Ecole Doctorale IAEM
Département Informatique

Thèse de Doctorat

**Lucie Daubigney**

# Gestion de l'incertitude pour l'optimisation de systèmes interactifs

**Membres du jury**

| | | |
|---|---|---|
| Directeur | Alain Dutech | Equipe projet MaIA, Loria – Nancy, F |
| Co-directeur | Olivier Pietquin | Equipe IMS-MaLIS, Supélec – Metz, F |
| Rapporteurs | Joëlle Pineau | Reasoning and Learning Laboratory, McGill University – Montréal, Ca |
| | Frédérick Garcia | INRA – Toulouse, F |
| Examinateurs | Matthieu Geist | Equipe IMS-MaLIS, Supélec – Metz, F |
| | Fabrice Lefèvre | LIA, Universités d'Avigon et des Pays du Vaucluse – Avignon, F |
| | Blaise Thomson | Dialogue Systems Group, University of Cambridge – Cambridge, UK |

**Equipe IMS-MaLIS**
**Supélec – Metz**

**Equipe projet MaIA**
**Loria – Nancy**

# Résumé

Le sujet des travaux concerne l'amélioration du comportement des machines dites « intelligentes ». Cette caractéristique se traduit par une capacité à s'adapter à l'environnement, même lorsque celui-ci est sujet à des changements. Un des domaines concerné est celui des interactions homme-machine. Dans ce cas, la machine doit faire face à différents types d'incertitude pour agir de façon appropriée. Tout d'abord, elle doit pouvoir prendre en compte les variations de comportements entre les utilisateurs et le fait que le comportement d'un même utilisateur peut varier d'une utilisation à l'autre en fonction de l'habitude à interagir avec le système. Ensuite, la machine doit s'adapter à l'utilisateur même si les moyens de communication entre ce dernier et la machine sont bruités. L'objectif est alors de gérer ces incertitudes pour exhiber un comportement cohérent. Ce dernier se définit comme la suite de décisions successives que la machine doit effectuer afin de parvenir à l'objectif fixé. Trouver le comportement optimal revient alors à résoudre un problème de décisions séquentielles sous incertitude. Traditionnellement, des connaissances expertes liées à la tâches sont utilisées pour résoudre ce problème. De ce fait, le développement des interfaces est coûteux, long et difficile à transférer à d'autres tâches.

Depuis quelques années, une méthode semble se démarquer pour optimiser automatiquement la gestion des interactions homme-machine à partir de données : l'apprentissage par renforcement. Cette méthode d'apprentissage automatique permet d'optimiser un problème de prise de décisions séquentielles sous incertitude par essais-erreurs. L'intérêt de cette méthode est que seul l'objectif est spécifié à la machine et non les différentes étapes pour y parvenir. Elle permet alors de s'affranchir de connaissances expertes qui étaient auparavant nécessaires pour définir les différentes étapes de résolution du problème et permet ainsi un gain de généricité. Cependant, le cadre habituel de résolution des problèmes d'apprentissage par renforcement propose de mettre la tâche sous la forme d'un processus décisionnel de Markov. Il est même souvent nécessaire de se placer dans le cadre des processus décisionnels de Markov partiellement observables. Ce cadre de travail nécessite alors à son tour d'apporter des connaissances expertes. En utilisant l'apprentissage automatique, l'expertise a été déplacée du modèle de la tâche spécifique au domaine concerné vers des compétences en apprentissage machine.

La thèse défendue par ces travaux est qu'il est possible de relaxer certaines contraintes liées à l'expertise humaine et obtenir ainsi des interactions de bonne qualité tout en limitant la perte de généricité liée l'introduction de modèles. Deux domaines applicatifs sont choisis pour illustrer le propos car ils répondent à des problématiques différentes. Ils s'agit des systèmes de dialogue parlé et des environnements informatiques pour l'apprentissage humain. Dans le premier cas, la décision de la machine peut se baser sur l'analyse d'un signal physique, la voix, et sur la construction du langage. Les incertitudes à gérer sont directement liées à la bonne reconnaissance de ce qui a été dit. Une estimation de l'incertitude peut être faite par la construction de modèles du langage, bien étudié depuis des dizaines d'années. En revanche, la cognition de l'apprentissage n'étant pas encore un mécanisme bien connu, il reste difficile d'en

construire un modèle. S'en affranchir serait alors un progrès.

Quatre contraintes ont été identifiées comme nécessitant des connaissances expertes. La première concerne la mise en place d'un simulateur d'utilisateurs. En effet, les algorithmes d'apprentissage par renforcement « historiques » étant gourmands en échantillons, il était nécessaire, pour pallier ce manque, d'utiliser un modèle mimant le comportement des utilisateurs pour la machine. La première contribution consiste à s'affranchir de ce modèle par l'utilisation d'un algorithme efficace sur les échantillons et par une gestion du dilemme exploration/exploitation efficace. Le second point propose de gérer le passage à l'échelle bien connu, qu'implique l'utilisation d'un processus décisonnel de Markov partiellement observable, par une représentation non-linéaire de la fonction de valeur. La troisième contribution permet de relaxer le respect de la propriété de Markov. Pour des problèmes partiellement observables réels, l'état de croyance construit selon l'expertise humaine n'est pas garanti de respecter strictement cette propriété. La recherche directe dans l'espace des politiques est alors expérimentée. La quatrième contrainte propose une construction automatique d'un état markovien. Une solution à ce problème passant par l'utilisation d'un réseau de neurones récurrent est proposée.

# Table des matières

# Table des figures

# Liste des Algorithmes

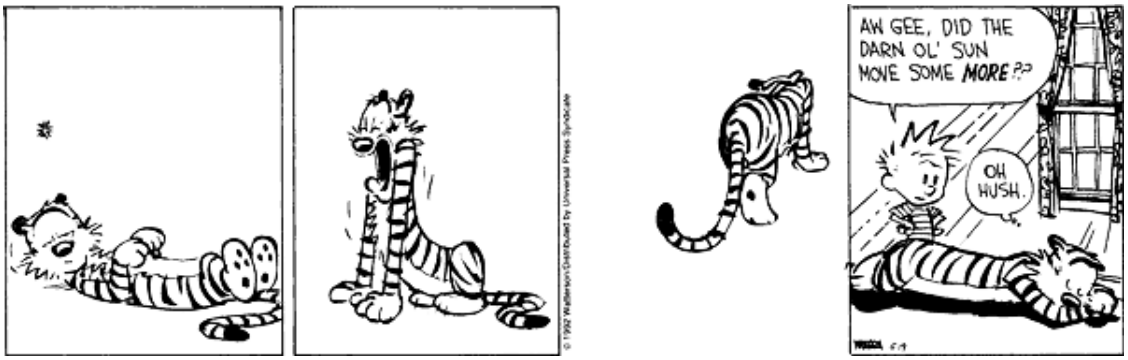
# Première partie

# Contexte et outils

# Chapitre 1

# Des interactions homme-machine performantes

## 1.1 Contexte




Les félins [1] ont cette capacité naturelle à tirer profit de ce qui les entoure. Ce comportement se retrouve chez les humains : toutes sortes d'objets ont été inventées pour s'adapter au mieux aux conditions extérieures. Dès les balbutiements de la construction, un soin particulier a été apporté à l'ergonomie physique des outils pour qu'ils s'adaptent aux capacités de celui qui les emploie. Par exemple, une fois la maîtrise de la taille de la pierre acquise, les bifaces ont été conçus pour épouser les dimensions et formes de la main humaine. L'ergonomie physique de ces premiers outils a été pensée à tel point qu'un outil que nous utilisons actuellement au quotidien, la souris d'ordinateur, en a gardé les mêmes caractéristiques. Une illustration de cette adaptation est proposée Figure 1.1 [2].

Si l'ergonomie physique des objets a pu être améliorée et pensée au cours du temps, il n'en est pas de même pour l'ergonomie cognitive des outils, c'est-à-dire leur adaptation à la cognition humaine. Jusqu'alors, les outils n'ont d'utilité que si une main experte s'en sert. Par exemple, quand un humain conduit une voiture pour se rendre à un lieu donné, de nombreux paramètres interviennent pour accomplir la tâche : il doit savoir utiliser le véhicule, connaître les règles de circulation, la densité du trafic, les conditions météorologiques, vérifier la présence

1. L'image utilisée provient d'un comic strip Calvin et Hobbes de Bill Waterson.
2. Photos provenant de `http://www.ginellames.fr/us/creations/bifaces` et du site Wikipédia français à l'article Souris.

**Figure 1.1** – Comparaison de l'ergonomie de deux objets faits pour être tenus en main, conçus à quelques millions d'années d'écart.

ou non de carburant dans le réservoir, le trajet à emprunter, la présence des bagages dans le coffre, *etc*.

Il a fallu attendre les récents progrès en électronique et informatique qui ont permis d'augmenter la puissance de calcul des ordinateurs et ceux réalisés en traitement du signal pour commencer à comprendre les mécanismes du cerveau, afin d'envisager la conception d'une nouvelle génération d'outils. Pour reprendre l'exemple de la conduite, le passager de la voiture se contenterait alors de monter dans le véhicule et celui-ci le conduirait automatiquement à la destination voulue, tel que le ferait un chauffeur.

Concevoir des outils s'adaptant à la cognition humaine est beaucoup plus difficile que de les adapter seulement aux capacités physiques de l'utilisateur car l'interaction entre ce dernier et la machine est beaucoup plus complexe. En effet, dans ce cas, ce n'est plus l'utilisateur qui doit s'adapter à la machine, mais la machine qui s'adapte à l'utilisateur tel que pourrait le faire un humain mis à disposition pour effectuer la tâche. La machine n'est plus alors prévue pour accomplir une suite de tâches mais comprend l'intention de l'utilisateur et est capable de prévoir par elle-même la suite d'actions à effectuer pour parvenir à satisfaire la requête globale de l'utilisateur. Lorsqu'un humain demande à un autre humain d'effectuer une tâche, de nombreux paramètres entrent en ligne de compte pour la réaliser. Un humain est en effet capable de comprendre la finalité de la tâche et de saisir toutes les implications que cela suppose. Il est aussi capable de prendre en compte les différentes variations liées au contexte. Chaque personne étant différente, il sait s'adapter à l'individu avec lequel il interagit. On ne s'adresse pas de la même façon à un enfant qu'à un adulte. Par ailleurs, la réaction à une même situation donnée peut, d'un jour à l'autre, différer. Pour reprendre l'exemple précédent, si le chauffeur transporte un enfant, il va devoir prévoir le siège auto. L'adaptation du comportement de l'humain aux différents contextes est devenue pour lui un automatisme car dès ses premiers apprentissages, il a appris à les détecter et à les gérer. Pour des interactions homme-machine naturelles, ces adaptations doivent aussi se retrouver dans le comportement de la machine.

De plus, pour pouvoir agir de façon appropriée, la machine a besoin d'informations sur l'environnement dans lequel elle évolue. Or celles qu'elle perçoit sont bruitées ou incomplètes. Si l'interaction se fait par la parole, dans l'exemple précédent, une partie de l'information transmise de la personne transportée au chauffeur peut être couverte par le bruit du moteur. Si le chauffeur est humain, il va s'apercevoir qu'une partie de l'information est manquante. Il peut demander à la personne de répéter. Ou bien, s'il ne veut pas le faire répéter pour ne pas la gêner, il va, avec ses connaissances antérieures comme une destination habituelle et les informations courantes disponibles, essayer d'inférer l'information manquante. L'humain présente une grande capacité à gérer l'incertain. Inversement, l'information peut être difficilement transmise de la machine

vers l'utilisateur. Toujours à cause du bruit ambiant, dans l'exemple précédent, le chauffeur ne peut jamais être sûr que l'information a bien été comprise par le client. A la réponse à une question oui/non, s'il n'a pas de réponse orale, c'est peut-être qu'il a manqué un hochement de tête.

Nous considérons ici que le rôle de la machine est régit par deux sous-modules, ainsi que présenté Figure 1.2. Un premier module modélisant la tâche permet de collecter les informations qui proviennent de l'utilisateur et qui sont inhérentes au domaine concerné. Le second module permet de définir le comportement de la machine et choisit la prochaine décision en fonction des informations fournies par le module précédent. Par exemple, un arbre de décisions permet de définir le comportement de la machine.

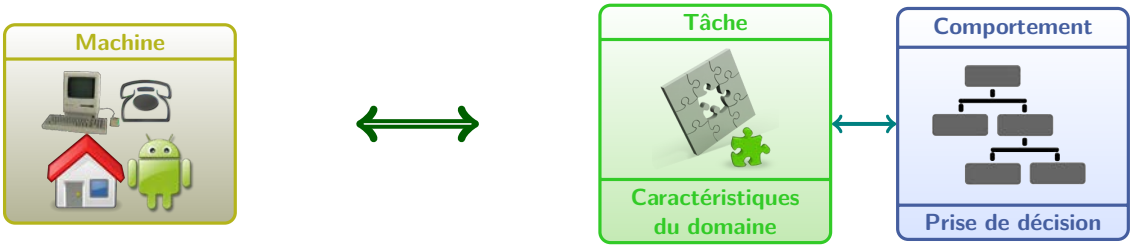



**Figure 1.2** – Modules définissant le rôle de la machine.

En remplaçant ces deux modules dans le contexte général de l'interaction homme-machine, cette dernière se résume sur le schéma présenté Figure 9.1.

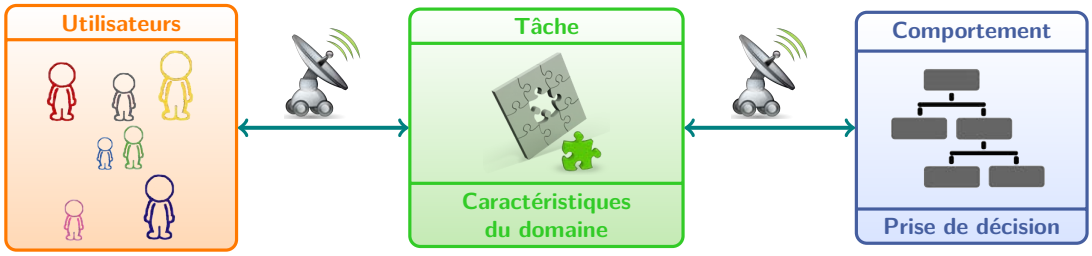



**Figure 1.3** – Résumé de l'interaction homme-machine.

Au travers de l'exemple du chauffeur, il apparaît que la définition d'une interaction performante entre la machine et l'utilisateur passe par une bonne gestion de plusieurs types d'incertains pour que la machine parvienne à effectuer la tâche qui lui est impartie. Différentes sources d'incertain ont été identifiées et sont représentées Figure 1.4. Tout d'abord, une part de l'incertain provient de la transmission de l'information entre l'utilisateur et la machine car celle-ci peut être bruitée. En effet, une interaction naturelle passe par un medium généralement complexe comme la parole, le geste ou encore l'écriture. Ces media sont en général difficiles à interpréter pour la machine et la plupart des méthodes de reconnaissance automatique de formes utilisées pour ce faire sont imparfaites. Ainsi, des erreurs d'interprétations interviennent de manière aléatoires et doivent être prises en compte. Une autre partie d'incertitude provient du comportement « stochastique » des utilisateurs : deux utilisateurs ne réagiront pas de la même façon lorsqu'ils seront mis face à la machine. De plus, un même utilisateur peut se comporter différemment s'il s'habitue au comportement de la machine, lorsqu'il constate qu'une

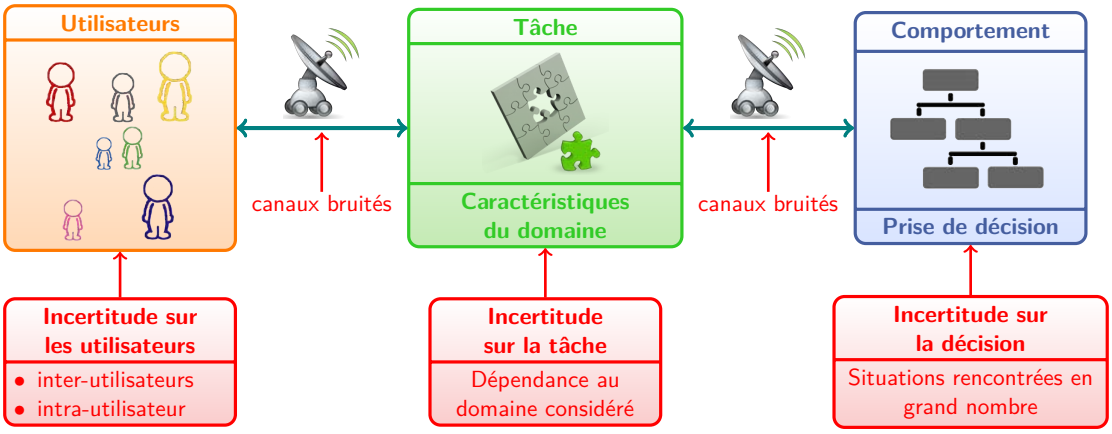



**Figure 1.4** – Identification de différentes sources d'incertitude.

action de sa part ne provoque pas la réaction attendue de la part de la machine ou encore afin de remédier à une évidente erreur de communication. Par ailleurs, les intentions de l'utilisateur restent masquées du point de vue de la machine ce qui introduit une incertitude au niveau de la tâche. Enfin, lorsque les situations rencontrées par la machine sont en très grand nombre, cela ne permet pas la définition d'une solution exacte pour tous les cas rencontrés ce qui représente aussi une source d'incertain.

Souvent, les concepteurs de systèmes interactifs font appel à des modèles pour pallier ces différents problèmes : modèles d'utilisateurs pour gérer les différents comportements, modèles de la tâche pour envisager les différentes solutions en fonction du comportement de l'utilisateur ou encore modèle du canal de communication pour gérer le bruit.

Cependant, la conception de modèles reste, dans la plupart des cas, *ad hoc* à la tâche envisagée et nécessite l'introduction de beaucoup de connaissances *a priori* de la part du concepteur du système. Par exemple, un jeu de règles permettant de définir les différentes étapes de la tâche à effectuer en fonction des informations collectées sur l'utilisateur, est à refaire pour toute autre tâche que celle initialement prévue. De plus, tous les comportements des utilisateurs ne sont pas possibles à prévoir. Des hypothèses simplificatrices sont alors faites ce qui entraîne une perte de généricité de la solution envisagée.

Ce constat a été fait il y a quelques années déjà [Levin 97] et une nouvelle tendance, s'appuyant sur l'apprentissage automatique et l'optimisation à partir de données, a vu le jour. Les techniques d'apprentissage automatique cherchent des solutions à partir de données et nécessitent beaucoup moins d'introduction de connaissances expertes *ad hoc* à la tâche pour parvenir à l'accomplissement de celle-ci sont maintenant reconnues pour traiter du problème de l'interaction homme-machine. En particulier, l'apprentissage par renforcement (*Reinforcement Learning*, RL en anglais) [Sutton 98] qui permet l'optimisation d'une séquence de décisions sous incertitude semble assez bien adapté puisque la gestion d'une interaction peut se voir, du point de vue de la machine, comme un problème de décision séquentielle. Il s'agit en effet de définir les différentes étapes à accomplir par la machine en fonction des informations courantes et passées collectées sur l'utilisateur pour faire évoluer l'interaction vers son but. A la différence des systèmes à base de règles (ou de la planification), la décision suivante de la machine est réévaluée en fonction des conséquences sur l'utilisateur de la décision précédente et seul le but final à atteindre est spécifié à la machine et chaque étape est réévaluée en fonction du

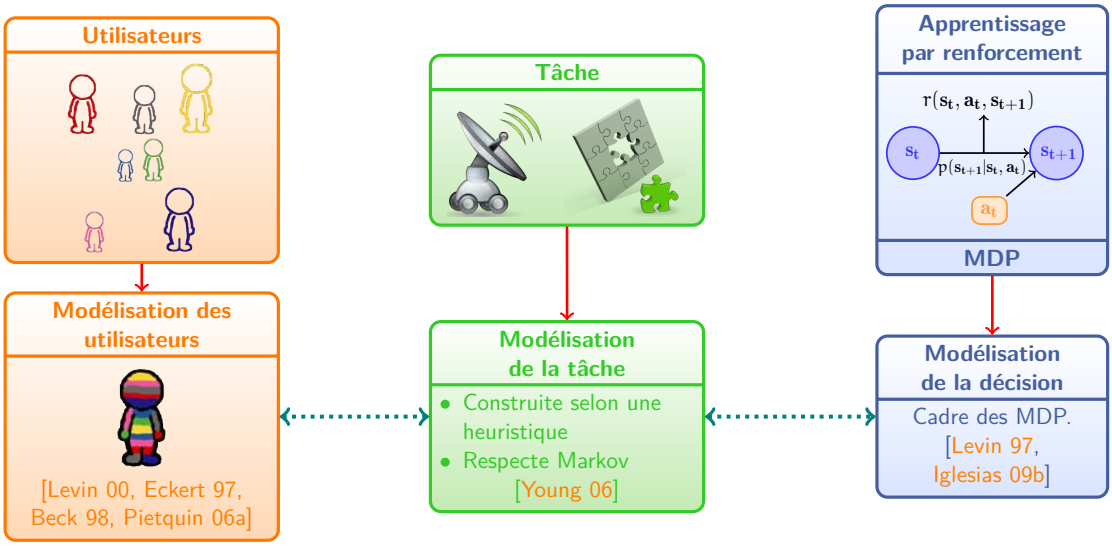



**Figure 1.5** – Modèles proposées pour gérer l'incertitude générale – Ces choix sont liés à l'utilisation du RL.

contexte courant. Ainsi, le caractère non-déterministe de l'interaction est pris en compte dans un paradigme formel bien défini.

Même si l'apprentissage par renforcement permet un gain de généricité puisqu'il permet facilement l'optimisation de différentes tâches, quelques hypothèses de modélisation demeurent. Un résumé des différentes hypothèses de modélisation est proposé Figure 1.5. Tout d'abord, les algorithmes historiques d'apprentissage par renforcement sont gourmands en échantillons. Une simulation des utilisateurs est alors nécessaire afin d'avoir à disposition une quantité de données suffisante pour apprendre une solution optimale. Cette simulation implique l'usage d'un modèle d'utilisateurs pour inférer l'état interne de ces derniers. De plus, les informations fournies au module de prise de décision doivent respecter la propriété de Markov.Enfin, la tâche à optimiser doit pouvoir se traduire dans le cadre mathématique des processus décisionnels de markov (partiellement observables)(*(Partially Observable) Markov Decision Process*, (PO)MDP en anglais) [Bellman 57b]. Les exemples d'applications pour lesquelles le cadre des (PO)MDP a été utilisé sont nombreux et variés : les systèmes de dialogue parlé [Scheffler 02, Pietquin 06b, Lemon 07], l'apprentissage tutoré par ordinateur [Iglesias 09a, Pietquin 11a], la visite guidée d'un musée [Thrun 00], l'aide à la conduite automobile [Pietquin 11e], les systèmes de recommendation [Golovin 04] ou encore un barman robotisé [Foster 12].

De nouvelles compétences expertes sont alors nécessaires pour traduire les contraintes imposées par ce cadre formel dans le contexte de la tâche. Ces compétences n'étant pas nécessairement plus répandues que les compétences *ad hoc* aux différentes tâches, l'introduction de ces méthodes d'optimisation automatique dans la communauté académique et industrielle est fortement ralentie. Par ailleurs, certaines critiques sur le respect de ces contraintes dans les systèmes actuellement développés ont aussi été émises [Paek 08]. Ainsi, l'application du RL pourrait entraîner, selon ces critiques, de nouveaux biais de modélisation, ce qui contribue encore à freiner la progression de ces méthodes d'optimisation dans le cadre de l'interaction homme-machine.

La thèse que ce manuscrit propose de défendre est qu'il est possible de relaxer certaines contraintes imposées par la modélisation de la tâche dans un cadre permettant son optimisation

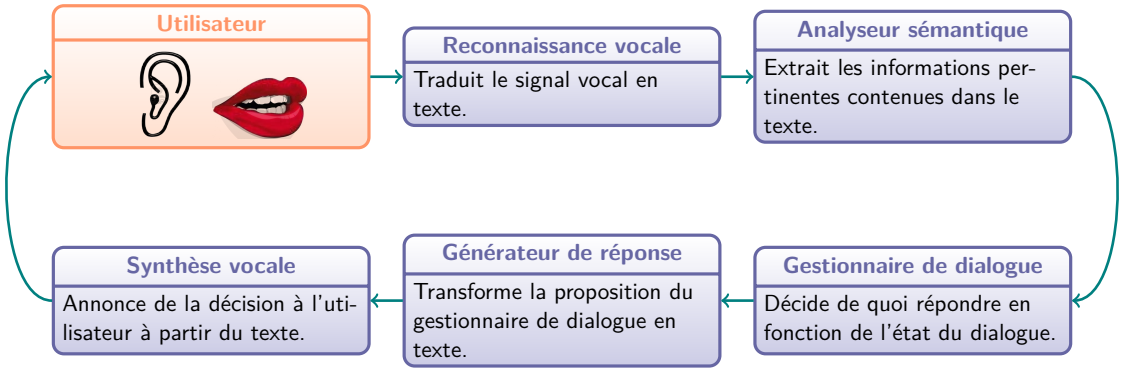



**Figure 1.6** – Présentation du système de dialogue parlé.

par apprentissage par renforcement et ainsi de réduire les biais de modélisation et l'introduction de connaissances expertes *ad hoc*. Les solutions envisagées seront illustrées au travers de deux exemples applicatifs d'interaction homme-machine : les Systèmes de Dialogue Parlé (SDP) et les Environnements Informatiques pour l'Apprentissage Humain (EIAH). Une description de ces systèmes est donnée dans les deux sections suivantes (Sections 1.1.1 et 1.1.2) ainsi que les contraintes qu'il est nécessaire de prendre en compte pour les concevoir.

Ces deux exemples applicatifs ont été choisis car ils permettent d'illustrer des sources d'incertitude différentes. Dans le cas du système de dialogue parlé, l'interaction avec la machine se fait au moyen de la parole qui est un medium encore imparfaitement transcrit et compris par la machine. Toutefois, l'analyse de ce signal physique et de la construction du langage permet de construire par apprentissage machine des modèles de reconnaissance et de compréhension du langage fournissent des taux de confiance quant à leur fiabilité. Grâce à ces données, il est possible de gérer les incertitudes sur l'utilisateur. En revanche, le problème est plus compliqué dans le cas du tutorat. En effet, il s'avère plus complexe d'estimer les compétentes réelles apprises au cours de l'interaction puisque ceci nécessiterait de modéliser le processus de cognition présidant à l'acquisition de connaissances.

### 1.1.1 Systèmes de dialogue parlé

Avec les progrès faits en traitement du signal et plus particulièrement en traitement de la voix humaine, des applications utilisant des conversations parlées ont pu être développées. Nous nous concentrons ici sur des sytèmes de dialogue parlé dont le but est d'accomplir une tâche précise, définie au préalable, *a contrario* des dialogues sociaux qui visent à faire échanger plusieurs parties sans but d'atteindre un objectif [Weizenbaum 66]. Une description du principe du système utilisé est donnée Figure 1.6.

Par exemple, les sytèmes de dialogue utilisés dans ces travaux sont déployés sur les standards téléphoniques dans le but de prendre un rendez-vous ou pour orienter les utilisateurs vers le bon service dans le cadre d'un diagnostic de panne. Dans ces cas, plusieurs personnes doivent pouvoir avoir accès au service sans devoir patienter trop longtemps au téléphone. Au lieu de mettre d'importants moyens humains en place afin de traiter à temps toutes les requêtes, des standards de réponses automatiques sont utilisés. Un exemple de dialogue représentatif des dialogues considérés ici est donné Figure 1.7. Dans cet exemple, l'utilisateur appelle le standard dans le but de prendre rendez-vous à la date qui lui convient pour faire intervenir un technicien.

| Locuteur | Réponse échangée | Début | Nom | Date | Lieu | Fin |
|---|---|---|---|---|---|---|
| Système | « Bonjour ! Comment puis-je vous aider ? | | | | | |
| Utilisateur | - Bonjour, je souhaite prendre rendez-vous. | ✓ | - | - | - | - |
| Système | - Quel est votre nom ? | | | | | |
| Utilisateur | - Daubigney | ✓ | ✓ | - | - | - |
| Système | - Quelle est votre adresse ? | | | | | |
| Utilisateur | - 2 rue Belin à Metz. | ✓ | ✓ | - | ✓ | - |
| Système | - Quelle date vous conviendrait ? | | | | | |
| Utilisateur | - Le 13 août 2013 à 14h. | ✓ | ✓ | ✓ | ✓ | - |
| Système | - D'accord pour le 13 août à 14h, au revoir. | | | | | |
| Utilisateur | - Merci, au revoir. » | ✓ | ✓ | ✓ | ✓ | ✓ |

**Figure 1.7** – Exemple de dialogue.

Trois informations sont considérées : le nom du client, le lieu et la date de l'intervention. Le système de dialogue doit faire en sorte de récupérer ces trois attributs. Les cinq dernières colonnes contiennent les informations reconnues par le système de dialogue juste après que l'utilisateur a répondu (les trois attributs et les signaux de début et de fin de dialogue). Ce sont les informations que le système garde en mémoire pour avoir une connaissance de l'état du dialogue.

Le système doit prendre les décisions nécessaires qui lui permettent d'obtenir tous ces attributs à l'issue de l'échange. Le problème de dialogue est ainsi un *problème de prise de décisions séquentielles*. Il faut que le système décide *quoi* dire et *quand* à partir de la réponse courante de l'utilisateur et des réponses fournies auparavant. Ces décisions sont prises par le gestionnaire de dialogue (Figure 1.6) qui est donc une composante centrale du système de dialogue parlé.

Le dialogue présenté dans l'exemple Figure 1.7 se déroule sans encombres, dans le sens où à aucun moment il n'est demandé à l'utilisateur de se répéter ou de confirmer sa réponse. Toutes les réponses, mêmes intégrées dans une phrase, sont comprises du premier coup par le système de dialogue parlé. En pratique, de tels dialogues sont très rarement rencontrés. Lors d'un dialogue entre deux humains, à cause du bruit ambiant, une partie de la phrase peut ne pas être comprise. A partir des informations du contexte, il est possible que le destinataire de l'information puisse néanmoins comprendre la phrase par voie détournée.

Un exemple de dialogue qui se rapprocherait plus de ceux réels est présenté Figure 1.8. En effet, deux composantes du système (Figure 1.6), la reconnaissance vocale et l'analyseur sémantique, sont sources d'erreurs. Il est possible qu'un mot ne soit pas identifié. En conséquence, le sens de la phrase ne pourra pas être extrait correctement. La combinaison des deux modules propose une liste d'hypothèses les plus problables de ce qui a été dit. A chaque hypothèse est associée un taux de confiance. Le gestionnaire de dialogue doit donc être capable de gérer les différents cas.

Proposer un dialogue fluide et naturel n'est ainsi pas des tâches les plus simples. Comme le système est face à un utilisateur humain, il ne devrait donc pas demander des confirmations incessantes pour valider les hypothèses. Cela risquerait de lasser l'utilisateur. Des stratégies plus subtiles doivent ainsi être mises place. Dans l'exemple proposé, au lieu de faire répéter le nom de l'utilisateur qui a mal été compris, le système le lui fait épeler. L'utilisateur n'a pas ainsi l'impression directe que le système a mal compris. Pour la reconnaissance de la date, le système associe le jour exact. L'utilisateur répond en utilisant aussi le jour ce qui apporte

| Locuteur | Réponse échangée | Début | Nom | Date | Lieu | Fin |
|---|---|---|---|---|---|---|
| Système | « Bonjour ! Comment puis-je vous aider ? | | | | | |
| Utilisateur | - Bonjour, je souhaite prendre rendez-vous. | 95% | - | - | - | - |
| Système | - Quel est votre nom ? | | | | | |
| Utilisateur | - Daubigney | 95% | 65% | - | - | - |
| Système | - Pouvez-vous me l'épeler, s'il vous plaît ? | | | | | |
| Utilisateur | - D-A-U-B-I-G-N-E-Y. | 95% | 99% | - | - | - |
| Système | - Quelle est votre adresse ? | | | | | |
| Utilisateur | - 2 rue Belin à Metz. | 95% | 99% | - | 80% | - |
| Système | - Quelle date vous conviendrait ? | | | | | |
| Utilisateur | - Le 13 août 2013. | 95% | 99% | 60% | 80% | - |
| Système | - Le vendredi 16 août à 14h conviendrait-il ? | | | | | |
| Utilisateur | - Non, pour mardi 13 août. | 95% | 99% | 90% | 80% | - |
| Système | - D'accord, 13 août à 14h, 2 rue Belin à Metz. | | | | | |
| Utilisateur | - Merci, au revoir. » | 95% | 99% | 90% | 80% | 95% |

**Figure 1.8** – Exemple de dialogue réel pour lequel la première hypothèse est prise en compte.

une information supplémentaire. En croisant les renseignements, le nombre de cas possibles se réduit. Le gestionnaire de dialogue ne fait donc plus face à un problème de prise de décisions séquentielles simple, mais à un problème de prise de décisions séquentielles *sous incertitude* puisque plusieurs hypothèses sont à considérer pour chaque décision.

La prise de décision est la partie sur laquelle nous nous concentrons. L'objectif du concepteur de ce module est donc de proposer un dialogue fluide dans lequel toutes les réponses sont acceptées , malgré des informations partielles. Dans ce cadre, les informations d'incertitudes proviennent du traitement du signal vocal. Elles se basent donc sur un signal physique. Le détail des contraintes à prendre en compte pour la conception d'un système efficace est présenté Section 1.2. Nous nous intéressons maintenant à un autre système dont la conception a été permise par les récents progrès technologiques.

### 1.1.2 EIAH

Avec la démocratisation de l'accès à l'ordinateur personnel, un des domaines concernés par le changement d'ergonomie est l'apprentissage tutoré. Habituellement, l'enseignement se fait par le biais d'un professeur qui transmet son savoir à un groupe d'élèves, une classe. Cependant, les élèves sont souvent nombreux et d'un niveau hétérogène : le professeur n'a ni le temps, ni les moyens de proposer un apprentissage personnalisé à chacun de ses élèves, ce qui serait un gage de nets progrès. Il serait donc intéressant de mettre les élèves face à un tuteur automatique pour que chacun puisse progresser à son rythme et bénéficier de conseils personnalisés. Là encore, le concepteur de tels systèmes est face à un défi technique. L'élève interagissant avec l'ordinateur s'attend à avoir un dialogue fluide et adapté à ses besoins. Enseignant et élève vont intervenir à tour de rôle au cours de l'interaction par le bais de différents media : dialogue parlé, dialogue écrit, expression faciale des émotions ou clics de souris sur un écran d'ordinateur par exemple.

L'expertise de l'enseignant réside dans le fait qu'à partir de matériel pédagogique (leçons, lectures, exercices, questionnaires, travaux pratiques), il est capable de décider de la suite d'activités à proposer pour qu'à la fin de l'apprentissage, l'élève ait vu ses connaissances augmenter. Le problème de tutorat peut ainsi également être considéré comme un problème de prise de

décisions séquentielles.

A l'instar d'un enseignant, le tuteur peut proposer différents types d'interactions. Un appel aux compétences orales peut être mis en place si le tuteur est capable de gérer les interactions parlées. Cet aspect serait intéressant pour l'apprentissage de la prononciation d'une langue étrangère. Il peut faire appel aux compétences écrites, par le biais de réponses libres comme par exemple une rédaction ou bien contraindre la réponse parmi un jeu de réponses défini à l'avance. Il peut donner la solution directement ou bien proposer une suite d'indices pour y parvenir. Il peut proposer à l'élève de refaire l'exercice ou un exercice sur le même thème. En fonction des capacités données au tuteur, la difficulté de conception du système sera plus ou moins élevée. Par exemple, il est plus ardu de vérifier la justesse d'une réponse dans une production libre (rédaction, phrase) que de vérifier si les réponses à questionnaire à choix multiples sont correctes.

Du point de vue de l'ordonnancement des activités proposées, différentes solutions sont possibles. L'enseignant peut, par exemple, proposer une suite d'exercices et de leçons dans un ordre pré-défini par le concepteur du système. Le bénéfice apporté est que l'élève progresse à son rythme, sans être obligé de suivre la moyenne de la progression d'un groupe. Le système peut aussi fournir des indices spécifiques aidant à la progression lors de la résolution d'un problème. De même que précédemment, l'avantage d'utiliser un tuteur réside dans le fait que les conseils sont personnalisés. Seuls ceux nécessaires sont donnés. Ceci a l'avantage de ne pas lasser l'élève qui maîtrise déjà un savoir et d'insister sur les points qui posent problème pour d'autres élèves. Cependant, la conception de tels systèmes est moins aisée car la séquence ne peut pas être pré-définie à l'avance. Il serait aussi envisageable que le tuteur, parmi un ensemble de matériel pédagogique (exercices, questionnaires, leçons), choisisse la séquence personnalisée la mieux adaptée à l'élève, sans que la séquence ne soit pré-définie à l'avance par le concepteur. Il pourrait même être envisageable de laisser le tuteur choisir en fonction des préférences de l'élève. Par exemple, s'il progresse mieux en lisant des leçons, les lui proposer ou s'il préfère résoudre des problèmes pour apprendre sur du concret, préférer cette approche.

Cependant, de même que pour le problème de dialogue, les connaissances de l'utilisateur restent inconnues. En effet, même s'il répond de façon exacte à une question, rien ne permet de détecter avec certitude s'il y a répondu correctement par le fait du hasard ou bien s'il maîtrisait réellement le savoir. Inversement, il peut aussi avoir répondu incorrectement à une question lors de la phase de réponse alors que le savoir était maîtrisé. Une des difficultés est donc de pouvoir collecter un maximum d'informations sur l'élève pour s'adapter au mieux. Contrairement au problème de dialogue, les informations d'incertitude ne proviennent pas que du traitement d'un signal physique. Elles sont donc beaucoup plus difficilement quantifiables. Pour pallier ce manque, un modèle d'élève peut être construit. En fonction des activités proposées, le modèle propose une estimation des connaissances de l'élève. Cette estimation est utilisée ensuite pour résoudre le problème de décision. Cependant, le modèle n'est pas basé sur l'analyse d'un signal physique. Les estimations ne sont pas alors aussi précises que dans le cas du dialogue parlé. Il serait alors intéressant de développer des approches sans modèle pour s'affanchir du biais qu'il introduit. Le problème de tutorat est donc un problème de prise de décision séquentielles sous incertitude avec une observabilité partielle des connaissances de l'élève.

Pour étendre l'expertise des machines à des domaines plus complexes, l'étude de l'état de l'art du domaine montre que l'apprentissage par renforcement semble apporter des solutions satisfaisantes pour l'optimisation de séquences de décisions sous incertitude. Cependant, toute modélisation implique de faire des choix *ad hoc* au problème et entraîne une perte de généricité de la solution. Une connaissance experte est alors nécessaire pour que les hypothèses soient les

plus proches possibles de la réalité. Or, dans certains cas, la seule connaissance experte ne suffit pas à construire des modèles fiables. Les travaux présentés dans ce manuscrit proposent une modélisation automatique de la tâche afin de limiter l'introduction de connaissances expertes, de telle sorte à ce que la modélisation choisie soit compatible avec le RL. Cependant, les solutions proposées doivent respecter les contraintes détaillées dans la section suivante, étant donné que l'interaction se fait avec un utilisateur humain.

## 1.2 Contraintes liées à l'interaction

Afin de mettre en place des solutions performantes, permettant de définir un comportement adapté pour faire face aux différents utilisateurs, des contraintes supplémentaires liées à l'utilisateur humain sont à prendre en compte. Les travaux de [Paek 08] proposent une réflexion sur les caractéristiques à prendre en compte pour la conception d'un gestionnaire de dialogue performant et ceux de [Self 90, Anderson 95] proposent une réflexion pour un système de tutorat assisté par ordinateur.

### 1.2.1 Adaptation

L'état du savoir de chaque utilisateur est différent. Il peut avoir ou non déjà interagi avec un système de même type ce qui aura pour conséquence de modifier son propre comportement face à la machine. Il a été montré dans [Komatani 07], par exemple, que dans le cas du dialogue parlé, le temps de réponse de l'utilisateur dépend de son habitude à interagir avec le système. Dans le cas du dialogue, à la permière interaction, à la question « Comment puis-je vous aider ? », un utilisateur ayant l'habitude d'interagir avec le système pourra répondre directement « Bonjour, je suis [nom], je souhaite un rendez-vous pour le treize août 2013 à 14h au 2 rue Belin à Metz » sans attendre que le gestionnaire de dialogue ne lui demande les informations une par une (comme présenté dans l'exemple, qui correspondrait plutôt à un utilisateur débutant). Dans le cas du tuteur, si ce dernier pose toujours les mêmes questions, il y a un risque que l'élève n'apprenne que les réponses sans faire augmenter son savoir.

De plus, si la solution proposée répond à une suite de règles pré-définies à l'avance, toutes ces règles sont à reconstruire dès que l'application est changée. Il est donc intéressant de développer un cadre de travail général en mettant le moins d'*a priori* possible quant à la forme de la solution. Cela évite d'avoir à refaire toute la conception lorsque le thème du problème change, ce qui réduit considérablement les temps de développement. Par exemple, la prise de rendez-vous téléphonique et un guide d'information touristique ne devraient pas nécessiter la réecriture de règles *ad hoc* au problème. De même que pour le tutorat, la solution proposée devrait pouvoir s'appliquer indépendamment de la matière enseignée ou du type d'activités pédagogiques choisies.

Pour des raisons de performance et de réutilisation de solutions existantes, il est important que la solution proposée puisse s'adapter à l'environnement dans lequel elle évolue. Le concepteur a donc interêt à ce que la solution soit apprise en fonction des situations rencontrées et continue à être améliorée au cours des interactions. Comme il est impossible de prévoir toutes les situations, du fait de l'incertitude sur les intentions de l'utilisateur, si la solution est déterminée à l'avance, un fort risque existe de ne pas savoir comment agir dans une situation particulière. Dans ce cas, l'utilisateur risque de mettre fin prématurément à l'interaction avec la machine.

### 1.2.2 Comportement cohérent

L'interaction avec l'utilisateur doit se faire de façon la plus cohérente possible jusqu'à la fin prévue par le concepteur du système (par exemple, dans le cas du dialogue que la date d'un rendez-vous puisse être fixée ou que, dans le cas du tutorat, un examen final soit réussi). Un comportement fluide et cohérent implique de pouvoir faire face à toutes les situations possibles et de prendre une décision appropriée en conséquence. Les solutions proposées par un jeu de règles pré-définies à l'avance ne rentrent alors pas dans ce cadre, étant donné qu'il est impossible, en laissant l'utilisateur répondre librement, de prévoir toutes les situations. Une possibilité pour résoudre ce problème est de laisser à la solution l'opportunité d'apprendre à agir en cas de situation nouvellement rencontrée (ce qui permet entre autres de gérer le problème d'adaptabilité).

Le problème, dans ce cas, est qu'à chaque nouvelle situation rencontrée, le système ne sait pas comment agir. Il y a un risque qu'il présente une décision qui perturbe le fil de l'interaction. Le but pour le concepteur du système est de faire face à ces nouvelles situations sans que l'expérience de l'utilisateur ne soit trop impactée. Deux solutions pour prendre en compte ce problème peuvent être mises en œuvre.

#### Différents types d'apprentissages

La première approche consiste à apprendre la solution optimale à partir d'exemples d'interactions déjà collectés. Dans ce cas, l'utilisateur n'est pas directement face à la politique apprise par le système et l'apprentissage de la solution est dit *off-policy* et *hors-ligne*. L'aspect *off-policy* signifie que la politique ayant servi à générer les données d'apprentissage, appelée *politique de contrôle*, est différente de celle apprise. L'aspect hors-ligne se traduit par le fait que des traces d'interactions préalablement collectées sont utilisées pour l'apprentissage. Une condition pour que cette méthode soit efficace réside dans le fait que le jeu de données employé pour l'apprentissage doit être représentatif de toutes les interactions possibles. Il faut donc que la politique de contrôle présente une phase d'exploration suffisante. Nous mentionnons juste l'approche *on-policy* et hors-ligne sans la développer puisqu'elle n'est que d'un intérêt limité. En effet, l'aspect *on-policy* signifie que la politique ayant servi à générer les données et celle calculée par l'algorithme sont identiques. Il n'y donc pas d'amélioration vers une solution optimale.

La seconde approche consiste à apprendre le comportement *en-ligne*. Dans ce cas, l'apprentissage de la politique se fait au fur et à mesure d'interactions entre la machine et l'utilisateur. De façon similaire au cas hors-ligne, l'apprentissage peut être soit *on-policy* soit *off-policy*. Dans le premier cas, la politique apprise est celle directement présentée à l'utilisateur, qui fait ainsi face à chaque nouvelle décision prise par la machine. Il est donc nécessaire de rester précautionneux dans le choix des décisions dans le cas en-ligne et *on-policy*.

#### Dilemme entre exploration et exploitation

Pour qu'un apprentissage soit efficace, il est nécessaire d'explorer les différentes possibilités [Kaelbling 96]. En effet, dans une situation donnée, la machine est face à deux choix. Soit une décision dont les conséquences n'ont jamais (ou peu) été expérimentées est prise, soit c'en est une dont les conséquences peuvent être connues. Dans le premier cas, les conséquences peuvent être hasardeuses et un comportement incohérent de la machine peut en résulter. Ou bien, au contraire, le fait d'avoir testé cette nouvelle décision peut en révéler une meilleure que toutes celles testées auparavant pour une même situation de départ.

Dans l'exemple d'un dialogue parlé, si le gestionnaire de dialogue demande systématiquement une confimation par une question oui/non dont la réponse est facile à reconnaître, il va parvenir à mener la tâche. Mais si à un certain moment, il choisit de ne plus demander de confirmation et de passer directement à la question suivante, il va s'apercevoir qu'il peut parvenir à accomplir la tâche tout en augmentant la fluidité du dialogue. S'il s'était contenté de garder la décision initiale, il n'aurait pas pu trouver de meilleure politique. Pour l'exemple du tuteur, s'il ne teste pas de nouvelle décision lors de l'apprentissage, il ne détectera pas forcément que plusieurs types d'élèves interagissent avec lui et, qu'en conséquence, la politique présentée doit être adaptée.

