc nécessaire de bien évaluer le poids-sec de chaque patient afin de pouvoir se baser sur le poids comme indicateur.

Bien entendu, le patient peut maigrir ou grossir de façon naturelle, ce qui impliquera une réévaluation du nouveau poids-sec. Tout le problème vient alors de cette ambigüité entre la détection d'un poids-sec mal réglé et la détection d'un trouble de l'hydratation. En effet, comme le patient reste à son domicile durant le traitement, cette détection ne peut se baser que sur la variation des paramètres physiologiques. Or, une augmentation de poids peut aussi bien être due à une prise de poids naturelle qu'à une rétention d'eau. Séparer ces deux causes est donc un problème difficile, mais primordial. Heureusement, une variation de l'hydratation d'un patient donné à d'autres influences que le seul poids. Pour la détecter, il faut donc rechercher plusieurs paramètres médicaux concordants, tels par exemple le poids et la tension artérielle.

Les troubles de l'hydratation sont particulièrement importants à diagnostiquer, car ils peuvent avoir des effets à court terme et à long terme très graves. De plus, l'installation de ces troubles peut se faire de façon très insidieuse. Une hyperhydratation prolongée peut ainsi entraîner l'apparition d'œdèmes. Les œdèmes sont des poches d'eau situées anormalement, comme par exemple dans les poumons, ce qui est le cas le plus fréquent. Une hyperhydratation va également entraîner une hausse de la tension artérielle, qui va à son tour perturber l'organisme. Une analyse plus précise de ces troubles est menée dans (37) (chapitre 4.5).

L'hyperhydratation est un trouble particulièrement vicieux, car le patient ne montre aucun signe extérieur d'une quelconque maladie. Bien souvent, ses proches lui trouvent plutôt une bonne mine, au contraire de la déshydratation qui est accompagnée de signes très visibles. Pourtant, on peut souvent constater a posteriori que l'étude attentive des données de dialyse rassemblées dans les jours précédant l'apparition d'un oedème aurait pu permettre de le prévoir, et donc de l'éviter.

2.2.1.2 La télémédecine

La télémédecine est l'évolution naturelle de la médecine classique, une fois couplée avec les technologies de l'information et des réseaux. Son but principal est de permettre l'application à distance du savoir des médecins, mais de nombreuses variantes sont possibles.

Depuis une dizaine d'années, plusieurs expériences d'application de télémédecine ont été menées de par le monde. Elles sont essentiellement basées sur deux grands axes : la visioconférence entre le patient et son médecin, et la mise à disposition des données médicales. Ainsi, (6) présente une méthode de compression de données vidéo pour permettre un transfert et un stockage optimal de ces dernières. Pendant ce temps, (33), (36) et (62) mettent en évidence les modalités et les intérêts du stockage et de la mise à disposition des données médicales des patients pour les médecins. Par ailleurs, (13) présente un système australien recouvrant ces principes. Cependant, ce système semble encore en phase de pré-développement, et je n'ai trouvé aucune information sur des résultats éventuels. Un principe similaire est proposé par Bellazzi et al. dans (5). Cependant, le système semble axé beaucoup sur la vidéoconférence et le stockage des données.

De son côté, (53) propose l'idée suivante : chaque patient transporte un micro-ordinateur qui amasse les données médicales du porteur et les mets à disposition des médecins. Pour sa part, Anderson travaille sur les problèmes de cryptographie que posent le transfert et le stockage des données médicales (2).

Les travaux menés à Nancy dans le cadre du projet DIATELIC se démarquent de ces expériences en axant le projet non seulement autour du transport de données, mais surtout sur l'analyse de celles-ci.

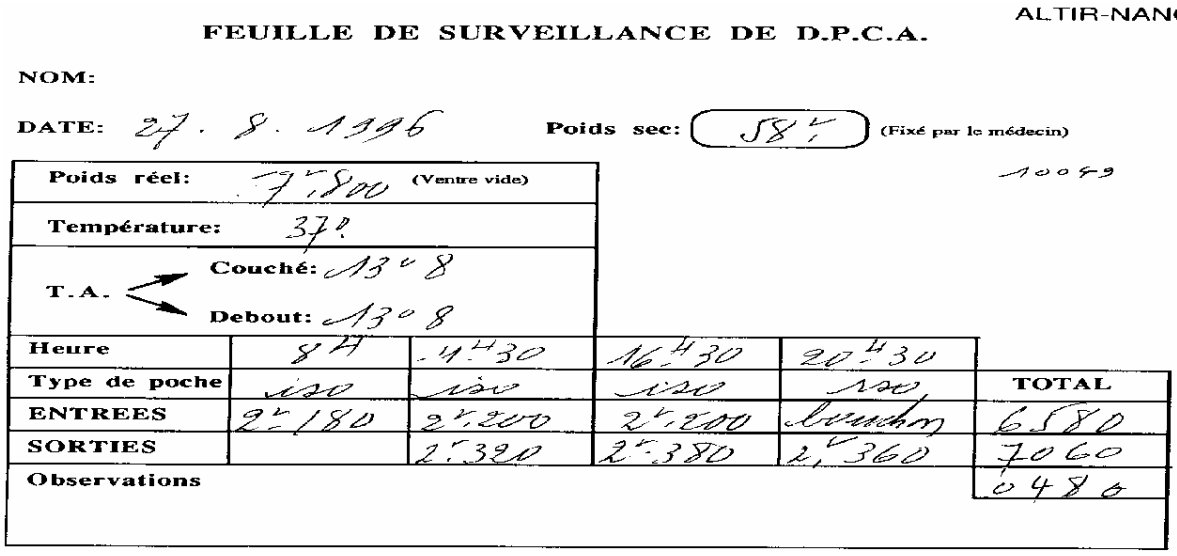
Dans le système Diatelic, chaque patient dispose d'un service personnalisé qui permet d' assurer le suivi quotidien de sa pathologie. Le suivi manuel quotidien de nombreux patients est cependant difficilement envisageable, parce que cela entraînerait une surcharge de travail pour les néphrologues. Un néphrologue suit en effet une centaine de patients simultanément. Il ne peut donc pas raisonnablement évaluer l'état de chacun d'entre eux chaque jour, tout en assurant les visites mensuelles et la gestion des urgences.

2.2.1.3 Le système mis en place

L'architecture que nous proposons est basée sur le principe du client-serveur. Un serveur est installé au LORIA, et les postes clients sont répartis chez chaque patient d'une part, et à disposition des différents médecins d'autre part. Les patients sont connectés au système grâce au réseau téléphonique classique, utilisant un simple modem. La sécurité des données est assurée d'une part par une identification par un système de login et de mot de passe, et d'autre part par un rappel du serveur. En effet, chaque patient a un numéro fixe à partir duquel il peut se connecter au système. Lorsqu'il désire envoyer ses données journalières, il appelle le serveur et s'identifie. Le serveur appelle alors le numéro correspondant à l'identifiant utilisé , afin d'effectuer le transfert des données. Ce dernier transfert est anonyme, car toute référence au nom du patient a été soigneusement éliminée.

Les postes des médecins sont reliés au serveur grâce au réseau internet. L'identification est réalisée également par un système de mot de passe permettant de protéger l'accès aux données stockées par le système. La confidentialité des données est assurée par le fait que toutes les données sont anonymes. Seuls les médecins ont la liste de la correspondance entre l'identifiant des patients et leur identité. De cette façon, seuls les identifiants circulent sur le réseau. De même, les données sont stockées dans une base de donnée sans aucune référence à l'identité réelle des patients. Seul l'identifiant fourni par les médecins est conservé. Un piratage éventuel des données utilisées par le système est donc difficilement exploitable.

Chaque jour, chaque patient entre ses données dans une fiche informatisée qui remplace la fiche papier habituelle. Ces deux fiches ont une présentation similaire, de façon à permettre une adaptation aisée des patients et des médecins au système. La Figure 1 présente la fiche papier que remplissent les patients non reliés à DIATELIC ; parallèlement, la Figure 2 montre la fiche informatique utilisée dans le système.



ALTIR-NANCY

FEUILLE DE SURVEILLANCE DE D.P.C.A.

NOM:

DATE: 27. 8. 1996 Poids sec: 58,7 (Fixé par le médecin)

10053

Poids réel: 57,800 (Ventre vide)					
Température: 37°					
T.A. Couché: 13° 8 / Debout: 13° 8					
Heure	8H	14H30	16H30	20H30	
Type de poche	1500	1500	1500	1500	TOTAL
ENTREES	2,180	2,200	2,200	illisible	6,580
SORTIES		2,320	2,380	2,360	7060
Observations					0480

Figure 1 : Les fiches–papier utilisées en dialyse

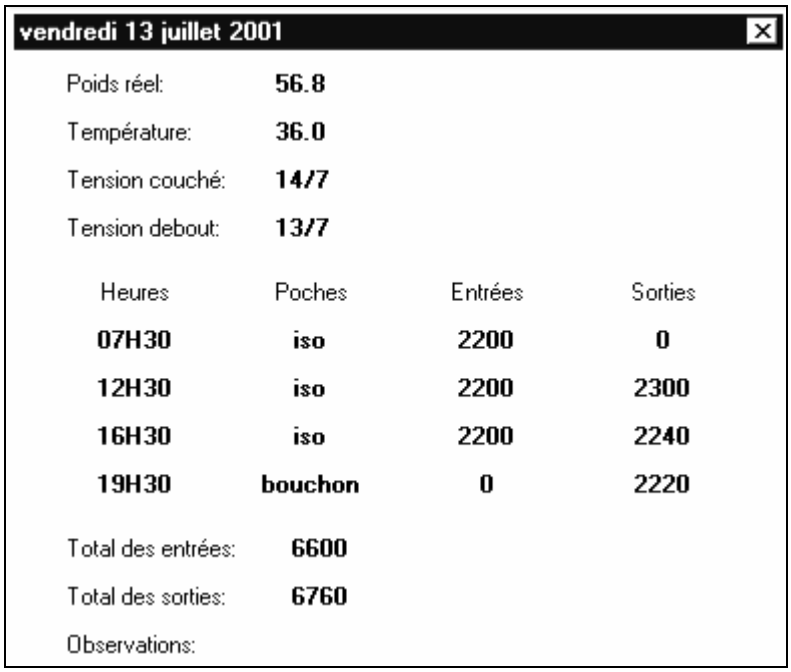


Figure 2 : Les fiches informatisées de dialyse

On peut donc voir la similitude entre ces deux fiches, ainsi que la relative pauvreté des données qu'elles contiennent. En plus de la traditionnelle fiche où sont consignées les données médicales, une messagerie électronique est mise à la disposition des patients. Celle-ci permet de faciliter la communication entre les néphrologues et leurs patients. De cette façon, les patients peuvent laisser des messages à destination des médecins et peuvent leur poser des questions. En retour, l'équipe médicale peut répondre par la même messagerie ou, si le besoin s'en fait sentir, par téléphone. Les médecins ont remarqué que cette messagerie a permis aux patients de poser plus de questions qu'ils ne le font habituellement. En effet, le médecin peut consulter les messages quand il en a le temps. Un appel téléphonique, au contraire, peut tomber au beau milieu d'une consultation, ou d'une autre tâche urgente. Les patients ont donc moins de scrupules à poser des questions qu'ils jugent peu importantes, ou non urgentes. Par effet de bord, cela permet donc d'augmenter la compréhension du patient envers sa maladie, son traitement, et ainsi de suite. On peut donc espérer en retirer un gain au niveau de l'application des directives des néphrologues. Ce gain a été observé, mais il est difficile de déterminer s'il est dû à la messagerie, ou au sentiment d'êtr e mieux suivi.

Lorsque le patient valide sa fiche de données quotidienne, celle-ci est transmise directement au serveur où elle est consultable à volonté par les médecins. En parallèle, un système *intelligent* analyse ces données et les compare au profil du patient. L'objectif premier de ce module consiste à générer des alertes dans le cas où les paramètres médicaux semblent annoncer une pathologie. Il se doit donc d'être suffisamment sensible pour détecter des évolutions insidieuses, mais suffisamment discriminant pour ne pas alerter les néphrologues sans raison valable. Grâce à un tel système, les médecins devraient pouvoir suivre plus de patients, tout en disposant de plus de temps pour gérer les problèmes de chacun.

Il est donc évident que la mise en place d'un tel système a demandé la participation active de nombreuses personnes, incluant en particulier l'équipe médicale de l'ALTIR, et de nombreux chercheurs du LORIA. Ma contribution à ce projet s'est limitée à l'étude, la conception et la maintenance du système de diagnostic informatisé. Ce module est en effet l'une des contributions majeures de mes recherches.

Une expérimentation sur 30 patients a débuté en 1999 et s'est terminée en août 2002. Le principe de cette étude et ses résultats sont présentés en annexe. Sans rentrer dans le détail, je peux déjà indiquer que les résultats sont très encourageants, et même au-delà de nos espérances. Pour cette raison, trois brevets ont été déposés (23),(24),(25), et une société destinée à offrir des services de ce type à une échelle plus importante a été créée.

2.2.1.4 Les données médicales utilisées

Quotidiennement, chaque patient envoie sa fiche au serveur. Le système doit alors analyser les données qu'elle contient de façon à alerter le médecin si une anomalie est détectée. Pour comprendre comment s'appliquent les algorithmes que je vais présenter dans les prochains chapitres, il est nécessaire de connaître les données médicales disponibles pour analyser l'état de chaque patient. Je vais donc expliciter chacun de ces paramètres, en expliquant leur source, leur nature, leur intérêt au point de vue médical, et les problèmes que pose leur exploitation.

a) Le poids

Le poids est mesuré par un pèse-personne classique. Cette valeur est comparée au poids-sec, valeur idéale fixée par le médecin. Comme chaque litre d'eau stocké par le patient pèse 1 kilogramme, cette comparaison fournit une appréciation directe de l'hydratation du patient. Cette mesure est pertinente, à condition cependant que la valeur de référence soit bien conforme à la physiologie actuelle du patient. Il est donc nécessaire de faire évoluer le poids-sec lorsque le patient grossit, ou lorsqu'il maigrit. Cette valeur est relativement sujette à des variations journalières, et plus particulièrement en fonction des repas que consomme chaque patient, de la température, et des efforts fournis par le patient.

b) La tension artérielle

La tension artérielle est prise par un système standard de brassard et de stéthoscope. Chaque jour, deux mesures sont effectuées. La première valeur est obtenue pendant que le patient est allongé, et la seconde est prise lorsqu'il se lève. Chacune de ces mesures donne deux valeurs différentes, basées sur le bruit que génère la pulsation du flux sanguin dans les vaisseaux : la tension *systolique* et la tension *diastolique*. Néanmoins, pour les besoins du calcul, nous avons besoin d'une valeur de tension unique. Nous utilisons donc la définition de la tension moyenne que l'on utilise classiquement en médecine :

$$TensionMoyenne = (Systolique+2*Diastolique)/3$$

En règle générale, la tension moyenne reflète assez finement le niveau d'hydratation du patient. En effet, plus le flux sanguin contient d'eau, plus son volume augmente. Plus le volume augmente, plus la pression exercée sur les parois des vaisseaux sanguins s'accroît. Finalement, plus cette pression augmente, plus la tension artérielle s'élève, et plus il faudra gonfler le brassard pour en masquer le bruit. Cette évolution dépend naturellement de l'élasticité des artères, valeur qui évolue naturellement avec l'âge du patient. Il est bon de noter que l'effet observé est exactement l'inverse chez un patient qui souffre d'insuffisance cardiaque.

Lorsque l'on étudie l'évolution de la tension d'un patient donné, on peut remarquer que cette dernière évolue naturellement, de façon assez lente, mais cyclique. Ses variations brusques sont donc plus intéressantes que sa valeur numérique brute. Cependant, pour éviter les aléas dus à des valeurs localement anormales, il est préférable de ne pas comparer la valeur du jour à la seule valeur de la veille. En effet, une augmentation localisée de la tension se traduirait alors par une baisse brutale de cette dernière dès le lendemain.

Bien entendu, il est nécessaire d'éliminer ces effets contradictoires et sans relation avec l'hydratation pour obtenir un diagnostic correct. Pour avoir une idée de la tension *normale* d'un patient typique, les médecins utilisent une moyenne portant sur la période des 15 jours précédant la consultation ; le système pourra donc comparer également la valeur de la tension quotidienne avec une valeur moyenne, calculée sur les 15 derniers jours de données envoyées par le patient. De cette façon, la mesure obtenue est comparable à celle utilisée par les médecins.

c) La tension différentielle

La différence entre les valeurs quotidiennement mesurées pour la tension lorsque le patient est debout et lorsqu'il est couché est nommée *tension différentielle.* Cet indicateur est également révélateur de l'état d'hydratation général du patient. Normalement, la tension qui est mesurée lorsque le patient est couché est légèrement supérieure à celle qui est mesurée lorsqu'il passe à une position verticale. En effet, la gravité tire alors brutalement le sang vers le bas du corps, ce qui réduit la pression observée dans les membres supérieurs.

Lorsque le patient se déshydrate, cette tendance s'amplifie, car le volume sanguin est insuffisant pour assurer le maintien de la pression. On observe alors une chute brutale de la pression artérielle, connue sous le terme d'*hypotension orthostatique*. C'est donc un facteur de diagnostic particulièrement intéressant, puisqu'il ne fait aucune référence à l'état moyen du patient. Ainsi, même chez un patient en début de traitement, cette valeur est suffisamment significative pour être utilisée directement. De plus, même lorsque le patient souffre d'une hypotension, ce facteur est clairement mesurable.

Par contre, ce facteur n'est pas influencé par un problème d'hyperhydratati on. Il ne peut donc pas révéler toutes les pathologies liées à l'hydratation.

d) Ultrafiltration

Le bas de la fiche contient le détail des poches utilisées pendant la journée par le patient. Pour chacune d'entre elles, la concentration du sérum est indiquée sous la forme du type de poche que le patient a utilisé. Additionnellement, l'heure à laquelle chaque poche a été injectée dans le péritoine, et l'heure à laquelle elle en a été retirée sont reportées dans les colonnes correspondantes.

De plus, chaque poche est pesée avant et après la dialyse, de façon à permettre l'analyse de l'effet de chacune d'entre elles. La différence de ces poids permet donc d'apprécier rapidement la quantité d'eau qui a été drainée pendant que le sérum résidait dans le péritoine. Cependant, on observe que les patients réagissent de façon très différente à un même type de poche, et, de plus, cela évolue au cours du temps. Il est donc nécessaire d'étalonner chaque type de poche pour chaque patient. Par similarité avec la tension, nous utilisons également une moyenne sur quinze jours pour apprécier le différentiel attendu par type de poche.

Cela pose néanmoins de gros problèmes, car certaines poches sont utilisées relativement rarement par certains patients. Les valeurs attendues dans ce cas sont donc assez incertaines. Néanmoins, cette approche permet de tenir compte de l'évolution du patient tout au long de son traitement. Enfin, un patient utilise plusieurs poches par jour. Il n'est pas rare d'observer un déficit sur une poche, compensé par un surplus sur l'une ou l'autre des poches suivantes. Cela reste tout à fait normal. Le différentiel cumulé de toutes les poches est connu des médecins sous le nom d'*ultrafiltration*.

Théoriquement, plus le patient est hydraté, plus l'ultrafiltration est importante. En effet, plus le sang contient d'eau, plus sa concentration baisse. Une même concentration de sérum aura donc tendance à drainer plus de liquide pour équilibrer les concentrations lors de l'osmose. Cependant, les différentes poches utilisées dans une journée peuvent être de type différent afin d'obtenir une dialyse plus fine. J'ai en effet déjà indiqué que les patients utilisaient majoritairement trois types de poches. Utiliser plusieurs types de poches dans une même journée permet donc d'obtenir une dialyse qui pourrait provenir de poches ayant des concentrations intermédiaires. Or cette variation de concentration au cours de la journée rend difficile l'appréciation de la quantité d'eau drainée dans la journée. En effet, le patient réagissant différemment à divers types de poches, il est difficile de savoir si un échange est normal, ou s'il est excédentaire.

Enfin, certains patients gardent leur péritoine rempli durant la nuit, alors que d'autres non ; les poches nocturnes restent plus longtemps que les autres en stase. Le débit observé est donc sensiblement différent, d'autant plus qu'un phénomène de réabsorption a tendance à se produire. En effet, en fin de dialyse, le sens du transfert osmotique tend à s'inverser. Cet effet a donc tendance à diminuer le différentiel de la poche nocturne. En définitive, l'importance des données issues de l'ultrafiltration est très discutée, et il est difficile de dire aujourd'hui si c'est un indicateur fiable de l'hydratation ou pas. Cependant, les néphrologues sont certains que c'est un indice important quant à la détection de troubles liés au péritoine. Par exemple, cela pourrait servir à détecter l'apparition de péritonites.

2.2.1.5 Synthèse

Le projet Diatelic a pour objectif d'aider les néphrologues à suivre l'évolution de plusieurs patients pendant leur traitement par DPCA à domicile. Le système d'aide reçoit donc les données médicales de chaque patient quotidiennement, et doit avertir les médecins de variations anormales de ces paramètres. Deux déviations sont plus particulièrement recherchées : les troubles de l'hydratation et la prise de poids du patient. La Table 1 résume tous ces paramètres.

Objectifs			
Détection des troubles de l'hydratation et la prise de poids chez des patients traités en DPCA à domicile			
Données disponibles			
Paramètre	**Référence**	**Avantages**	**Inconvénients**
Poids	Poids-sec	Directement lié à l'hydratation	Le poids-sec peut évoluer au cours du temps
Tension	Moyenne sur 15 jours	Lié au volume sanguin, et donc à l'hydratation	Pas de valeur fiable avant 15 j. Sensible au stress et aux émotions
Tension différentielle	Néant	Permet une bonne détection de troubles de déshydratation Pas de référence	Pas de lien connu avec les problèmes d'hyperhydratation
Ultrafiltration	Moyennes sur 15 jours	Lié à l'hydratation (?) Permet une détection des péritonites (?)	Moyenne sur 15 jours nécessaire pour chaque type de poche Utilité très discutée

Table 1 : Résumé des paramètres de Diatelic

2.2.2 Assistance à l'anesthésie

2.2.2.1 Cadre médical

L'anesthésie est un processus qui permet de mener une opération chirurgicale dans les meilleures conditions possibles pour le patient, mais aussi pour le chirurgien. Une anesthésie standard se compose de trois parties : la perte de conscience (ou *neuroplégie*), la suppression de la douleur (ou *analgésie*), et la perte de tonus musculaire.

La neuroplégie est le point le plus important pour le patient, car cette perte de conscience le protège des effets psychologiques de l'opération. Si l'on éliminait uniquement cette partie de l'anesthésie, l'opération pourrait se dérouler de façon correcte, mais il serait possible que le patient conserve des souvenirs subconscients de cette dernière. Des cas de ce type de traumatismes ont été observés lors de certaines anesthésies. Bien que l'opération ait semblé parfaitement réalisée, le patient en a conservé une terreur irraisonnée qu'il est incapable d'expliquer. Il refuse alors catégoriquement toute nouvelle opération, sans savoir exactement pourquoi. L'analyse a posteriori des données de l'opération ont montré une petite insuffisance au niveau de l'injection des neuroplégiques. On suppose donc que le traumatisme pourrait y être lié. La perte de conscience est généralement obtenue par injection d'un composé opiacé dérivé de la morphine.

L'analgésie permet d'inhiber les comportements défensifs que l'organisme met en œuvre lorsqu'il est agressé. Ainsi, en temps normal, des hormones de stimulation telles que l'adrénaline, par exemple, sont déversées dans le flux sanguin à chaque coup de scalpel. Par exemple, ces hormones peuvent entraîner des modifications du rythme cardiaque et du rythme respiratoire. Ces effets sont préjudiciables à la bonne conduite d'une opération, et doivent être évités. Pour cela, on recourt généralement à l'injection de morphiniques. Ces produits permettent l'inhibition des neurotransmetteurs au niveau du cerveau. Lorsqu'un patient est sous l'influence de ces substances, et bien que l'information de douleur arrive au cerveau de façon normale, aucun des messages nerveux figurant la douleur ne peut être interprété.

La perte de tonus musculaire permet au chirurgien d'opérer dans de bonnes conditions. Le relâchement des muscles permet en effet de supprimer les réflexes primaires de défense, et de conserver le patient dans un état de passivité. Sans cela, le corps se débattrait à chaque stimulation, même si la neuroplégie et l'analgésie sont correctes. Cela permet aussi de faciliter l'intubation en relâchant les muscles du cou, et d'éviter les mouvements des muscles abdominaux qui pourraient gêner le chirurgien. Cet effet est obtenu par l'injection de composés à base de curare. Ces derniers bloquent en effet le fonctionnement des synapses neuromusculaires en se fixant sur les sites récepteurs. De cette façon, les messages nerveux arrivent bien au niveau des muscles, mais ces derniers sont incapables de les interpréter.

Malheureusement, chacun de ces produits a des effets indésirables, aussi bien pendant l'opération que durant la période postopératoire. Par exemple, on sait que les morphiniques perturbent le fonctionnement du cœur et des poumons. On doit donc surveiller en permanence le rythme cardiaque, la pression artérielle, et les paramètres du respirateur. D'autre part, on a observé que plus la quantité de produits injectés augmentait, plus les effets postopératoires étaient importants. Il est donc impossible d'injecter des doses massives de ces produits de façon à garantir la perte de connaissance, de douleur ou de tonus musculaire. On doit donc chercher le juste-milieu, permettant de conduire l'opération dans de bonnes conditions, tout en limitant au maximum les doses injectées, de façon à réduire les troubles postopératoires.

Pour mener à bien une anesthésie, le médecin anesthésiste doit donc surveiller l'évolution d'une multitude de paramètres médicaux, et anticiper les agressions que va déclencher le chirurgien afin d'en amoindrir les effets. Pour cela, il peut jouer sur les doses de produits injectées. Il peut ainsi modifier le débit des pompes qui injectent une quantité constante de produit dans le flux sanguin, espacer les bolus (injection d'une grande quantité de produit en une seule fois), ou modifier la concentration des produits introduits par le respirateur.

Le projet d'assistance à l'an esthésie est mené en collaboration avec les anesthésistes du centre hospitalier universitaire de Brabois (le CHU). Son but est donc de fournir une aide à l'anesthésiste en regroupant les différentes données disponibles, en les recoupant, et en analysant leur évolution. Dans un premier temps, l'objectif sera d'acquérir les données, et de les présenter à l'anesthésiste sous une forme qu'il peut utiliser. Ensuite, il faudra analyser ces données, afin d'isoler des règles de comportement et des motifs d'évolution des différents signaux.

Finalement, on pourra envisager de piloter directement les injections de produits, de façon à garantir une profondeur d'anesthésie suffisante tout en limitant au maximum les doses injectées. Ce projet est encore à un stade assez peu avancé, ce qui limite en particulier la quantité et la diversité des données disponibles. Cela rend difficile la construction d'un modèle adapté, comme je le montrerai par la suite.

2.2.2.2 Les données disponibles

Pendant une opération au CHU, les données disponibles sont relativement peu nombreuses. Actuellement, trois instruments sont connectés sur un PC.

Les pousse–seringues permettent de contrôler l'injection de produits au patient. Le pilotage se fait grâce à des modèles pharmacocinétiques inspirés de ceux de Stan-Pump (63). Ces modèles prédisent l'évolution temporelle de la concentration d'un produit dans l a portion du cerveau où il fait effet, en fonction de la quantité de produit injectée par voie veineuse. L'inversion de ces modèles permet donc de calculer la quantité de produits à injecter pour obtenir une concentration donnée en un temps optimal. Cet outil permet alors à l'anesthésiste de se concentrer sur l'effet à obtenir plutôt que sur la façon d'y arriver. Cela contribue donc à l'obtention d'une anesthésie de qualité.

Néanmoins, les données concernant l'injection des produits ne sont disponibles qu'après l'opération, au prix d'une saisie et d'une synchronisation manuelles des données. En effet, l'anesthésiste remplit au cours de l'opération une fiche administrative sur laquelle il consigne les modifications des consignes des pousses-seringues, les bolus injectés, et les paramètres du respirateur. Cependant, les volumes réellement injectés ne sont pas rendus disponibles par l'appareil. Si l'on veut pouvoir les utiliser, il est donc nécessaire de les recalculer en utilisant une nouvelle fois les modèles pharmacocinétiques.

L'ASPECT 2000 est un instrument médical qui interprète les signaux fournis par un électroencéphalogramme (EEG) pour fournir un indice de perte de conscience, le BIS. Pour y parvenir, il utilise principalement une analyse bi-spectrale des signaux émis par le cerveau. La répartition fréquentielle obtenue permet alors de calculer un indice que les anesthésistes de Brabois considèrent comme étant assez fiable. De nombreux signaux sont fournis par cet instrument toutes les cinq secondes. Parmi ceux-ci, il semble que seuls le BIS, le RS (rapport de suppression) et l'EMG (électromyogramme) aient une utilité aux yeux des anesthésistes. La Figure 3 montre l'évolution de ces signaux en fonction de l'état du patient.

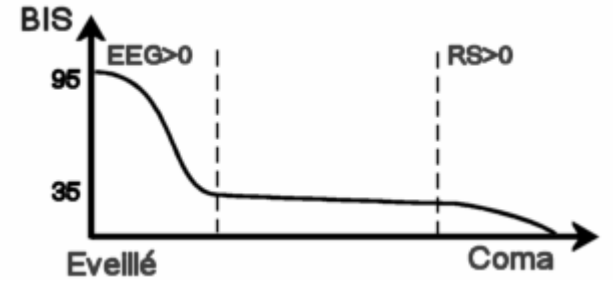


Figure 3 : Evolution du BIS en fonction de la profondeur de l' anesthésie

L'EMG est une mesure du courant généré par les muscles, et plus particulièrement par les mouvements oculaires. En effet, les électrodes de l'EEG étant placées sur le front du patient, les mouvements oculaires perturbent les mesures en générant de forts signaux électriques de basse fréquence. C'est donc un indice de réveil imminent.

Inversement, le RS est une indication que certaines portions de l'EEG sont plates. C'est donc un signal d'alarme qui indique un risque de coma, risque que l'on désire éviter tout autant que le réveil du patient. La Figure 3 rappelle toutes ces données en montrant ces divers signaux en fonction de l'endormissement du patient.

Comme je l'indique par ailleurs dans le chapitre sur l'apprentissage, je pense néanmoins que l'on peut tirer profit des autres données fournies pour affiner le diagnostic. Ces données incluent par exemple la SEF (Spectral Edge Frequency, fréquence en dessous de laquelle 90% des signaux se trouvent) et la puissance totale reçue. De plus, l'appareil fournit ces données pour trois canaux, alors qu'un seul est utilisé actuellement.

L'ANEMON est un instrument qui analyse le rythme cardiaque pour fournir un indice sur la douleur ressentie par le patient. Pour calculer cet index, il utilise une analyse fractale, dont les détails sont mal connus. Selon les anesthésistes de Brabois, les informations données ne sont fiables que lorsque la perte de conscience est totale. Dans le cas où le patient n'est pas totalement endormi, il semblerait que les systèmes sympathique et parasympathique, deux systèmes nerveux complémentaires, agissent simultanément sur cet indice, faussant sa valeur de façon imprévisible.

Par contre, lorsque le patient est profondément anesthésié, seules restent actives les fonctions les plus basiques. A ce moment, cet indice semble indiquer de façon relativement fiable le niveau de douleur qu'endure le patient, même si ce dernier n'en a pas conscience. Toutes les 5 secondes, cet instrument fournit trois valeurs intéressantes : l'index représentant la douleur du patient, et la fréquence de ses pulsations cardiaques, mesurée de façon instantanée et sur un intervalle de temps.

L'ASPECT et l'ANEMON ont des périodes de mesures similaires (5 secondes), mais ils ne sont en général pas synchronisés. Il est donc possible que les deux paquets de données arrivent simultanément, ou alors qu'il en arrive un toutes les 2,5 secondes. Bien entendu, tous les intermédiaires sont envisageables. Enfin, si l'on se place dans l'idée de fournir des indices à l'anesthésiste, il faut bien prendre garde à ne fournir que des données temporellement cohérentes. Il serait en effet particulièrement troublant pour l'anesthésiste d'avoir sous les yeux des informations récentes, directement affichées par les instruments médicaux, ainsi que des informations basées sur les données des minutes précédentes. Cela limite donc beaucoup le temps de calcul disponible pour fournir un résultat, et donc par conséquent, les algorithmes susceptibles d'être utilisés.

Un projet parallèle, en partenariat avec une société finlandaise, vise à acquérir plus de données en utilisant une centrale d'acquisition spécialisée. L'avantage principal de cette collaboration tient à la récupération des données hémostatiques (la pression artérielle), et des mesures fournies par le respirateur (concentration des différents gaz). Cependant, l'inconvénient majeur de cette approche est que ces données ne seront accessibles qu'après l'opération, car aucune sortie de données n'a été prévue pendant la phase d'acquisition . Ces données pourront donc servir à la recherche de motifs intéressants, mais pas à l'assistance à l'anesthésie durant l'opération. Bien entendu, il reste possible d'envisager une telle sortie pour une version ultérieure du matériel, voir même d'incorporer directement nos modèles dans la machine.

2.2.2.3 Synthèse

Le projet d'aide à l'anesthésie vise donc à fournir au médecin un diagnostic portant sur l'évolution de l'anesthésie. Cette analyse se fait selon deux axes, la profondeur du sommeil et l'intensité de la douleur.

A plus long terme, nous espérons parvenir à la boucle fermée, gérant l'injection des produits anesthésiants sans passer par l'anesthésiste. A ce moment, l'objectif sera de minimiser les doses de produits injectées, de façon à réduire les effets postopératoires. Cependant, le projet n'est pas suffisamment mature pour envisager un tel dispositif. De plus, l'aspect éthique de ce projet reste à étudier. La Table 2 résume les paramètres de ce projet.

Objectifs			
Analyser la profondeur de l'anesthésie et l'intensité des stimulations douloureuses du patient de façon à conseiller au mieux l'injection de produits anesthésiants			
Données disponibles			
Paramètre	**Référence**	**Avantages**	**Inconvénients**
BIS	A établir	Lié à l'activité du cerveau, et donc à la profondeur du sommeil	Très sensible aux perturbations électriques telles que le bistouri
RS	Néant	Directement lié au risque de coma	Apparaît relativement tard
EMG	Néant	Directement lié au risque de réveil	Perturbe le BIS Apparaît relativement tard
ANEMON	A établir	Lié au réactions du corps à la douleur, et donc à l'insuffisance d'analgésiques	Valide uniquement si le patient dort profondément
Pousse Seringue	A établir	Injecte les produits au patient	Données non encore disponibles

Table 2 : Paramètres de l' aide à l' anesthésie

2.2.3 Navigation d'un robot mobile

Ce projet se situe dans un contexte totalement différent de la médecine. L'intérêt de choisir un cadre totalement différent est double : tout d'abord, cela permet de généraliser la problématique afin d'éviter de trop spécialiser les méthodes recherchées au seul cadre de la médecine.

Ensuite, d'un point de vue pratique, travailler sur une application dont aucune vie ne dépend est sécurisant. Non seulement les données nécessaires peuvent être rassemblées au fur et à mesure de l'apparition des besoins, mais en plus, j'ai pu essayer des méthodes d'apprentissage et de planification sans arrière-pensée. En effet, les seuls risques encourus sont les dégâts matériels infligés au robot.

Dans ce projet, l'objectif est de piloter un robot mobile autonome à travers un environnement structuré, tel que des bureaux. Cet environnement peut être connu à l'avance ou non, selon le problème que l'on désire étudier. De plus, cet environnement peut être dynamique et non contrôlé.

2.2.3.1 Présentation générale

La navigation est une fonction qui permet à un agent mobile de se diriger dans un environnement vers un but défini. C'est donc une méthode utile à la résolution de nombreux autres problèmes. Par exemple, on pourrait imaginer une application de surveillance d'un bâtiment, ou une distribution automatisée de courrier. Dans ce dernier cas, il faudra à notre robot se diriger tout d'abord vers la boîte aux lettres, y prélever le courrier et se rendre dans les bureaux de chaque personne devant recevoir l'une de ces lettres.

Il est donc nécessaire de pouvoir préparer un chemin permettant de rejoindre une position précise à partir de n'importe quelle autre position. C'est le rôle du module de navigation d'un robot. La problématique peut être décomposée de façon à isoler des modules plus simples à traiter.

a) Localisation

Comme le robot est mobile, il peut occuper différentes positions à divers instants dans l'environnement. Il est alors nécessaire de recourir à une série de déplacements et d'observations, de façon à déterminer sa position avec un minimum d'incertitude. Ces actions destinées à accroître la précision de la localisation peuvent être passives ou actives : le robot peut mener une tâche tout en affinant sa connaissance de sa position. Alternativement, il peut aussi se consacrer à cette localisation et entreprendre les actions qui devraient lui amener un maximum d'informations, quitte à le ralentir dans sa progression vers le but.

b) Planification

Une fois que la position du robot est connue, ou tout du moins, supposée connue, il est nécessaire de planifier la suite d'actions à entreprendre afin de résoudre le but qui lui est fixé. Il peut s'agir de se rendre à un point précis, pour se recharger, par exemple, ou simplement de parcourir la zone régulièrement. La tâche d'un robot aspirateur pourrait être de passer partout, alors qu'un robot de surveillance peut se contenter d'un passager rapide dans chaque salle. Lorsque le plan d'action est connu, il est nécessaire de l'exécuter correctement. Cela peut être très facile si les instructions du plan sont simples et certaines, mais cela peut se compliquer très rapidement dans le cas contraire. Si la localisation du robot est incertaine, le plan peut en tenir compte de façon à assurer une efficacité et une sécurité optimales.

Ainsi, si le robot sait qu'il se trouve au milieu d'un couloir, mais qu'il ignore s'il est dos à un escalier ou si l'espace derrière lui est vide, choisir de reculer peut être dangereux. Il peut alors être intéressant pour lui de choisir d'autres actions, comme par exemple de se retourner pour vérifier.

c) Exécution du plan

Si les instructions sont d'un niveau d'abstraction supérieur à ce que sait exécuter le robot, il sera nécessaire d'utiliser une traduction de chacune de ces directives en une suite de commandes exécutables directement. Puisque le résultat d'une commande est toujours plus ou moins incertain, il peut être nécessaire de modifier le plan si ce denier n'a pas prévu d'échec ou d'erreur possible. En effet, de nombreux aléas, tels que des imprécisions de vitesse, de position, ou de temps d'exécution, sont susceptibles de perturber chaque commande.

Dans le cas où toutes les hypothèses sont prévues, et que le plan permet au robot d'atteindre son but quelles que soient les perturbations qu'il reçoit, on parlera d'une politique d'asservissement.

d) Adaptation

Finalement, si l'environnement est dynamique, il faut que le robot soit capable de s'adapter aux modifications de ce dernier. Par exemple, dans un environnement de bureaux, une porte peut s'ouvrir ou se fermer. Si le robot est incapable d'agir sur une porte fermée, il lui faudra choisir un autre chemin pour arriver au point voulu. De même, les conditions ambiantes peuvent biaiser plus ou moins les capteurs, faussant ainsi la localisation du robot. Par exemple, les conditions d'éclairage peuvent modifier les caractéristiques d'une image obtenue par une caméra, ou déplacer l'échelle de distances mesurée par un capteur infrarouge.

2.2.3.2 Simplet, notre robot

Afin de pouvoir tester et mettre en pratique les algorithmes que nous proposons, nous avons conçu et réalisé un robot mobile autonome. Les détails techniques de ce robot seront donnés en annexe. Je vais résumer rapidement les fonctionnalités disponibles de façon à permettre la mise en évidence de leurs effets sur les algorithmes testés. La Figure 4 présente une vue schématique de Simplet montrant l'agencement de ses capteurs et ses effecteurs.

Nous avons conçus ce robot dans l'optique d'en créer une population permettant de tester des algorithmes de coopération multi-agents. Cela implique donc une réduction maximale des coûts. Nous avons alors choisi de déporter la puissance de calcul sur des stations de travail et de limiter celle qui est embarquée sur le robot lui-même. Le robot se contentera donc d'exécuter les ordres au fur et à mesure que le logiciel de navigation les lui enverra. L'aspect correspondant aux communications est donc primordial. Afin de permettre une gestion transparente du robot, nous avons conçu le système sous la forme d'un serveur relié à un ou plusieurs robots. Des clients peuvent se connecter à ce serveur à travers une interface réseau standard. De cette façon, l'implantation du client peut être différente de celle du serveur.

L'inconvénient de cette architecture repose sur la latence qu'elle engendre. Lorsque le robot envoie une lecture de ses capteurs à son client, le message passe tout d'abord par le serveur. Ce dernier le redirige alors vers le client associé à ce robot. Ce client, à son tour, va interpréter ces données, et fournir une action que le robot doit exécuter pour atteindre son but. Cet ordre va parcourir le chemin inverse, passant par le serveur avant d'atteindre le robot pour y être exécuté. Chacune de ces transmissions est rapide, mais cette suite de délais peut être gênante dans les cas où une réponse rapide est nécessaire.

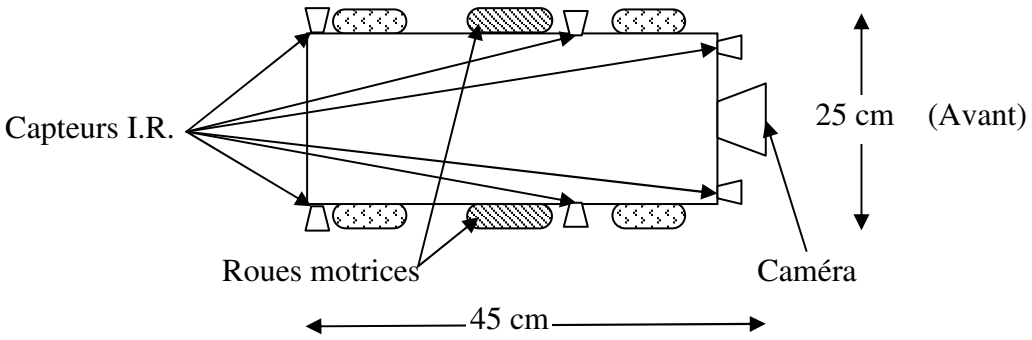


Figure 4 : Représentation schématique de Simplet

La propulsion de Simplet est assurée par deux moteurs pas à pas situés de part et d'autre du robot. Cela permet d'obtenir un mouvement simple et précis au détriment de la vitesse. La palette de mouvements disponibles inclut donc le déplacement linéaire vers l'avant et l'arrière, la rotation sur place, et le mouvement en arc de cercle. Les moteurs sont commandés en vitesse, mais il est possible de demander un mouvement d'un certain nombre de pas, et donc sur une distance précise.

Les capteurs du robot incluent une caméra couleur et une série de 6 capteurs infrarouges. Ces derniers sont placés tout autour du robot, de façon à donner une image de l'environnement immédiat de ce dernier. La caméra permet d'obtenir une information riche et détaillée, mais elle est limitée à une position fixe, à l'avant du robot. Ironiquement, la caméra peut être un capteur *trop* riche pour être utilisable en temps réel sans circuits dédiés à son traitement.

Les capteurs infrarouges donnent quant à eux une indication sur la proximité des obstacles situés entre 10 et 80 centimètres. Nos expériences montrent que leur mesure est assez fiable sur des distances allant de 10 à 60 centimètres, sous réserve de ne pas être en plein soleil. Au-delà, la réponse des capteurs est soumise à un bruit beaucoup plus important. Cette disposition des capteurs donne donc une vue de l'environnement local du robot.

Néanmoins, l'espace environnant le robot n'est pas entièrement couvert, et de gros angles morts sont présents de chaque côté du robot. Cela n'est pas véritablement gênant, puisque le robot ne peut pas se déplacer latéralement. Il faudra cependant en tenir compte lors des rotations. De même, aucune mesure n'est disponible à l'arrière du robot. Il est donc préférable d'éviter de reculer, à moins que l'espace ne soit connu par une exploration précédente.

2.3 Une problématique commune

Bien que ces trois applications semblent très différentes, elles ne sont en fait que trois instances d'une même classe de problèmes. On peut en effet ramener très facilement nos trois applications à une problématique de localisation d'un *agent situé*. Une fois que cette identification est réalisée, l'application des mêmes algorithmes devient naturelle. Cependant, cette diversité permet de mettre en évidence des problèmes plus ou moins spécifiques à chaque application. En retour, cela permet de généraliser la problématique et de renforcer les solutions que nous proposons.

Je vais donc montrer, en particulier, comment la robotique fournit des algorithmes capables de surveiller une dialyse ou une anesthésie. Je présenterai ensuite, dans les prochains chapitres, comment les aménagements nécessaires à ces deux projets peuvent profiter à la simplification des modèles utilisés en robotique.

Tout d'abord, avant de pouvoir appliquer des algorithmes à un problème, il faut s'assurer que ce problème entre bien dans le cadre d'application de ces méthodes. L'application robotique se traduit sans peine en termes d'agent situé : le robot est notre agent, et son environnement est le monde physique qui l'entoure. Il perçoit cet environnement grâce à ses senseurs, c'est-à-dire ses capteurs de distance infrarouge et sa caméra. Pour les applications médicales, la mise en correspondance est plus difficile. Pour y parvenir, nous avons considéré que la condition médicale d'un patient était un agent mobile qui évolue dans un environnement constitué par ses différentes conditions physiologiques.

Les conditions modélisées dans cet environnement virtuel sont celles qui contribuent à l'apparition de pathologies, ou au contraire, au maintien du patient dans un état de bonne santé. De cette façon, le diagnostic médical se traduit naturellement par la localisation de cet agent virtuel dans son environnement, tout aussi imaginaire. En effet, en déterminant la position de notre agent dans son environnement, nous sommes capables d'isoler une cause ou plusieurs causes permettant probablement d'expliquer les symptômes observés. De par la construction de cet environnement, ces causes rejoignent le diagnostic du médecin.

A partir de cette représentation initiale, on peut commencer à dégager des observations qui justifient le bien-fondé de ce modèle, et permettent d'étoffer progressivement ce dernier. Tout d'abord, la position de l'agent situé représentant un patient tient compte aussi bien des subtilités de la condition instantanée du patient que de la cohérence temporelle du diagnostic. En effet, de la même façon qu'un robot est incapable de se déplacer instantanément d'une pièce à une autre, il est impossible à un patient d'évoluer d'une pathologie à une autre sans passer par tous les stades intermédiaires.

Ensuite, la perception du robot est influencée par sa position dans l'environnement. De même, les observations que l'on peut obtenir à partir des données médicales sont influencées par l'état physiologique du patient, et la présence ou non de pathologies. Les signaux médicaux pris à un instant donné ne permettent pas de donner avec certitude l'état global d'un patient. De la même façon, la lecture ponctuelle des capteurs du robot ne permet pas de donner sa position exacte sur une carte de son environnement.

C'est la raison pour laquelle on dit que l'environnement est partiellement observable. Chaque agent n'a qu'une vision locale et limitée de son environnement. L'observation va donc jouer un grand rôle dans la localisation du robot, tout comme elle concourt au bon diagnostic d'une pathologie médicale. Dans les deux cas, l'observation ne suffit pas, pour obtenir une localisation cohérente, il est nécessaire de tenir compte de la dynamique du système, et donc de la suite des observations perçues jusqu'à l'instant du diagnostic.

Enfin, l'environnement est très dynamique, autant dans le cas du pilotage d'un robot que dans le cas de la gestion d'une anesthésie ou d'une dialyse. Pour le robot, j'ai déjà indiqué qu'il pouvait être influencé par des variations naturelles, telles que l'éclairage qui fausse la lecture des capteurs. Cependant, l'environnement de chacune des applications présente également des variations dont la source est complètement artificielle. Ces variations perturbent le système, et échappent à tout contrôle. Etant externes au système, ces variations sont difficilement prévisibles. Ainsi, le poids d'un patient dialysé peut évoluer naturellement au cours du temps, ce qui fausse l'estimation de la quantité d'eau que contient son organisme. Cette variation peut par exemple dépendre des différents repas que consomme le patient. Au niveau de l'anesthésie, un coup de bistouri donné par le chirurgien va avoir tendance à réveiller le patient par le jeu des réflexes de défense innés de l'organisme. Parallèlement, le déplacement d'un objet, ou l'ouverture d'une porte par un être humain peut modifier les positions relatives du robot et de son but, en l'obligeant par exemple à effectuer un détour.

La notion de plan d'action est plus difficile à rapprocher. Alors que le but d'un robot est clairement spécifié par son concepteur, le but d'une dialyse est rempli de non-dits. Il est difficile d'exprimer ce qu'est une bonne dialyse, ou une bonne anesthésie. Globalement, il s'agit de maintenir le patient dans un état de bonne santé. On pourrait alors comparer cela au fait de faire traverser un couloir à un robot. Il lui faut se maintenir au centre du couloir et éviter les murs et les passants. Au niveau de l'anesthésie, cela correspond au maintien de la profondeur de l'anesthésie pendant toute la durée de l'opération.

L'expression de l'environnement en tant qu'espace de pathologies permet une définition simple du but : il suffit de rester éloigné des différentes pathologies, même si les perturbations du système visent à nous en rapprocher. Accessoirement, il peut être nécessaire de planifier une suite d'actions thérapeutiques pour revenir à un état normal dans le cas où une pathologie n'a pu être évitée. En dialyse, il peut s'agir de résorber une hyperhydratation détectée tardivement alors que, pendant une anesthésie, on peut vouloir empêcher un patient de sombrer dans le coma. En robotique, cette situation peut se produire de façon à permettre au robot de sortir d'une impasse.

Dans ces trois cas, la situation est asservie aux capteurs : lorsque l'on détecte une déviation par rapport à l'état idéal, on choisit une ou plusieurs actions permettant de s'y ramener. Pour résorber une hyperhydratation, on augmentera la concentration du dialysat ; pour réduire les risques de coma, on diminuera les doses d'anesthésiants, et pour sortir d'une impasse, on utilisera une marche arrière ou une série de manœuvre permettant de faire demi-tour. Pourtant ces trois actions traduisent la même volonté de ramener le système dans un état correct.

Pourtant, comme je l'ai déjà signalé, chacune de ces applications a des spécificités propres. Ainsi, la surveillance d'une dialyse ou d'une anesthésie peut profiter de l'expérience et de la connaissance du médecin. La navigation d'un robot mobile ne peut quant à elle compter sur personne pour corriger une éventuelle imprécision. A l'inverse, alors que les règles d'évolution d'un robot sont claires et bien connues, l'évolution de l'hydratation d'un être humain est bien plus hasardeuse à prédire. De même, au cours d'une anesthésie, l'impact de l'augmentation des doses d'analgésique sur un patient commence seulement à être compris et par les médecins. On peut donc disposer d'ébauches de règles d'évolution, mais celles-ci sont généralement trop informelles pour être utilisées directement.

Au niveau des échelles de temps, les trois systèmes sont très différents. Pour la dialyse, on travaille au niveau de la journée. On peut donc consacrer énormément de temps à traiter les données d'un même patient. Alternativement, on peut suivre plusieurs patients simultanément, sans générer d'interaction entre leurs données. Au niveau de l'anesthésie, les données arrivent de toute façon toutes les cinq secondes pour chaque instrument. En parallèle, le patient évolue au gré des doses de produits injectés et des actes chirurgicaux mis en œuvre. La fenêtre temporelle est donc beaucoup plus serrée. Enfin, pour la robotique, le problème se pose encore différemment : plus le robot va vite, plus il faut traiter les données rapidement. Si un calcul a besoin de beaucoup de temps pour s'effectuer, il suffit de ralentir le robot, voire même de l'arrêter complètement pour disposer du temps nécessaire.

Ces diverses remarques, résumées dans la Table 3 et la Table 4 permettent donc de mettre en évidence la possibilité de rapprochement de ces problématiques apparemment très différentes. Parallèlement, ces tables permettent aussi de mettre en relief les différences qui séparent des problèmes qui pourraient sembler équivalents en de nombreux points. Cela nous permet ainsi de confirmer que les algorithmes présentés dans les chapitres suivants peuvent s'appliquer. Cela montre également que ces méthodes se doivent d'être suffisamment robustes et génériques pour permettre une interprétation variée de leurs différentes composantes.

Il en va de même pour les spécificités de chaque application. La prise en compte des spécificités de l'une d'entre elles perme t la généralisation des autres, et l'application à des problèmes plus diversifiés. Ainsi, la gestion d'une anesthésie peut nous donner de bonnes bases qui pourront nous permettre de traiter une Dialyse Péritonéale Automatisée dans le futur, par exemple. De la même façon, une meilleure localisation dans le cadre de la robotique pourra se traduire par un meilleur respect de la dynamique d'une anesthésie.

Application	Perception	Action	But, Planification	Environnement, Localisation
Navigation Robotique	Infrarouges Sonars Caméra	Avancer Reculer Tourner	Atteindre une position donnée	Position dans un bâtiment
Surveillance de Dialyse	Poids Tension artérielle Ultrafiltration	Poches de dialyse Anti-hypertenseurs Régimes	Maintenir une hydratation correcte	Niveau d'hydratation du patient
Aide à Anesthésie	EEG ECG EMG	Changer les doses d'analgésique d'hypnotique de curarisant	Maintenir un niveau de sommeil optimal	Profondeur du sommeil de l'analgésie

Table 3 : Résumé des caractéristiques des trois applications

Application	Echelle Temporelle	Difficultés Majeures	Perturbations
Navigation Robotique	Dynamique	Environnement de grande taille	Météorologie Humains présents Aléas matériels
Surveillance de Dialyse	Quotidien	Modèle variant au cours du temps	Perturbations imprévisibles régulières
Aide à Anesthésie	5 secondes	Echelle de temps instable selon les opérations	Données perturbées (bistouri électrique)

Table 4 : Différences principales des trois applications

3 Travailler avec le monde réel

Durant la présentation du cadre applicatif de ma thèse, j'ai souvent fait référence aux notions d'observation, de capteurs et de signaux. Ces outils permettent en effet de tenir compte de cette triste réalité : le monde cache les causes qui permettent d'expliquer son comportement. Ce chapitre a donc pour but d'expliciter ces différentes notions, car elles conditionnent toute la suite du travail que j'ai effectué.

Je vais présenter tout d'abord la notion d'état caché. En effet, toute la difficulté des travaux de diagnostic repose sur la découverte des raisons qui expliquent ce que le système perçoit. Cette notion est intrinsèquement liée à la notion de capteur et de perception locale, car c'est à travers ces capteurs que le système reçoit des indices lui permettant de localiser l'agent situé au sein de son environnement.

Pour pallier au manque de cet état caché, on utilise des modèles. Ces modèles substituent à cet inaccessible état caché une version qui est supposée avoir les mêmes effets. Tout le problème est alors de construire un modèle suffisamment correct pour qu'il soit utilisable. Généralement, ce modèle est imparfait. On peut alors espérer que, dans la majorité des cas, ce modèle corresponde à la réalité. Les autres cas seront attribués à des perturbations plus ou moins aléatoires. Ainsi, le système semble avoir une évolution incertaine, ce qui nous amène au deuxième point important : non seulement le monde cache ses raisons, mais en plus il n'est pas entièrement prévisible.

Ces différentes considérations m'amèneront donc à définir un modèle perceptif qui permettra de conduire un diagnostic correct. Etant donné que nombre des signaux issus de nos capteurs sont continus (le poids, la tension, le BIS, la distance à un mur,…), il faut que notre modèle de perception puisse prendre en compte des valeurs continues. Or la majeure partie des algorithmes utilisés en intelligence artificielle ont des bases discrètes. Je montrerai donc comment l'utilisation d'intervalles flous permet de disp oser des avantages des représentations continues, tout en affichant la simplicité des intervalles discrets.

Ce modèle de perception servira ensuite de base afin de construire, dans les chapitres suivants, des modèles de diagnostic, d'interaction avec les m édecins, et des modules d'apprentissage adaptés à ces formalismes.

3.1 Notion d'état caché, de diagnostic

Je suis intimement persuadé que le monde qui nous entoure est foncièrement déterministe. Il n'y a pas de hasard ; tout peut être expliqué. Le problème vient du fait que les causes ne sont pas toujours apparentes. En fait, elles le sont rarement. Ce qui peut expliquer ce que l'on observe est toujours caché, de manière plus ou moins partielle. C'est alors le rôle de la science que d'expliquer ce qui nous para ît naturel. Les scientifiques cherchent toujours à découvrir des raisons logiques qui permettent d'expliquer pourquoi le monde est tel qu'il semble être. D'un point de vue moins philosophique, c'est aussi la base de la notion de *diagnostic*. On observe un ou plusieurs faits, concordants ou non ; tout l'art du diagnostic consiste alors à trouver la cause, ou les causes, qui permettent d'expliquer ces faits.

Lorsque le médecin analyse la tension de son patient, observe son évolution et la compare à la courbe de son poids pour en déduire qu'il souffre d'un début de déshydratation, tout le monde s'accorde à dire qu'il s'agit d'un diagnostic. Lorsque le garagiste fait des tests pour découvrir pourquoi le moteur de votre voiture refuse de démarrer, compare les courbes de réponse du moteur à ses abaques, est-ce encore du diagnostic ?

Le dictionnaire nous donne la définition suivante du mot diagnostic : *Evaluation d'une situation donnée, jugement porté sur telle conjecture, tel ensemble de circonstances*. On peut donc facilement étendre cette notion en dehors du monde médical d'où elle est issue. La position dans l'espace d'un robot permet d'expliquer les valeurs observées par ses capteurs. L'inversion du processus d'observation permet d'évaluer la position du robot en partant des lectures de ses capteurs. C'est donc une forme de diagnostic, car elle évalue la situation, résumée par la valeur de chaque capteur, et porte un jugement sur la position du robot dans son environnement.

