Contribution to multiagent planning for active information gathering

Jennifer Renoux

Université de Caen Basse-Normandie

Ecole doctorale : *SIMEM*

Thèse de doctorat

présentée et soutenue le : 18*/09/2015*

par

Jennifer RENOUX

pour obtenir le

Doctorat de l'Université de Caen Basse-Normandie

Spécialité : *Informatique et Applications*

Contribution to multiagent planning for active information gathering

Directeur de thèse : *Abdel-Illah Mouaddib*

Jury

Ouiddad Labbani-Igbida, Professeur, Université de Limoge (rapporteur)
Raja Chatila, Directeur de Recherche CNRS, ISIR Paris (rapporteur)
....
Amal El Fallah Seghrouchni, Professeur, Université Pierre et Marie Curie
Laurent Jeanpierre, Maître de Conférence, Université de Caen Normandie
Stephan Brunessaux, Senior Expert, Airbus Defence and Space (invité)
Simon Le Gloannëc, Docteur, Airbus Defence and Space (encadrant)
Abdel-Illah Mouaddib, Professeur, Université de Caen Normandie (directeur de thèse)

Remerciements

Je tiens à adresser toute ma reconnaissance au Professeur Abdel-Illah Mouaddib et à Simon Le Gloannëc pour m'avoir encadrée durant cette thèse. Leur patience, leurs encouragements et leurs conseils m'ont permis de mener ce travail dans de très bonnes conditions. Merci à Stephan Brunessaux de m'avoir accueillie dans son équipe et sans qui cette thèse n'aurait pas pu exister. Merci également à Dafni et Sylvain pour l'intérêt qu'ils ont montré à mes travaux et pour leurs conseils sur la réalisation de ma thèse. Un immense merci à Laurent pour toutes nos discussions dans son bureau, pour toutes ses questions très pointues, parfois difficiles à entendre lorsqu'elles ébranlent le modèle, mais toujours pertinentes et intéressantes. Tu pourras dire à Isabelle que le jour où j'ai écrit ces lignes, j'avais un pull violet. Merci à Bruno Z. pour ses conseils et pour ses mots d'encouragement. Un grand merci également à Arnaud pour les serveurs de calculs, mais surtout pour sa précieuse aide sur les différents problèmes techniques que j'ai pu rencontrer. Merci également à Esther pour toutes les relectures d'articles qu'elle a effectuée.

Je tiens à adresser tous mes remerciements à l'ensemble les membres de l'équipe IPCC pour la bonne ambiance, les pauses croissants, les discussions et tout ce qui m'a permis de me sentir à l'aise parmi vous pendant trois ans. Merci également aux membres de l'équipe de Caen pour leur bonne humeur et leurs encouragements en toute occasion. Merci Esther, Guillaume, Arthur, Clément, Arnaud, Simon pour les soirées jeux. Merci aux membres d'AS, à e-penser, Axolot, Vsauce, LinksTheSun, au Joueur du Grenier et tous les autres pour les moments de détente. De même, merci à Bastion, Journey, The Piano Guys et tous les autres pour les heures de musiques qui m'ont accompagnée durant la rédaction de ce manuscrit.

Du fond du cœur, merci Guillaume pour ton soutien, ta patience, tes encouragements. Pour avoir supporté ma mauvaise humeur et mes moments de doute. J'aurais pu dérailler mille fois si tu n'avais pas été là pour me remettre sur les rails.

Merci Caro pour nos fous-rires, nos balades, nos discussions, nos joies, nos doutes. Merci pour ces 25 ans d'amitié sincère, pour avoir toujours été là pour moi et pour avoir partagé avec moi les bons comme les mauvais moments. Avec une grande majorité de bon, fort heureusement. Merci Marion pour les week-end couture, pour les bons conseils et surtout pour être une frangine fantastique. Merci Gaël pour les discussions de geek, pour les parties de Mario kart et surtout pour être un frangin d'enfer.

Vielen Dank Uwe für deine Liebe. Danke, dass du mir gelehrt hasst, wie Go und Netrunner spielen. Danke auch für alle unsere Abende, Dr. Who, Sherlock oder Hikaru anzusehen. Und vor allem vielen Dank, dass du meine Leben schöner machst. Leider ist mein Deutsch nicht gut genug, um dir zu sagen, alles was ich wünschte. Ich liebe dich mein Schkatz.