Ce compromis est connu sous le nom de dilemme *exploration-exploitation*. Le choix se fait entre tester une nouvelle décision, l'*exploration*, ou le fait d'en suivre une dont les conséquences sont déjà connues, l'*exploitation*. L'exploration est nécessaire pour trouver des solutions les plus performantes possibles. Cependant, il arrive que le comportement qui en résulte soit incohérent. Par exemple, dans le cas du dialogue, cela pourrait se traduire par le fait de poser une question alors que la réponse est déjà connue. Dans celui du tutorat, cela pourrait correspondre à poser des questions sur un savoir nécessitant des bases n'ayant pas été abordées précédemment au cours de la session. Une solution consiste alors à utiliser une simulation d'utilisateurs pour explorer toutes les situations possibles sans gêner les utilisateurs réels [Paek 06]. Cela présente néanmoins l'inconvénient d'introduire un modèle supplémentaire. Le mieux est donc de proposer des solutions qui essayent le plus rapidement possible d'explorer de nouveaux cas sans perturber les utilisateurs. Ceci est d'intérêt lorsque des solutions en-ligne et *on-policy* sont préférées.

En résumé, l'approche en-ligne et *off-policy* consiste à apprendre une politique en intéragissant directement avec l'utilisateur grâce à une politique de contrôle différente. Cette approche a l'avantage de ne pas présenter de décisions trop aléatoires à l'utilisateur. L'approche en-ligne et *on-policy*, sous réserve de proposer un schéma d'exploration sûr, comparée aux approches en-ligne et *off-policy*, et hors-ligne, présente l'avantage de ne pas à avoir à choisir le moment auquel passer à la politique apprise. En effet la politique présentée est continuellement améliorée. En revanche, l'apprentissage doit être le plus efficace possible étant donné qu'au début, la politique peut être très incohérente.

La combinaison des deux approches est aussi envisageable. Au début, l'apprentissage se fait *off-policy*, soit en-ligne soit hors-ligne, avec une politique de contrôle basique implémentée à la main, puis, une fois que la politique apprise est suffisamment améliorée, un apprentissage en-ligne et *on-policy* est proposé.

### 1.2.3 Généralisation

Lorsque un humain interagit avec une machine, afin qu'il ne soit pas déstabilisé, il est important qu'elle puisse gérer toutes les situations potentiellement rencontrées. Cependant, il n'est pas possible de toutes les envisager *a priori* parce qu'elles sont en trop grand nombre voire même parce qu'elles sont en nombre infini. Il est alors important de proposer une approche permettant de choisir des décisions efficaces même si la situation n'a jamais été rencontrée. Ces méthodes généralisent la solution à partir de points d'échantillons et sont capables d'inférer les connaissances là où les données manquent.

### 1.2.4 Rapidité de l'apprentissage

Que le choix se porte sur une approche hors-ligne ou en-ligne, il est souhaitable que la solution proposée soit efficace sur les données. Cette caractéristique implique un apprentissage

rapide nécessitant un minimum de données pour être performant. En effet, la collecte de données réelles peut se révéler coûteuse puisqu'il faut contacter un nombre important d'utilisateurs pour avoir une bonne représentativité des différents comportements. De plus, une annotation des données à la main est parfois nécessaire afin d'identifier les bons comportements. Par ailleurs, plus la solution est apprise rapidement, moins d'utilisateurs sont potentiellement touchés par des comportements incohérents au cours de l'apprentissage de la solution optimale.

## 1.3 Conclusion

L'interaction homme-machine peut se modéliser comme un problème d'optimisation de prise de décision séquentielle sous incertitude. L'objectif du concepteur du système est de proposer une solution générique afin d'introduire le moins de biais possible et d'apprendre à partir seulement des données. En conséquence, les coûts de développement sont réduits car il n'y a pas besoin d'apporter une expertise pour chaque domaine d'application considéré. Pour cela, les techniques d'apprentissage automatique comme l'apprentissage par renforcement semblent particulièrement prometteuses et ont déjà été proposées dans la littérature. Seulement, sans adapation, elles nécessitent elles aussi une connaissance experte pour pouvoir être utilisées de façon adéquate. Par exemple, il est nécessaire que les informations sur lesquelles se base la décision de la machine respectent la propriété de Markov, ce qui nécessite un important travail d'ingénierie. Les connaissances d'experts en linguistique ou en éducation sont alors transférées à des experts en apprentissage machine. Les travaux présentés dans ce manuscrit visent à réduire la quantité d'*a priori* nécessaire pour transposer une tâche d'interaction dans un format compatible avec le RL. Pour ce faire, quatre points ont été identifiés. Ils sont présentés dans le chapitre suivant.

# Chapitre 2

# Contributions

Nous avons étudié au chapitre précédent les contraintes à prendre en compte pour que les interactions homme-machine soient les plus naturelles possibles. Depuis quelques décennies, une méthode appelée apprentissage par renforcement [Sutton 98] a montré son efficacité sur de tels problèmes. Elle se base sur un apprentissage de la solution à partir d'essais et d'erreurs. La façon dont sont traitées les contraintes sera présentée plus en détails par la suite. La méthode est une approche bio-inspirée. Lorsqu'un animal reçoit une récompense extérieure pour sanctionner un bon comportement, le lien entre la situation dans laquelle il est et la décision juste effectuée s'en trouve renforcée. Au contraire, s'il n'est pas récompensé, il n'associe pas sa situation à la récompense. Le but pour l'animal est d'apprendre à se comporter de telle façon à maximiser le cumul des récompenses qu'il obtient au cours de l'échange.

Le but est de chercher à faire reproduire cette attitude par la machine. Celle-ci se trouve dans un environnement, duquel elle perçoit des informations. A partir de ces dernières et des décisions qu'elle prend, elle va chercher à maximiser le cumul de récompenses sur le long terme. Aucun *a priori* n'est donné à la machine sur l'environnement dans lequel elle se trouve. A partir d'essais pour lesquels une récompense est donnée à chaque fois, elle est capable d'apprendre le comportement souhaité. L'apprentissage par renforcement est donc une méthode d'apprentissage automatique (*machine learning* en anglais).

La récompense est définie par rapport à l'objectif que veut voir atteint le concepteur du système par la machine mais ne spécifie pas la manière d'y parvenir. Elle doit être suffisamment informative pour que le système « comprenne » ce qui est attendu de lui. Le but pour le système est de trouver la suite de décisions, en fonction d'une situation de départ, qui lui permette de maximiser le cumul des récompenses sur le long terme. Résoudre le problème de l'apprentissage par renforcement revient donc à résoudre un problème d'optimisation. Les récompenses peuvent être construites de façon à privilégier les solutions qui parviennent rapidement à l'objectif. Lorsqu'un humain intervient, la définition de la récompense est liée à sa satisfaction à l'issue de l'interaction. Dans le cas du dialogue parlé, la satisfaction peut être définie en fonction de la longueur de dialogue ou de l'objectif atteint [Walker 97]. Dans le cas du tutorat, la récompense peut être fonction de la progression de l'élève.

Des solutions au problème d'optimisation de la séquence d'interactions entre un utilisateur humain et une machine sont envisagées sous l'angle du RL dans le cadre des travaux présentés dans ce manuscrit. Une présentation plus formelle de cette méthode est donnée au Chapitre 3.

Les travaux de ce manuscrit proposent la relaxe de contraintes imposées par le RL en quatre étapes afin de gagner en généricité et de réduire le recours à des modèles *ad hoc*. Leur description est la suivante. Les algorithmes « historiques » qui permettent de résoudre le problème du RL

requièrent un grand nombre de données pour trouver la solution optimale. L'utilisation d'une simulation d'utilisateurs est alors dans la plupart des cas envisagée [Sison 98, Tetreault 06, Pietquin 06a]. Cependant, cette solution n'est pas satisfaisante car elle nécessite l'introduction d'un modèle d'utilisateurs reproduisant son comportement comme vu par la machine. D'autre part, des algorithmes de RL plus performants sur les données ont été découverts. Il est alors devenu envisageable d'apprendre directement à partir des interactions avec les utilisateurs ce qui limite l'introduction de biais lié au modèle. Cependant, si les utilisateurs sont directement face à la stratégie de la machine, l'apprentissage doit être le plus rapide possible. En effet, comme la stratégie est apprise par essais-erreurs, si le comportement est incohérent, l'utilisateur risque d'abandonner la réalisation de la tâche ce qui ne permettra pas une bonne représentativité des données collectées. Les premiers travaux présentés visent à améliorer la gestion de l'exploration lors de l'apprentissage de la solution optimale afin qu'il soit le plus rapide possible.

Le point abordé par la suite concerne l'amélioration du cadre dans lequel sont habituellement résolus les problèmes d'apprentissage par renforcement. La plupart des problèmes de RL sont posés dans le cadre des processus décisionnels de Markov (*Markov Decision Process*, MDP en anglais) dont plus de détail est donné à la Section 3.1.1. Cependant, dans le cas considéré, ce modèle n'est pas tout à fait adapté car il suppose que le processus de prise de décision peut s'appuyer sur des informations suffisantes ce qui est rarement vrai pour des problèmes réels où ces informations sont souvent non-observables.

Or, dans le cas d'une interaction homme-machine, les intentions de l'utilisateur ne peuvent pas être déterminées avec précision. Dans ce cas, la situation est dite *partiellement observable*. Des informations sur les intentions peuvent être fournies, ce sont des *observations*, mais elles ne sont que le reflet partiel de la réalité. Il est toutefois nécessaire qu'elles soient les plus complètes possibles pour que la décision de la machine puisse se faire sur de solides bases, au risque sinon de la voir exposer un comportement incohérent. La machine doit être capable d'agir tout en gérant l'incertitude sur les informations.

Le cadre des MDP est modifié pour donner lieu aux processus décisionnels de Markov *partiellement observables* (*Partially Observable MDP*, POMDP en anglais) lorsqu'il est utilisé dans le cadre d'interaction homme-machine [Roy 00, Williams 05a]. Cependant, il est connu que le passage à l'échelle pour gérer un grand nombre de situations rencontrées est difficile dans ce cadre. Le deuxième point abordé dans ces travaux concerne donc ce passage à l'échelle.

Le troisième point abordé concerne l'hypothèse de Markov. Il est nécessaire que cette dernière soit respectée pour appliquer la plupart des solutions au problème de RL. Or cette condition n'est que rarement garantie sur des problèmes réels du fait de l'observabilité partielle [Paek 06]. La solution envisagée est donc de s'affranchir de l'apprentissage par renforcement en cherchant directement la solution optimale dans l'espace des politiques grâce à une optimisation en boîte noire.

Enfin, les POMDP suggérant une complexité du modèle accrue par rapport aux simples MDP, nous proposons une approches permettant de construire une représentation d'état markovienne automatiquement à partir des observations afin de nous replacer dans le cadre des MDP et appliquer les méthodes standard de RL. Pour cela, une méthode d'apprentissage automatique basée sur des réseaux de neurones récurrents est utilisée.

Les quatre points présentés sont illustrés sur les problèmes applicatifs présentés Partie II pour le problème de dialogue parlé et Partie III pour le problème de tutorat.

## 2.1 Plan du manuscrit

Le manuscrit se découpe en trois parties et le plan suit l'ordre des quatre points présentés en introduction de ce chapitre. La **Partie I** qui est une partie introductive, s'organise autour de trois chapitres. Le **Chapitre 1** présente la problématique globale. Viennent ensuite les contributions des ces travaux au **Chapitre 2** et la description au **Chapitre 3** des outils employés, l'apprentissage par renforcement et les algorithmes classiques de résolution.

La **Partie II** présente les travaux sur les systèmes de dialogues. Le **Chapitre 4** est consacré à l'état de l'art sur les systèmes de dialogue parlé ainsi que les problématiques à prendre en compte à leur conception. La question de la gestion du dilemme exploration/exploitation pour s'affranchir d'un modèle d'utilisateur est présentée au **Chapitre 5** à travers l'utilisation d'un algorithme convenant particulièrement à la gestion d'interactions homme-machine. Dans ce chapitre, le dialogue est placé dans le cadre des POMDP pour sa résolution. Le chapitre suivant (**Chapitre 6**) est consacré à la gestion de l'observabilité partielle au travers de deux axes, qui concernent respectivement la question du passage à l'échelle du POMDP abordée dans la **Section 6.1** et la question de la contrainte du respect de la propriété de Markov à la **Section 6.2**. La solution apportée pour traiter de ce dernier axe est la recherche dans l'espace des politiques.

Enfin, la **Partie III** propose les contributions au problème de tutorat assisté par ordinateur. Après une présentation de l'état de l'art proposées au **Chapitre 7** et une expérience préliminaire sur un système de tutorat apprenant avec des données déjà collectées au **Chapitre 8**, le **Chapitre 9** présente le quatrième contribution de ces travaux. Elle consiste à construire de façon automatique une représentation Markovienne basée sur l'historique des interactions. Cette méthode permet la prise en charge de la question de l'observabilité partielle et constitue un pas supplémentaire vers le traitement par apprentissage automatique de toute la chaîne d'information, du moment où elle est émise par l'utilisateur à celui où la décision lui est retournée.

Le **Chapitre 10** clôt les travaux présentés dans ce manuscrit et propose de futures pistes de recherche.

## 2.2 Contributions détaillées

### 2.2.1 Contributions au problème de dialogue

L'état de l'art sur le RL appliqué au dialogue parlé est présenté Section 4.1. Viennent ensuite les deux contributions au problème de dialogue.

Le premier point est la question de l'exploration de l'espace d'états, abordé Chapitre 5. Il est considéré dans un cadre plus général utilisant l'algorithme KTD (pour *Kalman Temporal Differences* en anglais). Après une présentation de cet algorithme donnée Section 5.2.1, ses avantages à une application au problème de dialogue sont fournis Section 5.2.2. Les travaux présentés ont conduit à la publication [Daubigney 12a]. Deux aspects sont développés en particulier : la gestion de l'exploration [Daubigney 11a, Daubigney 11c] et la rapidité de l'apprentissage grâce à une représentation non-linéaire de la fonction de qualité. Ils sont testés sur la tâche *CamInfo* (Section 5.1), développée par l'Université de Cambridge, respectivement Section 5.3 et Section 5.4.

Le second point est la gestion de l'observabilité partielle abordée au Chapitre 6. Ce point est très important pour une gestion cohérente du dialogue. En effet, les intentions de l'utilisateur restent masquées car des erreurs sont introduites par la reconnaissance vocale et l'analyseur

sémantique. Deux façons de le prendre en compte sont présentées. La première consiste à utiliser plus d'information à chaque tour de dialogue venant de l'utilisateur, ce qui réduit l'incertitude [Daubigney 12c, Daubigney 12b]. Le test de cette approche est présenté Section 6.1. L'autre façon est de travailler directement dans l'espace des politiques au lieu de passer par le cadre des MDP et des équations de Bellman. En effet, ces équations ne peuvent s'écrire que si la propriété de Markov est vérifiée. Or dans le cas du dialogue, il n'y a aucune garantie stricte sur les états construits à partir des observations retournées par l'utilisateur. L'utilisation d'une méthode particulière de recherche directe dans l'espace des politiques permet d'apprendre de bonnes solutions [Daubigney 13c, Daubigney 13b]. Cette approche est présentée Section 6.2.

### 2.2.2 Contributions au problème de tutorat

L'état de l'art pour les systèmes de tutorat est présenté au Chapitre 7. Suivent ensuite deux contributions. La première propose la mise en œuvre d'un algorithme hors-ligne pour proposer une solution au problème du tutorat au Chapitre 8. Après une présentation des différents modèles et simulations utilisés, respectivement Section 8.1.1 et Section 8.1.2, les résultats sont fournis Section 8.2 et ont été publiés dans [Pietquin 11a, Daubigney 12d, Daubigney 11b]

L'autre contribution exposée au Chapitre 9 concerne la gestion de l'observabilité partielle dans le cadre du tutorat. Pour résoudre ce problème, habituellement, un modèle d'élève est utilisé. Il maintient une approximation du savoir de l'élève en fonction des différentes réponses obtenues. Cependant, ces modèles introduisent un biais difficilement mesurable. Une approche sans modèle est préférée, basée sur les *Echo State Networks* (Section 9.2). Grâce à ceux-ci, une représentation markovienne qui prend en compte l'historique des interactions est construite. La méthode est développée Section 9.3.1. Elle est ensuite testée et les résultats présentés Section 9.3.2. Ces travaux ont été publiés dans [Daubigney 13a].

D'autre part, la conception d'un système de tutorat fait intervenir différents acteurs. En effet, il est nécessaire de construire le matériel pédagogique qui permet au mieux de faire progresser l'élève, de détecter les notions clés permettant la maîtrise totale d'un savoir, d'implémenter les différentes idées du point de vue informatique et aussi d'avoir un retour des enseignants et d'utiliser leur expertise pour améliorer le système de tutorat. Une discussion entre les différents acteurs impliqués dans la conception d'un *environnement informatique pour l'apprentissage humain* (EIAH) a été menée lors d'un atelier de travail. Les résultats et conclusions sont présentées dans [Lefevre 12].

## 2.3 Liste des publications

Le choix a été fait de publier les contributions liées à cette thèse dans des conférences et journaux concernés par les deux tâches visées (SDP et EIAH) afin de leur assurer la meilleure visibilité. Ainsi les communications suivantes ont été publiées :

### 2.3.1 Liées à la partie dialogue parlé

#### Journal international

- L. Daubigney, M. Geist, S. Chandramohan & O. Pietquin. *A Comprehensive Reinforcement Learning Framework for Dialogue Management Optimisation*. IEEE Journal of Selected Topics in Signal Processing, vol. 6, no. 8, pages 891–902, 2012

**Conférences internationales**

- L. Daubigney, M. Geist & O. Pietquin. *Particle Swarm Optimisation of Spoken Dialogue System Strategies*. In Proceedings of Interspeech, 2013
- L. Daubigney, M. Geist & O. Pietquin. *Off-policy Learning in Large-scale POMDP-based Dialogue Systems*. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4989 – 4992, 2012
- L. Daubigney, M. Gašić, S. Chandramohan, M. Geist, O. Pietquin & S. Young. *Uncertainty management for on-line optimisation of a POMDP-based large-scale spoken dialogue system*. In Proceedings of Interspeech, 2011

**Conférences francophones**

- L. Daubigney, M. Geist & O. Pietquin. *Optimisation par essaims particulaires de stratégies de dialogue*. In Journées Francophones de Planification, Décision et Apprentissage (JFPDA), 2013
- L. Daubigney, M. Geist & O. Pietquin. *Apprentissage off-policy appliqué à un système de dialogue basé sur les PDMPO*. In Actes du Congrès francophone sur la Reconnaissance de Formes et l'Intelligence Artificielle (RFIA), 2012
- L. Daubigney, M. Geist & O. Pietquin. *Gestion de l'incertitude pour l'optimisation en ligne d'un gestionnaire de dialogues parlés à grande échelle basé sur les POMDP*. In Journées Francophones de Planification, Décision et Apprentissage (JFPDA), 2011

### 2.3.2 Liées à la partie tutorat

**Conférence internationale**

- L. Daubigney, M. Geist & O. Pietquin. *Model-free POMDP optimisation of tutoring systems with echo-state networks*. In Proceedings of the SIGdial workshop, 2013

**Atelier international**

- O. Pietquin, L. Daubigney & M. Geist. *Optimization of a Tutoring System from a Fixed Set of Data*. In Proceedings of the ISCA workshop on Speech and Language Technology in Education (SLaTe), 2011

**Journal francophone**

- M. Lefevre, J. Broisin, V. Butoianu, P. Daubias, L. Daubigney, F. Greffier, N. Guin, S. Jean-Daubias, R. Monod-Ansaldi & H. Terrat. *Personnalisation de l'apprentissage : comparaison des besoins et approches à travers l'étude de quelques dispositifs*. Revue des Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation (Sticef), 2012

**Conférences francophones**

- L. Daubigney, M. Geist & O. Pietquin. *Apprentissage par renforcement pour la personnalisation d'un logiciel d'enseignement des langues*. In Actes de la Conférence sur les Environnements Informatiques pour l'Apprentissage Humain (EIAH), 2011

- L. Daubigney, M. Geist & O. Pietquin. *Optimisation d'un tuteur intelligent à partir d'un jeu de données fixé*. In Actes des Journées d'Etudes de la Parole (JEP), pages 241–248, 2012

### 2.3.3 Autre publication

**Conférence internationale**

- L. Daubigney, M. Geist & O. Pietquin. *Random Projections: a Remedy for Overfitting Issues in Time Series Prediction with Echo State Networks*. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013

# Chapitre 3

# L'apprentissage par renforcement

Ce chapitre présente les outils utilisés pour améliorer les interactions homme-machine, avant de présenter leur utilisation spécifique dans les cadres du dialogue parlé et du tutorat, respectivement dans les Parties II et III.

Dans le paradigme de l'apprentissage par renforcement (*Reinforcement Learning*, RL en anglais) [Bertsekas 95, Sutton 98] dont la définition est basée sur les travaux de [Thorndike 32, Bellman 57b], l'**agent** est celui qui apprend à contrôler un système, c'est-à-dire qu'il apprend à associer un comportement à chaque situation à laquelle il fait face. L'ensemble de ces situations constitue son **environnement**. Le principe du RL est présenté Figure 3.1. L'agent peut avoir une influence sur son environnement par le biais d'**actions**. En retour, il perçoit des informations qui le renseignent sur le statut des différentes caractéristiques de l'environnement, appelées **états**, et une **récompense**. Cette dernière contient une quantification locale de la qualité de l'interaction. Plus elle est élevée, plus l'action choisie pour une situation donnée est appropriée. L'objectif pour l'agent est d'obtenir le cumul moyen maximum de récompenses sur le long terme.

Dans le cadre des interactions homme-machine, l'agent est celui qui doit apprendre des décisions en fonction de la volonté d'un humain avec lequel il interagit. L'utilisateur ainsi que toutes les informations disponibles à son sujet constituent donc son environnement. Elles sont fournies grâce à des capteurs, comme par exemple un enregistrement audio de la voix ou vidéo

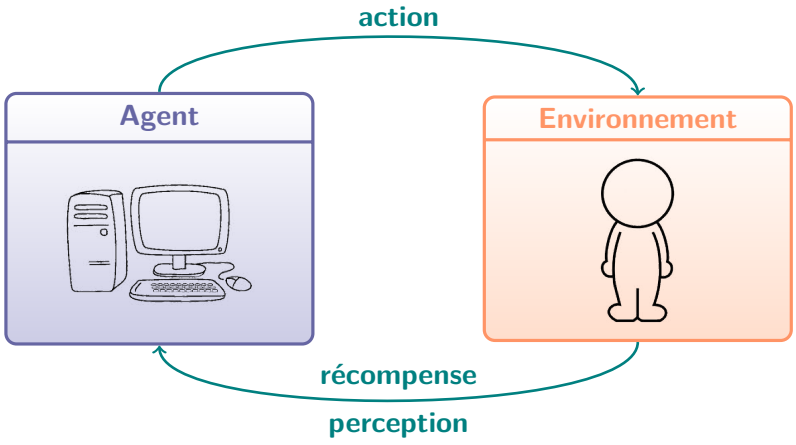



**Figure 3.1** – Principe de l'apprentissage par renforcement.

des expressions du visage, de la posture ou des réponses écrites à une question posée. Plus les informations recueillies seront précises, plus le comportement de l'agent sera efficace. Le RL convient particulièrement aux interactions homme-machine car la solution est apprise à partir de données, par une méthode d'essais-erreurs et donc sans modèle d'utilisateur. Au début de l'apprentissage, l'agent ne connaît aucune information *a priori* sur son environnement. Au fur et à mesure des interactions, grâce à la récompense, il va apprendre à associer quel est le comportement attendu étant donné la situation dans laquelle il se trouve.

Cette approche est intéressante du point de vue des coûts de développement. Comme elle est générique, des règles spécifiques à chaque problème n'ont pas besoin d'être définies. Réserver un billet d'avion avec le système de dialogue parlé ou bien prendre rendez-vous avec un technicien serait considéré comme deux tâches différentes alors qu'elles s'abstraient toutes deux comme des tâches de remplissage de champs. De même pour le problème de tutorat, enseigner les mathématiques et une langue étrangère ne serait pas considéré comme une même tâche. Avec l'apprentissage par renforcement, la modélisation du problème se résume à quantifier la qualité de l'interaction par une récompense tous les problèmes se traitent ensuite par une méthode de résolution unique.

La connaissance de la dynamique du système à contrôler n'est pas nécessaire. Dans le cas d'interactions homme-machine, cela reviendrait à connaître le fonctionnement du cerveau humain. Si c'était le cas, l'influence d'une action serait alors prédictible, ainsi que les variations de comportements inter-utilisateurs. Or, les méthodes actuelles ne permettent pas de constuire un modèle fiable de la cognition humaine. La seule contrainte pour poser un problème de contrôle comme un problème d'apprentissage par renforcement est la définition d'une fonction de récompense quelle que soit l'action proposée par l'agent et l'état dans lequel se trouve l'environnement.

Ce chapitre aborde les questions générales soulevées par le choix de l'apprentissage par renforcement. Une présentation du cadre de résolution habituellement employé est donnée Section 3.1. La Section 3.2 propose une solution permettant de gérer les cas réels où le nombre de situations possibles est trop grand pour être traité en l'état. Enfin, la Section 3.3 énumère les contraintes inhérentes au choix du RL qui peuvent poser problème dans le cadre d'une interaction homme-machine et la façon dont elles sont prises en compte par les algorithmes de résolution de référence.

## 3.1 Présentation générale

Cette section présente tout d'abord le cadre habituel utilisé pour résoudre le problème d'apprentissage par renforcement Section 3.1.1 avant de définir quelques notions essentielles Section 3.1.2. Elle se termine par l'introduction de méthodes standard de résolution Section 3.1.3.

### 3.1.1 Cadre de résolution

La majorité des solutions proposée utilise le cadre des processus décisionnels de Markov (*Markov Decision Process*, MDP en anglais) [Bellman 57b, Puterman 94]. La définition des MDP est donnée dans la section suivante, avant de présenter les limites de cette approche pour une application aux interactions homme-machine.

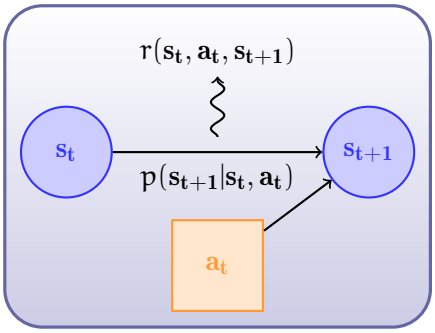



**Figure 3.2** – Interaction de l'agent avec son environnement.

### Présentation d'un MDP

Un MDP est un cadre formel permettant de décrire le comportement d'un agent. Sa définition est la suivante :

**Définition 3.1.** *Un Processus Décisionnel de Markov est un n-uplet* $\{S, A, T, R, \gamma\}$ *où :*

- $S$ *est l'environnement de l'agent, l'espace d'états (fini)*
- $A$ *est l'ensemble des actions à disposition de l'agent, l'espace d'actions (fini)*
- $T$ *est un jeu de probabilités markoviennes définies telles que* $T = \{p(s'|s, a), \forall (s', s, a) \in S \times S \times A\}$
- $R$ *est la fonction de récompense définie telle que* $R : S \times A \times S \to \mathbb{R}$
- $\gamma$ *est un facteur d'actualisation tel que* $\gamma \in ]0, 1[$.

L'agent évolue dans son environnement par des actions $a \in A$, dont il perçoit des informations $s \in S$ et $s' \in S$ (Figure 3.2). A partir d'une situation $s_t$, après avoir choisi l'action $a_t$, l'agent va se retrouver en $s_{t+1}$ et recevoir la récompense $r_t = R(s_t, a_t, s_{t+1})$. Le n-uplet $(s_t, a_t, s_{t+1}, r_t)$ constitue une *transition*. La probabilité de transiter d'un état à un autre est donnée par T. Il est à noter que les probabilités sont markoviennes. Après avoir rencontré une suite d'états $s_0, s_1, ..., s_t$ et après avoir effectué la suite d'actions $a_0, a_1, ..., a_t$, la probabilité de se retouver dans l'état $s_{t+1}$ ne dépend que de $s_t, a_t$ et non de l'historique des états/actions rencontrés : $p(s_{t+1}|s_0, ..., s_i, a_0, ..., a_t) = p(s_{t+1}|s_t, a_t)$. Cette propriété est importante car cela signifie qu'il n'y a pas de mémoire mise en jeu. Dans le cadre de l'apprentissage par renforcement, le modèle de la dynamique du système n'est pas disponible : ni le jeu de probabilités T ni la fonction de récompense R ne sont connus.

Le comportement de l'agent est décrit par une fonction appelée politique $\pi$. Nous nous intéressons dans ce manuscrit aux politiques *déterministes*. Cette fonction associe à chaque situation rencontrée dans l'environnement une action : $\pi : S \to A$ (et non une distribution sur les actions pour le cas non-déterministe). La politique optimale est celle qui permet d'obtenir le plus important cumul de récompenses dans toutes les situations : $E\left[\sum_i \gamma^i r_i\right]$. Ici nous choisissons de pondérer les récompenses par le critère $\gamma$ afin de donner plus d'importance aux récompenses obtenues le plus tôt possible et de traiter des problèmes en horizon infini. La résolution du MDP consiste à déterminer la politique optimale.

### Limites à l'utilisation de cet environnement

Lorsque des problèmes réels sont considérés, comme par exemple les interactions homme-machine, l'application du cadre des MDP reste difficile du fait du respect de la propriété de

Markov. Par exemple, si un complément d'information est demandé par le système, ce dernier doit se souvenir de celle principale pour pouvoir mettre en lien la précédente réponse de l'utilisateur et celle nouvellement donnée. Il est ainsi nécessaire d'extraire et de retenir les informations apportées par chaque interaction pour le bon déroulement global de l'échange. De plus, les données renvoyées par l'utilisateur restent *partielles* puisque la machine n'a jamais accès aux mécanismes qui régissent la cognition. L'environnement de l'agent n'est donc plus parfaitement disponible. Seule une **observation** de l'état sous-jacent est accessible. Celle-ci n'est pas suffisante pour prendre une décision satisfaisante. En effet, deux observations identiques peuvent résulter d'un état sous-jacent différent. C'est le cas d'une réponse à une question oui/non. L'observation est la même mais l'état sous-jacent est lié à la question ayant amené cette réponse. En gardant une trace de l'historique, c'est-à-dire de l'ensemble des observations et des actions qui ont été échangées depuis le début de l'interaction, il est possible d'identifier de façon unique chaque observation courante rencontrée.

Deux façons de prendre en charge l'observabilité partielle existent. La première, la plus classique, consiste à placer le problème dans le cadre des processus décisionnels de Markov partiellement observables (*Partially Observable Markov Decision Process*, POMDP en anglais) [Astrom 65].

**La résolution avec un POMDP** La définition formelle d'un POMDP est la suivante :

**Définition 3.2.** *Un POMDP est un n-uplet* $\{S, A, T, R, \gamma, O, \Omega\}$ *où :*

- $\{S, A, T, R, \gamma\}$ *est un MDP*
- $O$ *est l'ensemble des observations possibles*
- $\Omega$ *est un jeu de probabilités d'observations conditionnelles tel que* $\Omega = \{p(o|s', a) \forall (o, s', a) \in O \times S \times A\}$
- $b_0(s)$ *pour tout* $s \in S$ *une distribution initiale pour les états de croyance.*

Les états sous-jacents respectent ainsi la propriété de Markov, mais pas les observations. L'approche classique pour garder une trace compacte de l'historique des observations et actions passées, permettant de savoir si deux observations identiques résultent du même état sous-jacent ou non, revient à définir un état de croyance (*Belief State*, BF en anglais) qui maintient une distribution de probabilités sur tous les états sous-jacents possibles. Le définition de l'état de croyance associée, $b_t$, c'est-à-dire la distribution de probabilités sur les états possibles $s_{t+1}$, étant donné qu'à l'instant $t$, l'état est $s_t$, l'observation $o_t$ et l'action $a_t$, est la suivante :

$$b_t(s_{t+1}) = \frac{p(o_t|s_{t+1}, a_t) \sum_{s_t \in S} p(s_{t+1}|s_t, a_t) b(s_t)}{\sum_{s_{t+1} \in S} p(o_t|s_{t+1}, a_t) \sum_{s \in S} p(s_{t+1}|s_t, a_t) b(s)}. \tag{3.1}$$

En utilisant cet état de croyance, un nouvel MDP continu peut être défini et résolu de façon exacte en montrant que la fonction de valeur sur ce nouvel MDP est linéaire par morceaux [Sondik 71]. Des exemples de résolutions exactes sont donnés [Cheng 88, Kaelbling 98]. Cependant cela requière que le nombre d'états soit petit et limité sous peine de voir la solution du problème non calculable. La question du passage à l'échelle a été abordée dans [Littman 95]. Il faut alors envisager de résoudre le POMDP de façon approchée [Parr 95, Pineau 03]. D'autre part, cette approche nécessite la connaissance des jeux de probabilités $T$ et $\Omega$. Dans le cas de problèmes réels, cette containte est rarement respectée. Une façon de la gérer est d'estimer le modèle des observations en construisant une base de données de tous les historiques possibles.

Cela devient impossible si le nombre d'états est trop grand. C'est pourquoi une autre approche est considérée pour les problèmes réels. Elle est présentée au paragraphe suivant.

**Construction de l'espace d'états** $S$ La seconde approche consiste à utiliser directement les informations échangées. Cependant, si toutes sont gardées en mémoire dès le début de l'interaction car cet historique désambiguïse toutes les situations, dans le cas d'échanges longs, la solution risque de ne plus pouvoir être calculable. Une représentation compacte de l'historique serait alors intéressante. La façon la plus simple de procéder consiste à ne retenir qu'un nombre d'intéractions fixe antérieur à l'observation courante. Dès qu'une nouvelle observation apparaît, celle la plus ancienne retenue au pas de temps précédent est effacée. C'est le principe d'une fenêtre glissante sur les observations. Mais dans cette approche, définir la taille de la fenêtre reste difficile et n'offre pas de garanties [McCallum 95]. Il faut en effet trouver un compromis entre la taille de la fenêtre et la quantité d'informations retenue. De plus, les informations redondantes sont mémorisées. Utiliser une méthode construisant automatiquement un historique compact serait alors intéressant [Littman 02, Bakker 02, Szita 06]. Un état de l'art plus détaillé est proposé à la Section 9.1.

Dans le cas du dialogue parlé, une bonne estimation de l'état sous-jacent du dialogue peut être fournie, moyennnant un rigoureux travail d'ingénierie car les analyseurs de parole et de sens fournissent plusieurs solutions potentielles auxquelles sont associées une information de certitude. Ces solutions sont basées sur l'analyse physique du signal (la voix) émis par l'utilisateur. Un exemple sera décrit Section 5.1. Dans le cas du tutorat, ces états sont beaucoup plus difficiles à estimer, à moins de construire un modèle de l'utilisateur. Dans ce cas, non seulement un biais est introduit, mais il est aussi beaucoup plus difficile d'inférer des informations précises sans connaître des données physiques sur le processus sous-jacent ayant généré les observations. Une approche sans modèle d'utilisateur est proposée à la Section 9.1.

### 3.1.2 Outils

Une fois le cadre de travail défini, ainsi que ses limites, une présentation des méthodes de résolution classiques du problème de RL est proposée. Elles sont très génériques et s'appliquent dans un cadre beaucoup plus général que celui des interactions homme-machine. Cette section propose quelques définitions de notions de base utilisées très classiquement.

#### Fonction de valeur – fonction de qualité

La qualité d'une politique est quantifiée au moyen d'une fonction appelée *fonction de valeur* $V$. Elle est définie de $S$ dans $\mathbb{R}$. Elle associe à chaque état $s \in S$ l'espérance de la somme des récompenses pondérées qui peuvent être obtenues en partant de $s$ et en suivant la politique $\pi$ :

$$V^{\pi}(s) = E\left[\sum_{i=0}^{\infty} \gamma^{i} r_i | s_0 = s, a_i = \pi(s_i)\right]. \tag{3.2}$$

La politique optimale $\pi^*$ est celle qui donne l'espérance maximale, c'est-à-dire la valeur de $V^{\pi}$ maximale pour tout $s \in S$ : $\pi^* \in \mathrm{argmax}_{\pi} V^{\pi}$. La fonction V associée à cette politique optimale est $V^*$.

En lien avec la fonction de valeur, une fonction de qualité peut être définie, notée $Q$, qui lie à chaque paire état-action $(s, a)$ une valeur réelle, $Q : S \times A \rightarrow \mathbb{R}$. Associée à la politique

$\pi$, elle est définie telle que :

$$Q^{\pi}(s,a) = E\left[\sum_{i=0}^{\infty} \gamma^{i} r_i | s_0 = s, a_0 = a, a_{i>0} = \pi(s_i)\right]. \tag{3.3}$$

Cette fonction autorise un degré de liberté supplémentaire dans le choix de l'action dans l'état $0$. Avec la fonction V, il n'est possible d'évaluer que l'action associée à la politique courante $V^{\pi}(s) = Q^{\pi}(s, \pi(s))$. En revanche, la fonction de qualité permet d'évaluer une action qui n'est pas celle de la politique courante $Q(s, a \neq \pi(s))$. La fonction V ne permet donc que d'évaluer la politique alors que la fonction Q permet éventuellement de l'améliorer en prenant la meilleure action suivante. Cette caractéristique est utile lorsque le modèle de la dynamique n'est pas connu. Lorsqu'il est choisi de suivre systématiquement la meilleure action, cette démarche est appelée choix de l'action gloutonne (*greedy* en anglais). En pratique, la fonction de qualité est donc la plus utilisée.

### Les équations de Bellman

La plupart des solutions pour résoudre un MDP utilisent les équations d'*évaluation* et d'*optimalité* de Bellman [Bellman 57a]. Pour les définir et les utiliser, il est impératif que la dynamique du système décrite soit markovienne. Ces équations sont les suivantes pour la fonction V :

| Equation d'évaluation | Equation d'optimalité |
|---|---|
| $V^{\pi}(s) = E_{s'\|s}\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$ | $V^{*}(s) = \max_a E_{s'\|s,a}\left[R(s,a,s') + \gamma V^{*}(s')\right]$ |

A partir de ces équations, deux opérateurs fonctionnels sont définis. Le premier est l'opérateur d'évaluation de Bellman $T_V^{\pi}$ associé à une politique $\pi$, avec $T_V^{\pi}$ de $S \to \mathbb{R}$ dans $S \to \mathbb{R}$, tel que :

$$T_V^{\pi} V(s) = E_{s'|s,\pi}\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]. \tag{3.4}$$

Le second est l'opérateur d'optimalité de Bellman $T_V^{*}$ défini tel que :

$$T_V^{*} V(s) = \max_a E_{s'|s,a}\left[R(s,a,s') + \gamma V(s')\right]. \tag{3.5}$$

Une non-linéarité est introduite dans l'équation d'optimalité par l'opérateur $\max$ ce qui complique la résolution de l'équation. Ces deux opérateurs sont des contractions. D'après le théorème du point fixe de Banach [Banach 22], ils admettent chacun un unique point fixe, respectivement $V^{\pi}$ et $V^{*}$. Ce dernier représente la fonction de valeur optimale.

En pratique, il est préférable d'utiliser la fonction de qualité pour estimer la politique optimale. De façon similaire, les équations de Bellman peuvent être définies pour cette fonction :

| Equation d'évaluation | Equation d'optimalité |
|---|---|
| $Q^{\pi}(s,a) = E_{s'\|s,a}\left[R(s,a,s') + \gamma Q^{\pi}(s',a)\right]$ | $Q^{*}(s,a) = E_{s'\|s,a}\left[R(s,a,s') + \gamma \max_{b\in A} Q^{*}(s',b)\right]$ |

De plus, de la même façon que pour la fonction de valeur, deux opérateurs fonctionnels de Bellman sont introduits. Ils sont définis de $S \times A \to \mathbb{R}$ dans $S \times A \to \mathbb{R}$. L'opérateur d'évaluation $T_Q^\pi$ associé à la politique $\pi$ est défini tel que :

$$T_Q^\pi Q(s,a) = E_{s'|s,\pi}\left[R(s,a,s') + \gamma Q^\pi(s',a)\right], \tag{3.6}$$

et celui d'optimalité $T_Q^*$ est défini tel que :

$$T_Q^* Q(s,a) = E_{s'|s,\pi}\left[R(s,a,s') + \gamma \max_{b \in A} Q(s',b)\right]. \tag{3.7}$$

Ces deux opérateurs sont contractants de points fixes respectifs $Q^\pi$ et $Q^*$.

### La programmation dynamique

Lorsque le modèle du MDP est connu, c'est-à-dire lorsque la dynamique du système est disponible au travers des probabilités de transition ainsi que de la fonction de récompense et que l'espace d'états est fini et petit, le problème d'optimisation peut être résolu de façon exacte. Deux algorithmes principaux existent pour ce faire, l'*itération de la politique* et l'*itération de la valeur*.

**Itération de la politique.** L'algorithme d'itération de la politique est basé sur une alternance de deux phases : la politique est tout d'abord évaluée en calculant sa fonction de valeur, puis améliorée en choisissant l'action gloutonne de la fonction de valeur qui vient d'être évaluée. Il a été montré que la politique à l'itération suivante est améliorée [Puterman 94]. Ce cycle est répété jusqu'à qu'à ce qu'il n'y ait plus d'amélioration. Le pseudo-code est donné Algorithme 3.1.

**Algorithme 3.1** Itération de la politique

Initialisation : $k \leftarrow 0$, $\pi_k \leftarrow \pi_0$
**répéter**
  Evaluation : $V_k = T_V^{\pi_k} V_k$
  **pour chaque** $s \in S$ **faire**
    Amélioration :

$$\begin{aligned}\pi_{k+1}(s) &= \operatorname*{argmax}_{a \in A} \sum_{s' \in S} p(s'|s,a)(R(s,a,s') + \gamma V_k(s')) \\ &= \operatorname*{argmax}_{a \in A} Q_{\pi_k}(s,a)\end{aligned}$$

  **fin**
  $k \leftarrow k+1$
**jusqu'à** $\pi_{k+1} = \pi_k$;

**Itération de la valeur.** Cet algorithme se base sur le théorème du point fixe de Banach pour trouver la politique optimale. Comme la suite $V_{k+1} = T^* V_k$ converge vers la fonction de valeur optimale, il est ensuite possible de déduire la politique optimale. Lorsque l'itération suivante n'apporte plus d'amélioration significative, les itérations sont stoppées. Le pseudo-code de l'algorithme est donné Algorithme 3.2.

**Algorithme 3.2** Itération de la valeur

Initialisation : $k \leftarrow 0$, $V_k \leftarrow V_0$, $\epsilon$ constante positive
**répéter**
  $V_{k+1} = T_V^* V_k$
  $k \leftarrow k + 1$
**jusqu'à** $|V_{k+1} - V_k| < \epsilon$;
**retourner** $\pi^*$

### 3.1.3 Méthodes de résolution de référence

La programmation dynamique est une méthode efficace pour trouver la politique optimale liée à un MDP. Cependant, elle nécessite la connaissance du modèle de transitions et de la fonction de récompense. Pour des problèmes réels, ces conditions sont rarement rencontrées. Une solution consiste alors à estimer le modèle des transitions par une méthode de Monte-Carlo. Cela n'est envisageable que si la dynamique du système peut être simulée. En effet, par cette méthode, le système interagit un grand nombre de fois avec son environnement afin d'estimer les probabilités de transitions.

Dès que l'estimation est suffisamment précise, un algorithme de programmation dynamique est appliqué au modèle estimé. Seulement, ces algorithmes restent gourmands en échantillons et donc peu efficaces. Le RL est une méthode qui s'affranchit de l'apprentissage du modèle et propose une solution approchée au problème de contrôle optimal. Une compilation de différents résultats à propos du RL est proposée dans [Kaelbling 96, Sutton 98].

#### Les différences temporelles

Les algorithmes aux différences temporelles (*Temporal Differences*, TD en anglais) sont très utilisés en apprentissage par renforcement. Le terme de différence temporelle a été utilisé une des premières fois dans [Sutton 88].

Ils se basent sur les équations de Bellman, dont il est possible de déduire une équation de type Widrow-Hoff pour proposer une estimation de la fonction de valeur associée à une politique $\pi$. L'estimation est faite itérativement. Elle est corrigée à chaque interaction entre l'agent et son environnement, grâce à la récompense obtenue lors de cette transition et grâce à la prédiction de cette même récompense calculée à l'aide de l'estimation courante de la fonction de valeur. La mise à jour de l'estimation est asynchrone car seule la valeur de la fonction associée à l'état rencontré lors de l'interaction courante est changée. Dans cette section, l'espace d'état est considéré comme fini. Le cas plus complexe où il est continu est considéré Section 3.2. Les différentes valeurs de la fonction de valeur sont ainsi stockées dans une table de correspondances (*lookup table* en anglais). Seule celle correspondant à l'état visité est mise à jour.

Comme la seule information fiable pour évaluer l'estimation de la fonction de valeur est la récompense, celle obtenue lors d'une transition $r$ et celle prédite $\hat{r}$ sont utilisées pour corriger l'estimation, $\hat{V}$. Etant donnée une transition $(s_t, a_t, s_{t+1})$, il est possible de réécrire l'équation d'évaluation de Bellman pour cette transition. Les équations suivantes sont alors obtenues :

$$V^\pi(s_t) \approx r_t + \gamma V^\pi(s_{t+1}) \tag{3.8}$$

$$\text{soit } r_t \approx V^\pi(s_t) - \gamma V^\pi(s_{t+1}) \tag{3.9}$$

Ainsi, un estimateur de la récompense est fourni : $\hat{r}_t = \hat{V}^\pi(s_t) - \gamma \hat{V}^\pi(s_{t+1})$. Si la notation $\delta$ est choisie pour noter l'erreur d'estimation entre la récompense observée et celle prédite, alors

$\delta_t = r_t - \hat{r}_t$ soit $\delta_t = r_t + \gamma \hat{V}^\pi(s_{t+1}) - \hat{V}^\pi(s_t)$. L'estimation de la fonction de valeur est alors corrigée en conséquence :

$$\hat{V}^\pi(s_t) \leftarrow \hat{V}^\pi(s_t) + \alpha_t \delta_t, \tag{3.10}$$

avec $\alpha_t$ un taux d'apprentissage qui permet de jouer sur l'importance de la nouvelle information. Le pseudo-code de l'algorithme TD est donné Algorithme 3.3.