Cette notion de diagnostic me semble importante, car je pense que c'est la base nécessaire à toute coopération avec des êtres humains. Si, au cours d'une anesthésie, le logiciel propose de doubler le débit de la pompe à morphine, le médecin anesthésiste est en droit de demander pourquoi. Le logiciel doit alors être capable d'exprimer la raison de son choix. Ce choix s'exprime souvent par la reconnaissance d'une situation, c'est-à-dire un diagnostic. Même placé hors du contexte de la collaboration avec un expert, je suis persuadé que toute prise de décision devrait être basée sur un diagnostic solide.

Le problème vient du fait que, pour faire un bon diagnostic, il faut comprendre la façon dont les observations découlent de l'état du problème. C'est là que tous nos ennuis prennent leur source : l'état réel du système est caché, et les observations qui en découlent sont souvent trop ambiguës pour l'identifier. Sans connaître cet état, le processus d'observation reste une boîte noire. Pour comprendre le fonctionnement de cette boîte, on recourt à la modélisation du problème. Cette modélisation consiste à proposer une série de règles régissant l'évolution et le comportement du système. Ce *modèle* n'est rien d'autre qu'une proposition d'état caché. Le processus d'observation s'écrit alors comme une fonction qui, sur la base de cet état, génère les valeurs que l'on observe sur le problème réel.

Ce processus de modélisation introduit donc une première source d'incertitude, car rien ne prouve que le modèle choisi soit conforme à la réalité. On observe simplement qu'il se comporte d'une manière similaire à l'original. Souvent, la modélisation est imparfaite et suffit tout juste à ressembler au système original. Même si ce modèle est suffisant pour exprimer ce que l'on attend de lui, il introduit souvent un bruit qui intègre l'inadéquation entre le système et son modèle. Ainsi, par exemple, le fait de ne pas tenir compte du niveau de stress d'un patient introduit une incertitude quant à l'évolution de sa tension.

3.2 Un modèle incertain de l'univers

Comme je l'ai déjà précisé ci-dessus, je suis persuadé que l'univers est parfaitement déterministe. Cependant, le nombre de paramètres qui régissent l'évolution d'un processus est tel que l'on peut rarement les prendre tous en compte. Par exemple, lorsque l'on lance un dé sur une table, le mouvement de ce dernier est régi par les lois de la mécanique. Nous devrions donc être capables de prédire précisément le résultat de chaque jet. Pourtant, ce n'est pas le cas. En effet, la trajectoire du dé dépend de paramètres tels que la force du lancer, la direction, le mouvement propre du dé, l'état d'usure de ses arêtes, et ainsi de suite. Les interactions entre la table et le dé sont tellement complexes qu'une variation infime des conditions du lancer peut avoir des conséquences incroyables. Un système de ce type est dit *chaotique*.

Dans les applications qui nous intéressent, la médecine et la robotique, les perturbations sont soit issues d'êtres humains, soit portant sur des êtres humains. En effet, dans le dernier cas de la médecine, le corps humain est une machine extrêmement compliquée, dont on ne perçoit que très peu de ses paramètres. Parmi ceux-ci, nous commençons enfin à comprendre l'influence de certains ; d'autres sont connus, mais leur influence est inconnue, ou simplement supposée. Enfin, le système étudié reçoit en permanence des perturbations qui échappent à tout contrôle. Tout au plus, on peut essayer d'en limiter les effets. Ainsi, dans le cas de la surveillance de malades dialysés, l'hydratation d'un patient dépend directement de son traitement. Cependant, elle va dépendre également de ce qu'il mange, de ce qu'il boit, des efforts qu'il fournit, de la température ambiante, et ainsi de suite. Ces éléments perturbent le système de manière assez imprévisible, mais devraient être pris en compte de façon à obtenir un diagnostic correct.

Pour ce qui concerne l'assistance à l'anesthésie, les perturbations extérieures au patient sont mieux contrôlées puisque ce dernier est endormi, et sanglé sur la table d'opération. Seules les manipulations ayant trait à la chirurgie sont donc susceptibles de modifier l'état du patient. On peut donc imaginer que ces facteurs puissent être intégrés au système de façon à amoindrir leurs effets. Néanmoins, la prise en compte de ces perturbations serait tellement lourde que cela rendrait le système inutilisable par l'équipe médicale. En effet, imaginons par exemple qu'il soit nécessaire d'ajouter un événement à chaque coup de bistouri du chirurgien pour éviter que le système ne déclenche une alerte lors de la stimulation du patient. Cela rendrait la vie impossible aux anesthésistes. De plus, il resterait toujours les incertitudes liées à notre modélisation du patient. Cette dernière est une abstraction des règles de comportement observées lors des anesthésies. Il est donc fort probable que de nombreux cas de figures peuvent se présenter et que fort peu ne soient gérés correctement par le modèle. On a déjà vu, par exemple, des cas de patients auxquels il fallait injecter trois fois la dose habituelle d'agents hypnotiques pour maintenir un état de sommeil suffisant. Ces patients dégradaient en effet les produits beaucoup plus vite que la normale. La durée correspondant à une dose donnée en était donc réduite d'autant.

Pour ce qui concerne la robotique, les perturbations auxquelles on peut s'attendre sont principalement dues à la présence d'humains sur le lieu de l'expérimentation. Cela inclut bien entendu les portes qui s'ouvrent et qui se ferment, mais aussi le passage d'êtres humains dans les couloirs et le déplacement d'objets dans l'environnement . Le fait que ces modifications soient dues à des êtres humains évoluant dans l'environnement n'aide en rien le robot, puisqu'il ne peut agir dessus. Toutes ces modifications, du point de vue du robot, restent donc spontanées et aléatoires, à moins de modéliser formellement la présence d'autres agents indépendants dans l'environnement. Ces perturbations sont donc à rapprocher des influences externes des applications médicales, telles que les repas des dialysés ou les stimulations infligées par le chirurgien.

A part ces perturbations artificielles, que l'on peut limiter en faisant appel à la bonne volonté des gens, d'autres incertitudes sont plus difficiles à éviter. Dans cette catégorie, on peut citer le dérapage des roues sur le sol, l'inégalité rencontrée au passage des seuils de portes, la variation de la luminosité qui dépend de la météorologie, les délais de transmission des commandes sur le réseau, et ainsi de suite. La prise en compte de toutes ces incertitudes est donc un passage obligatoire si l'on veut obtenir une méthode robuste capable de s'adapter à de nombreux cas. Cette influence bruitée s'exerce à tous les niveaux : on la trouve aussi bien sur la condition initiale du modèle, que sur son évolution, ou sur la perception de l'état à chaque instant.

3.3 Un monde continu

L'une des grandes questions qui reste sans réponse tient à la notion de continuité. Le monde est-il continu, ou discret ? Intuitivement, je dirais que le monde est continu. Lorsque l'on mesure une distance, on utilise une unité discrète, comme le mètre. Pourtant, entre chaque mètre se situent 1000 millimètres. Entre chaque millimètre se trouvent 1000 micromètres. D'où la question : jusqu'où peut-on aller ? On peut descendre à la taille d'un atome, car nous savons qu'un objet est composé d'un nombre fini d'atomes. Cependant, chaque atome est constitué de particules en mouvement. Ces mouvements sont tellement fins que l'on n'est même plus très sûr de savoir s'ils sont continus ou discrets. De même, la notion de temps est ambiguë. Le temps coule de manière continue, mais nous passons notre temps à le discrétiser. Une heure, une seconde, une année. Le temps est-il continu, ou existe-t-il des quanta de temps ?

D'un autre côté, l'informatique est une science discrète. Nos machines ont une taille mémoire finie, décomptée au bit près. De même, le temps machine se réduit à un certain nombre de cycles d'horloge. Tout modèle informatique ne peut donc qu'être discret. On peut cependant biaiser le système en ne tenant pas compte du fait qu'un élément est discret. C'est pourquoi l'intelligence artificielle ne s'est longtemps intéressée qu'à des modèles symboliques ou discrétisés. Le problème de ces modèles vient du fait qu'il s'agit d'une abstraction des systèmes réels. Un modèle symbolique est une abstraction mathématique, généralement très difficile à construire, mais donnant des résultats souvent parfaits ; peut-être même trop parfaits. Un modèle discrétisé est une abstraction numérique qui se base sur l'idée qu'être proche d'un point, c'est presque comme être en ce point. A ce moment, on ne représente que quelques points importants, que l'on appelle un modèle du réel.

Car le monde réel est continu. Et même s'il n'est pas vraiment continu, cela correspond à un nombre d'éléments tellement grand qu'il devient impossible à traiter. Imaginez-vous mesurer les distances en nombre d'atomes… Dans ces conditions, plus on désire un modèle proche de la réalité, plus le nombre d'éléments de ce modèle doit être grand.

C'est le cas par exemple de la discrétisation par intervalles : lorsque l'on veut approcher une fonction continue, par un tableau de valeurs, on l'échantillonne régulièrement. Puisque la fonction est continue, sa valeur ne change pas beaucoup tant que l'on ne s'éloigne pas trop de l'un des échantillons choisis. Bien entendu, plus le nombre de ces valeurs de référence est grand, moins un point donné pourra être loin de l'une d'entre elles, et plus la valeur de la fonction approchée sera proche de la réalité. Augmenter le nombre de ces valeurs permet donc d'affiner le modèle. La Figure 5 illustre cette idée en montrant une fonction et deux de ses discrétisations par intervalle. La courbe de gauche est la fonction continue que l'on cherche à modéliser. A droite, deux discrétisations utilisant des pas différents sont présentées.

On voit donc que la version du bas est beaucoup plus fidèle à la fonction originale. Cependant, elle utilise deux fois plus de données que celle du haut. Ce phénomène reste d'autant plus valable que l'on utilise des dimensions élevées. Pour doubler la précision d'une courbe à deux variables, il faut affiner chacun de ses axes, c'est-à-dire multiplier par quatre le nombre de valeurs nécessaires ! Cela devient particulièrement gênant lorsque l'on utilise cette représentation au sein d'un modèle statistique : il faut alors exprimer la probabilité d'apparition de chaque valeur de chaque signal pour chacune des situations envisagées. Si l'on désire tenir compte de toutes les combinaisons possibles de plusieurs signaux, il faut recourir au produit cartésien des intervalles de chacun de ces signaux. Il est donc nécessaire de restreindre au maximum le nombre d'éléments décrivant chaque signal afin de limiter un maximum le nombre de paramètres du modèle.

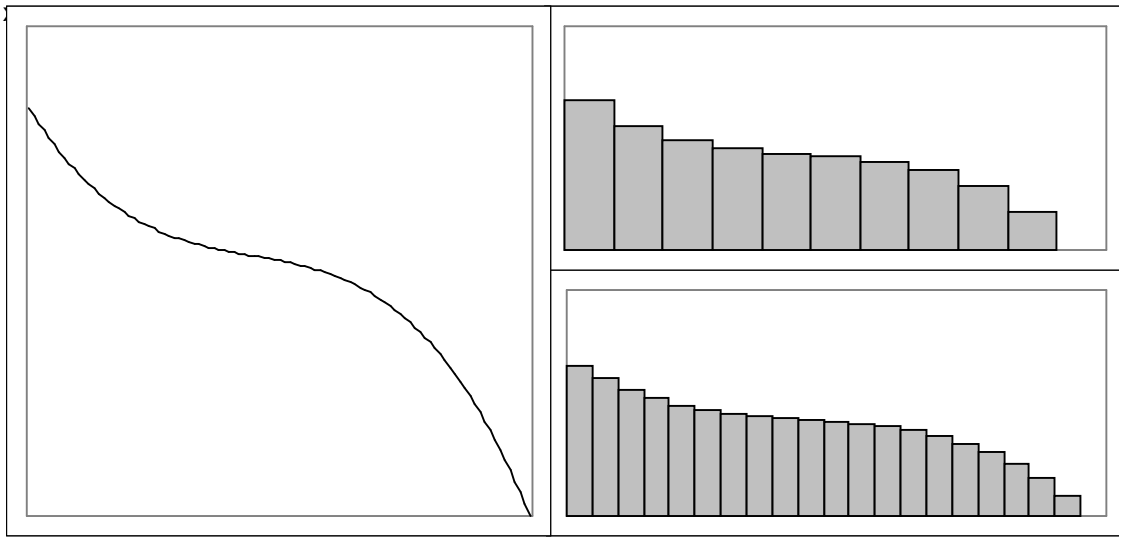


Figure 5 : Relation entre précision et nombre de données

Les mêmes considérations s'appliquent à la discrétisation d'un état continu. Lorsque l'on dit que le patient est hyperhydraté, on discrétise sa valeur d'hydratation. Si l'on désire un diagnostic plus fin, il est nécessaire d'augmenter le nombre de ces valeurs : légèrement hyperhydraté, hyperhydraté, vraiment hyperhydraté, et ainsi de suite. A chacune de ses valeurs doit être associé un ensemble de règles permettant de la diagnostiquer, de la séparer de ses voisines. La complexité du modèle augmente alors très rapidement au fur et à mesure que l'on demande un modèle plus précis.

Le problème devient encore plus complexe lorsque l'on considère le problème des transitions entre ces valeurs. Par exemple, dans le cadre de la robotique mobile, le mouvement d'un robot s'exprime comme un système d'équations déterministes. Cependant, il est intéressant de discrétiser son environnement, de façon à permettre l'expression des valeurs des capteurs du robot. A ce moment, le mouvement déterministe devient stochastique lorsqu'il est discrétisé.

La Figure 6 montre un exemple simple d'un tel cas : partant de l'état *A*, dirigé vers la droite, avancer d'un mètre (l'une des flèches) devrait nous conduire à l'état *B*. Cependant, une position sur la gauche de l'état peut faire en sorte que le mouvement nous conduise en *A*, et non en *B*. La même action appliquée une deuxième fois nous conduira alors directement en *C*, et non en *B*. Si l'on ajoute une incertitude sur l'orientation, même légère, on va pouvoir passer de *A* à *D*, ou à *E* directement, toujours en avançant d'un mètre. On voit alors qu'une action certaine peut être modélisée par un mouvement probabiliste lorsque l'on discrétise l'environnement sous-jacent.

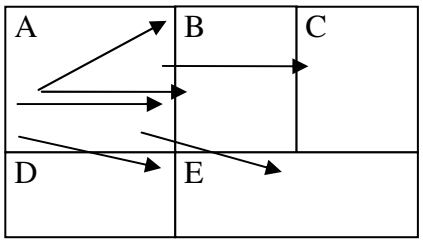


Figure 6 : Mise en évidence du bruit de discrétisation

3.4 Modèle de perception

Nous avons vu dans les sections précédentes que l'observation est l'un des pôles primordiaux du diagnostic. Nous avons vu également que la plus grande partie des signaux que l'on peut utiliser sont continus. Il nous a donc fallu définir un modèle de perception capable de traiter des signaux continus de manière robuste.

Je vais donc présenter dans cette section le modèle de perception que nous avons conçu pour nos applications. Je montrerai en particulier que ce modèle, de par sa nature symbolique riche en sémantique, permet une interaction aisée avec des experts humains tels que des médecins. En outre, le faible nombre de paramètres de ce modèle permet son utilisation directe dans des applications concrètes.

3.4.1 Principe général

Comme la majorité des algorithmes usuels utilisent des perceptions discrètes, nous avons choisi de discrétiser les valeurs des capteurs dont la valeur est naturellement continue. Dans l'optique du diagnostic et de la collaboration avec des experts humains, nous avons choisi d'utiliser 3 valeurs symboliques. Cela permet donc de garder un nombre de paramètres acceptables, évitant ainsi l'explosion combinatoire liée aux discrétisations par intervalle.

Cependant, afin de conserver un maximum d'expressivité, et puisque Steimann montre dans (64) et (65) que les intervalles flous se prêtent bien à la modélisation de pathologies médicales, ces trois valeurs seront définies par des intervalles de la logique floue (71). L'appartenance à l'une de ces valeurs sera donc incertaine, ce qui permet de retrouver une partie de l'information perdue lors de la discrétisation. Ces trois valeurs sont définies par étiquettes symboliques, comme *normale, trop petite*, et *trop grande*. Cette partition est bien adaptée aux problèmes de diagnostic où le système est censé rester dans un état dit *normal*. Cela facilite par là même la coopération avec des humains, car ces notions sont beaucoup plus intuitives qu'un découpage artificiel des données en intervalles, ou qu'une décomposition en terme de fonctions mathématiques telles que des distributions normales.

De plus, l'utilisation de la logique floue permet en même temps de rendre ce système robuste. En effet, une petite variation de la valeur observée n'entraînera jamais l'observation d'une valeur symbolique différente, mais seulement le décalage des probabilités d'appartenance à ces trois valeurs. C'est ainsi que, pour le projet DIATELIC, le poids est projeté sur une échelle centrée sur le poids-sec fixé par le médecin. Par exemple, si le poids-sec est défini comme étant égal à 60±1,5 kg pour un patient donné, un poids réel observé de 58,6 kg serait normal avec des intervalles classiques. Avec 200 grammes de moins, ce même poids serait trop bas. Avec les intervalles flous, 58,6 kg seraient caractérisés comme normal (3%) et comme trop bas (97%). Si l'on retire les mêmes 200 grammes, on passe à 100% trop bas. On voit donc que cette variation entraîne une modification légère de la valeur observée, ce qui évite les effets de seuil et les conséquences désastreuses qui peuvent y être associées.

Un autre avantage de cette méthode tient à ce qu'elle permet de tenir compte aussi bien des valeurs continues que des valeurs discrètes, puisqu'en définitive, toutes sont considérées comme discrètes. De plus, certaines de ces valeurs peuvent être incertaines ou manquantes. Pour prendre en compte ces paramètres, il suffit alors de faire tendre la projection sur l'ensemble flou vers une distribution uniforme. Cela reflète en effet l'incertitude que nous avons sur la valeur du capteur. A l'extrême limite, un capteur ne donnant pas de valeur du tout donnera une projection uniforme, ce qui dénotera une totale incertitude sur l'appartenance de *pas de valeur* à chacun des intervalles flous.

Par exemple, dans le projet d'assistance à l'anesthésie, nous avons rencontré le problème suivant : lorsque le chirurgien utilise un bistouri électrique, les parasites électromagnétiques générés empêchent la lecture de l'EEG. Grâce à la projection que nous proposons, il suffit donc d'indiquer que les signaux issus de ce capteur sont incertains : les chances d'appartenance aux trois valeurs floues tendront donc d'autant plus vers une distribution uniforme que les parasites durent plus longtemps. En effet, les mesures faites par l'ASPECT sont basées sur des tendances. Une donnée manquante réduit donc la confiance globale de cette tendance, sans pour autant la rendre inexploitable. Cela permet donc la prise en compte des capteurs qui n'ont pas été perturbés. De la même façon, on sait que l'activité musculaire perturbe la lecture de l'EEG. On peut alors diminuer graduellement son importance dans le processus de diagnostic très facilement en se servant de la mesure de l'EMG comme indice de confiance quant à la valeur du BIS.

3.4.2 Les aspects techniques

Pour implanter ces ensembles de valeurs flous, j'ai utilisé des fonctions sigmoïdales, ou du moins de forme similaire. En effet, les sigmoïdes usuelles sont dérivées de fonctions exponentielles. Ces dernières ont de très bons comportements aux limites, mais elles n'atteignent jamais leurs bornes, ce qui peut être préjudiciable dans notre cas. En effet, avec une telle fonction, aucune valeur ne serait jamais sûre d'appartenir à un ensemble donné. Ainsi, même une distance kilométrique ne serait lointaine pour le robot qu'avec une certitude de 99,99%, alors que ses capteurs ne portent même pas à un mètre.

J'ai donc choisi de modeler ces fonctions en partant de leurs caractéristiques : ces fonctions doivent être continues, et elles doivent atteindre leurs bornes. Elles doivent assurer également la stabilité des valeurs proches de leurs points d'ancrage. Cela implique que leur dérivée première doit être nulle en ces points. Pour préciser les choses, je vais prendre l'exemple de la valeur *trop grande* dont la forme est représentée sur la droite de la Figure 7. Cette fonction sera définie entre 0 et 1, sachant que l'on peut la prolonger par continuité au-delà de ces points. En définitive, cette fonction sera donc définie par morceaux, la première partie étant une droite d'ordonnée nulle, et la dernière une droite d'ordonnée égale à 1. La partie qui assure la transition sera un polynôme de degré 3. Ce dernier est en effet le polynôme de degré minimal permettant de prendre en compte les 4 contraintes correspondant à la valeur et à la dérivée de chacune de ses bornes.

$$\begin{cases} \text{trop\_grand}(x)=1 & \forall x \geq 1 \\ \text{trop\_grand}(x)=0 & \forall x \leq 0 \\ \dfrac{d\ \text{trop\_grand}}{dx}(1) = 0 & \Leftrightarrow \textit{trop\_grand}(x) = 3x^2 - 2x^3\ \forall x \in [0,1] \\ \dfrac{d\ \text{trop\_grand}}{dx}(0) = 0 & \end{cases}$$

De façon similaire, la valeur *trop_petit* sera définie de la façon suivante :

$$\begin{cases} \text{trop\_petit}(x)=1 & \forall x \leq -1 \\ \text{trop\_petit}(x)=0 & \forall x \geq 0 \\ \dfrac{d\ \text{trop\_petit}}{dx}(-1) = 0 & \Leftrightarrow \textit{trop\_petit}(x) = 3x^2 + 2x^3\ \forall x \in [0,1] \\ \dfrac{d\ \text{trop\_petit}}{dx}(0) = 0 & \end{cases}$$

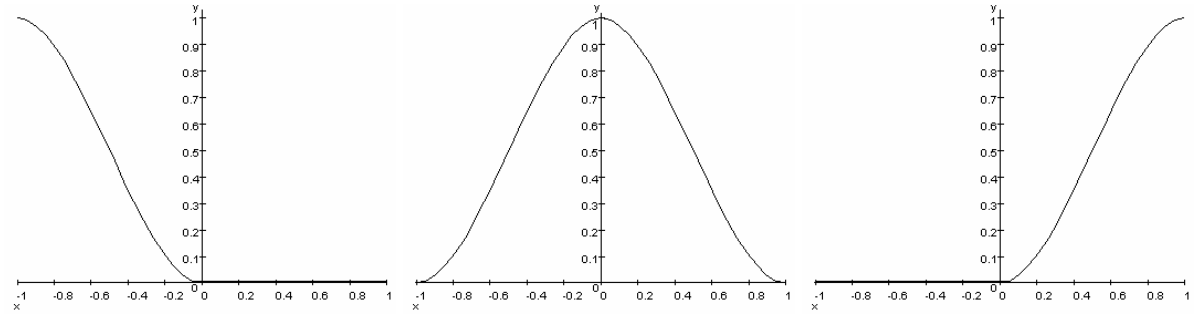


Figure 7 : Les trois valeurs floues du modèle perceptif

Finalement, la valeur *normale* sera simplement définie comme le complémentaire à ces deux fonctions, de façon à ce que la somme des trois fasse 100% pour toute valeur réelle. Cette valeur sera donc différente de 0 uniquement dans l'intervalle ouvert] -1 ; 1[.

Il est naturellement possible d'utiliser ces fonctions sur un autre intervalle que [-1 ; 1], en utilisant une simple formule de changement de base :

$$x = \frac{y - centre}{largeur}$$

Avec ce modèle, tout capteur continu peut exprimer sa valeur sous la forme d'un vecteur de certitudes. Si la valeur d'un capteur est indisponible, un vecteur de probabilités uniforme suffit pour le modéliser. De même, si un capteur hésite entre deux valeurs numériques, il suffit de donner la moyenne des distributions correspondant à ces deux valeurs pour le prendre en compte. On peut même pondérer cette moyenne par la croyance du capteur en chacune des valeurs entre lesquelles il hésite, si ces croyances sont disponibles.

Si l'on désire obtenir plus de précision au niveau de la discrétisation, il est possible d'ajouter autant de ces fonctions que nécessaire pour atteindre la finesse voulue. Ainsi, on pourrait utiliser une répartition basée sur cinq valeurs : *Normale, Grande, Petite, Très Grande, Très Petite*.

Ce modèle est donc particulièrement adapté aux problèmes de diagnostic, puisqu'il permet de prendre en compte des valeurs continues avec une bonne précision, tout en gardant un nombre acceptable de paramètres. En effet, on associera à chaque capteur fournissant une valeur continue un jeu d'intervalles flous. A chaque observation reçue, cet ensemble associera alors une distribution de probabilités d'appartenance à chacun des intervalles. Bien entendu, la forme des fonctions floues peut être modifiée à volonté, tant que l'on conserve une somme égale à 100% sur l'ensemble des intervalles.

De plus, l'aspect symbolique des intervalles utilisés permet de simplifier la collaboration avec un expert humain, puisque la sémantique de ces valeurs est définie de manière logique. Une approche similaire a été suivie dans (1), mais les notions floues ont été utilisées pour caractériser l'état et les actions du système plutôt que les perceptions du système. Le but en était inverse et complémentaire à la fois : tirer parti de l'expression des règles des experts pour obtenir un système informatique autonome.

4 Tirer parti des connaissances des experts

4.1 Motivation

Lorsque l'on désire appliquer des algorithmes d'intelligence arti ficielle à un problème, il est souvent nécessaire de faire appel à des spécialistes du domaine d'application. Comme je l'ai montré dans le chapitre précédent, pour obtenir un diagnostic utilisable, il faut tout d'abord modéliser le problème sous une forme claire et précise. Cette forme dépend bien sûr du but que l'on recherche, mais aussi des connaissances que l'on a du domaine en question. Pour pouvoir mettre en évidence les causes qui expliquent les observations du système, une connaissance fine du processus qui aboutit à la création de ces observations est nécessaire. Or cette connaissance est justement l'apanage des experts du domaine auquel se rattache le problème. En effet, qui connaît mieux ce domaine qu'eux ?

Lors de son apparition, l'intelligence ar tificielle a tout d'abord tenté de reproduire le raisonnement de ces experts humains. Cela semble en effet être une approche prometteuse, puisque c'est ainsi que ces experts mènent à bien leurs tâches. Ce mimétisme a conduit à la conception des systèmes à base de règles. Je vais donc présenter ce type de modèle, et en expliquant son application au projet Diatelic.

Je conclurai ce chapitre en montrant les faiblesses de cette approche, qui recoupent pour beaucoup celles que l'on reproche à ces systèmes en gén éral : règles trop touffues pour être exploitables, absence de profils personnalisés, mauvaise adaptabilité, et évolution trop déterministe.

4.2 Systèmes à base de règles

Historiquement, ce furent les premiers systèmes implantés en intelligence artificielle. Ils permettent en effet de reproduire sur une machine le raisonnement que pourrait faire un expert, en appliquant une série de règles telles que celles que l'expert utilise. C'est la raison pour laquelle on nomme souvent ces programmes des *systèmes experts*.

4.2.1 Présentation

Un système à base de règle est un moteur d'inférence qui travaille sur une base de faits à l'aide d' un ensemble de règles préétablies. Les faits représentent ce que le système connaît. Ils sont donc tous supposés corrects. Dans cette base, en général, il n'existe aucune information indiquant qu'un fait donné est faux. Par extension, tous les faits qui ne sont pas dans la base sont supposés faux, jusqu'à preuve du contraire. Comme le montre le premier chapitre de (39), il existe de nombreux systèmes, tous basés sur les mêmes principes. Je vais donc expliciter rapidement ces derniers.

La base de règles est fournie par un ou plusieurs experts du domaine auquel appartient le problème à traiter. Chaque règle se présente sous la forme d'une relation qui permet d'inférer de nouveaux faits sous réserve que les prémisses de la règle soient respectées.

Ces conditions sont souvent exprimées dans le formalisme de la logique et des prédicats du premier ordre, car ce mode d'expression se prête bien à cet exercice, et surtout parce qu'elle permet de prouver formellement la validité du raisonnement suivi. Par exemple, la règle suivante permet de formaliser la transitivité de l'opérateur *sur(x, y)* dans un monde où des blocs peuvent être entassés.

sur(A,B) et sur(B,C) → *sur(A,C)*

Toute variable non liée est supposée universellement quantifiée. Ainsi, la règle précédente est applicable pour toute combinaison des variables A, B, et C. Evidemment, seules celles qui valident les prémisses de la règle présentent un intérêt pour la suite du raisonnement. En règle générale, il n'existe pas de quantificateur existentiel. Bien que ce formalisme semble limitant au premier abord, il permet en fait d'exprimer une multitude de problèmes. De plus, l'assi se théorique de ces méthodes est très forte puisqu'elle repose sur la logique du premier ordre.

Une fois que la base de règles a été constituée, et que les faits connus sont rassemblés, une question peut être posée au système. Cette question prend la forme d'un ou plusieurs faits dont l'utilisateur désire connaître la validité. Pour dire si oui ou non le système peut prouver ces faits, deux grandes classes d'algorithmes peuvent être utilisées : les algorithmes par *chaînage avant*, et les algorithmes par *chaînage arrière*. Ces deux classes se différentient par l'utilisation qu'ils font de la question posée et des faits de la base.

Tout d'abord, la recherche par *chaînage avant* permet d'utiliser les faits connus avec les règles de la base, de façon à créer de nouveaux faits. Il est alors possible de prouver un fait donné si et seulement si ce fait peut être généré à partir de la connaissance initiale. Ce processus peut être long, et peut aussi ne pas terminer si la base de règles permet de générer un nombre infini de nouveaux faits. En effet, dans ce cas, la recherche ne peut aboutir à une impossibilité que s'il est impossible de générer de nouveaux faits.

La méthode par *chaînage arrière* part quant à elle du fait à prouver. Elle cherche alors si l'une des règles de la base permet de prouver ce fait. Si plusieurs règles peuvent être appliquées, l'une d'entre elles est testée ; les autres sont conservées au cas où la suite de la preuve échouerait. Lorsqu'une règle a été sélectionnée, l'algorithme regarde récursivement s'il est possible de prouver chacune des prémisses de cette dernière. Lorsque l'une des prémisses correspond à un fait de la base, il est validé et l'on passe à la preuve des autres prémisses nécessaires à la preuve en cours.

Si l'une de ces preuves aboutit à un fait qu'il est impossible de prouver, il est nécessaire de regarder si l'une des règles qui ont été mises en réserve permet de faire aboutir la preuve. Lorsque toutes les règles ont été essayées en vain, on sait qu'il est impossible de prouver la validité de la question et on peut répondre par la négative. Bien évidemment, il est possible de faire boucler cette recherche d'une façon similaire à la recherche précédente, en créant une règle qui nécessite sans cesse de nouvelles prémisses.

D'autres types de recherches ont été mis au point, mais ces deux approches sont les plus couramment utilisées. La plupart sont des variantes qui utilisent des approches hybrides cumulant chaînage avant et arrière, simultanément ou en alternance. Nombre d'entre elles incluent également la notion de règle prioritaire. Cela permet de choisir une règle précise lorsque plusieurs sont envisageables.

4.2.2 Application à DIATELIC

Durant les phases initiales du projet DIATELIC, nous avons beaucoup discuté avec les médecins de l'ALTIR afin de comprendre les bases de leur raisonnement. Les règles que nous en avons retirées étaient si simples qu'il était possible de les implanter directement dans un système à base de règles. Ces règles étaient semblables à celles-ci :

« Si le patient a une tension supérieure de 1,5 cm de mercure à sa tension moyenne, il peut y avoir une hyperhydratation »

« Si la tension différentielle est négative, il est probable que le patient souffre de déshydratation »

4.2.2.1 Implantation du système

Afin de rendre exécutables ces règles, je les ai traduites dans le langage de Fuzzy CLIPS. Ce moteur d'inférence est basé sur CLIPS, système à base de règles du domaine public développé par la NASA en 1984 (15). Par rapport au système général présenté dans la section précédente, CLIPS associe à chaque règle une priorité permettant de choisir une règle lorsque plusieurs sont applicables. Les règles sont aussi plus étoffées, et permettent en particulier d'exécuter une suite d'actions au lieu de si mplement créer un nouveau fait.

En plus de nombreuses améliorations au principe de fonctionnement même du moteur, de nombreuses commandes ont été ajoutées de façon à permettre l'insertion, la suppression ou la modification de faits dans la base. Le système final est donc au moins équivalent à un système de recherche par chaînage avant avec priorités. En particulier, il n'est pas nécessaire de poser une question au système pour pouvoir utiliser les règles de la base ; il est possible de laisser simplement le moteur suivre toutes les règles possibles.

Fuzzy CLIPS est une surcouche qui ajoute à CLIPS une bibliothèque permettant de traiter la logique floue (20). Elle associe donc à chaque fait une valeur de certitude qui permet de prendre en compte des faits dont la véracité n'est pas totalement assurée. De la même façon, chaque règle possède un facteur indiquant si la relation associée est certaine, ou si une incertitude peut la tempérer.

Le calcul des faits est alors bien plus compliqué, car chaque fait est considéré comme incertain. En particulier, il faut que le moteur soit capable de trancher entre l'exécution d' une règle sûre ayant une priorité faible, et une règle incertaine ayant une priorité élevée. Cependant, le moteur est capable de résoudre ces problèmes de façon fort efficace en utilisant une formule permettant de balancer ces deux valeurs et d'en déduire un ordre total sur les règles.

L'interface entre ce moteur et le reste du projet est faite en plusieurs étapes : tout d'abord, la base de règles est introduite dans le moteur. A ces règles s'ajoute le chargement des données du patient à qui appartiennent les données à traiter. On insère alors un nouveau fait contenant les données nouvellement reçues.

A ce moment, le calcul des inférences possibles est lancé, ce qui se traduit par l'exploitation du nouveau fait et la mise à jour du profil des données du patient. Ce calcul se termine finalement par une règle de priorité minimale, règle qui exporte les résultats vers l'application. Pour cela, j'ai dû ajouter une nouvelle fonction au moteur, ce qui est rendu possible par l'architecture ouverte de Clips.

4.2.2.2 Principe de l'analyse

L'analyse est basée sur le calcul de tendances ; ces valeurs particulières représentent une portion de l'état général du patient et sont limitées à l'intervalle [-100 ; +100]. Cela permet en effet de limiter l'impact d'une valeur étrange ou incorrecte , tout en autorisant la consolidation des variations à long terme. De plus, cela permet de modérer légèrement l'application directe des règles de la base en rassemblant tous les symptômes avant de tirer une conclusion.

Ces tendances sont influencées linéairement par la valeur des paramètres observés ; cependant, les règles de diagnostic que nous avons déduites des discussions avec les néphrologues de l'ALTIR sont utilisées pour ajouter de pénalités à certaines tendances. Pour simuler la capacité de guérison du patient, chaque tendance est amputée de 33% quotidiennement. Ainsi, en l'absence de symptômes inquiétants, les tendances convergent rapidement vers leur valeur *normale*, zéro. Le système est donc composé de 19 faits répartis en quatre groupes. 11 d'entre eux permettent de stocker les données du patient en conservant un historique de 2 semaines. 5 de ces faits permettent d'inférer les tendances, et les 5 derniers servent à mémoriser les valeurs de référence qui ont été calculées sur les 15 derniers jours.

La base de règles contient 15 éléments répartis en trois groupes. Le premier groupe correspond à la préparation des données brutes, leur ajout dans l'historique et la modification des valeurs de référence. A ce moment, les valeurs de chaque capteur influencent directement les différentes tendances de façon à obtenir un lissage des diagnostics proposés. Ce sont en effet les seules influences qui ne sont pas basées sur des seuils de déclenchement. Elles sont donc appliquées dans tous les cas.

La différence entre le poids réel du patient et son poids-sec, fixé par le médecin, modifie la tendance d'hydratation sur la base d'une pénalité de 37,5% par kilogrammes d'écart. Cette règle représente donc la prise en compte de la part du poids du patient due à son hydratation.

Hydratation = Hydratation + 37,5(Poids-PoidsSec)*

La variation de la tension est contrôlée de la même façon que le poids, mais la valeur de référence est la valeur moyenne des 15 derniers jours. Le calcul est fait séparément sur la tension prise couché, et sur celle qui est prise lorsque le patient est debout. Tous les calculs sont faits en centimètres de mercure.

Tension = Tension + 37,5(Tension-TensionMoyenne)*

De plus, une alerte est envoyée au médecin si l'une des tensions moyennes change de plus de 1,5 point par rapport à la valeur de la veille. Néanmoins, aucune pénalité n'est ajoutée à une quelconque tendance, car cette instabilité peut être due à des causes totalement distinctes des problèmes d'hydratation. En fait, on ne peut pas rattacher avec certitude ces observations à un problème d'hydratation précis.

La prise en compte de l'ultrafiltration est effectuée séparément sur chacune des poches utilisées. Une fois le calcul terminé, le cumul des influences des différentes poches nous donne un bilan journalier unique. En définitive, cela permet donc de tenir compte de la compensation éventuelle d'une poche par une autre. La tendance est modifiée de 15 points par 200 grammes d'écart entre le différentiel mesuré et le différentiel attendu pour ce type de poche. Cette dernière valeur est obtenue grâce à la moyenne des valeurs mesurées à chaque utilisation de ce type particulier de poche sur les 15 derniers jours.

Poche = Poche + 15((PoidsSortie-PoidsEntree) – DifferentielMoyen)/200*

Une fois ces modifications directes effectuées, les règles de diagnostic que nous avons obtenues des médecins peuvent s'appliquer :

- Si le poids est supérieur au poids-sec + 1,5 kg, il y a un risque d'hyperhydratation ; ajouter 30% à la tendance Hydratation.
- Si le poids est inférieur au poids-sec - 1,5 kg, il y a un risque de déshydratation ; retirer 30% à la tendance Hydratation.
- Si la différence entre la tension debout et la tension couchée est négative, il y a un risque de déshydratation ; retirer 25% à la tendance Hydratation.

Il faut noter que tout déclenchement de l'une de ces règles envoie une alerte mineure aux médecins. Ces alertes sont basées sur les valeurs journalières, et sont relativement incertaines. Néanmoins, les médecins nous les ont demandées, car elles peuvent révéler un problème quant à la santé du patient.

Finalement, un troisième groupe de règles s'applique. A ce moment, toute tendance dont la valeur absolue atteint la valeur seuil de 100% déclenche une alerte majeure à destination des médecins. En effet, cela montre soit une conjonction de plusieurs facteurs concordants vers le même diagnostic, soit une suite d'alertes mineures envoyées jour après jour. En effet, aucune règle n'a assez d'influence à elle seule pour faire atteindre ce seuil à une quelconque tendance. Il faut donc plusieurs déclenchements simultanés, ou jours après jour pour obtenir ce débordement. La confiance dans ces alertes est donc plutôt bonne, et l'alerte qui en découle mérite d'attirer l'attention du médecin.

4.3 Avantages et inconvénients

L'avantage principal de cette méthode tient à la traduction directe des règles de diagnostic utilisées par les médecins. Outre les aspects techniques inhérents au moteur utilisé, cette traduction est relativement facile à effectuer. Il est donc relativement facile d'intégrer et de mettre à jour les connaissances du système, à condition toutefois de bien connaître le fonctionnement de Fuzzy Clips.

Malheureusement, les inconvénients de cette méthode sont légion. Tout d'abord, j'ai montré que, dans le détail des règles utilisées, les règles de déclenchement recouraient à de nombreuses valeurs arbitraires. Celles-ci servent en effet à divers effets, très différents les uns des autres. Elles peuvent ainsi montrer l'influence de tel ou tel paramètre sur une tendance particulière. Par exemple, un poids supérieur au poids-sec augmentera la tendance d'hydratation de 30%. Inversement, la règle d'affaiblissement ampute chacune des tendances d'un tiers chaque jour. Enfin, le seuil de déclenchement de la règle exprimant le lien entre le poids et l'hydratation du patient est retardé jusqu'à ce que le poids dépasse le poids -sec de 1,5 kg. Bien entendu, toutes ces valeurs sont arbitraires, et correspondent plus ou moins aux valeurs que les médecins utiliseraient pour un patient moyen.

Cet effet de seuil est particulièrement gênant, car il introduit des discontinuités dans le diagnostic. Alors qu'un médecin modérera son diagnostic en fonction de l'importance des symptômes, la machine appliquera les règles à la lettre, donnant ainsi des diagnostics très tranchés. Même dans les expériences que nous avons menées avec la logique floue, afin de réduire ces effets néfastes, ces seuils sont restés particulièrement apparents.

Ensuite, le système est bien adapté au diagnostic d'une situation statique, ce qui est un avantage indéniable. Il est ainsi très capable de déterminer chaque jour si la tension d'un patient est dans les normes, ou pas.

Par contre, la prise en compte de l'aspect dynamique, et donc de l'évolution possible des pathologies d'un patient au cours du temps est bien plus difficile et délicate à mettre en œuvre. J'ai dû recourir pour cela à la noti on de tendances. Ces valeurs permettent alors de reporter le résultat du diagnostic d'un jour sur le lendemain, et d'assurer ainsi une certaine continuité. Néanmoins, cela reste une démarche tout à fait artificielle. Les tentatives qui ont été faites pour intégrer une partie basée sur la logique floue ont permis d'améliorer un peu la sensibilité du système, tout en réduisant le taux de fausses alertes. Néanmoins, le moteur ne permet pas d'évaluer simultanément plusieurs valeurs de tendances ayant des certit udes différentes. Pour y parvenir, il faudrait alors dupliquer chacun des faits, chaque copie étant dotée d'une valeur particulière et de la certitude associée. Dans un tel cas, la gestion des faits devient vite inextricable et très difficile à maintenir. Enfin, cette multiplication des faits serait très difficile à traduire en langage naturel, si l'on devait expliquer le raisonnement de la machine à un médecin. Cela va donc à l'encontre même de l'avantage principal des systèmes à base de règles.

Finalement, l'application des règles du système se fait en effet en plusieurs étapes. Par exemple, il est préférable de mettre à jour les tendances en utilisant le maximum de règles possible avant de générer les alertes à destination des médecins. Cela évite ainsi de générer autant de messages que de règles déclenchées, de même que cela évite l'envoi de messages contradictoires. Il est donc nécessaire de mettre en place un système de priorités pour assurer l'échelonnement de l'activation des différentes règles . L'ajo ut d'une nouvelle règle à la base doit alors être fait en respectant les priorités établies, tout en autorisant sa mise en œuvre au bon moment. Cela nécessite donc fréquemment de reprendre le calcul de la totalité des priorités du système, de façon à ménager un espace permettant d'insérer une nouvelle règle.

Le dernier inconvénient de cette méthode est lié plus à son implantation informatique qu'au modèle de raisonnement utilisé : étant donné que chaque patient possède son propre profil, avec sa base de seuils personnalisés et sa base de faits personnelle, il est nécessaire de stocker toutes ces données séparément pour chaque patient. Deux hypothèses ont été envisagées : stocker toutes ces données dans la base globale du système et stocker chacun de ces paquets dans des fichiers séparés du serveur. La première version impose donc de reconstituer les différentes bases point par point, à chaque appel du système pour chaque patient. De plus, ces informations techniques sont stockées conjointement avec les données structurées des patients dans la base. Toute modification du système implique donc un arrêt et une reconfiguration de toute la base de données. Cette approche est donc difficilement envisageable.

La deuxième approche nécessite le chargement des bons fichiers à chaque appel du système. Heureusement, Fuzzy CLIPS nous offre les commandes nécessaires à la sauvegarde et au chargement des différentes bases sur le disque. Cependant, cette approche n'est pas satisfaisante, car les données se trouvent stockées directement sur le disque dur du serveur, sans sauvegarde ni sécurité. De plus, rien n'est structuré autrement que par le nom du fichier, dont une partie correspond à l'identifiant du patient. Cette solution serait donc difficilement applicable en situation réelle, avec de nombreux patients dont le traitement dépendrait du diagnostic fourni.

Une troisième approche requerrait la réécriture complète d'un nouveau moteur d'inférence capable de résoudre nos problèmes spécifiques. Il s'agit cependant d'un travail ti tanesque, et cette approche a été abandonnée rapidement au vu des défauts inhérents à ce modèle, indépendamment du moteur utilisé.

4.4 Conclusion

En définitive, la mise en place d'un système à base de règles permet une application directe des règles qu'un expert du domaine utilise habituellement. Cependant, cette implantation requiert l'utilisation de nombreux artifices qui permettent une application correcte des règles énoncées. De plus, le déclenchement des règles est soumis à des seuils qui sont noyés dans le code de ces règles, à moins de les déporter dans des faits séparés. Cette dernière solution permet en effet de les rendre facilement accessibles. Cela se fait néanmoins au prix de la simplicité de la description des règles, puisque cela multiplie artificiellement le nombre de prémisses nécessaires à l'application de chaque règle.

Finalement, l'utilisation d'un tel système est bien adaptée au diagnostic d'une situation statique, mais elle est relativement mal adaptée au suivi d'un processus dynamique qui évolue au cours du temps. En effet, cette évolution se doit d'être réalisée dans tous ses détails par le concepteur de la base de règles du système. Puisque les relations temporelles ne sont pas disponibles dans le moteur d'inférence, tout doit être simulé par les règles de la base.

Le projet DIATELIC nous a montré que le système pouvait se comporter de façon assez satisfaisante, mais que son application sur plusieurs patients simultanément restait impossible. En effet, les paramètres du diagnostic étant réglés pour un patient donné, son application sur un autre patient implique de nombreuses erreurs, même si nombre d'entre elles sont assez bénignes. Le recours à un patient moyen permet de contrebalancer quelque peu cette tendance, mais elle implique en contrepartie que le système n'est vraiment adapté à aucun patient. Cela se traduit alors par une hypersensibilité à de petites variations chez certains patients, tout en ignorant certaines situations graves chez d'autres. Les médecins se sont alors trouvés noyés sous une multitude d'alertes (3 à 4 par patient et par jour), à un tel point qu'ils ont fini par nous demander de retirer le système du projet.

Les alternatives permettant d'individualiser les règles pour chaque patient sont particulièrement difficiles à mettre en œuvre, car une fois toutes les règles explicitées, le système devient si complexe que le médecin est totalement incapable de régler les paramètres du système sans l'aide d'un informaticien.

Toutes ces raisons impliquent que l'utilisation d'un système à base de règles pour le suivi régulier d'un ou plusieurs systèmes évoluant dans le temps est impraticable, surtout si les divers systèmes utilisent des paramètres différents. Il est donc nécessaire d'envisager d'autres modèles dans lesquels le temps serait pris en compte de manière intrinsèque.

Afin de pouvoir traiter correctement les données reçues par le système, et comme l'indique Werner Horn dans (26), il est nécessaire de passer d'un système centré sur la connaissance à un système centré sur les données. Ce faisant, nous reconnaissons ne pas pouvoir décrire totalement le processus. Nous allons donc nous concentrer sur l'aspect modélisation d'un processus observé à travers des données imparfaites.

5 Prendre en compte l'imprévisible

5.1 Motivation

Dans le chapitre explicitant le cadre applicatif de cette thèse, nous avons vu que les systèmes que l'on cherche à analyser sont fortement bruités. Que ce bruit soit dû à une modélisation approximative du problème, à des perturbations inconnues, ou à des règles d'évolution incertaines ne change en rien la question. Il est nécessaire de prendre en compte cette incertitude à tous les niveaux si l'on désire obtenir un système suffisamment robuste pour être utilisé hors du laboratoire.

Je présenterai donc dans ce chapitre la grande famille des modèles stochastiques. Pour être plus précis, je vais m'intéresser aux modèles Markoviens. Après la description préalable de l'arborescence de ces modèles, je présenterai les principaux algorithmes de localisation et de planification qui s'appliquent à eux.

Ces modèles peuvent s'adapter à de nombreux types de problème, comme je le montrerai en présentant leur application à chacun de nos trois problèmes. Ces exemples me permettront alors de conclure en exhibant leurs facultés de modélisation et d'adaptation à des cas réels. J'expliquerai en particulier comment leur fonctionnement simple, s'appuyant sur un modèle perceptif intuitif, permet une interaction aisée avec les médecins.

5.2 Les modèles stochastiques

5.2.1 Présentation

5.2.1.1 Le cadre général

Les modèles stochastiques sont des représentations de systèmes dynamiques basées sur les probabilités. Bien que cet aspect aléatoire semble assez fruste de prime abord, c'est en fait une représentation très naturelle. Souvent, lorsque l'on observe un système, on est capable de trouver intuitivement des règles que le système suit, au moins globalement.

L'approche standard consiste à prendre le cas moyen, celui qui ressemble le plus à toutes les observations, et à considérer qu'il modélise le système de façon suffisante. La variance du processus, autour du cas moyen, est tout simplement ignorée. L'approche statistique consiste à tenir compte de chacune des différentes possibilités, tout en lui attribuant une probabilité qui exprime la fréquence de ses occurrences. Ainsi, plus un cas se produira souvent, plus sa probabilité sera importante.

L'étude des processus stochastiques a commencé au début du 20° siècle grâce à un mathématicien Russe, Markov Andreï Andreïevitch. Son étude statistique du langage l'a conduit à formuler l'hypothèse Markovienne, qui peut se résumer ainsi :

« L'évolution future d'un système ne dépend que de son état présent ».

Autrement dit, cette hypothèse implique que l'état courant du système contient toute l'information apportée par le passé. C'est donc une hypothèse très forte, mais qui semble relativement logique. En pratique, on constate que de nombreux systèmes enfreignent cette condition. Cependant, en affinant le modèle, on peut souvent le rendre markovien.

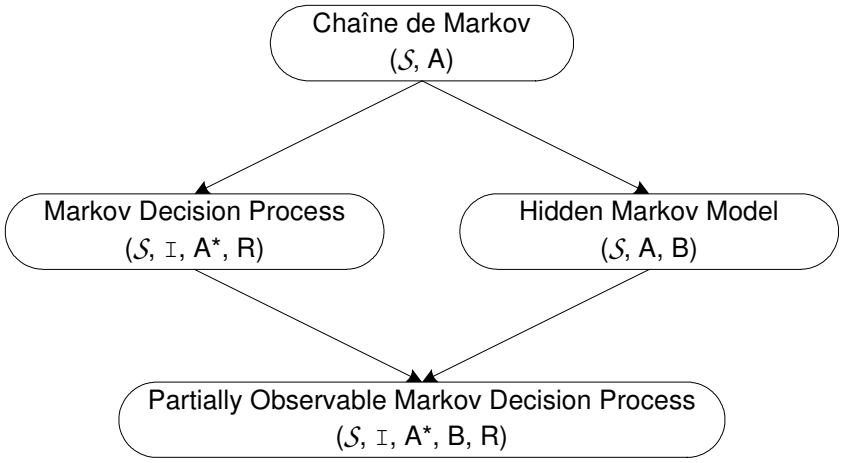


Figure 8 : Diagramme des modèles Markoviens

Ainsi, par exemple, le mouvement oscillatoire d'un pendule est non markovien si l'on considère que son état correspond à sa position spatiale. En effet, à une position donnée, le pendule peut aller indifféremment dans un sens ou dans l'autre. La connaissance de la position passée permet quant à elle d'affiner la description, puisqu'elle nous donne la direction du mouvement. Par contre, si l'on ajoute à la description de l'état une indication de la vitesse instantanée du pendule, on peut appliquer simplement les lois de la physique pour démontrer que le système devient markovien. Cette information suffit à caractériser le mouvement de manière déterministe.

L'hypothèse Markovienne est à la base de nombreux modèles stochastiques. Parmi ceux-ci, je vais m'intéresser aux modèles graphiques, et plus spécifiquement aux chaînes de Markov. En effet, la configuration naturelle de ces modèles permet de représenter de façon visuelle les lois régissant le système étudié. Ainsi, la collaboration avec des experts du domaine en est facilitée, puisque le modèle devient très rapidement compréhensible pour eux. De plus, ces modèles ont de très bonnes propriétés mathématiques (57) et ont été énormément étudiés durant ces dernières années par la communauté des chercheurs.

Je montrerai ensuite comment on peut mettre en œuvre cette famille de modèles dans le cadre des problèmes de diagnostic, en utilisant en particulier les processus Markoviens partiellement observables. La Figure 8 montre la hiérarchie de ces modèles. Sur ce diagramme, les modèles les plus simples sont au sommet, et chaque modèle dérivé ajoute de nouveaux paramètres (entre parenthèses), ainsi que de nouvelles fonctions.

5.2.1.2 Les chaînes de Markov

Les chaînes de Markov forment la base de tous les modèles dits *Markoviens*. Ce modèle permet en effet de modéliser l'évolution d'un système dont on connaît l'état à chaque instant. Il se focalise donc principalement sur la partie évolution, en oubliant totalement la partie observation, de même que les aspects de diagnostic ou de planification. La Figure 9 présente un exemple de modèle Markovien tiré de l'article de Rabiner (59).

Cet exemple correspond à un modèle particulièrement simplifié de la météorologie. Chacun des rectangles représente un état du système, c'est-à-dire une condition météorologique particulière. Il s'agit donc d'un système à états finis. Je me limiterai à la description de ce type de modèles, car ils correspondent bien au problème de diagnostic tel que je l'ai présenté.

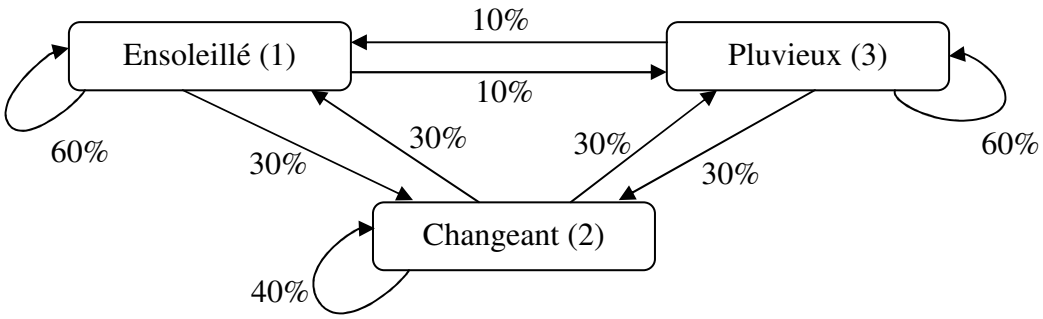


Figure 9 : Un exemple de chaîne de Markov, la météorologie

$$\begin{array}{l} 1 \rightarrow \\ 2 \rightarrow \\ 3 \rightarrow \end{array} \begin{bmatrix} 0,6 & 0,3 & 0,1 \\ 0,3 & 0,4 & 0,3 \\ 0,1 & 0,3 & 0,6 \end{bmatrix}$$

Figure 10 : Une matrice de transition pour la météorologie

Chacune des flèches représentées sur le diagramme permet de changer d'état en un pas de temps. Le modèle suppose donc que le temps est lui aussi discret. Chaque transition est associée à une probabilité qui représente les chances que cette évolution se produise si le système se trouve dans l'état considéré. Dans cet exemple, chaque état peut être atteint, quelque soit la position actuelle du système. On dit alors que le système est *ergodique*. Bien sûr, les transitions étant probabilistes, passer d'un état à un autre peut potentiellement prendre un temps infini.Le système cesse d'être ergodique au moment où certains états deviennent définitivement inaccessibles à partir d'au moins une position.

La notion d'accessibilité ne suppose pas que la transition puisse être effectuée en un seul pas de temps. En fait, il suffit qu'il existe un chemin menant de l'état courant à n'importe quel autre état du modèle, quelle que soit sa longueur, pour que le modèle soit ergodique. Ainsi, un modèle en forme d'anneau, où chaque état ne permet d'aller que vers son voisin immédiat, est un modèle ergodique. Toutes ces transitions sont résumées dans une matrice stochastique (ou Markovienne) telle que celle présentée sur la Figure 10. Dans cette matrice, chaque ligne correspond à un état du système. Une même ligne indique donc quels sont les états accessibles en une seule transition, et quelles sont les chances d'y aboutir en partant de l'état correspondant à cette ligne.

Prenons un exemple : si le soleil brille aujourd'hui, que sera le temps de demain ? La matrice nous indique que le temps sera beau à 60%, pluvieux à 10%, et changeant à 30%. Quel sera alors le temps du surlendemain ? Puisque le système est markovien, la transition de l'état du lendemain au surlendemain ne dépend plus de l'état du lendemain : ce sont deux processus indépendants. Il suffit de composer une nouvelle transition, c'est-à-dire de calculer le carré de cette matrice. Cela nous donne alors 46% de beau temps, 33% de pluie, et 21% de temps instable. Pour conserver cet état de fait en mémoire, il faut introduire la notion de *belief-state*. Ce belief-state est une distribution de probabilités sur les états du système. Ce vecteur–colonne correspond alors à la croyance du système quant à son état. Cette distribution est donc une mesure continue de l'appartenance à un ensemble discret d'états.

L'intérêt de ce type de modèles tient principalement au calcul de l'évolution à long terme du modèle. On peut en effet calculer assez facilement des données telles que l'existence d'une distribution stationnaire, c'est-à-dire un belief-state invariant au cours du temps. On peut aussi détecter l'existence d'états, ou d'ensembles d'états, absorbants. En effet, dans les modèles non-ergodiques, il arrive souvent qu'aucun état ne soit absorbant, mais qu'il soit impossible de sortir d'un groupe d'états donné, quelle que soit la transition envisagée.

Enfin, la période d'un état, ou sa fréquence d'occurrence, peut dans certains cas être une propriété importante pour le système. Ainsi, par exemple, pour un robot mobile, il est intéressant de savoir le nombre moyen de transitions qui séparera deux passages en un point donné. Ce point peut par exemple être un lieu particulier qu'il doit surveiller, ou une borne lui permettant de recharger ses batteries.