Pour finir, je tiens à remercier infiniment mes parents à qui je dédie cette thèse. Vous m'avez toujours soutenue et encouragée quels que soient mes choix, vous avez toujours été là même dans les moments les plus durs, vous avez parcouru des centaines de kilomètres pour être près de moi quand j'avais besoin de vous. Vous m'avez appris à aimer ce que faisais et poussée à faire ce que j'aimais. C'est grâce à cela que j'en suis là aujourd'hui. Pour ceci et pour tout le reste, un immense merci à vous. Je vous aime.

Shoot for the moon,
even if you miss
you'll land among the stars
– Les Brown

Résumé

Dans cette thèse, nous considérons le problème de l'exploration d'événements. Nous définissons l'exploration d'événements comme le processus d'explorer un environnement topologiquement connu dans le but de récolter de l'information à propos d'événements se déroulant dans cet environnement. Les systèmes multiagents sont habituellement utilisés dans les applications de collecte d'informations, mais posent de nombreux problèmes tels que la coordination des agents et les communications entre agents. Notre travail propose un nouveau modèle décentralisé de planification multiagents pour la collecte d'informations. Dans ce modèle, appelé MAPING (Multi-Agent Planning for INformation Gathering), les agents utilisent un état de croyance étendu qui contient non seulement leurs propres croyances sur l'environnement, mais également des approximations des croyances des autres agents du système. Cet état de croyance étendu leur sert de base pour quantifier la pertinence d'une information, que ça soit pour eux ou pour un autre agent du système. Ils peuvent ainsi décider d'explorer l'environnement ou de communiquer une information à un autre agent en fonction de l'action qui apporte de plus d'information au système dans sa globalité. L'inconvénient majeur de ce modèle est sa complexité. En effet, la taille de l'espace des états de croyances augmente exponentiellement avec le nombre d'agents et la taille de l'environnement. Pour pallier ce problème, nous proposons un algorithme de résolution utilisant l'hypothèse classique et couramment adoptée de l'indépendance des variables.

Enfin, nous avons étudié le fait que l'exploration d'événements est un problème impliquant une exploration théoriquement infinie. Les agents doivent réévaluer leurs croyances régulièrement même après avoir atteint un bon niveau de croyance. Pour résoudre ce problème, nous proposons une fonction de lissage permettant aux agents d'oublier régulièrement les informations trop vieilles et pouvant être obsolètes.

Nous avons évalué notre approche sur différents scénarios s'inspirant de cas d'application réels. Les expériences ont montré la capacité de MAPING à effectuer une exploration efficace sous des contraintes de communication fortes.

Mots-clés: Planification décentralisée, Système multiagents, Pertinence de l'information

Abstract

In this thesis, we address the problem of performing event exploration. We define event exploration as the process of exploring a topologically known environment to gather information about dynamic events in this environment. Multiagent systems are commonly used for information gathering applications, but bring important challenges such as coordination and communication. This thesis proposes a new fully decentralized model of multiagent planning for information gathering. In this model, called MAPING (Multi-Agent Planning for INformation Gathering), the agents use an extended belief state that contains not only their own beliefs but also approximations of other agents' beliefs. With this extended belief state they are able to quantify the relevance of a piece of information for themselves but also for others. They can then decide to explore a specific area or to communicate a specific piece of information according to the action that brings the most information to the system in its totality. The major drawback of this model is its complexity: the size of the belief states space increases exponentially with the number of agents and the size of the environment. To overcome this issue, we also suggest a solving algorithm that uses the well-known adopted assumption of variable independence.

Finally we consider the fact that event exploration is usually an open-ended problem. Therefore the agents need to check again their beliefs even after they reached a good belief state. We suggest a smoothing function that enables the agents to forget gradually old observations that can be obsolete.

We evaluated our model on different scenarios inspired by real-type applications. These experiments show the ability of MAPING to tackle the event exploration problem with limited communications.