**Algorithme 3.3** TD

Initialisation : $\hat{V}$ ;
**pour** *t = 1,2,3, ...* **faire**
  Observer $(s_t, s_{t+1}, r_t)$

$$\delta_t = r_t + \gamma \hat{V}^\pi(s_{t+1}) - \hat{V}^\pi(s_t)$$

$$\hat{V}^\pi(s) \leftarrow \begin{cases} \hat{V}^\pi(s) + \alpha_t \delta_t \text{ si } s = s_t \\ \hat{V}^\pi(s) \text{ sinon} \end{cases}$$

**fin**

Cependant, cet algorithme ne peut pas être utilisé seul pour calculer la politique optimale. En effet, il propose seulement une estimation de la fonction de valeur. Il faut ensuite *améliorer* la politique pour apprendre à contrôler le système dynamique. C'est pourquoi au lieu d'utiliser la fonction de valeur, il lui est préféré la fonction de qualité. Celle-ci permet un choix sur l'action suivante, notamment celui glouton qui améliore la politique. Les algorithmes qui l'utilisent peuvent réaliser non seulement une évaluation de la politique à travers la fonction Q, mais aussi une amélioration. Les deux algorithmes présentés dans ce qui suit sont deux algorithmes classiques respectivement on- et off-policy permettant de trouver la politique optimale avec une approche TD.

### SARSA

L'algorithme SARSA est nommé d'après les initiales de la transition $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$. Il utilise l'équation d'évaluation de Bellman appliquée à la fonction de qualité et le principe des différences temporelles dans un schéma d'itération de la politique asynchrone et approximatif, auquel est ajouté un mécanisme d'exploration. La partie estimation de la fonction de valeur est assurée par l'algorithme TD. La partie contrôle, qui fixe le choix des actions, n'est pas entièrement détaillée dans cette section. Plus de précisions dans le choix des actions sont données Section 3.3.2. Seul le choix des actions $\epsilon$-glouton ($\epsilon$-*greedy* en anglais) est présenté ici. C'est une solution classique pour gérer le dilemme exploration-exploitation dans l'apprentissage de la solution optimale.

L'agent doit choisir l'action suivante tout en apprenant la solution optimale. Pour ce faire, il peut soit utiliser l'information déjà connue à propos de son environnement, soit aller découvrir des zones non-explorées. L'agent ne se contente donc pas d'observer des transitions, il est acteur dans la génération de ces dernières. Le choix de l'action est le suivant, étant donné un état $s_t$ :

$$a_t = \begin{cases} \text{argmax}_{a \in A} \hat{Q}^\pi(s_t, a) \text{ avec la probabilité } 1 - \epsilon \\ \text{aléatoire dans A avec la probabilié } \epsilon, \end{cases} \tag{3.11}$$

avec $\epsilon \in ]0, 1]$. Le pseudo-code de cet algorithme est donné Algorithme 3.4. SARSA est un algorithme on-policy car la politique utilisée pour générer les transitions (la politique de contrôle) est celle améliorée. Il est à noter que seule l'estimation de la valeur de la fonction associée à la paire état-action visitée à chaque pas de temps est corrigée. Le cas off-policy est présenté dans la section suivante.

**Algorithme 3.4** SARSA

Initialisation : $\hat{Q}$, $s_t \leftarrow s_0$, $a_t \leftarrow a_0$
**pour** $t = 0, 1, 2, \ldots$ **faire**
  Observer $r_t, s_{t+1}$
  Générer $a_{t+1}$ selon la politique de contrôle choisie – Appliquer $a_{t+1}$
  Mettre à jour l'estimation de la fonction Q :

$$\delta_t = r_t + \gamma \hat{Q}(s_{t+1}, a_{t+1}) - \hat{Q}(s_t, a_t)$$

$$\hat{Q}(s, a) \leftarrow \begin{cases} \hat{Q}(s, a) + \alpha_t \delta_t \text{ si } s = s_t \text{ et } a = a_t \\ \hat{Q}(s, a) \text{ sinon} \end{cases}$$

  Préparer l'itération suivante : $s_t \leftarrow s_{t+1}$, $a_t \leftarrow a_{t+1}$
**fin**

#### Q-learning

L'algorithme Q-*learning* est basé sur l'équation d'optimalité de Bellman appliquée à la fonction de qualité dans un schéma d'itération de la valeur approchée et asynchrone. Il a été introduit dans [Watkins 89]. La fonction Q optimale est directement estimée. La politique optimale (qui est déduite de la fonction Q optimale) est différente de celle utilisée pour générer les transitions. Le choix de la politique de contrôle est donc libre. C'est elle qui est en charge de l'exploration de l'espace d'état et l'espace d'action. Le dilemme exploration/exploitation ne se gère donc pas de la même façon que dans le cas on-policy. Dans le cas extrême où les actions sont choisies aléatoirement, l'apprentissage est efficace car les espaces $S$ et $A$ sont explorés partout. De plus, au lieu d'utiliser une transition du type $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$, une transition $(s_t, a_t, s_{t+1}, r_t)$ est générée, le choix de la nouvelle action étant laissé à la politique de contrôle. Le pseudo-code du Q-learning est donné Algorithme 3.5.

## 3.2 Approximation de la fonction de valeur

Dans de nombreux problèmes réels, l'espace d'état n'est pas de taille finie. Dans ce cas, il faut envisager une représentation paramétrique de la fonction de valeur (ou de qualité). Les représentations non-paramétriques ne seront pas abordées en détail dans cette section. Nous soulignons le fait que l'approche *Gaussian Process Temporal Differences* (GPTD) [Engel 03], présentée à la Section 3.2.2 utilise une représentation non-paramétrique de la fonction de valeur. Un dictionnaire de points est construit au cours de l'apprentissage.

Lorsque la fonction de valeur est approximée, le but n'est plus alors d'apprendre la valeur associée à chaque couple état/action, mais d'apprendre un jeu de paramètres permettant, pour tout couple de l'espace d'état/action, de trouver la valeur de la fonction Q associée. Ceci peut aussi s'appliquer au cas où l'espace d'état est fini mais de trop grand cardinal pour qu'une implémentation efficace puisse être proposée. Il est seulement considéré ici le cas où l'espace

**Algorithme 3.5** Q-learning

Initialisation : $\hat{Q}^*$, $s_t \leftarrow s_0$
**pour** $t = 0, 1, 2, \ldots$ **faire**
Générer $a_t$ selon la politique de contrôle choisie – Appliquer $a_t$
Observer $r_t, s_{t+1}$
Mettre à jour l'estimation de la fonction Q :

$$\delta_t = r_t + \gamma \max_{a \in A} \hat{Q}^*(s_{t+1}, a) - \hat{Q}^*(s_t, a_t)$$

$$\hat{Q}^*(s, a) \leftarrow \begin{cases} \hat{Q}^*(s, a) + \alpha_t \delta_t \text{ si } s = s_t \text{ et } a = a_t \\ \hat{Q}^*(s, a) \text{ sinon} \end{cases}$$

Préparer l'itération suivante : $s_t \leftarrow s_{t+1}$
**fin**

d'état est continu, le nombre d'actions restant raisonnable. La présentation des approximations envisagées pour la fonction de qualité est suivie de l'application de ces dernières aux algorithmes de RL les plus classiques dans cette section.

### 3.2.1 Paramétrisations

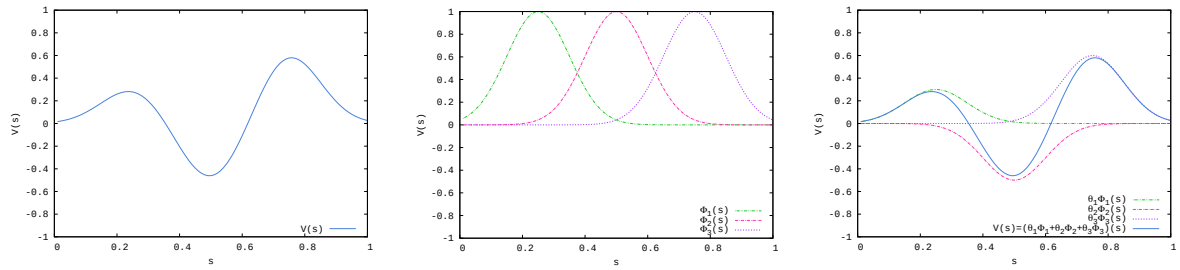
Les représentations paramétriques permettent de calculer les valeurs d'une fonction en tout point de l'espace sur lequel elle est définie. Le but est, à partir d'échantillons de la fonction, de trouver le jeu de paramètres permettant de la représenter au mieux dans l'espace d'hypothèses.

Le premier type de représentation est une paramétrisation linéaire. Elle présente entre autres les avantages d'être dérivable et facilement manipulée. Si le concepteur reste dans l'espace dans lequel la fonction est définie, il ne pourra l'approcher que par des hyperplans ce qui ne fournira en général que de mauvaises approximations. L'idée est alors d'utiliser une estimation linéaire de la fonction non plus dans l'espace initial mais dans un espace engendré par un ensemble de fonctions, appelé fonctions de base. Par exemple, pour la fonction de qualité, une représentation linéaire s'écrit sous la forme :

$$\hat{Q}_\theta(s, a) = \theta^T \Phi(s, a), \tag{3.12}$$

pour les $(s, a) \in S \times A$, avec $\theta \in \mathbb{R}^p$ le vecteur de paramètres et $p$ leur nombre, et $\Phi$ l'ensemble des fonctions de base. Les approximateurs utilisant des fonctions de base radiales (*Radial Basis Functions*, RBF en anglais) sont connus pour proposer des estimations de qualité [Powell 87]. Ce sont des fonctions dont la valeur dépend uniquement de la distance à un autre point (appelé centre $c \in S \times A$). Les fonctions gaussiennes en N dimensions sont un exemple souvent utilisé. Cette paramétrisation est illustrée Figure 3.3. L'espace d'état est l'intervalle $[0, 1]$. Il est pavé par trois gaussiennes équi-réparties. En adaptant les poids des fonctions de base, il est possible de retrouver la fonction de valeur.

Cependant, les paramétrisations linéaires souffrent de la malédiction de la dimensionalité. Dès que la taille initiale de l'espace augmente, le nombre de paramètres devient très important. Cela a pour conséquences de nécessiter beaucoup plus de données d'entraînement. C'est pourquoi des paramétrisations non-linéaires sont utilisées. Pour un nombre moindre de paramètres, la fonction de valeur peut représenter une quantité équivalente d'information. Un exemple précis sera décrit Section 6.1.2 pour illustrer ce propos. Le réseau de neurones en est un exemple parmi




a) Fonction de valeur à estimer b) Fonctions de base c) Adaptation de θ

**Figure 3.3 –** Illustration d'une paramétrisation linéaire de la fonction de valeur basée sur un réseau de gaussiennes avec $s \in [0, 1]$ et $N = 3$.

les plus utilisés. Les poids du réseau constituent alors le vecteur de paramètres $\theta$ et donc :

$$\hat{Q}_\theta(s, a) = f_\theta(s, a), \tag{3.13}$$

avec $f$ une fonction non-linéaire sur les paramètres. L'espace engendré par toutes les paramétrisations possibles de la fonction de qualité s'appelle l'espace d'hypothèses, $\mathcal{H}$ avec $\mathcal{H} = \left\{ \hat{Q}_\theta | \theta \in \mathbb{R}^p \right\}$.

En adoptant donc une représentation approchée de la fonction de valeur, il est alors possible de gérer des problèmes réels qui impliquent la gestion de grands espaces d'état. Dans le cadre du dialogue, ce problème est rencontré lorsque sont prises en compte les probabilités issues de la reconnaissance de parole et de l'analyseur sémantique. Dans le cadre du tutorat, c'est aussi le cas si un pourcentage est associé à la maîtrise d'un savoir et non juste une valeur binaire.

### 3.2.2 Application aux algorithmes classiques

Le but est de trouver une estimation de la fonction qui soit valable sur tout l'espace où elle est définie à partir de points d'échantillonnage. Cette approche peut se voir comme un problème d'optimisation dans lequel le coût dépendrait de la vraie fonction et de la fonction estimée, identifiée par son vecteur de paramètres. Dans le cas du RL, le problème est complexifié par le fait que les informations sur la vraie fonction ne sont disponibles qu'au travers de la récompense. Une description complète des différents algorithmes utilisant une approximation linéaire de la fonction de valeur est proposée dans [Geist 13]. Trois des plus classiques sont présentés dans ce qui suit. Une quatrième approche, moins classique, complète les descriptions. Elles seront utilisées par la suite comme point de comparaison aux méthodes que nous proposons d'utiliser dans ces travaux.

#### SARSA

L'optimisation la plus simple consiste à essayer de minimiser l'erreur entre la vraie fonction de qualité liée à une politique $\pi$ et son estimée. Le critère $J$ à minimiser est alors le suivant :

$$J(\theta) = \|Q^\pi - \hat{Q}_\theta\|^2. \tag{3.14}$$

Cependant, la vraie fonction $Q^\pi$ n'est pas connue. Les informations à son propos sont donc remplacées par une séquence de $j$ observations $q_j^\pi$. Cela donne une approximation du critère

empirique à minimiser :

$$\hat{J}_{Q^\pi}(\theta) = \sum_j \left(q_j^\pi - \hat{Q}_\theta(s_j, a_j)\right)^2. \tag{3.15}$$

Le vecteur de paramètres optimal s'obtient par une descente de gradient stochastique, ce qui correspond à l'algorithme SARSA. Ils sont mis à jour de la façon suivante, à l'instant $t < j$ :

$$\theta_t = \theta_{t-1} - \frac{\alpha_t}{2}\nabla_{\theta_{t-1}}\left(q_t^\pi - \hat{Q}_\theta(s_t, a_t)\right)^2 \tag{3.16}$$

$$= \theta_{t-1} + \alpha_t\left(\nabla_{\theta_{t-1}}\hat{Q}_\theta(s_t, a_t)\right)\left(q_t^\pi - \hat{Q}_{\theta_{t-1}}(s_t, a_t)\right). \tag{3.17}$$

Etant donné que $q_t^\pi$ n'est pas directement observé, cette information est remplacée par une estimation obtenue grâce à l'équation de Bellman échantillonnée appliquée à l'estimation courante $\hat{Q}_{\theta_{t-1}}(s_t, a_t)$. L'équation échantillonnée remplace l'espérance présente dans la définition de l'équation par un jeu d'exemples. Les paramètres sont donc mis à jour de la façon suivante, après observation de la transition $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ :

$$\theta_t = \theta_{t-1} + \alpha_t\left(\nabla_{\theta_{t-1}}\hat{Q}_\theta(s_t, a_t)\right)\left(\hat{T}_Q^\pi\hat{Q}_{\theta_{t-1}} - \hat{Q}_{\theta_{t-1}}(s_t, a_t)\right) \tag{3.18}$$

$$= \theta_{t-1} + \alpha_t\left(\nabla_{\theta_{t-1}}\hat{Q}_\theta(s_t, a_t)\right)\left(r_t + \gamma\hat{Q}_{\theta_{t-1}}(s_{t+1}, a_{t+1}) - \hat{Q}_{\theta_{t-1}}(s_t, a_t)\right) \tag{3.19}$$

Le pseudo-code de l'algorithme SARSA avec une approximation de la fonction de qualité est présenté Algorithme 3.6. C'est une généralisation de celui présenté Algorithme 3.4 car le cas tabulaire est un cas particulier d'approximation.

**Algorithme 3.6** SARSA avec approximation de la fonction de valeur

Initialisation : $\theta_t \leftarrow \theta_0$, $s_t \leftarrow s_1$, $a_t \leftarrow a_1$
**pour** $t = 1, 2, 3, \ldots$ **faire**
  Observer $r_t, s_{t+1}$
  Générer $a_{t+1}$ selon la politique de contrôle choisie – Appliquer $a_{t+1}$
  Mettre à jour l'estimation des paramètres :
  $\theta_t = \theta_{t-1} + \alpha_t\left(\nabla_{\theta_{t-1}}\hat{Q}_\theta(s_t, a_t)\right)\left(r_t + \gamma\hat{Q}_{\theta_{t-1}}(s_{t+1}, a_{t+1}) - \hat{Q}_{\theta_{t-1}}(s_t, a_t)\right)$
  Préparer l'itération suivante : $s_t \leftarrow s_{t+1}$, $a_t \leftarrow a_{t+1}$
**fin**

### Q-learning

Le principe du Q-learning avec approximation de la fonction de valeur est le même que celui de SARSA avec approximation sauf que le critère d'optimisation concerne la fonction de qualité optimale. Dans ce cas, il est défini de la façon suivante :

$$J_{Q^*}(\theta) = \|Q^* - \hat{Q}_\theta\|^2. \tag{3.20}$$

Un raisonnement similaire à celui présenté précédemment pour SARSA dans le paragraphe précédent peut se faire dans le cas du Q-learning. Etant donnée une séquence de $j$ observations

$q_j^*$ observations de la fonction $Q^*$, il est possible de proposer une estimation du critère à optimiser :

$$\hat{J}_{Q^*}(\theta) = \sum_j \left( q_j^* - \hat{Q}_\theta(s_j, a_j) \right)^2 . \tag{3.21}$$

De la même façon que pour SARSA, les paramètres $\theta$ sont estimés en effectuant une descente de gradients stochastique :

$$\theta_t = \theta_{t-1} - \frac{\alpha_t}{2} \nabla_{\theta_{t-1}} \left( q_t^* - \hat{Q}_\theta(s_t, a_t) \right)^2 \tag{3.22}$$

$$= \theta_{t-1} + \alpha_t \left( \nabla_{\theta_{t-1}} \hat{Q}_\theta(s_t, a_t) \right) \left( q_t^* - \hat{Q}_{\theta_{t-1}}(s_t, a_t) \right) . \tag{3.23}$$

Comme les observations $q_j^*$ ne sont pas disponibles, l'équation d'optimalité de Bellman échantillonnée est utilisée pour approcher leur valeur :

$$\theta_t = \theta_{t-1} + \alpha_t \left( \nabla_{\theta_{t-1}} \hat{Q}_\theta(s_t, a_t) \right) \left( \hat{T}_Q^* \hat{Q}_{\theta_{t-1}} - \hat{Q}_{\theta_{t-1}}(s_t, a_t) \right) \tag{3.24}$$

$$= \theta_{t-1} + \alpha_t \left( \nabla_{\theta_{t-1}} \hat{Q}_\theta(s_t, a_t) \right) \left( r_t + \gamma \max_{a \in A} \hat{Q}_{\theta_{t-1}}(s_{t+1}, a) - \hat{Q}_{\theta_{t-1}}(s_t, a_t) \right) \tag{3.25}$$

Le pseudo-code de Q-learning utilisant une fonction de valeur approchée est donné Algorithme 3.7.

**Algorithme 3.7** Q-learning avec approximation de la fonction de valeur

Initialisation : $\theta_t \leftarrow \theta_0$, $s_t \leftarrow s_1$
**pour** $t = 1, 2, 3, \ldots$ **faire**
  Générer $a_t$ selon la politique de contrôle choisie – Appliquer $a_t$
  Observer $r_t, s_{t+1}$
  Mettre à jour l'estimation de la fonction Q :
  $\theta_t = \theta_{t-1} + \alpha_t \left( \nabla_{\theta_{t-1}} \hat{Q}_\theta(s_t, a_t) \right) \left( r_t + \gamma \max_{a \in A} \hat{Q}_{\theta_{t-1}}(s_{t+1}, a) - \hat{Q}_{\theta_{t-1}}(s_t, a_t) \right)$
  Préparer l'itération suivante : $s_t \leftarrow s_{t+1}$
**fin**

### *Least Square Policy Iteration* (LSPI)

LSPI est un algorithme d'itération de la politique approchée (Algorithme 3.1) dans lequel l'évaluation de la fonction de qualité est faite en utilisant l'algorithme *Least Square Temporal Differences* (LSTD) [Bradtke 96]. C'est un algorithme de résolution classique avec approximation de la fonction de valeur [Lagoudakis 03]. Il se déroule en deux phases répétées itérativement jusqu'à ce qu'un seuil soit atteint. La première vise à évaluer la politique courante en minimisant la distance entre la fonction estimée et son image au travers de l'opérateur de Bellman projetée sur l'espace d'hypothèses $\mathcal{H}$. Ce principe est illustré Figure 3.4. La seconde vise à améliorer la politique.

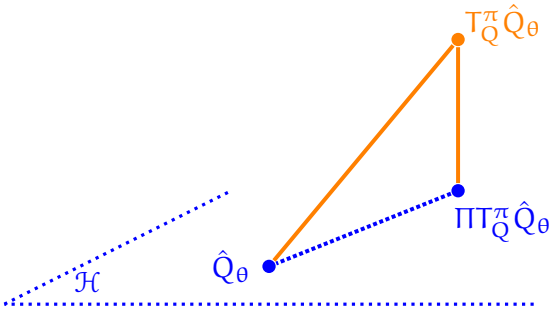



**Figure 3.4** – Illustration du principe de l'algorithme LSPI.

Durant la phase d'évaluation, étant donnée une politique $\pi$, le critère J à optimiser est le suivant :

$$J(\theta) = \|\hat{Q}_\theta - \Pi T_Q^\pi \hat{Q}_\theta\|^2 \text{ avec } \Pi f = \underset{\hat{f}\in\mathcal{H}}{\operatorname{argmin}} \|f - \hat{f}\|^2. \tag{3.26}$$

L'évaluation de la politique $\pi_k$ à l'itération $k$ à partir de N transitions se fait en utilisant sa fonction de qualité associée $\hat{Q}_{\theta_k}$, à l'aide de l'algorithme *Least Square Temporal Differences* (LSTD en anglais). Le détail de l'évaluation est le suivant, à l'itération $k$ :

$$\theta_k = \underset{\omega\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{N} \left(\hat{Q}_\omega(s_i, a_i) - \hat{T}_Q^\pi \hat{Q}_{\theta_k}(s_i, a_i)\right)^2. \tag{3.27}$$

Une représentation de la fonction de qualité linéaire est utilisée. En remplaçant l'estimée de Q dans l'Equation 3.27 par sa paramétrisation linéaire $\hat{Q}(s,a) = \theta^T \Phi(s,a)$ donnée à l'Equation 3.12, en utilisant l'équation d'évaluation de Bellman échantillonnée et la politique $\pi_k$ associée à la fonction de qualité, l'expression suivante du vecteur de paramètres est obtenue :

$$\theta_k = \underset{\omega\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{N} \left(r_i + \gamma\theta_k^T \Phi(s_{i+1}, \pi_k(s_{i+1})) - \omega^T \Phi(s_i, a_i)\right)^2. \tag{3.28}$$

En utilisant l'hypothèse de la linéarité par rapport aux paramètres, une expression analytique du vecteur peut être exprimée :

$$\theta_k = \left(\sum_{i=1}^{N} \Phi(s_i, a_i)(\Phi(s_i, a_i) - \gamma\Phi(s_{i+1}, \pi_k(s_{i+1}))^T\right)^{-1} \sum_{i=1}^{N} \Phi(s_i, a_i) r_i. \tag{3.29}$$

La seconde phase se déroule avec l'amélioration de la politique en prenant l'action gloutonne par rapport à l'estimation courante de la fonction de qualité :

$$\pi_{k+1}(s) = \underset{a\in A}{\operatorname{argmax}}\, \hat{Q}_{\theta_k}(s, a). \tag{3.30}$$

Ces deux phases sont répétées jusqu'à ce qu'il n'y ait plus de changements notables entre deux mises à jour du vecteur $\theta$ ou qu'un nombre d'itérations soit atteint. Le pseudo-code de l'algorithme LSPI est présenté Algorithme 3.8. Le jeu de données d'entraînement est fourni au départ.

**Algorithme 3.8** LSPI

Données : un jeu de N transitions
Initialisation : $k \leftarrow 0$, $\theta_k \leftarrow \theta_0$, $\epsilon$ constante positive
**répéter**
  Evaluation des paramètres $\theta_k$ avec LSTD
  Amélioration de la politique $\pi_{k+1}(s) = \text{argmax}_{a \in A} \hat{Q}_{\theta_k}(s, a)$
  $k \leftarrow k + 1$
**jusqu'à** $\|\theta_k - \theta_{k+1}\| < \epsilon$;

### *Gaussian Process Temporal Differences* (GPTD)

Une application des processus gaussiens aux problèmes de RL est présentée dans [Engel 03]. Cette méthode est moins classique que les précédentes, mais comme elle a été utilisée avec succès sur des problèmes de RL pour les intractions homme-machine [Gašić 10] grâce à la gestion de l'incertitude sur l'estimation de la fonction de qualité qu'elle fournit, elle est succintement présentée ici. C'est une méthode du second ordre qui propose un apprentissage en-ligne et on-policy.

Elle se base sur l'hypothèse que la fonction de qualité est un processus gaussien [Rasmussen 06], c'est-à-dire que chaque valeur de l'espace sur lequel il est défini est considérée comme étant une variable aléatoire de distribution gaussienne. Ce processus est alors entièrement déterminé par ses informations de moyenne $\mu(s, a)$ et de variance $\sigma^2(s, a)$. Cette dernière renseigne sur la précision de l'estimation de la fonction de qualité. Plus cette information est grande moins l'estimation est certaine. Plus la gaussienne est fine en ce point, meilleure est l'estimation.

La fonction de qualité n'étant pas directement observée, un modèle génératif est mis en place pour lier les observations à cette dernière. Les observations sont fournies par les récompenses. Le modèle génératif est déterminé de la façon suivante, en utilisant l'équation d'évaluation de Bellman échantillonnée et en introduisant un bruit blanc gaussien $n$ qui permet de prendre en compte l'écart entre la vraie équation et celle échantillonnée. A l'instant $t$,

$$r_t = \hat{Q}(s_t, a_t) - \gamma \hat{Q}(s_{t+1}, a_{t+1}) + n(s_t, a_t). \tag{3.31}$$

Pour une trajectoire de longueur $t$, les notations suivantes sont introduites pour stocker les données rencontrées :

$$R_t = [r_1 \ \cdot \ r_t]^T \tag{3.32}$$

$$Q_t = [\hat{Q}(s_1, a_1) \ \cdots \ \hat{Q}(s_t, a_t)]^T \tag{3.33}$$

$$N_t = [n(s_1, a_1) \ \cdots \ n(s_t, a_t)]^T \tag{3.34}$$

$$H_t = \begin{bmatrix} 1 & -\gamma & 0 & 0 & \cdots & 0 \\ 0 & 1 & -\gamma & 0 & \cdots & 0 \\ \vdots & & \ddots & & & \vdots \\ 0 & \cdots & 0 & 0 & 1 & -\gamma \end{bmatrix} \text{ telle que } H_t \in \mathbb{M}_{t-1 \times t}. \tag{3.35}$$

Avec ces nouvelles notations il est possible de réécrire l'équation de Bellman pour l'ensemble de la trajectoire :

$$R_{t-1} = H_t Q_t + N_{t-1}. \tag{3.36}$$

Cette équation constitue le modèle génératif. L'algorithme GPTD se base sur cette dernière pour calculer l'estimation de la fonction de qualité conditionnée aux observations. Cependant, par défaut, la méthode utilise tous les couples état/action rencontrés pour construire l'estimation de la fonction de qualité. Très rapidement, la représentation choisie contient trop de paramètres. Une méthode d'élagage en-ligne a été proposée dans [Engel 03] afin de construire une représentation de la fonction à estimer dont la taille reste raisonnable. Cette méthode est non-paramétrique puisque les fonctions de base ne sont pas définies par avance.

### 3.2.3 Conclusion

La résolution du problème mis sous la forme du RL a été étendue aux espaces de grande taille, ce qui est souvent le cas pour des problèmes réels. Dans le reste du manuscrit, tous les algorithmes utiliseront donc une fonction de valeur approchée. Cependant, les autres contraintes que le choix du RL implique de gérer, comme par exemple les questions du choix de la politique de contrôle et de l'exploration de l'espace d'état pour la gestion du dilemme exploration/exploitation ou la question de l'efficacité sur les échantillons, ne sont pas traitées de la même façon par les algorithmes présentés. Dans la section suivante, ces contraintes sont passées en revue ainsi que la façon dont elles sont prises en charge.

## 3.3 Prise en charge des contraintes

Les contraintes à prendre en compte pour proposer une solution efficace lors de l'interaction d'une machine avec un humain, comme par exemple un dialogue parlé ou dans le cadre d'un tutorat, ont été présentées en introduction : l'interaction doit être la plus naturelle possible et la machine doit réussir à mener à bien la tâche qui lui est impartie. L'apprentissage par renforcement a été choisi car c'est une méthode qui se base sur l'expérience de la machine, ce qui limite les *a priori* de conception. En laissant interagir la machine avec son environnement de façon légèrement guidée, la solution optimale peut être estimée. Néanmoins, la phase d'apprentissage doit se faire en perturbant le moins possible l'interaction. Elle doit donc se faire le plus rapidement possible et la phase d'exploration, gage de qualité de la solution proposée doit être faite de façon précautionneuse. De plus, la question de l'observabilité partielle doit être abordée pour récupérer le plus d'informations fiables au cours de l'interaction.

### 3.3.1 Efficacité de l'apprentissage

L'efficacité de l'apprentissage est liée à la manière dont se fait l'optimisation. Les approches les plus classiques, SARSA et Q-learning avec approximation de la fonction de valeur, présentées respectivement Algorithmes 3.6 et 3.7, sont des méthodes du premier ordre car une descente de gradient est effectuée. Elles ne sont donc pas efficaces sur les échantillons et l'apprentissage de la solution reste lent. En revanche, les approches du second ordre, représentées par LSPI [Lagoudakis 03] et GPTD [Engel 03] sont connues pour être efficaces sur les échantillons.

### 3.3.2 La nécessité d'une phase d'exploration

Il a brièvement été montré dans la Section 3.1.3, que trouver la solution optimale dépend de la manière dont est exploré l'espace d'état. Une solution différente de celle optimale courante, lors de la phase d'apprentissage, doit être testée afin de vérifier qu'il n'y a pas de meilleure

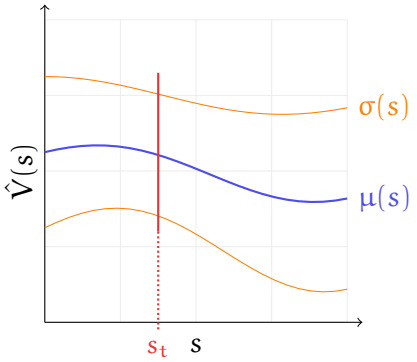



**Figure 3.5** – Modélisation de la fonction de valeur.

possibilité que la solution initialement prévue. Une des manières classiques d'appréhender l'exploration est de tester quelques fois au cours de l'apprentissage une décision aléatoire. Or, dans le cadre d'une interaction homme-machine, ces décisions aléatoires peuvent perturber l'utilisateur. Ce dilemme est particulièrement crutial dans le cas d'un algorithme en-ligne et on-policy, pour lequel toutes les décisions prises sont présentées à l'utilisateur.

### Des méthodes d'exploration sûres

Pour les méthodes de TD classiques en-ligne et on-policy comme SARSA avec approximation de la fonction de valeur, la phase d'exploration est délicate. Une solution consiste alors à utiliser, sous réserve qu'elles soient disponibles, des informations d'incertitude sur l'estimation de la fonction pour guider l'exploration. Ceci est le cas, par exemple, lorsque l'algorithme GPTD est utilisé car la fonction de qualité estimée est considérée comme étant un processus aléatoire. L'illustration de cette modélisation pour la fonction de valeur à une dimension est présentée Figure 3.5. Pour chaque état $s_t$, la valeur de $\hat{V}(s_t)$ est connue grâce à $\mu(s_t)$ et $\sigma^2(s_t)$. Ces informations permettent une exploration de l'espace d'état plus subtile et donc plus efficace. Différents schémas d'exploration utilisant ces informations d'incertitude ont été comparés dans [Geist 11]. L'approche présentée ici est celle ayant obtenu les résultats les plus satisfaisants.

Elle est appelée **bonus-glouton** inspirée de [Kolter 09]. Pour toutes les décisions à prendre, le choix de l'action suivante, à l'instant $t+1$ est donné par :

$$a_{t+1} = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left( \mu_t(s,a) + \beta \frac{\sigma_t^2(s,a)}{\beta_0 + \sigma_t^2(s,a)} \right), \tag{3.37}$$

avec $\beta$ et $\beta_0$ deux constantes, $\mu_t$ la moyenne et $\sigma_t^2$ la variance associées à l'estimation courante de la fonction de qualité $\hat{Q}_t$. Le choix de l'action est fait selon la moyenne courante estimée pondérée par l'incertitude à propos de la fonction de qualité pour le couple $(s,a)$. Quand la variance $\sigma_t^2(s,a)$ est élevée, un bonus (valant jusqu'à $\beta$) est donné pour le choix de cette action, favorisant ainsi l'exploration. Quand l'espace est de plus en plus exploré, l'incertitude décroît et la variance tend à diminuer. Le choix de l'action gloutonne est ainsi favorisée. Au cours de ce processus, les actions ayant une valeur moyenne haute seront favorisées par rapport à celle ayant des valeurs plus faibles, mais si deux actions ont une valeur moyenne équivalente alors celle présentant la plus grande incertitude sera choisie.

#### Une alternative

Alternativement au développement de méthodes d'exploration sûres afin de présenter des stratégies les plus fiables possibles rapidement à l'utilisateur dans le cadre d'une utilisation en-ligne et on-policy, des méthodes **off-policy** peuvent être envisagées. En effet, dans ce cas, la politique présentée à l'utilisateur est une politique de contrôle. Elle est déterminée à l'avance par le concepteur du sytème ce qui assure qu'il n'y ait pas de décisions aléatoires lors de l'interaction. Le problème de l'exploration hasardeuse ne se pose donc pas directement. Cette approche est employée dans LPSI qui propose un apprentissage hors-ligne et Q-learning que ce soit en mode hors-ligne ou en-ligne.

### 3.3.3 Gestion de l'observabilité partielle

Les algorithmes utilisent la représentation d'état qui leur est fournie pour procéder à l'apprentissage. La précision des résultats qu'ils vont être capables de donner dépend directement de la qualité des informations qui leur est fournie. Ceci est d'autant plus vrai que tous les algorithmes présentés se basent sur les équations de Bellman. Or, ces équations ne sont définies que si la propriété de Markov est respectée. Dans le cas d'interactions homme-machine, un soin particulier doit être apporté dans la construction de l'état pour que les informations qu'il contient la respectent, sous peine de voir les algorithmes présenter des résultats incohérents. Etant donné que de la taille de l'historique dépend directement la qualité de l'état construit, il est intéressant de jouer sur la représentation de la fonction de qualité afin de prendre en compte une quantité plus importante d'information sur laquelle se base chaque décision. Par exemple, une paramétrisation non-linéaire est plus efficace à nombre de paramètres égal avec une représentation linéaire. Les paramétrisations non-linéaires ne sont acceptées ni par les algorithmes LSPI, ni par GPTD qui utilise une représentation non-paramétrique.

### 3.3.4 Conclusion

Différentes méthodes ont été présentées dans ce chapitre pour prendre en compte les contraintes imposées par l'apprentissage par renforcement. La question de la gestion des grands espaces est abordée au travers de l'utilisation de fonctions de valeurs approchées. La manière dont différents algorithmes classiques et GPTD prennent en compte les contraintes imposées par le choix du RL ont été présentées dans cette section. Chaque algorithme présente des avantages, comme par exemple le choix d'une utilisation en-ligne et on-policy (SARSA) avec un schéma d'exploration efficace (GPTD), en-ligne et on-/off-policy (Q-learning), hors-ligne et off-policy (LSPI), efficace sur les échantillons (GPTD et LSPI) ou encore autorisant une paramétrisation non-linéaire, mais aucun d'entre eux ne réunit toutes les caractéristiques à la fois. C'est pourquoi, dans la partie suivante, nous proposons d'utiliser un algorithme autorisant la prise en charge de toutes ces contraintes. Il est ensuite testé sur le problème de dialogue parlé.

# Deuxième partie

# Systèmes de dialogue parlé

# Chapitre 4

# Systèmes de dialogue

Avec les progrès réalisés ces soixantes dernières années en traitement du signal et de l'information, il est devenu envisageable de faire interagir un utilisateur humain avec un serveur vocal. Cette machine a pour but de faire gagner du temps à l'utilisateur en l'orientant directement vers un service précis ou à faire gagner de l'argent à l'entreprise qui fait appel à ses services afin de traiter plusieurs demandes qui lui parviennent en même temps. Cependant, de nets progrès sont encore à faire car les services proposés restent la plupart du temps rudimentaires, contraignant l'utilisateur à n'employer qu'un seul mot à la fois ou à lui faire souvent confirmer ses dires au cours de l'interaction. Des efforts de recherche sont alors déployés afin de concevoir des systèmes proposant une interaction proche de celle que pourraient avoir deux personnes dialogant ensemble. Après une présentation plus détaillée des serveurs vocaux auxquels nous nous intéressons dans ces travaux, un état de l'art est proposé.

## 4.1 Présentation des systèmes de dialogue parlé

Les systèmes de dialogue étudiés dans ces travaux sont ceux pour lequel le dialogue est parlé et est mené afin de voir une requête de l'utilisateur satisfaite à l'issue de l'interaction. Ils ont pour but, par exemple, de fixer les modalités d'un rendez-vous [Janarthanam 11], d'aider à la composition de numéros grâce à la voix [Williams 08],de fournir des renseignements touristiques [Young 10], de présenter un musée [Swartout 10], ou encore de soumettre une tâche à un robot [Roy 00]. La requête de l'utilisateur contient plusieurs attributs qu'il incombe au système de découvrir et de satisfaire. Pour ce faire, chacune des deux parties participant au dialogue, la machine et l'utilisateur, interviennent à tour de rôle, plusieurs fois, chaque intervention de l'une puis de l'autre constituant un *tour* de dialogue. Ce mécanisme est présenté Figure 4.1. Nous rappelons que dans le cas d'un dialogue parlé, plusieurs modules doivent être conçus, pour d'abord faire parvenir l'information de l'utilisateur à la machine et ensuite la retransmettre de la machine à l'utilisateur. Un système de dialogue est donc composé, à partir de l'utilisateur, d'un module de compréhension du langage parlé (*Spoken Language Understanding*, SLU en anglais), qui regroupe les modules de reconnaissance vocale et l'analyseur sémantique, d'un gestionnaire de dialogue (*Dialogue Manager*, DM en anglais) chargé de prendre les décisions à partir d'informations collectées en provenance du module de compréhension du langage parlé, et enfin d'un module de génération de language naturel (*Natural Language Generation*, NLG en anglais) qui regroupe les modules retranscrivant la décision en texte puis le texte en signal audio par synthèse vocale. Tous les modules, sauf le gestionnaire de dialogue, utilisent des

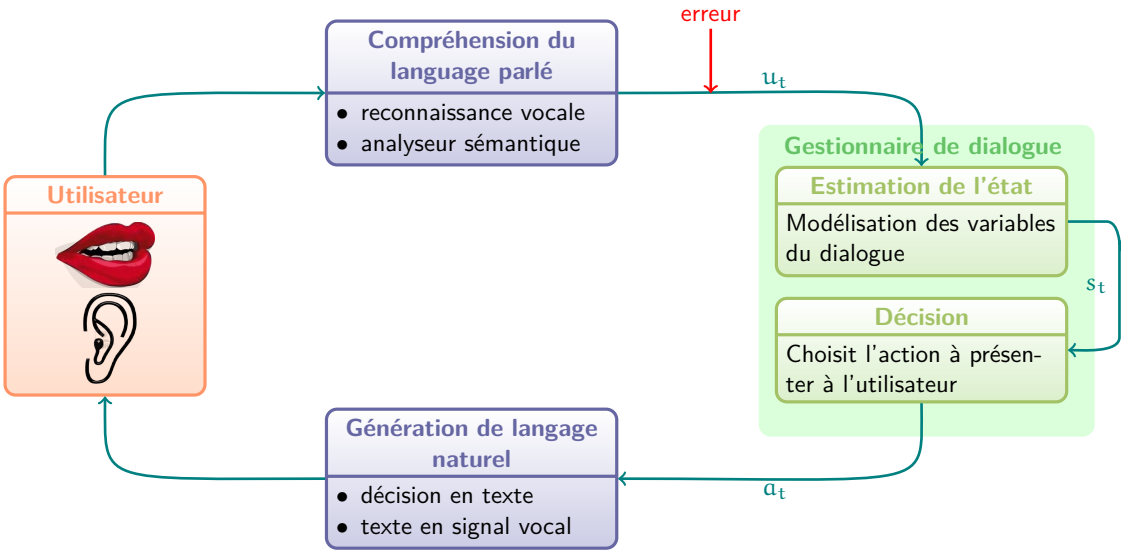



**Figure 4.1** – Détails d'un tour de dialogue à l'instant $t$. Le gestionnaire de dialogue utilise les informations issues de la compréhension du langage parlé $u_t$ pour construire l'état $s_t$ sur lequel se base sa décision $a_t$. Cette dernière est ensuite transférée à l'utilisateur via le module de génération de langage.

méthodes d'apprentissage automatique : la reconnaissance de parole [Jelinek 97], la génération de langage [Reiter 00] ou encore la synthèse vocale à partir de texte [Damper 01]. Un progrès consisterait à faire bénéficier le gestionnaire de telles méthodes.

A chaque nouveau tour, le gestionnaire est chargé de prendre une décision avec l'objectif final de mener à bien le dialogue. La gestion est ainsi un problème de prise de décisions séquentielles. Lors de la conception des premiers gestionnaires, les stratégies étaient définies à la main par un jeu de règles [Larsson 00] ou par des machines à états [Pietquin 03]. Cependant, dans ce cas, le nombre de situations envisagées est fini et petit. Or, dans un dialogue naturel, tous les contextes doivent pouvoir être pris en compte. De plus, ces solutions requièrent de connaître l'objectif à l'avance et une fois la solution trouvée, elle ne peut pas être modifiée ni réutilisée pour un autre système de dialogue. Dès lors qu'une implémentation sur une nouvelle tâche est envisagée, tout le travail de conception des stratégies est à refaire. C'est pourquoi la recherche s'est tournée vers des solutions apprises à partir de données.

Les problèmes de dialogue ont alors été mis sous la forme d'un MDP, cadre de travail classique pour résoudre le problème de prise de décision séquentielle. D'après la Figure 4.1, le gestionnaire de dialogue se base sur les informations fournies par le module de compréhension du langage. Or, ce module est sujet à erreur. La plupart de ces modules actuels proposent donc une liste d'hypothèses à laquelle sont associés des taux de confiance sur ce qui a été probablement reconnu, qu'il incombe alors au gestionnaire de dialogue de prendre en compte pour proposer la meilleure décision possible. Ce dernier travaille donc dans un environnement partiellement observable pour prendre ses décisions. Les actions à disposition du gestionnaire de dialogue sont, par exemple : informer, demander plus d'informations, ou encore demander une confirmation implicite ou explicite. Elles sont en nombre relativement restreint. La récompense est donnée en fonction de la satisfaction de l'utilisateur et de la qualité du déroulement du dialogue. Des mesures objectives sont utilisées comme le nombre de tours de dialogue, l'accomplissement de la tâche requise, ou une combinaison de plusieurs de ces facteurs [Walker 97]. Enfin le facteur d'actualisation $\gamma$ est fixé par le concepteur du système. Il pondère les récompenses futures. La

mise en forme du problème selon un MDP a été proposée pour la première fois dans [Levin 97]. Cependant, la collecte des données pour cette première approche est très coûteuse en terme d'échantillons et d'annotations ce qui a mené à l'introduction de simulations d'utilisateurs et d'apprentissages efficaces en-ligne ou hors-ligne. Dans la suite du manuscrit, la tâche utilisée pour illustrer les travaux sur le dialogue est la tâche *CamInfo* et la construction de l'état du dialogue est basée sur le paradigme *Hidden Information State* (HIS) [Young 06, Young 10] sur lesquelles des précisions détaillées seront données au Chapitre 5 à la Section 5.1.

## 4.2 Simulations d'utilisateurs

Les premiers gestionnaires de dialogue ayant été très gourmands en échantillons, il a fallu donc construire des simulations d'utilisateurs afin de générer suffisamment de données pour pouvoir entraîner le système [Levin 00]. La première a consisté à utiliser l'historique des paires observations/actions pour construire ce qui est appelé un N-*gram*, le N correspondant à la profondeur de l'historique. Cet état est basé sur le fait que l'hypothèse d'apparition de la paire suivante dépend de celles qui sont déjà apparues dans la séquence. Un modèle $1-$gram a été proposée dans [Eckert 97]. D'autres ont été employés dans [Georgila 05, Georgila 06]. Les probabilités sont apprises à partir de corpus de données réelles. Cependant, aucun corpus ne peut couvrir toutes les possibilités. De plus, la difficulté d'utilisation de cette simulation réside dans le choix de la profondeur de l'historique.

Une amélioration de ces simulations a été proposée dans [Pietquin 02]. Elle consiste à ne plus considérer seulement ce qui a été dit par l'utilisateur mais aussi de considérer ses intentions et de travailler ainsi à plus haut niveau dans la modélisation du dialogue. Cette idée a été reprise dans [Schatzmann 07a] qui détermine le but de l'utilisateur et lui associe une liste, appelée *agenda* qui a la structure d'une pile. Elle contient les actions de l'utilisateur rencontrées au cours du dialogue qui sont nécessaires pour la détermination du but fixé par ce dernier. Dans ce premier article, les paramètres de la simulation sont déterminés manuellement. Des amélioration pour apprendre les paramètres à partir de corpus de données sont proposées dans [Schatzmann 07c, Keizer 10].