Cependant, pour pouvoir appliquer ces modèles aux problèmes de diagnostic, il faut tenir compte de la notion d'observation. Celle-ci a été ignorée jusqu'à présent, puisque l'on connaît avec certitude la position du système à chaque instant. Or cette omniscience est tout sauf réaliste, comme je l'ai expliqué dans le chapitre portant sur notre modèle perceptif. Pour introduire cette notion d'observation, il est nécessaire de cacher l'état derrière un processus d'observation stochastique. On aboutit alors à la notion de modèles de Markov cachés.

5.2.1.3 Les modèles cachés

Les chaînes de Markov cachées (Hidden Markov Models, ou HMM) sont une évolution des chaînes de Markov standard où l'on perd la connaissance de l'état actuel du système. De nouveau, Rabiner fournit dans (60) une bonne introduction, même si elle est un peu biaisée par son application à la reconnaissance de la parole. En remplacement de cette connaissance, le modèle reçoit à chaque pas de temps une observation, dont la probabilité dépend de l'état réel du système. Il faut donc ajouter à la description du modèle une fonction d'observation qui donne à chaque pas de temps la probabilité de l'observation, en supposant que le système se trouve dans l'un des états du système. Le modèle complet se compose donc de :

S : un ensemble fini d'états

A : une fonction de transition

$$A : S \rightarrow \Pi(S)$$

B : une fonction d'observation

$$B : (S \times O) \rightarrow \mathbb{R}$$

Le système peut alors réestimer son belief-state, en tenant compte de l'observation reçue. En effet, la probabilité d'un état donné dépendra aussi bien de sa capacité à générer une telle observation que des chances d'y aboutir en partant du *belief-state* précédant :

$$\begin{aligned} P(S_t = s, O_t \mid S_{t-1}) &= P(O_t \mid s) \cdot \sum_{q \in S} P(S_t = s \mid S_{t-1} = q) \cdot P(S_{t-1} = q) \\ &= B(s, O_t) \cdot \sum_{q \in S} A(s, q) \cdot P(S_{t-1} = q) \end{aligned}$$

Cette fonction d'observation joue donc un rôle central dans le processus de localisation, ou de diagnostic. Elle peut être explicitée de nombreuses façons. Deux approches principales sont utilisées, selon que la valeur observée est continue, ou qu'il s'agit d'un symbole discret.

L'approche la plus ancienne, et la plus étudiée, est l'approche discrète. Elle consiste à recevoir à chaque pas de temps un symbole discret, issu d'un ensemble fini, connu à l'avance. Comme l'ensemble des valeurs de cette fonction est fini, il suffit de spécifier la probabilité d'apparition de chacun des symboles dans chacun des états pour exprimer cette fonction totalement. On peut alors rassembler ces probabilités sous la forme d'une matrice, comme le montre la Figure 11.

$$\begin{bmatrix} P(O_1 \mid S_1) & P(O_2 \mid S_1) & \ldots & P(O_k \mid S_1) \\ P(O_1 \mid S_2) & P(O_2 \mid S_2) & \ldots & P(O_k \mid S_2) \\ \mid & \mid & & \mid \\ P(O_1 \mid S_n) & P(O_2 \mid S_n) & \ldots & P(O_k \mid S_n) \end{bmatrix}$$

Figure 11 : Une matrice d' observation à symboles discrets

La seconde approche couramment utilisée consiste à utiliser une fonction de distribution de probabilités continue. En règle générale, ces fonctions se résument à une somme pondérée de gaussiennes. Cette formulation permet en effet d'obtenir une grande expressivité, puisqu'il est possible d'obtenir toute fonction à support fini par la combinaison linéaire d'un nombre fini de distributions normales. De plus, les sommes de distributions normales sont pratiquement les seules fonctions continues qui ont suffisamment de propriétés pour garantir la convergence des algorithmes d'apprentissage que je présenterai dans le chapitre sur l'apprentissage et l'adaptation. Le dernier avantage de cette formulation tient à la flexibilité des gaussiennes : rien n'interdit d'utiliser des fonctions multidimensionnelles. On peut alors représenter directement de multiples signaux continus dans une seule fonction.

L'inconvénient majeur de la méthode utilisant une combinaison linéaire de distributions normales tient dans le nombre de paramètres qu'elle nécessite. Une distribution normale unidimensionnelle requiert en effet deux paramètres : sa valeur moyenne et son écart-type. Une combinaison linéaire de n de ces fonctions demandera donc n valeurs moyennes, n écarts-types, et n facteurs pondérant cette combinaison. Au total, $3n$ paramètres sont nécessaires. Si l'on désire passer à un nombre de dimensions supérieur, il faut remplacer l'écart–type par une matrice de covariance qui indique l'étalement relatif de la gaussienne sur les différentes dimensions. Une gaussienne représentant p signaux utilisera donc p^2 paramètres de covariance. Une somme de n de ces gaussiennes demandera donc $p*n$ valeurs moyennes (n vecteurs de dimension p), n matrices de covariance de dimension p^2, et n facteurs pour pondérer la combinaison. Le nombre de paramètres nécessaires pour spécifier complètement la fonction passe donc à :

$$n\,(1+p+p^2) = n\,\frac{p^3-1}{p-1}.$$

Lorsque la fonction d'observation est choisie, il est possible d'intégrer dans le système des séquences d'observations. Ce type de modèles pose alors trois grands problèmes :

- Quelle est la probabilité que le modèle puisse générer une suite d'observations donnée ?
- Quelle suite d'états permet d'expliquer une suite donnée d'observations ?
- Comment apprendre le modèle à partir d'une suite donnée d'observations ?

Ces trois problèmes donnent lieu à plusieurs algorithmes ; le dernier problème sera étudié séparément, dans le chapitre qui porte spécifiquement sur l'apprentissage. Je vais maintenant présenter les solutions classiques des deux autres questions.

La majorité de ces algorithmes sont basés sur l'application récursive de la loi de Bayes. Cette loi est une relation fondamentale qui est à la base de nombreux travaux sur les probabilités. Elle exprime la relation entre les probabilités conditionnelles reliant deux événements statistiques :

$$P(A.B) = P(A \mid B)\,.\,P(B) = P(B \mid A)\,.\,P(A)$$

a) Calcul de la probabilité d'une suite d'observations discrètes

Soient le modèle Markovien partiellement observable $\lambda = (\mathcal{S}, A, B)$ et la suite temporelle d'observations $O = O_1\ O_2\ \ldots\ O_T$. La probabilité de l'observation de O dépend naturellement de la suite S des états traversés par le système. Il faut donc envisager chacun de ces chemins :

$$P(O \mid \lambda) = \sum_S P(O \mid \lambda, S)\ P(S \mid \lambda)$$

Or, on a

$$P(O \mid \lambda, S) = \prod_{t=1}^{T} P(O_t \mid \lambda, S_t) = \prod_{t=1}^{T} B(O_t, S_t)$$

$$P(S \mid \lambda) = \prod_{t=2}^{T} P(S_t \mid \lambda, S_{t-1})\ P(S_1 \mid \lambda) = \prod_{t=2}^{T} A(S_{t-1}, S_t)\ P(S_1 \mid \lambda)$$

D'où, finalement :

$$P(O \mid \lambda) = \sum_S \left(B(O_1, S_1)\ P(S_1 \mid \lambda) \cdot \prod_{t=2}^{T} B(O_t, S_t)\ A(S_{t-1}, S_t) \right)$$

L'application de la programmation dynamique permet alors de simplifier cette relation. On peut en effet remarquer que les relations ne portent que sur t et $t-1$, et que plusieurs chemins peuvent partager les mêmes états au même instant. On peut alors factoriser les calculs, pas de temps par pas de temps.

Soit α_t la probabilité d'un état s connaissant les observations faites jusqu'alors :

$$\begin{aligned}
\alpha_t(s) &= P(O_t \ldots O_1, S_t = s \mid \lambda) \\
&= P(O_t \mid \lambda, O_{t-1} \ldots O_1, S_t = s)\ P(O_{t-1} \ldots O_1, S_t = s \mid \lambda) \\
&= B(O_t, s) \sum_{s' \in \mathcal{S}} P(O_{t-1} \ldots O_1, S_t = s, S_{t-1} = s' \mid \lambda) \\
&= B(O_t, s) \sum_{s' \in \mathcal{S}} P(S_t = s \mid \lambda, S_{t-1} = s')\ P(O_{t-1} \ldots O_1, S_{t-1} = s' \mid \lambda) \\
&= B(O_t, s) \sum_{s' \in \mathcal{S}} A(s, s') \cdot \alpha_{t-1}(s')
\end{aligned}$$

Cette formule nous donne donc la base d'un algorithme récursif. On peut cependant simplifier ce calcul, en calculant les facteurs α en parallèle pour tous les états du système. Cela nous permet donc finalement d'obtenir un algorithme itératif, particulièrement efficace.

$$\alpha_1(s) = B(O_1, s)\ P(S_1 = s \mid \lambda)$$

$$\alpha_t(s) = B(O_t, s) \sum_{s' \in \mathcal{S}} A(s, s') \cdot \alpha_{t-1}(s') \qquad \text{pour } t \text{ de } 2 \text{ à } T$$

Cet algorithme est à la base des différents algorithmes que je présenterai par la suite. J'y ferai référence sous le nom de *procédure Forward*. Une fois que ces coefficients sont calculés, on peut en déduire très facilement l'expression de la probabilité de la séquence d'observations :

$$P(O \mid \lambda) = \sum_{s \in S} \alpha_T(s)$$

Cet algorithme est particulièrement efficace, puisque sa complexité est proportionnelle au nombre d'états que multiplie le nombre d'observations. Ajoutons qu'il est possible de le rendre incrémental, tout en ne conservant que le dernier ensemble de valeurs α_T à chaque nouvelle observation. Le calcul de la nouvelle probabilité devient donc très rapide, tout en conservant une taille mémoire très restreinte.

b) Calcul du chemin optimal

Tout d'abord, je voudrais préciser la notion de chemin optimal. En effet, deux interprétations sont possibles, et chacune conduit à un algorithme différent. Le chemin optimal peut être considéré comme la suite des états où la probabilité de générer l'observation est maximale, ce qui nous conduit à l'algorithme *Forward–Backward*. Alternativement, on peut considérer que le chemin optimal est la suite d'états qui permet de générer la suite d'observations avec une probabilité maximale. Cette seconde version nous mène à l'algorithme de *Viterbi*.

- **Algorithme de Viterbi**

Cet algorithme a pour but le calcul du chemin ayant la meilleure probabilité de générer une suite d'observations donnée. Elle est basée sur la procédure *Forward* que j'ai présentée dans la section précédente. La principale différence tient au remplacement des sommes par des maxima. En effet, l'algorithme considère seulement un chemin, et non la réunion de tous les chemins possibles. Voici donc la version définitive de cet algorithme :

$\alpha_0(s) = P(S_0 = s \mid \lambda)$ (La distribution probabilité initiale du modèle)

$\alpha_t(s) = max_{s' \in S} \left(A(s, s') \cdot B(O_{t-1}, s') \cdot \alpha_{t-1}(s') \right)$ pour t de 1 à T

$chemin(T) = arg\_max_{s \in S} \left(\alpha_T(s) \right)$

$chemin(t-1) = arg\_max_{s' \in S} \left(A(chemin(t), s') \cdot B(O_{t-1}, s') \cdot \alpha_{t-1}(s') \right)$ pour t de T à 1

Pour résumer, l'algorithme de *Viterbi* procède pas de temps par pas de temps, tout comme la procédure *Forward*. Pour chacun d'entre eux, il faut évaluer la probabilité que le chemin optimal partiel se termine par chacun des états du système. Pour évaluer la probabilité de passer par l'état s au temps t, on teste les chemins venant de chacun des états s' du système. La Figure 12 résume ce processus de manière schématique.

Lorsque toutes les observations de la séquence ont été prises en compte, le meilleur chemin est connu. Il s'agit en effet de celui ayant la meilleure probabilité à cet instant. On peut alors remonter le temps en cherchant à chaque pas de temps l'état le plus susceptible de mener au prochain état, que l'on sait optimal.

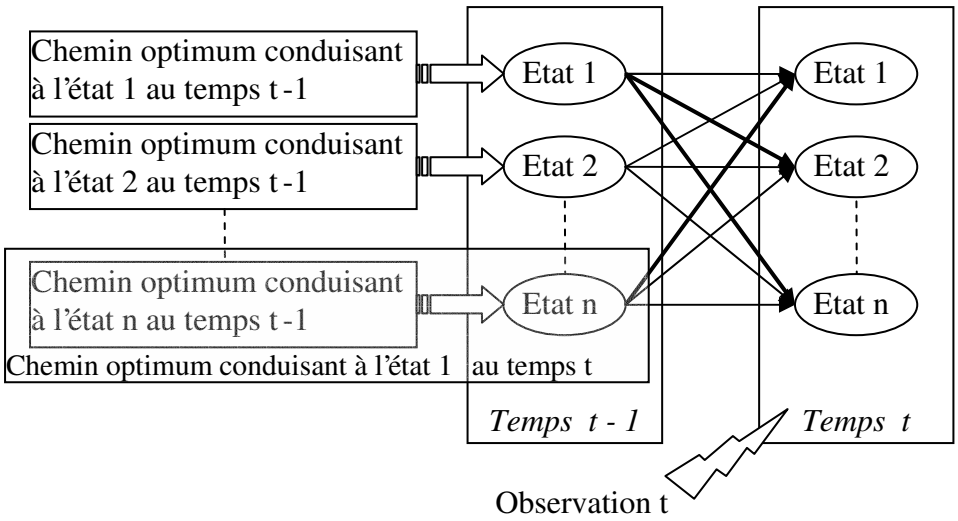


Figure 12 : Construction du chemin optimal par l'algorithme de Viterbi

Au contraire de la procédure *Forward*, cet algorithme ne peut donc pas facilement être rendu incrémental. Il est en effet nécessaire de reprendre tous les pas de temps pour créer le chemin à partir de son état final. La dernière phase nécessite en particulier de faire référence à tous les pas de temps ; il est donc nécessaire de les conserver tous en mémoire.

– **Algorithme Forward-Backward**

L'algorithme *Forward-Backward* est basé sur la procédure *Forward* que j'ai présentée lors du calcul de l'adéquation du modèle à une suite d'observations. On y ajoute alors une procédure symétrique, la procédure *Backward.* Celle-ci permet de calculer la probabilité de générer la fin de la séquence d'observations en partant d'un état donné. Ensuite, une troisième série de calculs est faite pour produire la probabilité de chaque état à chaque pas de temps. Finalement, le chemin optimal est calculé en sélectionnant le meilleur état de chaque pas de temps. La phase *Backward* de la procédure calcule la probabilité de la fin de la séquence d'observations, en partant d'un état donné. Voici donc les calculs correspondants :

$$\begin{aligned}
\beta_t(s) &= P(O_{t+1}\, O_{t+2} \ldots O_T \mid \lambda, S_t = s) \\
&= \sum_{s' \in S} P(S_{t+1} = s' \mid \lambda, S_t = s) \,.\, P(O_{t+1}\, O_{t+2} \ldots O_T \mid \lambda, S_{t+1} = s') \\
&= \sum_{s' \in S} A(s', s) \,.\, P(O_{t+1}\, O_{t+2} \ldots O_T \mid \lambda, S_{t+1} = s') \\
&= \sum_{s' \in S} A(s', s) \,.\, P(O_{t+1} \mid \lambda, S_{t+1} = s') \,.\, P(O_{t+2}\, O_{t+3} \ldots O_T \mid \lambda, S_{t+1} = s') \\
&= \sum_{s' \in S} A(s', s) \,.\, B(O_{t+1}, s') \,.\, \beta_{t+1}(s')
\end{aligned}$$

A nouveau, la programmation dynamique permet de simplifier le processus de calcul ; cette formule peut alors se transformer très facilement en un algorithme itératif, de manière similaire au calcul de la procédure *Forward*. L'algorithme itératif est donc le suivant :

$$\begin{aligned}
\beta_T(s) &= 1 && \forall s \in S \\
\beta_t(s) &= \sum_{s' \in S} A(s', s) \,.\, B(o_{t+1}, s') \beta_{t+1}(s') && \forall s \in S
\end{aligned}$$

Nous avons donc maintenant à notre disposition les valeurs suivantes :

$$\alpha_t(s) = P(O_t \ldots O_1, S_t = s \mid \lambda)$$
$$\beta_t(s) = P(O_{t+1} \ldots O_T \mid \lambda, S_t = s)$$

On peut donc très facilement calculer la probabilité finale de chaque état :

$$\gamma_t(s) = P(S_t = s \mid \lambda, O) = \frac{\alpha_t(s) \,.\, \beta_t(s)}{\sum_{s' \in \mathcal{S}} \alpha_t(s) \,.\, \beta_t(s)} \qquad \forall s \in \mathcal{S}$$

Finalement, la construction du chemin optimum est triviale, puisqu'il suffit de sélectionner à chaque pas de temps l'état affichant la plus grande probabilité. On peut remarquer que ce calcul est relativement coûteux, et qu'il n'est pas possible de le rendre incrémental non plus.

Il est bon de noter que ce chemin optimal peut ne pas être cohérent avec le modèle. Par exemple, la séquence d'états peut suivre des transitions dont la probabilité est nulle. En effet, chaque état est optimum à chaque pas de temps. Néanmoins, deux états optimaux successifs peuvent appartenir à deux chemins différents. Rien ne garantit alors que le chemin joignant ces deux états soit légal au sens de la matrice de transitions.

5.2.1.4 Processus de décision Markoviens

Les modèles que j'ai présentés jusqu'à présent permettent d'analyser le fonctionnement d'un système dynamique et d'en découvrir l'état caché en se basant sur des observations partielles et incomplètes. Cependant, nous nous sommes contentés de *regarder* le processus passivement. Si l'on désire influencer l'évolution du système, il faut compléter ces modèles. On aboutit alors à la classe des *modèles de décision Markoviens* (12).

Les modèles de décision Markoviens peuvent être répartis en deux branches : les modèles observables (les MDP) et les modèles partiellement observables (les POMDP). On peut d'ailleurs considérer les POMDP comme un raffinement des MDP. La Figure 8 synthétise cette hiérarchie de modèles en rappelant les paramètres de chacun.

Par rapport aux modèles présentés précédemment, les MDP et les POMDP partagent les mêmes ajouts : la notion d'action et de la notion de récompense. La différence entre ces deux modèles est la suivante : les MDP sont construits sur la base d'une chaîne de Markov simple, alors que les POMDP sont construits à partir d'un HMM.

a) La notion d'action

Les processus de décision intègrent la notion d'action. Cette dernière correspond à une influence volontaire portée sur le processus. Contrairement aux modèles que j'ai présentés jusqu'à présent, où le modèle observait passivement le système, le système va désormais être modifié par l'agent logiciel dont le modèle est le cœur.

Dans les modèles statistiques, les actions sont généralement considérées comme discrètes, et en nombre fini. Cela permet en effet de spécifier beaucoup plus facilement le modèle. Nous ajouterons donc à la description du modèle l'ensemble I des influences que l'on peut exercer sur le modèle, accompagnées de l'ensemble des matrices de transition propre à chacune d'entre elles.

En effet, dans tous les modèles Markoviens, nous avions la fonction *A* qui associe à tout couple d'états la probabilité d'aller du premier au second en un seul pas de temps. La matrice de transition correspondante représente donc simplement l'évolution normale du système. Or, une action va perturber ce système ; chaque action doit donc avoir sa propre matrice de transition indiquant en quoi le système va être influencé. Dans les modèles de décision Markoviens, nous aurons donc la fonction *A**, qui associe à un couple d'états et à une action une probabilité. Celle-ci représentera les chances que le système a de passer du premier état au second lorsque cette action est entreprise :

$$A^* : \mathcal{S} \times I \rightarrow \Pi(S)$$

Enfin, les actions choisies peuvent modifier la perception de l'environnement. On peut ainsi choisir de faire un examen complémentaire au lieu de prendre une décision inconsidérée. Par exemple, dans le problème du tigre proposé par Cassandra (12), un agent se trouve devant deux portes. Derrière l'une d'entre elles se trouve une récompense, alors que derrière l'autre se trouve un tigre mangeur d'agents intelligents. Trois actions sont disponibles : ouvrir une porte, ouvrir l'autre, et écouter. Il est clair que les deux premières actions ont un impact radical sur la perception de l'agent, alors que l'action d'écouter prête plus à confusion, puisque la perception résultante est incertaine. Dans un projet temps–réel, une telle action pourrait correspondre à un traitement coûteux en temps, mais donnant des informations plus fiables.

Pour tenir compte de toutes ces possibilités, il est nécessaire de modifier la fonction d'observation. Cette dernière prend alors la forme suivante :

$$B : \left(\mathcal{S} \times I \times O\right) \rightarrow [0, 1]$$

Néanmoins, dans le cas où les actions ne modifient pas l'aspect perceptif, il est aisé de se ramener au cas précédent en définissant la fonction B en oubliant l'action choisie.

b) La notion de récompense

Afin de pouvoir prendre une décision, le système doit être capable de déterminer son but. Pour y parvenir, on procède traditionnellement à l'ajout d'une fonction de récompense. Celle-ci peut alors indiquer au système si ce qu'il fait est intéressant, inutile, ou néfaste vis-à-vis du but qui lui est assigné. Il existe plusieurs manières de définir une récompense.

On peut tout d'abord assigner une récompense au système, simplement parce qu'il se trouve dans un état donné. Alternativement, on peut offrir une récompense pour l'arrivée dans un certain état, ou pour passer d'un état à un autre. Finalement, on peut aussi donner une récompense au système pour avoir choisi une certaine décision dans un certain état. Dans la suite de cette présentation, j'utiliserai cette dernière définition. En effet, elle permet de représenter presque exactement la première des définitions : si la récompense est la même dans un état donné, quelle que soit l'action choisie, la récompense porte alors sur le fait d'agir dans cet état. La seule différence correspond donc au moment où cette récompense est donnée.

$$R : \mathcal{S} \times I \rightarrow \mathbb{R}$$

A partir de cette récompense, le modèle de décision va pouvoir procéder au calcul d'une *politique* permettant d'atteindre le but d'une façon optimale. Une telle *planification* permet donc de préparer un plan d'action, c'est-à-dire une fonction qui attribue à un état du modèle une action à effectuer.

Dans un MDP, la politique π s'écrira donc sous la forme :

$$\pi : S \rightarrow I$$

Finalement, un nouveau type de politiques commence à émerger des travaux actuels : la notion de *politique stochastique*. Ces politiques ne donnent plus une action déterministe en réponse à la situation actuelle, mais plutôt une distribution de probabilités sur l'ensemble I des actions disponibles. Cette approche permet en effet d'éviter les minima locaux qui peuvent piéger les méthodes déterministes, car elle autorise l'agent à utiliser des actions variées lorsqu'il se trouve dans un état donné. Ce dernier a donc moins de chances de rester bloqué dans un état où son action optimale le piège.

c) Les algorithmes de planification

Pour les MDP, il existe de nombreux algorithmes de planification. Les plus courants sont *Value Iteration* et *Policy Iteration* (12). Pour les POMDP, on trouve des algorithmes de résolution exacts tels que *Witness* (42), ou l'algorithme du *support linéaire de Cheng* (14). Parmi les méthodes de résolution approchée, Q_{MDP} (43) me semble être la plus intéressante de par la simplicité de son algorithme.

Pour ce qui concerne les méthodes fournissant une politique stochastique, on peut citer la méthode du *gradient de Baxter* (4).

En plus du type de politique engendrée, les différents algorithmes se différentient aussi sur la façon de choisir une action. On distingue en effet les politiques à horizon fini des politiques à horizon infini. Les premières cherchent l'action qui permet de maximiser la récompense qui sera obtenue pendant les quelques pas de temps dans le futur désignés sous le nom d'horizon. Les politiques à horizon infini cherchent quant à elles à trouver l'action optimale en tenant compte de l'intégralité des récompenses qu'elles obtiendront, et ce jusqu'à la fin des temps.

De façon à assurer la convergence de la valeur de la récompense d'une politique à horizon infini, on recourt généralement à un facteur d'atténuation exponentiel. Ce dernier permet en effet de limiter cette somme infinie à une valeur finie :

$$V(t_0) = \sum_{t=t_0}^{\infty} \gamma^t . R(t)$$

- **Value Iteration**

Cet algorithme (12) ne s'applique que sur les MDP. Cependant, il est sans nul doute le plus important de tous, car il est à la base de la grande majorité des méthodes de résolution des POMDP. Il se base sur le calcul successif des différentes valeurs des politiques à horizon fini, pour des horizons de plus en plus grands. Cette valeur dépend naturellement de la politique utilisée. La valeur attendue pour l'exécution de la politique π pendant t pas de temps, à partir de l'état s sera notée $V_{\pi,t}(s)$.

Calculons tout d'abord la politique optimale à 1 pas : une seule et unique récompense sera donc prise en compte dans le calcul de la valeur de la politique. Cette valeur devra donc correspondre exactement à la récompense instantanée acquise lors de l'exécution de l'action recommandée par la politique :

$$V_{\pi,1}(s) = R\left(s, \pi(s)\right)$$

La politique optimale à 1 pas de temps sera donc celle pour laquelle V est maximale pour chaque état. Dans chaque état, comme la valeur de la politique ne dépend que du choix de l'action à cet instant, il suffit donc de prendre l'action donnant la récompense la plus forte.

$V_{\pi,1}(s) = max_{a \in I} R(s, a)$

$\pi_1(s) = argmax_{a \in I} R(s, a)$

Pour définir la politique à 2 pas, on va procéder de la même façon : calculer la valeur de la politique optimale, puis en déduire la politique correspondante. Pour calculer cette valeur, on va évidemment pouvoir se servir de la politique optimale à 1 pas que l'on vient de définir.

Il ne reste donc qu'à trouver l'action à effectuer dès maintenant, sachant que selon le résultat de cette dernière, on connaîtra l'action optimale suivante :

$$V_{\pi,2}(s) = max_{a \in I} R(s, a) + \gamma \sum_{s \in \mathcal{S}} A(s', s).V_{\pi,1}(s')$$

$$\pi_2(s) = argmax_{a \in I} R(s, a) + \gamma \sum_{s \in \mathcal{S}} A(s', s).V_{\pi,1}(s')$$

On peut ainsi définir récursivement toutes les politiques à horizon fini optimales. On peut alors définir la politique à horizon infini optimale en itérant un nombre infini de fois. Evidemment, on peut aussi s'arrêter lorsque la fonction V n'évolue plus suffisamment.

Il est bon de noter que la politique cesse d'évoluer bien avant que sa valeur ne soit stable. C'est néanmoins plus difficile à détecter efficacement, car l'évolution de la politique peut se faire par paliers.

- **Policy Iteration**

Cet algorithme a été proposé dès 1960 par Howard (27). Dans certains cas, il peut être beaucoup plus efficace que *Value Iteration*; il ne s'applique néanmoins pas aux POMDP dans sa version originelle.

Plutôt que de construire la valeur optimale de la politique et d'en déduire la politique correspondante, cet algorithme travaille directement sur la politique. Avant de commencer, on initialise donc la politique π avec une version initiale. Si aucune n'est disponible, on en crée une aléatoirement en assignant une action arbitraire à chaque état.

On va ensuite essayer d'améliorer la politique courante progressivement. Pour cela, on va commencer par calculer la valeur de cette politique en résolvant le jeu d'équations V(s). On peut aussi utiliser *Value Iteration* pour cette résolution. Il est bon de remarquer que ce calcul est relativement rapide pour deux raisons : tout d'abord, la politique est fixée, et donc la valeur que l'on cherche à calculer n'évolue pas d'itération en itération. Ensuite, la valeur de cette fonction est conservée tout au long de l'amélioration de π. La mise à jour de sa valeur est donc très rapide, car en général, la valeur évolue lentement pour chaque modification de politique.

Cette valeur sert enfin à améliorer la politique : on choisit pour chaque état l'action qui maximise la récompense attendue, en se basant sur la valeur de la politique que l'on vient de calculer :

$$\pi'(s) = argmax_{a \in I} R(s, a) + \gamma \sum_{s \in \mathcal{S}} A(s', s).V_{\pi}(s')$$

Finalement, on itère jusqu'à ce que la politique ne soit plus modifiée. On recalcule donc la valeur, puis la politique correspondante, et ainsi de suite, jusqu'à stabilité.

On peut donc remarquer que, contrairement à *Value Iteration*, on s'arrête dès que l'on a atteint la politique optimale. L'algorithme précédant, lui, continue à améliorer la valeur de la politique, même si cela n'apporte rien au choix des actions.

d) Planification dans les POMDP

Dans un POMDP, la politique est plus difficile à formaliser que pour un MDP. En effet, *l'état courant* du modèle se traduit par un *belief-state*, c'est-à-dire une distribution continue multidimensionnelle. En effet, dans ces conditions, l'hypothèse de Markov peut s'appliquer, car cet *état* renferme toute l'information nécessaire.

Avec un tel état, l'expression de la politique pose un réel problème. En effet, comme l'ensemble sur lequel s'applique cette fonction est continu, il n'est plus possible de la tabuler directement. De plus, la politique doit tenir compte des observations, de façon à prévoir l'évolution du belief-state au fur et à mesure de son déroulement.

Il est donc possible d'exprimer la politique d'un POMDP sous la forme d'un ensemble contenant un arbre pour chaque belief-state possible : chaque nœud de cet arbre correspondra alors au choix d'une action à effectuer, et chaque branchement prendra en compte les observations qui seront reçues dans le futur. L'évolution du belief-state est alors implicitement codée dans l'arbre, ce qui implique qu'il n'est plus besoin de mettre à jour ce dernier pour exécuter la politique.

La profondeur de l'arbre correspond bien entendu à l'horizon de la politique : une politique à 3 pas utilisera 3 actions successives, et sera donc représentée par un arbre à 3 niveaux. La Figure 13 présente un tel arbre.

La valeur d'un tel arbre T se calcule simplement par récursion sur sa structure, en fonction de l'état initial s sur lequel il est appliqué :

$$V_s(T) = R\left(s, sommet(T)\right) + \gamma . \sum_{s' \in S} A(s', s) . \sum_{o} B\left(s', sommet(T), o\right) . V_{s'}\left(fils(T,o)\right)$$

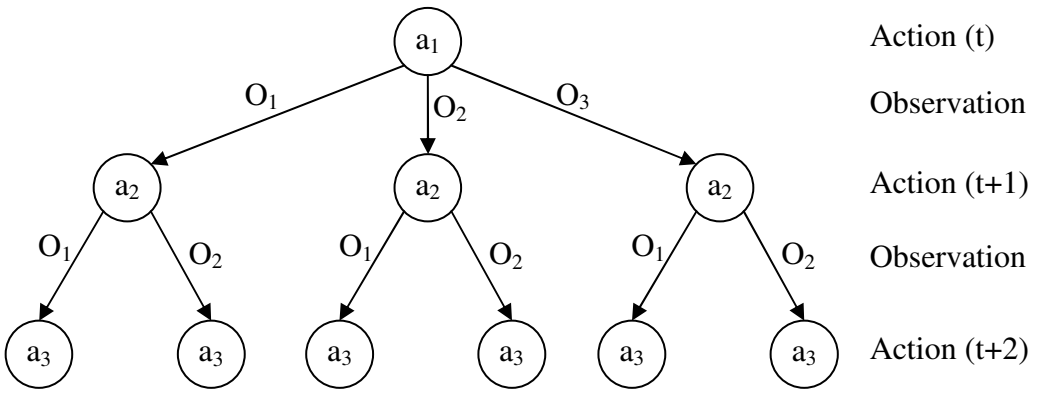


Figure 13 : Une politique à 3 pas pour un belief-state et une action donnés

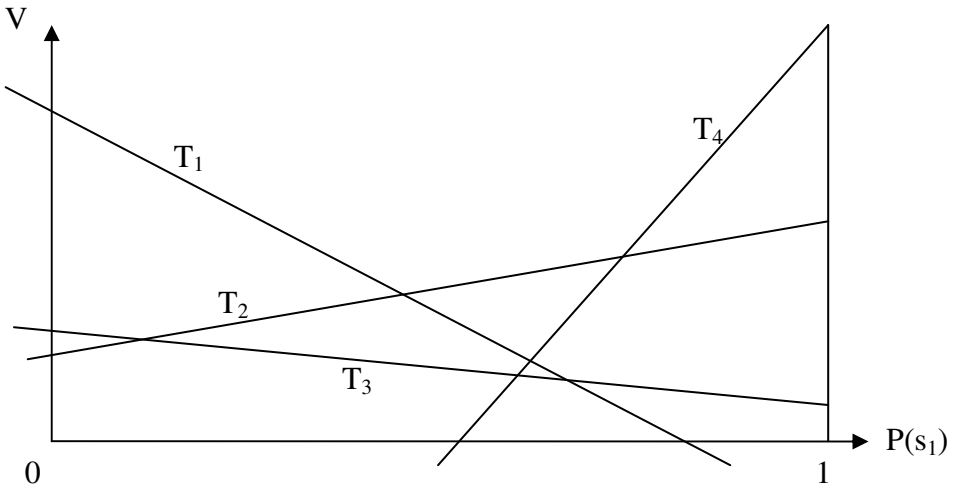


Figure 14 : Comparaison de la valeur de quatre arbres candidats à devenir une politique

On peut alors en déduire la valeur attendue pour l'exécution de cet arbre à partir d'u n belief-state donné de la façon suivante :

$$V(T, b) = \sum_{s \in S} b(s).V_s(T)$$

La valeur d'un tel arbre est donc une fonction linéaire du belief-state. Elle correspond donc à un hyperplan dans un espace de dimension *card(S)*. En effet, puisque la somme des probabilités atteint toujours 100%, l'une de ces valeurs est redondante. La comparaison de plusieurs arbres peut donc être visualisée de manière graphique. La Figure 14 montre une telle représentation, pour un belief-state à deux états, et 4 arbres *T1, T2, T3,* et *T4*.

Cette figure illustre bien le fait que certains arbres peuvent être dominant sur un intervalle donné, alors que d'autres peuvent être complètement dominés par un ou plusieurs arbres concurrents. Ainsi, sur la figure qui me sert d'exemple, c'est le cas de *T3* qui est dominé successivement par *T1*, *T2*, et *T4*.

La politique optimale pour le modèle, pour cette valeur de l'horizon, est donc la conjonction des arbres dominants. Sa valeur est alors convexe et linéaire par morceaux. Cela reste vrai dans le cas des belief-states plus gros, mais en remplaçant les droites par des hyperplans. Exprimer cette fonction est donc bien plus simple que s'il s'agissait d'une fonction quelconque.

Tout le problème revient donc finalement à trouver une telle politique pour un horizon donné. La politique à horizon infini peut de nouveau être calculée comme la limite de ces politiques, lorsque l'horizon tend vers l'infini. Le problème, c'est que le nombre d'arbres potentiellement utiles tend également vers l'infini. La fonction peut donc être constituée d'un nombre infini de segments, ce qui lui retire son aspect sympathique. En règle générale, on observe qu'un nombre fini de segments suffit, mais rien ne le garantit.

- **Q_{MDP}**

Cet algorithme se démarque des autres en ce qu'il ne fait pas de calculs compliqués pour générer une politique de POMDP. En fait, il se base sur le calcul de la politique d'un MDP. C'est donc plus une heuristique qu'un vérita ble algorithme. Cette heuristique a été proposée dans (43).

La seule différence qui le sépare du calcul de la politique d'un MDP tient au calcul de la fonction de valeur qui est faite pour chaque belief-state *b*, à partir de la valeur du MDP obtenu en retirant l'aspect partiellement observable :

$$V(a, b) = \sum_{s \in \mathcal{S}} b(s).V_{MDP}(s, a)$$

Une fois cette valeur calculée pour chaque action dans le belief-state courant, il suffit naturellement de sélectionner l'action donnant la meilleure récompense. Tout se passe donc comme si l'incertitude sur l'état disparaissait miraculeusement après la première action. Le résultat est donc une politique approximative, puisque l'incertitude de la localisation n'est pas propagée sur les pas de temps suivants.

De plus, cette formulation ne prend pas en compte la notion d'observation du tout. Il lui est donc impossible de traiter les observations conditionnées par les actions telles que celles présentées dans la section précédente. Cela reste néanmoins une politique efficace, particulièrement rapide à calculer.

– **Value Iteration pour POMDP**

De même que pour la version de *Value Iteration* utilisée pour les MDP, on va construire des politiques successives ayant un horizon de plus en plus éloigné. On va donc commencer par la politique à 1 pas. Pour cette première politique, l'agent a droit à une seule action. Il va donc se baser sur la récompense instantanée que lui apporte le choix de cette action dans l'état courant. Cet état étant un belief-state *b*, on a simplement :

$$V_1(a, b) = \sum_{s \in \mathcal{S}} b(s).R(s, a)$$

Pour la politique à *n* pas, l'agent peut agir, observer, et agir à nouveau pendant *n-1* pas. Pour chaque action *a*, on calcule donc la valeur de ce choix, en tenant compte du belief-state courant et de l'application de la politique optimale d'horizon *n-1* à partir du rang suivant, quelle que soit l'observation reçue entre-temps. Dans cette équation, afin d'en simplifier l'écriture, on utilisera la notation $b_{a,o}$ pour représenter le belief-state résultant de l'exécution de l'action *a* et de l'observation du symbole *o* à partir du belief-state *b*.

$$V_n(a, b) = \sum_{s \in \mathcal{S}} b(s) \cdot \left(R(s, a) + \gamma . \sum_{s' \in \mathcal{S}} A(s', s) \cdot \sum_{o} B(s', a, o) \cdot V_{n-1}(b_{a,o}) \right)$$

Il existe un nombre fini de ces arbres pour un horizon fini donné, sous réserve que le nombre d'actions et le nombre d'observations soient finis. On peut donc évaluer la valeur de tous ces arbres, ce qui permettra finalement de définir la politique optimale en sélectionnant l'enveloppe convexe supérieure de ces fonctions.

– **Witness**

Cet algorithme, présenté dans (42), procède à la construction de la politique de façon incrémentale, de façon à éviter d'énumérer tous les arbres possibles. Leur nombre peut en effet être très conséquent, ce qui implique que leur évaluation peut être particulièrement longue.

Pour arriver à cette construction, on procédera en ajoutant successivement des arbres à l'ensemble de ceux nécessaires à la réalisation de la politique optimale. On commencera donc, un peu à la façon de *Policy Iteration*, par créer un arbre arbitraire pour initialiser le processus.

On recherche ensuite un belief-state pour lequel la fonction actuelle est moins bonne que la valeur optimale. On construit alors un nouvel arbre qui soit optimal pour ce belief-state. Ce nouvel arbre est ajouté à la politique pour compléter les précédentes et améliorer la valeur de la politique. Il est alors nécessaire de purifier cet ensemble, car l'ajout de ce nouvel hyperplan peut rendre inutile un ou plusieurs de ceux qui y sont déjà.

Il reste donc le problème consistant à identifier un belief-state qui rend non–optimale la valeur actuelle. Ce témoin de non-optimalité, *Witness* en anglais, donne son nom à l'algorithme. Pour en isoler un, Littman propose de tester différentes variantes de chacun des arbres contenus dans la politique actuelle. Une de ces variantes s'obtient alors en modifiant soit le nœud principal de l'arbre, en changeant son action, soit en remplaçant l'un des sous–arbres qui seront utilisés après la prochaine observation. Il est inutile de descendre plus bas dans l'arbre, puisque l'on sait que chacun des sous–arbres est optimal à *n-1* pas. Il faut donc envisager l'ajout à la politique de tout arbre composé d'une action, et pour chacune des observations possibles, de l'un des arbres issus de la politique d'horizon immédiatement inférieur.

Pour réduire le nombre de ces tests, on peut utiliser à nouveau le principe d'optimalité de Bellman : la politique optimale peut mettre en jeu diverses actions pour divers belief-states. Cependant, pour un belief-state donné, l'arbre optimal doit être le meilleur de ceux qui ont l'action optimale à leur sommet. Dans le cas contraire, la politique ne serait pas optimale. *Witness* travaille donc action par action. Ainsi, seuls les arbres ayant l'action en cours d'évaluation à leur nœud principal seront comparés aux nouveaux candidats. Cette étape donne donc en définitive un ensemble d'arbre ayant tous la même action au sommet, chacun étant meilleur que les autres pour au moins l'un des belief-states possibles.

Considérons maintenant la création de cet ensemble : pour chaque arbre T de l'ensemble U_a des arbres qui sont nécessaires à la politique optimale et qui débutent par l'action a, et pour chaque observation o, on crée une série de nouveaux arbres NT_i. Chacun de ces nouveaux arbres correspondra à T en tout point, sauf pour le sous–arbre utilisé si l'on observe le symbole o. Pour cette observation, le sous–arbre correspondant sera remplacé par l'arbre T_i issu de la politique d'horizon *n-1*. Chacun des arbres obtenus ainsi sera ensuite comparé à ceux de l'ensemble courant, U_a. S'il existe un belief-state b où T est supérieur à chacun des arbres de l'ensemble, alors b témoigne de la non–optimalité de l'ensemble U_a. Le nouvel arbre doit donc être ajouté à l'ensemble pour en améliorer la valeur.

Comme la valeur de chaque arbre est une fonction linéaire du belief-state, il est possible de trouver ce b par simple programmation linéaire. Il faut alors itérer ce processus jusqu'à ce qu'aucun des arbres ne puisse améliorer la valeur de l'ensemble. A ce moment, U_a est l'ensemble des arbres débutant par l'action a, et dont la valeur est optimale pour au moins l'un des belief-states. Cet ensemble correspond donc à l'utilisation optimale de l'action a dans le modèle. La valeur de cet ensemble correspond donc à celle du choix de cette action pour l'horizon considéré.

Une fois que l'ensemble U_a est construit pour chacune des actions de l'ensemble des influences I, on peut procéder à leur union. On purifie alors cet ensemble une nouvelle fois, de façon à éliminer les arbres qui sont dominés en chaque belief-state par au moins l'un des autres arbres de l'ensemble.

Lorsque cette opération est terminée, il ne reste dans l'ensemble que les arbres dont la valeur est optimale pour au moins l'un des belief-states. Ce dernier ensemble d'arbres correspond alors à la politique optimale, puisqu'il n'existe plus de point où l a valeur de l'un de ces arbres peut être améliorée, soit en modifiant l'action courante, soit en choisissant une autre politique au pas de temps suivant.

- **Support linéaire de Cheng**

Cet algorithme ressemble énormément à *Witness*. Il repose sur les mêmes principes, sauf pour ce qui correspond à la localisation des témoins. Cheng propose de les rechercher aux extrémités des belief-states possibles (14). En effet, cela permet d'éviter de prendre en compte tous les points de l'ensemble, mais de se rabattre sur un nombre fini de 'coins'. Le problème de cet algorithme vient du nombre de ces coins.

En fait, alors que pour un belief-state de petite taille, on peut déterminer ces points de manière géométrique, très efficace, leur nombre augmente de façon exponentielle avec la dimension de l'espace. D'autre part, au-delà de 3 ou 4 dimensions, les algorithmes géométriques deviennent inapplicables. Il devient alors nécessaire de faire des recherches coûteuses pour les localiser.

En définitive, cet algorithme présente donc assez peu d'améliorations par rapport à *Witness*. Selon les cas envisagés, l'un ou l'autre des algorithmes peut se montrer intéressant, mais aucun n'est supérieur pour toutes les situations possibles.

5.2.1.5 Champs d'application

Les modèles partiellement observables sont majoritairement utilisés en robotique mobile, ainsi qu'en reconnaissance de la parole.

Au niveau de la reconnaissance de la parole, l'intérêt de cette modélisation repose sur sa capacité à représenter des processus temporels ; on utilise donc ces modèles pour implanter l'évolution des sons qui est caractéristique de chaque phonème, ou de chaque groupe de phonèmes. La couche la plus basse d'un système de reconnaissance est donc classiquement basée sur un ensemble de chaînes de Markov partiellement observables : ces dernières analysent en parallèle les signaux sonores reçus par le système.

Bien sûr, une phase de prétraitement de ces signaux est nécessaire, de manière à réduire le volume de ces données. En effet, classiquement, plusieurs milliers d'échantillons par seconde sont utilisés pour obtenir une mesure suffisamment correcte de l'onde sonore. En même temps, ce traitement préalable permet l'extraction d'informations utiles, en mettant en évidence, par exemple, les paramètres fondamentaux du signal. Ainsi, ce prétraitement permet de simplifier les modèles de reconnaissance tant au niveau des observations qu'au niveau de leur structure.

Finalement, chacun des modèles de phonème donne une probabilité qui représente l'adéquation de ce modèle à l'onde reçue. Le modèle le plus probable correspond alors au phonème reconnu. Chacun de ces modèles est donc très simple, puisqu'il ne modélise qu'un seul phonème, sans relation avec les autres. L'architecture consacrée est un modèle gauche–droite à 4 ou 5 états. Dans ce type de modèle, les transitions ne permettent que d'avancer vers les états suivants ou de rester dans l'état courant ; aucune transition ne permet de revenir à un état antérieur. La Figure 15 illustre cette structure.

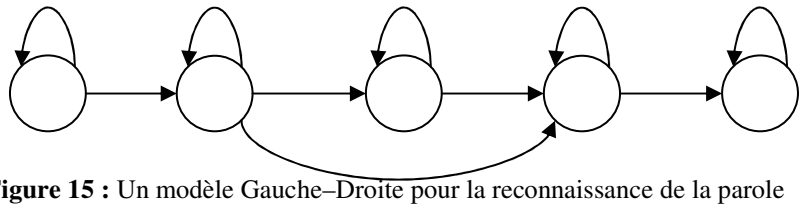


Figure 15 : Un modèle Gauche–Droite pour la reconnaissance de la parole

En robotique, la classe de modèles utilisée se rapproche plus du POMDP que du HMM. En effet, les problèmes étudiés portent principalement sur la planification (32), (3) et sur la cartographie (67). Dans ces deux problématiques, le logiciel agit directement sur le robot pour faire évoluer le processus, au contraire de la reconnaissance de la parole où l'on écoute sans intervenir. En particulier, pour la planification, il est important de connaître l'effet attendu de telle ou telle action afin de pouvoir choisir laquelle il faut mettre en œuvre.

De plus, la structure du modèle généralement utilisé est très différente : chaque état correspond à une position physique du robot dans l'environnement. Il est donc relié par des transitions aux états que le robot est capable d'atteindre en effectuant une action donnée en un pas de temps. Ce type de modèle est donc classiquement ergodique, sauf s'il existe des états absorbants. Un tel état pourrait être une cage d'escalier où le robot pourrait être détruit par une chute, par exemple. De même, la présence d'une porte que le robot ne pourrait pousser que dans un sens empêcherait son retour en sens inverse. Si c'est le seul chemin disponible, tous les états en amont de cette porte deviennent alors inatteignables une fois la porte passée.

Les observations du modèle correspondent à la lecture des capteurs embarqués sur le robot, voire même de capteurs disséminés dans l'environnement. On peut donc constater que chaque état est doté d'une sémantique propre qui le distingue des autres. L'observation peut permettre de distinguer les états les uns des autres, mais il est beaucoup plus probable que certains de ces états soient similaires au niveau de leurs perceptions. Ainsi, deux positions différentes le long d'un couloir donneront certainement des lectures similaires pour tous les capteurs de proximité.

C'est là qu'intervient l'aspect temporel du modèle , car une mémoire du passé permet souvent de séparer ces états jumeaux. En effet, le principe de localité ne permet au robot que de se déplacer d'une zone à une zone connexe.

La planification peut ensuite être utilisée pour atteindre un but fixé, en se basant sur le belief-state actuel et sur les perceptions à venir. Ainsi, un robot–facteur pourrait être chargé de se rendre dans un bureau précis pour y remettre un paquet. Il doit donc calculer une politique lui permettant d'atteindre ce point en partant de sa position actuelle. Le long de ce chemin, ses perceptions lui permettront de confirmer sa position ou, au contraire, de la remettre en question. Cela permet donc de réduire l'impact de l'incertitude du système.

5.2.2 Application à DIATELIC

Afin d'atteindre l'objectif de diagnostic que j'ai présenté dans le premier chapitre, nous avons choisi d'utiliser un modèle de décision Markovien partiellement observable (POMDP). Ce choix est justifié par le fait que le patient peut être considéré comme un processus qui évolue de manière continue. C'est ainsi que l'apparition de certaines pathologies peut se combiner à d'autres, soit séquentiellement, soit simultanément. La transition entre deux états physiologiques du patient est donc progressive. Cette progression m'a dissuadé d'utiliser de multiples chaînes, comme en reconnaissance de la parole. Cela séparerait en effet l'évolution du patient en de multiples processus indépendants, ce qui est incompatible avec la réalité.

Je pense au contraire que l'état du patient est atomique. Séparer le diagnostic pathologie par pathologie ferait donc perdre des informations précieuses. En effet, outre le fait que de multiples pathologies peuvent apparaître simultanément, l'évolution d'une pathologie à une autre est différente de l'évolution d'un état de bonne santé relative à un état pathologique. Un système unique est donc, je pense, un modèle mieux adapté au suivi d'un patient.

Les problèmes qui ont motivé cette étude sont principalement basés sur des troubles de l'hydratation. En effet, comme le rein ne régule plus la quantité d'eau présente dans l'organisme, celle-ci peut évoluer rapidement. C'est donc une des causes les plus importantes des accidents survenant pendant une DPCA.

Comme je l'ai indiqué dans le chapitre sur le modèle de perception, le poids est un facteur révélateur particulièrement intéressant. Il est comparé quotidiennement à une valeur fixée par les médecins, le poids-sec. Le problème vient du fait que la morphologie du patient évolue au cours du temps. Cette valeur est donc condamnée à évoluer pour suivre ces modifications imprévisibles. Nous avons donc dédié une partie du système à l'étude de ces divergences. Deux types de pathologies sont détectés par le système :

- Les troubles de l'hydratation
- Les dérèglements du poids-sec

Pour permettre la construction d'un modèle simple et compréhensible par les médecins, nous avons choisi de classer chacun de ces deux axes selon trois grandes valeurs, ce qui nous donne en définitive un total de 9 états. La Figure 16 montre la structure de ce modèle. Dans chacun des états, représentés par des cercles, deux symboles rappellent les composantes qui sont à son origine : le premier correspond au niveau d'hydratation du patient, et le second rappelle l'évaluation du poids sec. Chacun de ces symboles prend trois valeurs, selon que la valeur évaluée est correcte (« = »), trop faible (« - »), ou excessive (« + »). Bien entendu, ces jugements qualitatifs se basent tous sur l'état de bonne santé du patient (« =, = »). Cette notation sera utilisée à nouveau dans la Table 5 et la Table 6, afin de leur associer des symptômes déductibles des observations médicales.

Maintenant que la structure du modèle est définie, il reste à y insérer les observations pour obtenir un HMM bien formé. Comme je l'ai présenté dans la section portant sur le modèle de perception, nous avons choisi de mettre en place un système semi–discret. Les valeurs observées sont continues, mais elles sont projetées sur une série discrète d'intervalles flous. Cette manipulation permet, je le rappelle, de réduire la complexité du modèle tout en restant expressif aux yeux du personnel médical. Il reste donc à prendre en compte ce modèle de capteurs dans notre HMM.

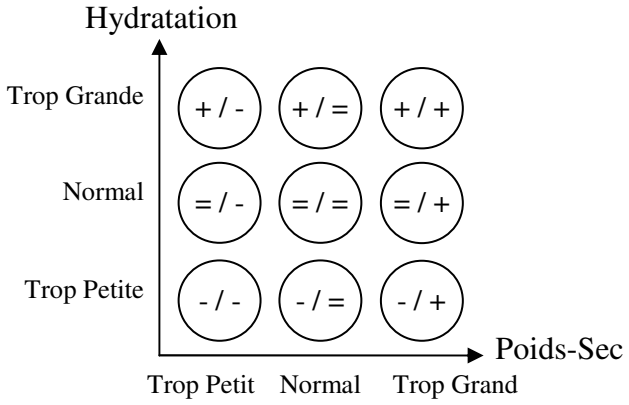


Figure 16 : DIATELIC : 2 pathologies, 3 grades, 9 états

L'approche consistant à choisir la valeur la plus probable de chaque capteur, en oubliant volontairement les autres, n'est pas satisfaisante. En effet, on retrouve un effet de seuil qui nous fait perdre une bonne partie de l'information disponible. La seule différence avec des intervalles stricts tient alors à la valeur du seuil séparant les deux intervalles. Cette approche, utilisée dans l'étude des systèmes à base de règles (chapitre 4), a clairement montré ses limites. Il nous faut donc rechercher des solutions mieux adaptées.

Dans leurs travaux sur la robotique mobile (34), Koenig et Simmons ont proposé une méthode alternative permettant de gérer l'observation de plusieurs capteurs incertains, à valeurs discrètes. L'intérêt de cette méthode tient à la prise en compte de l'incertitude des capteurs. De plus, en supposant que les capteurs sont statistiquement indépendants pour chaque état, le nombre de paramètres nécessaire se réduit au produit du nombre de valeurs (tous capteurs confondus) par le nombre d'états du modèle.

$$P(O|s) = \prod_{c \in CAPTEURS} P\left(c(O) \mid s\right)$$

$$P\left(c(O)|s\right) = \sum_{v \in VALEURS(c)} certitude\left(c(O)=v\right).P(c \rightarrow v| s) \qquad \forall c \in CAPTEURS$$

Afin de rassembler dans un seul modèle les avantages de cette méthode et ceux des filtres flous que j'ai exposés dans le chapitre sur les modèles de perception, nous avons utilisé un modèle hybride : le modèle d'agrégation de Koenig et Simmons est alimenté par la valeur de nos capteurs flous. En effet, on peut interpréter la valeur de la fonction floue comme la probabilité que la valeur observée appartienne à chacun des intervalles. Cette procédure nous donne alors un capteur à valeurs discrètes, mais dont les résultats sont relativement incertains.

Cette approche nous permet en particulier de combiner l'utilisation de capteurs continus et de capteurs à valeurs discrètes dans un même modèle. En effet, pour les capteurs continus, la probabilité que l'un de ces capteurs émette l'une de ses valeurs est donnée par nos filtres flous ; alternativement, la probabilité d'un capteur à valeur discrète est soit donnée directement par le capteur, soit supposée égale à 100% si la fiabilité du capteur est considérée comme certaine. Ensuite, cette probabilité est insérée dans la formule d'agrégation présentée ci-dessus sans modification additionnelle.

De plus, si l'un des capteurs est indisponible, il suffit d'injecter dans le modèle une répartition uniforme à la place de ses résultats. Cette opération annule purement et simplement son influence dans le calcul. En effet, si l'on suppose que l'observation d'un capteur est régie par une distribution de probabilités par état, et si le capteur renvoie une distribution uniforme, la formule se simplifie de façon à modifier la probabilité d'observation de chacun des états de la même façon. Ainsi, les probabilités relatives des différents états restent cohérentes, ce qui permet d'utiliser la même formule quelle que soit la disponibilité des différents capteurs.

$$P\left(c(O)|s\right) = \sum_{v \in VALEURS(c)} \frac{P(c \rightarrow v| s)}{card\left(VALEURS(c)\right)}$$

$$= \frac{1}{card\left(VALEURS(c)\right)} \sum_{v \in VALEURS(c)} P(c \rightarrow v| s) = \frac{1}{card\left(VALEURS(c)\right)}$$

Capteur / Pathologie	Poids	Tension	Hypotension Orthostatique	Ultrafiltration
Hyperhydratation (+ / =)	Elevé	Elevée	Normale +	Excédentaire
Déshydratation (- / =)	Bas	Basse	Inversée	Déficitaire
Poids-Sec trop haut (= / +)	Bas	Normale	Normale	Normale
Poids-Sec trop bas (= / -)	Elevé	Normale	Normale	Normale

Table 5 : Description de l' influence des troubles de la dialyse isolés les uns des autres

Capteur / Etat	Poids	Tension	Hypotension Orthostatique	Ultrafiltration
+ / +	Elevé&Bas	Normale +	Normale +	Normale +
+ / -	Elevé	Normale +	Normale +	Normale +
- / +	Bas	Normale -	Normale -	Normale -
- / -	Elevé&Bas	Normale -	Normale -	Normale -

Table 6 : Description de l' influence des troubles combinés de la dialyse

Il nous reste donc finalement à exprimer la probabilité de chaque capteur dans chaque état. Comme chacun des états est défini avec une sémantique médicale forte, et que le modèle ne requiert que des liens de causalité directs, ces probabilités peuvent être fixées assez facilement. Il nous suffit d'y exprimer les effets attendus des diverses pathologies sur les signaux médicaux.

D'après les médecins de l'ALTIR, les troubles de l'hydratation et du poids-sec ont un impact sur les valeurs des 4 paramètres médicaux tel qu'il est décrit dans la Table 5. Dans cette table, les parenthèses de la première colonne correspondent à la description de l'état correspondant de la Figure 16. La mention *Normale+* correspond à une valeur normale, mais pouvant accepter une légère dérive vers les valeurs trop grandes.

Il nous faut maintenant définir les observations qui peuvent trahir la présence conjointe de deux troubles simultanés. En réalisant l'union des symptômes des deux pathologies isolées, la description de nos quatre états additionnels se traduit par les observations de la Table 6. Dans cette table, les mentions *Normale+* et *Normale-* correspondent aux valeurs normales pour ce capteur, mais avec une tolérance quant à leurs dérives possibles. Les deux cases affichant la mention *Elevé&Bas* pour le poids montrent que deux tendances opposées se cumulent. L'hydratation tend à faire évoluer le poids réel dans une direction, alors que le poids sec tend à inverser la perception de cette évolution. Selon l'importance relative de ces deux dérives, toutes les valeurs peuvent donc être observées.

De plus, sur cette table, on peut voir apparaître un nouveau problème : les états combinant ces deux pathologies peuvent être confondus deux à deux, mais aussi avec les problèmes de poids-sec seuls. En effet, ce paramètre n'a rien à voir avec l'état de santé du patient ; il permet juste de juger de la dérive de la valeur du poids réel. Il peut donc très facilement faire croire à un désordre de l'hydratation, tout comme il peut le masquer, simplement en faussant l'appréciation du poids par le système.