Keywords: Decentralized planning, Multiagent systems, Information relevence

Contents

Chapter 2
Representing uncertainty and sharing knowledge

Chapter 3
Planning under uncertainty

Chapter 8
Low communication cost

Chapter 11

Conclusion of part III

Partie IV   Conclusion and perspectives   **189**

Chapter 12

Synthesis

Chapter 13

Perspectives

Bibliography   **197**

Résumé étendu : Vers un modèle décentralisé d'agents autonomes pour la perception active

1 Introduction

Les systèmes multi-agents sont de plus en plus utilisés dans divers domaines et tout particulièrement en exploration et perception active. Durant cette thèse, nous nous sommes intéressés à la perception active dans des environnements dynamiques et avec communication restreinte. Dans tout type de mission, l'agent est intéressé par certaines caractéristiques de l'environnement, que nous nommons points d'intérêt. Nous considérons le problème de *l'exploration d'événements*, qui nous définissons comme le processus de parcourir un environnement topologiquement connu de manière à détecter tout événement se produisant dans cet environnement. Un événement est décrit comme la modification des caractéristiques d'un point d'intérêt. Par exemple, si l'ont considère comme point d'intérêt la position d'une cible, tout déplacement de cette cible sera un événement. L'exploration d'événements peut considérer de nombreuses applications réelles telle que des applications de surveillance, de maintenance industrielle ou encore des applications de recherche et sauvetage.

L'utilisation de systèmes multi-agents pour automatiser l'exploration d'événement semble intuitive, mais pose différents types de problèmes tels que la communication entre agents et la coordination d'agents hétérogènes. De nombreux systèmes multi-agents ont été développés pour de nombreuses applications. Beaucoup de ces systèmes partagent un point commun : ils ne considèrent pas la communication entre les agents comme un problème. Or la communication peut devenir un problème dans certains cas, par exemple lorsqu'un ennemi peut intercepter la communication ou dans des zones sinistrées où la bande passante est limitée. Durant nos travaux, nous avons développé un système multi-agents totalement décentralisé capable de fonctionner et de se coordonner tout en limitant ses communications au maximum. Le second problème concerne l'hétérogénéité des agents. Les capacités de perception des robots peuvent être très différentes et affectent grandement la façon dont les robots vont explorer et coopérer. Cependant, d'un point de vue du système, les données provenant des différents robots doivent être fusionnées pour obtenir une représentation complète et précise. Il est également possible pour les robots de préciser leurs connaissances et de vérifier des détections effectuées par d'autres robots du système. Nous nous sommes donc intéressés durant cette thèse au problème de la fusion de données hétérogènes dans un système de perception active.

1.1 Contributions de la thèse

Cette thèse a contribué à progresser dans le domaine de la perception active multi-agents et de la planification sous incertitude. Nos contributions majeures sont :

- la définition d'un *degré de pertinence* orienté agent afin de quantifier l'intérêt d'une information pour un agent donné sans que celui-ci n'exprime de requête

- MAPING (Multi-Agent Planning for INformation Gathering), un modèle de planification décentralisée multi-agents pour l'exploration d'événements. MAPING fonctionne sans agent central et chaque agent calcule sa propre politique d'exploration et de communication indépendamment des autres agents.

2 État de l'art

2.1 Perception active

La perception active a été définie par [Bajcsy, 1988] comme le problème d'optimiser les stratégies appliquées à l'acquisition de données, qui dépend de l'état courant du système et de la tâche à effectuer. Différents aspects font de la perception active un problème très complexe, notamment le fait que la solution dépend d'un critère multi-objectifs incluant le gain d'information et le coût des actions. Un autre problème majeur est que, par définition, l'environnement est partiellement observable et le robot fait face à de nombreuses incertitudes, tant dans son modèle et celui de l'environnement, mais également dans les données collectées par ses capteurs.

Dans le cas de la perception active multiagents, les robots doivent coopérer pour effectuer le plus efficacement leur mission. La stratégie utilisée dépend alors du type de mission posé. Deux cas de figure sont principalement étudiés dans la littérature :

1. le cas *antagoniste* : il s'agit typiquement les applications de patrouilles. Le système est confronté à un ou plusieurs ennemis et doit collecter de l'information afin de contrer ces ennemis. Dans le cas l'environnement est topologiquement connu et c'est le comportement des ennemis qui est exploré. C'est un cas d'exploration d'événement.

2. l'exploration de carte : dans ce cas l'environnement est topologiquement inconnu et le système doit en construire une carte.