Des approches différentes consistent à utiliser des réseaux bayésiens [Pietquin 06b] et des chaînes de Markov cachées (*Hidden Markov Model*, HMM en anglais) [Cuayáhuitl 05]. Elles présentent l'avantage de pouvoir modéliser une large gamme de dépendances conditionnelles et peuvent être entraînées à partir de données. En revanche, elles présentent l'inconvénient de nécessiter de larges corpus pour déterminer tous les paramètres du modèle.

Enfin, l'approche la plus récente consiste à reproduire directement le comportement de l'utilisateur grâce à l'apprentissage par renforcement inverse pour déterminer la récompense de l'utilisateur à partir de dialogues entre deux humains [Chandramohan 11].

Cependant, pour que les simulations d'utilisateurs soient performantes, il est nécessaire de générer des conditions de bruit similaires à celles qui pourraient se rencontrer dans des situations réelles. Plusieurs travaux ont proposé une modélisation de situations bruitées [Pietquin 05, Garcia 07, Schatzmann 07b]. Ils interviennent au niveau de la reconnaissance vocale. D'autre part, bien que fournissant de bons résultats, les différents modèles d'utilisateurs et de bruit, qu'ils soient construits de toutes parts à partir d'heuristiques ou appris, introduisent un biais ayant pour conséquences de proposer une stratégie de dialogue, non pas adaptée à chaque utilisateur, mais prévue pour un utilisateur moyen [Pietquin 11c]. De plus l'influence de ce biais sur l'apprentissage est difficile à quantifier [Schatzmann 05, Ai 06]. Les approches sans simulation d'utilisateurs pour apprendre les stratégies de dialogue avant d'être testées sur des

utilisateurs réels, peuvent donc être préférables.

## 4.3 L'observabilité partielle

Comme présenté dans l'introduction, le gestionnaire de dialogue doit faire face à des informations incertaines en provenance du module de compréhension du langage. Deux façons de résoudre le problème peuvent être envisagées. La première consiste à estimer les paramètres du MDP (ou du POMDP) dans une approche que l'on appelle *model-based*. La stratégie est ensuite calculée à partir de ce modèle (et non apprise). La seconde approche place le problème de dialogue dans le cadre des MDP en définissant, à partir des hypothèses sur la construction du langage, un état contenant suffisamment d'information de l'historique des interactions pour que la décision courante soit efficace. Cette approche est dite *model-free*. La stratégie est ensuite apprise avec un algorithme d'apprentissage par renforcement.

### 4.3.1 Approche basée sur un modèle

Les premières approches visant à estimer les paramètres du MDP pour ensuite calculer la stratégie optimale à partir de ce dernier ont été proposées dans [Levin 98, Singh 99]. Puis, avec l'apparition d'algorithmes plus performants pour résoudre les POMDP, ces derniers ont été utilisés dans le problème d'optimisation de dialogue. Une des premières approches utilisant les POMDP est proposée dans [Roy 00]. Une application utilisant des commandes vocales pour aider des personnes atteintes de troubles de la mémoire est proposée dans [Hoey 07]. D'autres améliorations suivent dans [Pinault 11, Chinaei 12] avec l'introduction d'un espace résumé pour gérer le fait que la résolution d'un POMDP devient impossible à calculer sur de trop grands espaces. Un apprentissage des paramètres du modèle basé sur une approche bayésienne est proposée dans [Png 12]. Par ailleurs, cette méthode reste basée sur l'apprentissage d'un modèle introduisant un biais dont il est préférable de limiter l'apport. Pour éviter cela, des approches sans modèle sont préférées.

### 4.3.2 Approche sans modèle

L'apprentissage d'une stratégie sans modèle se fait directement à partir des informations qui parviennent au gestionnaire de dialogue. Un important travail d'ingénierie doit être fait pour construire une représentation compacte des informations nécessaires à la prise de décisions. Cela peut se faire de deux façons différentes. La première consiste à utiliser une liste des N meilleures hypothèses reconnues par le module de compréhension du langage à chacune desquelles il est associé un taux de confiance. Par exemple, dans [Young 10], l'état est inféré à partir des observations. Ce paradigme est appelé *HIS* (pour *Hidden Information State*). Une approche similaire pour construire un modèle d'observation est développée dans [Henderson 08]. De plus, étant donné que les espaces d'états sont grands, une méthode a été proposée dans [Williams 05b] pour factoriser les états. L'autre solution envisagée consiste à utiliser une approche bayésienne qui factorise les buts de l'utilisateurs en concepts [Thomson 10]. Cependant, avec ces constructions de l'espace d'état, aucune garantie quant au fait que l'état nouvellement construit respecte la propriété de Markov n'est apportée. La question du cadre markovien est abordée dans [Paek 05]. De plus, l'apprentissage d'une stratégie sans modèle peut se faire de deux façons. Soit il se fait hors-ligne, à partir de données déjà collectées. Cela permet la réutilisation de données, qui sont coûteuses à collecter. Ces approches ont été proposées dans [Li 09a, Pietquin 11b]. Un

algorithme LSPI est utilisé pour réaliser l'optimisation. L'autre solution consiste à faire l'apprentissage de la politique optimale en-ligne, par exemple dans [Gašić 10] à l'aide de l'algorithme GPTD ou encore [Pietquin 11d] pour une approche avec KTD, où une approche on-policy est proposée. Une approche basée sur une méthode de recherche directe dans l'espace des politiques à partir d'une descente de gradient a été présentée dans [Jurčíček 10]

## 4.4 Conclusion

Afin d'éviter l'utilisation d'une simulation d'utilisateurs ou d'un modèle de (PO)MDP, les approches sans modèle, se basant directement sur les données seront préférées. Cependant, comme dans toute interaction homme-machine, des contraintes sont à prendre en compte lors de la conception du système afin qu'il puisse mener à bien la tâche impartie. A notre connaissance, il n'existe pas d'algorithme capable à la fois de prendre en compte l'efficacité sur les données, un apprentissage au choix en-ligne ou hors-ligne, on-policy ou off-policy, une gestion de la représentation de la fonction de qualité linéaire ou non-linéaire et la gestion de la non-stationnarité.

Les travaux présentés dans ce manuscrit se concentrent sur deux aspects précis. Le premier concerne la gestion de l'exploration quant à la prise de décision et l'autre la question de l'observabilité partielle. Un algorithme convenant particulièrement bien à la résolution du problème de dialogue parlé est testé car il permet de prendre en compte de manière efficace de nombreux aspects inhérents à ce problème. Après les descriptions de l'application du dialogue au cadre markovien et des hypothèses faites pour présenter nos travaux, les contributions pour gérer les deux aspects évoqués au début de ce paragraphe sont présentées au Chapitre 5 pour l'algorithme et au Chapitre 6 pour la présentation d'une alternative au cadre des MDP.

# Chapitre 5

# Gestionnaire de dialogue basé sur KTD

Dans le chapitre d'introduction, les qualités que devrait exhiber le système de dialogue pour être performant ont été présentées. Le cadre de l'apprentissage par renforcement a été choisi car il permet de prendre en compte un certain nombre de ces contraintes. La majorité des solutions au problème de RL ne peuvent être calculées que si le problème est mis sous la forme d'un MDP (Section 3.1.1).

Nous rappelons brièvement les caractéristiques attendues du système de dialogue parlé. Tout d'abord, les informations collectées à propos de l'utilisateur doivent être les plus complètes possibles. Si une partie de l'information est manquante, la décision ne sera pas adaptée. D'autre part, l'exploration de l'espace d'état, nécessaire au bon déroulement de l'apprentissage doit se faire de façon efficace, surtout s'il est effectué en-ligne et on-policy. L'approche classique consistant à tester une action aléatoire de temps en temps (l'approche $\epsilon$-glouton) ne devrait pas être choisie pour ce type d'application car elle implique un comportement de la machine qui n'est pas cohérent pour l'utilisateur. De plus, étant donné que différentes personnes font face à la machine et que la manière dont ils répondent varie en fonction de leur habitude à long terme à se servir du système, le comportement de ce dernier devrait pouvoir s'adapter. Ainsi, il serait souhaitable qu'une approche permettant de gérer les non-stationnarités soit proposée. Enfin, l'apprentissage de la solution optimale devrait être rapide en termes de données d'entraînement. En effet, si cette condition est respectée, cela permet de s'affranchir d'une simulation d'utilisateurs pour générer suffisamment de données et cela limite ainsi les *a priori* de conception et permet un gain de généricité. Si l'algorithL'illustration de la relaxe des contraintes est présentée Figure 5.1. La contribution que les travaux présentés dans ce chapitre apportent consiste à s'affranchir du modèle d'utilisateurs.

Après quelques explications concernant la manière dont est gérée l'observabilité partielle par le système de dialogue parlé utilisé dans nos expériences, une présentation d'un algorithme convenant particulièrement à notre application sera proposée. La façon dont l'algorithme des différences temporelles de Kalman (*Kalman Temporal Differences*, KTD en anglais) prend en compte les points critiques pour construire un bon système de dialogue est détaillée par la suite.

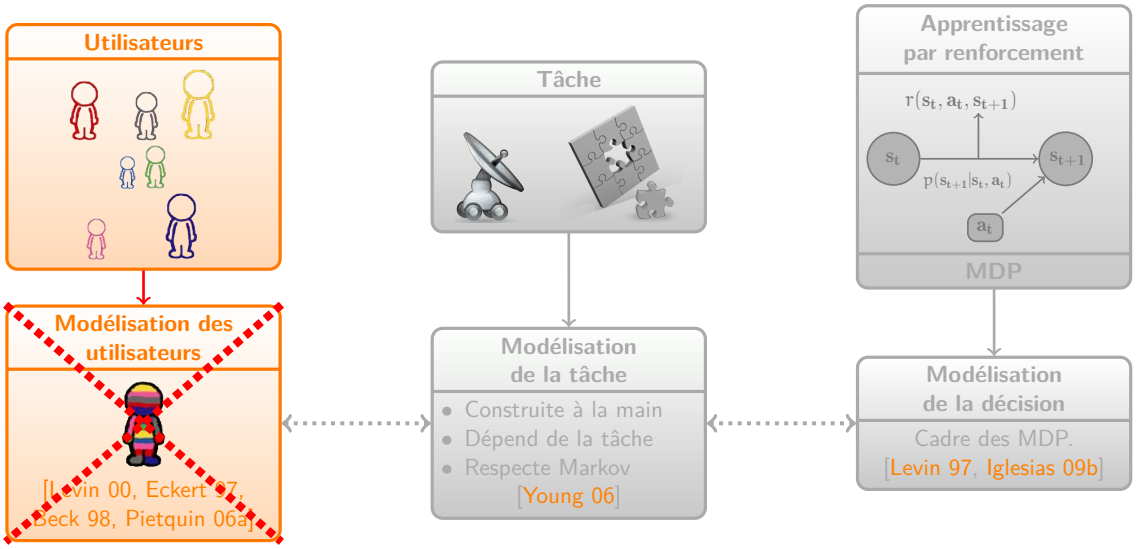



**Figure 5.1** – La première contribution présentée se focalise sur la modélisation de l'utilisateur.

## 5.1 L'observabilité partielle

La tâche utilisée pour les expériences présentées est *CamInfo*. Elle a été développée par l'Université de Cambridge afin de fournir à l'utilisateur des informations touristiques sur la ville de Cambridge. La requête de d'utilisateur peut contenir jusqu'à douze attributs : le type de lieu (restaurant, musée, bar, ...), l'adresse, le numéro de téléphone, la gamme de prix si c'est un restaurant ou un bar, la mise à disposition d'une connexion internet, *etc*. La tâche est remplie une fois que le système a récupéré les différents attributs de la requête de l'utilisateur et a pu satisfaire à sa demande. Un exemple de dialogue obtenu avec la tâche CamInfo est présenté Figure 5.2. Le formalisme du dialogue vient d'un standard mis en place par les concepteurs de la tâche [Young 10]. A chaque tour de dialogue, apparaît ce qui a été dit par le système, puis ce qui a été dit par l'utilisateur et enfin ce qui a été reconnu. Entre accolades, la probabilité des hypothèses reconnues est donnée et entre crochets la décision du gestionnaire est précisée. A l'issue du dialogue la requête de l'utilisateur est satisfaite. Il a été testé avec des utilisateurs réels [Jurčíček 11, Gašić 11, Gašic 13]. Pour des besoins de reproductibilité des expériences, toutes celles présentées ici ont été réalisées avec une simulation d'utilisateurs basée sur un agenda [Schatzmann 07a]. Cette tâche utilise le paradigme *Hidden Information State*, HIS [Young 06, Young 10] pour construire l'espace d'état permettant au gestionnaire de dialogue de prendre des décisions efficaces.

### 5.1.1 L'espace d'états

Le paradigme HIS construit un espace de travail à partir des hypothèses retournées par le module de reconnaissance vocale et l'analyseur sémantique (le taux d'erreur de reconnaissance simulé étant fixé à $10\%$ lors des expériences). Ces hypothèses prennent en compte les actions effectuées par l'utilisateur, à chacune desquelles un taux de confiance est associé. L'aspect partiellement observable a pour conséquence que ces seules informations sont insuffisantes pour prendre une décision efficace. En effet, l'état réel du dialogue est constitué de trois variables. A un instant donné, la première contient les intentions de l'utilisateur (quelle est la requête qu'il

- Sys> hello() [Greet]
  Usr> hello(=supermarket)
  Rec> hello(=supermarket) {1.00}
- Sys> affirm(area) [BoldRQ]
  Usr> select(area=romsey)
  Rec> select(area=romsey) {1.00}
- Sys> select(name=Winfield,type=shopvenue,shoptype=supermarket,area=romsey) [FindAlt]
  Usr> affirm(phone,postcode)
  Rec> affirm(phone,postcode) {0.94}
  affirm(stars,stars) {0.06}
- Sys> select(name=Winfield,phone="01223322888",postcode="CB1 2AS") [Inform]
  Usr> ack()
  Rec> ack() {0.94}
  affirm(hasinternet) {0.06}
- Sys> select(name=Winfield,type=shopvenue,shoptype=supermarket,area=romsey) [Offer]
  Usr> thankyou()
  Rec> thankyou() {0.94}
  negate(hasmusic=true) {0.06}
- Sys> select(name=Winfield,type=shopvenue,shoptype=supermarket,area=romsey) [SysRepeat]
  Usr> bye()
  Rec> bye() {0.94}
  ack() {0.06}

**Figure 5.2** – Exemple de dialogue obtenu avec la tâche CamInfo.

souhaite voir accomplie), la deuxième, l'action qu'il vient d'effectuer (ce qu'il a dit au système de dialogue) et la dernière contient un résumé du dialogue. Comme ces variables sous-jacentes ne sont pas accessibles, un étatde croyance (voir Section 3.1.1) est défini. Il est construit à partir d'informations provenant de quatre modèles différents. Etant donné le biais que ces modèles peuvent introduire, aucune garantie n'est apportée quant au respect strict de la propriété de Markov.

La première hypothèse faite est la mise à disposition de ce que les auteurs appellent un **modèle d'observations**. Celui-ci est contruit à partir de la liste d'hypothèses avec le taux de confiance associé, contenant ce qui a possiblement été reconnu. La deuxième est un **modèle d'action** de l'utilisateur. Il permet de définir quelle est l'action la plus probable effectuée par l'utilisateur. Il est basé sur un modèle de paires adjacentes, contruit à partir des actions du gestionnaire de dialogue (qui sont, elles, connues). Il a en effet été remarqué qu'à une question est généralement associée une réponse, à une confirmation une affirmation, *etc.* La troisième information pour construire l'état de croyance est un **modèle de l'intention** de l'utilisateur. Il est construit à partir de règles simples sous forme d'arbres qui définissent quelles sont les caractéristiques à associer à un élément de la requête. Si, par exemple l'utilisateur demande un restaurant, il faut lui associer le type de nourriture et le prix. Enfin, la quatrième information contient un **modèle de l'historique** du dialogue. Ce modèle tient à jour les différents attributs de la requête de l'utilisateur à partir de règles pré-établies.

Cependant, cet espace est trop important pour être utilisé directement. La notion d'espace résumé a alors été introduite [Williams 05a]. Elle est basée sur la supposition que les taux de confiance associés aux deux meilleures hypothèses fournissent une information suffisante pour prendre la décision. Si les taux sont très différents, la première hypothèse est considéré comme étant la bonne. S'ils sont proches, des informations récupérées de l'espace initial sont utilisées pour confirmer quelle hypothèse est la meilleure.

En pratique, l'espace résumé comporte donc quatre variables qui reprennent en partie les composantes de l'état sous-jacent estimé. C'est à partir de cet espace de taille raisonnable que se basent les décisions du gestionnaire de dialogue. La première variable, $s^1$, contient une estimation des intentions de l'utilisateur. C'est une variable discrète pouvant prendre six valeurs différentes. La deuxième variable, $s^2$, contient l'estimation de la dernière action de l'utilisateur. Cette variable discrète peut prendre jusqu'à vingt-deux valeurs. Les deux dernières dimensions, $s^3$ et $s^4$, contiennent respectivement le taux de confiance (valeur réelle comprise entre zéro et un) associé aux deux historiques de dialogue les plus probables. L'espace d'état sur lequel se base la décision du gestionnaire de dialogue est donc résumé ainsi : $S = \{< s^1, s^2, s^3, s^4 > | s^1 \in [\![0, 5]\!], s^2 \in [\![0, 21]\!], (s^3, s^4) \in [0, 1] \times [0, 1]\}$. Sa validité est conditionnée à la fiabilité des modèles utilisés.

Dans le cas du dialogue parlé, après quelques décennies de recherche scientifique, ces modèles ont atteint une maturité permettant de donner de très bons résultats grâce aux progrès réalisés en traitement de signal (la voix), et grâce à l'élaboration de modèles statistiques concernant le langage. Même si ce qu'a dit l'utilisateur n'est pas parfaitement reconnu, il est possible de prendre une décision efficace par l'analyse de la construction du langage et de la façon dont se déroule un dialogue. Nous profitons de cette section pour donner les informations complémentaires sur la modélisation de la tâche CamInfo, résumée Figure 5.3.

### 5.1.2 L'espace d'actions

Le choix des actions du gestionnaire de dialogue se fait parmi douze meta actions possibles. Cependant, pour proposer une plus grande diversité d'actions aux utilisateurs, ce choix peut

être raffiné. Ainsi, jusqu'à vingt-deux actions différentes peuvent être présentées. Le passage de l'action choisie à l'action effectivement proposée se fait en utilisant les hypothèses de dialogue les plus probables et en se basant sur des règles expertes dépendantes du domaine. Les actions (parmi les douze) sont par exemple : saluer l'utilisateur, demander ou confirmer une information, ou faire un choix entre deux propositions. L'espace d'action se résume ainsi : $A = \{a | a \in [\![0, 11]\!]\}$.

### 5.1.3 La fonction de récompense

La récompense associée à chaque contexte de dialogue et à chaque action dépend de la satisfaction du but de l'utilisateur et du temps mis pour y parvenir. Ainsi, chaque tour de dialogue est pénalisé par une récompense négative de $-1$. A l'issue du dialogue, si l'utilisateur a vu sa requête satisfaite, un récompense de $+20$ est apportée et de $0$ sinon. Le facteur $\gamma$ est fixé à l'unité. Le cumul de récompense informe donc directement sur la durée du dialogue.

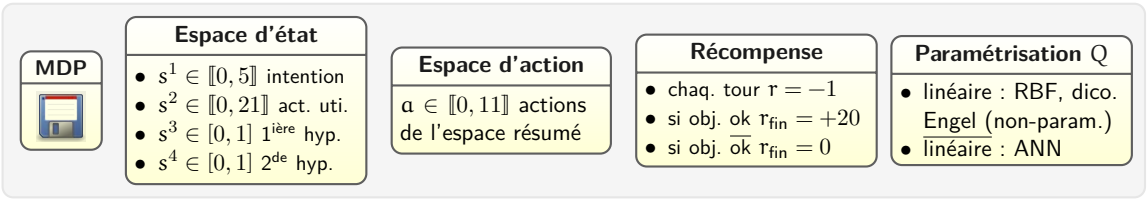



**Figure 5.3** – Modélisation du dialogue avec HIS.

## 5.2 Prise en compte des autres critères

Un algorithme d'apprentissage par renforcement, *Kalman Temporal Differences*, KTD, est maintenant utilisé pour résoudre le problème d'optimisation du dialogue parlé. Cette méthode convient à ce problème car elle prend en compte les divers points critiques pour concevoir un système de dialogue performant. La manière dont cela est fait sera présenté dans cette section, après une présentation du principe de l'algorithme.

### 5.2.1 Les différences temporelles de Kalman

Une présentation complète de l'algorithme est donnée dans [Geist 10]. L'algorithme vise à estimer les paramètres de la représentation de la fonction de qualité. Les points principaux sont rappelés ici. L'utilisation du filtre de Kalman [Kalman 60] dans des problèmes de prédiction est très fréquente. L'objectif de ce filtrage est de retrouver la valeur de variables cachées à partir de mesures bruitées. Les variables cachées sont modélisées par un vecteur de variables aléatoires. Le filtre de Kalman vise à estimer les premier et second moments (moyenne et variance) de la distribution de l'état caché, conditionné aux mesures. Pour cela, il fonctionne de façon récursive dans un cadre en-ligne. La mesure courante est utilisée dans la mise à jour de l'estimation.

Le vecteur de variables aléatoires évolue dans le temps par le biais d'un *modèle d'évolution*. D'autre part, les variables cachées sont liées aux mesures par un *modèle d'observation*. Le filtre de Kalman fournit le meilleur estimateur linéaire qui minimise l'erreur quadratique entre la mesure et la prédiction courantes, conditionnée aux mesures passées. L'estimateur est linéaire au sens où la mise à jour des variables cachées est linéaire par rapport à l'erreur de prédiction de la mesure future. L'algorithme KTD adapte ce principe au cadre de l'apprentissage par renforcement et plus particulièrement à l'estimation de la fonction de qualité.

Une représentation paramétrique $\hat{Q}_\theta$ est choisie dans ce cas (telle que définie Chapitre 3 à la Section 3.2), $\theta$ représentant la variable cachée au sens du filtrage de Kalman. La première équation à définir pour utiliser ce cadre est un modèle d'évolution. Comme il n'y a aucune raison qui orienterait ce choix, le principe du rasoir d'Occam est appliqué. Le modèle d'évolution est considéré comme une marche aléatoire : $\theta_t = \theta_{t-1} + \nu_t$, avec $\nu_t$ un bruit blanc, de matrice de covariance $P_{\nu_t}$. Grâce à ce bruit blanc, des solutions non-stationnaires peuvent être envisagées. Le second modèle lie les mesures aux variables cachées représentées par $\theta$. Les seules mesures disponibles ici sont les récompenses. Elles sont liées à $\theta$ par les équations d'évaluation ou d'optimalité de Bellman échantillonnées. Nous rappelons que les équations de Bellman font appel à la notion mathématique d'espérance dans leur définition. Comme celle-ci n'est pas calculable avec exactitude, elle est remplacée par les informations apportées par les données. Cela donne comme modèles d'observation respectivement : $r_t = \hat{Q}_{\theta_t}(s_t, a_t) - \gamma \hat{Q}_{\theta_t}(s_{t+1}, a_{t+1}) + n_t$ ou $r_t = \hat{Q}_{\theta_t}(s_t, a_t) - \gamma \max_a \hat{Q}_{\theta_t}(s_{t+1}, a) + n_t$. Le terme $n_t$ est un bruit supposé blanc, de matrice de covariance $P_{n_t}$. Ce bruit permet de prendre en compte le fait que c'est l'équation de Bellman échantillonnée qui est utilisée et que la paramétrisation choisie peut ne pas représenter exactement la vraie fonction de qualité. Les bruits $\nu_t$ et $n_t$ sont fixés initialement au travers leur matrice de covariance, respectivement $P_{\nu_t}$ et $P_{n_t}$. Empiriquement, il apparaît que le choix de ces paramètres n'est sensible qu'à plusieurs ordres de grandeur près.

Le modèle d'évolution et d'observation est donc le suivant :

$$\begin{cases} \theta_t &= \theta_{t-1} + \nu_t \\ r_t &= g_t(\theta_t) + n_t \end{cases} \tag{5.1}$$

avec

$$g_t(\theta_t) = \begin{cases} \hat{Q}_{\theta_t}(s_t, a_t) - \gamma \hat{Q}_{\theta_t}(s_{t+1}, a_{t+1}) \text{ (évaluation)} \\ \hat{Q}_{\theta_t}(s_t, a_t) - \gamma \max_a \hat{Q}_{\theta_t}(s_{t+1}, a) \text{ (optimalité).} \end{cases} \tag{5.2}$$

Selon que l'équation d'évaluation ou celle d'optimalité est utilisée, l'algorithme sera appelé respectivement KTD-SARSA ou KTD-Q. En effet, selon l'équation, l'approche proposée par KTD sera en-ligne et on-policy (comme SARSA) ou bien en-ligne et off-policy (comme Q-learning). KTD-Q est donc aussi un algorithme pouvant être utilisé hors-ligne.

Etant donné les modèles d'évolution et d'observation, KTD fournit le meilleur estimateur linéaire minimisant l'erreur moyenne quadratique entre le vecteur de paramètres $\theta_t$ et son estimation $\hat{\theta}_t$, conditionné aux mesures passées (les récompenses passées étant $r_{1:t} = r_1 \dots r_t$). L'erreur est :

$$J_t(\theta) = E[\|\theta_t - \theta\|^2 | r_{1:t}] \tag{5.3}$$

et l'estimation du vecteur de paramètres est donnée par : $\hat{\theta}_t = \text{argmin}_{\theta, \text{màj linéaire}} J_t(\theta)$. L'aspect non-stationnaire de l'algorithme KTD est pris en charge grâce au bruit d'évolution $\nu_t$ qui permet de traquer la solution au lieu de converger vers elle.

Les raisons pour lesquelles cet algorithme convient pour résoudre le problème d'optimisation de dialogue parlé sont présentées ci-après.

### 5.2.2 Application au problème de dialogue parlé

Les trois aspects essentiels pour qu'un gestionnaire de dialogue soit de qualité, présentés en introduction de ce chapitre, sont les suivants : une gestion de l'exploration efficace, une prise en compte de la non-stationnarité et enfin une solution rapidement apprise.

#### Exploration efficace possible

L'exploration est prise en compte de deux façons différentes par KTD. Si l'approche KTD-SARSA est choisie, c'est-à-dire une approche en-ligne et on-policy, l'exploration doit être faite de façon précautionneuse. Une approche $\epsilon$-glouton est donc à déconseiller. En effet, chaque décision lors de l'apprentissage est présentée directement à l'utilisateur. Plus l'exploration est efficace, moins l'apprentissage est long et plus le nombre d'utilisateurs faisant face à une décision incohérente est petit. Etant donné que le vecteur de paramètres $\theta$ est modélisé une vecteur de variables aléatoires, la fonction $Q_\theta(s_t, a_t)$ est aussi une variable aléatoire. En conséquence, il est possible de calculer sa moyenne $\mu_t(s,a) = E[\hat{Q}_{\theta_t}(s,a)]$ ainsi que la variance associée $\sigma_t^2(s,a) = Var[\hat{Q}_{\theta_t}(s,a)]$. Un schéma d'exploration plus évolué peut alors être utilisé. Différentes explorations ont été testées dans [Geist 11]. L'approche bonus-glouton, présentée Section 3.3 a présenté les résultats les plus concluants. Si l'approche KTD-Q est utilisée, il est tout autant important d'avoir une exploration de l'espace d'état efficace, mais la décision présentée est issue d'une politique de contrôle, dont l'exploration peut être définie en amont par le concepteur.

#### Gestion de solutions non-stationnaires

La gestion de la non-stationnarité est prise en compte par la conception même de l'algorithme. En effet, l'algorithme est conçu de façon à traquer la solution plutôt que de converger vers elle grâce à l'utilisation d'un modèle d'évolution pour la solution. Une étude sur la capacité de KTD à gérer la non-stationnarité est proposée dans [Geist 09]. Tant que l'apprentissage n'est pas stoppé, à chaque nouvelle donnée reçue, la solution peut s'y adapter. Cet aspect permet de prendre en compte le fait que les utilisateurs peuvent s'habituer au comportement du système de dialogue parlé sur le long terme. Leurs réactions en seront ainsi modifiées et prises en compte.

#### Une solution du second ordre

La rapidité de l'apprentissage est garantie par le fait que KTD est une approche du second ordre, qui est connue pour être efficace sur les échantillons. Cet aspect est aussi retranscrit au travers du fait que KTD autorise des paramétrisations non-linéaires de la fonction de qualité. Ce type de paramétrisation, à nombre de paramètres égal avec une approche linéaire, ont potentiellement un pouvoir de représentation plus important.

Maintenant que les différents avantages à l'utilisation de KTD au problème de gestion de dialogue parlé ont été présentés de façon théorique, les résultats expérimentaux fournis par cette méthode sont comparés à d'autres approches. Comme les aspects « méthode efficace sur les échantillons » et « gestion de solution non-stationnaire » sont traités lors du fonctionnement même de l'algorithme, une rubrique spécifique ne leur est pas consacrée. La gestion de l'exploration et la gestion d'une paramétrisation non-linéaire en revanche font l'objet des deux sections suivantes.

## 5.3 Test de différentes stratégies d'exploration

Nous avons vu précédemment que l'exploration dépendait de la manière dont se déroulait l'apprentissage. Si l'approche est en-ligne et on-policy, elle doit être la plus efficace possible. Cela peut passer par une utilisation des informations de moyenne et de variance sur l'estimation de la fonction de qualité lorsque l'algorithme les fournit. L'autre approche et celle off-policy.

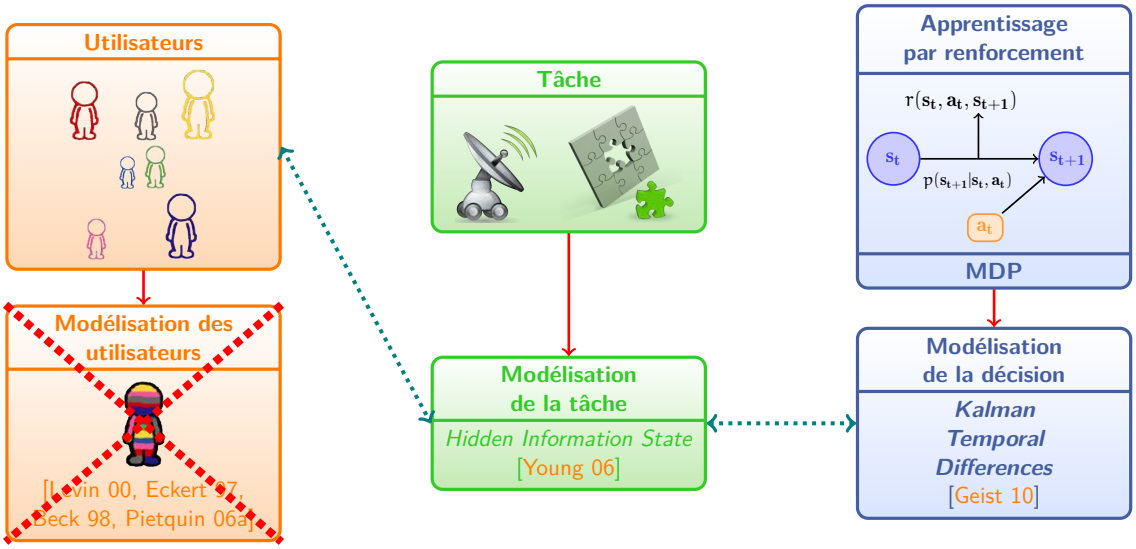



**Figure 5.4** – Solution envisagée : utilisation de KTD pour s'affranchir du modèle d'utilisateur. La modélisation de la tâche est maintenue avec HIS ainsi que le cadre markovien.

L'exploration est toujours nécessaire pour que l'apprentissage soit performant, mais elle est faite par la politique de contrôle dont le concepteur du sytème peut s'assurer qu'elle ne propose pas de décisions incohérentes. Un rappel de la configuration de la solution envisagée est présenté Figure 5.4. Après une présentation de la paramétrisation utilisée pour la fonction de qualité dans cette section, les deux approches seront testées. Les résultats donnés par KTD seront comparés à ceux retournés par d'autres approches : SARSA, Q-learning, GPTD et LSPI. Ces travaux font l'objet des publications [Daubigney 11a, Daubigney 11c, Daubigney 12a].

### 5.3.1 Paramétrisation linéaire

La paramétrisation utilisée dans cette section pour décrire la fonction de qualité est linéaire, basée sur un réseau de fonctions de base radiales et présentée Section 3.2. Elle est utilisée avec KTD, SARSA, Q-learning et LSPI. GPTD ne fait pas appel à cette paramétrisation car il en possède une en propre dans sa version originale, non-paramétrique et construite en-ligne. Nous rappelons que l'espace d'état est $S = \{< s^1, s^2, s^3, s^4 > | s^1 \in [\![0, 5]\!], s^2 \in [\![0, 21]\!], (s^3, s^4) \in [0, 1] \times [0, 1]\}$ et que l'espace d'action est $A = \{a | a \in [\![0, 11]\!]\}$ (Section 5.1).

Trois gaussiennes équi-réparties $\varphi$ sont utilisées pour chaque dimension continue de l'espace d'état, $s^3$ et $s^4$. Le choix du positionnement des gaussiennes se fait de telle façon à ce qu'elles soient bien réparties dans l'espace qu'elles doivent paver. Un compromis est à faire entre le nombre de gaussiennes dans l'espace et la précision de l'approximation. Plus il y a de gaussiennes (judicieusement réparties), meilleure est l'approximation, mais alors le nombre de paramètres à apprendre augmente lui aussi. Dans notre problème, les gaussiennes pavent un espace à deux dimensions et sont représentées par le vecteur $\phi_{3,4}$ tel que :

$$\phi_{3,4}^T(s) = [1, \varphi_1^1(s^3, s^4), \ldots, \varphi_3^3(s^3, s^4)], \tag{5.4}$$

$$\text{avec } \varphi_i^j(s^3, s^4) = \exp\left(-\frac{\|s^3 - s_i\|^2 + \|s^4 - s_j\|^2}{2\sigma^2}\right), \tag{5.5}$$

$(s_i, s_j)$ étant les centres des gaussiennes et l'écart-type ayant une valeur de $\sigma = \sqrt{0.2}$. Le deux premières dimensions sont discrètes, représentées respectivement par les vecteurs $\phi_1$ et $\phi_2$ tels que :

$$\phi_1^T(s) = [\delta_{s^1,0}, \ldots, \delta_{s^1,5}], \tag{5.6}$$
$$\phi_2^T(s) = [\delta_{s^2,0}, \ldots, \delta_{s^2,21}], \tag{5.7}$$

avec $\delta$ le symbole de Kronecker défini tel que $\delta_{a_i,a_j} = 1$ si $a_i = a_j$, $0$ sinon. Les fonctions de base $\Phi$, définies pour tout $s \in S$ et pour tout $a \in A$ sont issues de la concaténation des trois vecteurs $\phi^T(s) = [\phi_1^T \phi_2^T \phi_{3,4}^T]$ :

$$\Phi^T(s, a) = [\delta_{a,a_0}\phi^T(s), \ldots, \delta_{a,a_{11}}\phi^T(s)]. \tag{5.8}$$

Le nombre de fonctions de base est donc le suivant : la dimension du vecteur $\phi$ est $\dim(\phi) = \dim(\phi_1) + \dim(\phi_2) + \dim(\phi_{3,4}) = 6 + 22 + (1 + 9) = 38$, ce qui donne une dimension pour $\Phi$ égale à $\dim(\Phi) = \dim(\phi) \cdot 12 = 456$ ($12$ étant le nombre d'actions). Il y a ainsi $456$ paramètres à apprendre. Cette paramétrisation est issue d'un travail d'ingénierie par essais-erreurs. Les résultats obtenus avec celle-ci sont cohérents avec ceux obtenus grâce à d'autre représentations (GPTD par exemple).

### 5.3.2 Approche on-policy et en-ligne

Pour tester l'algorithme KTD, des comparaisons ont été faites dans plusieurs configurations avec différents algorithmes. Nous nous intéressons tout d'abord à une approche on-policy et en-ligne. En effet, elle présente l'avantage d'améliorer continuellement la solution trouvée, ce qui est d'intérêt pour une interaction contenant un utilisateur humain puisqu'une adaptation permanente est réalisée. A mesure que les utilisateurs apprennent à interagir avec la machine, leur comportement s'adapte, et dans cette configuration, celui de la machine aussi. Une première comparaison de trois approches est proposée, en utilisant la tâche proposée par CamInfo avec le gestionnaire de dialogue HIS décrite Section 5.1 et les algorithmes KTD-SARSA, SARSA et GPTD.

Pour ces premiers essais, le schéma d'exploration est $\epsilon$-glouton avec $\epsilon = 0.1$. En moyenne une fois sur dix, une action aléatoire est testée, et ce, tant que dure l'apprentissage. Pour tous les essais, le vecteur initial de paramètres, $\theta_0$ est fixé à $0$. Pour KTD, les valeurs des paramètres sont les suivantes : $P_{n_t} = 0.1$ et $P_{0|0} = \mathrm{Id}$, $\mathrm{Id}$ étant la matrice identité. Ces paramètres ont été trouvés par essais-erreurs et leur valeur n'est pas sensible aux petites variations. Ce n'est peut-être pas le meilleur jeu de paramètres, mais les résultats trouvés sont cohérents avec ceux des autres approches. Le taux d'apprentisage pour l'algorithme SARSA vaut $\alpha = 0.1$. Les résultats de la comparaison sont présentés Figure 5.5. La courbe présente la moyenne des récompenses cumulées obtenues lors de l'apprentissage de la solution, pour différentes tailles de jeu de données. La moyenne est tracée à partir de $50$ essais. Elle donne un indice sur la qualité du dialogue. En effet, à partir de celle-ci, la longueur moyenne du dialogue peut être déduite. Si le dialogue a abouti, une récompense de $+20$ est donnée. De plus, chaque tour de dialogue est pénalisé par une récompense de $-1$. Par exemple, sur la figure, la moyenne des récompenses cumulées est à $6$ pour SARSA, cela signifie qu'il aura fallu $20 - 6 = 14$ tours en moyenne pour voir la dialogue aboutir (car $\gamma$ vaut un).

Ainsi, KTD-SARSA retourne de meilleurs résultats que ceux donnés par SARSA et GPTD : les dialogues sont plus courts et la variance plus faible. Le fait que KTD soit plus performant

que SARSA s'explique par le fait que ce dernier est un algorithme du premier ordre et donc que l'apprentissage qu'il propose est plutôt lent. KTD présente aussi un apprentissage plus rapide et qui permet d'atteindre des scores plus élevés que GPTD. Ceci peut s'expliquer d'une part car la représentation de la fonction de valeur est différente puisque dans le cas de GPTD elle est non-paramétrique et d'autre part, par le fait que KTD propose une approche capable de gérer les non-stationnarités au travers du bruit d'évolution. En effet, dans le cas contrôlé, la politique courante varie puisqu'elle dépend de la fonction Q estimée et KTD-SARSA traque la solution alors que GPTD converge vers elle.

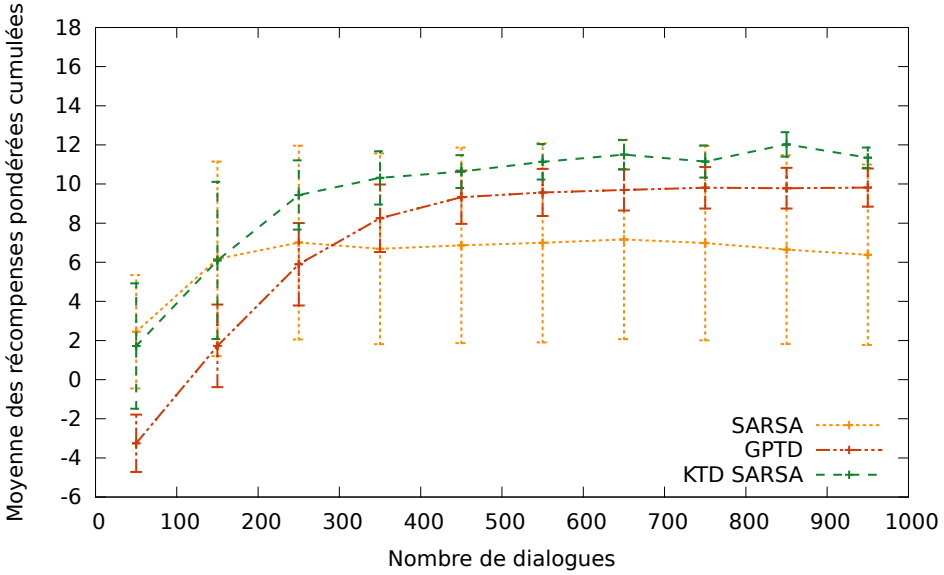



**Figure 5.5 –** Comparaison des approches online et on-policy (cas linéaire).

Néanmoins, pour que la solution on-policy et en-ligne aboutisse à de bons résultats, il est nécessaire que la période du début de l'apprentissage de la solution optimale, pendant laquelle le comportement de la machine risque d'être incohérent, soit raccourcie le plus possible. Cela passe par un schéma d'exploration avancé. De même que pour KTD, il est possible d'appliquer ce schéma à GPTD. En effet, les informations de moyenne et de variance sont fournies, étant donné que la fonction de qualité est modélisée par un processus gaussien. Le schéma bonus-glouton est choisi pour l'exploration, avec $\beta = 1000$ et $\beta_0 = 100$ (Equation 3.37). Les paramètres ont été choisis par essais-erreurs. Une comparaison de la moyenne des récompenses cumulées obtenues lors de l'apprentissage de la solution à l'aide des deux différentes explorations est proposée Figure 5.6 pour les algorithmes KTD-SARSA et GPTD. L'algorithme SARSA n'apparaît plus sur les courbes car il n'y a pas d'information disponible à propos de l'incertitude.

Avec l'approche bonus-glouton, l'apprentissage est de meilleure qualité dès le début pour les deux algorithmes par rapport au schéma $\epsilon$-glouton. La phase pendant laquelle les utilisateurs sont face à des décisions potentiellement sous-optimales est raccourcie. Par exemple, pour KTD-SARSA, il faut attendre d'avoir effectué $250$ dialogues (en moyenne) avec l'approche $\epsilon$-glouton pour atteindre le niveau de résultats obtenus aux alentours de $125$ dialogues avec l'approche bonus-glouton. Dans les deux cas, les résultats renvoyés par KTD-SARSA et GPTD avec l'approche $\epsilon$-glouton ne seront jamais au niveau de ceux retournés par celle bonus-glouton car dans le premier cas, l'exploration reste constante (une action aléatoire sur dix en moyenne) alors que dans l'autre, l'exploration décroît lorsque l'apprentissage progresse car l'incertitude diminue. Il serait n'éanmoins envisageable de choisir $\epsilon$ diminuant au fur et à mesure que l'ap-

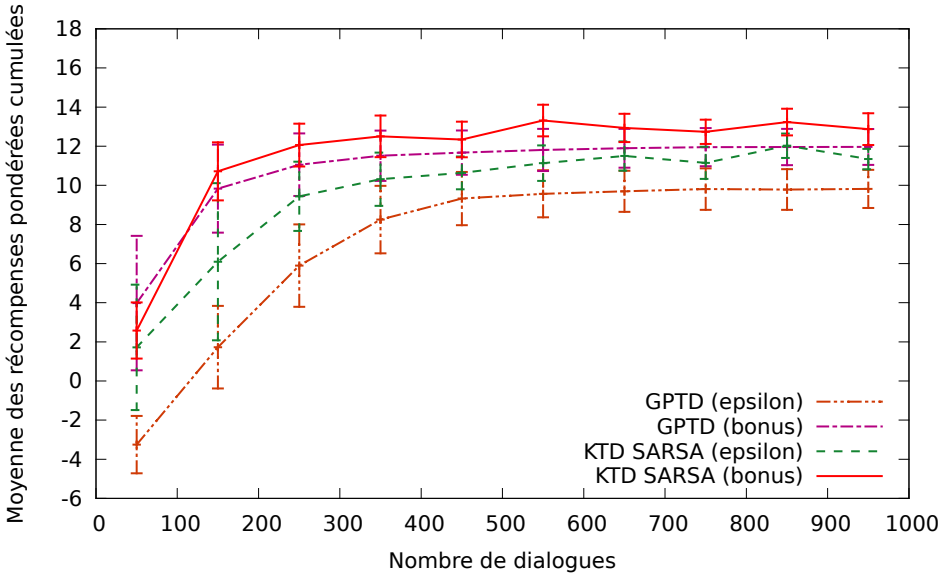



**Figure 5.6** – Comparaison des approches online et on-policy avec un schéma d'exploration bonus-glouton (cas linéaire).

prentissage progresse. Pour les mêmes raisons que précisées plus haut, KTD-SARSA donne de meilleurs résultats que GPTD, quelle que soit l'exploration choisie (comparaison deux à deux).

### 5.3.3 Approches off-policy

L'approche off-policy est utilisée comme alternative au cas on-policy car elle propose une gestion différente de l'exploration. En effet, dans le cas off-policy, la politique présentée à l'utilisateur (politique de contrôle ou comportementale) lors de l'apprentissage n'est pas celle apprise. La solution optimale est apprise à partir de données générées par une politique non-optimale. Il incombe alors au concepteur du système de proposer une politique de contrôle qui explore l'espace sans gêner l'utilisateur. Deux approches off-policy sont possibles. La première est une approche en-ligne durant laquelle la politique optimale est apprise pendant que le système interagit avec les utilisateurs tandis que la seconde, hors-ligne, utilise des données préalablement collectées. Cette dernière approche a déjà été testée sur le problème de dialogue dans [Li 09b, Pietquin 11b]. Une comparaison des résultats fournis par KTD-Q est donnée avec les algorithmes Q-learning et LSPI. Ce dernier, dans sa forme originale, ne propose qu'un apprentissage hors-ligne. L'approche GPTD n'apparaît pas car c'est une approche on-policy exclusivement. Les deux approches sont testées dans les sections suivantes.