En définitive, nous avons donc décidé de ne retenir que 5 états sur les 9 : un état représente l'état normal de bonne santé du patient, et les 4 autres figurent les 4 pathologies, isolées les unes des autres. En effet, ajouter les états diagonaux n'ajouterait que peu d'information, tout en augmentant la complexité du modèle. La Table 6 montre bien que les influences combinées de ces deux pathologies ne sont pas suffisantes pour les isoler des pathologies séparées. Dans la moitié des cas, il n'est même pas possible de leur donner une description.

De plus, si les symptômes sont ambigus et ne permettent pas de choisir un état particulier, le calcul du belief-state devrait le faire apparaître lors de l'agrégation des différents capteurs. Les tests que nous avons faits montrent que c'est effectivement le cas. Lorsque le poids augmente alors que la tension chute, le système indique un problème de poids sec. Si le poids augmente et que la tension reste stable, il se montre plus indécis. Le belief-state se répartit grossièrement entre 30% et 50% pour ces deux états, bien que l'un d'eux domine souvent.

Finalement, l'étude de notre corpus étiqueté par les médecins nous a permis de définir un modèle moyen, correspondant à un patient typique. La Figure 17 présente les probabilités d'observation correspondant à ce profil-type. Elles sont représentées sous forme graphique, forme sous laquelle les médecins interagissent avec le modèle. Cet affichage met donc en évidence l'influence des différentes valeurs possibles de chaque état (les colonnes) sur chacun des capteurs (les lignes).

Chacune de ces courbes est centrée sur la valeur *normale* pour le patient. Plus on s'écarte du centre vers la gauche, et plus la valeur analysée diminue. En ordonnée, la courbe montre la probabilité d'observer cette valeur particulière dans l'état correspondant à sa colonne. En effet, chaque courbe est la somme des trois fonctions floues que j'ai présentées dans le chapitre sur le modèle de perception. Ces trois courbes sont pondérées par la probabilité d'observer les trois valeurs floues correspondantes dans l'état considéré.

En particulier, l'amplitude relative des courbes montre l'importance d'un capteur dans le processus du diagnostic. Par exemple, on peut voir que la différence des deux tensions, prises lorsque le patient est couché et lorsqu'il se lève, a une forte influence sur l'état de déshydratation, mais une influence toute relative sur l'état d'hyperhydratation. De plus, cette constatation permet aux médecins d'annuler l'importance d'un capteur donné pour un certain patient, simplement en faisant de la courbe correspondante une droite horizontale. Cette configuration correspond en effet à une distribution de probabilités uniforme, ce qui indique que chacune des valeurs de ce capteur a les mêmes chances d'être observée que les autres.

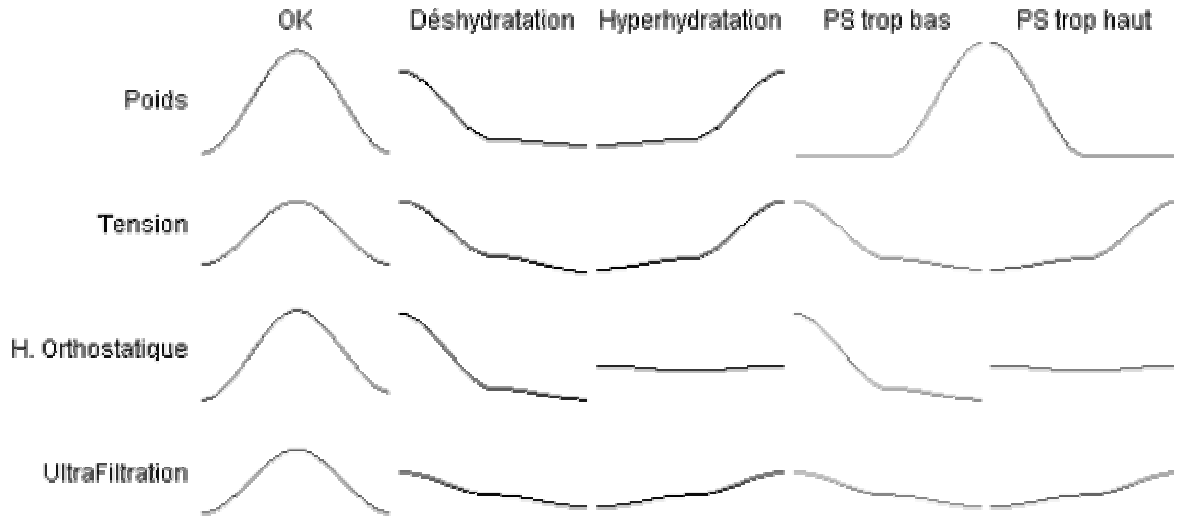


Figure 17 : Les perceptions de DIATELIC

Pour obtenir un modèle de POMDP complet, il nous reste encore à définir la notion d'action. C'est là que le problème se complique ; en effet, les actions devraient correspondre aux influences connues que l'on peut exercer sur le système, à savoir le patient. Or, ces influences sont légion.

On peut recenser par exemple les diverses combinaisons de poches que le patient utilise, ainsi que les médicaments qu'il prend. On pourrait aussi ajouter le type de régime alimentaire auquel le patient est contraint, car cela modifie considérablement la quantité d'eau que le patient ingère quotidiennement. Il faudrait bien sûr être capable de combiner ces diverses actions, car les patients sont souvent soumis simultanément à une dialyse, un traitement médical, et un régime. La diversité de ces actions est alors telle qu'il faudrait un nombre considérable de données étiquetées pour pouvoir apprendre leurs caractéristiques, et pour pouvoir ensuite les valider expérimentalement.

Cependant, à l'époque où l'expérimentation a été lancée sur 30 patients, les données dont nous disposions pour valider le système se résumaient aux données étiquetées de 60 patients. Selon les patients, nous avions entre 5 et 300 jours de données, soigneusement étiquetés par les néphrologues de l'ALTIR. Cet étiquetage correspond à un diagnostic médical certain ; il ne s'applique donc pas à tous les jours de données. Finalement, une fois retirés tous les patients pour lesquels les données étaient insuffisantes pour avoir un profil valable, nous avions encore 5 ou 6 patients dont les données s'échelonnent entre 30 et 300 jours.

Nous avons donc décidé de trancher, et nous n'avons finalement conservé que deux actions. La première consiste à observer l'évolution du patient sans agir sur le traitement, et la seconde permet de prendre en compte les modifications du modèle par le médecin. En particulier, la modification du poids-sec entraîne des changements radicaux dans la perception du poids. Or ce capteur a une influence très importante sur le diagnostic ; nous avons donc opté pour l'ajout de cette action, dont la matrice de transition est uniforme. Cela nous permet ainsi de gommer l'impact du diagnostic des jours précédents. En effet, puisque les paramètres étaient biaisés par la mauvaise estimation du poids, le diagnostic n'était pas réellement fiable.

L'action consistant à observer permet de ne pas influencer le système avec des *a priori* sur son évolution. Sa matrice de transition doit donc tenir compte de la faible connaissance que nous avons sur l'évolution future du patient et l'inertie nécessaire pour assurer la continuité temporelle. Cela se traduit donc par une matrice très homogène, où seuls les coefficients diagonaux se singularisent. En effet, ces coefficients représentent la probabilité que le patient reste dans le même état. Ils incarnent donc toute l'inertie du système. L'étude sur les données de nos patients nous a permis de définir une matrice moyenne. Cette dernière est présentée sur la Table 7.

Avec ces deux actions, *observer* et *changer le modèle*, il n'est pas question d'envisager une quelconque planification. Cependant, avec les données rassemblées pendant l'expérimentation (présentée en annexe), nous espérons pouvoir y remédier. Cela représente cependant une longue période d'apprentissage et de tests nécessaires avant de pouvoir mettre en œuvre cette nouvelle version.

70%	7,5%	7,5%	7,5%	7,5%
7,5%	70%	7,5%	7,5%	7,5%
7,5%	7,5%	70%	7,5%	7,5%
7,5%	7,5%	7,5%	70%	7,5%
7,5%	7,5%	7,5%	7,5%	70%

Table 7 : La matrice de transition de l'action « Observer la dialyse »

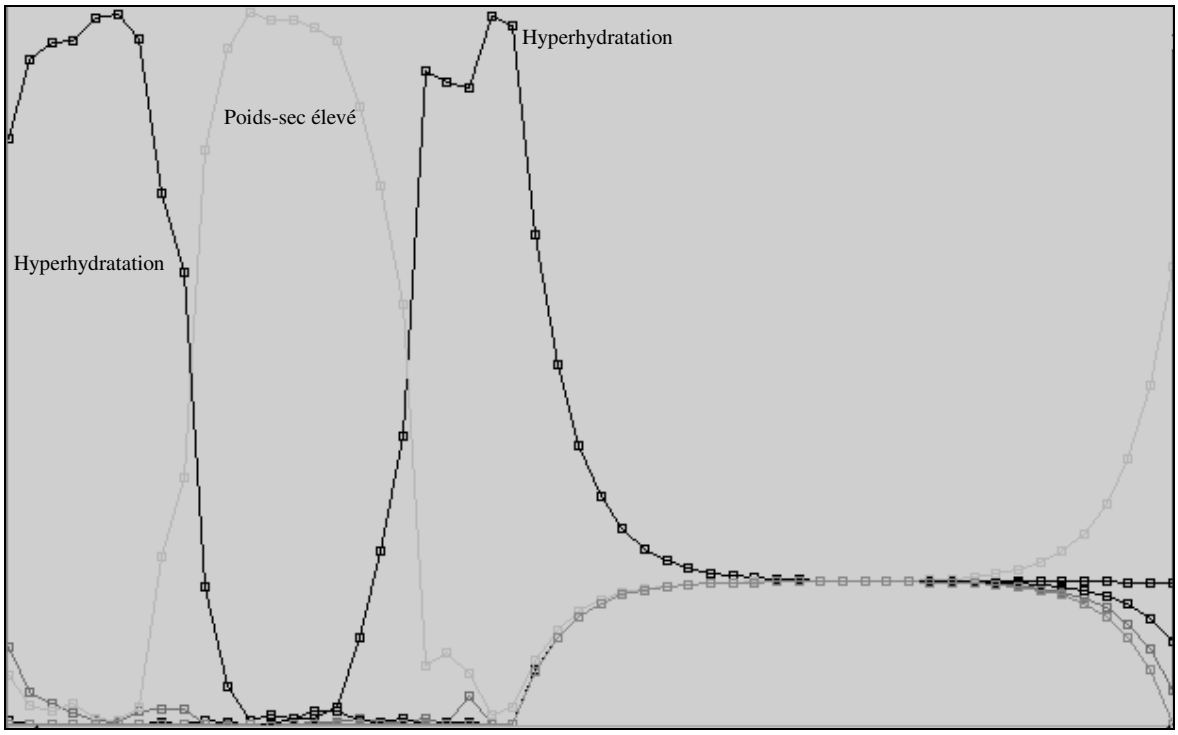


Figure 18 : Un exemple de diagnostic fourni par DIATELIC

Finalement, puisqu'il n'est pas question de faire de la recommandation d'actions, faute d'une définition suffisante de ces dernières, il nous reste à exprimer le diagnostic du système afin d'aider le médecin à prendre les décisions qui s'imposent. La solution que nous avons choisie est simple : le système affiche simplement au médecin les courbes représentant l'évolution des probabilités de nos cinq états.

Comme ces états ont une signification médicale claire pour le médecin, les courbes sont explicites par elles-mêmes. Un exemple d'un tel diagnostic est présenté sur la Figure 18. Cette analyse porte sur un patient hyperhydraté tout au long des données présentées, et le profil utilisé est celui du patient moyen. On peut voir sur ce diagramme que deux courbes sont dominantes par alternance. La courbe qui domine initialement le diagramme correspond à l'état *Hyperhydratation* du système. La seconde correspond à l'état *Poids-Sec Trop Bas*. Ensuite, la première courbe redevient dominante pour un court laps de temps, avant de rejoindre les 4 autres dans une situation d'incertitude totale.

Cet état d'incertitude se comprend aisément lorsque l'on connaît l'histoire de ce patient : il est parti en vacances, et n'a donc pu envoyer aucune donnée pendant plusieurs semaines. Le système converge alors vers un point stable caractéristique de la matrice de transition, puisque aucune donnée ne vient modifier son évolution. Comme la matrice est homogène, le belief-state correspondant à cet état stable est un vecteur uniforme.

Les données présentées ici ont été testées d'abord sur deux patients pendant quelques mois, puis sur 15 autres patients pendant une période de trois ans. Durant cette période de tests, le système à base de règle, que j'ai présenté dans le chapitre portant sur l'utilisation des connaissances des experts, a été utilisé en parallèle avec celui-ci, afin de pouvoir comparer leurs résultats.

Le bilan de cette étude est sans équivoque. Le premier système est trop sensible aux variations locales des données, alors que le système markovien semble être plus robuste.

De même, au dire des médecins, l'intégration des différents paramètres semble plus fine, ce qui autorise un meilleur diagnostic. Enfin, le modèle est compréhensible par les médecins. Ceci est mis en évidence par le fait que les médecins soient capables de régler les paramètres du modèle sans aucune aide de notre part. Ainsi, l'étude clinique s'est déroulée avec très peu d'interventions de notre part. Ces dernières, surtout au début de l'expérience, se sont limitées à la correction de petites bogues dans l'interface et à l'analyse de situations que les médecins jugeaient pertinentes.

Ainsi, suite à une apparente guérison de problèmes cardiaques, le profil d'un patient avait été modifié. Ce profil contient en effet les probabilités d'observation des capteurs dans chaque état. Or, des problèmes cardiaques modifient la réponse de la tension en cas de variation de l'hydratation. Cette modification a entraîné, quelques jours plus tard, une hyperhydratation se traduisant par un œdème pulmonaire. L'analyse de ce problème nous a permis, après-coup, de montrer que les symptômes de cette hyperhydratation étaient discernables par le système, mais uniquement avec le profil d'une personne cardiaque. Nous avons donc pu conclure que le système était correct, mais que la guérison de l'insuffisance cardiaque de ce patient n'était pas suffisante pour rétablir une évolution normale de la tension artérielle. On peut donc justifier la notion de profil pour chaque patient : non seulement ce profil permet un diagnostic plus adapté, mais il est totalement nécessaire à la prise en charge de certaines pathologies.

5.2.3 Application à l'anesthésie

De façon similaire à la surveillance de la dialyse de type DPCA, nous avons choisi de traiter le problème de l'aide à l'anesthésie en utilisant un POMDP. Nous avons également utilisé un jeu d'états ayant une sémantique forte aux yeux du médecin.

Pour un patient anesthésié, les risques principaux sont le réveil, le coma, et les arrêts respiratoires ou cardiaques. J'ai déjà indiqué dans le chapitre présentant ce projet que l'hypnose devait être suffisante pour empêcher les souvenirs, qu'ils soient conscients ou inconscients. Avec les données disponibles (fournies, je le rappelle, d'une part par l'ASPECT, et d'autre part par l'ANEMON), les médecins anesthésistes de Brabois ne pensent pas qu'il soit possible de prédire un quelconque type d'arrêts.

De plus, la faible quantité de données disponibles nous empêche de procéder à une analyse de ces dernières en espérant y trouver un lien entre les arrêts réellement observés pendant une opération et les données précédant cet événement. Nous avons donc décidé de nous concentrer sur l'analyse de la profondeur de l'anesthésie. L'ASPECT étant encore une machine expérimentale, cette analyse permettra de conforter, ou, au contraire, de mettre en doute l'analyse de ce dispositif.

Comme je l'ai indiqué précédemment, cette profondeur est en lien direct avec la quantité de produits injectés, et avec les stimulations douloureuses que le patient reçoit. Nous retrouvons donc de nouveau une répartition en deux classes des processus à observer : la douleur, et la profondeur du sommeil. Cependant, il y a des différences importantes avec le projet DIATELIC : tout d'abord, il n'y a pas de niveau de douleur trop faible. Au contraire, moins le patient endure de douleur, mieux se déroule l'opération, tant au niveau du confort du patient qu'au niveau de la gestion de l'anesthésie. La seconde différence tient au fait que la douleur est une donnée physique, même si elle est encore mal comprise. Il n'y a donc pas de problème tel que celui posé par la relation entre le poids-sec et le poids. Notre espace d'états couvrira donc trois profondeurs d'anesthésie et la présence ou l'absence d'un niveau de douleur important. Six états seront donc nécessaires.

Cependant, les différences entre ces deux projets médicaux ne s'arrêtent pas là. Tout d'abord, l'aspect temporel est très différent : alors que pour DIATELIC, le même patient peut être suivi pendant plusieurs années, les gens ne se font anesthésier qu'assez rarement. On disposera donc de très peu de temps pour adapter le modèle au patient. De plus, dans le meilleur des cas, le système va recevoir deux données toutes les 5 secondes, alors que dans DIATELIC, le patient envoyait un paquet de données complet une fois par jour. Le temps disponible pour traiter l'un de ces paquets est donc beaucoup moins important dans la surveillance d'une anesthésie.

Finalement, les deux instruments dont nous disposons nous donnent déjà une première approximation de l'état que nous recherchons, alors qu'aucun capteur ne permet de mesurer l'hydratation d'un patient dans DIATELIC. Nous avons donc décidé de nous consacrer d'une part aux problèmes temporels, et d'autre part à l'adaptation du modèle au patient. En effet, si les patients suivent à peu près le même modèle de comportement, les valeurs numériques correspondantes sont très variables d'un patient à l'autre. Il est donc nécessaire de s'y adapter pour travailler de façon fiable. Ce problème relevant de l'apprentissage, il sera étudié dans le chapitre dédié à cette classe de problèmes.

Les aspects temporels du problème proviennent aussi bien du fait qu'il est impossible d'attendre une correction explicite des médecins durant l'opération, que de la non-synchronisation des capteurs. Ce dernier problème est particulièrement gênant, puisque chaque appareil a sa propre horloge indépendante. Même si on peut tenter de les synchroniser manuellement, cela reste très approximatif. Les différentes données peuvent donc arriver à des intervalles arbitraires, mais fixes. Néanmoins, des parasites peuvent aussi rendre certaines données illisibles, même si c'est assez rare ; un tel événement entraînera donc l'allongement temporaire de l'intervalle séparant deux données consécutives. Enfin, si l'on décide d'ajouter de nouveaux instruments, rien ne nous garantit que les intervalles entre les envois de données seront eux aussi réglables à cinq secondes. Il me paraît donc important de tenir compte de ce paramètre.

Pour résoudre ce problème, plusieurs approches sont envisageables. On peut naturellement discrétiser le temps en intervalles suffisamment petits pour assurer la précision dont nous avons besoin. Le problème est alors à deux facettes : on peut découper le temps soit en intervalles fins, soit en intervalles longs. Dans le premier cas, une grande précision est disponible, mais chaque donnée devra s'accompagner d'un grand nombre de mises à jour du belief-state, ce qui est coûteux en temps.

Si l'on calcule la matrice résultante avant de l'intégrer au calcul du belief-state, à chaque réception de données, la complexité de l'algorithme est :

$O(Nb_{Intervalles} \times Nb_{Etats}^{3})$.

Si l'on évalue le belief-state à chaque étape, en utilisant la procédure Forward pour chaque intervalle de temps, on peut limiter cette complexité à :

$O(Nb_{Intervalles} \times Nb_{Etats}^{2})$.

A cette valeur, il convient cependant d'ajouter la normalisation du résultat qui permet de réduire les erreurs de calcul faites par la machine, ce qui revient à autant de divisions par la somme des probabilités du belief-state que d'états. Cette normalisation est obligatoire, car les petites erreurs s'ajoutant, la somme des probabilités des différents états ne fait plus 100%. Pire encore, il arrive que certaines probabilités deviennent négatives au cours du calcul ! Dans ces deux cas, toute la suite des calculs s'effondre et les erreurs s'amplifient.

Ces deux options sont donc très coûteuses en temps. Or la machine responsable de ces calculs doit aussi assurer le pilotage des pousses–seringues responsables de l'injection des drogues. En particulier, il existe une sécurité qui veut que le délai de réponse à une requête de cet appareil soit très court. Si aucune réponse n'est reçue dans cet intervalle, les pousses–seringues cessent immédiatement toute injection de produit au patient. L'effet résultant d'un calcul trop long pourrait donc être désastreux pour le patient.

De plus, on peut se demander quelle serait la sémantique d'une matrice de transition qui exprimerait l'évolution de l'état du patient durant un dixième de seconde. Ses probabilités convergeraient vers la matrice identité, car le système ne pourrait avoir beaucoup évolué dans cet intervalle de temps. Dans ces conditions, il devient difficile de définir la matrice, et encore plus sa sémantique.

Il reste donc à évaluer le découpage du temps en intervalles importants. Ce n'est malheureusement pas une solution satisfaisante, car on perdrait beaucoup en précision pour ce qui concerne le suivi du processus. A ce moment, autant considérer que les différents appareils sont synchronisés, et n'utiliser ces données que lorsque tous les capteurs ont envoyé leurs résultats. Dans certains cas, cela reviendrait à prendre en compte des données qui ne sont peut-être plus valides. Il n'est donc pas recommandable d'utiliser un tel découpage.

Dans la littérature, on parle beaucoup des modèles semi–markoviens (8). Ces modèles permettent en effet d'associer aux différentes actions des temps d'exécution variables. Dans ce cas, il est possible d'optimiser des valeurs telles que le temps nécessaire pour atteindre un but. Malheureusement, cela ne correspond pas à notre problème, puisque nous ignorons la nature exacte des actions, comme nous le verrons un peu plus tard. Par extension, on ne peut donc pas donner de temps nécessaire à leur accomplissement. De plus, nos actions ont un effet continu, et le temps considéré ne correspond pas à l'action, mais à l'observation.

Nous avons donc envisagé l'utilisation d'équations dynamiques régissant l'évolution du système. En particulier, les équations différentielles semblent relativement prometteuses, comme le montre (28). En effet, ces équations permettent de modéliser l'évolution d'un système dynamique de façon directe. Leur intégration dans le processus de localisation permettrait donc une prise en compte optimale du temps. Cependant, étant assez peu familier avec ce formalisme et les mathématiques qui le supportent, j'ai préféré étudier une solution alternative.

Cette méthode repose sur l'adaptation de la matrice de transition au temps dévolu à celle-ci. Notre matrice de transition est donc associée à un temps de référence auquel elle correspond. La matrice réellement utilisée sera alors la résultante de l'élévation à une puissance fractionnaire de la matrice de transition :

$$A(t) = A^{\,t / t_{réf}}$$

Le problème, c'est que ce type d'opération n'est pas défini proprement dès que la puissance est fractionnaire. Je vais donc présenter le calcul que nous utilisons pour calculer ce résultat. La méthode se base sur un algorithme permettant de calculer rapidement des puissances arbitraires d'une même matrice : pour y parvenir, on procède tout d'abord à une diagonalisation de la matrice initiale. Cette transformation est bien connue, et je ne la répéterai pas ici. Pour plus de détails sur cette diagonalisation, je vous renvoie au chapitre 11 de Numerical Recipes (54).

On procède donc à un changement de base, représenté par une matrice inversible Q, qui permet d'exprimer notre matrice initiale sous la forme d'une matrice diagonale D.

On obtient donc la formulation suivante :

$A = Q \,.\, D \,.\, Q^{-1}$

D'où finalement,

$$\begin{aligned} A^n &= A \,.\, A \,.\, \ldots \,.\, A \\ &= Q \,.\, D \,.\, Q^{-1} \,.\, Q \,.\, D \,.\, Q^{-1} \,.\, \ldots \,.\, Q \,.\, D \,.\, Q^{-1} \\ &= Q \,.\, D \,.\, D \,.\, \ldots \, D \,.\, Q^{-1} \\ &= Q \,.\, D^{\,n} \,.\, Q^{-1} \end{aligned}$$

Or, comme D est diagonale, son élévation à la puissance *n* se réduit au calcul des puissances de ses coefficients diagonaux :

$$D^{\,n} = \begin{bmatrix} D_1^n & 0 & \ldots & 0 & 0 \\ 0 & D_2^n & 0 & \ldots & 0 \\ 0 & 0 & & & \vdots \\ \vdots & \vdots & & D_{k-1}^n & 0 \\ 0 & 0 & 0 & \ldots & D_k^n \end{bmatrix}$$

On peut donc facilement généraliser ce calcul aux puissances fractionnaires, à la condition toutefois que les coefficients diagonaux soient tous positifs ou nuls. Dans le cas contraire, la puissance fractionnaire d'un tel coefficient serait un complexe, et non un réel. La matrice perdrait alors sa sémantique de matrice de transition. J'ajouterai enfin que le calcul de la diagonalisation peut être fait une fois pour toutes pour chaque action. Il n'est donc pas nécessaire de la refaire à chaque réception d'une observation. A chaque étape, il suffira donc de calculer chaque coefficient de la diagonale avant de multiplier la matrice obtenue par les matrices de changement de base.

Néanmoins, pour que cette solution fonctionne, il est nécessaire de rassembler plusieurs conditions. Tout d'abord, la matrice doit être diagonalisable. Ensuite, chacune de ses valeurs propres doit être positive ou nulle. On ne peut donc pas appliquer cette méthode dans tous les cas. Pour tempérer cette remarque, on peut signaler que le nombre de matrices qui ne conviennent pas est un infini assez petit. Il y a donc presque toujours une matrice diagonalisable proche de celle que l'on veut représenter.

Nos tests de cette méthode se montrent très concluants : la combinaison de deux transitions est bien équivalente à la transition associée à la somme des temps des deux transitions envisagées. Cela permet aussi d'exprimer la matrice simplement, puisqu'elle correspond à une période relativement longue de 5 secondes. Néanmoins, rien n'empêche d'allonger cette période de référence pour prendre en compte une évolution plus lente ; l'inverse est également valable pour gérer une évolution plus rapide. Finalement, peu de problèmes ont été rencontrés suite à des matrices non diagonalisables. C'est donc une solution qui semble tout à fait viable.

De même que pour le projet DIATELIC, le problème le plus difficile à traiter dans la modélisation du processus tient à la notion d'action. Dans une anesthésie, le médecin peut agir sur la conduite de l'opération par le biais de l'injection de plusieurs drogues. Parmi celles-ci, un hypnotique permet de provoquer et de maintenir la perte de conscience du patient. Simultanément, l'injection d'un analgésique permet de limiter les effets des stimulations douloureuses qu'exerce le chirurgien lorsqu'il opère. Finalement, un dérivé de curare permet de faire perdre au patient son tonus musculaire, de façon à rendre l'opération plus accessible au chirurgien. Ce dernier produit a très peu de lien avec la douleur ou l'hypnose du patient ; nous ne le considérerons donc pas dans notre raisonnement.

Tout le problème vient de la gestion de ces produits : leur influence est doublement continue. Tout d'abord, plus la dose injectée est importante, plus l'effet attendu est important. Or, cette dose est obtenue par réglage d'une pompe qui envoie en permanence une certaine quantité de produits dans le flux sanguin. Ce débit peut être réglé par l'envoi d'une commande numérique, et donc discrète. Cependant, tellement de valeurs sont possibles qu'il peut être intéressant de les exprimer sous la forme d'une seule commande continue afin de simplifier le modèle. De plus, ces commandes sont fortement liées les unes aux autres quant à leurs effets. Il serait donc difficile de les traiter séparément, tout en conservant ce lien.

La seconde facette du problème de l'injection est le temps : lorsque l'on modifie le débit d'une pompe, on modifie la quantité de produit injectée dans le flux sanguin. Or, seule la concentration de ce produit au site d'effet, dans le cerveau, a une influence sur l'anesthésie. Le problème vient du fait que le transfert du flux sanguin au site d'effet prend du temps. Nous avons donc un délai entre l'expression de la consigne et l'apparition de son effet. Ensuite, l'effet des drogues n'est pas instantané. Donc une modification de la concentration au site d'effet va entraîner une modification progressive de l'état du patient.

La troisième et dernière facette de la gestion des actions tient à la variabilité des patients : l'effet des drogues peut être très différent selon ces derniers. Cela rend donc difficile la création d'un modèle typique, qui serait valable pour la majorité des patients.

Pour tenir compte de ces différentes contraintes, il est nécessaire d'utiliser un modèle complexe et de disposer d'une grande quantité de données permettant de valider ce modèle. Actuellement, ces données ne sont pas disponibles. Nous sommes donc fortement limités dans l'étude de ces actions. Deux approches sont envisagées, mais aucune n'a pu être testée en situation. La première solution consisterait à utiliser une matrice de transition par produit. Une combinaison astucieuse de ces matrices, en fonction de leurs doses respectives, devrait alors permettre d'obtenir une action adaptée à la situation. Pour être plus précis, je pense que la concentration au site d'effet serait plus utile que la dose injectée, ou que le débit de la pompe correspondante. Nous avons heureusement des modèles pharmacocinétiques basés sur des modèles physiques de diffusion qui permet de faire ce calcul. Ces modèles ont été utilisés par les anesthésistes de Brabois depuis plusieurs années ; ils sont donc tout à fait dignes de confiance. Il faudrait donc intégrer ces modèles en amont du calcul de la matrice de transition, et cela à chaque instant. L'implantation de ces modèles sur machine existe, et est disponible sur le site de Stanford (63).

La seconde méthode que nous envisageons vise à l'utilisation d'un jeu d'équations différentielles qui exprimeraient l'effet de chaque produit en continu. La résolution de ce système permettrait alors d'obtenir un modèle fin de l'influence conjuguée des produits. Néanmoins, cela nécessite tout d'abord d'établir ces équations, puis de résoudre le système avant d'intégrer son résultat dans le calcul du belief-state. Or ces équations dépendent de chaque produit, ainsi que des caractéristiques du patient (son poids, son âge, ses antécédents, …). Il faudrait donc refaire une bonne partie du travail qui se trouve à la base des modèles pharmacocinétiques. Néanmoins, cette solution permettrait d'obtenir une bonne assise théorique, et éliminerait le besoin d'une formule magique permettant le mélange des différentes matrices.

Actuellement, nous sommes donc dans une impasse pour l'étude de ce projet. Nous attendons impatiemment des données suffisantes pour pouvoir mieux étudier ces différentes approches. Dans l'intervalle, nous sommes réduits à l'observation de l'anesthésie sans prise en compte des produits injectés. La matrice d'observation du système est donc primordiale, puisque la matrice de transition se contente d'assurer une certaine inertie au modèle.

Cela limite donc quelque peu l'intérêt du système, puisqu'il se borne à résumer les informations des deux capteurs, l'ASPECT et l'ANEMON. Néanmoins, je pense que l'inclusion des autres données fournies par ces capteurs, voire d'autres capteurs pourrait fortement augmenter son utilité.

5.2.4 Application à la robotique

L'application à la robotique mobile de notre modèle nous remet en phase avec les études menées de par le monde ((3),(19),(32),(67),…) . Ce modèle étant issu en grande partie de la robotique, il est juste qu'il permette en retour d'améliorer ce domaine de r echerche. Deux principaux avantages distinguent ce domaine de ses homologues médicaux : la vie d'aucun patient ne dépend des résultats du système, et les données sont abondantes et facile à récupérer.

Pour la navigation du robot, un modèle s'impose de lui-même : le POMDP. La notion d'état est relativement facile à définir, puisqu'elle correspond à une situation du robot dans l'environnement. Cela inclue donc aussi bien sa position que son orientation ou la présence d'objets dans l'environnement. Ces considé rations contribuent donc à la création d'un nombre considérable d'états, si l'on veut disposer d'une précision suffisante. L'objectif de ce projet consiste donc à montrer que l'on peut réduire efficacement cette quantité en recourant à notre modèle de perceptions floues.

Notre robot dispose de deux classes de capteurs, une caméra et un jeu de capteurs de proximité infrarouges. La prise en compte de la caméra constitue l'un des apports d e la thèse de Franck Gechter avec qui j'ai travaillé sur le robot . Je me contenterai donc d'en rappeler les grandes lignes. Les infrarouges seront traités par un jeu de filtres flous de modèles similaires à ceux utilisés dans nos projets médicaux.

L'avantage et l'inconvénient de la caméra tient à la richesse de l'information q u'elle fournit à chaque image. Même avec la faible résolution que nous utilisons, 256x256 pixels, le nombre de paramètres explose rapidement. Si l'on se base sur des paramètres standard de couleur, cela nous donne 16 millions de valeurs possibles pour chaque pixel. Si l'on se réduit à une image en niveaux de gris, il reste encore 256 possibilités pour chacun d'entre eux. L'idée proposée dans (21) consiste donc à utiliser une analyse en composantes principales (ACP) pour réduire la taille de cet espace.

Grâce à cette analyse, une image donnée peut être approchée par la combinaison linéaire d'un certain nombre d'images de référence. Le résultat de l'ACP peut donc être vu comme un capteur donnant un vecteur de poids réels, de taille raisonnable. Un état se caractérisera alors par un vecteur typique. Le vecteur observé à un instant donné pourra donc être comparé au vecteur de chaque état. Cette mesure de distance, après normalisation, peut être vue comme une probabilité de correspondance de l'image au vecteur de référence de chaque état. Cela permet alors sa prise en compte dans la formule d'agrégation des différents capteurs , au même titre que n'importe quelle autre source de données .

Les capteurs infrarouges donnent quant à eux une mesure de distance comprise entre 10 et 80 centimètres. Je propose donc de partager cette mesure en trois valeurs floues : *proche*, *à distance*, et *loin*. La première de ces valeurs représentera donc une proximité immédiate, dont le robot doit absolument tenir compte. Le dernier de ces intervalles correspondra soit à un obstacle détecté à longue distance, soit à l'absence totale de détection d'un quelconque obstacle. En effet, les capteurs que nous utilisons ont l'inconvénient de donner systématiquement une distance, même si rien n'a été détecté. Il est donc bon de tenir compte de cette imprécision.

La notion d'action pour le robot peut être vue sous plusieurs aspects. On peut ainsi restreindre le robot à des actions discrètes, telles que *avancer d'un mètre*, ou *tourner de 30 degrés*. On peut également considérer que le robot est piloté en vitesse, puisqu'il est basé sur un châssis inspiré de celui des tanks. Les commandes seraient alors telles que *vitesse gauche à 10 cm par seconde, vitesse droite à 5 cm par seconde*. Ces commandes peuvent alors être discrétisées, ou considérées comme continues. Cette approche permet de nous rapprocher des problèmes que nous rencontrons dans la surveillance d'une anesthésie. La différence tient alors au fait que les équations de la dynamique du mouvement sont bien connues. La résolution des problèmes que posent ces actions sur le problème de la robotique nous donnera donc de précieuses indications quant à la résolution des problèmes équivalents dans les domaines de la dialyse ou de la surveillance d'anesthésie.

Les dernières expériences que j'ai menées dans le cadre de la robotique ont dévoilé une nouvelle dimension des problèmes de discrétisation. En effet, la pratique montre que la composition de deux transitions n'est pas équivalente à une transition double. La Figure 19 illustre cette remarque : le robot se situe avec certitude dans l'état de gauche, et il avance vers la droite d'un tiers de case. Il va donc aboutir dans l'état suivant avec une probabilité de 33%, et rester dans l'état initial avec 67%.

Une nouvelle transition nous conduit alors à rester dans l'état initial avec 44%, de se trouver dans le second avec 44%, et dans le troisième avec 12%. Une troisième transition d'un tiers de case va enfin nous conduire à 30% dans l'état initial, 44% dans le second, 22% dans le troisième et 4% dans le dernier.

On peut donc constater que le fait d'avancer d'une case exactement nous amène dans le second état avec moins d'une chance sur deux ! Encore pis, cette évolution nous permet d'atteindre le quatrième état avec 4% de chances. Cela n'est absolument pas réaliste, d'autant plus qu'aucun aléa n'a été ajouté au système. A elle seule, cette observation permet d'expliquer certaines évolutions étranges des diagnostics que l'on peut observer. En effet, toute transition possible, aussi improbable soit-elle, permet au système d'évoluer à une vitesse irréaliste.

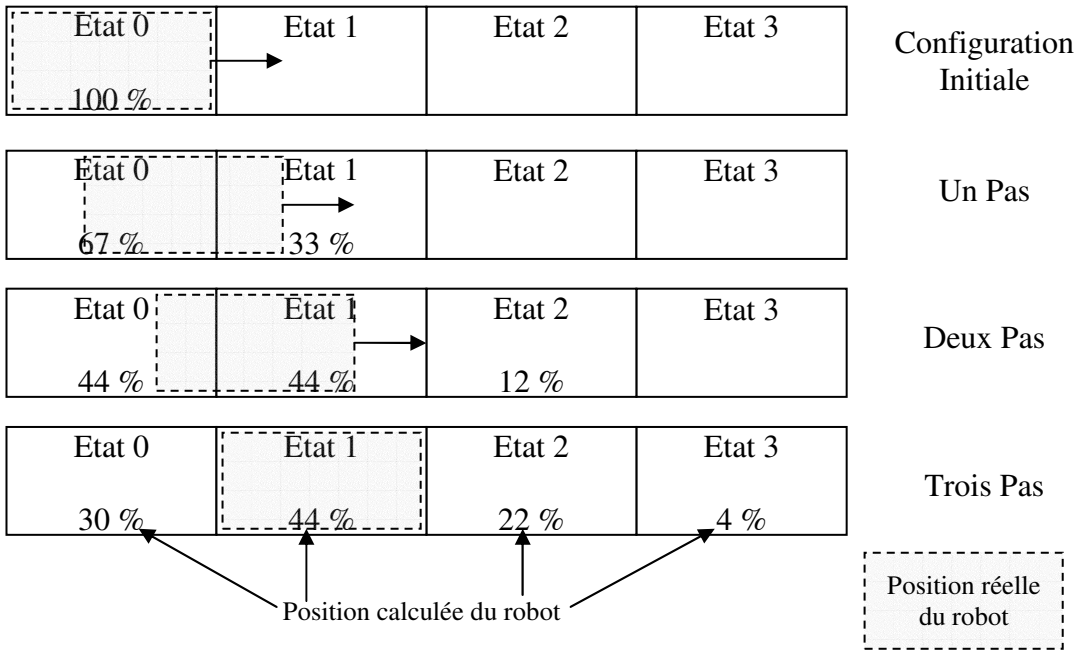


Figure 19 : Les travers de la discrétisation spatio-temporelle

A ce moment, la fonction d'observation devient prépondérante dans le diagnostic ; la matrice de transition ne joue plus que le rôle de modérateur, et en tranchant les cas ambigus. Actuellement, je n'ai pas trouvé de parade permettant de contrer ce problème. L'ajout d'un capteur virtuel, basé sur la dynamique du système, pourrait peut-être suffire, en interdisant les états non atteignables. Ce capteur pourrait d'ailleurs être fondé sur les équations différentielles citées précédemment.

5.3 Avantages et inconvénients

Les modèles stochastiques permettent donc une modélisation relativement fine d'un processus temporel dynamique. Ils incluent en standard la notion d'incertitude, ce qui permet de prendre en compte une bonne partie des aléas du système. Ainsi, l'imprécision portant sur une mesure peut être absorbée par les probabilités des matrices d'observation … du moins, jusqu'à un certain point ; au-delà d'une certaine quantité de bruit, la mesure originale n'est plus récupérable, quelque soit le traitement qu'on lui fasse subir.

Une partie des erreurs attribuables à la modélisation peut être absorbée par les probabilités de transition, de même que l'influence de certaines perturbations imprévisibles. De nouveau, au-delà d'un certain point, le bruit que représentent ces incertitudes prend le pas sur le phénomène observé, et cela peut conduire à de faux diagnostics, des erreurs de localisation, et ainsi de suite.

Le point fort de ce type de modèle tient à la cohérence temporelle. Il s'agit bien d'un même processus qui évolue au cours du temps. Cela permet donc une modélisation d'un système réel de façon directe. Contrairement à d'autres modèles, tels que les réseaux Bayésiens statiques, il n'est nul besoin de dupliquer le modèle dans le temps pour tenir compte du passé. Le même principe peut d'ailleurs s'appliquer au niveau d'une prédiction du futur.

L'inconvénient majeur des modèles que nous utilisons tient à leur aspect discret. En effet, ils possèdent un ensemble fini d'états, même si la distribution de probabilités sur ces états est continue. Bien sûr, il est possible d'interpoler la sémantique des états, mais la véracité n'en est pas assurée. Lorsque deux états dominants sont équiprobables dans un diagnostic, on peut facilement déduire que l'on se trouve à mi-chemin entre les deux. Cependant, l'interprétation correcte consiste à dire que le système est incapable de décider lequel de ces deux états correspond au mieux à l'état actuel du système. Souvent, ces deux interprétations se rejoignent et sont équivalentes. Pourtant, il est possible que cela soit en fait dû à un ensemble de données atypique qui ne peut appartenir qu'à un et à un seul de ces deux états. Par malchance, cette combinaison inattendue de données peut entraîner une confusion avec un autre état du modèle.

Au niveau de l'évolution du système, le temps est généralement considéré comme discret. Très peu de travaux semblent avoir été conduits sur ce problème. Les tentatives que j'ai conduites dans le cadre de la surveillance de l'anesthésie restent au stade des balbutiements. On ne peut réduire le temps à une succession de quanta (grains de temps insécables) sans retirer la sémantique de la matrice de transition. En effet, cette dernière tend alors inéluctablement vers la matrice identité, puisqu'en un temps infinitésimal, le belief-state devrait rester relativement constant. De même, la diagonalisation de la matrice de transition ne peut se faire qu'avec une matrice diagonalisable…Cela restreint quelque peu la flexibilité du modèle. Reste l'éventualité de l'utilisation d'équations différentielles, mais cela reste du travail à effectuer.

L'un des plus gros avantages de notre méthode de fusion des données des divers capteurs est qu'elle est complètement transparente aux yeux de l'utilisateur. Contrairement à la méthode des sommes de gaussiennes, un capteur donné voit les paramètres qui le concernent très localisés. Un capteur disposant de trois valeurs aura simplement trois probabilités qui lui seront associées pour chaque état. Cela permet donc une interaction facile avec des utilisateurs en présentant des valeurs directement liées à la sémantique sous-jacente.

Par rapport à la discrétisation par intervalles, l'avantage certain que présente ce modèle porte sur le sens associé à chaque division. On ne se contente pas de découper un signal en valeurs brutes, également réparties ; l'ensemble des possibles est scindé en fonction de leur impact sur les différents états. Cela contribue autant à renforcer la sémantique des états qu'à diminuer le nombre de paramètres nécessaires à la description du modèle.

Enfin, l'ajout ou la suppression d'un nouveau capteur n'obligera pas à revoir toute la matrice d'observation de chaque état, au contraire de la méthode des sommes de gaussiennes. Au pire, la modification du nombre de valeurs d'un capteur obligera à ajouter ou à retirer une ou plusieurs colonnes à la matrice de chaque état. Cette modification n'influe en rien sur la valeur des colonnes associées aux autres capteurs, puisqu'ils sont supposés indépendants.

Il s'agit donc d'une méthode particulièrement adaptée à la collaboration avec un expert non informaticien, car elle permet un accès simple et compréhensible à chaque paramètre du modèle. Ainsi, elle permet l'expression du diagnostic sous une forme directe qui montre en même temps l'aspect général du système (les états dominants) et les subtilités de ce dernier (l'importance relative des états dominés). Malheureusement, cette méthode n'a pas que des avantages. J'y vois deux inconvénients principaux : l'assise théorique et la complétude.

Le premier pose le problème de la validité des suppositions que nous avons faites. Comme nous le verrons dans le chapitre sur l'apprentissage, cette formulation ne respecte pas les conditions suffisantes à la convergence des algorithmes usuels, et en particulier l'algorithme de Baum&Welsh. Cette situation nous a d'ailleurs conduits à produire une méthode originale d'apprentissage permettant d'assurer la cohérence du modèle.

Le second inconvénient de notre méthode tient à la formule d'agrégation des capteurs. En effet, en supposant l'indépendance statistique des capteurs, nous admettons que les capteurs ne dépendent directement que de l'état caché du système. Cela se traduit par l'économie de la matrice de covariance, mais aussi par la diminution du nombre de profils que peut engendrer cette formule. Je pense néanmoins que, dans tous les cas, l'ajout de nouveaux états devrait permettre la prise en compte de la dépendance des capteurs. En effet, la dépendance apparente de capteurs physiquement indépendants ne peut être due qu'à un défaut de modélisation. Si l'état choisi est insuffisamment précis pour modéliser le système, l'erreur engendrée peut se répercuter sur le processus d'observation en créant des aléas.

Ces incertitudes peuvent en retour empêcher l'expression d'un modèle correct. En particulier, l'absence de la modélisation d'une facette du problème peut entraîner l'agrégation des situations correspondantes à d'autres situations proches qui, elles, ont été modélisées. A ce moment, outre l'imprécision que déclenche cette confusion, des dépendances simulées peuvent apparaître. Le raffinement du modèle devrait donc permettre d'en lever les imprécisions, au prix de l'augmentation de sa complexité en termes de nombre d'états.

6 Apprendre et s'adapter

6.1 Définition et Motivation

L'apprentissage est un processus qui permet d'acquérir et d'intégrer des connaissances non initialement incluses dans le bagage initial d'un agent. C'est , à mon goût, l'une des vertus les plus importantes des systèmes dits intelligents. L'adaptation est une forme restreinte d'apprentissage dans laquelle les connaissances acquises ne sont pas vraiment nouvelles. Elles correspondent plutôt à la modification de connaissances déjà connues pour améliorer leur adéquation au milieu environnant. S'adapter est donc souvent un passage nécessaire à l'amélioration de ses performances.

La notion d'apprentissage est partagée en deux grandes classes : les apprentissages supervisés, et ceux qui ne le sont pas. Le superviseur, ou professeur, dirige l'apprentissage et aide l'élève à atteindre le but qu'il lui fixe. Dans le cadre de l'apprentissage non supervisé, l'agent apprend par lui-même. Dans cette classe, on trouve surtout l'apprentissage par renforcement et diverses méthodes de classification. Une classification semble relativement mal adaptée à nos problèmes de diagnostic. En effet, par définition, ces algorithmes visent à associer des échantillons à des classes discrètes. Cependant, notre approche vise plus à établir un diagnostic qui soit progressif. Je ne détaillerai donc pas les méthodes de classification.

Dans le cas de l'apprentissage par renforcement , l'agent est récompensé lorsque son but est atteint. Inversement, il peut être pénalisé s'il commet des erreurs ; plus l'erreur est grave, plus la punition est sévère. Dans le cadre de l'intelligence artificielle, le terme d'apprentissage supervisé est réservé aux cas où la solution est explicitement montrée à l'élève. Ce dernier peut alors essayer, imiter, et recommencer tant que le processus ne convient pas au maître.

Les algorithmes d'apprentissages sont très importants dans la conception d'agents intelligents. Cela permet en effet au concepteur de l'agent de ne pas spécifier intégralement le problème. Poussé à l'extrême, cela permet à l'informaticien de créer un agent capable de résoudre un problème pour lequel il ne connaît pas de solution. De façon plus modeste, les algorithmes d'adaptation peuvent simplement aider l'agent à résoudre son problème, en lui faisant tirer parti de l'expérience qu'il a acquise.

Dans les projets que je considère ici, DIATELIC, la surveillance d'une anesthésie et la navigation d'un robot mobile, nous mélangeons un peu les attraits cités ci-dessus : il est souvent nécessaire de s'adapter à la situation courante pour permettre une résolution acceptable du problème. Dans des cas comme DIATELIC ou l'anesthésie, la solution exacte n'est même pas connue ! Au mieux, on a une bonne idée de la façon de résoudre le problème, comme dans le cas de la surveillance d'une anesthésie. Il reste donc à exprimer cette solution sous une forme exécutable, puis à laisser le système s'adapter aux patients, afin d'améliorer la fiabilité du diagnostic.

Dans le cas de la robotique, l'apprentissage permet de ne pas spécifier complètement le modèle de l'environnement. Ainsi, dans l'idéal, aucune carte n'est nécessaire à la navigation. On indique un but au robot, et ce dernier se déplace, apprend l'environnement et la carte associée, et trouve le chemin lui permettant de rejoindre le but. Nous ne sommes pas encore à un résultat tel que celui-ci, mais la communauté des chercheurs y travaille activement.

Je vais donc tout d'abord présenter assez rapidement les possibilités d'apprentissage dans les systèmes à base de règles. En particulier, j'évoquerai les principes du raisonnement à base de cas. Néanmoins, les systèmes à base de règles souffrant de nombreux problèmes pour prendre en compte l'évolution de systèmes dynamiques, je ne vais pas rentrer trop dans le détail de ces algorithmes.

Puisque les modèles stochastiques, et plus particulièrement les modèles markoviens, fonctionnent très correctement dans nos problèmes de diagnostic, je vais leur consacrer plus de temps. Je présenterai donc dans un premier temps les algorithmes standard d'apprentissage applicables à ces modèles. Or ces algorithmes sont difficilement applicables à des problèmes de diagnostic médical. En effet, la sémantique des modèles appris peut être très différente de celle que demandent les médecins. De plus, les données dont nous disposons respectent rarement les hypothèses de convergences de algorithmes, ce qui explique en partie qu'ils soient mal adaptés à ces problèmes.

Pour contrer ces défaillances, je proposerai successivement deux formes d'apprentissage mettant en jeu les connaissances du médecin. Ainsi, la sémantique du modèle est garantie par ce dernier. Le premier algorithme se base sur la création de modèles intermédiaires sur lesquels on peut apprendre facilement. Ces modèles étant construits sur la base du diagnostic fourni par les médecins, le respect de la sémantique est assuré.

La seconde approche se base sur la différence entre le diagnostic fourni par la version actuelle du modèle et celui corrigé par le médecin. En réduisant ces différences par une descente de gradient, le système est alors capable d'adapter son modèle pour se conformer au diagnostic du médecin tout en assurant de bonnes propriétés numériques. Je présenterai en particulier une méthode de relaxation simple permettant d'accélérer la recherche.

Je montrerai finalement comment ces algorithmes peuvent être appliqués au projet Diatelic aisément. J'expliquerai aussi pourquoi ce type d'algorithme n'est pas applicable à des projets où le temps nécessaire à l'expert pour corriger le diagnostic est trop important pour être mis en œuvre. Pour ce dernier type de problèmes, je proposerai alors une voie de recherche possible : remplacer la coopération avec l'expert par une prédiction de l'évolution future. Cependant, cette voie n'a pas pu être suffisamment étudiée pour donner de résultats probants. En effet, la prédiction de valeurs mesurables est un problème relativement difficile à résoudre. Sans cette prédiction, qui doit dépendre du modèle de diagnostic, il est impossible de définir un renforcement permettant d'apprendre.

6.2 Apprentissage dans les systèmes à base de règles

Dans ces systèmes, que j'ai présentés dans le second chapitre, il est difficile d'apprendre, car leur formalisme permet difficilement d'inférer des règles à partir de cas concrets. Dans les systèmes à base de règles classiques, il n'y a généralement aucun moyen d'ajouter des règles de déduction en cours de diagnostic. On se limite donc à de l'adaptation très légère.

Par exemple, dans la première version du système DIATELIC, l'adaptation au patient se résumait au calcul des moyennes mobiles. Ce n'est pas vraiment de l'adaptation du modèle, mais cela permet néanmoins de gérer les divergences du jour courant, par comparaison avec les jours passés. C'est le minimum de l'adaptation, mais il est nécessaire. Des formes plus poussées d'adaptation auraient été possibles, mais au prix d'un énorme coût en terme de développement. Il aurait par exemple fallu ajouter des métarègles telles que :

« *Si le patient reste longtemps à la limite d'une alarme, il pourrait être intéressant d'augmenter le seuil de cette alarme*. »

Il faut alors gérer de cette façon tous les cas possibles et envisageables. Il suffit pour cela d'ajouter à la base autant de règles que nécessaire pour prendre en compte chacun de ces cas. On peut néanmoins remarquer que ces règles font de nouveau appel à des seuils. Il faut donc régler ces derniers très finement, afin d'empêcher le système d'apprendre à transformer ses règles en éléments dénués de sémantique. Par exemple, on peut imaginer qu'un patient soufrant d'une hyperhydratation lente et progressive verrait son seuil d'alarme augmenter indéfiniment, sans jamais déclencher d'alertes.

Cependant de nombreuses variantes des systèmes à base de règles permettent d'apprendre à partir d'exemples. Ainsi, (41) présente un système d'apprentissage à base d'algorithmes génétiques. Cependant, cette méthode présente les troubles habituellement liés aux algorithmes génétiques : la mise à jour des règles est lente, et de nombreux essais sont nécessaires avant de trouver un ensemble de règles qui corresponde à ce que l'on cherche.

Dans le cadre d'un apprentissage supervisé de règles, on peut citer par exemple R-Clips (51) qui procède à une amélioration itérative de la base de règles, jusqu'à ce que tous les exemples soient reconnus. Cela se limite toutefois à la modélisation de processus statiques, car il est très difficile d'exprimer le résultat attendu lorsque ce dernier évolue au cours du temps. Il est alors impossible de superviser un tel apprentissage pour un processus qui évolue dynamiquement.

Par ailleurs, (22) présente un système permettant d'apprendre des règles pour un système dynamique médical. Cependant, pour y arriver, Guimarães et al. ont dû utiliser successivement des réseaux neuronaux et des méthodes d'apprentissage portant sur ces derniers. Finalement, un mécanisme d'extraction de règle a permis d'expliciter la connaissance apprise par ces réseaux afin de l'incorporer dans un système expert. Ce dernier devait d'ailleurs recourir à des opérateurs flous très particuliers pour tenir compte de l'aspect dynamique du processus. Il s'agit donc d'une méthodologie lourde, difficilement utilisable par les médecins eux-mêmes.

Une présentation très détaillée de toutes les phases nécessaires à l'obtention d'un système à base de règles utilisables dans un service d'urgences d'un hôpital est donnée dans (10). Cette présentation couvre aussi bien l'obtention de la connaissance que son intégration dans un système à base de règles, Mycin. En particulier, le chapitre 22 montre de façon claire les difficultés rencontrées pour tenir compte de l'aspect dynamique temporel nécessaire à une tâche de surveillance médicale.

Une autre approche couramment employée utilise des systèmes de raisonnement à base de cas. Lorsqu'il est soumis à un exemple réel, le système cherche alors des ressemblances avec les cas qu'il connaît, de façon à produire un résultat cohérent. Par exemple, un tel système est en cours de développement pour le traitement du cancer du sein (40). Cependant, ces systèmes sont relativement mal adaptés à l'analyse de systèmes dynamiques. De plus, dans des cas tels que le diagnostic de troubles de l'hydratation chez un dialysé, la connaissance nécessaire pour adapter un diagnostic est difficile à obtenir.

Muggleton présente dans (47) les fondements de la programmation logique inductive, formalisme permettant d'inférer des règles de déduction à partir d'une base de faits. Cependant, Page expose dans (52) les limitations actuelles de ce formalisme : son inaptitude à gérer des données probabilistes, la nécessité d'exprimer les bases du raisonnement en logique des prédicats, le manque d'interaction avec les experts, et l'énorme quantité de ressources (mémoire, temps de calcul) nécessaire à l'inférence.

L'induction d'arbres de décision se base sur les travaux de Quinlan sur ID3, C4.5, et leurs successeurs (58). Ici, l'objectif est de construire un classificateur permettant de séparer une base de faits afin de correspondre au diagnostic d'un expert humain. On procède donc par séparations successives des attributs des faits à classer. Cependant, comme le montre très bien la thèse de Kohavi (35), ce formalisme conduit généralement à des arbres immenses où les répétitions sont nombreuses ; en conséquence, ces arbres deviennent incompréhensibles pour les experts. Même les graphes proposés dans le chapitre 6 de (35) sont parfois difficiles à comprendre, bien que de taille largement inférieure aux arbres de décision.

Tous ces systèmes se basent presque exclusivement sur un apprentissage supervisé. Pour entraîner un tel système, un expert du domaine de l'application fournit plusieurs séries d'exemples, voire de contre-exemples soigneusement étiquetés par le diagnostic correct associé. Le système construit alors sur la base de ces exemples un système de décision donnant un diagnostic équivalent à celui de l'expert.

Cependant, une remarque semble attachée à chacun de ces systèmes : outre R-Clips (51) et (40), tous ces algorithmes nécessitent un temps de calcul très conséquent, ce qui les rend difficilement applicables à l'adaptation d'un modèle à un patient donné. De plus, les règles apprises sont fréquemment incompréhensibles par des experts du domaine d'application.

6.3 Apprentissage dans les modèles stochastiques

Les modèles stochastiques sont basés sur les outils mathématiques très puissants que sont les statistiques et les probabilités. Ces modèles sont définis par l'observation d'un grand nombre de cas, et, par exemple, l'estimation d'un cas moyen et d'une déviati on standard. Il s'agit donc d'une classe de modèles particulièrement bien adaptés aux problèmes d'apprentissage et d'adaptation. Dans ces problèmes, l'objectif visé consiste en effet à améliorer le modèle courant afin de mieux représenter un échantillon donné.

Parmi ces modèles, j'ai plus précisément sélectionné les modèles markoviens, puisqu'ils permettent une modélisation assez aisée des processus temporels dynamiques. Ces modèles observables sont naturellement dotés de fonctions d'apprentissage simples, basées sur le dénombrement de cas fournis en exemple. Pour les modèles partiellement observables, l'algorithme le plus usité est celui proposé par *Baum&Welsh*. Il a été présenté très clairement dans l'article de Rabiner (59). Je vais ici en rappeler le principe.