Les problèmes de patrouille

Le cas de la patrouille, mono-agent comme multi-agents, a été grandement étudiée dans la littérature. Certaines études ne considèrent pas de modèle pour l'ennemi et ne font qu'optimiser la patrouille de manière à ce que le laps de temps entre deux explorations d'une même région soit aussi court que possible [Machado et al., 2003, Elmaliach et al., 2009]. Selon Agmon et al. [Agmon et al., 2008b, Agmon et al., 2011], ce type de stratégie n'est pas suffisant puisqu'un ennemi ayant connaissance de la trajectoire des robots ou la possibilité d'observer le système avant d'intervenir pourrait alors agir de façon à éviter les robots et ainsi tromper la patrouille. La recherche sur la théorie des jeux pour les problèmes de type patrouille est très active, notamment les jeux de Stackleberg [Fudenberg and Tirole, 1991]. Dans un jeu de Stackleberg, un *leader*, ici un patrouilleur, optimise sa stratégie en premier. Les *followers*, ici les adversaires, optimisent ensuite leurs propres stratégies en fonction de celle du leader. Ce type de solution a été considéré notamment par Basilico et al. [Basilico et al., 2009] qui définirent récemment un nouveau type de jeu nommé *Patrolling Security Games (PSG)* [Basilico et al., 2012], pour faire face spécifiquement aux problèmes de patrouille.

L'aspect multi-agents de la patrouille a également été soulevé assez tôt dans la littérature. Dans [Machado et al., 2003], Machado et al. a proposé plusieurs architectures pour des patrouilles multi-agents, qu'ils classent en fonction du type des agents (réactifs ou cognitifs), de la présence ou non de communication et de la stratégie de coordination (émergente ou centrale). Plus récemment, les POMDPs ont gagné en attention notamment dans les travaux de Paruchuri et al. [Paruchuri et al., 2006].

La cartographie

Le problème de la cartographie assez peu étudié comme un problème à part, mais énormément considéré en association avec la localisation dans le cadre du problème de cartographie et localisa-

tion simultanées. L'étude principale concernant la cartographie elle-même est celle de Yamauchi sur l'exploration fondée sur les frontières [Yamauchi, 1997]. Une frontière correspond à la limite entre une partie connue de la carte et une partie inconnue. L'idée de cette étude est que ce sont les frontières qui fournissent le plus d'information au système. Ces travaux ont été grandement étudiés dans le cadre de l'exploration multi-agents où le problème devient celui d'un allocation de tâche pour répartir l'ensemble des robots sur les frontières. Yamauchi a lui-même proposé une version multi-agents de son algorithme dans [Yamauchi, 1998]. De nombreuses études considèrent qu'un agent central distribue les cibles aux agents du système en fonction du coût nécessaire pour rejoindre la cible et du gain espéré [Simmons et al., 2000, Burgard et al., 2002]. Certaines études se sont également intéressées à la répartition des cibles de manière décentralisées. Dans [Bautin et al., 2011], la coordination est implicite et chaque agent choisit sa cible en fonction du gain espéré et du nombre de robots à proximité de la cible et donc susceptibles de la choisir. L'inconvénient est évidemment que plusieurs robots peuvent choisir la même cible. La plannification probabiliste a été suggérée, notamment dans [Matignon et al., 2012] ou un Dec-MDP est utilisé afin de calculer une stratégie qui minimise les interactions entre les robots et maximise la couverture de l'environnement par le système. Les filtres de Kalman et leurs extensions ont également été grandement utilisés pour la cartographie, à la fois dans le cadre mono et multi-agents [Kontitsis et al., 2013].

Autres types d'exploration

L'exploration est la découverte des caractéristiques inconnues d'un environnement et ne se limite donc pas à l'exploration de cartes. D'autres types de caractéristiques peuvent être explorés, comme la présence ou non de gaz et sa source [Loutfi et al., 2009] ou la détection de cibles multiples en mouvement [Chanel et al., 2013].

2.2   Représenter l'incertitude et partager les connaissances

Représenter l'incertitude

[Dubois, 2007] a défini trois types d'incertitude : 1. l'incertitude aléatoire, qui résulte de la variabilité du monde 2. l'incertitude épistémique, qui résulte d'une ignorance ou d'une incomplétude des connaissances 3. l'incertitude inconsistante, qui résulte d'un conflit entre différentes sources d'information Ces trois types d'incertitude sont par nature présente dans le monde réel. Ainsi, pour pouvoir y agir, un agent a besoin d'être doté d'une représentation de l'incertitude sur laquelle il va pouvoir appuyer son raisonnement. Différents modèles mathématiques de représentation de l'incertitude ont été développés, parmi lesquels les mesures de probabilités, les fonctions de croyance de Dempster-Shafer et les mesures de possibilité. Dans ce résumé nous ne présenterons que le modèle utilisé durant la thèse à savoir les mesures de probabilité. Le chapitre 2 détaille les trois modèles nommés ainsi que quelques autres moins utilisés.