#### Cas hors-ligne

L'approche off-policy est tout d'abord testée hors-ligne. Un jeu de données est récupéré d'un apprentissage effectué avec SARSA. Une comparaison de KTD-Q avec Q-learning (avec un taux d'apprentissage $\alpha = 0.1$) et LSPI est proposée. La moyenne des récompenses cumulées obtenues lors du test de la politique optimale apprise (donc après apprentissage) est tracée en fonction de différentes tailles de jeu de données d'entraînement sur la Figure 5.7. La politique de contrôle est précisée sur le graphique. La moyenne est tracée à partir de $50$ essais différents testés chacun $1000$ fois. Une comparaison entre les différentes approches montre que les algorithmes

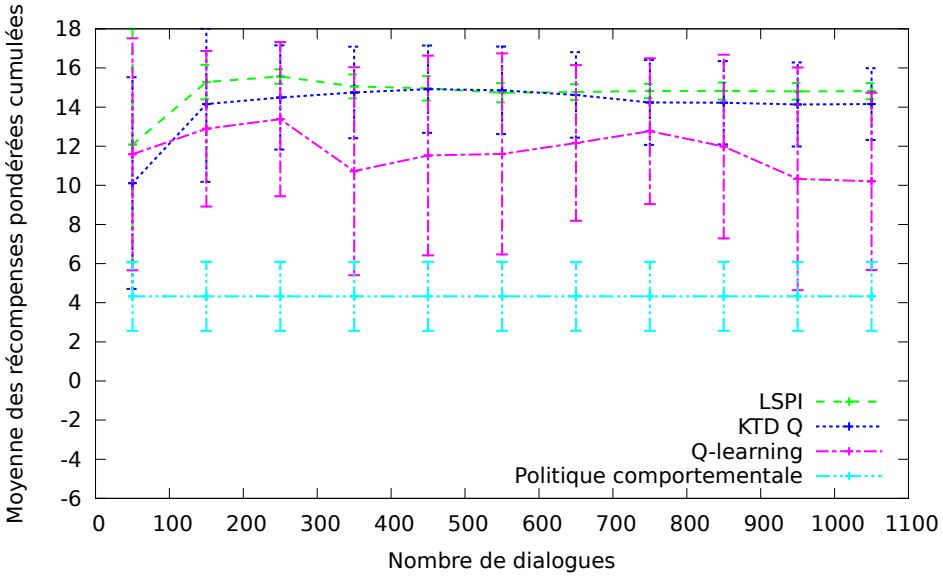



**Figure 5.7** – Comparaison des résultats pour une approche hors-ligne.

KTD-Q et LSPI donnent des scores quasiment identiques et élevés, plus que dans le cas on-policy et en-ligne, et ce, dès le début de l'apprentissage. L'approche avec Q-learning donne les résultats les moins bons. C'est une approche du premier ordre et donc moins efficace sur les échantillons, contrairement aux deux autres. A partir d'une politique comportementale ne donnant pas forcément des résultats satisfaisants, de bonnes solutions peuvent être apprises.

### Cas en-ligne

L'approche en-ligne et off-policy est présentée maintenant. Seuls KTD-Q et Q-learning sont comparés car la version standard de LSPI n'est prévue que pour le cas hors-ligne. La politique de contrôle utilisée dans ce cas est l'estimation courante de la fonction de qualité. Pour une politique de contrôle plus fiable, le schéma d'exploration bonus-glouton est utilisé avec KTD. Cette exploration n'est pas possible avec Q-learning car une information d'incertitude sur l'estimation n'est pas fournie. La moyenne des récompenses cumulées obtenues lors du test de la politique optimale ainsi que celle obtenue avec la politique de contrôle lors de l'apprentissage de la solution optimale sont présentées Figure 5.8. Dans le cas où les politique de contrôle sont tracées, chaque point est le résultat d'une moyenne faite à partir d'une fenêtre glissante d'une largeur de $100$ points. Ceci explique qu'un point soit manquant sur ces courbes par rapport à celles présentant la politique optimale.

Les politiques optimales renvoient de meilleurs résultats que les politiques de contrôle. Ces dernières sont le pendant des courbes présentées Figure 5.6 (cas KTD-SARSA avec le schéma bonus) pour le cas off-policy. En effet, elles montrent les scores obtenus lors de l'apprentissage de la politique optimale. En comparant la courbe KTD-SARSA (bonus) (Figure 5.6) et KTD-Q (contrôle) (Figure 5.8), il semblerait que l'apprentissage dans le cas off-policy soit légèrement plus rapide. Cela peut s'expliquer par le fait que c'est directement la solution optimale qui est recherchée et non la solution courante qui est améliorée.

Ainsi qu'attendu, KTD-Q donne des scores supérieurs à ceux de Q-learning. De plus, les résultats donnés par le test de la politique optimale apprise par KTD-Q sont moins bons au départ mais similaires lorsque le nombre de dialogues d'entraînement dépasse les $200$, par rapport

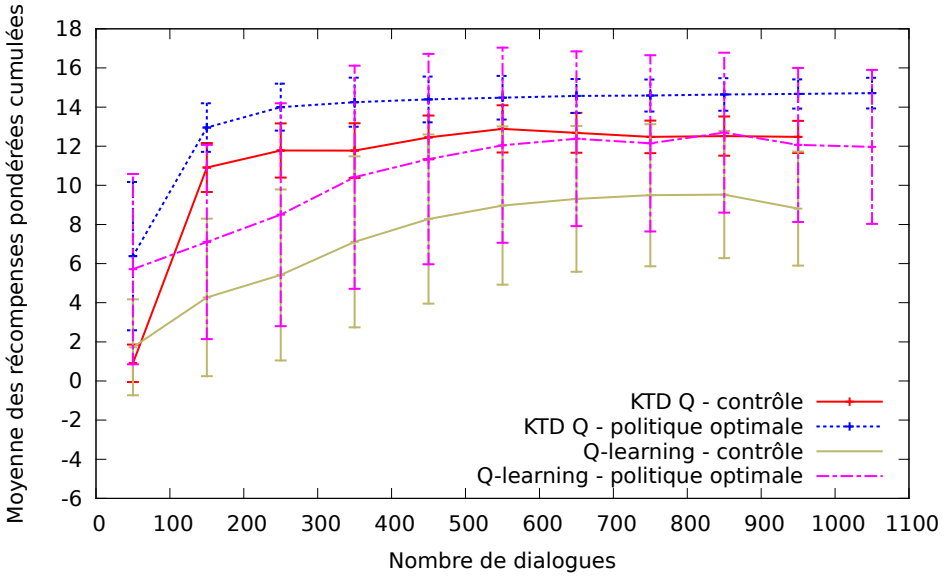



**Figure 5.8** – Comparaison de KTD et Q-learning dans le cadre d'une approche en-ligne et off-policy (cas linéaire).

à l'approche off-policy et hors-ligne. Cela peut s'expliquer par le fait que la politique comportementale retourne de moins bons scores pour des tailles de jeux d'entraînement petits puisqu'elle visite des endroits moins intéressants de l'espace d'état du fait de l'exploration. En revanche, ce qui est intéressant dans cette approche est que la politique comportementale s'améliore au fur et à mesure des interactions et est bien meilleure, donc moins gênante pour l'utilisateur que celle présentée Figure 5.7.

## 5.4 Test d'une représentation non-linéaire

Dans la section précédente, il a été montré qu'il est possible de jouer sur la phase d'exploration pour rendre l'apprentissage de la solution optimale plus rapide. Dans cette section, nous allons présenter l'utilisation d'une représentation non-linéaire de la fonction de valeur. A nombre de paramètres égaux, le pouvoir de représentation d'une paramétrisation non-linéaire pourrait être augmenté par rapport à celle linéaire. Ceci est d'importance car la rapidité de l'apprentissage est liée au nombre de paramètres à apprendre. Après une présentation de la paramétrisation utilisée ici, le test de cette paramétrisation à l'aide de KTD sur la tâche CamInfo est proposé.

### 5.4.1 Réseau de neurones artificiel

La représentation choisie est basée sur un réseau de neurones artificiel (*Artificial Neural network*, ANN en anglais). Elle est illustrée Figure 5.9. En entrée sont donnés l'état et l'action et la sortie retourne $Q(s, a)$. La couche d'entrée comporte $42$ neurones : les $6$ premières entrées sont utilisées pour représenter $s^1$, les $22$ suivantes pour $s^2$, puis les deux suivantes pour respectivement $s^3$ et $s^4$, et enfin les $12$ dernières pour l'action $a$. Les valeurs discrètes de l'espace d'état et de l'espace d'action sont représentées respectivement par autant de neurones qu'elles peuvent prendre de valeurs de façon à ce qu'une seule entrée soit active à la fois, étant donné qu'il n'y a pas de métrique naturelle sur ces composantes. Ces neurones ne prennent alors que

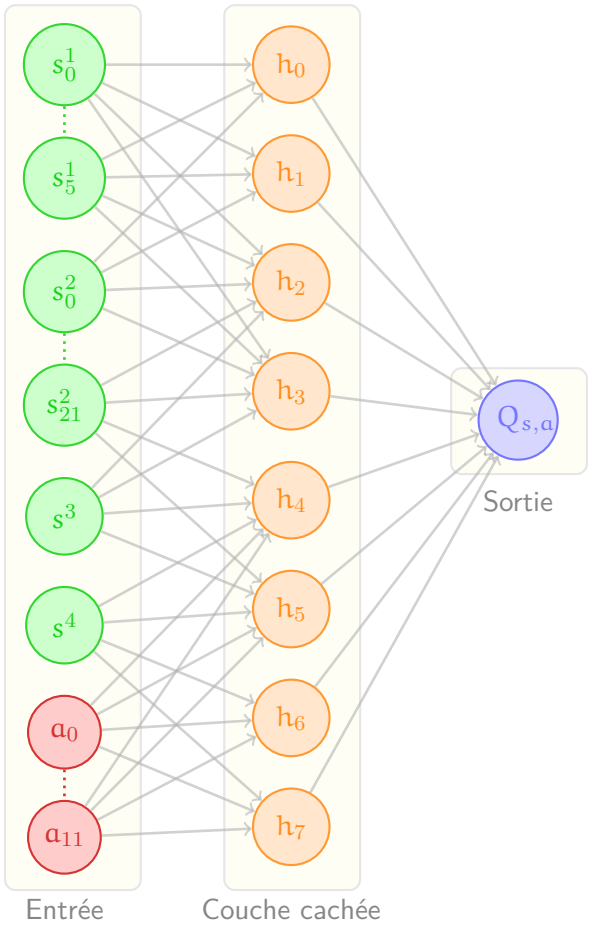



**Figure 5.9** – Illustration du réseau utilisé pour approximer la fonction de qualité. Pour des raisons de lisibilité, toutes les connections de la couche d'entrée vers la couche cachée n'apparaîssent pas.

deux valeurs possibles. La couche cachée comporte $8$ neurones avec une fonction d'activation *tangente hyperbolique*. Ces choix sont faits par le concepteur du réseau. La couche de sortie ne comporte qu'un seul neurone puisque le rôle du réseau est d'estimer la valeur de la fonction Q ($Q : S \times A \to \mathbb{R}$). La fonction d'activation de la couche de sortie est la fonction *identité*. Les paramètres de la représentation sont les poids des connections synaptiques entre les différentes couches. Il y a $42 \cdot 8$ connections entre la couche d'entrée et la couche cachée. Le nombre de connections entre la couche cachée et la couche de sortie est $(8 + 1) \cdot 1$. Un neurone dont la valeur est $1$ est toujours ajoutée dans l'utilisation d'un réseau de neurones. Le nombre total de paramètres est donc $42 \cdot 8 + 9 = 345$.

Soient $E \in \mathbb{M}_{42\times1}$ la matrice contenant les valeurs d'entrées du réseau, $H \in \mathbb{M}_{9\times1}$ la matrice contenant les valeurs des neurones cachés, $W_E \in \mathbb{M}_{8\times42}$ la matrice contenant les poids synaptiques de la couche d'entrée vers la couche cachée et $W_S \in \mathbb{M}_{9\times1}$ la matrice contenant

les poids synaptiques de la couche cachée vers la couche de sortie tels que :

$$E^T = \begin{bmatrix} s_0^1 \dots s_5^1 & s_0^2 \dots s_{21}^2 & s^3 & s^4 & a_0 \dots a_{11} \end{bmatrix} \tag{5.9}$$

$$H^T = \begin{bmatrix} h_0 \dots h_8 \end{bmatrix} \tag{5.10}$$

$$W_E = \begin{bmatrix} w_{0,0}^E & w_{0,1}^E & \dots & w_{0,41}^E \\ \vdots & \vdots & \ddots & \vdots \\ w_{7,0}^E & w_{7,0}^E & \dots & w_{7,41}^E \end{bmatrix} \tag{5.11}$$

$$W_S = \begin{bmatrix} 1 & w_0^S & \cdots & w_8^S \end{bmatrix}, \tag{5.12}$$

alors, pour un couple $(s, a)$ donné, représenté par la matrice $E_{s,a}$, la fonction de qualité est estimée de la façon suivante :

$$H = \tanh\left(W_E \cdot E\right) \tag{5.13}$$

$$\text{et donc } Q_{s,a} = W_S \cdot H. \tag{5.14}$$

### 5.4.2 Approche en-ligne et on-policy

Les résultats de l'optimisation de la politique avec une approche en-ligne et on-policy, avec une paramétrisation non-linaire de la fonction Q à l'aide de l'algorithme KTD-SARSA et d'une exploration bonus-greedy sont donnés Figure 5.10. La moyenne des récompenses cumulées obtenues lors de l'apprentissage de la politique est tracée en fonction de différentes tailles de jeux de données d'entraînement. La valeur moyenne ainsi que les résultats obtenus pour chaque essai ayant servi à la tracer apparaissent sur le graphique pour expliquer le comportement de la moyenne. Il apparaît ainsi que l'apprentissage à l'aide d'un réseau de neurones est binaire : soit il donne de bons résultats, soit il n'a pas abouti. De nombreux essais parviennent très rapidement à de hauts scores. En comparaison avec la Figure 5.6 qui reprend les mêmes conditions expérimentales (on-policy et en-ligne) avec une paramétrisation linéaire, dès $200$ dialogues, de nombreux apprentissages atteignent déjà le score de $14$ alors que ce score n'est quasiment jamais atteint dans le cas linéaire. En revanche, il serait intéressant de proposer une méthode pour détecter dès le début de l'apprentissage les essais qui n'aboutiront que tardivement ou jamais. Cela pourrait se faire en trouvant des conditions sur les paramètres du réseau et en proposant une initialisation plus fine que celle aléatoire actuelle.

### 5.4.3 Approches en-ligne et off-policy

Une approche en-ligne et off-policy est proposée, avec la paramétrisation non-linéaire définie ci-avant. Les résultats du test sont donnés Figure 5.11. Les conditions expérimentales sont les mêmes que celles ayant servi à générer la Figure 5.8. Les mêmes constatations à propos du déroulement de l'apprentissage se font que pour le cas non-linéaire on-policy (Figure 5.10) : la politique proposée est soit bonne soit mauvaise. Mais contrairement à ce cas, la politique apprise est en moyenne rapidement bonne (le score est de $12$ autours de $500$ dialogues d'entraînement) et hormis quelques essais donnant de mauvais résultats, la très grande majorité atteint un score de $15$ avec des jeux de $1000$ dialogues d'entraînement. Ces résultats sont d'autant meilleurs lorsque le constat est fait qu'une double non-linéarité est introduite dans ce cas. En effet, la première vient du fait que l'équation d'optimalité de Bellman est utilisée, avec son opérateur $\max$. La seconde provient de la paramétrisation.

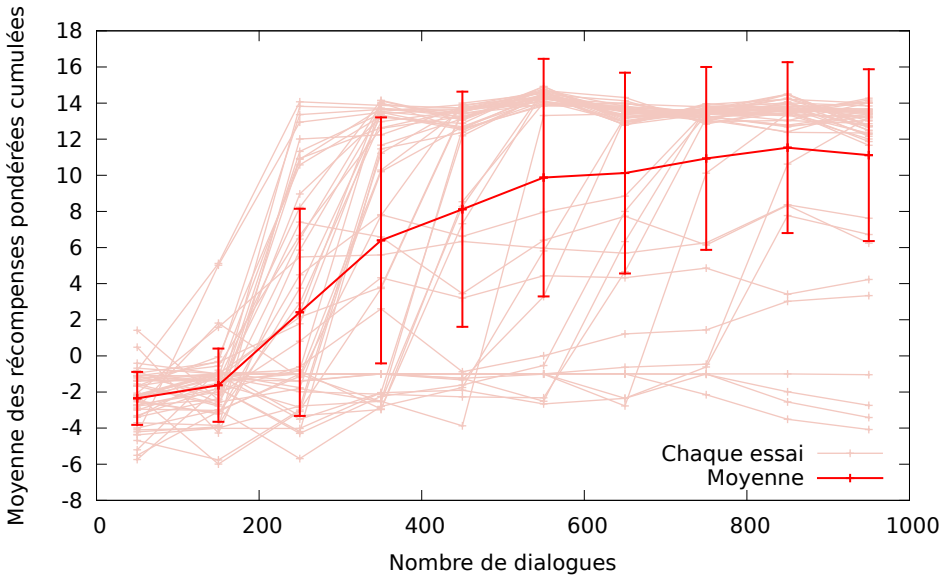



**Figure 5.10** – Approches en-ligne et on-policy avec KTD-SARSA (cas non-linéaire).

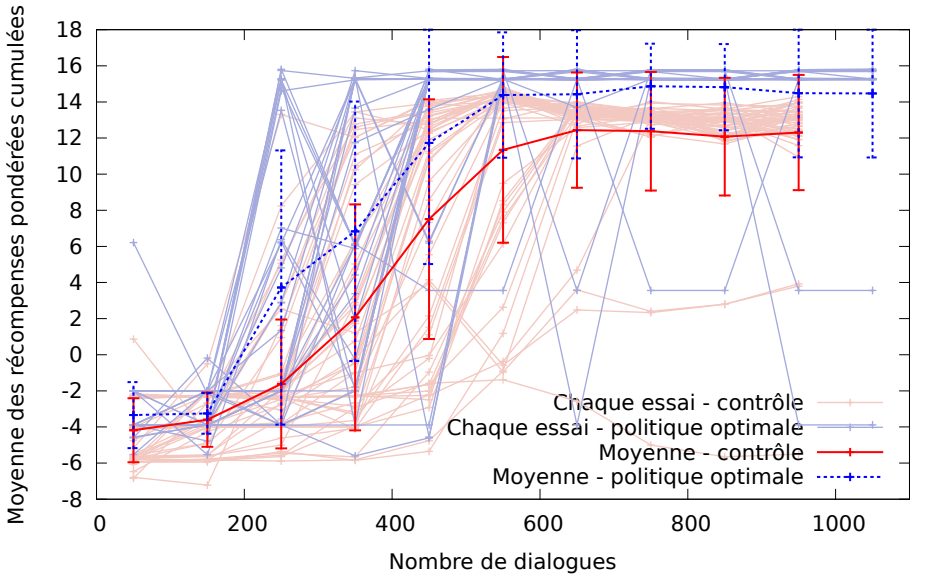



**Figure 5.11** – Approches en-ligne et off-policy avec KTD-Q (cas non-linéaire).

## 5.5 Conclusion

Un tableau récapitulatif des propriétés des différentes approches est proposé Figure 5.12 [1]. Il apparaît que KTD présente toutes les caractéristiques requises, faisant de lui un algorithme adapté à la résolution du problème de dialogue. En pratique, cela est conforté par diverses comparaisons des algorithmes de référence. Etant donné que ces expériences n'ont pas été réalisées avec des données réelles, les résultats ont été croisés avec ceux obtenus sur un autre système de dialogue, DIPPER [Bos 03]. D'une part, les simulations d'utilisateurs sont différentes : dans un cas, elle est basée sur un agenda et dans l'autre sur un N-gramme. D'autre part, les espaces d'états sont différents. Malgré ces changements, les conclusions concernant KTD sont identiques dans les deux cas [Daubigney 12a]. Cela montre la capacité de KTD à s'adapter à des conditions expérimentales différentes. Proposer une solution de qualité sur des utilisateurs réels, changement qui concerne les conditions d'apprentissage, semble alors à portée de cet algorithme.

En ce qui concerne l'exploration, aucune des deux approches, l'approche on-policy ou off-policy ne peut être privilégiée. Cela dépend de ce que souhaite le concepteur du système. S'il préfère que le système continue à s'adapater perpétuellement aux utilisateurs, il mettra en place une approche on-policy et en-ligne. Si au contraire, il souhaite ménager ses utilisateurs pour que ce derniers ne rencontrent pas de décisions malencontreuses au début de l'apprentissage, il préfèrera une approche off-policy, avec l'option hors-ligne ou en-ligne au choix. L'option hors-ligne est intéressante si le concepteur du système souhaite réutiliser des données déjà collectées. Celle en-ligne l'est car il y a possibilité d'améliorer la politique comportementale. Dans le cas off-policy, l'apprentissage est censé être plus rapide que dans le cas on-policy car c'est directement la solution optimale qui est recherchée et non la solution courante qui est améliorée.

En ce qui concerne la rapidité de l'apprentissage, une autre donnée peut entrer en jeu. Il s'agit de la paramétrisation non-linéaire. En comparaison avec une paramétrisation linéaire, le même pouvoir de représentation peut être obtenu avec un nombre de paramètres moindre. Ceci permet le passage à l'échelle. Cette question est abordée au chapitre suivant.

Ce chapitre a eu pour but d'illustrer la première hypothèse faite en introduction qui suppose que l'amélioration des interactions homme-machine repose sur le fait que les solutions apportées doivent nécessiter le moins de connaissances expertes *a priori*. Il a en effet été montré qu'il est possible, par un choix judicieux d'algorithme, d'apprendre à partir d'interactions effectuées avec des utilisateurs réels, ce qui permet de ne pas introduire de modèle d'utilisateurs pour générer suffisamment de données et limiter ainsi l'introduction de biais supplémentaire.

1. La non-stationnarité n'est pas gérée par SARSA et Q-learning car le taux d'apprentissage $\alpha$ est considéré comme étant constant.

| | KTD-SARSA | KTD-Q | GPTD | SARSA | Q-learning | LSPI |
|---|---|---|---|---|---|---|
| En-ligne & on-policy | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ |
| En-ligne & off-policy | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ |
| Hors-ligne | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ |
| Exploration efficace | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| Param. non-linéaire | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |
| $2^{nd}$ ordre | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ |
| Gestion non-station. | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |

**Figure 5.12** – Récapitulatif des algorithmes testés et de leurs propriétés.

# Chapitre 6

# Gestion de l'observabilité partielle

Dans le chapitre prédédent, un algorithme de RL permettant de prendre en considération les différents aspects nécessaires au bon fonctionnement d'un gestionnaire de dialogue a été présenté. La question de la gestion de l'observabilité partielle n'a pas directement été abordée. Il a été supposé que la représentation des informations concernant l'utilisateur pour le système de dialogue était déjà mise à disposition au travers du paradigme HIS (Section 5.1). Les approches présentées dans ce chaptire proposent de deux manières différentes d'améliorer la prise en charge de l'observabilité partielle.

La première vise à d'augmenter la quantité d'information mise à disposition du gestionnaire de dialogue sur l'utilisateur, ceci dans le but de diminuer l'incertitude sur les informations fournies. A chaque tour de dialogue, il est attendu que les décisions soient plus efficaces. Mais qui dit augmentation de la taille de l'espace d'état dit augmentation du nombre de paramètres représentant la fonction de qualité. Une prise en charge d'une telle augmentation peut passer par une paramétrisation non-linéaire. La section suivante de ce chapitre présente un test de cette approche basée sur la tâche CamInfo.

La deuxième approche consiste à utiliser une méthode de recherche directe de la politique sans passer par la fonction de valeur (*Direct Policy Search*, DPS en anglais) grâce à une optimisation en boîte noire (*Black Box Optimisation*, BBO en anglais). Dans ce cadre particulier, même si l'hypothèse de Markov n'est pas vérifiée, cette méthode permet d'estimer une bonne politique relativement aux données fournies, appelée politique *réactive*. Ainsi, un état pour lequel des garanties ne peuvent pas être apportées quant au respect strict de la propriété de Markov, peut être utilisé de façon plus sûre. Ceci n'est pas le cas avec le RL dans le cadre classique pour lequel les équations de Bellman ne sont pas vérifiées si l'hypothèse de Markov n'est pas respectée. Cette approche sera développée dans la deuxième section de ce chapitre.

## 6.1 Amélioration de la prise en charge de l'information

### 6.1.1 Motivation

Dans le cadre d'une interaction homme-machine, la définition d'une politique doit tenir compte du fait que les informations concernant l'utilisateur ne sont pas parfaitement observables. D'une part ses intentions ne sont pas disponibles dès le début de l'interaction et d'autre part, la reconnaissance vocale et l'analyseur sémantique ne sont pas capables d'identifier parfaitement ce qui a été dit. Malgré ces incertitudes, le gestionnaire de dialogue doit être capable de proposer des décisions cohérentes. Pour cela, le concepteur du système doit récolter le maximum

d'information à chaque tour de dialogue et les stocker de façon compacte pour une utilisation efficace au moyen d'un état de croyance. Dans le cas du dialogue, il peut aussi se baser sur des modèles fiables issus de la théorie du langage. Cependant, plus la quantité d'information est importante, plus l'espace d'état est grand aussi. Cela implique donc de trouver un compromis entre la taille de cet espace et celle de la représentation de la fonction de valeur. Si le nombre de paramètres est trop important à apprendre, les données d'apprentissage deviendront impossibles à réunir et la solution apprise ne sera pas de bonne qualité. La gestion de ce compromis peut par exemple se faire en utilisant une représentation non-linéaire de la fonction de qualité. Cela permet d'augmenter la richesse de l'espace d'état tout en gardant un nombre de paramètres raisonnable.

### 6.1.2 Test de l'approche

L'augmentation de la quantité d'information utilisée pour prendre la décision courante a été testée sur le tâche CamInfo présenté Section 5.1. Les probabilités correspondant aux deux hypothèses les plus probables sont utilisées. Nous proposons ici d'augmenter la richesse de l'information en ajoutant à l'état existant la probabilité que la troisième hypothèse soit juste. La représentation de la fonction de qualité en est ainsi modifiée : il y a une dimension de plus à prendre en compte. Avec une représentation non-linéaire tel un réseau de neurones, cette augmentation reste facile à prendre en compte car il suffit d'ajouter une entrée supplémentaire. Le nouvel MDP associé est résumé Figure 6.1.

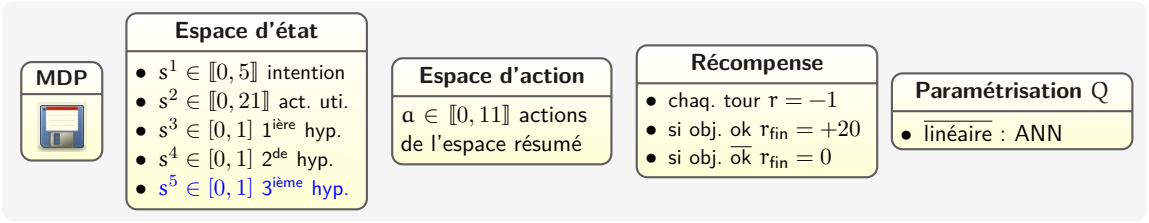



**Figure 6.1** – Modélisation du dialogue avec HIS.

En reprenant le réseau de neurones défini Section 5.4.1, en ajoutant une entrée, le nombre de paramètres devient $(42+1)\cdot 8+9 = 353$. Ce qui revient à une augmentation d'environ $3\%$ du nombre de paramètres seulement, en prenant en compte une dimension en plus pour construire l'état. Si une répréstentation linéaire était choisie, l'augmentation du nombre de paramètres serait beaucoup plus significative. En effet, si une dimension continue était ajoutée à la paramétrisation linéaire décrite Section 5.3.1, il faudrait ajouter trois gaussiennes supplémentaires pour cette dimension. Ainsi, $3^3$ gaussiennes en trois dimensions et une constante seraient nécessaires pour décrire l'espace décrit par les trois probabilités des meilleures hypothèses. A cela, il faut ajouter les paramètres nécessaires pour décrire les deux autres dimensions discrètes. Le nouveau nombre de paramètres dans ce cas serait : $(1+3^3+22+6)\cdot 12 = 672$ (contre 456). Ce qui reviendrait à une augmentation d'environ $32\%$.

Les résultats de cette approche sont présentés Figure 6.2. Ils ont donné lieu aux publications [Daubigney 12c, Daubigney 12b]. La moyenne des récompenses pondérées cumulées obtenue lors du test de la politique optimale dans un schéma en-ligne et off-policy pour KTD-Q est tracée en fonction de différentes tailles de jeu de données d'entraînement. La politique de contrôle utilisée est l'estimation courante de celle apprise pour laquelle une exploration bonus-glouton est implémentée. Le test de la nouvelle représentation de l'état se fait avec le réseau de neurones décrit Section 5.4.1 auquel une entrée supplémentaire a été ajoutée. En comparant les

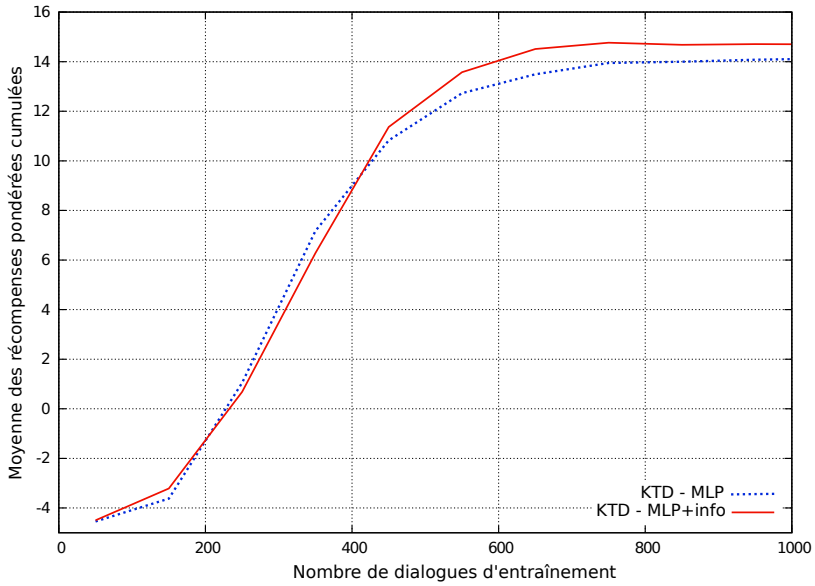



**Figure 6.2** – Comparaison des politiques optimales données par KTD lorsque l'espace d'état est augmenté d'une composante.

deux résultats de KTD-Q, la politique apprise avec un espace d'état augmenté semble meilleure pour les nombres de données supérieurs à $500$ dialogues puisque le dialogue est écourté d'un tour en moyenne. Plus d'informations concernant l'état du dialogue, réduisant ainsi l'incertitude quant à l'observation, permettent alors de rendre la solution trouvée plus efficace.

Une autre façon de gérer l'observabilité partielle basée sur une méthode de recherche directe de la politique est présentée maintenant.

## 6.2 Recherche directe dans l'espace des politiques

Une autre façon de résoudre le problème de l'apprentissage par renforcement est non plus de passer par les équations de Bellman mais de rechercher la solution directement dans l'espace des politiques. Une des méthodes pour ce faire est l'optimisation en boîte noire (*Black Box Optimisation*, BBO en anglais). La contribution de cette section concerne donc la relaxe de la contrainte du cadre markovien. Elle consiste à d'affranchir de la modélisation de la prise de décision. Ce principe est illustré Figure6.3. Les solutions potentielles au problème sont définies de façon paramétrique. La BBO cherche celle optimale en testant plusieurs candidates $\theta_i$ de l'espace des possibles. La qualité de chaque solution est quantifiée par une fonction de score, $J$. La meilleure solution est celle qui maximise ce critère. Ces approches sont intéressantes car elles permettent de trouver une bonne solution relativement aux données, ce qui n'est pas le cas du RL classique lorsque les données ne vérifient pas l'hypothèse de Markov. Cette approche permet donc de gérer différemment la question de l'observabilité partielle. La méthode retenue est tout d'abord présentée à la section suivante. Une application à la tâche CamInfo est proposée ensuite. Ces travaux ont été présentés dans [Daubigney 13c, Daubigney 13b].

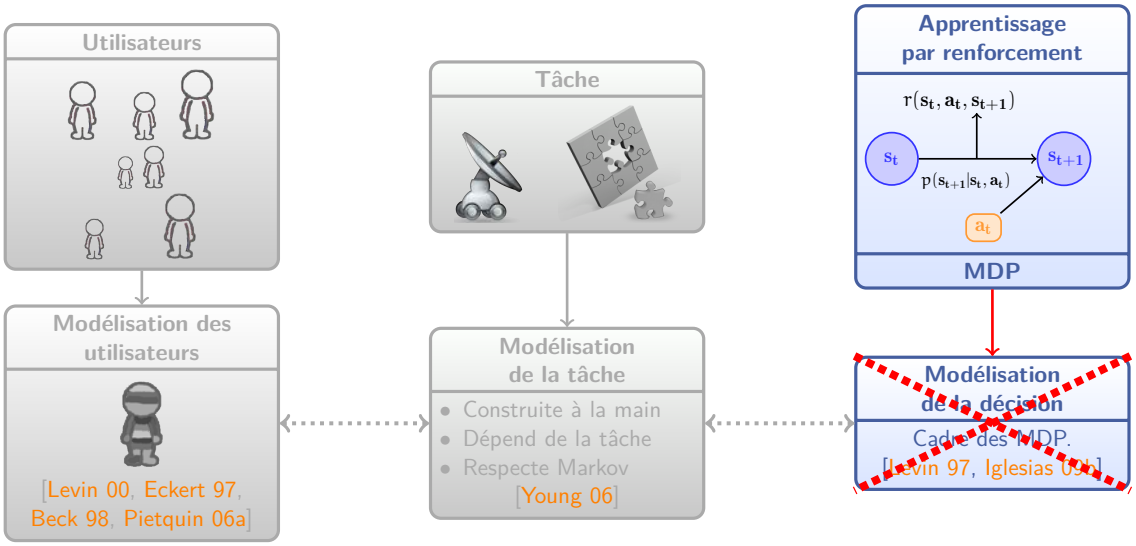



**Figure 6.3** – La seconde contribution se focalise sur la modélisation de la prise de décision. La contrainte du cadre markovien est relaxée.

### 6.2.1 Une approche particulaire

L'optimisation par essaim particulaire (*Particle Swarm Optimisation*, PSO en anglais) est la méthode retenue pour gérer de façon plus sûre le problème de l'observabilité partielle dans le cas de l'optimisation du problème de dialogue. Pour la preuve de concept présentée dans ces travaux, un algorithme *pure vanilla* a été retenu. Le futur de ces travaux pourrait consister à comparer d'autres approches en boîte noire. La PSO est inspirée des méthodes visant à modéliser le comportement global d'un banc de poissons ou d'oiseaux. C'est un algorithme qui cherche à reproduire le comportement de l'essaim à partir de règles simples fixées pour chacun des éléments le composant, appelé *particule*. Ces règles concernent la vitesse et la position par rapport aux particules voisines. Chaque mouvement d'une particule est répercuté à ses voisines selon les règles définies par le concepteur, qui vont à leur tour mettre à jour leurs position et vitesse. Le mouvement d'une particule est transmis ainsi à tout l'essaim.

Lors de l'utilisation de ces modélisations, les concepteurs se sont aperçus qu'il était alors possible d'effectuer une optimisation pour autant qu'un score soit associé à chaque membre de l'essaim (le score étant fixé par le concepteur en fonction de la tâche qu'il veut voir optimisée). Chaque membre est alors considéré comme une solution candidate au problème. Trouver la meilleure solution revient donc à trouver le membre présentant le meilleur score. Ceci est effectué itérativement, en sélectionnant à chaque tour le ou les membre(s) présentant le meilleur score et en déplaçant l'essaim vers ces membres sélectionnés. Cette approche a pour la première fois été utilisée dans [Kennedy 95] puis améliorée dans [Engelbrecht 05]. Elle a été testée sur des problèmes de référence en RL dans [Fix 12].

La PSO peut s'appliquer au problème de recherche directe de la politique. Cette dernière est dans ce cas définie par une paramétrisation, $\pi = \pi_\theta$. L'espace des solutions est engendré par l'ensemble des politiques possibles selon la paramétrisation choisie. Chaque particule de l'essaim correspond à une politique différente et donc instancie un contrôleur. La PSO estime le jeu de paramètres optimal en testant itérativement un nombre constant, fixé par le concepteur du système, de différentes solutions candidates de l'espace des solutions. Une illustration de la

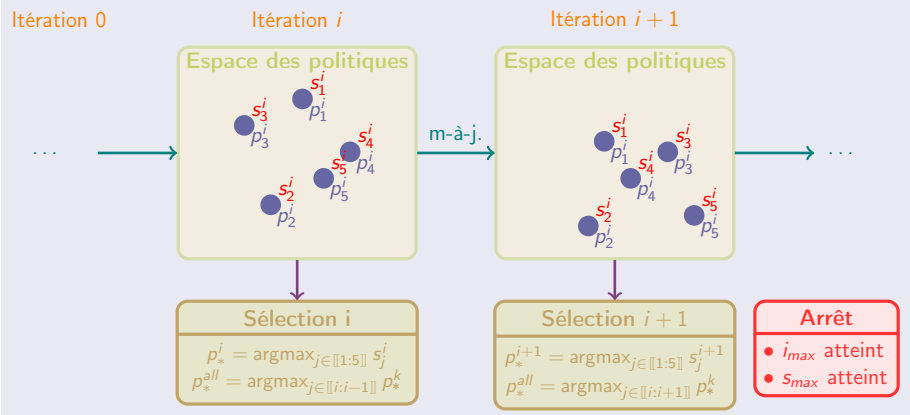



**Figure 6.4** – Illustration de la PSO. La valeur $s_k^i$ représente le score de la particule $k$ à l'itération $i$.

méthode est proposée Figure 6.4.

La seule information dont a besoin la PSO pour l'optimisation est une évaluation de la qualité de chacune des solutions envisagées. Cette information est donnée au moyen d'une fonction de score, $J$. Elle est définie de l'espace des paramètres ($\theta \in \mathbb{R}^p$, $p$ étant le nombre de paramètres) dans l'espace des réels, $J : \mathbb{R}^p \to \mathbb{R}$. Par cette information, les meilleures solutions de l'itération courante sont retenues pour la suivante. Le type de sélection dépend de l'algorithme choisi. Les itérations sont répétées jusqu'à ce qu'un critère global soit vérifié. Ainsi, la meilleure solution $\pi^*$ est définie par le jeu de paramètres $\theta^*$ tels que le score soit maximisé :

$$\theta^* = \underset{\theta\in\mathbb{R}^N}{\text{argmax}}\, J(\theta). \tag{6.1}$$

Cette méthode est avantageuse car peu d'*a priori* sont nécessaires pour la voir fonctionner. De plus, elle présente un caractère parallèle qui est intéressant dans le cadre du dialogue. En effet, à chaque itération, un certain nombre de solutions candidates peut être testé en même temps sur les utilisateurs qui interagissent avec la machine simultanément.

### 6.2.2 Test de cette approche sur CamInfo

La BBO est testée sur le système de dialogue présenté Section 5.1. Nous rappelons que le paradigme HIS construit un espace d'état à partir duquel l'optimisation se fait. Comme les données sur l'utilisateur sont partielles, un modèle d'observation basée sur la théorie du langage est construit pour obtenir et mémoriser le maximum d'information lors de l'interaction. Cependant, aucune garantie stricte n'est fournie quant au fait que le nouvel état soit markovien. Cela justifie l'utilisation de la BBO, solution qui fournit la meilleure politique réactive par rapport aux données fournies, indépendamment du fait que l'hypothèse de Markov soit vérifiée. Un rappel de la configuration de la solution envisagée est proposée Figure 6.5. Après la définition des conditions expérimentales, les résultats du test de cette approche sont présentés.

#### Conditions expérimentales

**L'essaim** L'essaim particulaire utilisé dans ces expériences est celui implémenté dans [Clerc 11]. Il est construit avec $N_{PART}$ particules. Ces particules sont réparties dans l'espace des paramètres

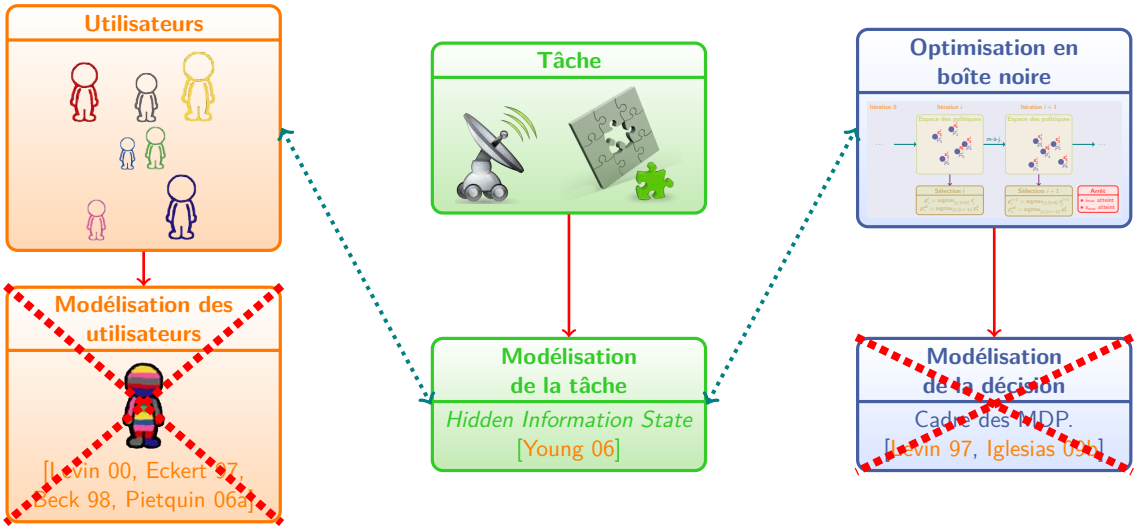



**Figure 6.5** – Solution envisagée : utilisation de la PSO pour de s'affranchir du cadre markovien. La modélisation de la tâche est maintenue avec HIS.

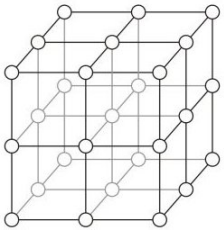


**Figure 6.6** – Illustration de la topologie de Von Neumann pour un espace en 3 dimensions et un essaim de 27 particules.

selon une topologie de Von Neumann. Une illustration de cette topologie est donnée Figure 6.6 [1].

Les règles de mise à jour de la vitesse $v^j$ et la position $p^j$ de la particule $j$ au pas de temps $i$ sont les suivantes :

$$v^j_{i+1} = w \cdot v^j_i + c \cdot r_1 \cdot (b^j - l^j) + c \cdot r_2 \cdot (l^j p^j) \quad (6.2)$$

$$p^j_{i+1} = p^j_i + v^j_{i+1}, \quad (6.3)$$

avec $w$, $c$, $r_1$ et $r_2$ des paramètres constants et $b^j$ la position de la meilleure particule trouvée jusqu'alors et $l^j$ la position de la meilleure particule au voisinage de la particule $j$. Les positions des particules sont initialisées aléatoirement dans l'espace des politiques et les vitesses sont initialisées à zéro.

**La fonction de score** Pour utiliser la PSO, il faut ensuite définir le critère à optimiser. Il est en relation avec la qualité d'un dialogue. Par example, il peut être lié à la récompense cumulée, ou, dans le cas d'interactions réelles, être un score subjectif donné par l'utilisateur à l'issue de

1. Illustration trouvée dans l'article Multi-ring particle swarm optimization, de Bastos-Filho, C., Caraciolo, M., Miranda, P. and Carvalho, D. publié dans les actes du Symposium brésilien sur les réseaux de neurones (2008).

l'interaction. Le critère suivant est utilisé ici :

$$J^{\pi} = \mathrm{E}\left[20 \cdot \delta_{\text{accompli}} - N_{\text{tours}}\right], \tag{6.4}$$

avec $\delta_{\text{accompli}}$ égal à $1$ si la tâche a été accomplie par le gestionnaire à l'issue du dialogue, $0$ sinon et $N_{\text{tours}}$ étant la longueur du dialogue. La politique est paramétrée en passant par la fonction de valeur. Cette dernière est paramétrée linéairement de la même façon que décrit Section 5.3.1. La politique estimée, associée à chaque particule $j$ est la suivante. Elle est inspirée de la paramétrisation linéaire de la fonction de qualité présentée Section 3.2.1 :

$$\hat{\pi}_{\theta_j}(s) = \underset{a \in A}{\operatorname{argmax}}\, \theta^{T} \Phi(s, a). \tag{6.5}$$

Cependant, le calcul exact de la valeur du critère à chaque itération est impossible du fait de l'espérance qui apparaît dans l'Equation 6.4. Seule une approximation est envisageable, au moyen d'un échantillonnage de Monte-Carlo, connu pour être un estimateur non-biaisé de l'espérance. Il consiste à tester sur un utilisateur le contrôleur instancié par une particule et de calculer le résultat du test. Plusieurs tests peuvent être effectués sur une même particule pour avoir une meilleure estimation de sa valeur, chacun d'eux étant effectué avec un utilisateur différent. Pour chaque particule $j$ de l'essaim, $N_{MC}$ essais d'une durée de $T_i^j$ interactions sont effectués avec la politique paramétrée par $\theta_j$. L'estimation du score est donc la suivante :

$$\hat{J}(\theta_j) = \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} (20 \cdot \delta_{\text{accompli}}^{i,j} - T_i^j).$$

L'architecture de la PSO convient au problème de dialogue de par son aspect parallèle, ce qui constitue un avantage par rapport aux autres méthodes proposées jusqu'alors. Par exemple, si plusieurs utilisateurs appellent en même temps le standard, chaque interaction peut être mise à profit pour tester une nouvelle politique. En revanche, KTD ne sait pas gérer les informations qui lui parviennent en même temps. La mise à jour n'utilise qu'une transition à la fois. Par ailleurs, le choix peut être fait de laisser seulement quelques utilisateurs interagir avec les différentes solutions candidates tandis que les autres interagissent avec la meilleure politique apprise jusqu'alors. Même si cela se fait au prix d'un apprentissage plus lent, moins d'utilisateurs doivent alors faire face à des comportements sous-optimaux.