6.3.1 Algorithme de Baum&Welsh

Cet algorithme est une implantation du principe général *Expectation–Maximisation* (17). Cette méthode permet l'apprentissage de modèles statistiques, en partant d'un modèle a priori, en se basant sur le dénombrement des résultats de sa propre classification. En effet, pour apprendre un nouveau modèle, on procède tout d'abord à la reconnaissance des données à l'aide du modèle initial.

Chaque observation est donc associée à chacun des états du système avec une certaine probabilité. Une fois ce processus terminé, on procède à un calcul statistique ayant pour but d'isoler les caractéristiques du système. Ainsi, chaque état est caractérisé par la distribution de probabilités représentant au mieux les observations qui lui ont été associées. De même, les règles d'évolution du système sont obtenues par des statistiques exprimant le nombre de fois que tel état a été suivi de tel autre, et ce pour chaque couple d'états. Le système va ensuite expliciter le nouveau modèle, puis l'utiliser dans un nouveau cycle de reconnaissance, d'étude statistique de la classification obtenue, et de création d'un nouveau modèle. Ce cycle est alors répété jusqu'à ce qu'aucune modification ne se produise.

Il s'agit donc d'un algorithme non supervisé dans lequel l'agent ajuste son modèle, afin de maximiser la probabilité d'observer les données qui lui sont proposées. La convergence de cet algorithme est assurée, sous la condition que chaque état soit traversé régulièrement, et un grand nombre de fois. A cette condition seulement, et si de plus les données sont représentatives du processus étudié, les statistiques seront suffisantes pour assurer la cohérence du modèle finalement obtenu.

Voici donc maintenant le détail de l'algorithme de *Baum&Welsh*. Il est basé sur les coefficients α, β, et γ issus de la procédure *Forward-Backward*. Pour simplifier la présentation, je supposerai que ces valeurs sont déjà calculées. Ce calcul correspond à la partie « Expectation » de l'algorithme *Expectation-Maximisation*. Je vous renvoie donc au chapitre traitant des modèles statistiques pour le détail de leur calcul. Par rapport aux formules présentées jusqu'alors, un petit aménagement a été fait : la matrice de transition n'est plus stationnaire ; elle dépend donc du moment où elle est utilisée. Cela permet en effet de tenir compte de l'action qui a été choisie à chaque pas de temps.

Rappelons les notations utilisées :

λ	le modèle utilisé
$A(s', s, t)$	$= P(q_{t+1} = s' \mid q_t = s, \lambda) = P(s \rightarrow s' \mid action_t, \lambda)$
$B(o, s)$	$= P(Observation = o \mid q = s, \lambda)$
$\alpha_t(s)$	$= P(O_t \ldots O_1, S_t = s \mid \lambda)$
$\beta_t(s)$	$= P(O_{t+1} O_{t+2} \ldots O_T \mid \lambda, S_t = s)$
$\gamma_t(s)$	$= P(S_t = s \mid \lambda, O_1 \ldots O_T)$

Introduisons une nouvelle valeur intermédiaire :

$$\xi_t(s,s') = P(q_t = s, q_{t+1} = s' \mid O, \lambda) \qquad \forall (s,s') \in S^2$$

$$= \frac{\alpha_t(s) \cdot A(s', s_t) \; B(o_{t+1}, s') \cdot \beta_{t+1}(s')}{\sum_{s \in S} \sum_{s' \in S} \alpha_t(s) \cdot A(s', s_t) \cdot B(o_{t+1}, s') \cdot \beta_{t+1}(s')}$$

A partir de ce jeu de probabilités, il est possible de ré-estimer les probabilités de transitions de chaque action (la partie « Maximisation »). Pour cela, on isole les pas de temps où chaque action a été utilisée, et on utilise pour chacune d'entre elles la formule suivante :

$$\bar{A}(s', s, a) = \frac{\sum_{t \mid action(t) = a} \xi_t(s, s')}{\sum_{t \mid action(t) = a} \gamma_t(s)} \qquad \forall (s, s') \in S^2$$

Cette formule permet donc d'expliciter la relation entre le nombre de fois où la transition de s à s' a été observée, par rapport au nombre de fois où l'état s a été visité. Afin de tenir compte de l'utilisation des différentes actions, on se limite naturellement aux pas de temps où cette action a été employée.

Il ne reste donc plus qu'à ré-estimer la fonction d'observation du modèle. Dans le cas où les observations sont discrètes, on procède de la façon suivante pour chaque symbole k observable par le système :

$$\bar{B}(k, s) = \frac{\sum_{t \mid o_t = k} \gamma_t(s)}{\sum_{t} \gamma_t(s)} \quad \forall s \in \mathcal{S}$$

Il s'agit donc simplement d'un dénombrement dans lequel on considère le nombre de fois que chaque symbole a été observé dans un état donné, en comparaison avec le nombre de fois où le système a visité cet état.

Cette formule peut alors être adaptée pour gérer nos capteurs flous, sachant que ces derniers renvoient une probabilité pour chacune de leurs valeurs à chaque pas de temps :

$$\bar{B}(c \rightarrow v, s) = \frac{\sum_{t=1}^{T} \gamma_t(s) \cdot P\left(c(o_t) = v\right)}{\sum_{t} \gamma_t(s)} \qquad \forall s \in \mathcal{S}$$

L'avantage de cette formulation est qu'elle est équivalente à la formule standard, si l'on suppose que la certitude du seul et unique capteur est alors 100% pour la valeur observée et 0% pour toutes les autres. L'inconvénient, c'est que sa formulation ne tient pas compte du fait que les valeurs d'un capteur donné sont liées entre elles.

Il existe d'autres formules de ré-estimation de la fonction d'observation lorsqu'elle est continue. En particulier, certaines ont vu leur convergence prouvée dans le cadre de leur intégration à l'algorithme de *Baum&Welsh*. Cependant, elles supposent que la fonction d'observation a la propriété suivante :

$$\int_{-\infty}^{+\infty} P(o \mid s)\, do = 1 \quad \forall s$$

Avec la formulation que je propose, dotée des intervalles flous, il n'est clairement pas question d'une telle propriété. En particulier, les valeurs *trop petite* et *trop grande* étant non bornées, leur assigner une probabilité d'observation non nulle fait diverger l'intégrale très vite. Il ne nous est donc pas possible d'utiliser de telles formules avec notre modèle perceptif, puisqu'il est hors de question de forcer ces probabilités d'observation à 0.

6.3.2 Applicabilité aux problèmes de diagnostic

Le problème l'algorithme de *Baum&Welsh* est qu'il est non–supervisé. En effet, il choisit les probabilités d'observation de façon à ce que la probabilité d'observation de la séquence sur laquelle se fait l'apprentissage soit maximale. Il regroupe donc les observations se ressemblant dans les mêmes états de façon à accroître leur adéquation. A nul endroit, la sémantique des états n'intervient. Il est donc possible qu'une observation un peu à cheval entre deux états soit classée dans le mauvais état, ce qui va entraîner l'adaptation de cet état. Ses probabilités d'observations vont donc évoluer pour inclure cette nouvelle donnée.

Dès lors, à l'itération suivante, il est possible que de nouvelles observations soient jointes à cet état, surtout si elles étaient proches de celle qui vient d'être assimilée à tort. De proche en proche, la sémantique des états dérive, et ne correspond plus à ce que l'expert attend. Le diagnostic perd alors toute sa valeur, car même s'il est mieux adapté aux données, la classification sous-jacente n'a plus de signification. Cet algorithme est donc inutilisable lorsque la sémantique des états est bien spécifiée, car il n'a aucun moyen de la déterminer, et donc de la respecter. Deux solutions se présentent alors : soit on utilise des états ayant une sémantique forte, soit on utilise cet algorithme efficace. Nous avons donc développé de nouvelles méthodes qui permettent de conserver la sémantique des états tout en adaptant le modèle aux données. Ces algorithmes se basent sur la collaboration avec les experts humains.

6.4 Utiliser l'avis des experts

6.4.1 Motivation

La question de la correction du diagnostic est difficile à résoudre. Comment peut-on exprimer ce qu'est un bon diagnostic ? Cette question reste aujourd'hui encore en suspens. Actuellement, notre meilleur atout est l'expert du domaine. Lui seul a l'autorité nécessaire pour juger de la qualité et de la correction du diagnostic du système. Avec cette idée en tête, il nous a semblé évident que l'avis d'un tel expert peut être inestimable, s'il est exploité correctement. J'ai donc envisagé plusieurs méthodes permettant de donner la parole à un expert, de façon à aider le système à parfaire son modèle. Je vais présenter ces différentes méthodes, en essayant d'expliquer quels sont les avantages et les inconvénients de chacune.

6.4.2 Apprentissage supervisé par étiquetage

Cette première approche est tirée des méthodes d'apprentissage utilisées en reconnaissance de la parole. Elle est basée sur un corpus de données étiqueté par l'expert. Ce corpus correspond à un ensemble de séquences d'observations dans lesquelles chaque observation est associée à un état particulier du modèle. Cette association, que nous appelons *étiquetage*, est faite par un ou plusieurs experts du domaine. Cette tâche peut être longue et fastidieuse, car le nombre d'observations par séquence peut être particulièrement important. De plus, pour obtenir finalement un modèle robuste, il faut disposer de nombreuses séries d'observations indépendantes, ce qui accroît encore la tâche de l'expert.

Une fois que le corpus est étiqueté, le système va procéder à la construction de modèles temporaires, adaptés à la suite d'étiquettes fournies par le corpus. Un de ces modèles est construit pour chacune des séries d'observations. Pour une suite d'étiquettes données, le modèle construit va comporter autant d'états différents que d'observations dans la séquence.

Chacun de ces états est basé sur l'étiquette fournie par l' expert : c'est en fait une copie de l'état du modèle initial désigné par cette étiquette. Les transitions sont simples : chaque état permet d'accéder à son successeur avec une probabilité de100%. La Figure 20 résume cette construction. Lorsque tous les modèles sont prêts, on utilise un algorithme d'apprentissage tel que celui de Baum&Welsh, modifié de façon à pouvoir traiter les liens entre ces états. Ces modifications sont très légères, puisqu'elles ne font que remplacer le numéro des états par le numéro de l'état du modèle initial dont il est issu. Je ne vais donc pas recopier ici le détail de l'algorithme. Une fois le modèle appris sur l'ensemble des séquences, les liens reliant les états des modèles temporaires aux états du modèle initial ont permis de calculer un nouveau modèle, de structure semblable au modèle initial. On n'a donc finalement utilisé que les algorithmes standard, mais en forçant la valeur du belief-state à chaque pas de temps.

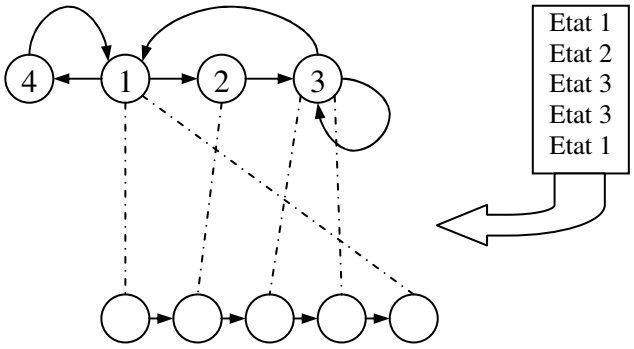


Figure 20 : Construction de modèles à étiquettes

Ce mode d'apprentissage est donc relativement facile à mettre en œuvre, sous réserve d'avoir à sa disposition un corpus étiqueté correctement. Cependant, cette solution pose de nombreux problèmes. En effet, il est tout d'abord nécessaire de pouvoir étiqueter les données. Cela semble simple, mais cela suppose que chaque observation puisse être associée à un état particulier. Ainsi, il est impossible d'obtenir une configuration permettant de garantir que deux états soient co-dominants à un instant donné. On peut bien sûr permettre deux transitions au lieu d'une, mais cela ne garantit pas que le belief-state sera réparti entre ces deux états. L'expert a donc le droit d'hésiter dans son diagnostic, mais on ne peut garantir que le modèle hésitera à son tour. Ensuite, ce modèle ne permet de prendre en compte que les données dominantes. Les états dominés ne seront pas appris du tout. Cela rejoint d'ailleurs la remarque précédente, dans le cas où la relation de dominant/dominé n'est pas franche.

Cet algorithme peut donc être utilisé à chaque fois que l'expert est sûr de son diagnostic, et que chacune des alternatives est prise en compte dans au moins quelques-unes des séries d'observations. Dans ces cas précis, cet algorithme est très efficace.

6.4.3 Apprentissage par descente de gradient

Les méthodes d'apprentissage que j'ai présentées jusqu'à présent ne sont pas applicables dans un domaine tel que la médecine. En effet, la quantité de données disponibles est en général assez restreinte, et il est rare qu'un diagnostic soit suffisamment tranché pour permettre un étiquetage satisfaisant.

De plus, ces deux méthodes ne font que travailler par analyse d'exemples positifs : étant basées sur le principe *Expectation–Maximisation*, elles utilisent des statistiques pour caractériser les faits qui ont été observés et qu'elles ont elles-mêmes triés. Par contre, aucun soin n'est pris pour s'assurer qu'un état dominé, par exemple, reste bien dominé au cours des itérations.

Ainsi, s'il existe un état dominé sur toute la longueur des séquences d'observations utilisées, cette situation va servir d'exemple pour caractériser cet état ! Par exemple, dans l'algorithme de Baum&Welsh, la phase de ré-estimation de la probabilité d'observation d'un symbole k dans un état s est la suivante :

$$\bar{B}(k, s) = \frac{\sum_{t \mid o_t = k} \gamma_t(s)}{\sum_{t} \gamma_t(s)} \quad \forall s \in \mathcal{S}$$

La probabilité d'observer le symbole *k* dans l'état *s* sera donc déterminée par les occurrences du symbole k, pondérées par la probabilité de cet état au moment de ces occurrences. Si, au maximum, cet état *s* apparaît avec une probabilité de 10%, cette formule de mise à jour sera tout de même utilisée. Le dénombrement ayant été fait par rapport au nombre de fois où cet état a été visité, une faible probabilité d'observation peut être compensée par une faible probabilité d'appartenance à cet état.

En définitive, cet état *s*, qui avait une probabilité initialement négligeable, pourra alors être l'état dominant à l'itération suivante, car il aura été modifié de façon à modéliser les données utilisées pour apprendre. A ce moment, la sémantique de l'état est totalement perdue.

Dans les conditions de convergence idéales de l'algorithme, ce cas ne se produit pas, puisque chaque état est visité un nombre infini de fois. Dans la réalité, ce n'est clairement pas le cas. Pour palier à ces inconvénients, j'ai proposé une méthode différente, permettant d'apprendre le modèle sans passer par un dénombrement.

6.4.3.1 Principe général

L'idée générale consiste à proposer à l'expert le diagnostic actuel, fourni par le modèle. Ce dernier peut alors y apporter des corrections, de façon à le rendre conforme à son propre diagnostic. Le système va alors rechercher dans l'ensemble des modèles possibles celui dont le diagnostic est le plus proche de celui désiré par l'expert. Cette notion de proximité peut être définie de façon quelconque grâce à une fonction d'évaluation de la qualité du résultat. En particulier, en définissant la fonction d'évaluation avec soin, il est possible d'intégrer des critères tels que l'écart entre deux courbes, la probabilité d'observation connaissant le modèle, et ainsi de suite. Le modèle résultant de l'apprentissage fournira alors un nouveau diagnostic, qui sera proposé à l'expert pour approbation.

Puisqu'il existe un nombre infini de modèles possibles (les paramètres sont continus), il est impossible de faire une recherche exhaustive. Cependant, de nombreuses méthodes ont été proposées pour approcher progressivement l'optimum. Parmi les algorithmes d'optimisation classiques, la recherche par descente de gradient m'a semblé la plus prometteuse. En effet, cet algorithme offre une grande flexibilité au niveau du critère à optimiser, ce qui permettra aux experts d'interagir facilement avec le processus.

L'idée majeure de l'algorithme de descente de gradient consiste à suivre la ligne de plus grande pente. La supposition sous-jacente est que la fonction soit suffisamment régulière pour que le minimum global se situe au bas de cette pente. Pour appliquer cet algorithme, il est donc nécessaire de calculer les dérivées premières de la fonction d'erreur.

Regardons donc tout d'abord cette fonction : c'est un calcul d'écart entre deux diagnostics, c'est-à-dire deux trajectoires dans l'espace des belief-states du système. Si l'on considère un belief-state comme un vecteur ayant autant de dimensions que d'états, l'écart entre deux de ces valeurs peut être vu comme la norme Euclidienne de la différence de ces vecteurs. L'écart entre les deux diagnostics se traduira donc par la somme des écarts à chaque pas de temps entre le diagnostic fourni par le médecin et celui fourni par le système :

$$Erreur = \sum_{t=1}^{T} \sum_{s=1}^{n} \left(Consigne_t(s) - P\left(S_t = s \mid \lambda, O\right)\right)^2$$

Pour pouvoir trouver la ligne de plus grande pente, c'est-à-dire la direction locale qui permet de réduire l'erreur le plus rapidement, le plus facile est alors de dériver cette erreur selon chacun des paramètres du modèle. Le vecteur des dérivées partielles nous donne alors le gradient de la fonction. Ce vecteur correspond à la direction du haut de la pente. Pour atteindre le minimum de la fonction d'erreur, il suffit alors de rechercher la valeur minimale de cette fonction le long de la droite portée par le vecteur de gradient.

Pour calculer ce gradient, nous avons la formule issue de la procédure *Forward-Backward* qui évalue la probabilité à un instant donné de chaque état :

$$\gamma_t(s) = P(S_t = s|\lambda, O) = \frac{\alpha_t(s) \,.\, \beta_t(s)}{\sum_{s' \in \mathcal{S}} \alpha_t(s) \,.\, \beta_t(s)}$$

Dérivons cette formule :

$$\begin{aligned} \mathrm{d}\gamma_t(s) &= \mathrm{d}\frac{\alpha_t(s) \,.\, \beta_t(s)}{\sum_{s' \in \mathcal{S}} \alpha_t(s) \,.\, \beta_t(s)} \\ &= \frac{\mathrm{d}(\alpha_t(s) \,.\, \beta_t(s))}{\sum_{s' \in \mathcal{S}} \alpha_t(s) \,.\, \beta_t(s)} - \frac{\alpha_t(s) \,.\, \beta_t(s) \,.\, d \sum_{s' \in \mathcal{S}} \alpha_t(s) \,.\, \beta_t(s)}{\left(\sum_{s' \in \mathcal{S}} \alpha_t(s) \,.\, \beta_t(s) \right)^2} \end{aligned}$$

De plus, nous avons également les formules permettant le calcul des coefficients α et β :

$$\begin{aligned} \alpha_t(s) &= B(O_t, \mathrm{s}) \sum_{s' \in \mathcal{S}} A(s, s') \,.\, \alpha_{t-1}(s') \\ \beta_t(s) &= \sum_{s' \in \mathcal{S}} A(s', s) \,.\, B(o_{t+1}, \mathrm{s}') \beta_{t+1}(s') \end{aligned}$$

Nous allons donc pouvoir calculer leurs dérivées également, afin de les intégrer au calcul du gradient de l'erreur. Il est cependant plus facile de séparer leur calcul, car il est relativement long à mener à terme, comme je vais le montrer en développant les premières étapes du calcul. Développer les étapes suivantes serait en effet purement mécanique, et cela n'apporterait rien de constructif.

$$\begin{aligned} \mathrm{d}\alpha_t(s) &= dB(O_t, \mathrm{s}) \sum_{s' \in \mathcal{S}} A(s, \mathrm{s}') \,.\, \alpha_{t-1}(s') + B(O_t, \mathrm{s}) \, d \sum_{s' \in \mathcal{S}} A(s, \mathrm{s}') \,.\, \alpha_{t-1}(s') \\ &= dB(O_t, \mathrm{s}) \sum_{s' \in \mathcal{S}} A(s, \mathrm{s}') \,.\, \alpha_{t-1}(s') \\ &\quad + B(O_t, \mathrm{s}) \sum_{s' \in \mathcal{S}} dA(s, \mathrm{s}') \,.\, \alpha_{t-1}(s') + A(s, \mathrm{s}') \,.\, d\alpha_{t-1}(s') \end{aligned}$$

$$
\begin{aligned}
d\beta_t(s) &= d\sum_{s' \in \mathcal{S}} A(s', s) \cdot B(o_{t+1}, s')\beta_{t+1}(s') \\
&= \sum_{s' \in \mathcal{S}} dA(s', s) \cdot B(o_{t+1}, s') \cdot \beta_{t+1}(s') \\
&\quad + \sum_{s' \in \mathcal{S}} A(s', s) \cdot dB(o_{t+1}, s') \cdot \beta_{t+1}(s') \\
&\quad + \sum_{s' \in \mathcal{S}} A(s', s) \cdot B(o_{t+1}, s') \cdot d\beta_{t+1}(s')
\end{aligned}
$$

On peut donc voir sans peine que ce calcul, même s'il n'est pas impossible, va être long et fastidieux. Je rappelle que cette démarche devra être faite pour chacun des paramètres du modèle, et cela jusqu'à ce que le minimum de la fonction soit atteint. En effet, la pente de la fonction peut varier selon l'endroit où l'on se trouve dans l'espace des paramètres. Il faut donc la recalculer pour chaque nouvelle position considérée. Heureusement, de nombreux morceaux de ces fonctions disparaissent selon la dérivée que l'on considère. Ainsi, par exemple, la dérivée de β selon la probabilité de transition allant de l'état 1 à l'état 0 sera :

$$
\frac{\partial \beta_t(1)}{\partial A(0, 1)} = B(o_{t+1}, 0) \cdot \beta_{t+1}(0) + \sum_{s' \in \mathcal{S}} A(s', 1) \cdot B(o_{t+1}, s') \cdot \frac{\partial \beta_{t+1}(s')}{\partial A(0, 1)}
$$

$$
\frac{\partial \beta_t(s)}{\partial A(0, 1)} = \sum_{s' \in \mathcal{S}} A(s', s) \cdot B(o_{t+1}, s') \cdot \frac{\partial \beta_{t+1}(s')}{\partial A(0, 1)} \qquad \forall s \neq 1
$$

Enfin, la descente de gradient peut mener à des probabilités illégales, puisque les contraintes de domaines de validité des paramètres ne sont pas prises en compte. Pour cette raison, cumulée au fait que les dérivées soient coûteuses à évaluer, j'ai préféré mettre en œuvre une méthode de descente de gradient sans utiliser de dérivées, mais avec respect des contraintes.

6.4.3.2 Descente de gradient sans dérivées

Mon choix s'est porté sur la méthode de *bracketing* qui consiste à rechercher le minimum par encadrements successifs. Cette méthode ne considère qu'une fonction à un seul paramètre. Je montrerai dans la section suivante comment on peut relâcher cette contrainte. Pour encadrer un minimum, il est nécessaire de connaître trois points disposés de façon à ce que la valeur de l'erreur soit minimale au point milieu. En effet, dans ces conditions, la Figure 21 montre les trois situations possibles pour une fonction suffisamment régulière.

Comme chacun de nos paramètres est borné, quelques cas particuliers apparaissent lorsque le point le plus bas des trois se trouve au niveau de l'une des bornes. A ce moment, seuls deux points suffisent, puisque l'on sait que le minimum ne peut dépasser la borne. Cependant, comme les bornes sont bien connues, cela ne pose pas de problème majeur. L'avantage de ces bornes, c'est que cela permet d'initialiser le processus facilement : on choisit comme points initiaux les deux bornes, plus un point intermédiaire. Le choix de ce troisième point est sujet à discussion : on peut choisir le point situé à la moyenne des deux bornes, ou on peut choisir un point particulier, tel que la position du paramètre actuel.

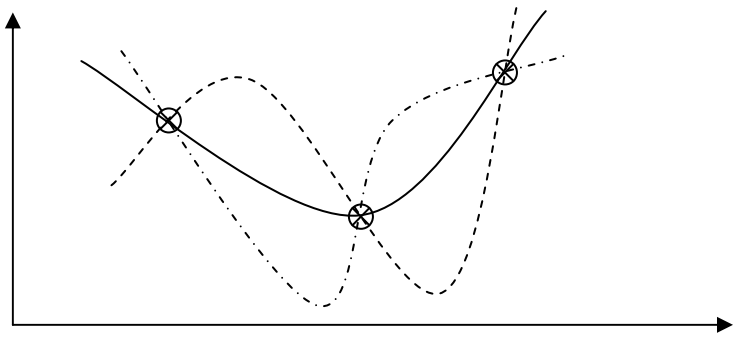


Figure 21 : Trois points, trois approximations de fonctions possibles

On peut aussi prendre un point aléatoirement, ou avec n'importe quelle répartition voulue. J'ai choisi de prendre mon point milieu égal à la valeur actuelle du paramètre. Cela permet ainsi de focaliser les recherches au voisinage du modèle courant. On peut en effet supposer que le modèle que l'on cherche ne sera pas totalement différent du modèle actuel. Après tout, la sémantique est la même, donc les paramètres devraient être relativement proches. Bien sûr, cette argumentation ne tient que si le paramètre actuel n'est pas situé sur l'une des bornes… Dans ce cas, aucun point milieu n'est nécessaire.

Lorsque ces trois points ont été choisis de façon à encadrer le minimum, l'algorithme va tenter de réduire cet intervalle progressivement, jusqu'à obtention de la précision voulue. Pour cela, on va utiliser, à chaque itération, deux nouveaux points. La valeur de la fonction en ces points va déterminer le prochain intervalle. Afin d'expliquer ce processus, je vais utiliser l'exemple de la Figure 22. Ce graphique présente la valeur de la fonction que l'on cherche à minimiser, en fonction du paramètre de cette dernière. Les points A et B correspondent aux bornes de l'intervalle en dehors duquel la fonction n'est plus définie. Les autres points sont envisagés successivement par l'algorithme de la façon suivante.

Le premier point, C, est choisi de façon à refléter la valeur initiale du paramètre. Comme la valeur de la fonction en ce point est inférieure à celle de la valeur en A et en B, on peut en déduire qu'il existe au moins un minimum local entre A et B. Pour améliorer l'encadrement, on choisit alors les point D et E, de part et d'autre de C. Ces points peuvent être choisis par dichotomie des intervalles, ou par application de la règle du nombre d'or. Cette proportion permet en effet d'optimiser les propriétés de régularité de l'algorithme, et donc sa performance. Pour avoir plus de précisions sur ces propriétés, vous pouvez consulter la section 10.1 de (54).

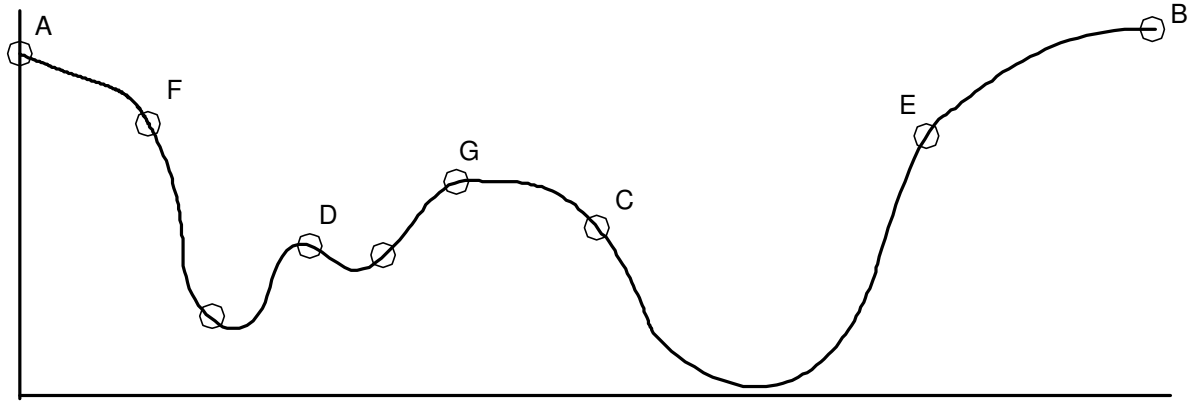


Figure 22 : Un paysage d'erreur possible

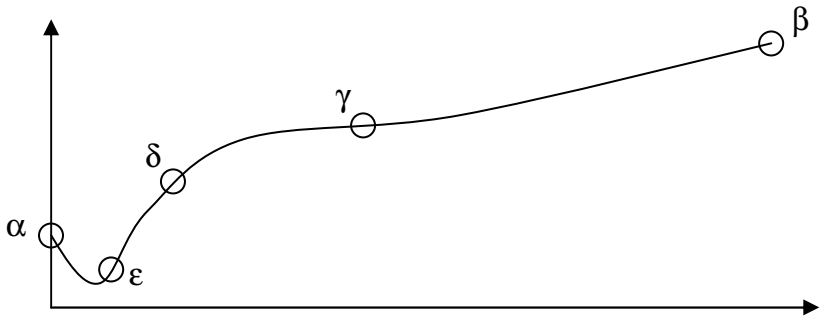


Figure 23 : Un autre paysage d' erreur

Parmi nos cinq points (A, D, C, E, et B), D est celui pour lequel la fonction a la valeur minimale. Il existe donc obligatoirement un minimum local entre A et C. On peut voir dans cet exemple que l'algorithme est très sujet aux minima locaux. En effet, si E avait été choisi un peu plus proche de C, l'algorithme aurait sélectionné l'intervalle [C, B], ce qui l'aurait conduit à l'optimum global de la fonction. Pour continuer, on va sélectionner à nouveau deux points, de part et d'autre de D. Ces points F et G sont associés à des valeurs de la fonction plus élevées que celle associée à D. Un optimum local se trouve donc forcément entre F et G. Les deux prochains points seront donc de nouveau de part et d'autre de D.

Lorsque la valeur minimum actuelle de l'algorithme est située sur l'une des bornes, j'ai déjà signalé que deux points suffisaient. La Figure 23 montre un tel cas. Sur ce nouvel exemple, les bornes sont α et β. La valeur actuelle du paramètre est β. Le nouveau point sera donc choisi au milieu de l'intervalle.

La valeur associée au point γ est comprise entre celle de α et celle de β. Un minimum doit dont exister dans l'intervalle [α, γ]. On choisira donc le point δ dans cet intervalle. De même, le minimum actuel se situant toujours en α, le point ε sera pris entre α et δ. A ce moment, la valeur minimale trouvée est celle associée à ε. Le minimum local cherché se trouve donc quelque part entre α et δ. Nous sommes revenus à un encadrement classique.

Un coup d'œil à l'algorithme met en évidence le fait que, à partir du moment où on encadre le minimum par deux points, il est impossible de revenir à un cas d'encadrement à un point. En effet, cet encadrement conserve systématiquement la valeur minimale trouvée au centre de l'intervalle, il est donc impossible de se trouver dans un cas où la valeur minimum se trouve sur l'une des bornes de l'intervalle.

Ce type d'algorithme d'optimisation est donc d'autant plus efficace que la fonction est régulière. Plus cette dernière présente de minima locaux, et plus l'algorithme risque de trouver l'un d'eux au lieu de l'optimum global. Cependant, mon expérience sur l'application de la méthode à Diatelic montre que les paysages d'erreur sont souvent lisses, et présentent très peu de minima locaux dans le cadre de l'apprentissage de diagnostic.

Avec cet algorithme, il nous est donc possible d'isoler une valeur optimale, ou presque optimale, pour laquelle une fonction est minimale (au moins localement). Cette fonction doit être continue, mais aucune autre contrainte n'est nécessaire. En particulier, il n'est aucun besoin de calculer ses dérivées. En contrepartie, il nous faut évaluer la fonction pour de nombreux points.

Afin de pouvoir appliquer cette méthode à l'apprentissage de diagnostic, il reste donc à tenir compte du fait que notre fonction d'erreur possède de multiples paramètres. Je vais donc présenter quelques méthodes permettant de relâcher la contrainte qui impose que la fonction n'ait qu'un seul paramètre. A nouveau, (54) donne de bonnes bases sur ces algorithmes.

6.4.3.3 Relaxation de la méthode : plusieurs paramètres

De façon triviale, si un ensemble de paramètres est optimal pour la fonction, alors chacun de ces paramètres est également optimal pour la projection de la fonction sur l'axe correspondant. En effet, si ce n'était pas le cas, le choix de la valeur optimale de ce paramètre permettrait d'améliorer la valeur de la fonction optimisée. L'ensemble des paramètres n'était donc pas optimal, puisque le nouvel ensemble obtenu est meilleur. On peut donc justifier que, tant qu'il existe un paramètre non-optimal, le vecteur des paramètres n'est pas optimal. Cependant, cette condition n'est pas suffisante pour garantir la découverte de l'optimum global de la fonction sur l'ensemble des paramètres.

En procédant de cette façon, les méthodes de relaxation permettent de travailler sur des fonctions à variables multiples : il suffit d'optimiser la fonction pour l'un de ses paramètres, tout en supposant les autres constants. Une fois que cela est réalisé, on choisit un autre paramètre, et on recommence. Tant que l'un, au moins, des paramètres permet d'améliorer la valeur de la fonction, il faut continuer d'envisager tous les paramètres. En effet, l'optimisation d'un paramètre peut conduire à la déformation du paysage d'erreur des autres paramètres. L'optimisation qui a été conduite sur chacun de ces paramètres n'est donc peut-être plus valide, et le point choisi sur cet axe peut ne plus être optimal.

Le processus d'optimisation est donc itératif, et le nombre d'itérations nécessaires est imprévisible. De plus, l'ordre dans lequel les paramètres sont optimisés peut être important. En effet, comme la modification d'un paramètre modifie les paysages d'erreurs des autres, il est possible de masquer des optima par malchance. Dans le meilleur des cas, cela conduira alors à un nombre d'itérations important pour atteindre le même optimum.

Dans d'autres cas, l'optimum global est perdu totalement. Ce nouvel algorithme est donc encore plus soumis aux optima locaux que son prédécesseur. Néanmoins, il est difficile d'évaluer l'existence de ces optima globaux sans tester l'espace tout entier, et sans essayer tous les ordres possibles quant au choix du paramètre à optimiser.

6.4.3.4 Relaxation de Powell

Cette méthode de relaxation est fondée sur l'observation suivante : il arrive fréquemment que l'optimisation d'une fonction à multiple paramètres prenne de nombreux pas, tous faisant évoluer les paramètres dans la même direction. Ainsi, la Figure 24 reflète un tel cas. La fonction considérée est une fonction quadratique à deux paramètres, représentée sous forme de courbes de niveaux. Ainsi, sur chacune des ellipses du graphe, la fonction conserve la même valeur. Cette fonction est particulièrement régulière, et ne possède aucun minimum local. Pourtant, comme le montre la figure, cette optimisation va prendre de nombreuses itérations avant d'atteindre son minimum.

La raison de cette lenteur tient à ce que chaque paramètre est optimisé indépendamment des autres. On travaille donc chaque fois sur une *coupe* de la fonction, selon l'un de ses deux axes. Or, l'optimisation d'un paramètre déplace légèrement l'optimum de la fonction projetée sur l'autre paramètre. L'optimisation de ce dernier compense ce déplacement, mais déplace simultanément l'optimum du premier paramètre. Ainsi, successivement, on se déplace par petits bonds à travers l'espace des paramètres.

Pour contrer ce comportement, Powell a proposé (56) de remplacer l'un des paramètres de la fonction par un vecteur correspondant à une nouvelle direction à chaque cycle. En fait, une fois que tous les paramètres ont été optimisés, la résultante de toutes ces modifications remplace l'un des paramètres.

Powell propose en particulier de remplacer le vecteur ayant donné la plus grande amélioration. En effet, le nouveau vecteur étant une composition des dernières modifications apportées au modèle, la plus importante d'entre elles sera également l'influence la plus forte sur le vecteur résultant. Eliminer ce paramètre permet donc de limiter la colinéarité des paramètres au minimum.

Pour clarifier cette explication, prenons un exemple : optimisons la fonction suivante :

$f(x, y) = x^2 + y^2 + x.\ y - 14\ x - 13\ y \qquad \forall (x, y) \in [0, 20]^2$
$x_0 = 0\ ;\ y_0 = 0\ ;\ f = 0$

Optimisons f selon x :

$f(x, 0) = x^2 - 14x$
$\rightarrow x = 7\ ;\ f = -49$

Optimisons maintenant f selon y :

$f(7, y) = y^2 - 6\ y - 49$
$\rightarrow y = 3\ ;\ \underline{f = -58}$

Arrivé à ce point, on va recommencer un nouveau cycle d'optimisation. Je vais donc présenter sur deux colonnes, en parallèle, l'approche standard (à gauche) et l'approche de Powell (à droite).

Optimisons f selon x :
$f(x, 3) = x^2 - 11\ x - 30$
$\rightarrow x = 5,5\ ;\ f = -60,25$

Optimisons maintenant f selon y :
$f(5,5\ ,\ y) = y^2 - 46,75 - 7,5\ y$
$\rightarrow y = 3,75\ ;\ \underline{f = -60,8125}$

Optimisons f selon x :
$f(x, 3,75) = x^2 - 10,25\ x - 34,6875$
$\rightarrow x = 5,125\ ;\ f = -60,9531\ldots$

Optimisons maintenant f selon y :
$f(5,1\ ,\ y) = y^2 - 7,875\ y - 45,43\ldots$
$\rightarrow y = 3,9375\ ;\ \underline{f = -60,9882}\ldots$

Optimisons f selon z, résultante du cycle précédant.
z remplace x puisque x est la composante principale de z.

$x = 7\ z$ et $y = 3\ z$
$f(z) = 79\ z^2 - 137\ z$
$\rightarrow z = 0,8670\ldots\ ;\ f = -59,3955\ldots$
$\rightarrow x = 6,0696\ldots\ ;\ y = 2,6012\ldots$

Optimisons maintenant f selon y :
$f(y) = y^2 - 6,9\ldots\ y - 48,1\ldots$
$\rightarrow y = 3,4651\ldots\ ;\ \underline{f = -60,1419}\ldots$

Optimisons f selon z_2, résultante du cycle précédant. z est la composante principale de z_2. z_2 remplacera donc z.

$z = 1 - 0.1310\ldots z_2$
$x = 5,125 - 0,9446\ldots z_2$
et $y = 3,9375 - 0,4723\ldots\ z_2$
$f(z_2) = 1,5615\ldots\ z_2^2 - 0,1771\ldots\ z_2 - 60,9882\ldots$
$\rightarrow z_2 = 0,0567\ldots\ ;\ f = -60,9933\ldots$
$\rightarrow x = 5,0714\ldots\ ;\ y = 3,9107\ldots$

Optimisons maintenant f selon y :
$f(y) = y^2 - 7,9285\ldots\ y - 45,2806$
$\rightarrow y = 3,9642\ldots\ ;\ \underline{f = -60,9961}\ldots$

Cet exemple montre bien que la méthode de Powell est plus difficile à calculer (au moins à la main), mais donne de meilleurs résultats que la méthode de relaxation de base. Cependant, on peut observer que le premier cycle d'optimisation selon Powell donne de moins bons résultats que le cycle utilisant la méthode de base. Ce phénomène est visible sur la grande majorité des optimisations, mais il n'enlève rien au gain de cette méthode. En effet, le premier cycle d'optimisation n'est pas représen tatif de la descente globale, principalement à cause des conditions initiales. Par contre, dès le second cycle, optimiser selon le vecteur résultant permet d'accélérer efficacement la descente.

L'inconvénient majeur de la relaxation de Powell tient à la réduction progressive de l'espace de recherche. En effet, à force de remplacer les vecteurs correspondant aux paramètres de la fonction par des combinaisons des autres vecteurs, deux problèmes apparaissent simultanément. Tout d'abord, l'algorithme tend à perdre certaines des dimensions du problème. En effet, chaque fois que l'on opère un remplacement, l'un des vecteurs support du problème disparaît. Ce paramètre ne sera donc plus optimisé qu'à travers les vecteurs dont il fait partie. Dans certains cas, on peut observer la disparition pure et simple de certains paramètres. Néanmoins, si l'on suit les recommandations de Powell quant au choix du vecteur à remplacer, cet effet secondaire reste relativement anodin.

Le second problème tient à la dégénérescence du modèle. En effet, si les descentes résultantes des divers cycles se font toutes plus ou moins dans la même direction, la présence de deux vecteurs colinéaires, ou presque, va arriver tôt ou tard. Ce faisant, le modèle perd irrémédiablement une famille de paramètres, puisque le système de dimension n ne possède plus que $n-1$ vecteurs linéairement indépendants. Pour lutter contre ces problèmes, Brent propose dans (9) de remettre la base de vecteurs à leurs valeurs originelles après un certain nombre de cycles. Ainsi, la dégradation du modèle est annulée, et l'optimisation peut reprendre dans de bonnes conditions. Cependant, on conserve l'accélération du processus dans la période de temps séparant deux remises à zéro de la base de vecteurs. Tout le problème consiste alors à choisir astucieusement le moment de chacune de ces remises à zéro, de façon à permettre une accélération optimale avec une dégénérescence minimale.

6.4.3.5 Relaxation proposée

Je propose pour ma part une variante de la relaxation de Powell : au lieu de remplacer un paramètre par la résultante du cycle précédant, je propose d'ajouter temporairement ce nouveau paramètre à la base, le temps d'une optimisation. Après cette dernière, il devient inutile et disparaît. Ainsi, cela permet d'a ccélérer les descentes en escaliers, sans pour autant causer de dégénérescence du modèle à aucun moment. Par contre, on perd la possibilité de cumuler cette descente avec d'autres pour avoir une direction encore meilleure. Ma supposition, ici, est que la résultante des descentes le long des paramètres originaux suffira à produire cette direction. Cette méthode de relaxation est appliquée en Figure 25, sur le même exemple que la Figure 24. On peut alors constater que l'on fait mieux en 5 cycles de trois optimisations qu'avec la méthode standard, avec 10 cycles de deux optimisations.

Pour appliquer cette méthode, il reste un point de détail à régler : puisque chacun de nos paramètres est borné par un intervalle de validité, quel sera celui de notre nouveau vecteur ? En effet, la relaxation de Powell s'applique a priori pour des cas où les paramètres ne sont pas contraints. Aucune réponse n'est donc apportée quant au calcul des contraintes des nouveaux vecteurs. Je propose donc de considérer l'union des contraintes de chaque paramètre. Ainsi, plus un vecteur a une grosse participation à la descente combinée, plus il limitera l'amplitude de cette dernière. Il faut également prendre en compte le fait que certains paramètres sont liés, comme par exemple dans le cas des probabilités d'une même distribution.

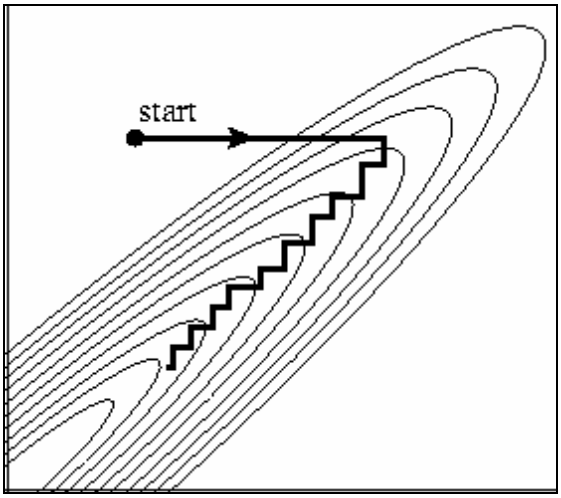


Figure 24 : Une longue descente en escaliers

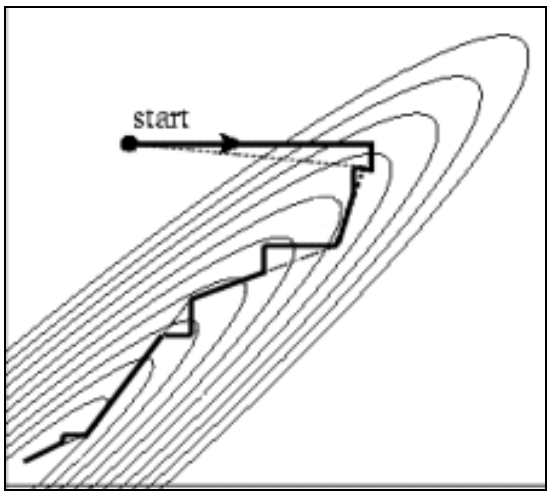


Figure 25 : Les escaliers court-circuités

Ignorons dans un premier temps ce problème de bornes liées. Il faut d'abord trouver l'intervalle de validité $[\lambda_{min}, \lambda_{max}]$ avec lequel le vecteur conjugué des n pas précédents peut être utilisé :

$$\begin{cases} V_{1\,min} \le V_1 + \lambda\,\Delta V_1 \le V_{1\,max} \\ V_{2\,min} \le V_2 + \lambda\,\Delta V_2 \le V_{2\,max} \\ \dots \\ V_{n\,min} \le V_n + \lambda\,\Delta V_n \le V_{n\,max} \end{cases} \qquad \forall\ \lambda \in [\lambda_{min}, \lambda_{max}]$$

Dans ces conditions, il est possible de résoudre le système en prenant l'union des contraintes imposées par chacune des équations. En effet, chaque équation du système impose des bornes à l'évolution de λ. Il suffit donc de prendre les contraintes les plus fortes pour satisfaire chacune des équations. Notons au passage qu'il existe forcément une solution, $\lambda=0$.

Considérons maintenant que certaines de ces équations soient liées entre elles par de nouvelles contraintes. On peut alors procéder à un regroupement qui permet d'obtenir un certain nombre de paquets d'équations tels que les contraintes soient indépendantes d'un paquet à l'autre. On peut donc procéder comme pour le système ayant des équations indépendantes : on calcule les contraintes de chaque groupe d'équations, puis on en prend l'union, ce qui nous donne finalement les bornes de la prochaine descente de gradient.

Pour un ensemble de k équations reliées par des contraintes linéaires telles que celles portant sur une distribution de probabilités, nous avons donc un système classique de simplex. Dans le cas qui nous intéresse, ce système présente les contraintes suivantes :

$$\begin{cases} 0 \le V_1 + \lambda\,\Delta V_1 \le 1 \\ \dots \\ 0 \le V_k + \lambda\,\Delta V_k \le 1 \\ \sum_{i=1}^{k} V_i + \lambda\,\Delta V_i = 1 \end{cases}$$

La première chose que l'on peut remarquer, c'est que l'un des paramètres est superflu. En effet, la dernière équation suffit à définir entièrement l'un des paramètres, au choix. De plus, on peut contraindre un peu plus chacune des équations restantes en tirant parti des autres, ce qui permet de se passer de l'équation exprimant la somme des paramètres.

L'autre avantage de cette formulation est qu'elle permet de faire apparaître explicitement les liens croisés de dépendance qui unissent chaque paramètre à l'ensemble des autres :

$$\begin{cases} 0 \le V_1 + \lambda\, \Delta V_1 \le 1 - \sum_{i \neq 1} V_i + \lambda\, \Delta V_i \\ \dots \\ 0 \le V_{k-1} + \lambda\, \Delta V_{k-1} \le 1 - \sum_{i \neq k-1} V_i + \lambda\, \Delta V_i \end{cases}$$

Soit, pour chaque paramètre ayant reçu un delta non nul :

$$\begin{cases} V_i + \lambda\,.\,\Delta Vi \le 1 - \sum_{j \neq i} V_j + \lambda\,.\,\Delta V_j \\ 0 \le V_i + \lambda\,.\,\Delta Vi \end{cases}$$

Pour définir les bornes λmin et λmax du facteur λ, il faut isoler λ dans chaque inéquation. On doit alors scinder ce système en deux, selon le signe du ΔVi de chaque paramètre :

$$\begin{cases} 0 \le 1 - \left(\sum_{j=1}^{k-1} V_j + \lambda\,.\, \sum_{j=1}^{k-1} \Delta V_j \right) \\ Si\ \Delta Vi > 0,\ \dfrac{-V_i}{\Delta Vi} \le \lambda \\ Si\ \Delta Vi < 0,\ \dfrac{-V_i}{\Delta Vi} \ge \lambda \end{cases}$$

$$\begin{cases} \lambda\,.\, \sum_{j=1}^{k-1} \Delta V_j \le 1 - \sum_{j=1}^{k-1} V_j \\ Si\ \Delta Vi > 0,\ \dfrac{-V_i}{\Delta Vi} \le \lambda \\ Si\ \Delta Vi < 0,\ \dfrac{-V_i}{\Delta Vi} \ge \lambda \end{cases}$$

Afin de simplifier l'écriture de ces formules, définissons les termes suivants :

$$\zeta = \sum_{j=1}^{k-1} \Delta V_j$$

$$\eta = \sum_{j=1}^{k-1} V_j$$

En définitive, nous disposons donc de quatre types de contraintes :

- $Si\ \Delta Vi > 0,\ \dfrac{-V_i}{\Delta Vi} \le \lambda$
- $Si\ \Delta Vi < 0,\ \dfrac{-V_i}{\Delta Vi} \ge \lambda$
- $Si\ \zeta > 0,\ \lambda \le \dfrac{1 - \eta}{\zeta}$
- $Si\ \zeta < 0,\ \lambda \ge \dfrac{1 - \eta}{\zeta}$

On peut donc de nouveau résoudre ces inéquations paramètre par paramètre, équation par équation, et ne prendre finalement que les plus contraignantes de chacune de ces bornes pour le paramètre λ.

Par exemple, pour illustrer mes propos, prenons le cas de 5 probabilités appartenant à une même distribution. Les contraintes de cette distribution permettant d'exprimer l'une de ces probabilités en fonction des autres, nous avons donc 4 paramètres *V1*, *V2*, *V3*, et *V4* à régler.

Supposons que l'optimisation par *bracketing* nous donne l'évolution suivante :

$$\begin{cases} V1 = 0.2 \rightarrow 0.15 \\ V2 = 0.3 \rightarrow 0.4 \\ V3 = 0.1 \rightarrow 0.1 \\ V4 = 0.3 \rightarrow 0.1 \end{cases}$$

La descente conjuguée de l'étape additionnelle se fera donc le long du vecteur résultant de ces quatre pas :

$[-0.05, 0.1, 0, -0.2]$

Il reste donc à calculer les bornes de cette dernière optimisation. Nous avons donc :

$$\zeta = \sum_{j=1}^{k-1} \Delta V_j = -0.05 + 0.1 + 0 - 0.2 = -0.15$$

$$\eta = \sum_{j=1}^{k-1} V_j = 0.15 + 0.4 + 0.1 + 0.1 = 0.75$$

Pour les divers paramètres, les contraintes seront donc :

$$V1 : \quad \lambda min \geq \frac{1 - \eta}{\zeta} = -1.666\ldots \text{ et } \lambda max \leq \frac{- Vi}{\Delta Vi} = 3$$

$$V2 : \quad \lambda min \geq \frac{- Vi}{\Delta Vi} = -4 \text{ et } \quad \text{ et } \lambda max \geq \frac{1 - \eta}{\zeta} = -1.666\ldots$$

$V3$: *pas de contraintes*

$$V4 : \quad \lambda min \geq \frac{1 - \eta}{\zeta} = -1.666\ldots \text{ et } \lambda max \leq \frac{- Vi}{\Delta Vi} = 0.5$$

Nous avons donc finalement l'intervalle de validité [-1.666, 0.5] pour le paramètre λ. En n'importe quel point de cet intervalle, le vecteur de paramètres respecte les contraintes. Regardons en particulier ce qui se passe aux bornes de l'intervalle :

$$\begin{bmatrix} 0.15 \\ 0.4 \\ 0.1 \\ 0.1 \end{bmatrix} - 1.666\ldots \begin{bmatrix} -0.05 \\ 0.1 \\ 0 \\ -0.2 \end{bmatrix} = \begin{bmatrix} 0.233\ldots \\ 0.233\ldots \\ 0.1 \\ 0.433\ldots \end{bmatrix} \quad (a)$$

$$\begin{bmatrix} 0.15 \\ 0.4 \\ 0.1 \\ 0.1 \end{bmatrix} + 0.5 \begin{bmatrix} -0.05 \\ 0.1 \\ 0 \\ -0.2 \end{bmatrix} = \begin{bmatrix} 0.125 \\ 0.45 \\ 0.1 \\ 0 \end{bmatrix} \quad (b)$$

Les deux vecteurs de paramètres obtenus sont révélateurs : la somme des composantes du vecteur *(a)* atteint tout juste 100%. La 5ème probabilité sera donc nulle.

Comme la somme des variations est négative, toute soustraction supplémentaire du vecteur–support entraînera le dépassement du seuil au-delà duquel la cinquième probabilité sera négative. Parallèlement, les paramètres atteints en *(b)* mettent en évidence la seconde limite : le paramètre V4 a atteint la valeur de 0 (zéro). Tout ajout supplémentaire d'une fraction du vecteur–support entraînera donc forcément le passage de ce paramètre vers les valeurs négatives, car le delta de V4 est négatif.

Dans ces deux cas, une modification de λ en dehors de son intervalle de validité entraîne donc bien une combinaison de paramètres illégale vis-à-vis de nos contraintes puisque la variable V5 (dans le cas 1) ou la variable V4 (dans le cas 2) deviendrait négative. On peut donc justifier que toute modification des paramètres le long de ce vecteur-support, dans les limites de l'intervalle de validité de λ, sera cohérente.

6.4.3.6 Conclusion

J'ai montré dans cette section comment l'utilisation d'un algorithme de descente de gradient sans dérivées pouvait permettre l'apprentissage d'un modèle quelconque, selon un critère arbitraire. Dans le cadre des modèles statistiques, je présente également une méthode de relaxation permettant d'optimiser des paramètres liés par une même distribution de probabilités. Basée sur la méthode proposée Powell, elle montre les mêmes qualités d'accélération de la recherche, sans souffrir de ses limites en termes de dégénérescence.

Le choix du critère selon lequel le modèle est optimisé n'étant pas conditionné par des problèmes de dérivabilité ou de forme particulière, cela laisse une grande liberté. Je vais montrer dans les sections suivantes comment cette liberté a été utilisée pour adapter les modèles de nos trois projets.

6.4.4 Application à DIATELIC

Dans le cadre du suivi des problèmes d'hydratation de patients dialysés à domicile, l'enjeu de l'apprentissage est double : la constitution du modèle générique, et l'adaptation à chaque patient. Ces deux problématiques sont similaires, mais leurs paramètres sont très différents ; elles ont donc été résolues par deux traitements différents.

J'aborderai donc dans un premier temps l'élaboration du modèle initial, avant de présenter en détail la résolution des problèmes que pose l'adaptation de ce modèle à chaque patient.

6.4.4.1 Constitution du modèle initial

La constitution de la structure du modèle initial a demandé beaucoup de travail manuel. En effet, les choix du nombre des états, de la sémantique de chacun d'entre eux, ainsi que des capteurs médicaux à utiliser sont primordiaux quant à l'utilité du système d'une part, et quant à la facilité de sa mise en œuvre d'autre part. Ils ont donc été définis à travers une série de réunions avec les médecins de l'ALTIR, au cours desquelles nous avons essayé de comprendre le schéma de diagnostic qu'ils utilisaient. Il nous a donc fallu ensuite traduire ce dernier dans le formalisme de notre modèle.

Ce premier modèle était un système expert, qui a rapidement montré ses limites. Nous avons donc finalement traduit ces règles dans un nouveau formalisme, celui des modèles markoviens. Une fois ce travail préliminaire effectué, nous avons utilisé l'algorithme de Baum&Welsh, en alternance avec la méthode d'étiquetage, ce qui a permis la constitution d'un premier modèle adapté à notre corpus de patients.

Dès lors, une phase de vérification manuelle a montré certains des travers que j'ai expliqué dans les sections consacrées à ces algorithmes. J'ai donc réalisé un cycle d'apprentissages alternant les deux algorithmes déjà cités et des séquences de validation/correction manuelles.

Pour y parvenir, j'ai dû limiter l'algorithme de Baum&Welsh à une seule itération, de façon à limiter les dérives du modèle. En effet, cet algorithme tend vers la définition de paramètres garantissant un modèle numériquement mieux adapté aux données, et non au diagnostic voulu.

Lorsque le modèle a semblé stable, nous l'avons présenté aux médecins de l'ALTIR. Ces derniers l'ont validé d'un point de vue médical, avant de lancer la première expérimentation sur 2 patients. Depuis, ce modèle a été très peu modifié au cours des différentes expérimentations.

6.4.4.2 Adaptation du modèle au patient

La seconde application de l'apprentissage à ce projet est beaucoup plus intéressante. Afin d'adapter le modèle à un patient donné, il faut en effet être capable d'apprendre avec très peu de données un modèle cohérent avec la sémantique que les médecins placent dans les états. Pour réaliser cette opération, j'ai implanté l'algorithme de descente de gradient que j'a i présenté dans les sections précédentes. La relaxation inspirée de la méthode de Powell a également été utilisée, de façon à accélérer la convergence vers le modèle recherché.

a) Critère d'optimisation

La définition de la fonction à optimiser suit globalement la formule présentée dans la section introduisant la descente de gradient. Cette formule calcule donc la distance entre le diagnostic demandé par le médecin et celui fourni par le système. Néanmoins, elle a été modifiée de façon à tenir compte des spécificités de cette application. Ainsi, il ne suffit pas de trouver un jeu de paramètres qui permet de fournir le diagnostic demandé par le médecin, mais aussi d'assurer la stabilité de ce diagnostic. J'ai donc ajouté un facteur permettant de tenir compte de l'adaptation du modèle au patient, c'est -à-dire la probabilité que le modèle génère la suite d'observations reçues.

De plus, cette méthode permet de trouver *le* meilleur modèle associant les données envoyées par le patient au diagnostic demandé par le médecin. Cependant, comme je l'ai déjà signalé, la physiologie d'un patient peut évoluer au cours du temps. Le modèle du jour présent n'est donc pas forcément valide pour interpréter les données mesurées quelques semaines auparavant. J'ai donc ajouté un facteur d'atténuation qui permet de prendre en compte les données récentes plus que les données passées.