Si l'on nomme $W = \{w_1, \ldots, w_n\}$ l'ensemble des mondes possibles, une mesure de probabilité assigne à chaque monde $w_i \in W$ un nombre – la probabilité – qui décrit la vraisemblance que ce monde soit le monde courant. Une mesure de probabilité respecte les contraintes suivantes :

$$\forall w_i \in W, P(w_i) \in [0, 1] \tag{1}$$

$$\forall w_i, w_j \in W, w_i \neq w_j, P(w_i \cup w_j) = P(w_i) + P(w_j) \tag{2}$$

$$\sum_{w_i \in W} P(w_i) = 1 \tag{3}$$

L'état de croyance d'un agent est défini par la mesure de probabilité que cet agent associe à l'ensemble des mondes possibles. L'état de croyance est la base du raisonnement d'un agent et doit être mis à jour à chaque fois qu'une nouvelle information arrive à l'agent. La méthode de mise à jour la plus utilisée est d'utiliser la règle de Bayes [Bayes, 1763] : $P(w_i|E) = \frac{P(E|w_i) \cdot P(w_i)}{P(E)}$, où

- w_i est le monde considéré dont la probabilité peut être affectée par l'arrivée de la nouvelle information

- E est la nouvelle information considérée

- $P(w_i)$ est la probabilité *a priori*, c'est à dire la probabilité de w_i avant de connaître E

- $P(w_i|E)$ est la probabilité *a posteriori*, c'est à dire la probabilité de w_i sachant E

- $P(E|w_i)$ est la *vraisemblance* de E, c'est à dire la compatibilité entre l'évidence E et le monde w_i

- $P(E)$ est la *vraisemblance marginale*

À l'initialisation de l'état de croyance, le *principe d'indifférence* est souvent considéré. Ce principe stipule qu'en l'absence de toute information sur le monde courant, il n'y a aucune raison de considérer un monde plus probable que les autres et que l'on devrait associer à chaque monde la probabilité $\frac{1}{n}$, n étant le nombre de mondes possibles $n = |W|$.

Dans de nombreuses applications, l'environnement peut être décrit en terme de variables, représentant les points d'intérêt de cet environnement. Chaque variable possède un ensemble de valeurs possible et un monde possible est une instanciation donnée de toutes les variables représentant le monde. L'ensemble des mondes possibles est donc l'ensemble de toutes les instanciations possibles des variables. Dans cette représentation factorisée, il est généralement impossible – et même non souhaitable – d'affecter une probabilité à chaque monde possible. Une méthode plus intuitive est d'affecter une probabilité à chaque valeur possible pour chaque variable. L'état de croyance représente donc l'ensemble des probabilités associées à chaque valeur possible pour chaque variable. Cependant, la mise à jour de l'état de croyance devient plus compliquée puisque la probabilité qu'une variable X ait pour valeur x peut dépendre des valeurs d'un ensemble $\mathcal{Y}$ d'autres variables. Les réseaux Bayesiens [Pearl, 1988, Neapolitan, 1990] permettent de représenter de façon pratique les dépendances entre des variables aléatoires. Un réseau Bayesien est un graphe acyclique orienté dont les nœuds sont labellisés avec les variables aléatoires et dont les arcs représentent une influence causale. Un réseau Bayesien spécifie également une unique distribution de probabilité sur l'ensemble de ses variables. À partir de cette représentation, différents algorithmes sont possibles pour inférer de nouvelles connaissances à partir d'une nouvelle information, c'est à dire de modifier la distribution de probabilité du réseau Bayesien à partir de l'évidence reçue [Chavira and Darwiche, 2005, Yedidia et al., 2005].

Pertinence de l'information

La pertinence d'une information est un concept que les humains manipulent tous les jours intuitivement. Il s'agit de sélectionner l'information à communiquer en fonction de son intérêt pour la personne qui la reçoit. La pertinence de l'information est étudiée dans le cadre de la *pragmatique* [Moeschler, 2007] et Grice a proposé dans ce cadre un *principe de coopération* que tout groupe d'agent souhaitant coopérer devrait appliquer afin d'échanger de l'information pertinente. Ce principe se décompose en quatre catégorie :

1. la quantité : la communication doit être aussi informative qu'il est nécessaire mais pas plus.