Cette approche se déroule en-ligne car les données ne sont pas collectées par avance et les politiques à tester sont directement présentées aux utilisateurs. La phase d'exploration dans ce cas, comparée à celle effectuée dans le cadre du RL classique, n'intervient non pas au niveau de l'action choisie par le gestionnaire mais concerne l'ensemble des décisions. En effet, les variations sont faites directement sur les politiques. Cela a pour avantage de présenter des solutions qui, à défaut d'être optimales, sont au moins cohérentes sur toute la durée de l'interaction.

### 6.2.3 Test de la PSO

L'optimisation a été testée sur la tâche CamInfo dans les mêmes conditions expérimentales que précédemment (Section 5.1). Le critère J obtenu lors de l'apprentissage a été tracé en fonction du nombre d'itérations en faisant varier différents paramètres.

#### Nombre d'échantillonnages de Monte-Carlo

L'influence du nombre d'échantillonnages de Monte-Carlo est présentée Figure 6.7. Les résultats présentent les scores obtenus par la *meilleure particule* courante. Plus le nombre de

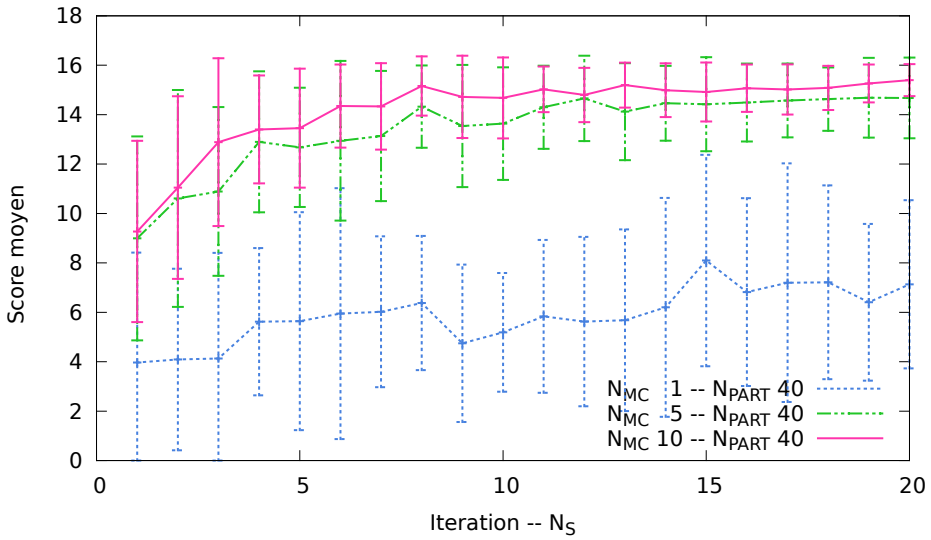



**Figure 6.7** – Influence du nombre d'échantillons à nombre de particules fixé.

tests des politiques ($N_{MC}$) à chaque itération est grand, meilleur est l'apprentissage. Au-delà d'un certain seuil, il ne semble plus nécessaire de l'augmenter : il y a beaucoup plus d'écart entre la courbe utilisant un échantillon et celle en utilisant cinq, qu'entre celle qui en utilise cinq et celle qui en utilise dix. Etant donné que dans le cadre du dialogue, les scores peuvent être très différents d'un essai sur l'autre, refaire plusieurs fois le test a son intérêt. Les résultats globaux avec $N_{MC} = 10$ sont légèrements meilleurs en moyenne que ceux retournés par KTD-SARSA dans un même cadre, en-ligne et on-policy : les scores obtenus sont d'un peu moins de $15$ contre un peu plus de $13$ pour KTD-SARSA (Figure 5.6). Cette différence peut s'expliquer par le fait que le nombre de dialogues d'entraînement est supérieur avec la recherche directe de politique. En effet, la courbe avec un échantillonnage de Monte-Carlo de dix se stabilise aux alentours de $N_s = 10$ itérations, ce qui donne un nombre de dialogues d'entraînement égal à $N_s \cdot N_{PART} \cdot N_{MC} = 10 \cdot 40 \cdot 10$ soit $4000$ dialogues. Ce qui est intéressant est que les variances restent aussi faibles quand le nombre d'essais de Monte-Carlo est raisonnable (supérieur à $5$).

### Nombre de particules

L'influence du nombre de particules de l'essaim est maintenant présentée. Le nombre de tests de Monte-Carlo a été fixé à dix, étant donné que cette valeur a donné les meilleurs résultats lors de l'expérience précédente (Figure 6.7). Le score moyen obtenu lors du test de la meilleure particule au cours de l'apprentissage est présenté Figure 6.8. Plus le nombre de particules de l'essaim est grand, plus l'apprentissage est rapide. En effet, plus il y a de particules, plus l'exploration de l'espace des solutions est important et donc plus la solution optimale est susceptible d'être trouvée. Passé un certain seuil, l'augmentation de la taille de l'essaim n'apporte pas d'amélioration supplémentaire : les scores obtenus sont similaires avec $25$ ou $40$ particules. En conséquence, le nombre de dialogues d'entraînement est réduit d'un peu moins de la moitié. A $N_S = 10$ itérations, au moment où les scores se stabilisent, le nombre de dialogues est $N_s \cdot N_{PART} \cdot N_{MC} = 10 \cdot 25 \cdot 10$ soit $2500$ dialogues. Cependant les résultats sont déjà très bons à partir de $N_S = 5$ itérations, soit aux alentours de $1200$ dialogues. Les mêmes ordres de grandeur qu'avec KTD-SARSA se retrouvent ce qui permet de confirmer que l'espace d'état construit par le paradigme HIS respecte la propriété de Markov.

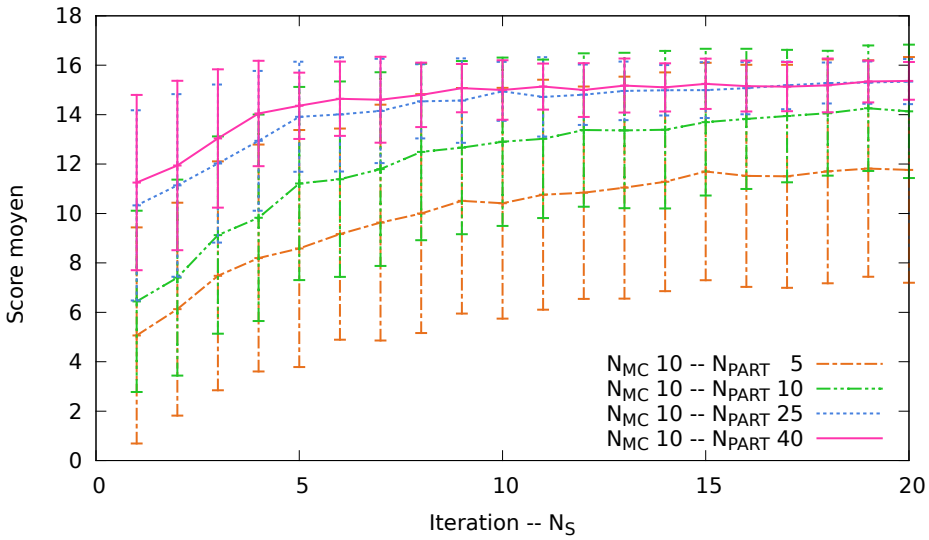



**Figure 6.8** – Influence du nombre de particules à nombre d'échantillons fixé.

### Comportement moyen

Les derniers résultats présentés exposent le score moyen pour l'ensemble des particules de l'essaim. Cette donnée est importante pour voir si le comportement global est correct, étant donné que ce sont les politiques effectives qui vont être testées sur les utilisateurs au cours de l'apprentissage. Les résultats sont donnés Figure 6.9 pour un nombre de particules fixe et un nombre d'échantillons de Monte-Carlo qui varie. Cette figure montre que le comportement des particules n'est pas homogène : il y a une différence entre les scores de la meilleure particule et de la moyenne. Cela est dû au fait que la nécessaire phase d'exploration teste parfois des solutions qui ne sont pas tout à fait satisfaisantes. Pour ne pas gêner trop d'utilisateurs à la fois, il serait envisageable de présenter à la majorité des utilisateurs la solution de la meilleure particule courante (qui est assez vite trouvée) et seulement de tester des nouvelles solutions sur quelques uns.

Malgré le changement total de paradigme, en passant par la recherche de la politique directement dans l'espace des politiques possibles, les résultats sur le problème de dialogue parlé sont prometteurs. La méthode choisie est une approche particulaire. Le principal attrait de cette approche réside dans le fait qu'elle retourne la meilleure politique réactive par rapport aux données fournies, ce qui n'est pas le cas lorsque les équations de Bellman sont utilisées alors que les données ne vérifient pas l'hypothèse de Markov. Cela apporte une garantie supplémentaire lorsque le concepteur du sytème n'est pas sûr que l'état construit vérifie cette propriété. L'autre avantage de cette méthode est son aspect parallèle qui convient particulièrement au dialogue étant donné que les centres d'appels reçoivent de nombreux appels en même temps.

## 6.3 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la gestion de l'observabilité partielle au travers de deux approches. La première consiste à jouer sur la représentation de la fonction de valeur afin de pouvoir prendre en compte plus d'information sur le contexte du dialogue. La décision courante est ainsi plus efficace. Cela est facilité en utilisant une représentation non-

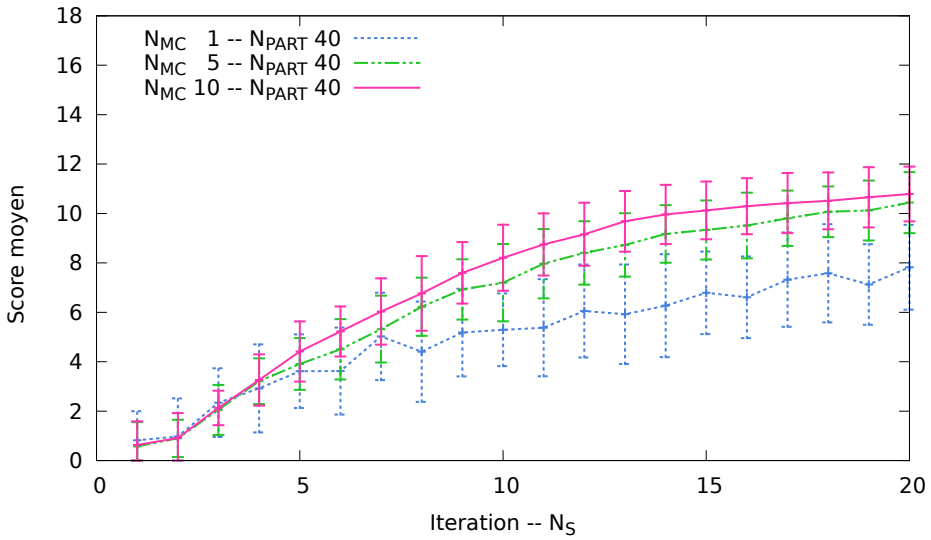



**Figure 6.9** – Score moyen obtenu pour l'ensemble des particules durant l'apprentissage.

linéaire de la fonction de valeur. Cette partie propose une vérification de la seconde hypothèse présentée en introduction. L'utilisation du paradigme des POMDP impose d'effectuer l'apprentissage, non pas sur des états, mais sur des distributions sur les états, ce qui implique la gestion d'un plus grand nombre de paramètres et nécessite un passage à l'échelle. Une solution pour passer à l'échelle a été proposée dans ce chapitre.

La deuxième manière de gérer l'aspect partiellement observable est de se placer directement dans l'espace des politiques possibles pour y chercher la politique optimale. Grâce à une optimisation en boîte noire, la solution retournée est la meilleure politique réactive par rapport aux données fournies. Cette solution est proposée pour répondre à la troisième hypothèse faite dans l'introduction qui suppose qu'il serait préférable de s'affranchir du cadre markovien. En effet, celui-ci n'est pas assuré lorsque des approximations de l'état de croyance sont faites. Pour améliorer cette preuve de concept, différents algorithmes de BBO pourraient être testés ainsi que d'autres paramétrisations de la fonction de qualité. De plus, il serait intéressant de se baser directement sur les informations collectées de l'utilisateur sans passer par le paradigme HIS en utilisant une représentation compacte de l'historique. En effet, dans des cas comme celui du tutorat, un tel paradigme n'est pas facile à constuire. Cette question est abordée au chapitre suivant.

# Troisième partie

# Environnements Informatiques pour l'Apprentissage Humain

# Chapitre 7

# Présentation des EIAH

Plusieurs études s'accordent à dire que l'efficacité de l'apprentissage pour un étudiant dépend fortement du ratio enseignant/élèves [Bloom 68, Bloom 84]. La progression est la plus nette lorsqu'un enseignant consacre toute son attention à un seul élève. Puis, plus la taille du groupe augmente, moins l'efficacité est marquée. Ceci peut s'expliquer par le fait que dans une relation 1-1, l'étudiant bénéficie de conseils personnalisés et non prévus pour un niveau moyen. L'enseignant suit pas à pas sa progression et adapte les conseils et la manière de faire à son tempérament. Cependant, pour des raisons de coûts et pour offrir à tous une éducation de bonne qualité, il n'est pas envisageable de mettre cette approche en œuvre à grande échelle. Ainsi est apparue l'idée au début des années 1990, grâce à la démocratisation de l'ordinateur pour les particuliers, de concevoir des tuteurs informatiques automatiques (*Intelligent Tutoring Systems*, ITS en anglais) pour faire profiter le plus grand nombre des avantages d'un enseignement personnalisé. Nous rappelons que nous avons vu en introduction Section 1.1.2 que le problème de tutorat s'apparente à un problème de prise de décisions séquentielles sous incertitude à résoudre.

Au cours des deux dernières décennies, ces tuteurs ont été expérimentés dans divers domaines de l'éducation. Un des premiers tuteurs a été developpé pour expliquer le langage de programmation LISP [Anderson 95]. Son utilisation a ensuite été étendue à l'enseignement de la géométrie et de l'algèbre. D'autres ont été developpés dans le domaine des mathématiques [Koedinger 97], de la physique [Vanlehn 05, Litman 04, Graesser 05], de l'informatique [Corbett 95], de la lecture [Mostow 01] ou encore des langues étrangères [Heift 07, Amaral 11]

Pour que ces systèmes fournissent une formation de qualité, il est nécessaire de prendre en compte les spécificités de la relation enseignant/apprenant. Le tuteur ou l'enseignant est celui en charge de proposer des activités pédagogiques au travers de leçons théoriques, d'exercices, d'applications pratiques, de questionnaires, d'auto-corrections, d'indices ponctuels, de répétitions, *etc.*, dans le but d'accroître les compétences de l'élève. Cette notion est difficile à appréhender car la définir reviendrait à répondre à la question : qu'est ce que maîtriser une connaissance ? Les compétences se traduisent en effet par plusieurs biais : par la capacité à répondre correctement à des questions posées sur un thème précis, à expliquer le contenu d'une notion à une tierce personne, à généraliser lorsque le contexte change (par exemple, pouvoir appliquer un même principe de physique à différents énoncés ou utiliser une notion de mathématiques sur des problèmes de physique), ou bien à retrouver ce savoir longtemps après qu'il ait été acquis. Les tuteurs doivent donc faire en sorte de jouer sur les différents types d'activités et sur le moment auquel les proposer dans le processus d'apprentissage.

Cependant, pour que l'interaction entre un tuteur automatique et un élève soit de bonne qualité, des contraintes similaires à celles du dialogue parlé sont à prendre en compte. A partir du

moment où la machine interagit avec un utilisateur humain, les informations qu'elle collecte sont incomplètes. Dans un cas comme dans l'autre, les intentions restent masquées et les données perçues par la machine sont bruitées. Cela est d'autant plus vrai que, dans le cas du tutorat, les réponses interprétées par le module qui prend les décisions ne proviennent pas du traitement et de l'analyse d'un signal physique (la voix) mais d'une réponse de l'utilisateur sur laquelle il est difficile de mesurer l'incertitude. En effet, si un élève s'est trompé dans sa réponse alors qu'il était capable de donner la bonne, rien du point de vue de l'extérieur ne permet de le détecter. Pour pallier ces lacunes et gérer l'observabilité partielle, de nombreuses approches se basent sur un modèle afin d'approximer les variables internes de l'élève pour qu'elles soient à tout instant disponibles. Par exemple, différents critères, comme le nombre de fois qu'un même thème a été abordé, le cumul de bonnes réponses ou le comportement de l'élève, peuvent servir à estimer ses connaissances.

La Section 7.1.1 propose tout d'abord une revue des méthodes employées pour déterminer les caractéristiques d'un bon tuteur. Puis, étant donné que son fonctionnement est dans de nombreux cas conditionné à l'utilisation d'un modèle d'étudiant, un inventaire des différentes méthodes est proposé Section 7.1.2. Enfin, la Section 7.2 présente les différentes solutions envisagées au problème de tutorat.

## 7.1 Caractéristiques d'un tuteur efficace

Dans la littérature, les tuteurs automatiques sont habituellement composés de quatre modules [Murray 99] : une interface avec l'élève, un modèle des connaissances, un modèle d'enseignant et un modèle d'étudiant. Le **modèle des connaissances** est une structure dans laquelle sont détaillées les imbrications et dépendances entre les différents savoirs, les solutions aux exercices ainsi que des variantes pour y parvenir. Le **modèle d'enseignant** est celui qui décide quoi proposer et à quel moment parmi les possibilités fournies par le modèle des connaissances et en fonction des réponses de l'élève. Ce module est le pendant du gestionnaire de dialogue dans le cas du dialogue parlé. Le **modèle de l'élève** représente ses connaissances en fonction des informations que le tuteur a pu obtenir de lui. Il peut utiliser par exemple les réponses aux questions posées, la fréquence des indices proposés, le temps mis pour répondre, l'évaluation de la production, ou le nombre d'essais avant de parvenir à la bonne réponse. C'est un organe central de nombreux tuteurs car il accumule les connaissances nécessaires à une bonne décision de la part de la machine et permet de gérer le problème de l'observabilité partielle. Cette section se divise en deux parties. La première présente les atouts d'un tuteur et la seconde, les différents types de modèles d'étudiants utilisés.

### 7.1.1 Réflexion autour du tuteur

Plusieurs pistes sur ce que devrait être un bon tuteur sont présentées dans [Self 90] sous forme de principes à respecter. D'autres caractéristiques que se devrait de présenter un tuteur automatique sont données dans [Anderson 95]. Elles s'appuient sur huit règles :

- les compétences de l'élève doivent être à tout moment connues par le tuteur à travers l'utilisation d'un modèle d'étudiant,
- l'élève doit savoir quelle connaissance le tuteur cherche à lui faire apprendre,
- le tuteur doit fournir des instructions quant au problème à résoudre,
- l'apprentissage doit se faire dans un soucis de généralisation des compétences, et non seulement basé sur des exemples,

- il n'est pas souhaitable de faire appel à des tâches sollicitant trop la mémoire de l'élève,
- une correction sur les erreurs doit être immédiatement fournie,
- la granularité des instructions doit être fine,
- enfin, le tuteur doit parvenir à un apprentissage final d'un savoir, même s'il manque des morceaux élémentaires car avec la pratique, les lacunes seront comblées.

Dans [Chi 10a], l'hypothèse est faite que dans le cadre d'interactions réelles, les élèves progressent grâce aux indices continuellement fournis par l'enseignant (encouragement, acquiescement). Cela se rapproche de ce qui a été proposé dans [Anderson 95] à propos du retour immédiatement fourni et de la division du savoir en briques élémentaires. Une autre approche [Wylie 09], émet l'hypothèse que l'élève progresse lorsqu'il doit justifier sa réponse. Cela l'incite ainsi à vérifier son raisonnement. Par ailleurs, différentes méthodes d'apprentissage sont comparées dans [Koedinger 09]. La première demande à l'élève d'expliquer ses choix. La seconde lui propose de se corriger immédiatement ou non, en fonction de l'aide qui lui est apportée. La troisième propose des indices pour le guider dans sa progression. Une autre approche s'inspirant de la théorie de Piaget sur le développement de la cognition est proposée dans [Arroyo 99]. Enfin, la méthode présentée dans [Barnes 08] génère des indices pour l'élève en fonction de sa progression.

Les approches pour construire des tuteurs performants sont donc nombreuses. Chaque méthode propose d'intervenir à différents endroits du processus d'apprentissage. Ceci est dû au fait qu'il n'y a pas de concensus sur les mécanismes d'acquisition du savoir. Certains concepteurs ont donc étudié les interactions réelles pour déterminer les caractéristiques d'un enseignement performant.

Dans [Vygotsky 78], l'auteur s'interroge sur les mécanismes d'acquisition du savoir. Il développe la notion de « zone de développement proximal » qui représente la différence entre le niveau de capacité d'un élève lorsqu'il évolue seul et celui qu'il est capable d'atteindre lorsqu'il est guidé par un enseignant. Différentes expériences ont été menées avec des tuteurs et des élèves réels dans [Chi 01] pour trouver quelle méthode était la plus efficace pour faire progresser l'élève. Trois stratégies ont été comparées : une pendant laquelle le tuteur mène les échanges, une pendant laquelle l'étudiant mène les échanges et une mêlant les deux premières (le tuteur et l'étudiant échangent). La deuxième expérience, toujours avec des échanges réels, a visé à répondre aux questions soulevées par la première. Elle a consisté à tester deux stratégies d'apprentissage : pendant la première, le tuteur guide l'interaction, pendant la seconde, il demande juste à l'élève d'expliquer son point de vue. Au vu des résultats, la seconde approche semble plus prometteuse. Cette étude conclut que pour un concepteur de tuteur automatique, il est souhaitable que divers aspects autres que l'enseignement habituel (le tuteur guide tous les échanges) soient pris en compte. L'étude du comportement d'un enseignant pour déterminer quelles caractéristiques font progresser l'élève a été complétée par l'étude menée dans [VanLehn 03]. Il en ressort que le savoir associé à des situations dans lesquelles l'élève s'est trouvé en échec est celui le plus solidement acquis. L'hypothèse pour expliquer ce phénomène est que lorsque l'élève bloque sur un point, il comprend que c'est précisément là qu'il peut progresser. Les explications de l'enseignant ne seraient pas si importantes dans le processus d'apprentissage, elles serviraient juste à ce que l'élève réfléchisse plus intensément.

Toutes les différentes études visant à determiner la statégie la plus efficace pour faire progresser l'élève sont cependant restées dans le paradigme de l'apprentissage classique qui se base sur l'hypothèse que l'enseignant possède un savoir qu'il faut tenter de reproduire avec le tuteur intelligent. Or, il n'est pas acquis que ce cadre même soit le plus efficace pour faire progresser l'élève. C'est la méthode la plus efficace trouvée jusqu'alors, mais peut-être qu'en utilisant

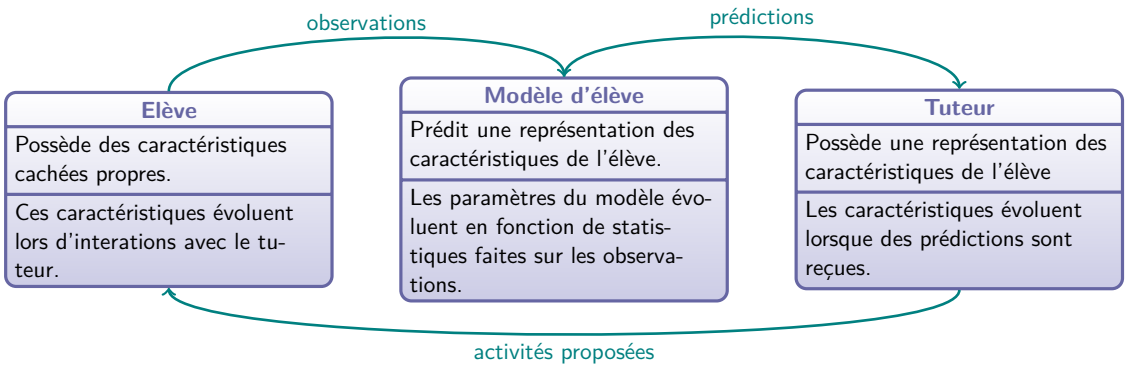



**Figure 7.1** – Relation entre élève et tuteur (ou enseignant) lorsqu'un modèle d'élève est utilisé.

la machine d'une façon autre, des techniques d'apprentissage différentes pour les élèves vont être découvertes. Par exemple, si la navigation de l'élève était laissée libre parmi les exercices proposés, le cheminement et la progression seraient différentes de celles attendues s'ils avaient été contraints. Les dépendances qui sont posés en hypothèses entre les concepts à enseigner seraient peut-être ainsi à revoir.

### 7.1.2 Modélisation de l'étudiant

Une modélisation du comportement de l'élève semble incontournable pour de nombreuses approches car elle permet d'obtenir à chaque instant une estimation du savoir et des intentions de l'élève. Cette information permet de gérer la question de l'observabilité partielle. En maintenant un historique des différentes réponses collectées sur l'élève ou en ayant une idée de comment évolue le savoir, cela permet de faire face aux situations pour lesquelles un morceau d'information est manquant. Si par exemple un élève se trompe dans sa réponse mais qu'il en a déjà fourni un certain nombre de correctes sur un même thème, le modèle peut détecter que celle courante n'est qu'une erreur et que la connaissance est toujours maîtrisée. De la qualité du modèle dépend donc la qualité de la décision. Une illustration des liens entre le tuteur et l'étudiant dans cette configuration est présentée Figure 7.1.

Les modèles proposent un cadre général qui peut être personnalisé aux moyens de paramètres réglés à l'aide de données réelles. Une étude approfondie des différentes méthodes d'apprentissage automatique existantes pour construire un modèle d'étudiant et des réflexions sur ce qui constitue un bon modèle sont données dans [Sison 98, Tetreault 06]. Plusieurs types peuvent être envisagés. Dans [Carr 77], les étudiants se départagent les uns des autres par une liste contenant les différences propres à chacun par rapport à un étudiant expert. Les caractéristiques du modèles sont modifiées en fonction des observations. Une étude menée dans [Newell 81] suppose que ce qui caractérise un élève est l'augmentation de son savoir en fonction du nombre de situations dans lesquelles il y a été confronté. Dans cette étude, un modèle de la loi de la pratique est proposé. Cette loi donne une relation entre le temps mis pour faire un exercice et le nombre d'essais effectués pour le résoudre. Une autre approche suppose que ce qui caractérise le mieux l'élève est le temps mis pour répondre à une question. Un modèle basé sur un réseau de neurones est donc chargé de prédire cette donnée dans [Beck 98]. L'efficacité de ces modèles reste cependant limitée puisque les dépendances entre les savoirs ne sont pas prises en compte. Une amélioration a été proposée avec la mise en place de modèles probabilistes.

L'approche présentée dans [Corbett 94] suppose que l'élève se définit par sa courbe de

progression tracée à partir de réponses à des questions, mais qu'elle doit être nuancée par la probabilité de deviner la réponse et celle de se tromper alors que la réponse juste était connue. Deux équations sont utilisées, l'une pour décrire la progression du savoir, d'autre pour donner la probabilité que la réponse à une question posée soit correcte. Les paramètres du modèle sont appris sur des données réelles à partir de différentes techniques de calcul [Beck 09]. Une approche similaire est développée dans [Baker 08, Baker 10]. Un modèle d'élève estimant aussi la probabilité de fournir une bonne réponse est proposé dans [Cen 05, Cen 06]. L'utilisation d'un réseau bayésien a été proposée dans [Arroyo 04, Jonsson 05, Conati 02] pour modéliser l'acquision de connaissances. Dans [Arroyo 04] ce modèle sert à prédire des variables cachées à partir de la manière dont l'étudiant a besoin des indices pour résoudre le problème. Dans [Jonsson 05] les paramètres du réseau sont appris par la méthode de maximum de vraisemblance. Des comparaisons de diverses approches sont proposées dans [Beck 00a, Hämäläinen 06] parmi des modèles basés sur une régression linéaire, un arbre de décision, des classifieurs bayésiens ou des machines à vecteurs supports. Les réseaux bayésiens présentent l'avantage de pouvoir prendre en compte les dépendances entre les différentes variables liées à l'acquisition d'un savoir.

L'inconvénient de ces modèles réside dans le fait qu'ils sont appris par avance, à partir de données collectées. Au moment où commence l'interaction avec l'utilisateur, le modèle est déjà figé. Il n'y a donc pas de possibilité d'adaptation aux nouvelles données. Or, toutes les situations ne sont pas envisageables à la conception. De plus, la fiabilité de cette approche n'est pas garantie. En effet, même si ces modèles sont construits à partir d'observations collectées, les hypothèses de base ne s'appuient pas sur des modèles statistiques éprouvés comme dans le cas du dialogue parlé. Pour appréhender la difficulté à construire un modèle, il suffit de se rendre compte de la diversité des réponses possibles lorsqu'il est demandé à un étudiant d'expliquer ce qu'il a compris après une leçon. En revanche, dans le cas du dialogue parlé, lorsqu'il est demandé à l'utilisateur de se répéter, ses intentions transparaissent au travers de ce qu'il dit. Les données modélisées dans ce cas sont donc plus représentatives de l'état de l'utilisateur que ne peuvent l'être des observations de la cognition de l'élève. Celle-ci fonctionne en effet comme une boîte noire dont les mécanismes internes restent totalement cachés. La question de la gestion de l'observabilité partielle à l'aide d'une approche sans modèle sera proposée par la suite. La seconde section de ce chapitre aborde maintenant la question de la prise de décision dans le cadre du tutorat.

## 7.2 La prise de décision

De la même façon que le gestionnaire de dialogue est chargé des décisions pour mener à bien un dialogue parlé, le tuteur est chargé ici de la séquence d'activités à proposer à partir d'un jeu de base, le but étant de faire progresser le savoir de l'élève et de le guider dans cette progression. Cependant, ce problème souffre des mêmes difficultés que celui du dialogue parlé. En effet, seules des informations partielles sont disponibles. Le problème est donc un problème de décisions sous incertitude. De plus, les méthodes classiques de résolution, telles que des jeux de règles codés à la main, ne permettent pas de prendre en compte toutes les différentes manières de maîtriser un savoir. Pour les mêmes raisons que le cas du dialogue parlé, les approches utilisant l'apprentissage par renforcement sont envisagées. Elles ont déjà montré leur efficacité sur différents problèmes liés au tutorat informatique. Un état de l'art des différentes applications basées sur le RL est donné dans [Chi 10b].

### 7.2.1 Utilisation du RL

Un des premiers travaux où le RL est utilisé pour déterminer la stratégie du tuteur a été proposé dans [Beck 97]. L'idée n'est pas mise en œuvre mais y est seulement décrite. Les données sur lesquelles le tuteur s'appuie pour prendre sa décision provient d'un modèle d'étudiant. Une implémentation de cette idée est donnée dans [Beck 00b] où l'algorithme TD(0) est utilisé. Elle est améliorée dans [Beck 00c]. La stratégie optimale est trouvée à partir des données fournies par un modèle d'étudiant dont l'usage est détourné en générateur de données, le modèle étant construit en utilisant des interactions réelles entre tuteur et étudiants à partir d'une politique sous-optimale. Le but pour le tuteur est de minimiser le temps de réponse des utilisateurs. La stratégie apprise à partir des données générées artificiellement est ensuite testée sur des étudiants réels. Les rapidités d'exécution sont ensuite comparées entre la stratégie initialement utilisée et celle apprise. Cette approche utilise un modèle d'étudiant, qui ne permet pas de prendre en compte toutes les situations possibles.

L'approche présentée dans [Tetreault 06] modélise le problème de tutorat comme un MDP discret. Les probabilités de transitions sont estimées *a priori* à partir de données réelles. La politique optimale est ensuite trouvée en résolvant le MDP dont les probabilités de transition viennent d'être calculées. Différentes informations à propos des élèves sont utilisées pour construire l'espace d'état. Elles sont testées en étant combinées de plusieurs façons. Les politiques sont ensuite comparées en mesurant les résultats obtenus par les élèves avant et après avoir interagi avec le tuteur. De même, l'article [Chi 10b] présente une approche basée sur l'estimation des paramètres d'un MDP discret. Un algorithme de programmation dynamique est chargé ensuite de déterminer la politique optimale. La progression de l'élève est comparée lorsque trois stratégies sont mises en place : une aléatoire, une utilisant un apprentissage classique et une apprise en résolvant le problème du RL. Ces deux approches, bien que prometteuses, ne permettent pas de s'affranchir du modèle. Un biais est introduit sans qu'il puisse être contrôlé. Les approches sans modèle seront alors préférées. De plus, les espaces d'état utilisés sont discrets, ce qui limite les nuances pour caractériser l'élève.

Les travaux se rapprochant le plus des nôtres sont ceux proposés dans [Iglesias 03, Iglesias 09a] où une méthode sans modèle est présentée. L'algorithme du Q-learning est utilisé en-ligne pour apprendre la stratégie optimale. Cependant, cet algorithme nécessite de nombreuses données pour converger. D'autre part, les connaissances à propos de l'étudiant du point de vue du tuteur sont représentées par un vecteur de nombres binaires. Chaque composante du vecteur correspond à une connaissance élémentaire ($0$ si elle n'est pas connue, $1$ si elle l'est) qui est évaluée au moyen de questions posées à l'étudiant après chaque action du tuteur, ce qui peut le lasser. D'autre part, une politique $\epsilon$-gloutonne est choisie pour la phase d'exploration lors de l'apprentissage de la politique. Une exploration sûre de l'espace n'est alors pas garantie, sauf si le temps est infini. Les expériences présentées sont menées avec des données collectées sur des étudiants simulés, mais le mécanisme de simulation reste invisible pour le tuteur. Dans [Iglesias 09a], différentes stratégies d'exploration sont expérimentées et des données réelles sont employées dans [Iglesias 09b]. Néanmoins dans cette approche, la progession de l'élève n'est pas prise en compte. Soit il maîtrise une connaissance, soit il ne la maîtrise pas. Or cela n'est pas le reflet de la réalité. Cela implique de pouvoir gérer les espaces continus lors l'apprentissage de la stratégie optimale. En effet, si un seuil est introduit pour juger de la maîtrise d'un savoir, après interactions avec le tuteur, un élève ne sachant rien au début et ayant acquis quelques connaissances sans avoir atteint le seuil, ne sera pas considéré comme ayant progressé.

### 7.2.2 Le tutorat comme un MDP

De la même façon que pour le dialogue, si le RL est utilisé pour résoudre le problème de l'apprentissage, le problème doit être mis sous la forme d'un MDP. Nous rappelons que les MDP (Section 3.1.1) sont définis par un n-uplet $\{S, A, T, R, \gamma\}$. L'agent devant apprendre une stratégie est le tuteur. L'environnement dans lequel il évolue est représenté par l'élève. Dans notre cadre précis du tutorat, l'espace d'état sous-jacents, $S$, n'est pas directement obervable (les connaissances de l'élève ne sont pas accessibles). Seules des observations $o \in O$ sont disponibles. Le problème du tutorat peut ainsi être aussi considéré comme un POMDP [Whitehill 09, Rafferty 11]. Plus de précisions sur la gestion de l'observabilité partielle seront fournies par la suite.

L'espace des actions $A$ est l'ensemble des activités qui se trouvent à disposition du tuteur. Généralement, chaque activité concerne une connaissance élémentaire. Elle vise à augmenter son savoir, à le tester ou à le laisser s'exercer. Elles sont par exemple, « répondre à une question », « remplir un texte à trous », ou « lire la leçon ». Les activités sont définies par le concepteur du sytème et couvrent l'ensemble du domaine de connaissances dans lequel l'élève doit progresser. La définition de ces activités n'est pas facile car elle suppose de connaître précisément comment se forme la représentation du savoir chez l'élève et quels sont les mécanismes qui permettent sa maîtrise. Le RL peut être intéressant dans ce cas car il ne nécessite ni la définition manuelle des dépendances entre les concepts ni des *a priori* nécessaires à l'acquisition d'une nouvelle compétence. En effet, par essais-erreurs, le tuteur s'apercevra des notions qui doivent être enseignées l'une avant l'autre. S'il propose des activités trop tôt dans l'apprentissage, les scores seront faibles. S'il les propose après avoir donné la bonne suite de leçons par exemple, les scores seront meilleurs.

L'espace des transitions n'est pas connu puisque le modèle de la cognition de l'élève n'est pas connu. La fonction de récompense est donnée en fonction d'information fiables à propos de l'élève. Cela peut être la réponse à des questions de divers degrés de difficulté, le score à un exercice posé pour une réutilisation de connaissances, ou le temps mis pour répondre par exemple. Des sytèmes plus perfectionnés peuvent prendre en compte les émotions, le nombre de clics sur l'écran, ou la trajectoire des yeux.

## 7.3 Conclusion

Le problème du tutorat se place dans le cadre des interactions homme-machine. Les problèmes soulevés en introduction de ce manuscrit doivent alors être pris en compte. Tout d'abord, la stratégie déployée doit être la plus performante possible et ce dès le début de l'apprentissage pour limiter les risques d'abandon. La question de l'exploration de l'espace d'état se pose alors ainsi que celle de la représentativité des données utilisées pour l'apprentissage. Ce problème est abordé dans le Chapitre 8 au travers d'un apprentissage off-policy et hors-ligne.

L'autre question qui se pose est la gestion de l'observabilité partielle. Elle est habituellement traitée par l'utilisation d'un modèle d'étudiant pour disposer d'informations les plus précises possibles à tout moment de l'interaction. Cependant, contrairement au cas du dialogue où l'incertitude du modèle provient du traitement statistique d'un signal, celle-ci ne peut pas être estimée dans le cas du tutorat. Une approche sans modèle sera donc préférée. Elle est représentée Figure 7.2. Dans ce cas, les informations de l'étudiant sont directement transmises au tuteur. Elle sera développée et illustrée par un exemple au Chapitre 9.

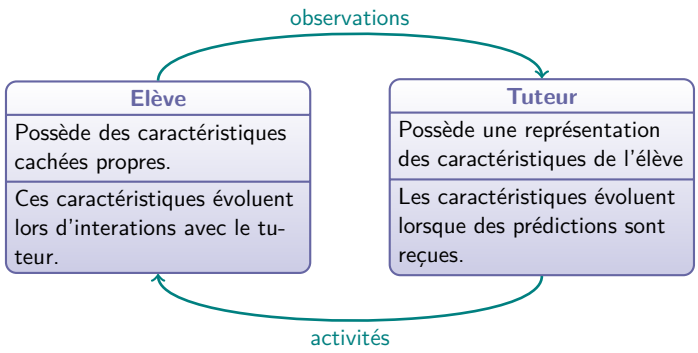



**Figure 7.2** – Relations entre élève et tuteur (ou enseignant) lorsqu'un modèle l'élève n'est pas utilisé.

# Chapitre 8

# Une approche hors-ligne

Pour chaque application mettant en jeu une interaction homme-machine, des contraintes supplémentaires sont à prendre en compte. En particulier, le comportement de la machine se doit d'être cohérent de bout en bout de l'échange. L'exploration de l'espace d'état doit se faire avec attention. Nous avons vu en détails dans la partie dialogue les différentes possibilités pour la prendre en charge : une approche on-policy et en-ligne en utilisant l'information d'incertitude sur l'estimation de la fonction de valeur ou celle off-policy qui présente à l'utilisateur une politique dont le concepteur peut s'assurer qu'elle présente un comportement stable. Cette dernière est mise en œuvre pour la partie tutorat à partir de l'algorithme LSPI, c'est-à-dire dans un cadre hors-ligne. Nous rappelons que cela permet de réutiliser des données dont la collecte est parfois coûteuse. Ce chapitre présente le test de cette approche. Après une présentation de la modélisation du problème utilisée comme preuve de concept contenant la présentation de la simulation d'étudiant employée pour générer les données à la Section 8.1, les résultats seront présentés Section 8.2. Ces travaux ont été publiés dans [Pietquin 11a, Daubigney 12d, Lefevre 12].

## 8.1 Cadre expérimental

Pour tester l'approche hors-ligne et off-policy, le cadre expérimental suivant a été défini. La première partie, Section 8.1.1, présente le problème de tutorat considéré et la seconde, la modélisation d'étudiant, Section 8.1.2.

### 8.1.1 Modélisation du problème de tutorat

Pour cette première modélisation du problème de tutorat, il est considéré que le tuteur dispose de deux activités pour faire progresser l'élève : une leçon qui permet d'augmenter ses connaissances et une question qui permet d'obtenir une mesure estimée du savoir. Le MDP résultant de cette modélisation est donné Figure 8.1. En ce qui concerne l'état, il est constitué de deux dimensions. La première représente le taux de bonnes réponses à la question dernièrement posée. Cette variable est continue entre zéro et un. Cela permet de prendre en compte toute progression. La deuxième dimension contient un compteur tenant à jour le nombre de leçons données. Cette information fournit un historique des interactions qui permet de traiter l'aspect partiellement observable. Avec ces deux données fournies à un instant précis, le tuteur possède toutes les données pour prendre sa décision. Les actions possibles sont au nombre de deux : soit une phase d'évaluation a lieu par le biais d'une question, soit c'est une phase d'apprentissage par

une leçon. Quand une question est posée, la récompense est proportionnelle au score obtenu, lorsque c'est une leçon, une récompense nulle est donnée. Le fait d'accorder une récompense même lorsqu'un seuil de bonnes réponses n'est pas atteint permet de prendre en compte la progression. Le facteur $\gamma$ est fixé à $0,9$.

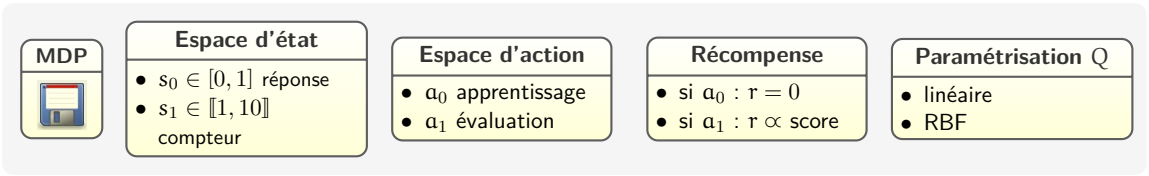



**Figure 8.1** – Modélisation du tutorat.

### 8.1.2 Simulation de l'étudiant

Le test de la méthode a été mené sur des données simulées. La mise en place de la logistique pour collecter des données est très coûteuse. De plus, elle n'assure pas la reproductibilité des expériences, nécessaire pour les raisons de comparaisons. La simulation d'élève exhibe deux caractéristiques. D'une part l'élève est capable de répondre à une question posée et d'autre part, il voit son savoir augmenter lorsqu'une leçon lui est proposée. Elle s'appuie sur le modèle d'étudiant présenté dans [Corbett 94] puis amélioré et utilisé ensuite dans [Corbett 97, Baker 08, Baker 10]. Il suppose que l'étudiant est caractérisé par sa courbe d'apprentissage. Le choix des paramètres a été fait selon ceux proposés dans [Chang 06], eux-mêmes estimés à partir de données réelles. Dans cet article, le modèle est utilisé pour approximer le comportement de l'utilisateur et maintenir une estimation de son savoir au fur et à mesure des interactions. Dans notre cas, il est exclusivement employé pour générer des données. Les paramètres du modèle ne sont donc pas connus du tuteur. Seules les informations décrites précédemment lui sont fournies.

Chaque étudiant est caractérisé par quatre probabilités : celles de deviner la bonne réponse à une question alors qu'il n'est pas censé la connaître $p(G)$, de se tromper en répondant alors que le savoir est connu, $p(S)$, d'augmenter le savoir de l'étudiant quand une leçon lui est présentée, $p(T)$ et de maîtriser le savoir au pas de temps $i$, $p(L_i)$. Après une leçon, la probabilité que l'élève ait acquis la compétence est :

$$p(L_{i+1}) = p(c) + (1 - p(c)) \cdot p(T), \tag{8.1}$$

$$\text{avec } p(c) = (1 - p(S)) \cdot p(L_i) + p(G) \cdot (1 - p(L_i)), \tag{8.2}$$

la probabilité que la réponse soit correcte. Une illustration de la progression de l'élève est donnée Figure 8.2 en traçant la probabilité $p(L_i)$ après chaque leçon. Quand une question est posée à l'utilisateur, la réponse est donnée en tirant un nombre au sort entre $[0,1]$. Si ce nombre est supérieur à $p(c)$, la réponse est incorrecte. Sinon, elle est correcte. Dans ce modèle, la probabilité de répondre correctement à une question est étroitement liée à l'augmentation des connaissances. Ceci est dû au fait qu'une réponse correcte à une question laisse bien présager que la compétence mise en jeu dans cette question est acquise.