La dernière des modifications apportées concerne le calcul de la différence entre les deux diagnostics. En effet, l'expérience nous a montré que, pour le médecin, le positionnement des probabilités des états est relativement grossier. Traiter une différence de 1% ne présente donc aucun intérêt au niveau de la sémantique du modèle obtenu. Par contre, l'ordre relatif des courbes est très important, car cela montre l'importance accordée à chacune des causes possibles des troubles de ce patient. Pour gérer ces contraintes, j'ai donc utilisé une fonction *flou* qui permet d'amoindrir les petites différences, tout en amplifiant les différences importantes de façon conséquente.

En définitive, la fonction d'erreur contenant tous les paramètres de l'apprentissage est donc la suivante :

$$Erreur = \sum_{t=0}^{T} \left(\sum_{s \in S} flou\left(P(S_t = s \mid \lambda, O) - consigne_t(s)\right) \right)^{t_{réf}/1+t} + \frac{\gamma}{P(O \mid \lambda)}$$

Bien sûr, les différentes parties de la fonction sont pondérées de façon à pouvoir modifier l'influence relative des différents paramètres dans l'apprentissage. Ainsi, $t_{réf}$ permet de définir la période de temps correspondant à un affaiblissement donné des valeurs passées. De même, γ permet de donner plus ou moins d'importance au modèle global, en comparaison avec la proximité du diagnostic demandé. La fonction *flou* est une fonction quadratique également paramétrable, de façon à dilater plus ou moins la zone d'atténuation qui entoure la consigne donnée par le médecin.

Tous ces paramètres permettent donc de définir le but de la prochaine optimisation, selon que l'on désire obtenir un diagnostic plus stable, ou plus proche de ce que désire le médecin. Le paramètre d'atténuation est réglé par le médecin lorsqu'il estime que le patient a évolué rapidement, de façon à se concentrer plus sur les données récentes.

Cependant, l'un des travers qui a déjà été signalé pour les autres algorithmes d'apprentissage menace cette méthode également, bien que cela soit moins gênant en ce qui concerne la fiabilité du modèle obtenu. Il s'agit de la dégénérescence des états non visités par le patient.

L'algorithme de Baum&Welsh calcule la fréquence des occurrences des différentes observations reçues dans chaque état, pondérée par la probabilité de l'état en question au moment considéré. Cette fréquence servira ensuite à caractériser l'état en question dans le modèle mis à jour. Un état non visité sera modifié de la même façon qu'un autre, ce qui induit la perte de la sémantique de cet état. En effet, ce dernier sera modélisé à partir des observations issues des autres états. Ce modèle ne sera donc plus adapté à la sémantique que les médecins prêtent à cet état.

A l'inverse, notre algorithme cherche les paramètres permettant d'obtenir le diagnostic demandé. Un état non visité sera donc modifié de façon à assurer que sa probabilité reste faible ; cependant, sans exemple positif, au moins un cas où cet état apparaît avec une probabilité non négligeable, rien n'empêche l'algorithme d'apprendre des paramètres correspondant à des situations jamais rencontrées, voire même impossibles. Ce choix de paramètres permet en effet d'assurer que la probabilité de cet état restera faible. Néanmoins, cela ne garantit en aucune manière qu'elle sera forte si l'état physiologique du patient pour lequel cet état a été introduit apparaît. La sémantique de l'état correspondant est donc corrompue également, et il devient inutile.

Pour empêcher l'apparition de ce problème, j'ai ajouté une série de gardes empêchant la modification d'un état dans le cas où le patient ne l'a pas visité pendant la série de données concernée. Une phase préalable analyse donc quels sont les paramètres du modèle qui sont pertinents, et quels sont ceux qu'il ne faut pas modifier, faute de données suffisantes. Cette phase est calculée une fois pour toutes, sur la base du diagnostic fourni par le médecin. Elle n'est donc pas influencée par le modèle actuel. Ainsi, même si un état non rencontré atteint le seuil de validité au cours des optimisations, l'algorithme ne cherchera pas à modifier son modèle.

Inversement, si un état valide passe sous le seuil de validité au cours des cycles d'apprentissage, l'algorithme continuera tout de même à optimiser son modèle. Cette modification permet donc d'assurer que la sémantique des états non visités reste identique, tout en autorisant l'algorithme à optimiser la séparation des états qui ont été visités au moins une fois dans la période de temps considérée. Bien évidemment, le seuil en dessous duquel un état est ignoré est paramétrable par le médecin. Ce dernier peut ainsi autoriser l'algorithme à apprendre un modèle d'un état peu visité. Néanmoins, ce paramètre est à prendre avec beaucoup de précautions si l'on veut éviter la dégénérescence du modèle.

Le modèle de DIATELIC utilise 5 états, et 4 capteurs ayant 3 valeurs floues. La matrice d'observation compte 60 probabilités qui se réduisent à 40 paramètres de la même façon. En effet, les probabilités des observations des trois valeurs de chaque capteur sont liées au sein d'une même distribution, ce qui entraîne la redondance de l'un de ces paramètres. Le modèle dépend donc finalement de 60 paramètres, dont 20 sont déduits.

En effet, notre matrice de transition exprimant seulement la cohérence temporelle d'un système sur lequel on n'agit pas, nous avons jugé peu opportun d'apprendre ces coefficients. La matrice de transition utilisée compte 25 probabilités qui se traduisent par 20 paramètres grâce aux contraintes imposées par les distributions de probabilités. L'algorithme ne perdra pas de temps à essayer d'optimiser ces paramètres qui n'interviennent que peu dans le résultat final. Cela contribue donc encore à réduire le nombre des paramètres du modèle, et donc améliore d'autant la rapidité de la recherche.

b) Interaction du modèle avec les médecins

Les médecins peuvent donc modifier le modèle à travers la correction du diagnostic du système, et l'apprentissage qui s'en suit. Néanmoins, ils peuvent également modifier le modèle directement, sans passer par le module d'apprentissage. Cette dualité permet donc d'obtenir d'une part un apprentissage complet du modèle, guidé uniquement par le but, et d'autre part la modification ponctuelle d'un paramètre précis par le médecin lui-même. En cas de problème, si le module d'apprentissage est incapable de trouver un modèle convenable, le médecin pourra donc avoir le dernier mot. Je vais maintenant présenter de manière plus précise la partie qui nous intéresse au premier plan, la modification du diagnostic.

Lorsqu'un médecin détecte une anomalie dans le diagnostic du système, pour un patient donné, il peut lancer le module d'apprentissage. Ce dernier affiche donc tout d'abord le diagnostic actuel, tel qu'il est évalué par le système markovien. La grande différence avec l'affichage habituel du diagnostic, c'est que sa modification est possible. Le médecin va donc pouvoir le corriger de façon à le rendre cohérent avec le sien.

Modifier le diagnostic est une tâche qui peut être longue. En effet, pour chacun des pas de temps dont dispose le système pour apprendre, les probabilités des cinq états sont affichées sous forme de points. Ces points sont reliés à leurs voisins temporels par une courbe dont la couleur dépend de l'état associé à cette série de points.

c) Modification du diagnostic par les médecins

Le médecin a donc à sa disposition un jeu de 5 courbes, dont les points sont représentés par des *poignées*. Ce symbole, petit carré entourant le point, permet de mettre en évidence les points modifiables. Le médecin peut alors interroger la valeur numérique de chacun de ces points. Il peut aussi modifier la probabilité de l'un ou l'autre de ces points en le saisissant à la souris et en l'amenant à la position qu'il pense que la donnée devrait occuper.

Les autres courbes sont automatiquement réajustées pour assurer que l'ensemble des probabilités de ce jour reste cohérent. Cet ajustement est fait en mettant en œuvre un simple facteur d'échelle, ce qui permet de ramener la somme des probabilités à 100%, sans modifier l'ordre relatif des courbes. Cette modification exclut bien entendu le point modifié par le médecin, de façon à ne pas corrompre la modification qu'il vient de faire.

Il est possible d'alléger un peu la tâche du médecin en mettant en place une fonction *élastique*. En effet, un point donné est lié à ses voisins par les probabilités de la matrice de transition. Il est donc possible de modifier la position des voisins, de façon à ce qu'ils suivent le mouvement du point en cours de modification. Cela pourrait donc permettre d'accélérer la modification d'un grand nombre de points. Néanmoins, cette formule n'a pas été testée par les médecins. Il est donc difficile de dire si cette fonction peut les aider ou non. La Figure 26 montre un diagnostic typique, fourni par le système. La Figure 29 montre quant à elle la version obtenue après correction par le médecin. Ce dernier diagnostic va ensuite servir à guider le processus d'apprentissage.

L'étape d'apprentissage, bien qu'étant essentielle au fonctionnement du processus de modification du modèle, est complètement cachée aux médecins. En effet, le module d'apprentissage tourne sur le serveur, et non sur la machine du médecin.

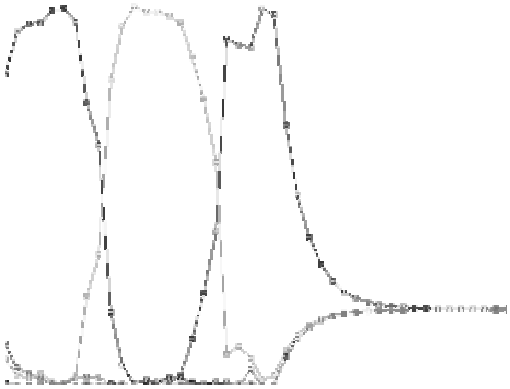


Figure 26 : Diagnostic du système avant correction

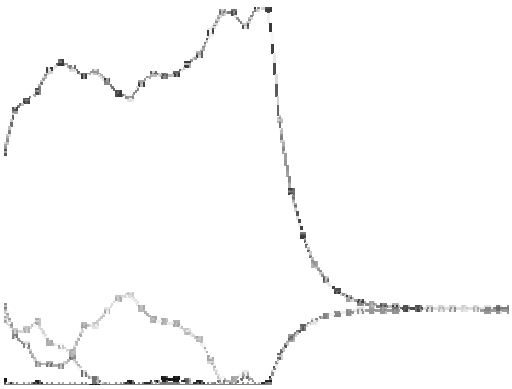


Figure 28 : Le diagnostic appris par le système

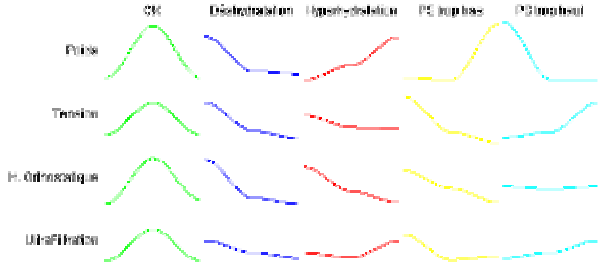


Figure 27 : Le modèle appris par le système

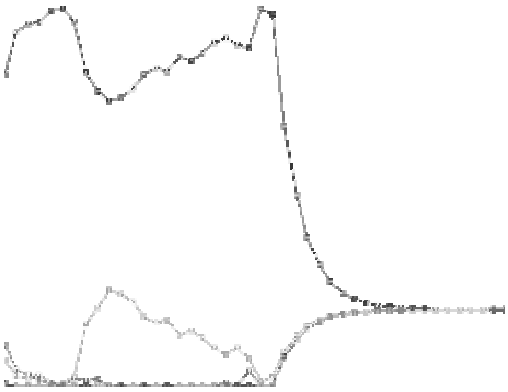


Figure 29 : Diagnostic corrigé par le médecin

Conserver l'apprentissage sur le serveur permet de s'affranchir des différences entre les machines, mais surtout, cela permet de garantir une puissance de calcul minimum disponible pour ce processus. D'autre part, puisque le système doit être utilisé à travers le réseau Internet, j'ai jugé préférable de ne pas transférer de grandes quantités de données pendant de longues périodes de temps. Or, si l'on prend le cas d'un apprentissage sur 60 jours, chaque affichage de courbes nécessiterait l'envoi de 240 probabilités.

Nous avons choisi de représenter ces dernières par des réels à double précision, afin de garantir que les erreurs de calculs ne soient pas trop importantes. Chaque affichage demande donc environ l'envoi de 1000 octets, plus ceux nécessités par le protocole de communication. Selon la précision demandée et la distance entre la consigne donnée et le diagnostic initial, le nombre d'itérations peut varier de 10 à 500. De plus, selon les paramètres, une itération peut durer entre un tiers de seconde et un vingtième de seconde, sur une machine de bureau classique.

Sur un serveur de calcul, on peut donc descendre bien en deçà. Si l'on se place dans le pire des cas, cela représenterait donc environ 20000 octets par seconde, ce qui suffirait à saturer n'importe quel modem. Nous nous sommes donc limités à l'affichage d'une barre de progression montrant au médecin que le système recherche une solution. Lorsque l'apprentissage est terminé, le système affiche au médecin le diagnostic obtenu, en conjonction avec le modèle correspondant. La Figure 28 et la Figure 27 montrent le profil obtenu par apprentissage sur l'exemple présenté précédemment. Le néphrologue peut alors valider ce résultat, et le nouveau modèle sera désormais utilisé pour ce patient.

Néanmoins, si le résultat obtenu n'est pas conforme à ses attentes, il peut soit annuler l'opération, soit recommencer un nouveau cycle d'apprentissage. A ce moment, il va retravailler le diagnostic fourni par le nouveau modèle et non le modèle initial. En effet, ce dernier devrait être plus proche des souhaits du médecin que le précédant.

Cette dernière manipulation lui permet en effet d'affiner à nouveau le modèle ; elle est donc particulièrement recommandée lorsque le nouveau modèle est meilleur que l'original, mais pas suffisant pour satisfaire le médecin. Du point de vue de l'informaticien, cette manipulation permet également de s'affranchir des minima locaux. En effet, le médecin va donner une nouvelle consigne qui sera certainement proche de la première. On peut néanmoins espérer qu'elle en sera suffisamment différente pour permettre au système de sortir du minimum où il se trouvait. En effet, le positionnement des courbes étant visuel, il est relativement difficile de recréer exactement la même configuration. Un minimum local dû à des aléas numériques est donc peu susceptible de se trouver au même endroit.

Afin d'obtenir un résultat équivalent, nous aurions pu utiliser des méthodes telles que le *recuit simulé* (16), ou encore simplement relancer l'apprentissage plusieurs fois, avec des modèles initiaux différents, avant de sélectionner le modèle donnant les meilleurs résultats. En effet, ces algorithmes permettent tous de se débarrasser des minima locaux, plus ou moins efficacement selon les problèmes. Cependant, ces différentes méthodes ont toutes le même inconvénient en commun : elles sont très longues à converger vers le modèle final. Comme le médecin qui a demandé l'apprentissage est devant son clavier, en train d'attendre le résultat, nous avons préféré un algorithme moins précis, mais plus interactif envers ce dernier.

Sur ce même jeu de données, mais avec une consigne différente, j'ai testé l'efficacité de notre méthode de relaxation. En utilisant l'algorithme standard, la convergence nécessite 340 itérations avant d'atteindre soit le seuil de stabilité de la fonction d'erreur, fixé à 0,001%, soit le seuil de précision des probabilités d'observations, fixé à 0,1%. Parallèlement, notre méthode de relaxation permet de trouver un résultat similaire en 52 itérations. L'amélioration représente donc une diminution d'environ 85% du nombre d'itérations nécessaires.

On peut donc considérer qu'il est rentable d'utiliser notre méthode, même si elle représente un surcroît de calcul. En effet, alors que la méthode standard dure 103 secondes (soit 0,3 seconde par itération), notre méthode dure 18 secondes (soit 0,35 seconde par itération). Chaque itération est donc 15% plus lente, mais le gain total en temps atteint 82,5%.

Bien sûr, ces résultats peuvent dépendre de l'exemple choisi, mais ce résultat numérique conforte l'impression que nous avions à ce propos. Ces valeurs ont été mesurées à plusieurs reprises, et dans des conditions similaires, sur un PC équipé d'un Pentium 3 cadencé à 550 MHz et de 256 Mo de mémoire.

6.4.5 Application à la surveillance d'une anesthésie

Dans le cadre de l'aide à l'anesthésie, l'apprentissage peut porter sur le modèle initial, et sur l'adaptation de ce modèle à chaque patient. Dans les deux cas, il est nécessaire de disposer de suffisamment de données pour apprendre le modèle et valider la démarche. Actuellement, ces données sont toujours insuffisantes. On reste donc au niveau du prospectif en attendant que ces données soient rassemblées.

6.4.5.1 Constitution du modèle initial

Nous avons résolu le problème de la constitution du modèle initial par une démarche inspirée de celle de DIATELIC. Néanmoins, le manque de données empêche une validation systématique. Le travail a donc été fait en grande partie par concertation avec les médecins anesthésistes du CHU de Brabois. Ces derniers nous ont indiqué les règles de diagnostic qu'ils utilisent, et nous avons traduit ces règles dans le formalisme des modèles Markoviens partiellement observables. Finalement, ce modèle a été testé sur quelques opérations dont nous avions une trace complète. Lorsque le résultat a été satisfaisant pour les médecins, le système a été évalué en salle d'opération.

6.4.5.2 Interaction des médecins avec le système

J'ai pu assister personnellement à quelques opérations. J'ai alors pu étudier l'interaction entre l'anesthésiste, l'infirmière –anesthésiste, et les différents instruments disponibles. Le résultat de cette étude montre que le diagnostic du système est considéré par les médecins comme un indicateur sans importance majeure.

Il semble donc que ce dernier ne présente pas d'information suffisamment pertinente pour s'imposer comme indicateur prioritaire. Je pense que cela est dû à la traduction systématique des règles de diagnostic du médecin. En effet, ce dernier n'utilisant que les paramètres affichés par les instruments, aucun des paramètres additionnels que nous envoient ces derniers n'est pris en compte. De plus, l'équipe médicale dispose d'indications venant d'autres instruments, pas encore informatisés. Parmi ceux-ci, la tension artérielle semble jouer un rôle primordial. Les données du respirateur sont également consultées de temps en temps, ce qui semble indiquer qu'elles peuvent se montrer intéressantes dans certaines situations. Ces dernières restent d'ailleurs à définir.

Parmi les signaux traités par le système, j'ai déjà indiqué que l'ASPECT nous donnait le BIS, l'EMG, et le rapport de suppression. Pour sa part, l'ANEMON nous envoie un indice de douleur et la fréquence cardiaque. Néanmoins, l'ASPECT envoie une multitude d'autres signaux que nous ne savons pas interpréter.

6.4.5.3 Autres signaux disponibles

L'analyse de ces données pourrait donc nous donner des informations additionnelles permettant d'accroître la précision du diagnostic. Parmi ceux-ci, nous avons :

- Calcul du BIS sur 3 canaux : un seul est utilisé par les anesthésistes, mais mes premières études montrent que la relation entre ces canaux pourrait être significative. En effet, par moments, les données de ces canaux sont équivalentes ; à d'autres instants, le canal 1 est supérieur au canal 0. Finalement, la situation inverse existe également. Je suis persuadé qu'il existe une raison expliquant ce comportement, mais cette dernière reste à découvrir.
- Puissance totale : pour chacun des canaux, une indication de la puissance relevée par les électrodes de l'électro-encéphalogramme est indiquée.
- *Spectral Edge Frequency* : cette mesure indique la fréquence en dessous de laquelle 75% des signaux relevés par l'EEG se situent. Cette indication permet donc de déterminer pour chaque canal si les signaux évoluent rapidement, ou non. La seule chose dont on est certain, c'est que l'activité de ces signaux est en relation directe avec la profondeur de l'anesthésie. Ces mesures pourraient donc peut-être nous donner une indication de réveil imminent.

Afin de déterminer l'importance de ces divers signaux, je propose d'utiliser tout d'abord une analyse statistique, en n'utilisant que les signaux actuels pour classifier les données. Cette analyse nous donnera donc une première mesure de la valeur des nouveaux capteurs, par comparaison avec ceux que les anesthésistes utilisent. Pour chacun de ces signaux, plusieurs cas peuvent se produire :

- Si la valeur est en adéquation avec le diagnostic initial, cette donnée est peut-être redondante. Dans ce cas, soit on ignore cette valeur purement et simplement, soit on l'utilise pour renforcer et consolider le diagnostic.
- Si la valeur semble en adéquation avec le diagnostic initial, mais qu'une forte variabilité est observée, on peut être en présence d'un signal bruité, ou d'un signal exprimant deux ou plusieurs causes sous-jacentes. Dans ce dernier cas, il peut être intéressant de séparer ces données, et de proposer cette nouvelle classification aux anesthésistes de Brabois. L'interprétation de ces nouvelles classes peut se montrer très intéressante si l'on est capable de lui associer une sémantique médicale.
- Si la valeur semble évoluer sans lien apparent avec le diagnostic, on peut être en présence d'un signal inutile, mais on peut alors se demander pourquoi les concepteurs de l'appareil médical ont jugé important de fournir cette donnée. Alternativement, ce signal peut trahir un processus indépendant de ceux que l'on étudie. On pourra alors soit étudier ce processus, et donc rajouter une dimension au problème, soit ignorer ce signal.

Dans tous les cas, une phase d'apprentissage de ces signaux sera nécessaire pour valider ou, au contraire, infirmer nos suppositions. Pour cela, on peut envisager d'utiliser un algorithme comme celui de Baum&Welsh, mais nous savons qu'il est soumis à de nombreux travers. Alternativement, nous pourrions utiliser un algorithme de descente de gradient, mais il nous restera à définir la fonction à optimiser. On pourrait par exemple optimiser la probabilité d'observation des séquences de données, connaissant le modèle. A ce moment, on risque de retomber dans le travers consistant à perdre la sémantique médicale a priori au profit du modèle mathématique.

Néanmoins, on peut aussi n'optimiser que les observations correspondant aux nouvelles données. Les anciennes probabilités d'observation restant fixées, on peut supposer que l'on conserve une bonne partie de la sémantique, puisque les médecins les jugent suffisantes pour évaluer la situation du patient.

Le second problème que nous avons envisagé de traiter, l'adaptation du modèle à chaque patient, est beaucoup plus ardu. En effet, on se retrouve dans la dualité du problème du poids-sec en dialyse : il faut être capable de déterminer si les divergences détectées par le système sont dues au patient ou au modèle. Cela pose donc une question d'ordre supérieur : qu'est-ce qu'un bon diagnostic ? A quoi peut-on le reconnaître? Intuitivement, un bon diagnostic est un outil qui apporte une information de qualité. Cette qualité est proportionnelle à la finesse du diagnostic, mais aussi à son adéquation à la situation réelle.

Cela repose ainsi le problème classique liant la réactivité du système à son taux de fausses alarmes. Le problème est donc d'évaluer si le diagnostic proposé est un reflet fidèle de cette situation ou pas. Dans DIATELIC, ce problème était résolu par le biais du médecin responsable du patient en question. Lorsqu'il n'est pas d'accord avec le diagnostic du système, il corrige ce dernier de façon à permettre au modèle de s'adapter en conservant la sémantique du modèle. Dans le cas de l'anesthésie, le médecin ne dispose pas d'un temps suffisant pour une telle correction. En effet, pendant qu'il corrige le diagnostic, il ne surveille plus le patient.

Or, nous avons vu avec le projet DIATELIC que cette correction pouvait prendre un temps considérable. La base de temps de la surveillance de l'opération étant de 5 secondes, le médecin devrait corriger de une à deux nouvelles données toutes les cinq secondes. Avec notre modèle courant, la modification du diagnostic correspond à repositionner 6 courbes les unes par rapport aux autres. Il n'est donc pas réaliste de demander un tel travail au médecin pendant l'opération. Cependant, on peut envisager cette approche pour apprendre un modèle moyen sur la base d'un corpus étiqueté.

Pour permettre une adaptation au patient pendant l'opération même, il nous faut donc trouver d'autres alternatives.

6.4.6 Avantages et Inconvénients

Nous avons vu que les méthodes d'apprentissage statistique classiques, telles que l'algorithme de Baum&Welsh, permettaient d'apprendre des modèles pertinents au sens statistique du terme. Sous réserve de disposer d'un corpus bien formé, la convergence vers un modèle stable est prouvée formellement.

J'ai montré également qu'un modèle statistiquement correct n'est pas forcément adapté au diagnostic médical. Il faut en effet associer à chacune des classes obtenues une sémantique médicale, ce qui n'est pas trivial. De plus, rien ne garantit que ces classes seront stables dans le temps. Il est possible, bien que peu probable, que l'ordre des classes soit modifié par un apprentissage donné, ou que leurs sémantiques soient inversées.

Pour contrebalancer ces inconvénients, j'ai proposé une méthode d'apprentissage par descente de gradient qui permet d'adapter le modèle à une situation donnée en tirant parti des conseils donnés par un expert. Cette démarche permet donc de s'affranchir des problèmes de corpus insuffisant ou mal formé, ainsi que de l'attribution d'une sémantique aux classes obtenues. En effet, l'algorithme apprend le lien entre la sémantique fournie par l'expert et les données observées, dans les limites des signaux disponibles.

La méthode de relaxation que je propose a montré qu'elle permettait un gain de temps considérable pour une surcharge de calcul très raisonnable. En particulier, elle démontre un comportement correct, aussi bien au niveau du respect des contraintes de validité des paramètres du modèle qu'au niveau de la couverture de l'espace de recherche.

Néanmoins, cette méthode d'apprentissage nécessite la collaboration active d'un expert du domaine. Cette interaction peut être longue et fastidieuse, ce qui empêche son application dans les cas où le temps est une ressource critique. Enfin, la descente de gradient est paramétrable, de façon à permettre son adaptation à chaque situation. Cependant, les valeurs de ces paramètres ne sont pas faciles à comprendre par une personne ne connaissant pas la technique d'optimisation sous-jacente. L'expérience montre ainsi que les médecins de l'ALTIR utilisent systématiquement les paramètres par défaut.

6.5 Apprendre sans professeur

6.5.1 Motivation

Les algorithmes que j'ai présentés dans les sections précédentes ont montré que l'apprentissage statistique est insuffisant dans le cadre de l'interaction avec un expert humain. De plus, l'apprentissage d'un diagnostic est, pour l'expert, une charge trop lourde pour permettre son application dans des cas où le système étudié évolue rapidement.

Il est donc nécessaire de trouver des méthodes alternatives permettant d'atteindre un résultat similaire à l'apprentissage de diagnostic, mais sans en faire endosser la charge de ce travail aux experts humains. Pour ne pas tomber dans les travers de l'apprentissage statistique, l'interaction avec l'expert est nécessaire de façon à assurer la cohérence du modèle obtenu. Il reste donc à trouver une interaction légère, mais riche en sémantique.

6.5.2 Apprentissage par renforcement

L'apprentissage par renforcement est un principe qui veut qu'un système autonome puisse apprendre un comportement en tirant parti de sa propre expérience. Cette expérience est exprimée sous la forme d'un signal de renforcement reçu pendant le fonctionnement même du système. Ce signal est un événement permettant d'indiquer au système que son comportement est bon, ou au contraire qu'il est incorrect. Il est donc associé à une valeur de récompense numérique, qui peut d'ailleurs être négative dans le cas où l'on juge utile d'infliger une punition à l'agent.

Ainsi, en robotique par exemple, il est préférable qu'un robot mobile cesse de percuter les murs rapidement. On attribuera donc une forte récompense négative à cette situation. L'objectif mathématique du système est alors de maximiser son renforcement, ou tout du moins, de trouver un comportement permettant de maximiser le renforcement attendu en l'appliquant. Le choix du renforcement est alors tel que le comportement obtenu correspond justement à une solution du problème original.

Traditionnellement, dans le jargon de l'intelligence artificielle, on oppose l'apprentissage par renforcement aux processus de décision ayant un modèle explicite. En effet, les algorithmes classiques ne construisent que la portion du modèle nécessaire pour obtenir l'action leur donnant le renforcement optimal. A l'opposé, les algorithmes basés sur des modèles utilisent une modélisation complète de l'environnement, qu'ils utilisent pour choisir leur action optimale. La récompense et le modèle sont donc connus à l'avance par le système. Néanmoins, je n'ai jamais réussi à obtenir une définition claire de l'apprentissage par renforcement.

La meilleure explication portant sur ce qu'est l'apprentissage par renforcement que je connaisse est celle donnée en introduction de (31) par Leslie Kaelbling :

> In the reinforcement learning scenario, the agent must choose an output to generate in response to each input. The reinforcement signal it receives indicates only how successful that output was; it carries no information about how successful other outputs might have been.

Rien ne s'oppose donc à l'interprétation de notre méthode de descente de gradient, par exemple, comme une technique d'apprentissage par renforcement. Dans ce cas précis, les actions du système correspondent à des modifications du modèle, et le renforcement sanctionne immédiatement chacune de ces modifications par une indication de progrès ou de régression par rapport au but fixé par l'expert. Cette vision nous ouvre cependant la voie à des méthodes hybrides, permettant d'améliorer le modèle du système en fonction des renforcements reçus.

La politique pourrait d'ailleurs être exprimée en terme de :

- Quand changer le modèle, et à quel coût ?
  Cette question est en effet le point central du problème. A quel moment est-il utile de modifier le modèle ? Autrement dit, est-ce que le modèle est correct, ou est-ce que le système étudié a évolué au-delà des possibilités du modèle courant ? Adjoindre un coût au changement de modèle permet de tempérer les décisions du système, retardant la modification du modèle jusqu'au moment où le gain espéré d'une modification dépassera ce coût. Cette valeur permet donc d'assurer une certaine inertie au système, afin d'empêcher qu'il ne modifie son modèle en permanence, pour s'adapter à chaque donnée reçue.

- Comment le modifier ?
  Cette question renferme toute la difficulté du processus. En effet, comme pour les modèles de décision Markoviens, les actions traditionnelles des algorithmes d'apprentissage par renforcement sont discrètes. Bien évidemment, les paramètres de notre modèle sont continus. Quelques tentatives ont été faites dans (48) et (49) pour généraliser le processus à des espaces ou à des actions continus, ce qui permet potentiellement son application à l'adaptation du modèle.

Si l'on choisit un renforcement qui exprime l'adaptation du modèle au système réel que l'on surveille, la modification du modèle sera directement liée aux renforcements futurs : si le modèle est modifié de façon incorrecte, il sera encore plus mal adapté que le modèle précédant, ce qui se traduira par des récompenses moins intéressantes. Par le jeu de l'apprentissage, le système pourrait alors apprendre à sélectionner les modifications nécessaires pour améliorer le modèle.

On peut aussi adapter le coût de la modification en fonction de l'ampleur des modifications faites. Ainsi, une modification superficielle aura un coût léger, alors que le remaniement de tous les paramètres du modèle devrait avoir un coût exorbitant.

L'apprentissage par renforcement pourrait donc permettre d'obtenir une adaptation efficace du modèle de diagnostic, sous réserve de trouver une expression du renforcement du système qui exprime l'adaptation du modèle. Je vais donc présenter rapidement les algorithmes de référence quant à l'apprentissage par renforcement, avant de m'intéresser à la définition de la récompense proprement dite.

6.5.3 Algorithmes standard d'apprentissage par renforcement

J'ai relativement peu étudié ces algorithmes, car ils sont mal adaptés à la collaboration avec des experts humains. Néanmoins, leur hybridation avec d'autres méthodes pourrait peut-être donner de bons résultats. Je ne vais donc pas détailler outre mesure ces algorithmes ; je vais juste donner quelques pistes.

L'algorithme le plus ancien est l'algorithme des différences temporelles (TD), présenté dans (66), qui se base sur la prédiction de la valeur de la récompense pour mettre à jour sa politique. Watkins a étendu ce modèle en proposant le Qlearning (69). Cet algorithme permet de simplifier TD en séparant les valeurs estimées de la politique apprise. McCallum a finalement proposé dans sa thèse (44) une méthode permettant d'apprendre l'espace d'états simultanément à la fonction de récompense, à partir de séquences d'observations, et de façon à optimiser la séparation des valeurs de la politique.

6.5.4 Obtention des renforcements

La question de fond, celle qui va conditionner tout le reste du processus, consiste à définir les signaux de renforcements. Dans les problèmes qui nous intéressent, aucun simulateur n'est disponible pour nous donner un renforcement idéal au moment opportun. Il nous faut donc trouver une méthode permettant d'obtenir un renforcement à partir d'une situation réelle. Nous avons vu en introduction qu'un coût associé à la modification du modèle pouvait être imposé facilement, de façon à assurer une certaine stabilité du modèle. Il reste donc à définir quelles sont les récompenses qui peuvent pousser le système à prendre la décision de modifier ce modèle, et donc d'endosser le coût associé à cette modification.

De plus, puisque l'on essaye d'optimiser le comportement à long terme, l'algorithme est amené à maximiser la somme d'un grand nombre de récompenses. Si l'on considère que la politique doit être optimale quelque soit le temps que dure le processus, il est nécessaire de tenir compte d'un nombre infini de récompenses. On utilise alors généralement un facteur d'atténuation exponentiel permettant de ramener cette somme à une valeur finie :

$$R = \sum_{t=0}^{\infty} \gamma^t . R(t)$$

Cette formulation ajoute néanmoins une nouvelle dimension au problème : comment faut-il choisir ce facteur γ ? En effet, plus il est proche de 100%, et plus la politique favorisera des récompenses à long terme. Plus il est proche de 0, plus la politique favorisera des récompense rapides. Ce facteur conditionne tout l'apprentissage ; il ne faut donc pas le négliger.

Dans le cadre du diagnostic, ces récompenses reviennent finalement à juger de la qualité de ce diagnostic. Comme je l'ai déjà signalé, c'est un problème difficile, ne serait-ce que pour le formaliser. Si l'on autorise une interaction avec l'expert du domaine, de façon à garantir la sémantique du modèle, on peut espérer que ce dernier va nous donner le renforcement dont nous avons besoin. Le problème, c'est que ce renforcement correspondra certainement à la situation actuelle, et non à la situation future. A l'extrême, dans un cas comme celui du diagnostic médical, le renforcement peut correspondre à une situation passée, car c'est avec du recul que l'expert se rend compte que la situation était mal diagnostiquée. On s'éloigne donc d'autant du schéma standard des algorithmes d'apprentissage par renforcement. Néanmoins, on pourrait utiliser ce renforcement pour effectuer un apprentissage a posteriori, en tentant de modifier le modèle passé. Cela revient donc à considérer le présent comme un futur, relativement au moment où l'on apprend.

On pourrait par exemple imaginer une méthode de renforcement où l'expert indiquerait son niveau de confiance dans le diagnostic fourni par le système. Cela permettrait donc d'imiter plus ou moins le mode de diagnostic de ce dernier, tout en demandant un investissement minimal en termes de temps d'interaction. Il reste cependant à vérifier d'une part que l'expert est capable de fournir cette appréciation, et d'autre part que le modèle est capable d'en tirer parti. On peut alors estimer que le modèle convient tant que le système ne reçoit pas de renforcement.

A l'inverse, l'arrivée de l'un de ces signaux va remettre en question les diagnostics récents. Peut-on en tirer parti pour améliorer le modèle ? Le risque vient de ce que le système peut alors proposer des modèles nouveaux jusqu'à ce que l'expert donne son aval sur le diagnostic. Ce dernier risque alors d'y passer plus de temps que d'exprimer le diagnostic lui-même. En effet, comme le renforcement indique une qualité du diagnostic, la modification du modèle entraîne une modification de la récompense. Comme cette dernière est fournie par un expert humain, une nouvelle interaction est donc nécessaire.

Alternativement, on pourrait permettre à l'expert d'indiquer que, sur une période de temps donnée, le diagnostic dominant est d'un type particulier, potentiellement différent de celui proposé par le système. Cela permet donc d'obtenir une information plus précise, puisque l'on a une partie du diagnostic correct, avec la période de temps concernée. Il faut néanmoins que cette opération soit suffisamment simple pour que l'expert puisse s'y adonner, sans pour autant perdre de vue sa propre surveillance du processus. De plus, on risque alors de tomber dans le travers de l'apprentissage par étiquetage, car on ne dispose plus que de l'information dominante sur certaines périodes de temps. Cette méthode nécessite cependant l'intervention active de l'expert. Elle ne permet donc pas d'automatiser cette adaptation totalement.

Pour permettre une adaptation réellement indépendante de l'expert, il nous faut donc trouver une méthode objective permettant de juger si un processus de diagnostic donné est adapté à une situation particulière. Afin de rester relié à la réalité du processus, il est nécessaire que ce renforcement soit tiré des observations du modèle. Pour cela, il est nécessaire de rapprocher cette observation d'une valeur normale attendue… Autrement dit, il nous faut diagnostiquer la dérive du diagnostic par rapport à la situation réelle.

6.5.5 Renforcement par prédiction

Je tiens à préciser que cette voie de recherche n'a pas été entièrement étudiée. Ce que je présente ici reste donc une étude prospective. J'ai mené différentes expériences dans ce cadre, mais actuellement, aucune n'a donné de résultats vraiment probants.

La méthode que je propose pour résoudre ce dilemme consiste à utiliser deux systèmes évoluant en parallèle. Le premier diagnostique le processus étudié, en utilisant des modèles tels que ceux qui ont été décrits dans les chapitres précédents. Pendant ce temps, le deuxième système évalue le résultat donné par le premier. En effet, nous avons vu à plusieurs reprises qu'il est pratiquement impossible de résoudre le dilemme consistant à déterminer si le système dérive, ou si le modèle n'est plus adapté. Il faut donc obligatoirement un troisième agent, chargé de juger de l'adéquation des deux premiers. Cette approche revient donc à substituer à l'expert humain un expert informatisé.

Sachant que l'on se place dans le cadre de l'hypothèse Markovienne, le futur ne dépend que de l'instant présent. Le diagnostic courant, qui incarne l'état caché du système, devrait donc logiquement permettre de prédire l'état futur du processus étudié avec une bonne certitude. Le problème vient alors du fait que l'état réel du système est caché derrière un processus d'observation imparfait et bruité.

Je propose donc de mettre en œuvre un modèle de prédiction qui, à partir du diagnostic courant, fournirait la valeur prédite d'un ou plusieurs capteurs à un instant futur. Cet instant n'a évidemment pas besoin d'être particulièrement lointain , car plus l'on s'éloigne dans le temps, plus les perturbations du système peuvent modifier son évolution. Lorsque l'on atteint l'instant de la prédiction, on peut comparer la valeur réellement observée avec la valeur qui avait été prédite. Cette comparaison peut alors nous donner le signal de renforcement dont nous avons besoin. En effet, la différence entre les deux valeurs nous donne simultanément l'amplitude de l'erreur, et la direction de cette dernière. Cela pourrait donc nous permettre d'adapter le modèle de diagnostic en conséquence.

Le problème de cette approche vient du fait qu'elle suppose l'existence d'un module de prédiction d'un type très précis. En effet, il doit être capable de fournir une prédiction de la valeur future d'un capteur, en ne se basant que sur la valeur actuelle de l'état diagnostiqué par le système à évaluer. Ce diagnostic se doit donc d'être suffisamment complet pour permettre cette prouesse. On peut toutefois relâcher quelque peu cette contrainte, en autorisant le modèle de prédiction à utiliser les valeurs actuelles des capteurs. Il faut alors s'assurer que la prédiction dépend toujours de l'état diagnostiqué, et non seulement des valeurs actuelles. Dans le cas contraire, il serait impossible de se servir de la comparaison entre la valeur prédite et la valeur observée comme renforcement pour adapter le premier système.

Si l'on suppose qu'un tel module de prédiction est disponible, on peut alors construire un module d'apprentissage qui, sur la base de l'observation d'une ou de plusieurs divergences observées, va modifier le modèle de diagnostic. Un tel module peut être conçu de plusieurs manières. On peut tout d'abord suivre l'approche conventionnelle qui, sur la base du renforcement reçu, va choisir une modification arbitraire du modèle. On peut alors utiliser des algorithmes plus standard tels que le *Q-learning* (69), ou encore des algorithmes tels que ceux proposés dans (44), ou dans (19). Ces derniers sont mieux adaptés à des environnements disposant d'un grand nombre d'observations , avec des situations ambiguës. En effet, ils ne sélectionnent que les parties de l'observation nécessaire pour obtenir une réponse adaptée aux situations. Ils construisent donc leur modèle au fur et à mesure.

A nouveau, on peut aussi utiliser des algorithmes de descente de gradient, en minimisant cette fois l'erreur entre la suite de valeurs prédites et la suite de valeurs observées. Cette approche est, certes, plus longue à mettre en œuvre qu'une modification arbitraire du modèle, mais elle permet aussi de s'approcher à coup sûr de l'objectif, au contraire de ses rivales. On pourrait aussi hybrider les deux approches, en utilisant des algorithmes tels que *Q-learning* pour désigner un jeu de paramètres à optimiser. Une descente de gradient permettrait alors de travailler sur ces paramètres, de façon à minimiser l'erreur entre les deux séries de valeurs. Cette approche hybride permettrait donc de réduire le temps nécessaire pour trouver l'optimal.

a) Application à l'anesthésie

J'ai tout d'abord étudié cette approche sur l'application d'aide à l'anesthésie. En effet, cette approche aurait permis d'adapter le modèle au patient durant l'opération elle-même. Les anesthésistes de Brabois utilisent déjà une méthode de titration manuelle :

- Le patient est endormi.
  → Mesure du temps nécessaire avec une dose d'anesthésique connue
- Le patient est intubé. (on lui introduit le tube du respirateur dans la gorge)
  → Mesure de la réaction du patient

A partir de ces mesures, les anesthésistes choisissent les doses de produits à injecter pour obtenir un effet donné.

Cependant, ils adaptent leurs dosages tout au long de l'opération, selon les effets observés de la chirurgie sur les paramètres physiologiques du patient.

En me basant sur cette approche, j'ai donc expérimenté divers algorithmes de prédiction d'une donnée médicale, le BIS. Comme le système reçoit ce signal toutes les 5 secondes, et vu son influence sur le diagnostic des médecins, cela en faisait donc un client rêvé. Néanmoins, aucune de mes tentatives de prédiction n'a donné de résultats cohérents avec les valeurs observées. Je pense que cet échec est dû en grande partie au manque de données d'une part, et à la volonté de prédire une valeur numérique d'autre part. En effet, sans connaître ni les doses de produits injectées, ni les opérations chirurgicales en cours, il est difficile de prédire l'évolution de la profondeur du sommeil d'un patient. D'autre part, j'ai essayé de reproduire une valeur numérique du BIS à partir des données du système. Comme le modèle utilise des intervalles flous, la valeur exacte se trouve être perdue. A partir de ce moment, prévoir la valeur exacte à l'instant suivant relève de l'impossible. Même la prédiction d'une valeur moyenne a été impossible.

Dans la littérature, la majeure partie des modules de prédiction se basent sur des réseaux de neurones (11),(61),(70). Hélas, dans notre cas, c'est impossible. En effet, il est absolument nécessaire que la prédiction s'appuie sur les paramètres du modèle de diagnostic. Sans cela, il serait impossible d'en tirer une valeur permettant de juger l'adéquation du modèle au patient.

b) Application à la robotique

La robotique est un excellent candidat permettant de tester une autre approche : en effet, l'environnement sous-jacent est celui de la physique du solide, univers dont les lois d'évolution commencent à être relativement bien connues. Le module de prédiction devrait donc être relativement facile à décrire. De plus, et par opposition aux problèmes médicaux, il est relativement facile d'expérimenter différentes méthodes sur un robot, puisque cela ne met en jeu la vie de personne.

- **Approche probabiliste**

Cette nouvelle approche sera basée sur la prédiction des probabilités des intervalles d'un capteur donné. Au lieu de prédire la valeur exacte du capteur, le module permettra plutôt de juger si oui ou non la valeur observée était prédictible… sans la prédire vraiment. Par exemple, supposons que le module prédise que l'on doit observer une valeur faible (80%), ou moyenne (20%). A l'instant suivant, la valeur observée est évaluée : elle peut être considérée comme forte avec 70% de chances, et moyenne avec 30% de chances. Dans ce cas, on peut considérer qu'avec 6% de succès (20% * 30%), le modèle est peu prédictif.

Il reste donc à choisir une donnée à prédire : notre robot étant doté de capteurs de distance à infrarouges, ce sont ces derniers qui vont naturellement servir afin de fournir le renforcement. La caméra fournit en effet une valeur trop riche pour être prédictible aisément, à moins de disposer d'une reconstruction tridimensionnelle complète de l'environnement . Il nous faut donc être capable de prédire l'évolution de ces mesures de distance, de façon à pouvoir les comparer aux mesures réellement effectuées.

Supposer que l'environnement est structuré permet de contraindre les configurations possibles en supprimant celles qui présentent des transitions chaotiques. Plaçons-nous donc dans le cadre de l'environnement présenté sur la Figure 30. Ce dernier est discrétisé en 8 positions, chacune étant divisée en 4 orientations. Nous avons donc un total de 32 états. Pour chaque état, le robot est supposé au centre du carré correspondant, dirigé vers la base du triangle correspondant à son orientation.

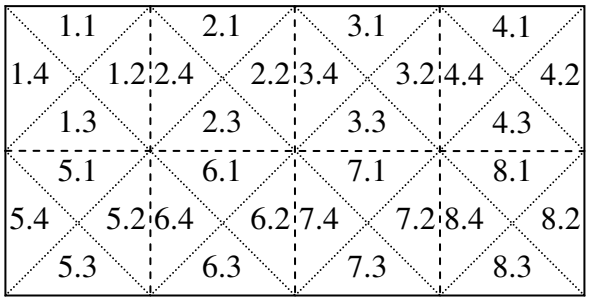


Figure 30 : Un environnement simpliste pour la robotique

Pour ce qui concerne les capteurs, j'utiliserai la discrétisation que j'ai présentée dans le chapitre portant sur la perception du monde : *proche, moyen, loin*. Par souci de simplification, je considérerai que le robot ne possède que 4 capteurs situés à l'avant, à droite, à l'arrière, et à gauche. L'état 1.1 sera donc caractérisé par la matrice d'observation suivante :

$$\begin{bmatrix} & \text{proche} & \text{moyen} & \text{loin} \\ \text{avant} & 80\% & 20\% & 0\% \\ \text{droite} & 0\% & 0\% & 100\% \\ \text{arrière} & 0\% & 75\% & 25\% \\ \text{gauche} & 80\% & 20\% & 0\% \end{bmatrix}$$

A chaque instant, le robot dispose donc d'un estimateur qui lui donne sa position dans l'environnement, sous la forme d'un belief-state portant sur les 32 états. Chacun de ces états possède sa propre matrice d'observation.

Connaissant l'action prise par le robot, il est possible d'évaluer la valeur que devrait donner le capteur *avant* dans chacun des états, en utilisant le modèle même du système d'évaluation de la position. De même, le modèle d'évolution de l'état correspondant à l'action en cours peut servir à prédire l'état probable du système à l'instant suivant. Ces modèles sont supposés corrects, puisque la valeur prédite va servir de référence pour évaluer l'adaptation de ce modèle à l'environnement.

Ainsi, par exemple, aller vers l'avant à partir de l'état *5.1* devrait nous conduire à l'état *1.1* ; néanmoins, les incertitudes liées à la modélisation discrète rendent ce processus hasardeux. Par exemple, supposons que l'on ait 10% de chances de rester sur place, 5% de chances de se retrouver en *2.1*, et 5% de chances de finir en *2.2*. Donc, partant de *5.1*, la prédiction nous indique que le capteur avant nous donnera les observations suivantes :

$$10\% * Obs_{5.1} + 80\% * Obs_{1.1} + 5\% * Obs_{2.1} + 5\% * Obs_{2.2}$$
$$= 10\% * \begin{bmatrix} 0\% \\ 25\% \\ 75\% \end{bmatrix} + 80\% * \begin{bmatrix} 80\% \\ 20\% \\ 0\% \end{bmatrix} + 5\% * \begin{bmatrix} 80\% \\ 20\% \\ 0\% \end{bmatrix} + 5\% * \begin{bmatrix} 0\% \\ 0\% \\ 100\% \end{bmatrix} = \begin{bmatrix} 68\% \\ 19{,}5\% \\ 12{,}5\% \end{bmatrix}$$

Un raisonnement similaire nous donnera les perceptions attendues des autres capteurs dans les mêmes conditions. Pour obtenir la perception attendue au pas de temps suivant, en partant d'un belief-state donné, il suffira donc de calculer le belief-state résultant de l'action choisie, et d'en déduire les probabilités d'observations correspondantes.

A l'instant suivant, on pourra alors comparer les valeurs mesurées par les capteurs du robot à celles qui ont été prédites par le modèle. Néanmoins, choisir un paramètre à modifier pour réduire la distance entre la valeur observée et la distribution de probabilités attendue est un problème difficile. Il est clair que l'on risque de perdre la sémantique des états en échange d'un gain sur la prédictibilité du système.

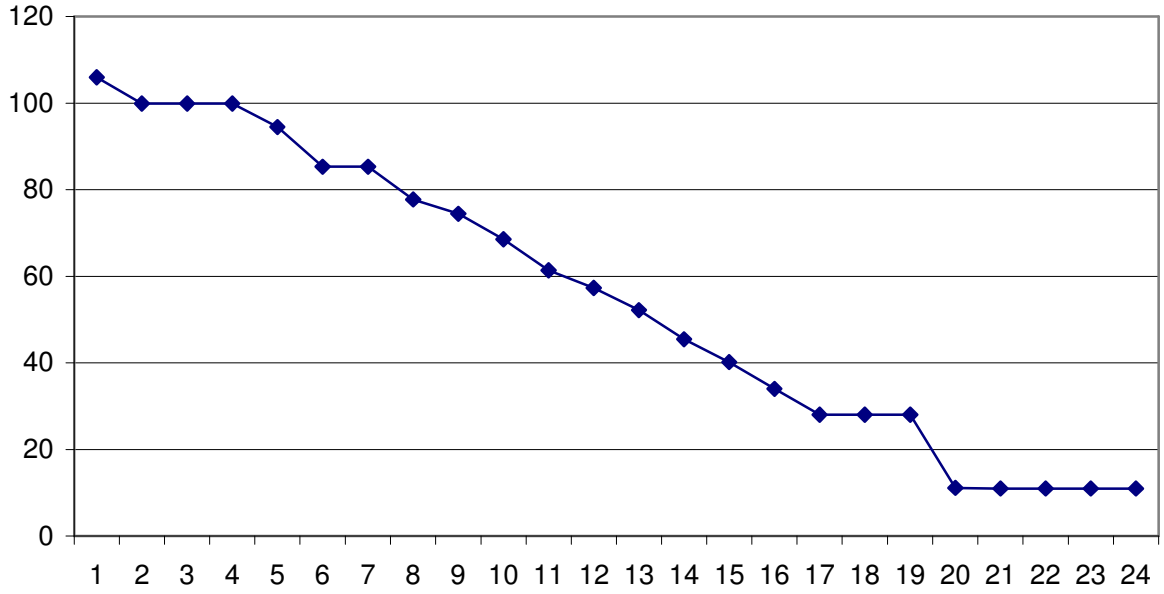


Figure 31 : Mesures à l'a vant lors de la traversée d' un couloir

Actuellement, des expériences sont en cours sur Simplet (voir Annexe 1), notre robot. Dans le but de tester la prédictibilité des capteurs infrarouges, nous avons fait parcourir la largeur d'un couloir (1,60 m) au robot. L'expérience a duré 24 secondes pendant lesquelles le robot avance à vitesse constante, à 5 cm/s. La Figure 31 montre les valeurs brutes relevées par le capteur situé à l'avant gauche du robot. On peut voir en particulier aux temps 18 et 19 que la mesure n'est pas cohérente avec le reste de la courbe. A ce moment, le robot a eu un *moment d'absence* pendant lequel une tâche urgente l'empêchait de mesurer les bonnes distances. En fait, c'est le moment où il est entré en contact avec mur. La priorité était donc de décélérer et s'arrêter. Les capteurs de contacts sont situés 12 centi mètres devant les capteurs infrarouges.

Pour exploiter ce capteur, j'ai utilisé un filtre flou à trois valeurs. Cependant, les fonctions de transfert utilisées sont linéaires et non sigmoïdales. En effet, les valeurs médianes de chaque intervalle (20, 50, et 80 cm) n'ont aucune signification particulière. La Figure 32 montre la projection de ces mesures sur les trois ensembles flous.

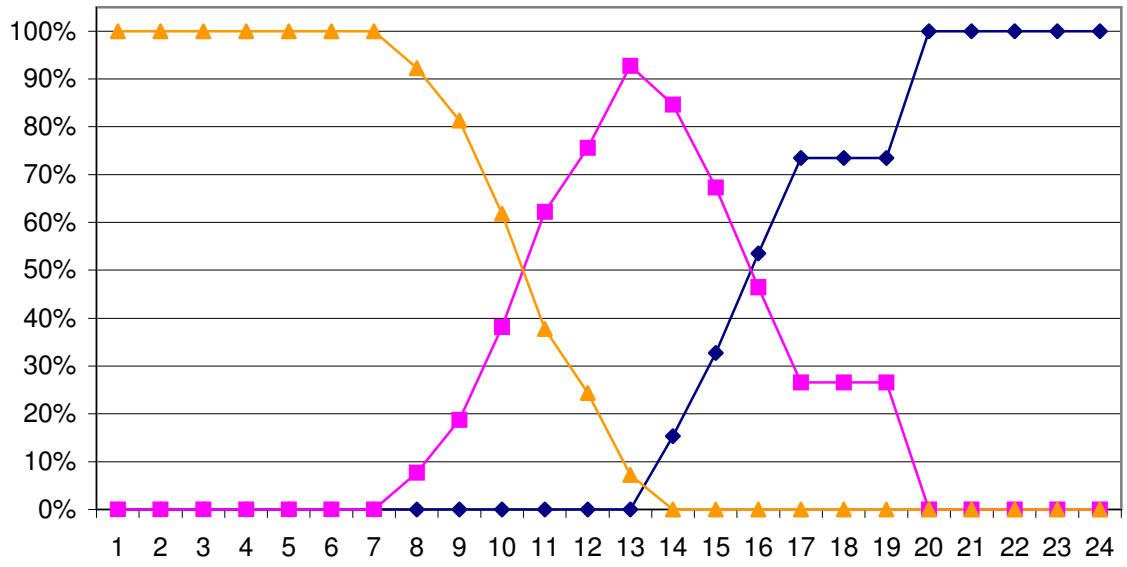


Figure 32 : Mesures floues à l' avant du robot

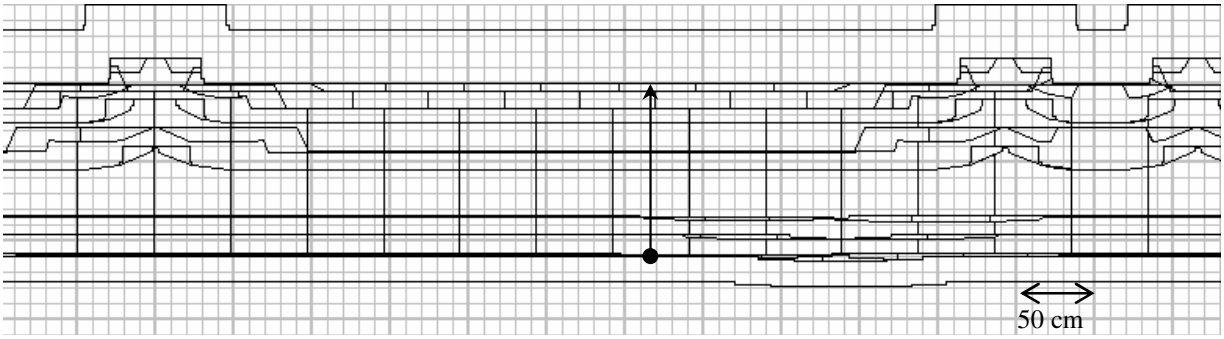


Figure 33 : Configuration de l'expérience

Notre modèle du couloir contient 2442 états, répartis selon 8 directions espacées de 45°. Cela amène notre modèle à 5963364 paramètres pour la matrice de transition, et 29304 pour la matrice d'observation. La portion du modèle correspondant à l'orientation du robot pendant l'expérience est représentée sur la figure Figure 33. Les états ont été choisis de façon à ce que chacun d'entre eux observe toujours la même configuration de capteurs, quelle que soit la position du centre de gravité du robot (à mi-chemin entre les deux roues motrices) dans cet état. Ce faisant, on se rapproche de la vision médicale d'une localisation : le robot est près du mur, à proximité du mur, ou loin du mur.

Pour chacun des capteurs, 5 configurations sont possibles : *Près*, *Proche*, *Moyen*, *Distant*, et *Lointain*. En effet, selon l'orientation réelle du robot (dans l'angle de 45° considéré), un même capteur peut donner diverses valeurs pour une même position dans l'état. Cette considération m'a donc amené à créer des stades intermédiaires entre *Près* et *Moyen*, et entre *Moyen* et *Loin*.

En se restreignant aux seuls états significatifs, ceux atteignant au moins 10% de probabilité à au moins un pas de temps, nos paramètres de transition se réduisent à 36, ce qui est plus raisonnable. Etant donné le nombre limité de données disponibles, il n'est pas raisonnable de tenter l'apprentissage des probabilités d'observation. J'ai donc limité l'algorithme à ces 36 paramètres. Cependant, l'intégralité des paramètres est utilisée pour calculer la prédiction des capteurs. La Figure 34 montre d'une part les valeurs prédites sous forme d'histogramme, et d'autre part les valeurs observées sous forme de courbes. Cette figure se trouve sur la page suivante, de façon à pouvoir lui accoler la figure montrant les résultats après l'apprentissage.