2. la qualité : ne doit pas être communiquée toute information que l'on sait fausse ou qui manque de preuves

3. la relation : l'information communiquée doit être en relation avec le sujet considéré

4. la méthode : la communication doit être simple à comprendre

Le principe de coopération a été réutilisé depuis afin de développer une théorie de la pertinence [Wilson and Sperber, 2002]. Néanmoins l'ensemble de ces considérations restent philosophiques et difficiles à appliquer dans des systèmes intelligents. Cependant de nombreuses études autour des systèmes de recherche d'information ont étudié la pertinence de l'information d'un point de vue computationnel afin de créer des systèmes plus performants. Borlund [Borlund, 2003] a notamment défini deux classes de pertinences pour la recherche d'information : la pertinence orientée système, qui analyse le nombre de correspondances entre la recherche effectuée et les documents du corpus, et la pertinence orientée utilisateur, qui est déterminée par un utilisateur en fonction de ses besoins, exprimés ou non.

Dans l'ensemble des études sur la pertinence en recherche d'information, une partie au moins des besoins de l'utilisateur sont exprimés en tant que requêtes à un système. Cependant, dans des systèmes multi-agents, les agents ne peuvent pas la plupart du temps exprimer leurs besoins en terme de requête. L'information doit simplement être échangée de façon à améliorer l'efficacité du système. Une autre théorie de la pertinence doit alors être envisagée pour permettre d'incorporer l'échange d'informations pertinente dans les systèmes multi-agents. Roussel et Cholvy [Roussel and Cholvy, 2009, Roussel, 2010] ont étudié ce type de pertinence, qu'elles nomment la pertinence orientée agent, pour des agents de type BDI (Belief-Desire-Intention). Elles notent en effet que la pertinence d'une information est souvent considérée par rapport à l'agent qui reçoit cette information mais rarement par rapport à l'agent qui doit évaluer a priori cette pertinence. Elles proposent donc un modèle basé sur la logique modale pour qu'un agent soit capable d'évaluer la pertinence d'une information par rapport à un autre agent.

2.3 Planification sous incertitude

La planification classique est historiquement la première forme de planification étudiée. Elle concerne les systèmes de transition d'états restreints [Ghallab et al., 2004], c'est à dire des systèmes à nombre d'états fini, complètement observables, déterministes, statiques, à buts restreints, et admettant des plans séquentiels comme solutions. Ce type de systèmes considèrent également le temps de manière implicite – les actions n'ont pas de durée – et ne considère aucun changement qui pourrait arriver dans l'environnement lors de l'exécution du plan. De nombreux planeurs ont été développés tels que les très connus General Problem Solver (GPS) [Newell et al., 1959] et STRIPS [Fikes and Nilsson, 1972]. La limite de ces planeurs a été très vite atteinte, à savoir qu'ils ne peuvent fonctionner que dans des mondes fermés et très codés. En effet l'ensemble des hypothèses décrites ci-dessus ne peut être considéré dès lors que l'on s'intéresse au monde réel.

L'incertitude a été introduite dans la planification par le développement de nouvelles logiques telle que la logique épistémique [Wright, 1951, Hintikka, 1962], la logique floue [Zadeh, 1988] et la logique multi-valuée [Béziau, 1997]. Les planeurs ont par la suite été modifié pour prendre en compte les contraintes d'incertitude. C'est notamment le cas de PKS [Petrick and Bacchus, 2002] capable de produire des plans conditionnels, adaptables à des connaissances incomplètes.

Néanmoins, la technique la plus couramment utilisée lorsqu'il s'agit de traiter de environnements incertains est la planification probabiliste.