## 8.2 Test de LSPI sur le tutorat

L'approche hors-ligne et off-policy a été testée grâce aux modélisations du problème et de l'étudiant présentées à la section précédente. Les données ont été générées en amont de l'expérience. Etant donné que la première dimension de l'espace d'état est continue, il est

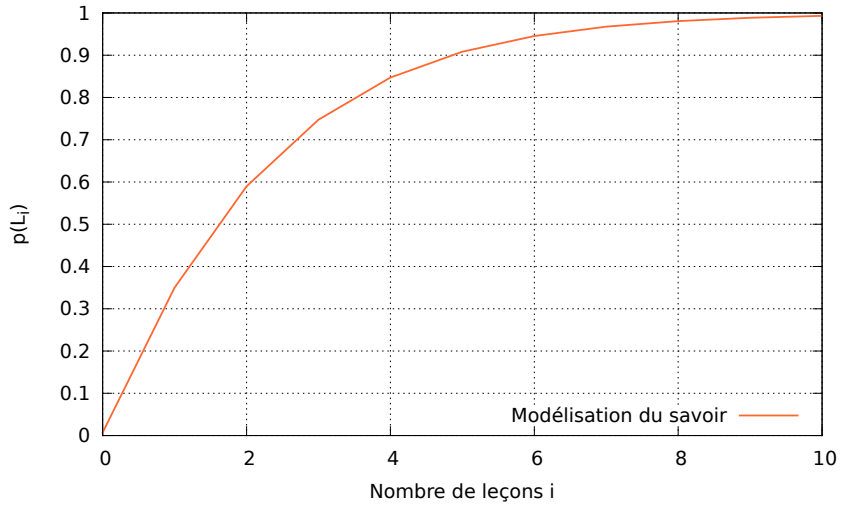



**Figure 8.2** – Evolution des connaissances de l'élève $p(L_i)$ en fonction du nombre de leçons donné $i$.

nécessaire de choisir une représentation de la fonction de qualité qui puisse estimer l'état sur tout l'espace où il est défini. Une paramétrisation linéaire basée sur un RBF est choisie (Section 3.2). Trois gaussiennes pavent la première dimension. La deuxième dimension est discrète. La moyenne des récompenses pondérées cumulées obtenue lors du test de la politique apprise avec LSPI est tracée Figure 8.3, en fonction du nombre de transitions utilisées pour l'entraînement. La courbe LSPI présente la moyenne de $100$ apprentissages de la politique. Chacun a été testé $1000$ fois.

Les résultats sont comparés à deux autres politiques, une politique aléatoire ayant servi à générer les données et une politique qui correspond à un schéma classique d'apprentissage, une alternance de questions et de leçons. Au début de l'apprentissage, la politique proposant un schéma classique donne de meilleurs résultats que celle apprise avec LSPI. L'utilisateur fait ainsi face à des situations moins critiques que s'il était directement confronté à la politique donnée par LSPI. L'intervalle de confiance à 95% est tracé pour les trois approches. Puis, plus le nombre de données d'entraînement augmente, plus la politique renvoyée par LSPI donne de bons résultats. Au delà de $600$ échantillons, elle est meilleure voire nettement meilleure pour des jeux de données d'apprentissage importants que celle classique. La politique apprise avec LSPI est plus efficace car elle apprend le nombre de leçons à donner et se contente ensuite d'une question pour obtenir la récompense. Le critère $\gamma$ pondère le nombre de leçons : à partir d'un certain seuil, il est plus intéressant d'obtenir la récompense en proposant une question, que de donner une leçon dont le gain sur l'augmentation du savoir de l'élève ne sera pas suffisant pour justifier d'attendre le tour suivant pour poser la question.

## 8.3 Conclusion

Ces expériences préliminaires dans le cadre d'un problème de tutorat mis sous la forme d'un MDP présentent un apprentissage hors-ligne et off-policy, ce qui, à notre connaissance est le premier test d'une approche de ce type. Grâce à la politique classique, l'exploration peut-être menée de façon sûre au début de l'apprentissage afin de collecter des données en nombre suffisant. De plus, la représentation paramétrique linéaire continue de la fonction de qualité permet de prendre en compte la moindre progression de l'élève contrairement aux approches

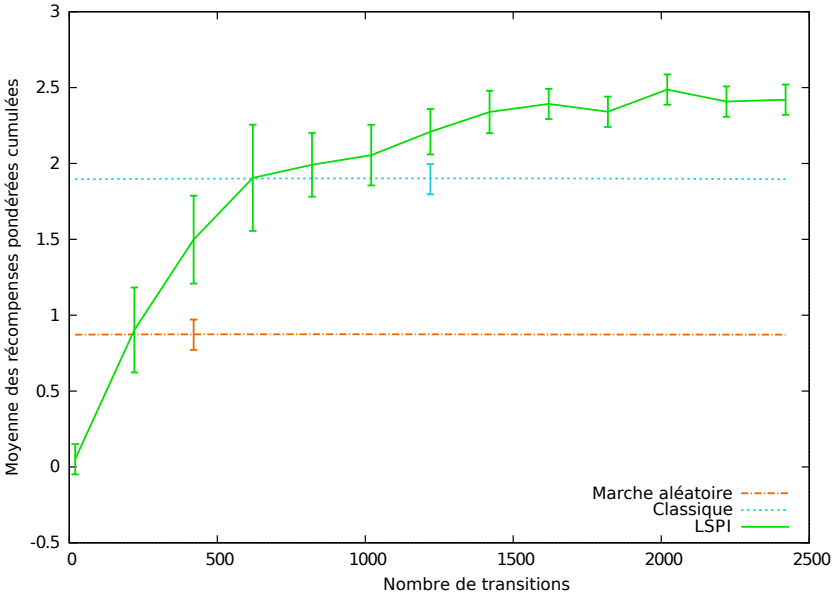



**Figure 8.3** – Test de la politique apprise avec LSPI.

binaires qui ne la valident que lorsqu'elle dépasse un certain seuil. Il serait intéressant ensuite de tester cette approche avec l'algorithme KTD afin que l'autres contraintes comme la gestion de la non-stationnarité soient prises en compte.

Dans cette preuve de concept, l'état du MDP a été construit de façon à ce que les informations qu'il contient à un instant donné soient suffisantes pour prendre la décision courante. Dans le chapitre suivant, la question de la gestion de l'observabilité partielle va être abordée. En effet, bien qu'elle fournisse un reflet précis des connaissances de l'utilisateur, l'information seule sur l'estimation du savoir au travers de la réponse à une question n'est pas complète et le compteur de leçons est une donnée qui n'est pas forcément assez précise ou disponible sur des problèmes plus complexes.

# Chapitre 9

# Gestion de l'observabilité partielle

Comme dans chaque application où se déroule une interaction homme-machine, le cadre du tutorat n'échappe pas au problème de l'observabilité partielle des informations provenant de l'utilisateur. Nous rappelons quelques exemples : lorsqu'il essaie de jouer avec la machine et apprend par coeur les réponses ou lorsqu'il se trompe en répondant alors que la bonne solution est connue, dans les deux cas, le score obtenu ne reflète pas l'état de ses connaissances. L'autre aspect non markovien inhérent au problème de tutorat est que l'information courante ne suffit pas à prendre une bonne décision. Par exemple, si le tuteur souhaite proposer l'apprentissage d'une notion qui dépend de la maîtrise d'une autre, il faut qu'il vérifie que celle en amont est déjà acquise. Pour cela, une représentation compacte des situations déjà rencontrée doit être tenue à jour.

Le problème de dialogue parlé traite de cet aspect en utilisant des modèles qui se basent sur l'analyse statistique de la constuction du langage. Ils permettent de gérer avec efficacité les incertitudes issues du traitement du signal vocal. Le cadre habituel de résolution du RL peut ensuite être apppliqué car le POMDP est transformé en MDP puisque un modèle d'observation liant les informations collectées aux états sous-jacents est construit. Cependant, des modèles présentant la même fiabilité ne sont pas encore disponibles dans le cadre du tutorat. Pour cela, il faudrait par exemple construire des représentations de la cognition humaine, en se basant éventuellement sur l'imagerie cérébrale ou des électro-encéphalogrammes. Ainsi, en analysant le signal physique, des modèles statistiques des zones spécifiques aux différents types d'apprentissages pourraient être construits. Plusieurs études ont déjà essayé d'identifier ces zones [Driemeyer 08, Lerch 11]. Malheureusement, toutes ces techniques ne sont pas encore suffisamment accessibles pour pouvoir être appliquées à grande échelle.

La solution consiste alors pour le moment à se baser sur les informations les plus fiables disponibles : l'historique des échanges entre l'élève et le tuteur. En reprenant l'exemple de la dépendance entre deux notions, si le tuteur se souvient avoir abordé la première, il sait qu'il peut alors passer à la suivante. De plus, s'il a obtenu de très bons scores pour une certaine connaissance et qu'une question est reposée quelques temps plus tard à ce même propos, si la réponse est fausse, cela laisse supposer que l'élève s'est trompé et non qu'il ne maîtrise plus le savoir. La première solution vise à mémoriser toutes les informations qui ont été échangées. Cependant l'interaction peut durer longtemps. Pour des raisons de mémoire, cette approche doit être modulée en se souvenant par exemple des informations échangées aux $T \in \mathbb{N}$ pas de temps précédant l'instant présent. Il reste alors le choix de la largeur de la fenêtre T à résoudre. Si c'est trop court, les informations essentielles ne seront pas collectées. Si c'est trop long, la solution ne se sera pas calculable. Une alternative réside dans le fait de construire une

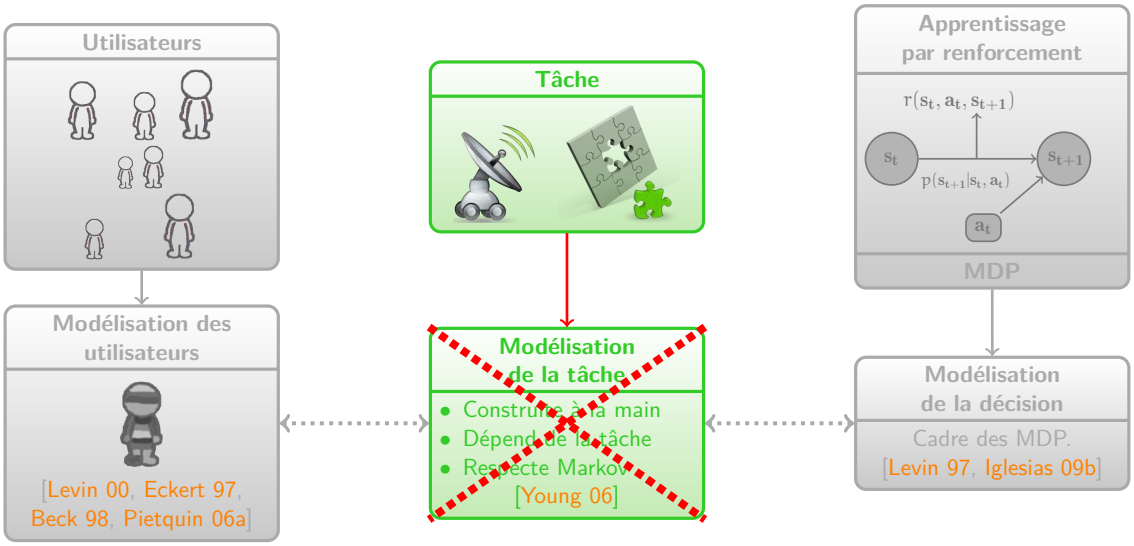



**Figure 9.1** – La troisième contribution se focalise sur la modélisation de la tâche. Celle-ci est évitée grâce à l'utilisation des ESN.

représentation synthétique de l'historique afin de prendre en compte un maximum de données. Cette alternative fait l'objet de la contribution présentée dans ce chapitre. Elle est illustrée Figure 9.1 et consiste à s'affranchir de la modélisation de la tâche. Du fait du stockage de l'information passée, le caractère partiellement observable disparaît et le POMDP devient un MDP en utilisant l'historique comme état. La résolution du POMDP est alors effectuée sans modèle.

Un inventaire non exhaustif des méthodes de résolution du POMDP sans modèle est présenté Section 9.1. Celle retenue pour mener nos expériences, basée sur des réseaux de neurones récurrents sera détaillée Section 9.2. Elle sera ensuite testée sur le problème de tutorat légèrement modifié (présenté Section 8.1.1) dans la Section 9.3. Les travaux présentés dans cette section ont été publiés dans [Daubigney 13a].

## 9.1 Résolution du POMDP sans modèle

Pour limiter l'introduction de biais lié à l'utilisation d'un modèle, les approches sans modèle d'observation sont préférées pour résoudre le problème de tutorat représenté comme un POMDP. Deux approches principales sont décrites dans ce qui suit. La première se base sur le fait de garder en mémoire les informations des T pas de temps précédent, une fenêtre glissante, présentée Section 9.1.1. La seconde est basée sur des réseaux de neurones récurrents qui permettent de mémoriser les informations passées mais sans avoir à se soucier du choix de la largeur de la fenêtre, présentée Section 9.1.2. Cette approche est ainsi préférée à la première car elle limite les *a priori* de départ. Il existe une autre approche, appelée *Predicitve State Representation* (PSR en anglais) [Littman 02] qui permet de résoudre les problème partiellement observés sans modèle. Une amélioration pour un passage à grande échelle est proposée dans [Boots 12, Hamilton 13].

### 9.1.1 Fenêtre Glissante

Une fenêtre glissante prenant en compte les observations et les actions passées pendant les T pas de temps précédents l'observation courante est une méthode pour construire une représentation markovienne [Lin 92]. Le problème qui découle de l'utilisation de cette approche est la détermination de la profondeur de la fenêtre glissante pour être sûr de bien pouvoir distinguer deux observations. Pour résoudre ce problème, une approche détectant automatiquement la largeur a été proposée dans [Dutech 03], inspirée par les travaux de [McCallum 96]. Une représentation de l'espace d'état, dont les membres sont appelés *états étendus*, est construite itérativement à partir de séquences de couples observation/action. A chaque fois que deux historiques différents mènent à deux obervations similaires, un nouvel état étendu est ajouté. Un algorithme de RL est ensuite appliqué. De même que pour l'approche utilisant les PSR, cette approche souffre d'un passage à l'échelle et n'est applicable qu'à des problèmes dont le nombre d'observations reste limité.

### 9.1.2 Réseaux de neurones récurrents

Les réseaux de neurones récurrents (*Recurrent Neural Networks*, RNN en anglais) sont des réseaux constitués de trois couches : une d'entrée, une cachée et une de sortie. Une description est donnée Figure 9.2. La particularité de ces réseaux réside dans le fait que les neurones de la couche cachée sont connectés à eux mêmes ainsi qu'à tous les autres neurones de la couche par des connexions synaptiques. A chaque nouvelle entrée reçue, la valeur au pas de temps précédent des neurones de la couche cachée est utilisée pour calculer celle courante des neurones de sortie. Une dépendance temporelle entre les différentes valeurs successives de l'entrée est ainsi prise en compte, ce qui construit une mémoire de la séquence des entrées passées. Plus le nombre de neurones de la couche cachée est grand, plus la mémoire construite est profonde. En effet, la taille des séquences pouvant être mémorisée est plus importante. Un cas particulier de RNN, les *Long Short Term Memory* [Hochreiter 97] est utilisé pour contruire une représentation de l'espace d'état pour une application au RL dans [Bakker 02].

Cependant, à cause de ces connexions récurrentes, les méthodes classiques d'entraînement des réseaux de neurones, telles que les descentes de gradient, sont difficiles à adapter. L'utilisation des *Echo State Networks* (ESN en anglais) leur sera donc préférée. Les ESN sont des RNN dont le nombre de neurones de la couche cachée est important et où seules les connexions synaptiques de cette couche, alors appelée réservoir, vers celle de sortie sont entraînées. Toutes les autres, celles de la couche d'entrée vers le réservoir et celles internes au réservoir sont définies à l'avance et aléatoirement par le concepteur du réseau. De plus, beaucoup de ces connexions sont nulles. Ces réseaux gardent néanmoins la caractéristique des réseaux de neurones récurrents et présentent une mémoire des observations/actions passées. Des outils pour entraîner ce type de réseau ont été initialement présentés dans [Jaeger 02] puis repris et améliorés dans [Lukosevicius 09].

Plusieurs exemples d'applications basés sur des ESN et du RL existent : problèmes jouet [Chatzidimitriou 11, Szita 06], ordonnancement des tâches d'un cluster [Perez 10], ou encore navigation robotique [Oubbati 11]. Un argument supplémentaire en faveur des ESN est apporté dans [Szita 07] où il est ainsi montré que la représentation de l'espace d'état fournie par les ESN est garantie d'être markovienne, sous réserve que certaines conditions raisonnables soient respectées , portant par exemple sur la construction des matrices de connections d'entrée et cachées. Une description détaillée de leur principe est proposée dans la section suivante ainsi que les avantages à les voir appliqués au RL.

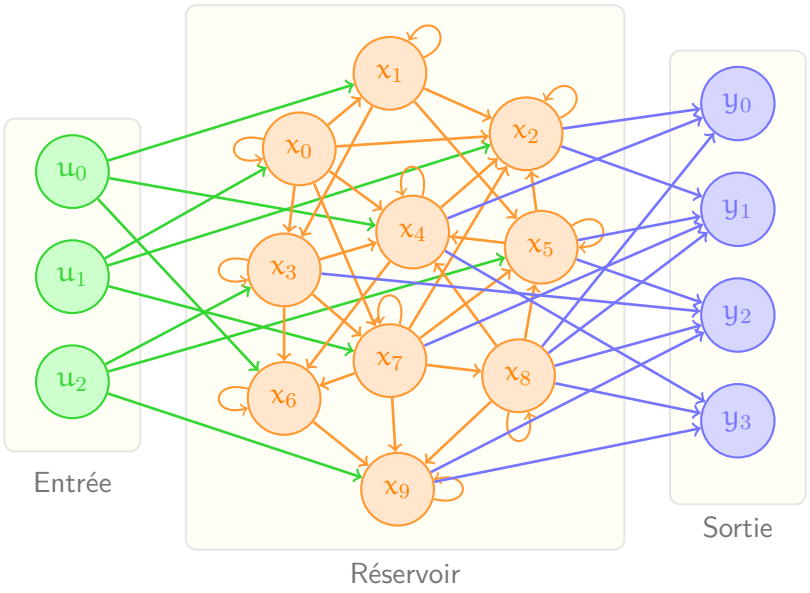



**Figure 9.2** – Principe d'un réseau récurrent (pour des questions de lisibilité, toutes les connexions ne sont pas représentées).

## 9.2 Echo State Networks

Le principe des ESN est tout d'abord rappelé Section 9.2.1. L'application à la résolution de problème de RL est ensuite présentée Section 9.2.2.

### 9.2.1 Principe

Les ESN sont un type particulier de réseaux récurrents. Les connexions internes sont générées de telle façon que la propriété d'*echo state* soit respectée [Jaeger 02]. Elle s'énonce ainsi : si, après $t$ mises à jour des neurones d'entrée avec deux séquences différentes, les deux états internes du réseau sont identiques, alors les deux séquences données en entrée sont identiques.

Les connexions entre les différentes couches de l'ESN sont présentées Figure 9.3, avec $u_t \in \mathbb{R}^{N_e}$, un vecteur colonne représentant l'entrée au pas de temps $t$ ($N_e$ étant le nombre de neurones sur la couche d'entrée), $x_t \in \mathbb{R}^{N_c}$, un vecteur colonne représentant l'activité des neurones cachés ($N_c$ étant le nombre de neurones cachés) et $y_t \in \mathbb{R}^{N_s}$, un vecteur colonne représentant l'activité des neurones de sortie ($N_s$ étant le nombre de neurones sur cette couche).

La matrice $W^e \in \mathbb{M}_{N_c \times N_e}$ contient les poids synaptiques des connexions entre les neurones de la couche d'entrée et ceux de la couche cachée. La matrice $W^c \in \mathbb{M}_{N_c \times N_c}$ contient les poids des connexions entre les neurones de la couche cachée. Enfin, la matrice $W^s \in \mathbb{M}_{N_s \times N_c}$ contient les poids des connexions entre les neurones de la couche cachée et ceux de la couche de sortie.

La matrice d'entrée $W^e$ peut être une matrice creuse. Les coefficients de la matrice cachée $W^c$ doivent être choisis de telle sorte que le réseau exhibe la propriété d'*echo state* [Jaeger 02]. Aucune condition permettant de définir une matrice ayant cette propriété n'a encore été trouvée. Seule une contrainte sur le rayon spectral de la matrice, $\alpha$, est à respecter. Ce coefficient correspond à la valeur de la plus grande des valeurs propres de la matrice. S'il est supérieur à l'unité, l'apprentissage est rendu instable et ne donne donc pas cette propriété à la matrice.

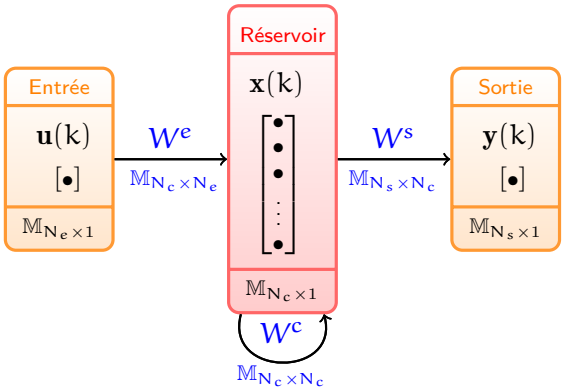



**Figure 9.3 –** Structure d'un ESN. Pour l'exemple, $N_e = 1$ and $N_c = 1$.

L'activation des neurones de la couche cachée est la suivante :

$$x_{t+1} = f(W^c x_t + W^e u_{t+1}) \tag{9.1}$$
$$y_{t+1} = W^s x_{t+1}, \tag{9.2}$$

avec $f$ la fonction d'activation (par exemple, $f = \tanh$). Après entraînement, la sortie de l'ESN $y$ renvoie une approximation linéaire de l'entrée, les fonctions de base étant les valeurs des neurones de la couche cachée. Dans un cas supervisé, les coefficients $W^s$ peuvent être appris grâce à, par exemple, un algorithme des moindres carrés récursifs.

### 9.2.2 Utilisation dans le cadre du RL

L'utilisation des ESN dans le cadre du RL est intéressante. En effet, sous réserve que le réseau exhibe la propriété d'echo state, la représentation fournie par les neurones de la couche cachée contient l'historique des entrées. Elle peut alors être utilisée comme état dans le cadre des MDP. Ici, plus précisément, elle sert de fonction de base pour calculer une approximation linéaire de la fonction de qualité. L'avantage des ESN réside dans le fait qu'aucune hypothèse explicite n'est faite quant au lien entre l'état sous-jacent de l'élève, représenté par les valeurs des neurones de la couche cachée, et les observations/actions.

Pour résoudre le problème de RL avec un ESN, le réseau présente en sortie, étant donné une observation $o_t$ et une action $a_t$ au pas de temps $t$, une estimation de la fonction Q, $\hat{Q}_{\theta_t}(h_t, a_t)$ (avec $h_t = \{o_0, a_0, ..., o_{t-1}, a_{t-1}, o_t\}$, l'historique des observations/actions). L'état $s_t$ qui contient les intentions exactes de l'utilisateur, n'apparaît pas directement dans l'estimation de la fonction Q puisqu'il n'est pas accessible. La connaissance de sa valeur est remplacée par l'historique $h_t$. Pour faire le lien avec la description des ESN fournie précédemment, l'entrée $u_t$ est la concaténation de l'observation courante $o_t$ et de l'action courante $a_t$ : $u_t = [o_t \ a_t]$. L'état interne $x_t$ est un résumé de l'historique $h_t$ et de l'action $a_t$. Les coefficents de la matrice de sortie, $W^c$, sont appris par un algorithme de RL, en utilisant une approximation linéaire de la fonction de qualité. Au pas de temps $t$, l'estimation de cette fonction est la suivante :

$$\hat{Q}_{\theta_t}(h_t, a_t) = \theta_t^T x_t. \tag{9.3}$$

Si la politique est déduite de la fonction de qualité par un $\mathrm{argmax}$ sur les actions, le calcul de l'état interne doit être fait avec précaution. En effet, avant chaque nouvelle action reçue, l'état interne doit être sauvegardé pour revenir à la configuration précédente après le test de

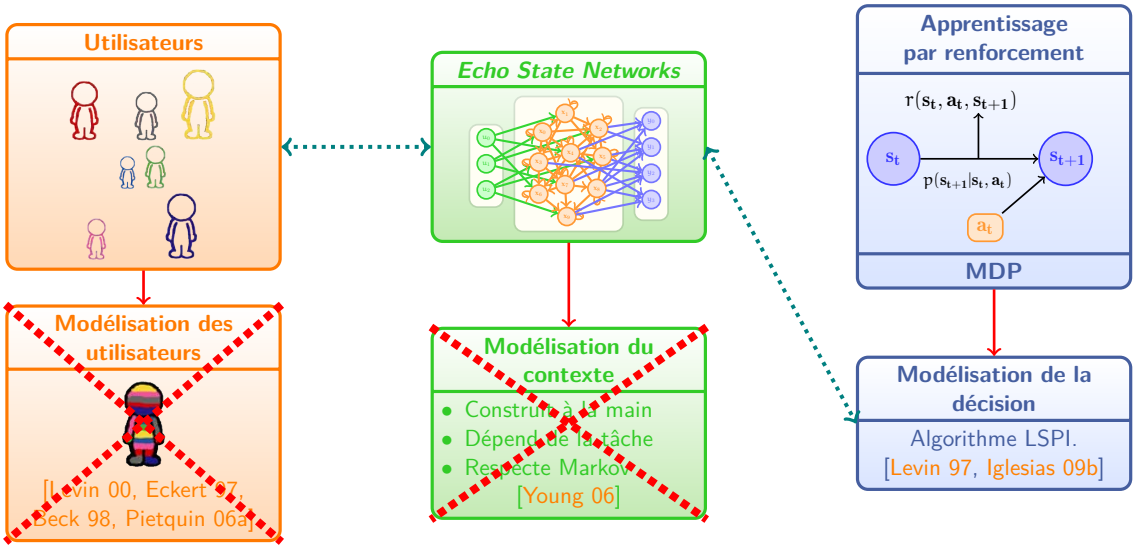



**Figure 9.4** – Solution envisagée : utilisation des ESN pour s'affranchir de la modélisation de la tâche. Le cadre markovien est maintenu par l'utilisation d'un algorithme de RL, LSPI.

chaque action. Ce processus est détaillé Algorithme 9.1. De plus, sur une tâche épisodique, il est nécessaire de remettre l'état interne à une configuration initiale à chaque fin d'épisode.

**Algorithme 9.1** Calcul de l'action gloutonne $a$ avec un ESN

Données : $x_{t-1}$, $o_t$
Initialisation : $a_{max} \leftarrow a_0$, $x_{mem} \leftarrow x_{t-1}$, $Q_{max} = \hat{Q}_{\theta_t}(h_t, a_0)$
**pour chaque** $a \in A \backslash \{a_0\}$ **faire**
  Sauvegarder $x_{t-1} \leftarrow x_{mem}$
  Calculer $\hat{Q}_{\theta_t}(h_t, a) = \theta_t^T x_t$ avec $u_t = [o_t \ a]$
  **si** $\hat{Q}_{\theta_t}(h_t, a) > Q_{max}$ **alors**
    $a_{max} \leftarrow a$
  **fin**
**fin**
Retourner $a_{max}$

## 9.3 Test des ESN sur le tutorat

La combinaison ESN et RL est maintenant illustrée sur le tutorat. La configuration de la solution envisagée est présentée Figure 9.4. La modélisation du problème présentée Section 8.1.1 est légèrement modifiée pour avoir une observation partielle des activités de l'élève. Celle de l'étudiant reste inchangée (Section 8.1.2). Ensuite, les résultats du test sont présentés Section 9.3.2.

### 9.3.1 Tutorat partiellement observable

Les modifications du problème précédemment présenté concernent la représentation de l'espace d'état et le nombre d'actions disponibles. La modélisation est résumée Figure 9.5. Trois

types d'observations sont renvoyés au tuteur. Si une leçon est proposée, l'observation est neutre : il n'y a pas d'information fournie par l'étudiant. Les conséquences de cette leçon ne sont pas directement observées non plus. Les deux autres observations apparaissent quand une question est posée (question fermée, deux seuls types de réponse possibles). Le modèle d'observation ainsi choisi est partiellement observable. En effet, par exemple, si la réponse à une question est correcte, il n'est pas possible de savoir combien de leçons ont permi d'aboutir à ce résultat d'après l'observation courante.

Trois activités sont proposées à l'élève. La leçon a pour but d'accroître les connaissances de l'élève, selon l'Equation 8.1. La question a pour but d'obtenir des informations sur les connaissances de l'élève. La réponse est donnée en fonction de l'Equation 8.2. La dernière action possible est un examen final. Il consiste à poser une série de questions pour évaluer l'élève. Une approximation plus précise du savoir de l'élève est ainsi perçue par le tuteur. A chaque fois qu'il propose l'examen final, un nouvel épisode commence. La récompense associée n'est non-nulle que dans le cas où l'examen final est proposé. Elle dépend de la réussite de l'élève à ce même examen. La récompense n'est pas conditionnée à un seuil. Toute progression de l'élève est prise en compte. La facteur $\gamma$ est fixé à $0,97$. Les résultats utilisant cette modélisation du problème sont présentés dans ce qui suit.

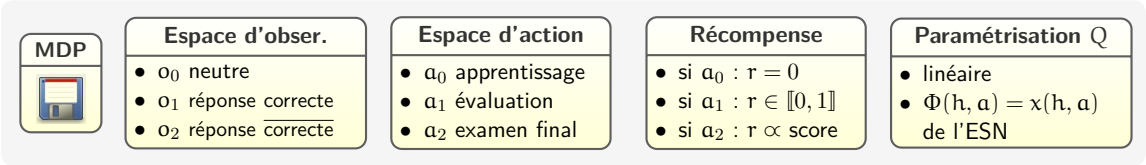



**Figure 9.5** – Modélisation du tutorat (partiellement observable).

### 9.3.2 Résultats expérimentaux

Plusieurs stratégies ont été comparées pour mesurer l'efficacité de la solution proposée combinant ESN et RL. La solution est déterminée en utilisant l'algorithme LSPI. A notre connaissance, c'est la première fois qu'une approche combinant ESN et LSPI est proposée. Une stratégie aléatoire a tout d'abord été expérimentée. Avec une probabilité de $0,6$ une leçon est proposée. Avec une probabilité de $0,2$ une question est choisie, de même que pour le choix de l'examen final. Une deuxième comparaison est proposée avec la politique réactive, c'est-à-dire celle n'utilisant que la seule observation pour prendre une décision, elle aussi apprise grâce à LSPI. La troisième comparaison utilise l'observation courante ainsi qu'un compteur de leçons comme état sur lequel se base le tuteur pour prendre sa décision. Grâce à cette donnée supplémentaire, l'information nécessaire est supposée être contenue dans l'état. Aucune mémoire n'est ainsi requise et les conséquences d'une leçon sont retranscrites au travers de l'information apportée par le compteur. La fonction de valeur est dans ce cas représentée de façon tabulaire puisque le nombre d'observations et de leçons données est fini et raisonnable. Cette approche est appelée « informée ».

D'autre part, trois groupes d'étudiants ont été testés. Chaque groupe possède son propre jeu de quatre paramètres. Ils ont été choisis de telle sorte que le premier groupe $\mathrm{T1}$ ait déjà un bon niveau et qu'il progresse vite. Le second groupe $\mathrm{T2}$ est un groupe moyen. Quelques leçons sont nécessaires pour le faire progresser. Enfin, le dernier groupe d'élèves $\mathrm{T3}$ a un niveau faible : il progresse lentement et a besoin de beaucoup de leçons pour progresser. Les données générées avec la stratégie aléatoire ont été utilisées pour mener les apprentissages de la politique où les ESN sont utilisés, de la politique réactive, et enfin de celle informée. L'ESN utilise $50$ neurones

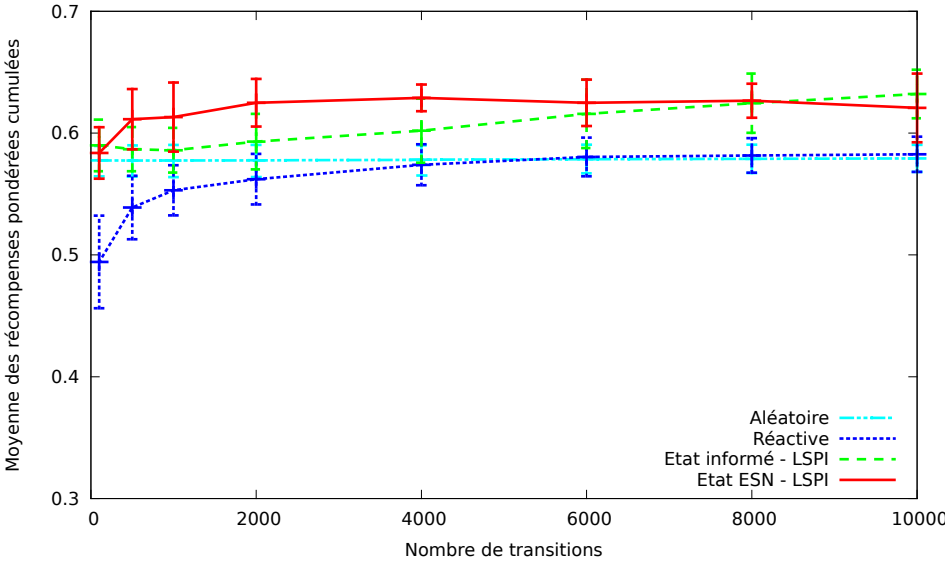



**Figure 9.6** – Comparaison des différentes stratégies.

cachés.

Pour les expériences présentées par la suite, différentes tailles de jeux de données sont testées. Parmi les données d'apprentissage, les trois types d'étudiants sont représentés en proportions égales. Une centaine de politiques est apprise pour chacune des méthodes sauf lorsque les ESN sont utilisés. Dans ce cas, $10$ ESN différents sont générés et servent $10$ fois chacun. Le nombre d'essais avec les ESN est donc aussi $10 \cdot 10 = 100$ politiques. La moyenne des récompenses pondérées cumulées fournie par le test, à l'aide des trois types d'étudiants, de chacune des politiques apprises ($1000$ séquences chacune) est donnée Figure 9.6. Les politiques aléatoire et réactive donnent les moins bon résultats. La moyenne de cette dernière augmente cependant à cause de la représentation des données d'entraînement. Pour des jeux de petite taille, il est peu probable d'avoir rencontré de longues séquences de leçons, donnant après examen final les plus hautes récompenses. Les deux autres courbes proposent de bons résultats avec un nombre raisonnable de transitions : autour de $8000$ données, une stratégie performante est trouvée grâce à l'état informé et LSPI. Celles apprises en utilisant l'état fourni par l'ESN et LSPI n'ont pas besoin d'autant de transitions pour atteindre un résultat comparable. L'écart-type est plus important quand les ESN sont utilisés. Cela peut s'expliquer par le fait qu'une incertitude supplémentaire est introduite dans ce cas car les matrices d'entrée et cachée sont générées aléatoirement. Autour de $10000$ transitions, les deux stratégies renvoient des résultats similaires. Bien qu'ils soient équivalents, l'approche utilisant les ESN nous semble plus performante car plus générique. En effet, rien qu'en fournissant la suite d'actions et d'états en entrée du réseau, l'état interne donne une représentation markovienne contenant l'historique sans qu'aucune connaissance experte liée au domaine considéré (ici le tutorat) ne soit apportée. Seules des hypothèses génériques sur l'architecture du réseau, indépendantes de la tâche traitée, sont faites. En revanche, l'information ajoutée à l'observation pour que toutes les données nécessaires à la décision soient contenues dans l'état courant n'est pas garantie comme étant suffisante pour rendre l'état markovien dans le cas de la politique informée.

Pour vérifier l'adaptabilité des stratégies apprises aux différents profils d'étudiants, la moyenne des récompenses pondérées cumulées obtenues pour chacun de trois types est présentée pour les stratégies informée et se basant sur les ESN Figure 9.7. Il est rappelé que toutes les stratégies

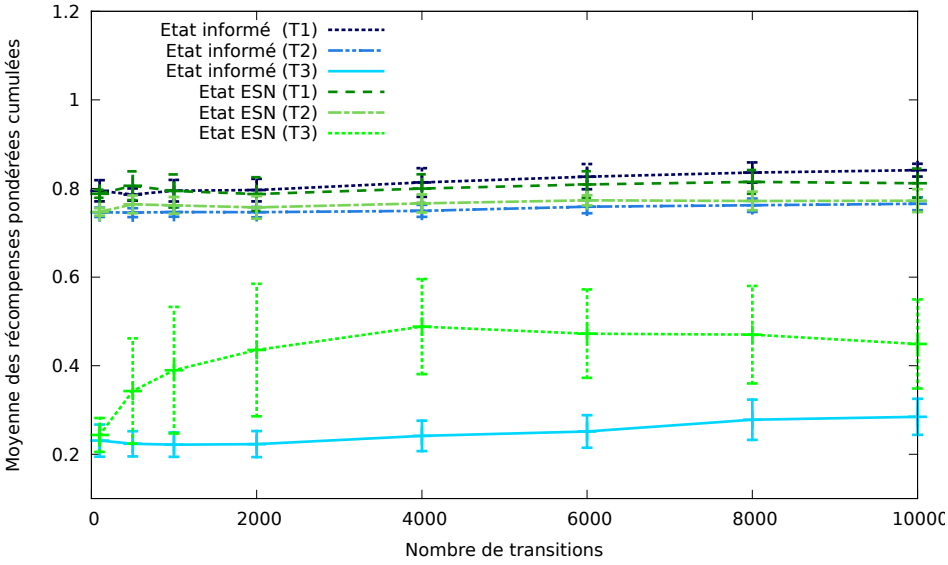



**Figure 9.7** – Résultats pour chacuns des groupes d'étudiants.

sont entraînées sur les mêmes données mais un seul type d'étudiant est testé à la fois. Pour les trois types d'étudiants, les résultats en moyenne sont supérieurs avec l'approche utilisant les ESN. La différence est la plus nettement marquée pour les étudiants du groupe ayant le plus de difficulés à apprendre. Pour comprendre cet écart, les actions proposées pendant le test des politiques apprises sont fournies Figure 9.8.

La comparaison révèle que le nombre de leçons est plus important dans le cas où les ESN sont employés (autour de trois) alors qu'une seule leçon est donnée en moyenne dans le cas informé. Cette stratégie est plus intéressante pour les étudiants du troisième groupe et donc aussi pour ceux des premier et deuxième groupes. En effet, dans le cas informé, la politique apprise est seulement bénéfique pour les élèves du premier groupe. C'est moins avantageux puisque ces élèves sont déjà à l'aise avec l'apprentissage (cette conclusion est cohérente avec les résultats de la Figure 9.7). D'autre part, lorsque les ESN sont utilisés, le nombre de leçons donné est légèrement supérieur pour les élèves du troisième groupe par rapport à ceux du premier groupe d'environ une demi leçon en moyenne, ce qui n'est pas le cas avec l'approche informée. Cela peut s'expliquer par le fait que la représentation d'état est beaucoup plus riche et qu'alors des différences subtiles entre les comportements peuvent être détectées. Nous remarquons qu'aucune question n'est posée lors du test de la politique. Ceci pourrait s'expliquer par le fait que lors de l'apprentissage, la machine apprend le nombre correct de leçons à donner grâce aux questions, qui constituent un bon indicateur des connaissances de l'élève. En revanche, lors du test, une fois que le nombre de leçons à donner est appris, les questions ne sont plus posées car elles diminuent la récompense d'un facteur $\gamma$. Un prolongement des travaux dans cette direction pourrait être envisagé.

## 9.4 Conclusion

La preuve de concept présentée dans ce chapitre donne des résultats prometteurs pour l'application des ESN aux problèmes partiellement observables. Elle illustre la quatrième hypothèse proposée dans l'introduction selon laquelle le retour au MDP, qui est un modèle plus simple

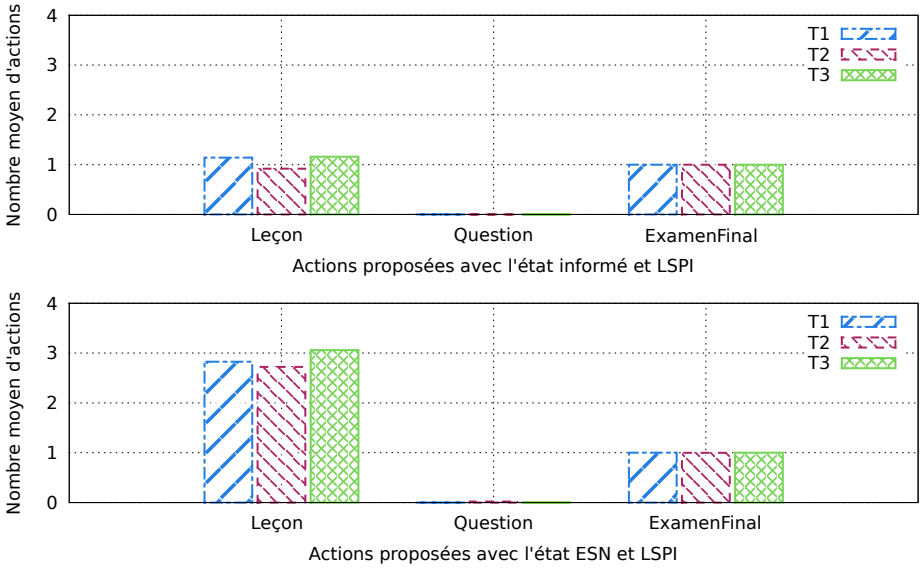



**Figure 9.8** – Distribution des actions proposées (la taille des jeux de données d'entraînement est fixée à 10000).

et qui suppose moins d'hypothèses pour appliquer le RL, est possible en construisant automatiquement une représentation markovienne d'état basée sur une mémoire. Même s'il n'est pas possible de s'affranchir totalement d'un modèle, comme par exemple le choix du nombre de neurones dans la méthode proposée, aucune connaissance experte n'est apportée ici sur le choix des informations nécessaires à retenir pour prendre la bonne décision. Elles sont automatiquement slectionnées.

Pour donner plus de poids à cette approche, il serait intéressant de la tester sur des corpora d'interactions réelles entre étudiants et tuteurs automatiques, d'autant plus qu'elle a été choisie pour fonctionner hors-ligne afin de pouvoir réutiliser des données déjà collectées. En effet, la complexité du problème présenté ici, bien que suffisante pour tirer les premières conclusions, ne permet pas de valider l'approche sans modèle pour résoudre un POMDP dans le cadre du tutorat. De plus, jouer sur la taille de l'ESN permettrait d'affiner la recherche de politiques. Dans les exemples précédents, si leur taille était supérieure, peut-être que l'apprentissage s'en sortirait mieux pour faire la distinction entre les types d'élèves. Nous avons abordé la question du choix du nombre de neurones sur la couche cachée dans [Daubigney 13d], sur un problème de prédiction de série temporelle, donc sans RL. Il est intéressant d'avoir beaucoup de neurones dans le réservoir pour apprendre des dynamiques complexes. En conséquence, le nombre de paramètres à apprendre augmente. Cela peut devenir problématique si le nombre de données à disposition pour effectuer l'apprentissage est faible. Une solution consiste alors à combiner les ESN avec des projections aléatoires [Achlioptas 03]. Elles permettent de réduire le nombre de paramètres tout en fournissant des garanties sur la quantité d'information perdue. Il serait intéressant d'appliquer la combinaison ESN et projections aléatoires au problème de RL.

# Chapitre 10

# Conclusion générale

La première partie de ce chapitre est consacrée au résumé des travaux et des avantages apportés par les solutions envisagées. La seconde propose une liste non-exhaustive de pistes de réflexions pour améliorer les interactions homme-machine.

## 10.1 Résumé des travaux

Les travaux de ce manuscrit se placent dans le cadre des interactions homme-machine. Ainsi que présenté en introduction, si un important travail de réflexion sur l'ergonomie physique des objets a été proposé, il n'en n'est pas de même pour ce que nous avons appelé l'ergonomie cognitive. En effet, il a fallu attendre les récents progrès réalisés dans le traitement de l'information pour pouvoir prendre en compte cette caractéristique. Même si certaines tâches ont été automatisées, elles n'ont consisté qu'à exécuter une suite d'instructions sans se soucier du contexe dans lequel elles sont effectuées.

Une importante difficulté pour définir une bonne solution au problème d'interaction réside dans la gestion de l'incertain liée à l'utilisateur humain et au medium de communication. En effet, le comportement n'est pas le même pour tout le monde, les intentions de l'utilisateurs restent cachés et il arrive parfois que l'information ne soit pas correctement transmise entre la machine et l'utilisateur. Nous rappelons les principales contraintes à prendre en compte pour une interaction naturelle. Elles concernent tout d'abord la généralisation de la solution. En effet, les situations auxquelles doit faire face la machine sont en nombre très grand, voire infini. Comme toutes les situations possibles ne seront jamais rencontrées dans la base d'entraînement, des solutions qui peuvent inférer un comportement là où elles n'ont jamais eu d'exemples sont à préférer. De plus, il est préférable d'employer une méthode qui traque la solution plutôt qu'elle ne converge vers elle. Cela permet de prendre en compte les différences entre les utilisateurs ainsi que les modifications de leur comportement sur le long terme, liées à l'habitude à utiliser le système. Il est préférable d'employer des solutions pour lesquelles le type d'apprentissage peut être spécifié. Au concepteur du système de pouvoir choisir s'il préfère une solution en-ligne ou hors-ligne, *on-policy* ou *off-policy* ou une combinaison des deux s'il dispose de données déjà collectées ou s'il souhaite poursuivre un apprentissage de façon en-ligne après s'être assuré que de mauvaises décisions n'étaient pas testées avec une approche off-policy dont la politique de contrôle est définie par ses soins. De plus, la collecte des données d'apprentissage étant coûteuse, les solutions où la meilleure politique est apprise rapidement sont à privilégier. Cet aspect est pris en charge par une gestion efficace du dilemme exploration/exploitation. L'apprentissage par

renforcement est une méthode prometteuse pour résoudre les problèmes de prise de décisions séquentielles sous incertitude car la machine apprend la tâche à effectuer à partir des données sans que les étapes successives pour y parvenir ne lui soit spécifiées. Le RL peut être réutilisé sur des tâches variées sans qu'un important travail d'ingénierie ne soit à apporter pour le transfert de tâche. Cependant, comme toute méthode d'optimisation, elle suppose quelques hypothèses de résolution qui peuvent entraîner la perte de généricité et ainsi rendre la solution moins performante et la méthode difficilement transférable à d'autres problèmes sans connaissance *ad hoc* à la nouvelle tâche..