Sur la Figure 34, on voit nettement que les valeurs prédites sont *en retard* sur les valeurs observées. L'adaptation de la matrice de transition devrait donc permettre d'améliorer cette situation. En effet, ce retard montre clairement que le système est trop conservatif : il se contente de suivre les variations imposées par les capteurs. On peut donc supposer que les probabilités de transitions correspondent à un mouvement trop lent.

La Figure 35 montre les valeurs prédites fournies par le modèle appris. Elle ressemble énormément à la Figure 34. Apparemment, l'apprentissage n'a pas pu trouver un modèle satisfaisant. Cependant, considérons attentivement les valeurs prédites, et en particulier pour la valeur moyenne du capteur. On peut alors noter que le sommet, qui se trouvait au temps 15, se retrouve avancé d'une seconde. La décroissance de la courbe est approximativement respectée.

D'autre part, les probabilités affichées aux extrêmes semblent plus tranchées : les valeurs résiduelles sont moins importantes, et la valeur correspondant à la valeur observée est plus proche de 100%. Enfin, l'apparition de la valeur inférieure se fait de façon plus rapide que dans le modèle original. Cela montre donc une meilleure réactivité.

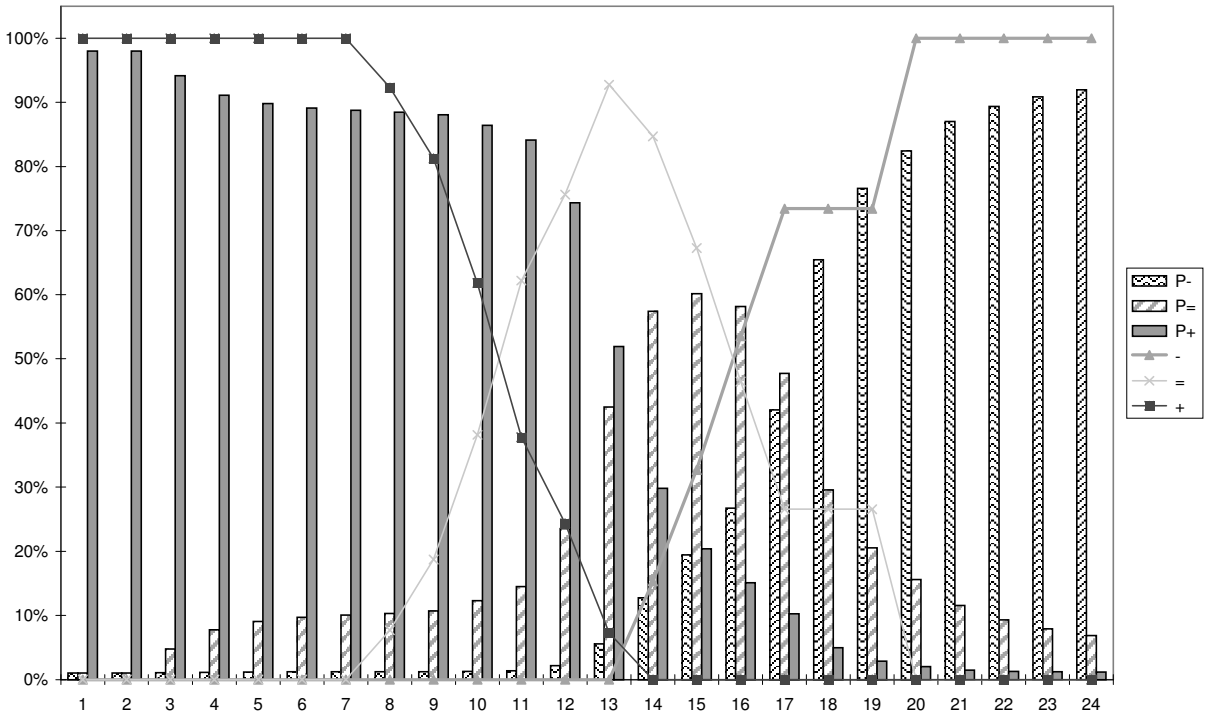


Figure 34 : Comparaison des valeurs prédites et observées

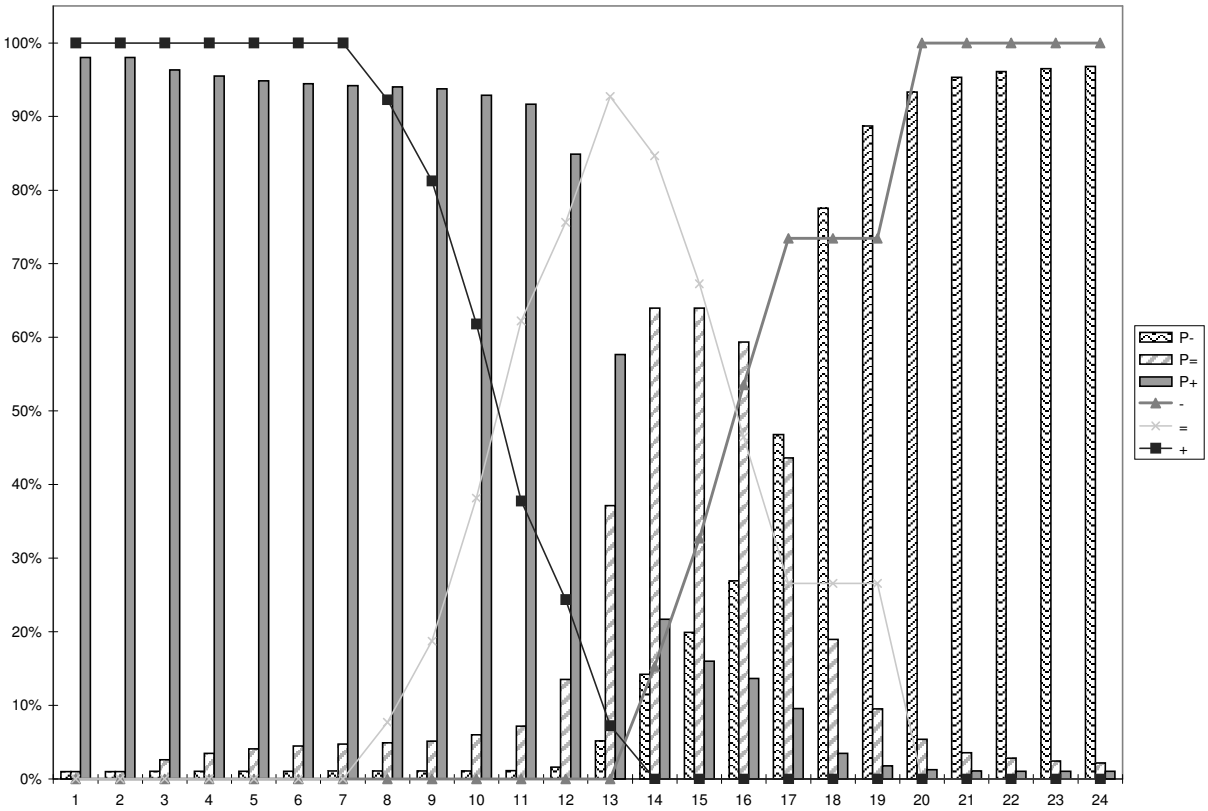


Figure 35 : Valeurs prédites et observées après apprentissage

Quelles conclusions peut on alors en tirer ? A priori, l'algorithme d'apprentissage a réussi à améliorer le modèle initial. Cependant, il a été incapable de corriger le retard observé entre les prédictions et les valeurs réelles. La question qui se pose est alors de savoir si l'on peut gommer cette erreur.

Pour répondre à cette question, j'ai tenté de modifier manuellement le modèle, en essayant de renforcer les transitions permettant de progresser le long du chemin réellement suivi par le robot. Cependant, aucun des résultats obtenus n'a été vraiment probants : lorsque le pic central est avancé dans le temps, le plateau situé au début de l'expérience perd sa saturation : sa probabilité tombe rapidement en dessous de 75%. En effet, en augmentant trop la probabilité de transition, on oblige le système à tenter de changer d'état. L'observation qui arrive ensuite rééquilibre la situation en renforçant l'état que le système vient de quitter. Il s'en suit un équilibre qui abaisse la probabilité de la prédiction en forçant le modèle à considérer un grand nombre d'états potentiels.

Pour conclure, je dirai que, à mon avis, il est difficile d'obtenir une meilleure prédiction sans inclure la dynamique de mouvement réelle du robot. Avec un tel ajout, le modèle pourrait simultanément réagir plus vite aux modifications, mais sans aller trop vite, comme le fait le modèle markovien de temps à autres.

- **Approche dynamique**

Prédire des statistiques sur les valeurs qui seront observées permet de progresser vers une adaptation du modèle à son environnement. Cependant, comme le met en évidence la section précédente, dès que le nombre d'états grossit, le nombre de paramètres à optimiser explose. Pour empêcher cette explosion, je pense que l'idéal est encore de revenir à une représentation continue de l'environnement. J'ai déjà indiqué à plusieurs reprises que cela pouvait conduire à une meilleure intégration des connaissances que nous avons sur le système et son évolution.

Pour ce qui nous intéresse dans cette section, la prédiction des capteurs comme jugement de la validité du modèle, cette approche continue ouvre une nouvelle voie : on pourrait très bien utiliser les lois de la physique pour tenter de prédire les valeurs des capteurs. En effet, à partir d'une position connue, le mouvement imposé par la commande que nous envoyons au robot déplacera ce dernier de façon prévisible.

Cependant, sans connaître la position réelle du robot dans l'environnement, il est difficile de prévoir de manière précise la valeur future des observations, car la perception instantanée est insuffisante pour caractériser la configuration des obstacles. Ainsi, la Figure 36 montre deux des configurations possibles de l'environnement du robot, sachant que les distances instantanées sont connues avec précision. Sur cette figure, le robot est représenté par un rectangle blanc, alors que les murs sont représentés par des blocs hachurés. Les capteurs montrent la distance observée par un segment pointillé, et sont équivalents dans les deux cas.

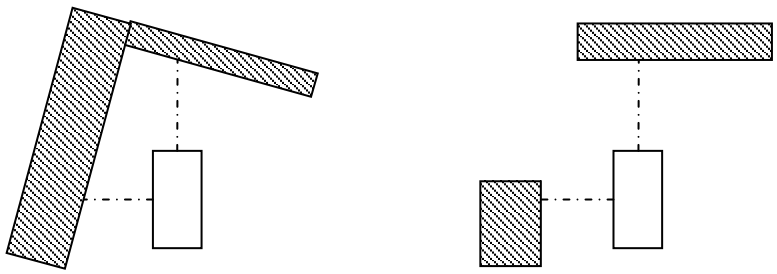


Figure 36 : Une lecture des capteurs, deux configurations différentes

Prenons un cas où le robot avance : ces deux configurations nous conduiront naturellement à deux prédictions très différentes quant à la valeur du capteur gauche du robot, selon que l'on se trouve dans le cas de droite ou dans celui de gauche. Alors que le premier montrera une diminution progressive, le second restera constant un moment, avant que le robot n'atteigne le coin du mur. Après cela, la valeur mesurée sera infinie, puisque plus aucun obstacle ne sera détecté.

La solution à ce problème pourrait être de reconstruire un modèle de l'environnement local, sur la base des valeurs mesurées par les capteurs. Néanmoins, cette reconstruction est difficile à réaliser de façon correcte à cause de l'incertitude portant sur les mouvements et les capteurs. De plus, la solution serait difficilement transposable à d'autres applications telles que la médecine, par exemple.

Une autre approche pourrait être appliquée aux cas où l'environnement est connu. Dans ce cas, il est possible d'obtenir un ensemble de positions probables du robot dans l'environnement (68). Nous pourrions alors utiliser une technique équivalente à celle de la section précédente, en utilisant cette incertitude pour pondérer l'analyse de la qualité de la prédiction.

6.6 Avantages et inconvénients

Dans ce chapitre, j'ai exposé les principales motivations qui nous incitent à permettre à un modèle de diagnostic d'apprendre. En particulier, on y trouve la création du modèle initial ou celui du patient moyen, et surtout l'adaptation de ce modèle à chaque cas particulier.

En effet, les différents cas montrent des divergences par rapport au modèle moyen. Ces différences peuvent être mineures, comme une légère variation de l'état dit « normal ». Ainsi, dans la surveillance d'une anesthésie, certains patients sont bien endormis avec un BIS de 45, alors que d'autres nécessitent un indice de 30 pour une condition optimale d'opération. Cependant, ces différences peuvent aussi avoir une influence très importante. Ainsi, dans la surveillance d'une dialyse péritonéale, un patient qui souffre d'une insuffisance cardiaque aura une tension artérielle qui réagit à l'inverse d'un patient « normal ». Ces deux exemples sont des extrêmes, mais de nombreuses situations intermédiaires apparaissent. Il est donc important de permettre la modification du modèle pour chaque patient. On aboutit donc à la notion de profil pour chaque patient. Créer ce profil requiert le réglage de très nombreux paramètres. Un module d'apprentissage permet alors de trouver les bons paramètres pour chaque cas.

Cependant, les conditions de cet apprentissage sont loin des conditions optimales : les données sont peu nombreuses, et surtout, elles ne représentent pas toutes les conditions possibles. Le risque de sur-apprentissage est donc particulièrement important. Une telle évolution conduit alors le modèle à classer les données selon leur forme, et non selon leur sémantique.

Pour résoudre ce problème, j'ai donc présenté une méthode d'apprentissage de diagnostic basée sur une descente de gradient sans dérivées. Grâce au principe du *bracketing*, c'est-à-dire en encadrant l'optimum de plus en plus précisément, il est alors possible d'optimiser une fonction arbitraire, continue et à support fini, sans autre propriété particulière. En particulier, il est alors possible de minimiser une fonction calculant l'écart entre le diagnostic fourni par le médecin et celui fourni par le système. En travaillant sur cette fonction d'erreur, il est même possible d'obtenir facilement, comme je l'ai montré, un compromis entre le diagnostic du médecin, des propriétés de stabilité numérique, et d'autres paramètres tels que la modification temporelle du modèle cherché.

Cette méthode d'optimisation permet de prendre en compte très facilement des propriétés d'intervalle de validité pour chaque paramètre. De plus, l'utilisation de la relaxation de Powell permet d'accélérer la convergence, mais au détriment du respect de ces intervalles de validité. J'ai donc mis au point une méthode permettant de conjuguer ces deux propriétés, tout en garantissant la non-dégénérescence du modèle. L'intérêt majeur de l'approche que je propose est qu'elle permet d'apprendre des paramètres dont les bornes sont liées, tels que les paramètres d'un modèle statistique.

Néanmoins, l'apprentissage de diagnostic par descente de gradient montre rapidement ses limites dans les cas où le temps est compté. En effet, elle nécessite la collaboration active d'un expert humain pour corriger le diagnostic proposé par le système. Or, cette correction est un processus long et fastidieux, surtout si le modèle contient de nombreux états, ou si la période de temps sur laquelle porte l'apprentissage est longue.

Dans ce cas, je propose de rechercher une solution dans la prédiction de valeurs mesurables. Néanmoins, cette approche se heurte au problème de la définition d'un module de prédiction fiable qui se base sur les paramètres, ou le résultat du modèle à apprendre. Dans l'hypothèse de l'existence d'un tel module, il serait possible de substituer la comparaison de la valeur prédite avec la valeur observée à la comparaison du diagnostic du médecin avec celui du système. A ce moment, il deviendrait possible d'utiliser à nouveau les mêmes algorithmes d'apprentissage, sans pour autant entraver le travail de l'expert humain.

Cependant, cette dernière approche repose le problème de la sémantique du diagnostic. En effet, ce n'est pas parce que le système est prédictif que les conclusions qu'il dégage sont pertinentes. Il reste donc encore une grande marge à couvrir avant de ne pouvoir utiliser de tels algorithmes à des problèmes critiques, telle la médecine.

7 Une architecture générique

7.1 Motivation

Tout au long des chapitres de ce mémoire, j'ai présenté des algorithmes visant à résoudre des problématiques variées, allant du diagnostic à la prise de décision, en passant par l'apprentissage et l'adaptation des modèles. J'ai montré à plusieurs reprises que l'architecture était applicable à une multitude de problèmes, et j'ai détaillé plus spécifiquement les trois applications qui me servent d'appui : le suivi à distance de dialysés en DPCA, la surveillance d'une anesthésie, et la navigation d'un robot mobile.

Cependant, même si j'annonce que l'architecture est commune, voire même universelle, cette architecture est restée jusqu'à présent dans le vague. Je vais donc présenter cette dernière en détail, avant d'introduire un outil que j'ai réalisé durant ces trois années. Cet outil, réalisé en Java, permet en effet de construire un système basé sur cette architecture de façon presque automatique. Il ne reste alors au développeur qu'à s'occuper des aspects d 'intégration de ce module dans son application.

Pour le chercheur, cette application permet de tester diverses approches pour un problème donné, sans avoir à réécrire la totalité des algorithmes, avec les risques d'erreur et les délais que cela implique.

7.2 Un générateur d'agents intelligents

7.2.1 Panorama sur les outils de développement existants

J'ai travaillé depuis plusieurs années avec les outils de conception i ntégrés commercialisés par la société Borland. Tout au long de ces années, j'ai donc pu apprécier les fonctions qu'offrent ces outils dans le domaine de la programmation. En particulier, le système de développement par composants, déposés à la souris sur des fenêtres ou des boîtes de dialogue, permet de développer en un temps record des interfaces graphiques de qualité. Ce développement cache au programmeur la complexité de la tâche sous-jacente : création des pompes à messages, des appels systèmes, des fonctions de renvoi, de l'agencement en termes de coordonnées, et ainsi de suite.

Parallèlement, avec le système Direct-Show, Microsoft propose une palette d'outils permettant de créer des filtres vidéo complexes à partir de modules de base. Le lien entre ces modules est fait par un système de flux de données, chargés de convoyer ces dernières de module en module.

Au cours de ma thèse, il est apparu que, au fur et à mesure des progrès réalisés, le système que j'ai développé pouvait s'appliquer à quantité de pr oblèmes. Néanmoins, la quantité de code correspondant à son adaptation à d'autres projets est relativement importante. Cela implique que le temps nécessaire à la création d'une nouvelle application est souvent non négligeable. J'ai donc décidé de réaliser tout d'abord une boîte à outils permettant de mettre en commun ce code. Cette bibliothèque a ensuite évolué progressivement vers une interface de développement graphique permettant de développer de façon visuelle un agent intelligent, pratiquement sans écrire une seule ligne de programme.

Cet outil a été initialement développé par Sen Debasish au cours d'un stage à vocation industrielle que j'ai encadré. J'ai ensuite repris son travail, en l'améliorant et en l'étoffant de nombreuses fonctions non prévues à l'origine.

7.2.2 Conception par Objets

7.2.2.1 Principe

La bibliothèque sur laquelle s'appuie l'interface graphique a été conçue selon les principes du développement par objets (7). Chaque modèle est implanté sous la forme d'un objet Java, exposant ses fonctionnalités à travers une série d'interfaces publiques documentées. De même, chacun des algorithmes prend la forme d'un objet indépendant qui fait appel aux fonctions publiées des modèles dont il a besoin. Même si Java n'est pas le meilleur des langages à objets, sa distribution multiplateforme mondiale en fait un choix intéressant.

Ce mode de conception peut donc sembler un peu lourd, puisqu'il nécessite la création de nombreux objets, ainsi que leur collaboration, pour pouvoir réaliser une tâche que l'on pourrait créer d'un seul tenant. Cependant, l'expérience montre qu'une réalisation modulaire est un gage de qualité (46), surtout en ce qui concerne l'évolution du projet au cours du temps. En effet, le polymorphisme des objets permet à un module d'endosser un rôle précis, sans spécifier la façon dont ce rôle est atteint. Ainsi, par exemple, une modification du système perceptif de l'agent peut être faite sans avoir à vérifier le détail des algorithmes de reconnaissance, ou de planification. De même, le changement de l'algorithme de reconnaissance et de localisation peut être effectué sans pour autant modifier le modèle, l'affichage ou la perception de l'agent.

Un autre des avantages du développement par objets est que cela permet d'isoler facilement chaque module des autres. Cette séparation est donc non seulement un gage de sécurité, puisque les modifications d'un module n'influeront pas directement sur les autres, mais aussi en extensibilité, puisque l'on peut ajouter sans peine de nouveaux objets à la bibliothèque. De plus, cela simplifie le développement de chaque module, puisque seule la partie du projet en relation directe avec cet objet est concernée.

7.2.2.2 Interfaces des objets Markoviens

Un modèle Markovien discret est un module basé sur un nombre fini d'états, et obéissant à des règles d'évolution exprimées sous la forme de transitions probabilistes. Son implantation requiert donc un objet mettant à disposition des états, avec une distribution initiale, et des actions, avec les matrices de transitions associées.

Un processus Markovien permet quant à lui de prédire l'évolution d'un système, connaissant le modèle associé à ce dernier, et l'expression de l'état du système sous forme d'un belief-state. Il utilisera donc le modèle Markovien présenté ci-dessus, et offrira de calculer un nouveau belief-state correspondant à une action donnée. La Figure 37 résume l'interaction de ces deux objets.

Un processus Markovien partiellement observable aura donc une interface équivalente, mais où l'on ajoute une notion d'observation. Afin de rester le plus général possible, aucune supposition ne peut être faite sur la forme de ces observations. La formulation la plus universelle que j'ai trouvée correspond donc à ne prendre en entrée que les distributions exprimant la probabilité de l'observation dans chacun des états. Tout cela est résumé sur la Figure 38.

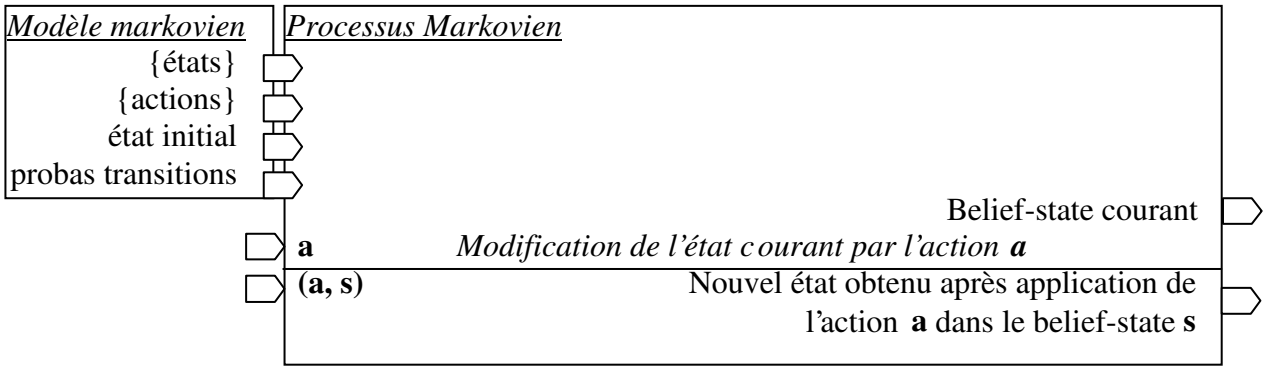


Figure 37 : Interface d' un processus Markovien

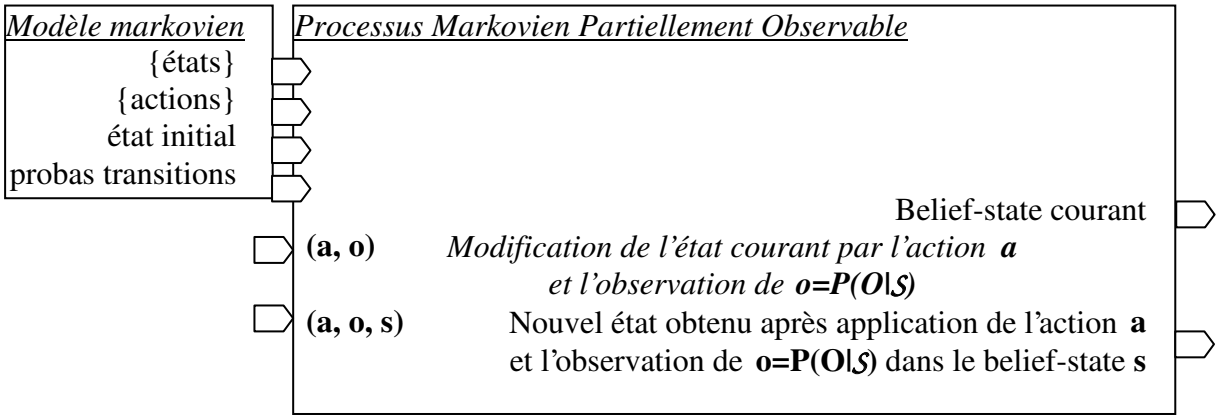


Figure 38 : Interface d' un processus Markovien partiellement observable

Afin de disposer des données correspondant aux observations, il m'a fallu concevoir un module fournisseur. Actuellement, ce fournisseur ne fait que fusionner les probabilités fournies par les divers capteurs, sur la base d'un jeu de capteurs statistiquement indépendants. Néanmoins, il est possible d'y adjoindre tout traitement nécessaire. Ainsi, ajouter une matrice de covariance permettrait de prendre en compte des capteurs non indépendants.

Chaque capteur est donc chargé de fournir une probabilité d'observation pour chacun des états du modèle. Cette répartition peut paraître étrange, puisque les probabilités d'observation des divers capteurs ne font pas partie du modèle. Néanmoins, cela permet de gérer des capteurs de types différents de manière beaucoup plus simple, car chaque capteur peut utiliser son formalisme propre. Même si l'on éparpille quelque peu les données du modèle, on gagne donc beaucoup au niveau de l'expressivité du système. Finalement, si une application donnée n'utilise qu'un seul et unique type de capteur, il est possible d'utiliser le polymorphisme pour regrouper toutes les probabilités d'observation dans le modèle. Il suffit de créer un nouvel objet, qui héritera du modèle standard en ajoutant ces probabilités. Le polymorphisme permet alors d'utiliser ce nouveau modèle à tout emplacement où le modèle standard est attendu. Cependant, les capteurs peuvent alors y faire appel pour obtenir les probabilités dont ils ont besoin.

7.2.2.3 Interfaces des modules liés à l'apprentissage

Dans la bibliothèque, il existe deux types d'apprentissages : ceux qui sont intrinsèquement liés à un modèle particulier, et les autres. Il est recommandé que les algorithmes liés à un modèle particulier soient intégrés à ces derniers. Par contre, les algorithmes d'apprentissage plus généraux sont intégrés sous forme d'objets.

Dans la première de ces classes, on trouve naturellement l'algorithme de Baum&Welsh. Son intégration est un peu particulière, puisqu'il met en jeu plusieurs modules. En effet, il procède tout d'abord à une reconnaissance. Le résultat de cette dernière est alors utilisé pour ré-estimer un nouveau modèle. La méthode utilise donc l'algorithme *forward-backward*, le modèle Markovien, et les différents capteurs.

Le module forward-backward intègrera le cœur de l'algorithme de Baum&Welsh. Il se chargera alors de répercuter les modifications du modèle sur le module correspondant, en appelant simplement les fonctions de modifications de ce dernier. Finalement, il fera appel à une fonction particulière, dédiée à l'apprentissage, de son capteur en lui passant en paramètre la suite des belief-states correspondant aux observations reçues.

En retour, le capteur se chargera d'adapter son modèle de perception en utilisant une méthode adaptée à ce capteur. Ainsi, un capteur discret pourra utiliser la formule standard de l'algorithme de Baum&Welsh, alors qu'un capteur flou utilisera plutôt la formule donnée par Koenig & Simmons. Un capteur purement continu pourra lui aussi utiliser une méthode adaptée à sa représentation.

Pour ce qui concerne les algorithmes généraux, chacun dispose de son jeu d'objets, indépendamment des autres. Actuellement, le seul algorithme implanté correspond à la descente de gradient. Pour implanter une descente de gradient pertinente, il faut deux objets principaux : celui qui contient l'algorithme d'apprentissage, et celui qui contient le modèle à optimiser.

Le premier est relativement simple, puisqu'il ne fait appel qu'au second, ainsi qu'à la source de données utilisée. Le modèle à optimiser est implanté sous la forme d'un objet *Fonction*. Cet objet a deux grandes parties : une partie *valeur* qui contient la valeur actuelle de la fonction à minimiser, et une partie *paramètres* qui contient l'ensemble des variables modifiables pour agir sur la valeur. Comme je l'ai signalé dans le chapitre sur l'apprentissage, les bornes du domaine de validité de chaque paramètre sont très importantes pour garantir la cohérence du modèle. Chaque paramètre s'accompagne donc de son domaine de validité, mais aussi d'une fonction permettant de calculer le domaine correspondant à un vecteur de descente donné.

Lorsque les paramètres sont répartis dans plusieurs modules, un nouvel objet permet de s'occuper de répartir les données : l'objet *multifonction*. Ce dernier permet donc de fusionner les paramètres de plusieurs modules indépendants, de récupérer la valeur fournie par un autre module, et présenter tout cela comme s'il s'agissait d'un module unique disposant d'une multitude de paramètres. Cet objet hérite donc de l'objet Fonction, comme tout module optimisable. Un autre objet a été introduit, de façon à permettre les optimisations discutées dans le chapitre portant sur l'apprentissage : l'objet *AntiEscaliers*. Ce dernier se charge donc de produire l'accélération de la descente de gradient en court-circuitant les descentes en escaliers.

Pour ce faire, il utilise une autre fonction à optimiser, dont il publie toutes les caractéristiques comme si c'étaient les siennes. Il se contente d'y ajouter un paramètre virtuel qui correspond à la descente cumulée que j'ai présentée. Intercaler cet objet dans la chaîne d'apprentissage permet donc d'ajouter localement un raccourci potentiel. On peut alors regrouper ensemble les paramètres que l'on suppose interdépendants, comme par exemple toutes les probabilités d'une même distribution. Ce module fournira alors un paramètre permettant d'optimiser ces valeurs-là ensemble, sans s'occuper des paramètres qui n'ont rien à voir avec ces dernières, mais qui pourraient limiter l'optimisation.

Considérons maintenant les autres modules de la bibliothèque, et leurs relations avec ceux dédiés à l'apprentissage général : l'algorithme forward-backward fournit une valeur correspondant à la probabilité que le modèle génère une suite d'observations donnée et ne propose aucun paramètre. De la même façon, l'algorithme de Viterbi fournit la probabilité du meilleur chemin, mais sans proposer de paramètre non plus. Enfin, un module comparateur permet de calculer la différence entre deux suites de belief-states, et propose donc comme fonction à optimiser la valeur cumulée de ces écarts. Ce module de comparaison a des paramètres (atténuation du passé, des petites variations,…) , mais l'algorithme d'optimisation ne peut les modifier. Il n'offre donc aucun paramètre à optimiser.

D'un autre côté, le modèle Markovien ne fournit aucune valeur (il renvoie toujours 0), mais présente un paramètre pour chaque probabilité de l'état initial, et un pour chacune des transitions de chacune des actions. De manière similaire, chacun des capteurs fournit un paramètre pour chaque probabilité d'observation, mais il peut aussi y ajouter des paramètres qui lui sont propres. Ainsi, les capteurs sigmoïdaux de la surveillance d'une anesthésie proposent également un paramètre pour chacune des bornes des ensembles flous.

Chacun de ces modules hérite donc de façon directe de l'objet Fonction. Ils sont ainsi directement utilisables par l'algorithme d'optimisation, et cet héritage pose peu de contraintes sur le code des objets concernés. Conduire une descente de gradient dans ces conditions est donc relativement simple, mais un détail mérite d'être soulevé : la modification de l'un des paramètres de l'un des capteurs, par exemple, nécessite de réinjecter la séquence complète des observations pour pouvoir calculer la valeur de la fonction. C'est pour cette raison que le module d'apprentissage possède un lien vers la source des données : lorsqu'un paramètre a été modifié, les données sont réinjectées dans tout le processus afin de prendre la modification en compte. A ce moment, les nouveaux paramètres sont utilisés à tous les niveaux, et la valeur de la fonction est mise à jour, prête à être fournie au module d'apprentissage.

7.2.2.4 Autres modules de la bibliothèque

Le reste de la bibliothèque est constitué de modules d'usage général, sans lien particulier avec l'intelligence artificielle. Je ne vais donc pas les détailler aussi précisément que les précédents. On trouve parmi ceux-ci :

- des afficheurs graphiques, capables d'afficher des courbes quelconques
- des mémoires pouvant servir de source pour un apprentissage futur
- des filtres divers tels, par exemple, ceux utilisés dans notre modèle perceptif.

7.2.3 Notion de flux

Dans les applications de diagnostic, ou de localisation, l'aspect perceptif est très important, comme je l'ai déjà signalé dans les sections précédentes. En fait, le processus est guidé par ces données. J'ai donc décidé de construire l'architecture du modèle autour de ces données. Ainsi, l'arrivée d'une série de données va déclencher automatiquement les calculs nécessaires à leur traitement, ce qui va entraîner la création d'informations nouvelles. Ces informations, à leur tour, vont alors pouvoir servir de donnée à des traitements ultérieurs.

On aboutit donc au déclenchement en chaîne des divers modules qui constituent l'agent. Ces déclenchements s'orchestrent autour de la progression des données de module en module. Les liens qui relient ces derniers véhiculeront donc les données produites par le module source vers le module destination. L'arrivée d'une observation va donc entraîner la création d'un *flux de données* qui va progresser de module en module, chacun d'entre eux opérant un traitement avant de transférer le résultat obtenu au suivant.

Cette évolution, guidée par la progression des données à travers le graphe des modules de traitement, permet donc de représenter assez simplement le modèle de calcul d'un agent. Au niveau de l'implantation sur machine, cette structure peut être réalisée très simplement, puisque le transfert des données peut être fait par un simple appel de méthode. Les données en question sont passées en argument de cet appel, et ce dernier déclenche les calculs nécessaires à la production du flux de sortie de ce module. Ce dernier transmettra les données correspondantes au module suivant de la même façon.

Cependant, le fait que les calculs soient guidés par les données pose de nouveaux problèmes. En effet, il arrive souvent que certains modules nécessitent plusieurs données différentes pour pouvoir travailler. On doit alors implanter des files d'attentes où les données arrivées vont patienter en attendant l'arrivée des données manquantes. L'appel de méthode se termine alors, sans créer de nouveau flux de données en sortie. A ce moment, le module ayant envoyé ses données récupère la main et peut procéder à la suite de ses opérations. Par exemple, il pourrait envoyer un second flux de données à destination d'un nouveau module. Lorsque toutes ses opérations sont terminées, il rend à son tour la main au module qui l'avait activé. Cela se poursuit jusqu'à ce que le premier module de la chaîne ait fini ses opérations. A ce moment, le calcul est fini et l'exécution de l'appelant peut reprendre.

On voit donc que l'appelant, celui qui fournit les données initiales au modèle, reste en attente tout au long du calcul des divers modules de traitement. Cela peut être préjudiciable, en particulier si le système diagnostiqué évolue rapidement. Ainsi, dans l'application surveillant une anesthésie, aucune acquisition de donnée ne peut être faite pendant que le modèle calcule. Toute donnée qui arrive durant ce laps de temps risque donc d'être perdue ou, au minimum, de prendre du retard. Pour éviter cette situation, j'ai implanté un module d'un type particulier, donc le seul rôle est de servir d'intermédiaire entre le processus de calcul et le processus d'acquisition. Il s'agit donc d'une tâche indépendante, dont l'exécution se déroule en parallèle avec celle du processus d'acquisition. Les données sont alors stockées par ce dernier, qui retourne immédiatement à ses occupations. Lorsque le module-tampon dispose d'un peu de ressources de calcul allouées par le système, il va transférer les données qui ont été mises en attente aux modules de calcul. Il va donc servir à son tour de source de données, attendant que les calculs s'achèvent avant d'envoyer les données suivantes.

Un autre problème est lié à l'activation des modules par l'arrivée des données ; il s'agit en fait de la non-arrivée de l'une des données. En effet, puisqu'il est possible que certains modules attendent d'avoir reçu leurs données avant d'exécuter leurs calculs, ces derniers vont attendre indéfiniment des données qui n'arrivent pas. A ce moment, plus aucun calcul n'est effectué en aval de ces modules. Pour chacun de ces modules, deux cas se présentent : soit il est capable de travailler avec une partie des données, soit il lui faut absolument que toutes soient présentes. Dans ce dernier cas, on ne peut rien faire, car il nous est impossible d'inventer les données manquantes. Dans le cas où l'on peut s'accommoder de données partielles, on peut tirer parti de celles qui sont arrivées, quitte à procéder à une correction si de nouvelles données arrivent. Pour cela, j'ai décidé d'associer à chacune des données véhiculées par les flux une date, la date à laquelle la donnée est arrivée dans le système. De cette façon, il est possible de dire si deux données, provenant de flux différents, sont associées, ou si au contraire elles représentent deux événements distincts.

L'application la plus flagrante de cette fonctionnalité tient à la fusion de plusieurs capteurs indépendants. En effet, chacun d'entre eux peut arriver dans le système de façon arbitraire. Attendre que toutes les données soient arrivées permet donc de les re-synchroniser, mais avec les travers que j'ai présenté dans le cadre du projet Anesthésie. Le problème qui m'intéresse ici tient plus au risque d'un manque de données.

En effet, un aléa peut endommager les données de l'un des capteurs, les rendant inutilisables. Dans ce cas, le module de fusion de données reste en attente, puisque l'une d'entre elles manque à l'appel. Lorsque cette donnée arrivera au prochain pas de temps, elle pourra alors être associée aux données des autres capteurs. A ce moment, le paquet de données est débloqué, mais les données des autres capteurs sont à nouveau mises en attente, puisqu'il manque de nouveau une donnée. D'une part, le problème de blocage se poursuit, et d'autre part les données associées recouvrent des périodes de temps différentes, ce qui est inacceptable.

A l'inverse, l'utilisation de la date associée à chaque donnée permet de différentier ces deux événements en isolant les données appartenant à chacun d'entre eux. La réception d'une donnée ayant une date différente entraînera alors le traitement des données précédemment reçues, l'envoi du résultat correspondant sur le flux de sortie du module en question, et la mise en attente de la donnée courante. On peut donc voir tout d'abord que les données associées les unes aux autres restent cohérentes au niveau temporel, mais surtout que le problème de blocage est résolu. En effet, comme la nouvelle donnée reçue n'est pas associée au paquet précédent, elle ne manquera pas au paquet suivant. Enfin, le manque d'une donnée peut être pris en compte dans le processus de traitement ou de fusion, de façon à ne pas biaiser ce dernier.

Néanmoins, un nouvel effet pervers se produit si l'on utilise cette méthode telle quelle : les données associées à la donnée manquante ne vont être traitées qu'au pas de temps suivant, lorsque l'on est sûr que celle qui bloque le processus n'arrivera plus. Pour contrer cet effet, on peut mettre en œuvre un traitement incrémental, au fur et à mesure que les données arrivent lorsque c'est possible. Ainsi, chaque nouvelle donnée reçue affine le résultat de la précédente. Cependant, cela entraîne aussi le déclenchement de tout le graphe de calculs en aval de ce module. Tous ces calculs vont donc être refaits à chaque arrivée de donnée. Dans le cas où les données arrivent de façon simultanée, comme dans le projet DIATELIC ou le projet de robotique, cet effet peut être très gênant. En effet, il entraîne le gaspillage des ressources de calcul disponibles en traitant des données incomplètes alors que les autres sont en attente. L'agent va donc traiter chacune des données qui arrivent comme si c'était la dernière et qu'il fallait absolument obtenir un résultat.

Pour résoudre ce dernier problème, j'ai donc conçu une méthode de verrouillage sélectif. Cette technique permet à la source de données de fournir des paquets cohérents au point de vue temporel. Pour cela, elle verrouille tout d'abord chacun des modules auxquels elle est reliée. Elle injecte ensuite ses données les unes à la suite des autres, comme si de rien n'était. Elle n'enverra le signal de déverrouillage que lorsque les données en attente présenteront une date différente de celles qui viennent d'être envoyées. Lorsqu'un module reçoit un signal de verrouillage, il doit donc le propager aux modules suivants et mémoriser l'information s'il en a besoin. Ainsi, un module ne nécessitant pas les données de plusieurs flux simultanés traitera les données qu'il recevra comme si de rien n'était, tout en propageant les résultats de ses traitements au fur et à mesure.

Par contre, un module qui utilise les données d'au moins deux flux pourra tenir compte de l'information de verrouillage afin de ne pas surcharger le processus avec des données incomplètes. En effet, lorsqu'un module verrouillé reçoit des données, il peut anticiper l'arrivée future de nouvelles données. Ainsi, il ne transmettra pas de donnée incomplète aux modules suivants, puisqu'il sait que plusieurs données simultanées sont en attentes. Il est alors fort possible que les données qui lui manquent soient parmi celles-ci. Lorsque ce module reçoit le signal de déverrouillage, à l'inverse, il sait qu'aucune nouvelle donnée n'est disponible. Du moins, si de nouvelles données arrivent, elles auront une date différente.

Il est alors judicieux de transmettre le résultat des calculs, même si les données sont incomplètes. Ces dernières ne seront en effet jamais complétées par des données temporellement cohérentes. Lorsque les données sont traitées, et le résultat envoyé, le module peut ensuite transmettre l'ordre de déverrouillage aux modules suivants. Ces derniers réitéreront alors le processus à leur tour, et ainsi de suite jusqu'à ce que tous les modules aient terminé leurs calculs.

Bien sûr, un module qui envoie ses résultats sur plusieurs flux doit utiliser le même système de verrouillage, puisqu'il sert temporairement de source pour les modules en aval.

7.2.4 Modules spécifiques au traitement des flux

Afin d'assurer la cohérence et l'expressivité du système, il a été nécessaire d'ajouter des objets à la bibliothèque, en plus de la simple traduction des modules intelligents présentés précédemment. Parmi ces nouveaux modules, j'ai déjà cité une source de données dotée d'un processus indépendant, mais on peut en ajouter beaucoup d'autres :

- Des points de sorties mettant à disposition de l'application les données reçues.
- Des moyennes mobiles permettant de calculer une moyenne sur une fenêtre de données de taille fixe. Ainsi, toute nouvelle donnée reçue chasse la plus ancienne de la fenêtre, ce qui entraîne une modification de la valeur moyenne.
- Des démultiplexeurs qui permettent de distribuer un même flux de données vers plusieurs modules. Ainsi, ce module permet au projet Anesthésie d'afficher les données reçues par le système en même temps qu'on les analyse.
- Des modules-fichier permettant de sauvegarder les données dans un fichier, et ainsi de garder une trace datée des flux ayant transité sur ce flux.

7.2.5 Une application complète

La bibliothèque de composants dits intelligents possède donc une structure propre, basée sur la conjonction de modules de traitement et de *boîtes* qui englobent ces modules en leur associant la notion de flux. La mise en œuvre de cette bibliothèque dans une application donnée ne nécessite donc que d'instancier chacun des modules, puis de les connecter par des flux. On constate donc rapidement qu'une grande partie du code Java nécessaire à ce travail est strictement identique d'une application à l'autre, et de plus très répétitive au sein d'un même projet. Nous avons donc développé une interface intuitive permettant de générer tout ce code automatiquement. Cette interface a progressivement évolué pour inclure de plus en plus de modules, grâce à des fonctionnalités toujours plus poussées.

Actuellement, le développement d'une application chargée de la surveillance d'un système dynamique réel consiste donc à choisir un à un les modules de la bibliothèque, à les configurer et à les relier par des flux. Toute cette opération se fait à la souris, car chacun de ces modules apparaît sous la forme d'une boîte noire munie de broches de connexion. Ces broches sont de trois types, et leur représentation graphique indique à l'utilisateur s'il s'agit d'une broche d'entrée, de sortie, ou d'utilisation. Le premier type de ces broches est destiné à recevoir les données fournies par l'une des broches de sortie d'un autre module. Les broches d'utilisation montrent quant à elles que le module en question nécessite la collaboration d'un autre module, d'un type très précis. Ainsi, un module implantant l'algorithme de Viterbi nécessitera bien évidemment un modèle markovien. Ces deux modules seront alors reliés par un lien d'utilisation, et non un flux de données.

Outre les fonctions habituelles de sauvegarde et de chargement d'un diagramme, on y trouve donc également les fonctions de création, de modification, et de suppression de modules. L'utilisateur dispose également de fonctions permettant d'arranger l'espace de travail en changeant la position, la taille, le nom, et la couleur de chacun des modules. Il est ainsi possible d'exprimer de façon visuelle la structure complète de l'agent. Bien entendu, il est possible de modifier à tout instant une quelconque propriété d'un module. Cependant, cela peut avoir une influence sur les entrées/sorties de ce module, en particulier en modifiant le nombre de broches utilisables. Certaines de ces fonctions sont donc restreintes pour garantir la sémantique du diagramme actuel.

A ces fonctions, relativement classiques, s'ajoute finalement la fonction de compilation. Cette fonctionnalité va donc traduire chacun des modules en une classe Java. Pour y parvenir, elle génère donc un squelette de classe habituel, et remplit son constructeur en instanciant successivement chacun des modules. Finalement, cette compilation se termine par l'ajout du code nécessaire à la liaison des divers modules, et par la fermeture des divers blocs ouverts. En définitive, l'utilisateur obtient donc une classe prête à compiler, et à intégrer dans un projet plus important. Dans ce dernier, il lui restera à instancier cette nouvelle classe, et à fournir les données aux modules concernés.

En effet, cette dernière fonction est trop complexe pour être générée automatiquement. Selon les applications, les données peuvent provenir de *ports série*, de *sockets réseau*, de fichiers, d'entrées de l'utilisateur, ou de traitements faits par le programme. Certaines de ces sources pourraient être ajoutées directement à la bibliothèque, mais la multitude de formats de données existants rend cette tâche particulièrement difficile. Pour cette raison, j'ai choisi de laisser cette option de côté pour m'intéresser à des fonctions plus utiles.

7.2.6 Une structure ouverte

La structure de la bibliothèque a été voulue la plus ouverte possible. En effet, étant une bibliothèque à destination des chercheurs, les modules qui la composent sont naturellement voués à évoluer au fur et à mesure des études conduites sur le sujet. Non seulement les modules existants peuvent être améliorés et leurs fonctions étendues, mais en plus, de nouveaux modules peuvent être ajoutés de manière très simple.

Pour atteindre cet objectif d'ouverture, j'ai utilisé les interfaces Java, de même que la notion de réflexivité. En effet, les interfaces sont une forme de *contrats* qui expriment ce qu'un objet doit être capable de réaliser, et sous quelle forme. Elles permettent donc de garantir certaines fonctionnalités sans pour autant contraindre leur implantation réelle. Ainsi, l'interface *IFlux*, qui est à la base de l'architecture complète, définit les fonctions d'ajout de données, les routines de connexion de deux modules, la gestion du verrouillage, et la vérification du type des broches du module. En effet, pour pouvoir connecter deux modules ensemble, il est nécessaire de s'assurer que la sortie du module source est compatible avec l'entrée du module destination. Pour cela, chaque module peut indiquer le type de chacune de ses connexions, que cela soit en entrée, en sortie, ou en utilisation. Si l'on ne vérifiait pas cette compatibilité à la connexion, il faudrait alors la vérifier pour chaque réception de donnée, ou risquer de déclencher une exception Java de manière incontrôlée. Pour ménager les problèmes de robustesse et d'efficacité, j'ai donc décidé de vérifier cette compatibilité à la connexion.

La notion de réflexivité permet quant à elle à un objet Java d'interroger les fonctionnalités d'un autre objet pendant l'exécution. Cette fonction permet donc de créer un objet sans connaître la structure exacte de son constructeur, par exemple. Cette fonction autorise donc l'interface graphique de construction à utiliser des objets indisponibles à sa construction.

Pour être plus précis, je vais prendre un exemple : imaginons que, demain, un collègue écrive un module capable d'intégrer un système à base de règles tel que Clips. Sous réserve que cet objet respecte certaines interfaces, il me suffit alors d'ajouter le nom de la classe correspondante à la liste des modules utilisables dans l'interface de création pour pouvoir l'utiliser. Il n'est nul besoin de recompiler aucune classe, ni même de posséder le code source de l'interface graphique ou du module. La jonction est faite entièrement à l'exécution.

Pour les besoins de la gestion de cette interface graphique, il est donc nécessaire que chacune des boîtes utilisables respecte certaines normes. Ces normes prennent la forme d'une nouvelle interface de classe Java, l'interface *BoxModelObject*.

Cette interface contient donc toute la partie spécifique à la partie dédiée à la conception. Elle contient donc les fonctions suivantes :

- Sauvegarde des données dans un fichier.
  Cette routine stocke les paramètres du module actuel dans un fichier texte, dont la structure est inspirée des fichiers à balises, tels que XML. Ce format permet donc une évolutivité importante et un passage simple d'une architecture matérielle à une autre.

- Restauration de l'objet à partir d'une sauvegarde .
  De façon triviale, cette fonction est la réciproque de la fonction de sauvegarde.

- Génération du code java nécessaire à la création de ce module.
  Comme je l'ai indiqué dans la section précédente, l'application peut compiler les modules sous la forme d'une classe Java. Pour y parvenir, tout en respectant les contraintes d'ouverture, il est impossible de générer ce code dans l'application elle-même. On utilise alors le grand principe de la conception par objets : puisqu'un module a toutes les informations nécessaires pour exprimer son état, c'est lui qui est le plus à même d'en tirer parti pour générer son propre code. On demande donc successivement à chacun des modules de s'auto-compiler en java, à destination d'un fichier passé en paramètre. Ainsi, chaque nouveau module vient avec une fonction de compilation adaptée à ce module. L'objectif d'ouverture est donc atteint à ce niveau.

- Génération du code java permettant de le relier aux autres modules.
  De façon similaire au point précédent, cette fonction génère du code Java. Cette fois, on s'intéresse à relier les modules les uns aux autres. Il est impossible de faire cette opération au moment de l'instanciation, car on ignore si les modules auxquels on doit se connecter ont déjà été instanciés, ou non. Au mieux, on pourrait choisir l'ordre d'instanciation de façon à optimiser ce problème, mais cela ne nous mettrait pas à l'abri de cycles dans le graphe des modules. Le plus simple est alors de séparer ces deux phases, ce que j'ai fait.

- Création d'un panneau permettant de configurer le module .
  Chaque module possède un jeu de paramètres qui lui sont propres. Régler ces paramètres de façon universelle est donc délicate. De plus, certains de ces paramètres peuvent être reliés les uns aux autres, ce qui complique encore la tâche. Par exemple, dans un modèle Markovien, le nombre des états conditionne la taille des matrices de transition. Chaque ligne de ces dernières est soumise aux contraintes des distributions de probabilités. Pour résoudre ce problème, je propose donc, un peu à la façon des concepteurs mis en œuvre par Borland, que chaque module publie un panneau de conception.

Ce dernier n'a que deux contraintes : pouvoir s'afficher et pouvoir générer une instance du module qui l'a créé. Cela permet donc aux modules simples, comme une moyenne mobile, de proposer des entrées distinctes pour chaque paramètre, comme la taille de la fenêtre glissante. Par contre, les modules plus évolués peuvent mettre en œuvre un affichage plus complexe pour présenter et modifier leurs données. Ainsi, un modèle markovien se construit en deux étapes : choix de la structure, et choix des probabilités. Dans la première, l'utilisateur peut modifier le nombre et la description des divers états et des actions. La deuxième page permet alors de choisir les probabilités de transitions, ainsi que l'état initial, à travers le remplissage de matrices visuelles, en imposant les contraintes.

Enfin, chaque module doit être lié à un avatar graphique, de façon à permettre sa représentation au sein de l'interface de conception. Il n'a donc aucune importance quant au fonctionnement même du module. D'ailleurs, lors de la compilation, cette partie n'est pas instanciée du tout. Comme cette partie est commune à de nombreux modules, nous avons trouvé judicieux d'en faire une classe à part entière. Cette classe, *BoxObject*, contient donc toutes les routines d'affichage de boîtes, de broches de connexion, et de liens. Elle contient en outre le nom du module, ainsi que le nom de la variable d'instance qui lui correspondra après la compilation. Cela est en effet nécessaire pour compiler le code Java correspondant à la liaison des modules. Finalement, elle contient également les routines liées à l'interaction avec l'utilisateur. Ainsi, lorsque l'on clique sur une boîte, cette dernière recherche la broche de connexion la plus proche du curseur et choisit l'action associée à cette dernière. Pour les broches d'entrée, cette action est vide, alors que pour les sorties ou les broches d'utilisation, cette action permet d'initier une nouvelle connexion.

7.3 Application à nos problèmes

Tout au long de ce chapitre, j'ai présenté les principes régissant la bibliothèque de composants, leur mise en œuvre grâce à des flux de données, et leur intégration au sein d'une interface de développement. Je vais maintenant montrer l'application réelle de ces principes sur les applications que j'ai développées tout au long de ce mémoire.

Sur les trois diagrammes figurant dans les pages suivantes, les modules représentés en gris sont les seuls à être visibles de l'application hôte. Celle-ci fournit donc les données à la source (à gauche des diagrammes) et reçoit les résultats à la sortie (à droite des schémas). Le détail des calculs est complètement masqué.

Chaque module dispose d'entrées situées sur sa gauche, et de sorties situées sur sa droite. Les flèches reliant les modules les uns aux autres représentent des flux de données. Au niveau de la terminologie objet, ces flèches se traduisent donc par un lien de clientèle, le module source appelant une fonction du module cible. Les liens issus du bas des modules correspondent à des relations de clientèle plus générale, et ne figurent pas de flux de données directs. Ainsi, un module d'apprentissage *utilise* un module dont il doit optimiser la fonction. Ce cas précis recouvre donc un flux de données asynchrone bidirectionnel.

La Figure 39 présente le diagramme de l'application d'aide au diagnostic des troubles de l'hydratation, dans le projet DIATELIC. Dans ce cas précis, cette représentation est biaisée, car le programme correspondant existait avant que la bibliothèque ne soit développée. En fait, le diagramme présente ce qui serait réalisé aujourd'hui. Le programme qui a été expérimenté durant deux ans est beaucoup plus monolithique ; tout tenait en trois objets. Le point d'entrée situé en haut à droite permet au médecin de corriger le diagnostic et de faire calculer le modèle correspondant. Seuls la consigne et l'ordre d'apprentissage sont utiles à cet endroit.

La Figure 40 montre le diagramme qui sera utilisé dans le projet d'assistance durant une anesthésie. Le module qui fonctionne actuellement est plus restreint, bien que basé sur la même structure générale. En fait, sur ce diagramme, tous les signaux disponibles ont été ajoutés, de façon à permettre l'étude de leur utilité. Dans ce cas précis, l'étude porte sur l'intérêt des deux canaux du BIS et de la qualité du signal en tant que donnée. En définitive, certains seront sans doute supprimés ; d'autres pourront être ajoutés. On peut remarquer qu'aucun module d'apprentissage n'apparaît. Dans la version finale, un module d'adaptation sera ajouté, mais il se contentera de modifier le modèle par petites touches. L'étude que je propose ici consiste à rechercher un modèle sans a priori. L'espace de recherche est donc trop important pour être parcouru durant l'opération. De plus, les risques de faux diagnostics seraient importants, ce qui pourrait être dangereux pour le patient.

Enfin, sur la Figure 41, j'expose le cœur du module de navigation de notre robot mobile. Par rapport aux diagrammes précédents, on peut noter l'ajout d'un module de décision basé sur Q-MDP. Ce module, bien que rudimentaire, a permis de faire évoluer le robot de façon autonome assez facilement. Dans l'avenir, des modules plus compétents seront développés, mais ils sortent du cadre de cette thèse. Le modèle de l'environnement est une entrée de données pour l'application, car c'est à cet endroit que le but est fixé au robot.

On peut facilement remarquer la similarité entre ce diagramme et les deux précédents. Cela montre bien que la même architecture, contenant les mêmes algorithmes, peut être appliquée à ces deux classes de problèmes, la médecine et la robotique.