La planification probabiliste permet de s'affranchir des contraintes de déterminisme, de plan séquentiels et de buts restreints. Le plus utilisés des modèles de planification probabiliste est le Processus Décisionnel de Markov (MDP) [Puterman, 1994]. Un MDP est un tuple $\langle \mathcal{S}, \mathcal{A}, H, T, \mathcal{R} \rangle$ avec

- $\mathcal{S}$ l'ensemble fini d'états possibles

- $\mathcal{A}$ l'ensemble fini d'actions possibles

- H l'ensemble des pas de décision

- $T : \mathcal{S}, \mathcal{A}, \mathcal{S} \to [0,1]$ la fonction de transition, avec $T(s,a,s') = P(s'|s,a)$

- $R : \mathcal{S}, \mathcal{A} \to \mathbb{R}$ la fonction de récompense

Dans ce tuple sont ainsi décrits l'environnement et sa possible évolution (ensemble des états et fonction de transition), les capacités de l'agent du système (l'ensemble d'actions) et le but à atteindre (la fonction de récompense). Le modèle a été étendu afin de traiter les environnements partiellement observables. Un Processus Décisionnel de Markov Partiellement Observable (POMDP) [Sondik, 1978] est un tuple $\langle \mathcal{S}, \mathcal{A}, \Omega, H, \mathcal{T}, \mathcal{O}, \mathcal{R}, b_0 \rangle$ avec,

- $\mathcal{S}$, $\mathcal{A}$, H, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$, $R : \mathcal{S}, \mathcal{A} \to \mathbb{R}$ identiques à ceux d'un MDP

- Ω l'ensemble fini d'observations

- $\mathcal{O} : \Omega \times \mathcal{S} \times \mathcal{A} \to [0,1]$ la fonction d'observation, où $\mathcal{O}(\omega, s, a) = P(\omega|s,a)$

- b_0 est l'état de croyance initial.

Dans un POMDP, l'agent possède un état de croyance similaire à celui décrit dans la section 0.2.2 : à chaque état $s \in \mathcal{S}$, l'agent associe une probabilité qui représente sa croyance sur le fait que l'état s soit l'état courant. L'état de croyance est mis à jour lorsque l'agent reçoit une observation – qui correspond ici à l'évidence mentionnée section 0.2.2 – en utilisant la mise à jour Bayesienne [Cassandra et al., 1994] :

$$
\begin{aligned}
b_o^a(s') &= \frac{O(\omega, s', a) \sum_{s \in \mathcal{S}} T(s,a,s')b(s)}{\sum_{s' \in \mathcal{S}} O(\omega, s', a) \sum_{s \in \mathcal{S}} T(s,a,s')b(s)} \\
&= \frac{O(\omega, s', a) \sum_{s \in \mathcal{S}} T(s,a,s')b(s)}{P(\omega|b,a)}
\end{aligned}
$$

Les MDPs, tout comme les POMDPs, sont adaptés à des systèmes mono-agent. Le passage au multi-agents apporte de nouveau un certain nombre de contraintes dépendant de la nature des relations entre les agents du système. Deux types extrêmes ont été définis dans [Poole and Mackworth, 2010]. Les agents sont dit entièrement coopératifs s'ils partagent la même fonction d'utilité et travaillent en coopération pour atteindre un but défini sur l'ensemble du système. Les agents sont dit entièrement compétitifs si l'un des agents ne peut gagner que si un autre perd. Toutes les variantes entre ces deux extrêmes peuvent être rencontrées. Dans la problématique de recherche d'information, nous nous intéressons uniquement au cas où les agents sont entièrement coopératifs.

Un Processus Décisionnel de Markov Partiellement Observable Décentralisé (Dec-POMDP) [Bernstein et al., 2002] est un modèle permettant de faire de planification sous incertitude avec un système multi-agents dont les agents sont entièrement coopératifs. Dans un Dec-POMDP, un agent central calcule la politique jointe du système et la distribue parmi les agents. Les agents exécutent ensuite leur propre politique et reçoivent leurs propres observations. Aucune communication explicite n'est effectuée. Néanmoins, les Dec-POMDPs permettent de gérer une forme de communication en passant par l'état de l'environnement, en considérant par exemple des actions permettant de modifier un paramètre de l'environnement et des observations rapportant cette modification. Les Dec-POMDPs ont été étendu en Dec-POMDP avec Communication (Dec-POMDP-COM) [Goldman and Zilberstein, 2003] afin de prendre en compte de manière explicite la communication. Dans un Dec-POMDP-COM, deux politiques sont calculées : une politique d'action et une politique de communication. Goldman et Zilberstein ont également montré qu'il était possible de transformer n'importe quel Dec-POMDP-COM en un Dec-POMDP équivalent.