Quatre points ont été identifiés entraînant une perte de généricité de la solution par l'utilisation de modèles et donc la nécessaire introduction de connaissances *a priori*. Les travaux présentés dans ce manuscrit ont visé à proposer une solution plus générique pour chacun d'entre eux afin de faire reculer les contraintes successivement et proposer une méthode graduellement plus globale. Des solutions ont été proposées sur deux exemples d'interaction, les sytèmes de dialogue parlé et le tutorat assisté par ordinateur qui répondent à des problématiques différentes. Dans le premier exemple, la décision de la machine peut se baser sur l'analyse du signal vocal émis par l'utilisateur par des méthodes de reconnaissance de formes exhibant des taux de confiance sur leur résultats.. Dans le second exemple, la décision ne se base que sur l'observation d'un phénomène caché (la cognition) dont les mécanismes sont totalement cachées pour un observateur extérieur, ce qui complique la prise de décision.

Le premier point concerne l'utilisation habituellement proposées dans la résolution du problème, d'un modèle d'utilisateur. D'une part ce modèle permet de gérer l'incertain quant aux différences de comportement observées. Et d'autre part, il permet de générer suffisamment de données pour des algorithmes de RL gourmands. Cependant, si dans le cas du dialogue les modèles d'utilisateurs peuvent être appris par des techniques d'apprentissage automatique performantes par l'analyse de la construction du langage, il n'est pas possible d'atteindre la même fiabilité pour la cognition humaine. La solution proposée dans ce manuscrit pour s'affranchir du modèle est de proposer une gestion efficace du dilemme exploration-exploitation pour proposer un apprentissage rapide de la solution. Cette hypothèse est testée sur le problème de dialogue parlé et illustrée par l'algorithme KTD qui convient particulièrement aux interactions homme-machine.

Le deuxième point identifié concerne l'utilisation du cadre des processus décisionnels partiellement observables (POMDP). Ceux-ci sont utilisés pour gérer le fait que l'information sur l'utilisateur n'est pas complète et qu'une partie de l'historique doit être retenue pour que la décision courante soit efficace. Lorsque le modèle des observations est connu, c'est-à-dire les probabilités permettant de relier les observations aux états sous-jacents réels du système, ils permettent de trouver une bonne solution. Or, dans le cas de problèmes réels, le modèle d'observation n'est pas connu. Il faut alors l'estimer pour chaque problème d'interaction grâce à une connaissance experte. Le paradigme HIS en est un exemple. Cependant, ce modèle présente deux limitations : d'une part l'information contenue n'est pas garantie d'être markovienne et d'autre part, il est difficile à définir lorsque des problèmes impliquant la cognition humaine sont considérés. Pour gérer le premier point, une solution visant à augmenter la taille de l'espace d'état de croyance est proposée. Elle permet, à moindre coût grâce à une représentation non-linéaire de la fonction de valeur, d'utiliser plus d'information pour prendre la décision, ce qui améliore la prise de décision. Cette solution a été testée sur le problème de dialogue parlé.

Le troisième point identifié est la question du respect de la propriété de Markov. La plupart des algorithmes de RL utilisent les équations de Bellman qui ne sont définies que dans le cas où le modèle respecte la propriété de Markov. Or, lorsque l'espace d'état est construit à

partir de connaissances expertes, aucune garantie n'est apportée quand au respect strict de la propriété. Utiliser des algorithmes standards peut mener à la divergence des algorithmes. C'est pourquoi la solution de l'utilisation d'algorithmes en boîte noire a été envisagée. Pour ce faire, le problème d'optimisation d'un contrôleur a été mis sous la forme d'un problème d'identification des paramètres de la politique optimale. Cette solution a été testée sur le problème de dialogue parlé en utilisant le paradigme HIS pour fournir les informations nécessaires au gestionnaire de dialogue. Par les trois améliorations proposées jusqu'alors, un gain de généricité a été apporté : le modèle d'utilisateurs a été supprimé, et la contrainte du cadre Markovien a été assouplie. Il reste alors la question de la construction automatique de la représentation de l'espace d'état pour tout la chaîne du traitement de l'information soit uniquement en apprentissage machine. Cette approche est l'objet du quatrième point du manuscrit.

La dernier point est donc l'introduction d'une méthode de construction automatique de la réprésentation de l'espace d'état. Elle est basée sur les *Echo State Networks*. Elle est aprticulièrement adaptée au problème impliquant des mécanismes qui ne sont pas encore bien connus telle la cognition humaine. En effet, les hypothèses de modélisation restent modérées et ne sont que peu dépendantes du problème. Il n'y a pas de connaissance experte sur le problème traité introduite ce qui mène à un gain de généricité par rapport aux solutions existantes. Les observations sont fournies au réseau qui se charge de proposer une répresentation compacte de l'historique sans que ne lui soient spécifiées les caractéristiques les plus importantes à prendre en compte pour trouver la solution.

Les travaux proposés se poursuivent dans la lignée de ceux initiés par [Levin 97] pour limiter l'introduction de connaissances expertes en amont de la résolution du problème. En effet, les dernières expériences présentées dans ce manuscrit donnent des résultats prometteurs pour un apprentissage automatique sur toute la chaîne du traitement de l'information. Les résultats préliminaires obtenus sur un problème de tutorat sont à approfondir sur des tâches plus complexes et à partir d'utilisateurs et de problèmes de tutorat réels. Cela suppose la mise en place d'expériences à large échelle impliquant différents acteurs. Cela introduit alors d'autres défis : si des apprentissages réels sont mis en place, il faut définir le matériel pédagogique pour effectuer les expériences comme le type d'exercices pour tester les compétences, les leçons pour comprendre une notion. Cela implique de découper le savoir en briques élémentaires par des experts ce qui, à nouveau, introduit des hypothèses de modélisation à propos de la cognition humaine. Cela ouvre donc des portes à des travaux futurs, présentés dans la section suivante.

## 10.2 Pistes pour le futur

De ces travaux, il ressort quelques pistes immédiates de travail. D'une part, la piste sur les ESN n'a été testée que sur le problème de tutorat. Il serait intéressant de s'affranchir du paradigme HIS et de tester cette approche directement sur les informations retournées par le module de compréhension dans le cas du dialogue parlé. Cela permettrait de s'affranchir de toutes les connaissances expertes nécessaires pour obtenir un espace construit suffisant pour prendre la décision. Il reste alors la question dont la réponse est loin d'être triviale de choisir quelles informations sur le dialogue doivent être fournies au réseau pour que la prise de décision soit bonne.

Inversement, l'algorithme KTD n'a pas été testé sur le problème de tutorat. Cette approche vaudrait la peine d'être testée au vu des résultats intéressants fournis dans le cadre du dialogue. De plus, la taille de la représentation de la fonction de valeur proposée par les ESN peut devenir très importante étant donné que de la taille du réservoir dépend la profondeur

de la mémoire. La combinaison avec des projections aléatoires serait une piste pour traiter de ce problème. La combinaison ESN et projections aléatoires a été testée sur un problème de prédiction [Daubigney 13d] mais pas sur un problème d'apprentissage par renforcement. L'optimisation en boîte noire n'a pas non plus été testées sur le problème de tutorat.

Pour rendre l'apprentissage des politiques plus efficace, peut-être qu'il serait intéressant de former automatiquement des groupes d'utilisateurs afin de proposer un comportement de la machine adapté à chaque groupe. Cela est particulièrement vrai dans le cadre du tutorat où les différences de capacités entre les élèves sont importantes. Cette classification pourrait se faire en utilisant l'état interne des ESN puisqu'elle fournit une identification unique de la trace d'interaction pour chaque utilisateur. Cependant, cette classification se baserait sur le fait que des données représentatives de toute la population soient disponibles. Cela est loin d'être le cas. En effet, réunir suffisamment d'utilisateurs et les faire travailler sur les mêmes applications sur le long terme constitue un défi en soi. Avec le développement des applications sur internet, la collecte des données serait peut-être facilitée. Internet remet aussi en cause l'environnement classique de l'apprentissage avec, par exemple, des sites collaboratifs sur lesquels chaque utilisateurs expose son domaine de compétences et les élargit à d'autres domaines grâce aux compétences des autres utilisateurs.

La principale contrainte pour que ces méthodes soient profitables réside dans l'efficacité de la gestion de l'observabilité partielle. Des progrès viendront alors certainement des avancées réalisées en neurosciences pour comprendre les mécanismes de la cognition humaine et pourquoi pas de la compréhension de l'émergence de la conscience. Un pas dans cette direction pourrait être effectué par une approche multimodale, en utilisant toutes les informations possibles sur l'utilisateur sans faire de distinction entre les différentes catégories, comme par exemple une image des émotions de l'utilisateur pour déterminer sa nervosité, ou encore une trace de son regard pour vérifier la temps mis pour répondre. Cela pourrait se faire au moyen par exemple des *Deep Neural Networks*. Ces réseaux ont déjà été utilisés en reconnaissance vocale [Seide 11]. Il serait alors possible d'envisager un traitement automatique de toute la chaîne de l'information provenant de l'utilisateur, à partir du moment où elle est détectée jusqu'à ce que la décision lui soit retransmise sans que ne soit posé plus d'*a priori* qu'une architecture globale. Cela faciliterait ainsi la conception et le déploiement de machines intelligentes puisqu'alors des connaissances expertes ne seraient plus alors nécessaires pour modéliser chacuns des domaines concernés. Un déploiement plus large permettrait de donner une visibilité certaine au domaine des interactions homme-machine. Il serait aussi intéressant de profiter des quantités massives de données transitant quotidiennement sur l'internet pour concevoir des expériences en conditions réelles et des applications à large échelle. La question du nombre de données à collecter ne serait alors plus alors qu'un lointain souvenir.

# Bibliographie


[Achlioptas 03] D. Achlioptas. *Database-friendly random projections : Johnson-Lindenstrauss with binary coins*. Journal of computer and System Sciences, vol. 66, no. 4, pages 671–687, 2003.

[Ai 06] H. Ai & D. Litman. *Comparing real-real, simulated-simulated, and simulated-real spoken dialogue corpora*. In Proceedings of the AAAI Workshop on Statistical and Empirical Approaches for SDS, 2006.

[Amaral 11] L. Amaral & D. Meurers. *On using intelligent computer-assisted language learning in real-life foreign language teaching and learning*. ReCALL, vol. 23, no. 1, pages 4–24, 2011.

[Anderson 95] J.R. Anderson, A.T. Corbett, K.R. Koedinger & R. Pelletier. *Cognitive tutors : Lessons learned*. Journal of the Learning Sciences, vol. 4, no. 2, pages 167–207, 1995.

[Arroyo 99] I. Arroyo, J. Beck, K. Schultz & B.P. Woolf. *Piagetian psychology in intelligent tutoring systems*. In Proceedings of the International Conference on Artificial Intelligence in Education (AIED), pages 600–602, 1999.

[Arroyo 04] I. Arroyo, T. Murray, B.P. Woolf & C. Beal. *Inferring unobservable learning variables from students help seeking behavior*. In Intelligent tutoring systems, pages 244–270. Springer, 2004.

[Astrom 65] K.J. Astrom. *Optimal control of Markov decision processes with incomplete state estimation*. Journal of Mathematical Analysis and Applications, vol. 10, no. 1, pages 174–205, 1965.

[Baker 08] R. Baker, A. Corbett & V. Aleven. *More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing*. In Intelligent Tutoring Systems, pages 406–415. Springer, 2008.

[Baker 10] R. Baker, A. Goldstein & N. Heffernan. *Detecting the Moment of Learning*. In Intelligent Tutoring Systems, pages 25–34. Springer, 2010.

[Bakker 02] B. Bakker. *Reinforcement learning with long short-term memory*. Advances in neural information processing systems (NIPS), vol. 2, pages 1475–1482, 2002.

[Banach 22] S. Banach. *Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales*. Fund. Math., vol. 3, pages 133–181, 1922.

[Barnes 08] T. Barnes & J. Stamper. *Toward automatic hint generation for logic proof tutoring using historical student data*. In Intelligent Tutoring Systems, pages 373–382. Springer, 2008.

[Beck 97] J. Beck. *Modeling the student with reinforcement learning*. In Machine learning for User Modeling Workshop at the Sixth International Conference on User Modeling. Citeseer, 1997.

[Beck 98] J. Beck & B. Woolf. *Using a learning agent with a student model*. In Intelligent Tutoring Systems, pages 6–15. Springer, 1998.

[Beck 00a] J. Beck & B. Woolf. *High-level student modeling with machine learning*. In Intelligent tutoring systems, pages 584–593. Springer, 2000.

[Beck 00b] J. Beck & B. Woolf. *Reasoning from data rather than theory*. In Proceedings of the International Florida Artificial Intelligence Research Symposium (FLAIRS), 2000.

[Beck 00c] J.E. Beck, B.P. Woolf & C.R. Beal. *ADVISOR : A machine learning architecture for intelligent tutor construction*. In Proceedings of the National Conference on Artificial Intelligence (), pages 552–557, 2000.

[Beck 09] J. Beck & K. Chang. *Identifiability : A Fundamental Problem of Student Modeling*. User Modeling, pages 137–146, 2009.

[Bellman 57a] R. Bellman. Dynamic programming. Dover Publications, sixth edition, 1957.

[Bellman 57b] R. Bellman. *A Markovian decision process*. Journal of Mathematics and Machanics, vol. 6, pages 679–684, 1957.

[Bertsekas 95] D. P. Bertsekas & J. N. Tsitsiklis. *Neuro-dynamic programming : An overview*. In Proceedings of the Conference on Decision and Control, volume 1, pages 560–564. IEEE, 1995.

[Bloom 68] B.S. Bloom. *Learning for mastery*. Evaluation comment, vol. 1, no. 2, pages 1–5, 1968.

[Bloom 84] B.S. Bloom. *The 2 sigma problem : The search for methods of group instruction as effective as one-to-one tutoring*. Educational Researcher, vol. 13, no. 6, pages 4–16, 1984.

[Boots 12] B. Boots, A. Gretton & G. Gordon. *Hilbert space embeddings of PSRs*. In Proceedings of the NIPS Workshop on Spectral Algorithms for Latent Variable Models, 2012.

[Bos 03] J. Bos, E. Klein, O. Lemon & T. Oka. *DIPPER : Description and formalisation of an information-state update dialogue system architecture*. In Proceedings of the SIGdial workshop, pages 115–124, 2003.

[Bradtke 96] S.J. Bradtke & A.G. Barto. *Linear Least-Squares algorithms for temporal difference learning*. Journal of Machine Learning Research (JMLR), vol. 22, no. 1-3, pages 33–57, 1996.

[Carr 77] B.P. Carr & I.P. Goldstein. *Overlays : A theory of modelling for computer aided instruction*. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1977.

[Cen 05] H. Cen, K. Koedinger & B. Junker. *Automating cognitive model improvement by A* search and logistic regression*. In Educational data mining workshop at National Conference on Artificial Intelligence, 2005.

[Cen 06] H. Cen, K. Koedinger & B. Junker. *Learning Factors Analysis–A general method for cognitive model evaluation and improvement*. In Intelligent Tutoring Systems, pages 164–175. Springer, 2006.

[Chandramohan 11] S. Chandramohan, M. Geist, F. Lefevre & O. Pietquin. *User simulation in dialogue systems using inverse reinforcement learning*. In Proceedings of Interspeech, pages 1025–1028, 2011.

[Chang 06] K. Chang, J. Beck, J. Mostow & A. Corbett. *A Bayes net toolkit for student modeling in intelligent tutoring systems*. In Intelligent Tutoring Systems, pages 104–113. Springer, 2006.

[Chatzidimitriou 11] K.C. Chatzidimitriou, I. Partalas, P.A. Mitkas & I. Vlahavas. *Transferring Evolved Reservoir Features in Reinforcement Learning Tasks*. Proceedings of the European Workshop on Reinforcement Learning (EWRL), 2011.

[Cheng 88] H.T. Cheng. *Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, University of British Columbia, 1988.

[Chi 01] M.T.H. Chi, S.A. Siler, H. Jeong, T. Yamauchi & R.G. Hausmann. *Learning from human tutoring*. Cognitive Science, vol. 25, no. 4, pages 471–533, 2001.

[Chi 10a] M. Chi, K. VanLehn & D. Litman. *Do micro-level tutorial decisions matter : Applying reinforcement learning to induce pedagogical tutorial tactics*. In Intelligent Tutoring Systems, pages 224–234. Springer, 2010.

[Chi 10b] M. Chi, K. VanLehn, D. Litman & P. Jordan. *Inducing Effective Pedagogical Strategies Using Learning Context Features*. In Proceedings of the International Conference on User Modeling, Adaptation, and Personalization (UMAP), page 147. Springer-Verlag New York Inc, 2010.

[Chinaei 12] Hamid R Chinaei, Brahim Chaib-draa & Luc Lamontagne. *Learning observation models for dialogue POMDPs*. In Advances in Artificial Intelligence, pages 280–286. Springer, 2012.

[Clerc 11] M. Clerc. *Standard Particle Swarm Optimisation From 2006 to 2011*. Technical Report, 2011.

[Conati 02] C. Conati, A. Gertner & K. Vanlehn. *Using Bayesian networks to manage uncertainty in student modeling*. User Modeling and User-Adapted Interaction, vol. 12, no. 4, pages 371–417, 2002.

[Corbett 94] A.T. Corbett & J.R. Anderson. *Knowledge tracing : Modeling the acquisition of procedural knowledge*. User modeling and user-adapted interaction, vol. 4, no. 4, pages 253–278, 1994.

[Corbett 95] A. Corbett, J. Anderson & A. O'Brien. *Student modeling in the ACT Programming Tutor*. Cognitively diagnostic assessment, pages 19–41, 1995.

[Corbett 97] A. T. Corbett & A. Bhatnagar. *Student modeling in the ACT programming tutor : Adjusting a procedural learning model with declarative knowledge*. COURSES AND LECTURES-INTERNATIONAL CENTRE FOR MECHANICAL SCIENCES, pages 243–254, 1997.

[Cuayáhuitl 05] H. Cuayáhuitl, S. Renals, O. Lemon & H. Shimodaira. *Human-computer dialogue simulation using hidden markov models*. In Proceedings of the Automatic Speech Recognition and Understanding workshop (ASRU), pages 290–295. IEEE, 2005.

[Damper 01] R. Damper. Data-driven techniques in speech synthesis. MIT Press, 2001.

[Daubigney 11a] L. Daubigney, M. Gašić, S. Chandramohan, M. Geist, O. Pietquin & S. Young. *Uncertainty management for on-line optimisation of a POMDP-based large-scale spoken dialogue system*. In Proceedings of Interspeech, 2011.

[Daubigney 11b] L. Daubigney, M. Geist & O. Pietquin. *Apprentissage par renforcement pour la personnalisation d'un logiciel d'enseignement des langues*. In Actes de la Conférence sur les Environnements Informatiques pour l'Apprentissage Humain (EIAH), 2011.

[Daubigney 11c] L. Daubigney, M. Geist & O. Pietquin. *Gestion de l'incertitude pour l'optimisation en ligne d'un gestionnaire de dialogues parlés à grande échelle basé sur les POMDP*. In Journées Francophones de Planification, Décision et Apprentissage (JFPDA), 2011.

[Daubigney 12a] L. Daubigney, M. Geist, S. Chandramohan & O. Pietquin. *A Comprehensive Reinforcement Learning Framework for Dialogue Management Optimisation*. IEEE Journal of Selected Topics in Signal Processing, vol. 6, no. 8, pages 891–902, 2012.

[Daubigney 12b] L. Daubigney, M. Geist & O. Pietquin. *Apprentissage off-policy appliqué à un système de dialogue basé sur les PDMPO*. In Actes du Congrès francophone sur la Reconnaissance de Formes et l'Intelligence Artificielle (RFIA), 2012.

[Daubigney 12c] L. Daubigney, M. Geist & O. Pietquin. *Off-policy Learning in Large-scale POMDP-based Dialogue Systems*. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4989 – 4992, 2012.

[Daubigney 12d] L. Daubigney, M. Geist & O. Pietquin. *Optimisation d'un tuteur intelligent à partir d'un jeu de données fixé*. In Actes des Journées d'Etudes de la Parole (JEP), pages 241–248, 2012.

[Daubigney 13a] L. Daubigney, M. Geist & O. Pietquin. *Model-free POMDP optimisation of tutoring systems with echo-state networks*. In Proceedings of the SIGdial workshop, 2013.

[Daubigney 13b] L. Daubigney, M. Geist & O. Pietquin. *Optimisation par essaims particulaires de stratégies de dialogue*. In Journées Francophones de Planification, Décision et Apprentissage (JFPDA), 2013.

[Daubigney 13c] L. Daubigney, M. Geist & O. Pietquin. *Particle Swarm Optimisation of Spoken Dialogue System Strategies*. In Proceedings of Interspeech, 2013.

[Daubigney 13d] L. Daubigney, M. Geist & O. Pietquin. *Random Projections : a Remedy for Overfitting Issues in Time Series Prediction with Echo State Networks*. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013.

[Driemeyer 08] J. Driemeyer, J. Boyke, C. Gaser, C. Büchel & A. May. *Changes in gray matter induced by learning—revisited*. PLoS One, vol. 3, no. 7, page e2669, 2008.

[Dutech 03] A. Dutech & M. Samuelides. *Un algorithme d'apprentissage par renforcement pour les processus decisionnels de markov partiellement observes : apprendre une rextension selective du passé*. Revue d'Intelligence Artificielle (RIA), vol. 17, no. 4, pages 559–589, 2003.

[Eckert 97] W. Eckert, E. Levin & R. Pieraccini. *User Modeling for Spoken Dialogue System Evaluation*. In Proceedings of the Automatic Speech Recognition and Understanding workshop (ASRU), 1997.

[Engel 03] Y. Engel, S. Mannor & R. Meir. *Bayes meets Bellman : The Gaussian process approach to temporal difference learning*. In Proceedings of the International Conference on Machine Learning (ICML), volume 20, page 154, 2003.

[Engelbrecht 05] A. Engelbrecht. Fundamentals of computational swarm intelligence, volume 1. Wiley London, 2005.

[Fix 12] J. Fix & M. Geist. *Monte-Carlo Swarm Policy Search*. In Proceedings of the Symposium on Swarm Intelligence and Differential Evolution (SIDE), 2012.

[Foster 12] M.E. Foster, S. Keizer, Z. Wang & O. Lemon. *Machine learning of social states and skills for multi-party human-robot interaction*. Proceedings of the workshop on Machine Learning for Interactive Systems (MLIS), page 9, 2012.

[Garcia 07] F. Garcia, L.F. Hurtado, D. Griol, M. Castro, E. Segarra & E. Sanchis. *Recognition and understanding simulation for a spoken dialog corpus acquisition*. In Proceedings of the International Conference on Text, Speech and Dialogue (TDS), pages 574–581. Springer, 2007.

[Gašić 10] M. Gašić, F. Jurčíček, S. Keizer, F. Mairesse, B. Thomson, K. Yu & S. Young. *Gaussian processes for fast policy optimisation of POMDP-based dialogue managers*. In Proceedings of the SIGdial workshop, 2010.

[Gašić 11] M. Gašić, F. Jurčíček, B. Thomson, K. Yu & S. S. Young. *On-line policy optimisation of spoken dialogue systems via live interaction with human subjects*. In Proceedings of the Automatic Speech Recognition and Understanding workshop (ASRU), pages 312–317, 2011.

[Gašic 13] M. Gašic, C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis & S. Young. *On-line policy optimisation of Bayesian Spokoen Dialogue Systems via human interction*. In Proceedings of the International Conference on Accoustics, Speech and Signal Processing (ICASSP), 2013.

[Geist 09] M. Geist, O. Pietquin & G. Fricout. *Tracking in Reinforcement Learning*. In Proceedings of the International Conference on Neural Information Processing (ICONIP), volume 5863, Part I, pages 502–511, 2009.

[Geist 10] M. Geist & O. Pietquin. *Kalman Temporal Differences*. Journal of Artificial Intelligence Research (JAIR), vol. 39, pages 483–532, 2010.

[Geist 11] M. Geist & O. Pietquin. *Managing Uncertainty within the KTD Framework*. In Proceedings of the AL&E workshop, Journal of Machine Learning Research C& WP, 2011.

[Geist 13] M. Geist & O. Pietquin. *An Algorithmic Survey of Parametric Value Function Approximation*. IEEE Transactions on Neural Networks and Learning Systems, vol. 24, no. 6, pages 845 – 867, 2013.

[Georgila 05] K. Georgila, J. Henderson & O. Lemon. *Learning user simulations for information state update dialogue systems*. In Proceedings of Interspeech, 2005.

[Georgila 06] K. Georgila, J. Henderson & O. Lemon. *User simulation for spoken dialogue systems : Learning and evaluation*. In Proceedings of Interspeech, volume 2006. Citeseer, 2006.

[Golovin 04] N. Golovin & E. Rahm. *Reinforcement learning architecture for web recommendations*. In Proceedings of the International Conference on Information Technology : Coding and Computing (ITCC), volume 1, pages 398–402. IEEE, 2004.

[Graesser 05] A. Graesser, P. Chipman, B. Haynes & A. Olney. *AutoTutor : An intelligent tutoring system with mixed-initiative dialogue*. IEEE Transactions on Education, vol. 48, no. 4, pages 612–618, 2005.

[Hämäläinen 06] W. Hämäläinen & M. Vinni. *Comparison of machine learning methods for intelligent tutoring systems*. In Intelligent Tutoring Systems, pages 525–534. Springer, 2006.

[Hamilton 13] W. Hamilton, M. Fard & J. Pineau. *Modelling Sparse Dynamical Systems with Compressed Predictive State Representations*. In Proceedings of the International Conference on Machine Learning (ICML), pages 178–186, 2013.

[Heift 07] T. Heift & M. Schulze. Errors and intelligence in computer-assisted language learning : Parsers and pedagogues, volume 2. Psychology Press, 2007.

[Henderson 08] J. Henderson & O. Lemon. *Mixture model POMDPs for efficient handling of uncertainty in dialogue management*. In Proceedings of the Annual Meeting of the North American Association for Computational Linguistics on Human Language Technologies (NAACL/HLT), pages 73–76. Association for Computational Linguistics, 2008.

[Hochreiter 97] S. Hochreiter & J. Schmidhuber. *Long short-term memory*. Neural Computation, vol. 9, no. 8, pages 1735–1780, 1997.

[Hoey 07] J. Hoey, A. Von Bertoldi, P. Poupart & A. Mihailidis. *Assisting persons with dementia during handwashing using a partially observable Markov decision process*. In Proceedings of the International Conference on Vision Systems, volume 65, page 66, 2007.

[Iglesias 03] A. Iglesias, P. Martínez & F. Fernandez. *An experience applying reinforcement learning in a web-based adaptive and intelligent educational system*. Informatics in Education, vol. 2, no. 2, pages 223–240, 2003.

[Iglesias 09a] A. Iglesias, P. Martínez, R. Aler & F. Fernández. *Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning*. Applied Intelligence, vol. 31, pages 89–106, 2009.

[Iglesias 09b] A. Iglesias, P. Martínez, R. Aler & F. Fernández. *Reinforcement learning of pedagogical policies in adaptive and intelligent educational systems*. Knowledge-Based Systems, vol. 22, no. 4, pages 266–270, 2009.

[Jaeger 02] H. Jaeger. Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the" echo state network" approach. GMD-Forschungszentrum Informationstechnik, 2002.

[Janarthanam 11] S. Janarthanam, H. Hastie, O. Lemon & X. Liu. *The day after the day after tomorrow ? : a machine learning approach to adaptive temporal expression generation : training and evaluation with real users*. In Proceedings of the SIGdial workshop, pages 142–151. Association for Computational Linguistics, 2011.

[Jelinek 97] F. Jelinek. Statistical methods for speech recognition. MIT press, 1997.

[Jonsson 05] A. Jonsson, J. Johns, H. Mehranian, I. Arroyo, B. Woolf, A. Barto, D. Fisher & S. Mahadevan. *Evaluating the feasibility of learning student models from data*. In Proceedings of the AAAI Workshop on Educational Data Mining, 2005.

[Jurčíček 10] F. Jurčíček, B. Thomson, S. Keizer, F. Mairesse, M. Gašić, K. Yu & S. Young. *Natural belief-critic : a reinforcement algorithm for parameter estimation in statistical spoken dialogue systems.* In Proceedings of Interspeech, pages 90–93, 2010.

[Jurčíček 11] F. Jurčíček, S. Keizer, M. Gašić, F. Mairesse, B. Thomson, K Yu & S. Young. *Real User evaluation of spoken dialogue systems using Amazon Mechanical Turk*. Proceedings of Interspeech, 2011.

[Kaelbling 96] L. Kaelbling, M. Littman & A. Moore. *Reinforcement Learning : A Survey*. Journal of Artificial Intelligence Research (JAIR), vol. 4, pages 237–285, 1996.

[Kaelbling 98] L.P. Kaelbling, M.L. Littman & A.R. Cassandra. *Planning and acting in partially observable stochastic domains*. Artificial intelligence, vol. 101, no. 1, pages 99–134, 1998.

[Kalman 60] R.E. Kalman. *A new approach to linear filtering and prediction problems*. Journal of basic Engineering, vol. 82, no. 1, pages 35–45, 1960.

[Keizer 10] S. Keizer, M. Gašić, F. Jurčíček, F. Mairesse, B. Thomson, K. Yu & S. Young. *Parameter estimation for agenda-based user simulation*. In Proceedings of the SIGdial workshop, pages 116–123. Association for Computational Linguistics, 2010.

[Kennedy 95] J. Kennedy & R. Eberhart. *Particle swarm optimization*. In Proceedings of the International Conference on Neural Networks, volume 4, pages 1942–1948, 1995.

[Koedinger 97] K. Koedinger, J. Anderson, W. Hadley & M. Mark. *Intelligent tutoring goes to school in the big city*. International Journal of Artificial Intelligence in Education (IJAIED), vol. 8, pages 30–43, 1997.

[Koedinger 09] KR Koedinger, V. Aleven, I. Roll & R. Baker. *In vivo experiments on whether supporting metacognition in intelligent tutoring systems yields robust learning*. Handbook of Metacognition in Education. New York : Routledge, 2009.

[Kolter 09] J.Z. Kolter & A.N. Ng. *Near-Bayesian Exploration in Polynomial Time*. In Proceedings of the International Conference on Machine Learning (ICML), New York, NY, USA, 2009. ACM.

[Komatani 07] K. Komatani, T. Kawahara & H. Okuno. *Analyzing temporal transition of real user's behaviors in a spoken dialogue system.* In Proceedings of Interspeech, pages 142–145, 2007.

[Lagoudakis 03] M. Lagoudakis & R. Parr. *Least-squares policy iteration*. Journal of Machine Learning Research (JMLR), vol. 4, pages 1107–1149, 2003.

[Larsson 00] S. Larsson & D.R Traum. *Information state and dialogue management in the TRINDI dialogue move engine toolkit*. Natural language engineering, vol. 6, no. 3 & 4, pages 323–340, 2000.

[Lefevre 12] M. Lefevre, J. Broisin, V. Butoianu, P. Daubias, L. Daubigney, F. Greffier, N. Guin, S. Jean-Daubias, R. Monod-Ansaldi & H. Terrat. *Personnalisation de l'apprentissage : comparaison des besoins et approches à travers l'étude de quelques dispositifs*. Revue des Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation (Sticef), 2012.

[Lemon 07] O. Lemon & O. Pietquin. *Machine learning for spoken dialogue systems*. In Proceedings of Interspeech, pages 2685–2688, 2007.

[Lerch 11] J.P. Lerch, A.P. Yiu, A. Martinez-Canabal, T. Pekar, V.D. Bohbot, P.W. Frankland, R.M. Henkelman, S.A. Josselyn & J.G. Sled. *Maze training in mice induces MRI-detectable brain shape changes specific to the type of learning*. Neuroimage, vol. 54, no. 3, pages 2086–2095, 2011.

[Levin 97] E. Levin, R. Pieraccini & W. Eckert. *Learning dialogue strategies within the Markov decision process framework*. In Proceedings of the Automatic Speech Recognition and Understanding workshop (ASRU), pages 72–79. IEEE, 1997.

[Levin 98] E. Levin & R. Pieraccini. *Using Markov decision process for learning dialogue strategies*. In Proceedings of the International Conference on Accoustics, Speech and Signal Processing (ICASSP), 1998.

[Levin 00] E. Levin, R. Pieraccini & W. Eckert. *A Stochastic Model of Human-Machine Interaction for learning dialog Strategies*. IEEE Transactions on Speech and Audio Processing, vol. 8, no. 1, pages 11–23, 2000.

[Li 09a] L. Li, S. Balakrishnan & J. Williams. *Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection*. In Proceedings of Interspeech, volume 9, 2009.

[Li 09b] L. Li, S. Balakrishnan & J. Williams. *Reinforcement Learning for Dialog Management using Least-Squares Policy Iteration and Fast Feature Selection*. In Proceedings of Interspeech, 2009.

[Lin 92] L.J. Lin & T. Mitchell. *Memory Approaches To Reinforcement Learning In Non-Markovian Domains*. Rapport technique, School of Computer Science, Carnegie Mellon University, 1992.

[Litman 04] D. Litman & S. Silliman. *ITSPOKE : An intelligent tutoring spoken dialogue system*. In Demonstration Papers at HLT-NAACL 2004, pages 5–8. Association for Computational Linguistics, 2004.

[Littman 95] M.L. Littman, A.R. Cassandra & L. Pack Kaelbling. *Learning policies for partially observable environments : Scaling up*. In Proceedings of the International Conference on Machine Learning (ICML), pages 362–370. Citeseer, 1995.

[Littman 02] M.L. Littman, R.S. Sutton & S. Singh. *Predictive representations of state*. In Advances in Neural Information Processing Systems (NIPS), volume 2, pages 1555–1562. Citeseer, 2002.

[Lukosevicius 09] M. Lukosevicius & H. Jaeger. *Reservoir computing approaches to recurrent neural network training*. Computer Science Review, vol. 3, no. 3, pages 127–149, 2009.

[McCallum 95] A. McCallum. *Reinforcment learning with selective perception and hidden state*. PhD thesis, Dept. of Computer Science, University of Rochester, 1995.

[McCallum 96] A.K. McCallum. *Learning to use selective attention and short-term memory in sequential tasks*. In Proceedings of the International Conference on Simulation of Adaptive Behavior (SAB), 1996.

[Mostow 01] J. Mostow & G. Aist. *Evaluating tutors that listen : an overview of project LISTEN*. In Smart machines in education, pages 169–234. MIT Press, 2001.

[Murray 99] T. Murray. *Authoring intelligent tutoring systems : An analysis of the state of the art*. International journal of artificial intelligence in education, vol. 10, no. 1, pages 98–129, 1999.

[Newell 81] A. Newell & P.S. Rosenbloom. *Mechanisms of skill acquisition and the law of practice*. Cognitive skills and their acquisition, pages 1–55, 1981.

[Oubbati 11] M. Oubbati, M. Kächele & P.K.H.G. Palm. *Anticipating Rewards in Continuous Time and Space with Echo State Networks and Actor-Critic Design*. In Proceedings of the European Symposium on Artificial Neural Networks (ESANN), 2011.

[Paek 05] T. Paek & D. Chickering. *The markov assumption in spoken dialogue management*. In Proceedings of the SIGdial Workshop, 2005.

[Paek 06] T. Paek. *Reinforcement learning for spoken dialogue systems : Comparing strengths and weaknesses for practical deployment*. In Proceedings of the Dialog-on-Dialog Workshop, Interspeech, 2006.

[Paek 08] T. Paek & R. Pieraccini. *Automating spoken dialogue management design using machine learning : An industry perspective*. Speech Communication, vol. 50, no. 8, pages 716–729, 2008.

[Parr 95] R. Parr & S. Russell. *Approximating optimal policies for partially observable stochastic domains*. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), volume 14, pages 1088–1095. Citeseer, 1995.

[Perez 10] J. Perez. *Apprentissage artificiel pour l'ordonnancement des tâches dans les grilles de calcul*. PhD thesis, 2010.

[Pietquin 02] O. Pietquin & S. Renals. *ASR System Modeling For Automatic Evaluation And Optimization of Dialogue Systems*. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), volume 1, pages 45–48, 2002.

[Pietquin 03] O. Pietquin & T. Dutoit. *Aided Design of Finite-State Dialogue Management Systems*. In Proceedings of the International Conference on Multimedia and Expo, volume III, pages 545–548, Baltimore (USA, MA), July 2003.

[Pietquin 05] O. Pietquin & R. Beaufort. *Comparing ASR modeling methods for spoken dialogue simulation and optimal strategy learning*. Proceedings of Eurospeech, pages 861–864, 2005.

[Pietquin 06a] O. Pietquin. *Consistent goal-directed user model for realistic man-machine task-oriented spoken dialogue simulation*. In Proceedings of the International Conference on Multimedia and Expo, pages 425–428. IEEE, 2006.

[Pietquin 06b] O. Pietquin & T. Dutoit. *A Probabilistic Framework for Dialog Simulation and Optimal Strategy Learning*. IEEE Transactions on Audio, Speech and Language Processing, vol. 14, no. 2, pages 589–599, 2006.

[Pietquin 11a] O. Pietquin, L. Daubigney & M. Geist. *Optimization of a Tutoring System from a Fixed Set of Data*. In Proceedings of the ISCA workshop on Speech and Language Technology in Education (SLaTe), 2011.

[Pietquin 11b] O. Pietquin, M. Geist, S. Chandramohan & H. Frezza-Buet. *Sample-efficient batch reinforcement learning for dialogue management optimization*. ACM Transactions on Speech and Language Processing (TSLP), vol. 7, no. 3, page 7, 2011.

[Pietquin 11c] O. Pietquin & H. Hastie. *A survey on metrics for the evaluation of user simulations*. The Knowledge Engineering Review, 2011.

[Pietquin 11d] O. Pietquin, M. M. Geist & S. Chandramohan. *Sample Efficient On-line Learning of Optimal Dialogue Policies with Kalman Temporal Differences*. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2011.

[Pietquin 11e] O. Pietquin, F. Tango & R. Aras. *Batch reinforcement learning for optimizing longitudinal driving assistance strategies*. In Proceedings of the Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS), pages 73–79. IEEE, 2011.

[Pinault 11] F. Pinault & F. Lefevre. *Semantic graph clustering for POMDP-based spoken dialog systems*. Proceedings of Interspeech, pages 1321–1324, 2011.

[Pineau 03] J. Pineau, G. Gordon & S. Thrun. *Point-based value iteration : An anytime algorithm for POMDPs*. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), volume 18, pages 1025–1032, 2003.

[Png 12] S. Png, J. Pineau & B. Chaib-draa. *Building adaptive dialogue systems via Bayes-adaptive POMDPs*. IEEE Journal of Selected Topics in Signal Processing, vol. 6, no. 8, pages 917–927, 2012.

[Powell 87] Michael JD Powell. *Radial basis functions for multivariable interpolation : a review*. In Algorithms for approximation, pages 143–167. Clarendon Press, 1987.

[Puterman 94] M. L. Puterman. Markov decision processes : discrete stochastic dynamic programming. Wiley-Interscience, 1994.

[Rafferty 11] A.N. Rafferty, E. Brunskill, T.L. Griffiths & P. Shafto. *Faster teaching by POMDP planning*. In Proceedings of the conference on Artificial Intelligence in Education (AIE), pages 280–287. Springer, 2011.

[Rasmussen 06] C. Rasmussen & C. Williams. Gaussian processes for machine learning. The MIT Press, January 2006.

[Reiter 00] E. Reiter & R. Dale. Building natural language generation systems, volume 152. MIT Press, 2000.

[Roy 00] N. Roy, J. Pineau & S. Thrun. *Spoken dialogue management using probabilistic reasoning*. In Proceedings of the Annual Meeting on Association for Computational Linguistics, pages 93–100, 2000.

[Schatzmann 05] J. Schatzmann, M. Stuttle, K. Weilhammer & S. Young. *Effects od the user model on simulation-based learning of dialogue strategies*. In Proceedings of the Automatic Speech Recognition and Understanding workshop (ASRU), 2005.

[Schatzmann 07a] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye & S. Young. *Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System.* In Proceedings of the NAACL conference on Human Language Technologies, 2007.

[Schatzmann 07b] J. Schatzmann, B. Thomson & S. Young. *Error simulation for training statistical dialogue systems*. In Proceedings of the Automatic Speech Recognition and Understanding workshop, pages 526–531. IEEE, 2007.

[Schatzmann 07c] J. Schatzmann, B. Thomson & S. Young. *Statistical user simulation with a hidden agenda*. In Proceedings of the SIGdial workshop, pages 273–282, 2007.

[Scheffler 02] K. Scheffler & S. Young. *Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning*. In Proceedings of the International Conference on Human Language Technology Research (ICHLTR), pages 12–19. Morgan Kaufmann Publishers Inc., 2002.

[Seide 11] F. Seide, G. Li & D. Yu. *Conversational Speech Transcription Using Context-Dependent Deep Neural Networks.* In Proceedings of Interspeech, pages 437–440, 2011.

[Self 90] J.A. Self. *Bypassing the intractable problem of student modelling*. Intelligent tutoring systems : At the crossroads of artificial intelligence and education, pages 107–123, 1990.

[Singh 99] S. Singh, M. Kearns, D. Litman & M. Walker. *Reinforcement Learning for Spoken Dialogue Systems*. In Advances in Neural Information Processing Systems, 1999.

[Sison 98] R. Sison & M. Shimura. *Student modeling and machine learning*. International Journal of Artificial Intelligence in Education (IJAIED), vol. 9, no. 1-2, pages 128–158, 1998.

[Sondik 71] E.J. Sondik. *The Optimal Control of Partially Observable Markov Decision Processes*. PhD thesis, Stanford University, 1971.

[Sutton 88] R.S. Sutton. *Learning to Predict by the Method of Temporal Differences*. Machine Learning, vol. 3, pages 9–44, 1988.

[Sutton 98] R. S. Sutton & A. G. Barto. Reinforcement learning : An introduction, volume 1. Cambridge Univ Press, 1998.

[Swartout 10] W. Swartout, D. Traum, R. Artstein, D. Noren, P. Debevec, K. Bronnenkant, J. Williams, A. Leuski, S. Narayanan, D. Piepol*et al. Ada and Grace : Toward realistic and engaging virtual museum guides*. In Intelligent Virtual Agents, pages 286–300. Springer, 2010.

[Szita 06] I. Szita, V. Gyenes & A. Lőrincz. *Reinforcement learning with echo state networks*. Proceedings of the International Conference on Artificial Neural Networks (IJCNN), pages 830–839, 2006.

[Szita 07] I. Szita. *Rewarding Excursions : Extending Reinforcement Learning to Complex Domains*. PhD thesis, Eötvös Loránd University, 2007.

[Tetreault 06] J. Tetreault & D. Litman. *Using reinforcement learning to build a better model of dialogue state*. In Proceedings of the Conference of the European Association for Computational Linguistics, 2006.

[Thomson 10] B. Thomson & S. Young. *Bayesian update of dialogue state : A POMDP framework for spoken dialogue systems*. Computer Speech & Language, vol. 24, no. 4, pages 562–588, 2010.

[Thorndike 32] E. Thorndike. *The fundamentals of learning*. 1932.

[Thrun 00] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy*et al. Probabilistic algorithms and the interactive museum tour-guide robot minerva*. The International Journal of Robotics Research, vol. 19, no. 11, pages 972–999, 2000.

[VanLehn 03] K. VanLehn, S. Siler, C. Murray, T. Yamauchi & W.B. Baggett. *Why do only some events cause learning during human tutoring ?* Cognition and Instruction, vol. 21, no. 3, pages 209–249, 2003.

[Vanlehn 05] K. Vanlehn, C. Lynch, K. Schulze, J. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein & M. Wintersgill. *The Andes physics tutoring system : Lessons learned*. International Journal of Artificial Intelligence in Education (IJAIED), vol. 15, no. 3, pages 147–204, 2005.

[Vygotsky 78] L.S. Vygotsky. *Interaction between learning and development*. Mind in society : The development of higher psychological processes, pages 79–91, 1978.

[Walker 97] M. Walker, D. Litman, C. Kamm & A. Abella. *PARADISE : A Framework for Evaluating Spoken Dialogue Agents*. In Proceedings of Annual Meeting of the Association for Computational Linguistics, pages 271–280, 1997.

[Watkins 89] C.J.C.H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge, England, 1989.

[Weizenbaum 66] J. Weizenbaum. *ELIZA—a computer program for the study of natural language communication between man and machine*. Communications of the ACM, vol. 9, no. 1, pages 36–45, 1966.

[Whitehill 09] J. Whitehill. *Optimal teaching machines*. Work in progress, 2009.

[Williams 05a] J. D Williams & S. Young. *Scaling up POMDPs for Dialog Management : The"Summary POMDP"Method*. In Proceedings of the Automatic Speech Recognition and Understanding workshop (ASRU), pages 177–182, 2005.

[Williams 05b] Jason D Williams, Pascal Poupart & Steve Young. *Factored partially observable Markov decision processes for dialogue management*. In 4th Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI), pages 76–82, 2005.

[Williams 08] J. Williams. *Demonstration of a POMDP voice dialer*. In Proceedings of the NAACL conference on Human Language Technologies, Demo Session, pages 1–4. Association for Computational Linguistics, 2008.

[Wylie 09] R. Wylie, K.R. Koedinger & T. Mitamura. *Is self-explanation always better ? The effects of adding self-explanation prompts to an English grammar tutor*. In Proceedings of the Annual Conference of the Cognitive Science Society, pages 1300–1305, 2009.

[Young 06] S. Young, J. Schatzmann, B. Thomson, H. Ye & K. Weilhammer. *The HIS Dialogue Manager*. In Proceedings of the Spoken Language Technology workshop, 2006.

[Young 10] S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson & K. Yu. *The Hidden Information State model : A practical framework for POMDP-based spoken dialogue management*. Computer Speech & Language, vol. 24, no. 2, pages 150–174, 2010.