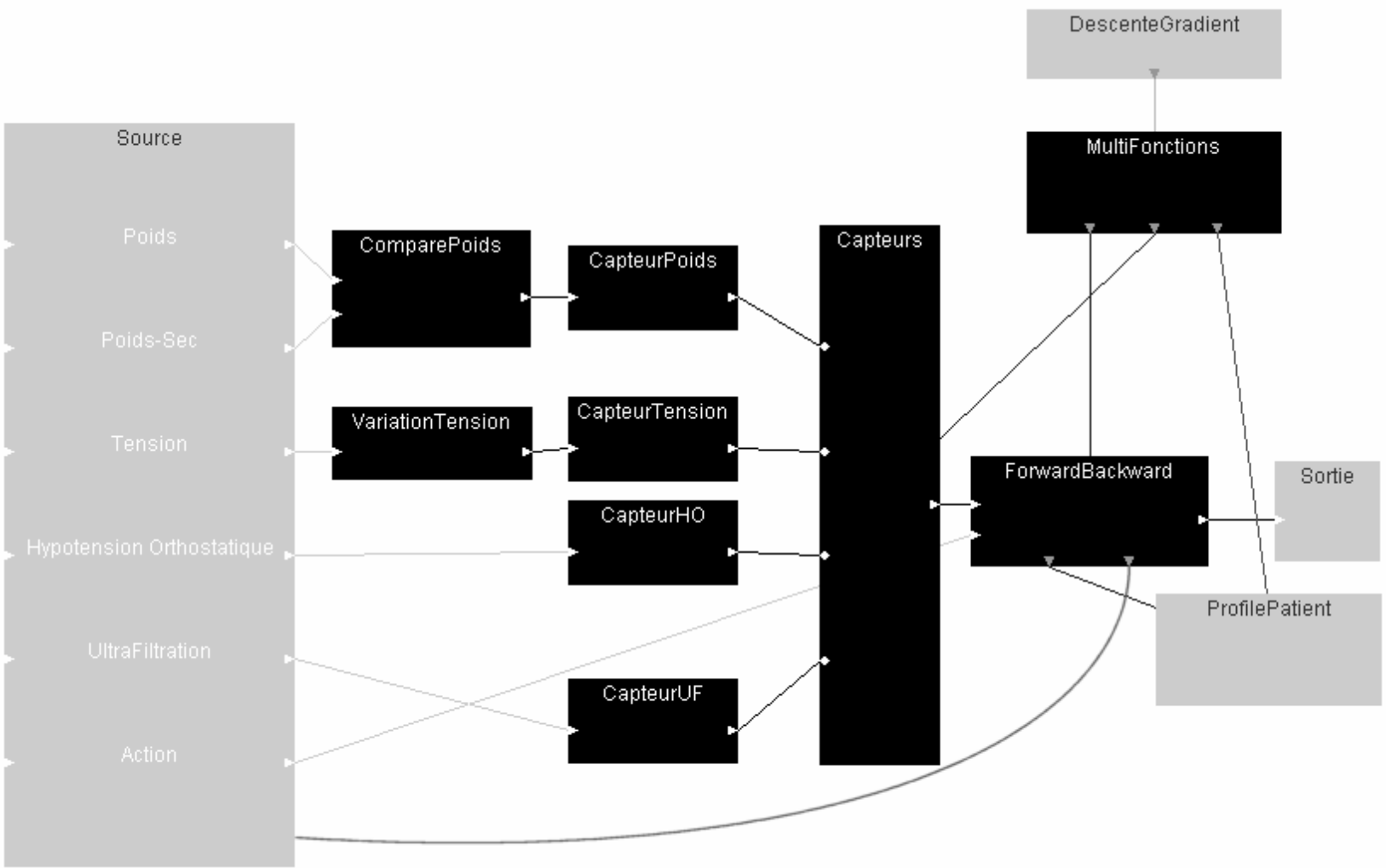


Figure 39 : Le diagramme du projet DIATELIC

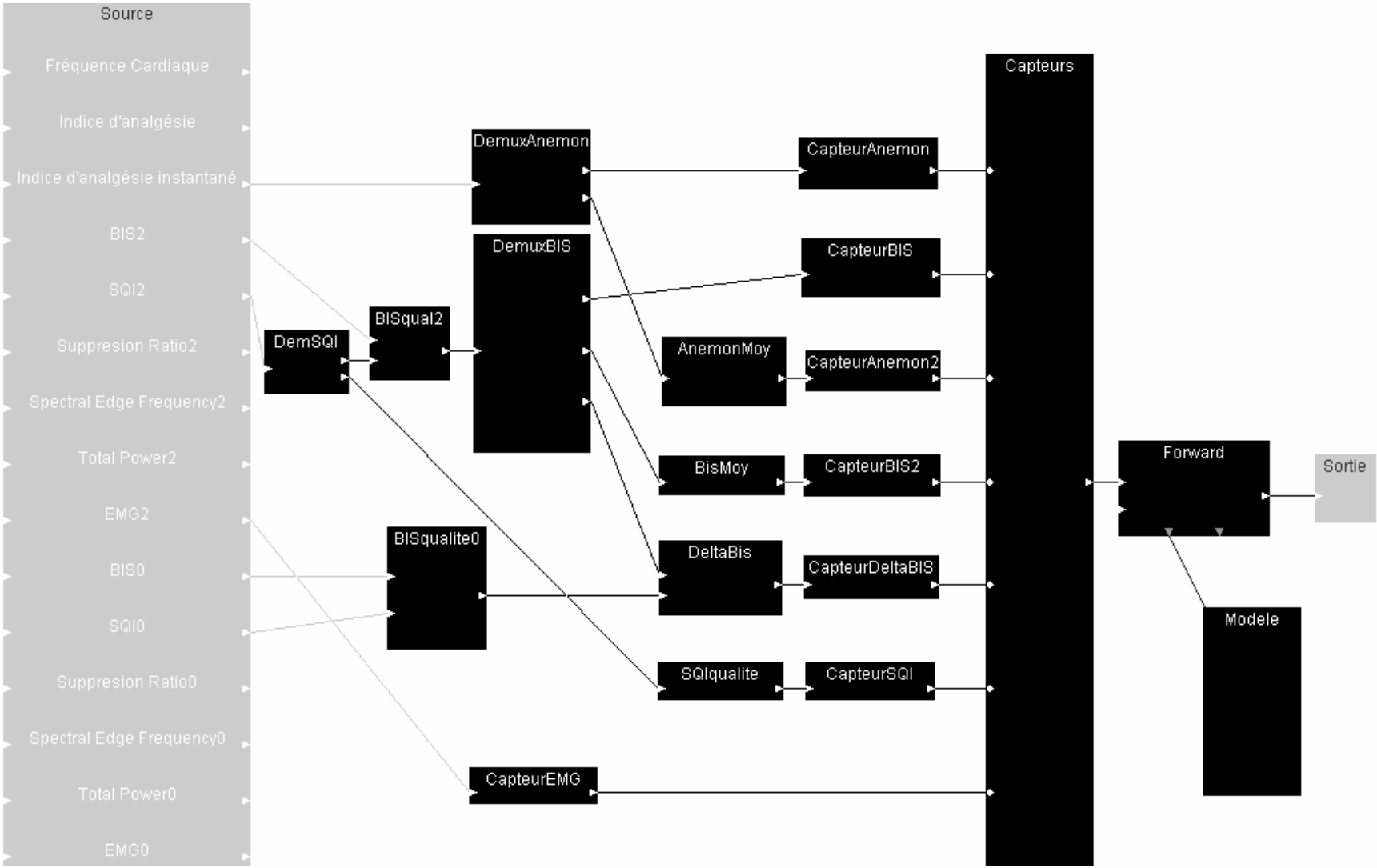


Figure 40 : Le diagramme du projet d' aide à l' anesthésie

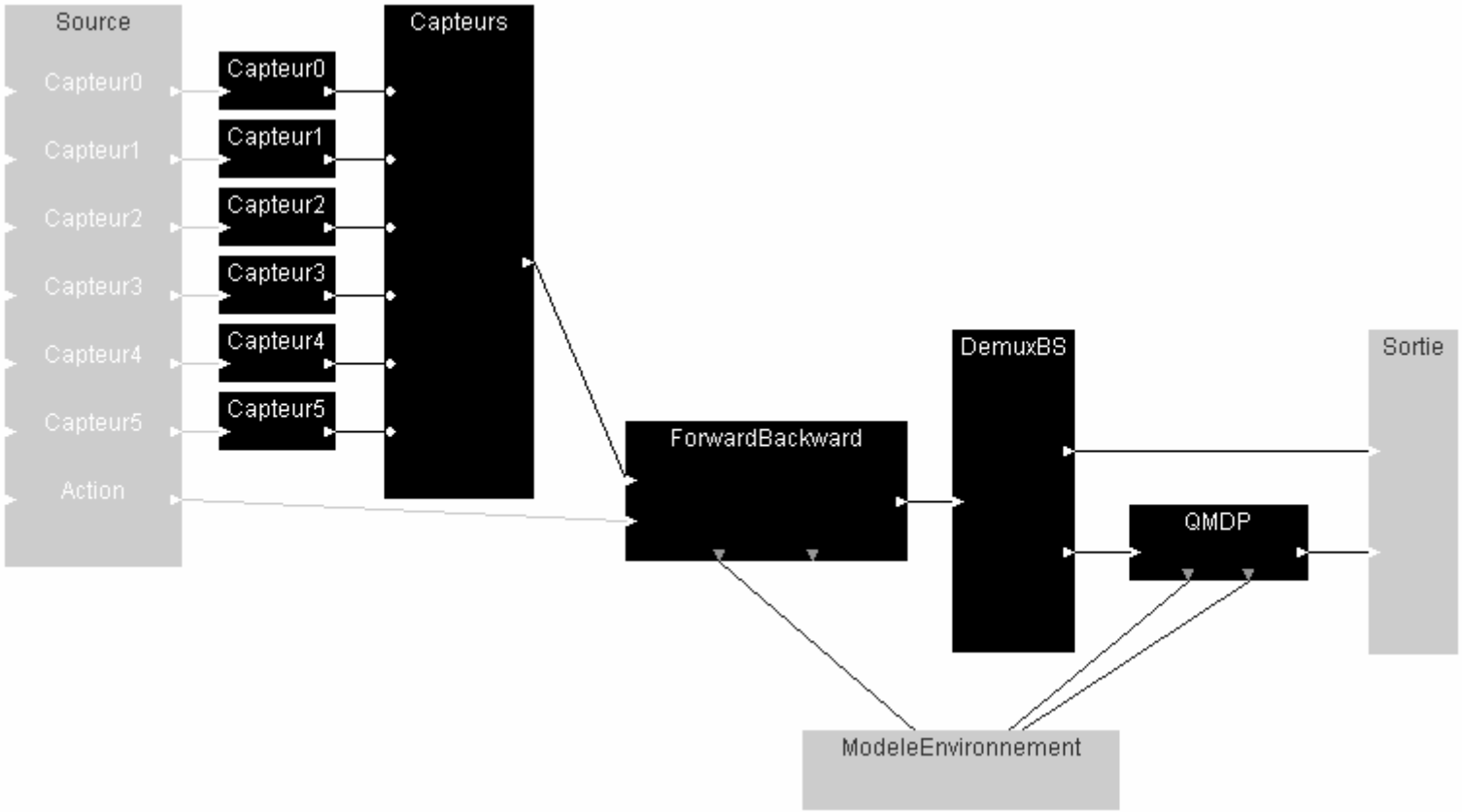


Figure 41 : Le diagramme du module de navigation de Simplet

8 Conclusion et Perspectives

Tout au long de ce mémoire, j'ai montré comment le choix d'une ou plusieurs applications réelles pouvait être une source de motivation et d'inspiration sans fin pour la recherche fondamentale. En effet, j'ai montré les difficultés que posaient ces problèmes, et les parades que l'on pouvait y opposer. Les résultats obtenus sont encourageants, au point de concurrencer directement les méthodes originales développées par ailleurs.

Parmi les problèmes que j'ai traités, on trouve notamment toutes les contraintes que pose la collaboration entre un système informatique intelligent et un expert humain. J'ai donc proposé un modèle de diagnostic basé sur des filtres flous. Ces filtres permettent de transcrire les données informelles, souvent imprécises, qu'utilisent quotidiennement les experts humains en un modèle utilisable par une machine. Cette même expression permet alors une interaction simple des experts avec le système, mais cette façon de représenter le modèle permet surtout à ces derniers de comprendre intuitivement son fonctionnement. En définitive, ces derniers sont capables d'utiliser le système très rapidement, d'en fixer les paramètres aisément, et surtout d'en comprendre les sorties sans requérir de fonction de traduction évoluée. La sortie brute du système est suffisamment claire pour permettre son interprétation directe.

Les modèles stochastiques que nous avons utilisés, les modèles de décision markoviens partiellement observables, nécessitent souvent le réglage de nombreux paramètres. L'adaptation du système à une situation précise, comme par exemple un patient qui va être suivi en dialyse péritonéale continue ambulatoire (DPCA), peut être longue et fastidieuse. J'ai donc proposé une nouvelle méthode d'apprentissage qui permet à un expert de fixer le diagnostic, et de laisser le modèle choisir les paramètres qui permettent d'obtenir ce dernier. J'ai donc montré comment cette méthode, basée sur la descente de gradient contrainte sans dérivées, permettait d'apprendre un modèle consistant au niveau numérique, mais surtout un modèle qui respecte la sémantique des états. En donnant la parole à un expert humain, cet apprentissage garantit la fiabilité du diagnostic, mais l'approche numérique garantit simultanément la stabilité de ce dernier et sa robustesse vis-à-vis de données approximatives.

J'ai montré au cours des précédents chapitres que ce modèle et son architecture pouvaient être appliqués à des cas réels, divers et variés, et présentant des contraintes plus ou moins fortes. La réalisation d'une bibliothèque logicielle cohérente et simple d'emploi, associée à une interface de développement visuel, permet désormais l'application aisée de ce type d'agents à des domaines variés. Son architecture ouverte permet d'intégrer facilement de nouveaux modules, et de mettre à jour les fonctionnalités de ceux qui sont déjà installés, montrant ainsi d'immenses possibilités d'évolutions.

Pour rendre cette bibliothèque véritablement opérationnelle, il reste bien entendu à lui adjoindre les modules de calcul que nous utilisons quotidiennement. Cela permettra alors de l'utiliser en tant qu'outils de prototypage rapide, évaluant la pertinence d'un modèle pour une application sans développer d'algorithme ad hoc. Parmi ces modules, j'envisage la création rapide de modules d'inférence Bayesienne, d'un système à base de règle, de planificateurs symboliques, et de filtres arithmétiques permettant des calculs simples entre les valeurs de plusieurs flux, de calculer la dérivée d'une séquence, d'analyse de Fourrier rapide, et ainsi de suite. Dans le cadre des modèles markoviens, il faudra implanter des modules capables de calculer et suivre une politique, que le modèle soit observable ou non.

On peut aussi envisager l'ajout de modèles semi-markoviens, ou d'autres variantes. Il faudra aussi prendre en compte les algorithmes d'apprentissage par renforcement tels que le *Q-learning*, le gradient de Baxter ou les algorithmes de construction d'observables tels que ceux de McCallum.

Les expérimentations que j'ai conduites sur la robotique mobile m'ont permis de réaliser combien la modélisation stochastique discrète d'un système continu était sujette à l'incertitude. Ainsi, dans le cas où le modèle de l'environnement est différent du modèle réel, j'ai pu observer des simulations où un robot qui avance en ligne droite, à vitesse constance, reste cloîtré dans le même état. En effet, les états où ce mouvement est susceptible de le conduire sont rendus hautement improbables par les observations du robot. De cette façon, l'état le plus probable reste toujours le même. Or ceci est complètement impossible si l'on considère la dynamique du robot.

Cela nous conduit donc à la conclusion suivante : il faut que les états soient suffisamment petits pour que le robot soit certain d'en changer à chaque mouvement élémentaire, ou suffisamment gros pour qu'il n'en change jamais. Or ceci nous conduit à générer une multitude d'états, ce qui rend l'application des meilleurs algorithmes disponibles aujourd'hui totalement impossible. En effet, si l'on considère juste un mouvement de 10 centimètres, modéliser un couloir long de 10 mètres, et large de 2 mètres nécessiterait à lui seul 16000 états si l'on considère une orientation discrétisée par pas de 45 degrés. Cela rend alors le calcul d'une politique, par un modèle de décision Markovien, assez long (plusieurs dizaines de secondes), et celui d'un modèle Markovien partiellement observable impossible.

Par similarité, on est alors en droit de se demander à quel niveau de fiabilité se trouvent nos modules de diagnostic médicaux. Sont-ils soumis à ce type de contraintes également ? Je pense que oui. En effet, j'ai observé à de nombreuses reprises que l'état de ces systèmes pouvait changer très rapidement, même en imposant une contrainte d'inertie forte. En effet, chaque état est considéré comme un tout, et l'appartenance y est booléenne. Soit le système s'y trouve, soit il se trouve ailleurs. Si le système est considéré comme dans un état donné, et que cet état correspond à une position délimitée dans un espace précis, alors le système étudié peut occuper indifféremment toutes les positions de cette zone. Aucun souvenir n'est conservé de l'endroit d'où l'on vient. Dans le cadre de la robotique, lorsque le robot entre dans un état d'un mètre carré, la probabilité de sa présence en tout point de cette zone est identique, alors qu'intuitivement, on imagine bien qu'il se trouve proche de la frontière qu'il vient de franchir. Pour pallier à cet inconvénient, j'envisage l'usage de filtres à particules (45),(68), couplés à un modèle de décision classique. En effet, ce filtre permet de conserver la partie symbolique du modèle pour chaque particule, prise indépendamment des autres. Ainsi, lorsque le robot avance de dix centimètres, chacune des particules va avancer de dix centimètres. L'observation des capteurs va alors renforcer ou discréditer certaines de ces particules, altérant ainsi la position spatiale du modèle du robot. Ces positions peuvent alors être projetées sur une discrétisation adaptée de l'environnement pour choisir l'action adaptée. Alternativement, tout autre modèle de décision peut être employé à cet effet. Il reste donc à choisir le modèle qui sera le plus adapté.

L'application de cette méthode à la dialyse ou à la surveillance d'une anesthésie pourra permettre l'application directe des lois d'évolution des patients, ainsi que d'empêcher le système de diagnostiquer une guérison instantanée d'une pathologie installée depuis plusieurs jours. Ainsi, un patient déshydraté depuis une semaine serait certainement peu susceptible de passer à un état de santé idéal en une seule nuit. Le filtrage particulaire permettrait alors de renforcer la localité du modèle, ou tout du moins son inertie. Le problème sera de trouver des lois d'évolution continues fiables.

A l'inverse, son application à l'anesthésie permettrait de prendre en compte la dynamique des produits anesthésiants de façon directe. Ainsi, l'augmentation des doses d'analgésiques entraîne une augmentation retardée de la concentration au site d'effet, et donc une diminution progressive de la réponse aux stimuli douloureux par le patient. Tenir compte de cette évolution permettra de mieux conduire le diagnostic, sans pour autant avoir à fixer des matrices de transitions simulant ces effets.

L'incertitude représentant la mise en question de nos connaissances sur les effets d'une action donnée sur un système précis peut être modélisée de façon simple dans le processus de mouvement des particules. De même, l'incertitude portant sur l'observation de l'état caché du système reste intacte. On garde donc toute cette incertitude dans le système, et donc la robustesse qui va de pair avec. La seule amélioration de cette méthode de filtrage repose donc sur la prise en compte de l'évolution physique de l'état caché du système. Cela permet alors aussi bien d'interdire la téléportation d'un robot que de modéliser l'inertie présentée par un patient. Je suis donc persuadé que cette méthode peut apporter beaucoup à chacun de ces domaines.

Finalement, la fin de l'expérimentation de DIATELIC nous permet d'en tirer moult enseignements, et l'ensemble des données rassemblées pendant ce temps va nous permettre d'affiner le modèle en lui ajoutant la notion d'action, sans perturber cette expérimentation. En effet, comme je l'avais déclaré précédemment, la seule chose qui nous empêchait de le faire était le manque de données permettant d'étudier, implanter puis valider ces actions. De façon similaire, la récupération des données de nombreuses anesthésies, opération en cours depuis quelques temps déjà, nous permettra d'envisager une approche équivalente pour cette application.

D'autre part, l'expérimentation de DIATELIC met en relief la qualité des résultats obtenus, comme le montre l'annexe 2. En effet, malgré le faible nombre de patients inclus dans l'étude, nous avons pu obtenir d'excellents résultats. En particulier, il est évident que la tension artérielle des patients est mieux contrôlée que dans le groupe témoin. Or, nous savons que cette valeur est directement liée à l'hydratation. Ce fait, combiné à la baisse du nombre de médicaments anti-hypertenseurs utilisés, montre donc que le système atteint son but. Il aide donc les médecins à réguler efficacement l'hydratation des patients, bien que ceux-ci restent à leur domicile.

Pour valoriser le travail mené dans ce cadre, trois brevets ont été déposés, en France, en Europe, et aux Etats-Unis. De plus, une petite entreprise a été créée afin de permettre l'application du système à un plus grand nombre de patients. Ainsi, une étude à plus grande échelle est en préparation. Cette fois, 150 patients devraient être suivis. A ce niveau, le système de diagnostic intelligent que nous avons conçu prendra toute son importance. En effet, il est totalement impossible à un néphrologue de considérer correctement les données de 150 patients tous les jours. La charge de travail que cela représente est tout simplement titanesque.

9 Bibliographie

(1) M. L. van Aartrijk, C. P. Tagliola, P. W. Adriaans, *AI on the Ocean: the RoboSail Project*. European Conference on Artificial Intelligence 2002.

(2) R. J. Anderson, *Problems with the NHS Cryptography Strategy*, 1997, Site internet : http://www.cl.cam.ac.uk/users/rja14/zergo/zergo.html

(3) O. Aycard, P. Laroche, F. Charpillet. *Mobile robot localization in dynamic environment using place recognition*. Proceedings of ICRA 1998.

(4) J. Baxter, P. Bartlett, *Direct gradient-based reinforcement learning: Gradient estimation algorithms*, Rapport Interne : The Austrialian National University, 1999.

(5) R. Bellazzi, P. Magni, R. Bellazzi, *Improving dialysis services through Information Technology: from telemedicine to data mining*.

(6) A. Bilgin, G. Zweig, and M. W. Marcellin, *Lossless medical image compression using three-dimensional integer wavelet transforms*, IEEE Transactions on Medical Imaging, 1998.

(7) G. Booch, *Analyse & conception orientées objets*, 2nde édition, Addison-Wesley 1997

(8) S. J. Bradtke & M. O. Duff, *Reinforcement learning methods for continuous-time markov decision problems*. Advances in Neural Information Processing Systems, Vol 7, 1995.

(9) R.P. Brent, *Algorithms for minimization without Derivatives*, Prentice-Hall, 1973.

(10) B.G. Buchanan & E.H. Shortliffe, *Rule-Based Expert Systems, The MYCIN Experiments of the Stanford Heuristic Programming Project*, Reading, MA: Addison-Wesley, 1984 Site Internet : http://www.aaai.org/Resources/Classics/Mycin/mycin.html

(11) B. Cannas, G. Celli, M. Marchesi et al., *Neural networks for power system condition monitoring and protection*. Neurocomputing 23, pp 111-123, 1998.

(12) A. Cassandra, L.P. Kaelbling and M. Littman. *Acting optimally in partially observable stochastic domains*. In Proceedings of the National Conference on Artificial Intelligence (AAAI 1994), pp 1023-1028.

(13) Z. Chen, X. Yu, D. Feng, *A Telemedicine System over Internet*, Selected papers from Pan-Sydney Workshop on Visual Information Processing, éditions ACS, 2001.

(14) H. T. Cheng, *Algorithms for partially observable Markov decision processes*, PhD thesis, University of British Columbia, 1988.

(15) http://www.ghg.net/clips/CLIPS.html
Site internet de Clips

(16) D. Delamarre and B. Virot, *Simulated annealing algorithm: amélioration techniques*, novembre 1993.

(17) A.P. Dempster, N.M. Laird, D.B. Rubin. *Maximum likelihood from incomplete data via the EM algorithm*. Journal of the Royal Statistical Society, Serie B, Vol. 34, 1977.

(18) P-Y. Durand et M. Kessler, *La dialyse péritonéale automatisée*, éditions Massons, septembre 1998.

(19) A. Dutech, *Apprentissage d'environnement : approches cognitives et comportementales* Thèse soutenue en 1999 à l'école nationale supérieure de l'aéronautique et de l'espace.

(20) http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyClips/fuzzyCLIPSIndex.html
Site internet de Fuzzy Clips

(21) F. Gechter, F. Charpillet, *Vision Based Localisation for a Mobile Robot,* IEEE International Conference on Tools with Artificial Intelligence, novembre 2000

(22) G. Guimarães et al., *A method for automated temporal knowledge acquisition apllied to sleep-related breathing disorders*, Artificial Intelligence in Medecine, novembre 2001.

(23) R. Hervy, L. Romary, F. Charpillet, J-M Pierrel, J-P Thomesse, E. Petitjean, L. Jeanpierre, P-Y Durand, et J. Chanliau, Brevet DIATELIC en France.
Brevet n° FR2804265 (27/07/2001)

(24) R. Hervy, L. Romary, F. Charpillet, J-M Pierrel, J-P Thomesse, E. Petitjean, L. Jeanpierre, P-Y Durand, et J. Chanliau, Brevet DIATELIC aux Etats Unis.
Brevet n° 09/539 988 (déposé le 30/03/2000)

(25) R. Hervy, L. Romary, F. Charpillet, J-M Pierrel, J-P Thomesse, E. Petitjean, L. Jeanpierre, P-Y Durand, et J. Chanliau, Brevet DIATELIC en Europe.
Brevet n° WO0154571 (02/08/2001)

(26) W. Horn, *AI in medecine on its way from knowledge-intensive to data-intensive systems*, Artificial Intelligence in Medecine, Vol 23, août 2001.

(27) R.A. Howard. *Dynamic Programming and Markov Processes*. The MIT Press, Cambridge, MA, 1960.

(28) C. Huang, *Intelligent Alarms: Allocating Attention Among Concurrent Processes*, PhD thesis, 1999.

(29) L. Jeanpierre, F. Charpillet, *Hidden Markov Models for Medical Diagnosis*. HEALTHCOM 2002 (International Workshop on Enterprise Networking and Computing in Health Care Industry).

(30) L. Jeanpierre, F. Charpillet, *Apprentissage de modèles en télémédecine, application à la dialyse*. CAp 2002, la conférence francophone sur l'apprentissage .

(31) L.P. Kaelbling. *Associative reinforcement learning: functions in k-DNF*. Machine Learning, 1994, Vol. 15, pp 279-298

(32) L.P. Kaelbling, A.R. Cassandra, J.A. Kurien. *Acting under uncertainty: discrete bayesian models for mobile-robot navigation*. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 1996

(33) D. Kilman, D. Forslund, *An international collaboratory based on virtual patient records*, Communication. of the ACM, Vol. 40, pp 111-117, août 1997.

(34) S. Koenig and R.G. Simmons. *Unsupervised learning of probabilistic models for robot navigation*. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 1996), pp 2301-2308.

(35) R. Kohavi, *Wrappers for Performance Enhancement and Oblivious Decision Graphs*, PhD thesis, Stanford University 1995

(36) H. Kosch, R. Slota, *PARMED - Information System for Long-Distance Collaboration in Medicine*, Proceedings of Software for Communication Technologies '99.

(37) O. Kourilsky, *Néphro Uro Dialyse*, Editions Lamarre, avril 1997.

(38) N. Lavrac, E. Keravnou, B. Zupan, *Intelligent Data Analysis in Medicine*, Encyclopedia of Computer Science and Technology, Vol.42 pp 113-157, Dekker, New York, 2000.

(39) J.L. Laurière, *Intelligence Artificielle*, Tome 2, chapitre 1, éditions Eyrolles, 1988

(40) J . Lieber, M. d'Aquin, P. Bey, et al., *The Kasimir Project : Knowledge Management in Cancerology,* HealthCom 2002.

(41) C.-E. Liedtke, T. Schnier, A. Blomer, *Automated Learning of Rules Using Genetic Operators*, Proceedings of the 5th International Conference on Computer Analysis of Images and Patterns, pp. 327-334, Budapest, septembre 1993

(42) M. Littman, *The witness algorithm: solving partially observable Markov decision processes*, TR CS-94-40, Department of Computer Science, Brown University, Providence, Rhode Island 02912, USA, 1994.

(43) M. Littman, A. Cassandra, et L. Kaelbling. *Learning policies for partially observable environments: scaling up*. In Prieditis, A. and Russell, S., editors, Machine Learning: Proceedings of the Twelfth International Conference, pp 362-370. Morgan Kaufmann Publishers, San Francisco, CA, 1995.

(44) A. McCallum, *Reinforcement Learning with Selective Perception and Hidden State*, PhD thesis, Rochester University 1995.

(45) R. van der Merwe, J.F.G. de Freitas, A.Doucet, et al., *The Unscented Particle Filter*. Technical report, Dept. of Engineering, Cambridge, 2000.

(46) B. Meyer, *Conception et programmation orientées objet*, Editions Eyrolles, 2000.

(47) S. Muggleton, *Inductive Logic Programming*, Academic Press, San Diego, 1992.

(48) R. Munos, *L'apprentissage par renforcement, étude du cas continu*, Thèse soutenue en 1997.

(49) R. Munos & A. Moore. *Variable resolution discretization in optimal control*. Robotics Institute Technical Report, CMU, 1999

(50) R. Munos, & A. Moore, *Rates of convergence for variable resolution schemes in optimal control*, International Conference on Machine Learning 2000.

(51) P. M. Murphy & M. J. Pazzani, *Revision of Production System Rule-Bases, Machine Learning*: Proceedings of the Eleventh International Conference, pp 199-207, New Brunswick, USA, 1994.

(52) D. Page, *ILP: Just Do It*, Lecture Notes in Computer Science Vol. 1861, 2000

(53) A.P. Pentland, M. Petrazzuoli, A. Gerega. *The digital doctor: an experiment in wearable telemedicine*, Proceedings of The First International Symposium on Wearable Computers, pp 173-174, 1997

(54) William H. Press and Saul A. Teukolsky and William T. Vetterling and Brian P. Flannery, *Numerical Recipes in C, 2nd. Edition*, Cambridge University Press, 1992. Site Internet : http://www.nr.com/

(55) R.P. Popovich, J.W. Moncrief, J.F. Decherd, J.B. Bomar, W.K. Pyle - *The definition of a novel portable/wearable equilibrium dialysis technique*. Trans Am Soc Artif Intern Organs 1976

(56) M.J.D. Powell, *Numerical Methods for Nonlinear Algebraic Equations*, 1970.

(57) M. Puterman, *Markov Decision Processes: discrete stochastic dynamic programming*, John Wiley & Sons publishers, 1994.

(58) J.R. Quinlan. *Induction of decision trees*. Machine Learning Vol.1, pp 81-106, 1986.

(59) L.R. Rabiner, *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* Proceedings of the IEEE, 77(2), pp. 257-285, février 1989.

(60) L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*, Prentice Hall Signal Processing Series, 1993.

(61) J. Schmidhuber, S. Heil. *Predictive coding with neural nets: application to text compression*, Advances in Neural Information Processing Systems 7, pp 1047-1054. 1995.

(62) E.H. Shortliffe, *The evolution of health-care records in the era of the Internet*. MEDINFO' 98

(63) http://anesthesia.stanford.edu/pkpd/HTML%20Web%20pages
Site internet du département d'anesthésie de l'université de Stanford.

(64) F. Steimann, *Fuzzy set theory in medicine*, Artificial Intelligence in Medecine, Vol 11, 1997.

(65) F. Steimann, *On the use and usefulness of fuzzy sets in medical*, Artificial Intelligence in Medecine, Vol 21, 2001.

(66) R.S. Sutton, *Learning to predict by the methods of temporal differences*, Machine Learning, 1998

(67) S. Thrun, J.-S. Gutmann, D. Fox, et al. *Integrating topological and metric maps for mobile robot navigation: a statistical approach*. Proceedings of AAAI-98.

(68) S. Thrun, D. Fox, W. Burgard, and F. Dellaert. *Robust monte carlo localization for mobile robots*. Articial Intelligence Vol. 128, pp 99-141, 2001

(69) C.J.C.H. Wattkins, *Learning from Delayed Rewards*, PhD thesis, Cambridge University 1989.

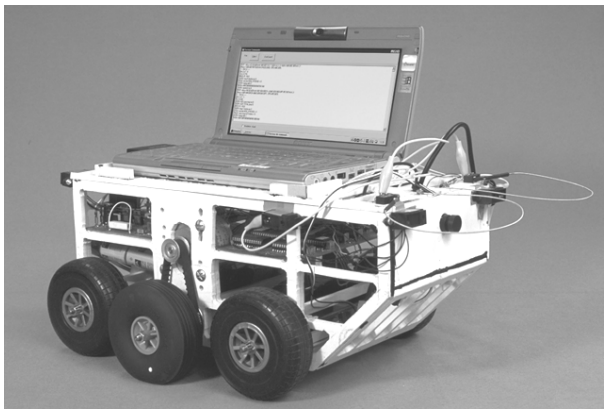
(70) M.A. Wiering, J. Schmidhuber, *Solving POMDPs with Levin search and EIRA*, Machine Learning 1996: Proceedings of the 30th International Conference, pp 534-542. Morgan Kaufmann Publishers.

(71) L. A. Zadeh, *Fuzzy Sets*, Information and Control, 1965, Vol. 8, pp 338-353.

Annexe 1 : Présentation de Simplet

Introduction

Simplet est un robot expérimental que nous avons développé entièrement. En effet, les problèmes de maintenance que nous avons eu avec des robots commercialisés nous ont encouragés à en maîtriser tous les aspects. Ce travail a été majoritairement conduit par Franck Gechter, Alain Dutech et moi-même, mais d'autres membres de l'équipe ont pu apporter leur touche personnelle à diverses étapes du développement. Ci-dessous se trouvent deux photographies de Simplet, prises à un an d'intervalle.



Ce robot mobile a été conçu originellement dans l'optique de mettre en œuvre des systèmes multi-agents hétérogènes. Cet objectif a conditionné une grande partie du développement. En majorité, ces contraintes ont porté sur la généralisation de l'architecture, ainsi que sur la réduction des coûts.

Architecture générale

Le robot a été conçu autour du bus I²C, inventé par la société Philips. Ce bus, constitué de seulement deux fils, permet en effet de faire communiquer des plateformes hétérogènes avec peu de soucis de synchronisation. C'est à la base un système centralisé, dans lequel un *maître* peut interroger à volonté chacun de ses *esclaves*. Il est donc très modulaire, puisque la seule interface nécessaire est un bus de deux fils.

Afin de pouvoir disposer d'une puissance de calcul suffisante, et dans le but de rendre le système le plus générique possible, nous avons décidé de déporter tous les algorithmes en dehors du robot. Pour cela, une transmission radio relie chaque robot à un ou plusieurs postes qui peuvent les commander. La tâche de chaque robot se réduit donc à récupérer les données de ses capteurs, les envoyer au serveur distant, et exécuter les ordres transmis par ce dernier.

Les postes communicant avec les robots peuvent être autonomes, ou être reliés à un réseau. Dans ce dernier cas, ils peuvent accepter des connexions sur le réseau et servir de relais. Cette architecture permet en effet à un client situé sur une machine quelconque du réseau de piloter n'importe quel robot. Le client n'a aucune contrainte tant qu'il respecte le protocole de discussion avec le serveur. Ainsi, un client utilisant le langage C sous Solaris peut travailler avec les capteurs et les effecteurs d'un robot donné sans se soucier des modalités techniques sous-jacentes, même si le serveur est écrit en Java sous Windows et que le robot est un système propriétaire écrit en langage d'assemblage.

Avec cette optique de commande déportée, il est donc très facile de mettre en œuvre un algorithme. Cependant, chacun de nos robots est doté d'une caméra. L'avantage de ce capteur est sa richesse. Son inconvénient tient dans la taille des données fournies. En effet, chaque image correspond à un grand nombre de pixels, et de nombreuses images sont transmises chaque seconde. Pour palier à ces inconvénients, nous avons doté chaque robot d'un système de transmission indépendant, afin de ne pas surcharger le canal partagé sur lequel les robots communiquent. Ce système est commercial, et chaque caméra émet sur un canal de la bande de fréquences allant de 1285 à 1340 MHz. Le récepteur est alors branché sur une carte d'acquisition vidéo standard, et le flux d'images est mis à disposition sur le réseau. Pour limiter le débit sur ce dernier, il est possible d'extraire et d'envoyer des informations pertinentes en lieu et place des images brutes. Des travaux sont en cours à ce sujet.

Architecture spécifique de Simplet

Le robot est construit sur la base d'un microcontrôleur embarqué de référence P51XAS3. Cette puce, de marque Philips, est dérivée du processeur 8051 inventé par Intel, mais revue et améliorée. Voici rapidement ses caractéristiques :

- Architecture 16 bits à hautes performances
- 1024 octets de mémoire vive
- 24 lignes d'adresses (jusqu'à 16Mo)
- Horloge jusqu'à 30MHz
- 8 convertisseurs analogique–digital multiplexés
- 3 compteurs / base de temps
- module de chien de garde
- bus I²C
- 2 ports série
- 50 entrées-sorties configurables séparément

La carte que j'ai conçue autour de ce processeur utilise 16 des lignes d'adressage pour alimenter 32Ko de mémoire vive additionnelle et 16Ko de mémoire morte. Ces 16Ko permettent de stocker largement le code nécessaire au fonctionnement du robot. Cela inclut aussi bien la configuration des divers modules du robot que la communication avec le poste serveur. Cette communication, à travers un port série RS232 standard, permet par ailleurs de télécharger des modules additionnels permettant de gérer diverses fonctions.

Le chien de garde est une sécurité qui assure que le robot reste sous contrôle. En d'autres termes, si le robot perd le contact avec le serveur distant pendant plus de deux secondes, il se réinitialise et s'arrête d'urgence. Ce protocole garantit donc que le client garde le contrôle absolu sur le robot tout au long de l'expérience.

La communication est basée sur un principe inspiré des trames Ethernet. Actuellement, le processeur communique avec un PC portable que le robot transporte. Ce PC héberge un serveur qui relaie les informations sur un réseau sans fil compatible avec celui équipant le laboratoire. Cette solution temporaire permet au robot d'évoluer librement presque dans tout le bâtiment.

Une fois que nous avons retiré toutes les broches nécessaires au fonctionnement de fonctions déjà citées, il reste donc 20 lignes d'entrées-sorties configurables à volonté. Le premier prototype utilise 8 de ces broches pour commander les moteurs, et 9 pour interroger les divers capteurs. Les versions suivantes seront plus modulaires, en délégant au moins les fonctions motrices à une carte spécialisée.

L'autonomie électrique du robot est assurée par trois batteries indépendantes : un bloc standard, issu des modèles réduits, alimente le processeur et les moteurs. Un bloc est réservé pour la caméra et son transmetteur, et le portable dispose de sa batterie personnelle. Cette configuration permet au robot de fonctionner pendant 2 heures, au-delà desquelles le portable doit être rechargé.

La propulsion

La partie motrice de Simplet est constituée de deux moteurs pas à pas situés de chaque côté du robot. La stabilité est assurée par deux paires de roues libres situées à l'avant et à l'arrière du robot. Cette configuration lui permet donc de se déplacer en ligne droite, de tourner sur place, ou de se déplacer en arc de cercle. Néanmoins, cela nécessite une grande force de la part des moteurs. Cela explique pourquoi la structure du robot a été modifiée cette année. En effet, les moteurs sont suffisamment robustes pour faire avancer Simplet en ligne droite, mais les virages entraînent des frottements importants. C'est pourquoi la première version tournait difficilement sur une table, et était incapable de tourner sur la moquette.

La deuxième coque a donc été usinée par Matthieu Menard à partir de matériaux de récupération qui semblaient moins lourds que le métal de la première version. Cette nouvelle coque dispose de la place suffisante pour intégrer un réducteur à courroies qui démultiplie par trois la puissance des moteurs.

En définitive, Simplet dispose donc d'une propulsion d'une grande précision, capable de déplacer le robot à une vitesse raisonnable (20 cm/s). Néanmoins, puisque toutes les commandes sont envoyées par le processeur lui-même, en plus de la gestion des communications et des capteurs, la vitesse n'est pas très stable. En particulier, on observe que le robot zigzague légèrement lorsqu'il avance. En effet, la vitesse des moteurs est régie par la fréquence des impulsions envoyées à ces derniers. Il arrive alors que le processeur soit occupé à une autre tâche au moment d'envoyer cette impulsion. Cette dernière va donc être envoyée avec un peu de retard, ce qui va réduire localement la vitesse du moteur. Il est fort probable que le fait de déporter cette tâche hors du processeur permettra d'améliorer la qualité de ces signaux, et donc du mouvement. En attendant, cet aléa est très largement compensé par les modèles de décision markoviens que nous utilisons ; cela n'empêche donc pas le robot de fonctionner.

Les capteurs

Outre la caméra, Simplet dispose de capteurs de proximité et de capteurs de contact. Ces derniers sont situés à l'avant du robot, et déclenchent un comportement réflexe lorsqu'ils sont activés. Par sécurité, ces capteurs permettent de détecter cet obstacle à 10 cm du robot. En effet, si le robot avance et que les *moustaches* détectent un obstacle, il y a un fort risque de collision imminente. A ce moment, le robot stoppe brutalement et effectue une marche arrière sur environ 5 cm. Vu la faible vitesse du robot, ces mesures devraient suffire à empêcher Simplet de heurter un mur.

Les capteurs de proximités sont des GP2D02 de marque Sharp. Ils permettent de mesurer avec une grande précision des distances comprises entre 10 et 70 centimètres. Au dehors de cette plage, la précision de la mesure dépend fortement de chaque capteur. D'après leur documentation technique, la mesure est basée sur l'angle entre un rayon infrarouge envoyé par le capteur et le rayon réfléchi par l'obstacle. L'expérience montre une bonne stabilité, avec toutefois quelques mesures aberrantes. La mesure est également assez robuste vis-à-vis de la lumière ambiante, à l'exception du rayonnement solaire direct.

En définitive, ces capteurs semblent largement suffisants pour permettre une navigation locale du robot. Cependant, pour se repérer dans un environnement de grande taille, ils sont clairement insuffisants.

Extensions futures

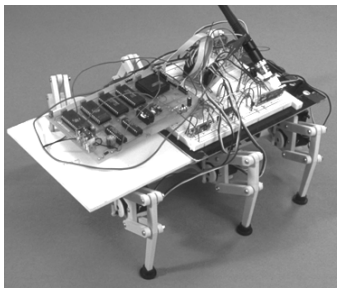
Afin de parfaire notre robot, de nombreuses extensions sont prévues, voire en cours d'étude ou de réalisation.

Au niveau de la communication avec le serveur, nous travaillons actuellement avec un portable et une liaison Ethernet sans fil. Pour remplacer cette connexion, j'étudie une transmission RS232 par ondes radio. Nous projetons d'utiliser la bande de fréquences située autour de 433,92 MHz. En effet, cette plage est disponible sans nécessiter de licence particulière, sous réserve de limiter la puissance d'émission. De nombreux modules existent, et particulièrement chez Radiometrix. Le débit attendu est de 40 kilobits par seconde, ce qui nous permettrait de créer une liaison à 19200 bauds avec un codage Manchester. Cependant, la portée reste très en deçà de nos espérances.

Au niveau de la propulsion, j'ai déjà parlé d'un module de commande de moteurs pas à pas autonome, commandé par I²C. Un autre module a été réalisé par des étudiants dans le cadre d'un stage. Ce module devait nous permettre de commander des moteurs à courant continu, mais le travail n'a pas été terminé dans les temps. Eric Lucchese travaille actuellement à finir cette carte et à la monter sur un autre robot. Finalement, un troisième module est en cours de conception. Ce dernier vise au pilotage des servomoteurs de notre robot hexapode. Cette commande est plus ardue, car la position absolue de chaque moteur est commandée par une largeur d'impulsion bien définie. Il faut donc assurer une stabilité parfaite de cette impulsion, et ce pour les 12 moteurs nécessaires à déplacer Atchoum.

Au niveau des capteurs, Eric Lucchese travaille sur un SIC. Ce module est un système de cartographie utilisant un balayage laser. Il est trop lourd pour pouvoir être embarqué sur nos robots, mais pourra facilement équiper Gaston, un robot de taille plus imposante, ou le Cycab, voiture électrique développée par l'INRIA. J'ai travaillé pour ma part sur un capteur d'accélération, mais les résultats étaient trop bruités pour être utilisables rapidement. J'envisage l'emploi d'un montage intégrateur pour passer d'une mesure instantanée à une mesure sur une période plus longue. Ainsi, on devrait pouvoir éliminer les vibrations, accélérations fortes mais brèves, des accélérations dues au mouvement du robot.

Le Cycab



Atchoum

Annexe 2 : Etude clinique de DIATELIC

Conditions de l'expérimentation

L'expérimentation du système DIATELIC est une étude pilote prospective randomisée monocentrique observationnelle sur 30 patients souffrant d'insuffisance rénale chronique terminale pendant 2 ans. En d'autres termes, il s'agit de la première étude clinique, afin de vérifier si le système permet réellement d'améliorer la condition médicale des patients. Cette étude porte sur un seul centre, l'ALTIR, et va observer l'effet du système sur 30 patients, répartis en deux groupes. Le groupe auquel appartient chaque patient est choisi de façon aléatoire.

Chacun des patients est un patient nouvellement installé (utilisant la DPCA depuis peu de temps), déjà formé à la pratique de la DPCA, et ayant donné son accord pour participer à l'étude, après qu'il ait été informé de sa nature et de ses buts. Un patient donné n'est inclus dans l'étude qu'au 30ème jour passé à son domicile. Le premier patient a été inclus dans le système le 08/06/1999 ; le dernier a été inclus dans l'étude le 07/08/2000. L'étude s'est donc terminée le 07/08/2002.

Objectifs de l'étude

L'étude a pour but de démontrer que, par rapport aux patients traités par DPCA de façon « classique », le système DIATELIC procure :

- une meilleure qualité de vie
- une meilleure survie technique
- une moindre morbidité
- une réduction des coûts.

Caractéristiques de la population étudiée

La randomisation a créé deux groupes statistiquement comparables. En effet, aucune des différences observées n'est significative :

Groupe	Diatelic	Témoin
Sexe (H / F)	8 / 7	9 / 6
Age moyen (écart-type)	69,8 (±14,8)	70,7 (±12,4)
Diabétiques	5	4
Comorbidité (Score Charlson)	5,7	4,8
Distance à l'ALTIR (km)	52	52,5

Le score Charlson, ou comorbidité est un indice qui reflète l'espérance de vie moyenne du patient. Ainsi, plus ce score est élevé, plus l'espérance de vie diminue. Ce score est calculé sur la base de l'age (1 point par tranche de 10 ans au-delà de 40 ans). Ce score augmente pour chaque pathologie potentiellement mortelle :

- infarctus, diabète, démence : 1 point
- hémiplégie, maladie rénale, leucémie : 2 points
- grave maladie du foie : 3 points
- SIDA, cancer avec métastases : 6 points

Au-delà de ces paramètres de base, les causes médicales de l'insuffisance rénale et les fonctions résiduelles sont similaires dans les deux groupes. La randomisation a donc été particulièrement équilibrée, ce qui est une chance car cela met en valeur les résultats de l'étude clinique.

Résultats de l'étude

Survie des patients

Sur les 30 patients suivis, 12 sont encore suivis en DPCA. Les 18 autres patients ont quitté l'expérimentation pour les raisons suivantes :

Cause de sortie de l'étude	Nombre de patients concernés	
	Diatelic	Témoin
Décès	8	4
Transfert Hémodialyse	3	1
Transfert Changement de région	0	1
Greffe du rein	0	1

A la lecture de ce tableau, il semble que le système entraîne la mort des patients au lieu de les soigner. Cependant, il est bon de noter deux points : tout d'abord, aucun de ces décès n'est dû à des troubles liés à la dialyse. En fait, ces morts sont dues principalement à l'apparition de démence, ou à des problèmes de mort subite (arrêt cardiaque, attaque). Rappelons que la moyenne d'âge des patients, en début d'étude, était de 70 ans. Ensuite, une analyse de variance entre les deux groupes montre que les chances que cet écart (4 morts de plus dans le groupe DIATELIC) soit dû au hasard sont de 41%. Enfin, bien que l'écart ne soit pas significatif non plus, je rappelle que la comorbidité moyenne du groupe Diatelic est plus élevée que dans le groupe Témoin (5,7 contre 4,8), ce qui entraîne un risque de décès additionnel non négligeable.

Fréquence des visites

Dans chacun des deux groupes, chaque patient est tenu de venir en consultation 1 fois par mois. Il arrive cependant que des problèmes arrivent entre deux visites programmées. Dans ce cas, une nouvelle consultation s'ajoute au total :

	Diatelic	Témoin
Visites programmées par an	10,6±2,3	11,0±2,3
Visites imprévues par an	2,8±2,1	5,3±2,7
Visites totales par an	13,4±3,4	16,3±2,4

Ce tableau nous montre donc que les visites prévues sont significativement équivalentes dans les deux groupes. Par contre, la fréquence des visites imprévues chute radicalement dans le groupe Diatelic par rapport au groupe témoin. Cette diminution est très significative. ($p<0,66\%$) Par effet de bord, le nombre total de visites diminue également, mais la baisse est moins sensible, bien que toujours significative. ($p<3,4\%$)

Par contre, si l'on sépare les visites en fonction de leurs causes (problèmes d'infection, d'hydratation, technique ou non lié à la dialyse), aucune des baisses n'est significative. On peut donc dire que l'état de santé général des patients est sensiblement meilleur, sans pour autant pouvoir cibler une pathologie en particulier.

Un autre point important dans l'évaluation de la santé des patients tient dans le taux d'hospitalisation. Le groupe Diatelic passe en moyenne 11 jours par an et par patient à l'hôpital, avec un écart-type de 14,5 jours. Les patients du groupe Témoin passent en moyenne 20,5 jours par an à l'hôpital, avec un écart-type de 36,1 jours. Bien que l'écart semble important, il n'est pas suffisant pour être significatif, car les écart-types sont importants au sein d'un même groupe. On ne peut donc tirer aucune conclusion de ces chiffres.

Qualité de vie

L'évaluation de la qualité de vie des patients est faite par les patients eux-mêmes. A plusieurs reprises, ils ont tous rempli le même questionnaire dans lequel ils jugent les difficultés rencontrées dans leur vie de tous les jours. Il s'agit donc d'une analyse très subjective.

Actuellement, l'ensemble des questionnaires n'est pas dépouillé. Aucun résultat n'est donc disponible.

Etat médical

L'évolution de l'état médical des patients est assez difficile à juger. Pour ce faire, les médecins étudient l'évolution de certains paramètres clé pour la dialyse. Ces paramètres sont le Poids, la tension, et le nombre de médicaments anti-hypertenseurs (antiHTA). En effet, le poids et la tension sont directement influencés par le niveau d'hydratation du patient. Les antiHTA, quant à eux, agissent sur la tension, afin de la réguler.

	Diatelic	Témoin
Variation du poids	+0,413kg (±4,3kg)	+2,631kg (±3,9kg)
Variation de la tension	-1,177 (±1,133)	-0,023 (±1,582)
Variation du nombre de médicaments antiHTA	-0,2 (±0,561)	+0,333 (±0,9)

Bien que la prise de poids semble bien mieux contrôlée dans le groupe Diatelic, la différence n'est pas significative, au vu de l'importance des écart-types. Par contre, la baisse de tension, elle, est significative ($p<0.0292$). De plus, la prise de médicaments est sensiblement plus faible dans le groupe Diatelic que dans le groupe Témoin. Néanmoins, une analyse de variance ne nous donne que $p<0,0614$, ce qui est tout juste supérieur au seuil de significativité retenu en médecine ($p<5\%$).

Néanmoins, ces différentes évolutions peuvent être recoupées pour obtenir une appréciation unique : au cours de l'expérimentation, le nombre de médicaments nécessaires à maintenir une tension artérielle correcte diminue dans le groupe traité par Diatelic, alors qu'elle augmente dans le groupe Témoin. Dans le même temps, cette même tension diminue sensiblement dans le groupe Diatelic, alors qu'elle reste plus ou moins constante dans le groupe témoin. Il est donc probable que cette baisse de tension soit due à une baisse du niveau d'hydratation, d'autant plus que cela est confirmé par la baisse du poids moyen du groupe. Cela est corroboré avec les chiffres bruts :

	Diatelic	Témoin
Tension Systolique initiale	13,73±1,6	13,33±2
Tension Diastolique initiale	7,87±1,2	8±1,1
Tension Systolique finale	11,53±1,6	13,9±1,5
Tension Diastolique finale	7,1±1,1	7,9±0,8

De façon visible, les patients frôlaient tous une hypertension légère au tout début de l'expérimentation. En effet, une tension systolique supérieure ou égale à 14cm de mercure suffit pour que le malade soit dit *hypertendu*. En fin d'expérimentation, le groupe Témoin se trouve toujours juste en dessous de cette limite. Par contre, le groupe Diatelic est revenu à des valeurs beaucoup plus normales.

Evaluation des coûts

Le coût d'un traitement tel que la dialyse est difficile à estimer, car il dépend de nombreux facteurs. Il inclut naturellement le coût des produits de dialyse et des médicaments. Il faut aussi y ajouter le prix de chaque visite à l'ALTIR, ainsi que celui de chaque hospitalisation. Enfin, il faut évidemment ajouter le coût des transports.

L'étude a montré une baisse très significative du nombre de visites imprévues, ce qui se traduit par une baisse non négligeable du nombre de visites totales. Les patients de chaque groupe étant situés à une distance équivalente de l'ALTIR, cette baisse se répercute naturellement sur le coût des visites ($p<0.0198$). Bien entendu, au vu des écart-types impressionnants, ni la baisse du coût des hospitalisations, ni la baisse du coût total ne sont significatifs.

Coût annuel en euros	Diatelic	Témoin
Des visites	3011,57±764,48	3679,37±724,22
Des hospitalisations	15435,75±20401,88	28846,05±50627,73
Total	18447,32±20597,59	32525,42±50510,24

Conclusions de l'étude

Bien que les questionnaires sur la qualité de vie des patients n'aient pas encore été dépouillés, cette première étude clinique est clairement concluante. En effet, les paramètres médicaux des patients semblent mieux contrôlés, alors que la consommation de médicaments est moins nécessaire.

De plus, en de nombreuses occasions, le système Diatelic a permis le retour à domicile de patients qui, en d'autres occasions, auraient nécessité une hospitalisation prolongée. Par exemple, l'un des patients suivis grâce à Diatelic a souffert d'une péricardite durant l'expérimentation. Cette pathologie se traduit par une rétention d'eau autour du cœur, et nécessite normalement l'hospitalisation jusqu'à résorption de cette poche. Ce patient a néanmoins pu réintégrer son domicile dès le lendemain. Les médecins ont suivi l'évolution de cette poche à distance jusqu'à sa disparition.

Au niveau informatique, les conclusions de l'étude sont plus mitigées, car rien ne permet de déterminer si les propriétés que nous observons sont dues au système de diagnostic, à un meilleur suivi de la part des médecins, ou même à un effet placebo. En effet, les néphrologues ont observé un suivi plus strict des directives médicales par les patients. Est-ce que les patients, se sentant observés, négligent moins le traitement ? C'est difficile à dire.

Pour pallier à cette incertitude, il faudrait créer un troisième groupe de patients, qui saisissent leurs données tous les jours comme le groupe Diatelic actuel. Cependant, pour ces patients, aucune alarme, ni aucun diagnostic ne serait fourni au médecin. Cela permettrait donc d'éliminer l'effet placebo sur les patients ; néanmoins, rien n'empêcherait les médecins de suivre plus attentivement les patients pour lesquels le système ne leur fournit pas d'alerte. J'ai peur que ce dernier effet soit difficile à éliminer.

Apprentissage et adaptation pour la modélisation stochastique de systèmes dynamiques réels

L'application des algorithmes issus de l'Intelligence Artificielle à des applications concrètes est un domaine de recherche intéressant de par les perspectives que cela ouvre. En effet, les contraintes de ces problèmes sont telles que les faiblesses des algorithmes sont mises en évidence de façon beaucoup plus efficace que sur les exemples académiques classiques. Dans cette thèse, je m'intéresse plus particulièrement à deux problèmes d'aide au diagnostic médical. Les outils développés sont donc en interaction constante avec l'équipe médicale correspondante. Je montre donc comment, en alliant la puissance de raisonnement des modèles Markoviens au côté intuitif des ensembles flous, il est possible d'obtenir un système de diagnostic viable. Pour aider encore cette coopération, j'introduis la notion d'apprentissage de diagnostic. Cette méthodologie permet en effet au médecin de corriger le diagnostic établi par le système sur un laps de temps donné. Le système adapte alors le modèle du patient, de façon à se rapprocher de la consigne, tout en respectant des contraintes de stabilité numérique. Ce processus autorise donc le médecin à modifier les paramètres du modèle de manière cohérente, et, surtout, sans avoir à régler chacun des paramètres manuellement. Je montre finalement comment cette approche peut être généralisée à des problèmes éloignés de la médecine, en prenant l'exemple de la localisation d'un robot mobile. Cette approche mène à la réalisation d'une interface de conception d'agents 'intelligents'. L'utilisateur désireux de construire une nouvelle application peut alors mettre cette bibliothèque en œuvre, en reliant des modules les uns aux autres afin d'obtenir les traitements nécessaires. De par sa conception à base d'objets, cette bibliothèque permet aisément l'ajout ou la modification d'algorithmes au fur et à mesure de leur développement. Cela devrait aider au développement de nouvelles applications, tout en réduisant le travail nécessaire des chercheurs impliqués.

Mots-clés : Intelligence Artificielle Située, Télémédecine, Diagnostic, Apprentissage, Processus stochastiques markoviens.

Learning and Adapting applied to Stochastic Modelling of Real Dynamic Systems

The exploitation of Artificial Intelligence algorithms in real problems is a very interesting method for their improvement. Actually, these applications have such constraints that any weakness of the intended algorithm is exposed, far quicker than using the usual academic problems. More precisely, in my PhD thesis, I studied two medical diagnosis helping projects. Thus, each tool I have created is constantly interacting with a medical team. One of my contributions consists in associating the reasoning capabilities of Markov models with intuitive-looking fuzzy sets. The resulting system has strong diagnosis capabilities, primarily based on this interaction. In particular, I propose a new method for learning directly from a diagnosis. Its application is useful when a doctor needs modifying the system's diagnosis. At this moment, the system starts learning from this correction to enhance the patient profile, so that the new diagnosis is compatible with the doctor's one while ensuring some numerical stability properties. Using this method, any doctor is able to correct a patient's profile without manually modifying every parameter of the model. Finally, I explain we can generalise this approach to apply it to non-medical applications. This is exposed in a classical problem, the localisation of a mobile robot that evolves in a structured environment. This generalisation allowed the realisation of a generic Object Oriented package. With this software, a given user could create a whole application, merely by putting aside some modules and linking them altogether so that the required computations are done in the right order. Thanks to its Object nature, this package allows for easily integrating and updating modules along their evolution. Finally, such an interface should help the creation of new applications by reducing the amount of time that is necessary. Actually, once the user has chosen the modules, their order and their parameters, the application handles the whole code generation to obtain a complete module.

Keywords: Localized Artificial Intelligence, Telemedicine, Diagnosis, Learning, Markov stochastic models.