Les POMDPs Interactifs (I-POMDP) [Gmytrasiewicz and Doshi, 2005] étendent les POMDPs en considérant que chaque agent possède des croyances sur l'environnement mais également des croyances sur les autres agents du système. Ces croyances sur les autres agents incluent non seulement leur propre état de croyance mais également leur modélisation complète (leurs actions, leurs états possibles, leur fonction de transition...). Doshi et al. ont proposé une représentation graphique pour les I-POMDPs, appelée *Diagramme d'Influence Interactif* (I-ID) afin de simplifier leur résolution [Doshi et al., 2009]. Néanmoins, et malgré les récentes études sur des techniques de résolution approchée [Rathnasabapathy et al., 2006, Doshi and Gmytrasiewicz, 2009], la résolution d'un I-POMDP reste très complexe et difficile à passer à l'échelle.

D'autre modèles de planification sous incertitude ont été développés spécialement pour le problème de collecte d'information. C'est par exemple le cas du modèle ρPOMDP décrit par Araya-Lopez et al. [Araya-Lopez et al., 2010], qui est une extension du modèle POMDP dans le but de permettre des fonctions de récompense définies sur les états de croyances et non uniquement sur les états du système. La plupart des modèles pour la recherche d'information utilise l'entropie négative de Shannon comme mesure de précision d'un état de croyance.

3 Contributions

3.1 Une théorie de la pertinence orientée agent

La pertinence d'un information correspond généralement à l'adéquation entre cette information et une requête. Mais quand il n'y a pas de requête exprimée, il peut être plus compliqué de quantifier cette pertinence. Il est dans ce cas indispensable de prendre en compte ce que l'agent sait déjà et ce que cette information lui apportera en plus. Durant cette thèse nous avons rassemblé certaines propriétés qu'une information doit présenter afin de pouvoir être considérée comme pertinente pour un agent donné.

Premièrement, une information doit être **correcte** pour être pertinente. Une information correcte est une information qui reflète l'état réel du système. En effet une information incorrecte entraînerait une dégradation des connaissances de l'agent qui la reçoit et ne peut donc pas être considérée comme pertinente pour cet agent.

Deuxièmement, la pertinence d'une information **dépend de l'agent qui la reçoit** et du **moment où il la reçoit**. En effet, la pertinence d'une information dépend des connaissances a priori d'un agent. Un autre agent avec d'autres connaissances n'aura donc pas le même intérêt pour cette information. De même, les connaissances d'un agent évoluant au cours du temps, une information pertinente à un instant donné pourra ne plus l'être quelques minutes plus tard.

Troisièmement, une information est pertinente si elle est **nouvelle** pour l'agent qui la reçoit. En effet, il est inutile de communiquer à un agent une information qu'il connaît déjà.

Enfin, une information est pertinente si elle **améliore les connaissances** de l'agent qui la reçoit. Dans la suite de cette thèse, nous appelons ce type d'information une information *précisante*, par analogie avec information nouvelles.

Les deux dernières propriétés peuvent paraître contradictoires. Si un agent dispose d'une connaissance a priori et qu'il reçoit une information qui vient contredire cette connaissance, cette information est considérée comme nouvelle mais remet en doute les connaissances de l'agent. Cependant, elle doit être considérée comme pertinente. En effet, une information est intéressante dans deux cas : si elle permet d'inférer de nouvelles connaissances et/ou si elle permet de confirmer des connaissances qui était incertaines.

En se basant sur ces propriétés, il est possible de définir un modèle mathématique de la pertinence orientée agent. Pour cela, il faut définir mathématiquement les connaissances d'un agent.

Un agent observe et croit

On considère que l'environnement est modélisé comme un ensemble de points d'intérêts. Chaque point d'intérêt est décrit comme une variable X_k dont les valeurs possibles appartiennent à $DOM(X_k)$.

$$\mathcal{E} = \bigotimes_{\forall k} X_k$$

$DOM(\mathcal{E})$ est l'ensemble de toutes les valeurs possibles pour tous $X_k \in \mathcal{E}$.

$$DOM(\mathcal{E}) = \langle DOM(X_k) \rangle_{\forall X_k \in \mathcal{E}}$$

Un agent possède des croyances à propos des point d'intérêt, notées $\mathcal{B}_i^{\mathcal{E}}$.

$$\mathcal{B}_{i,t}^{\mathcal{E}} = \bigotimes_{\forall X_k \in \mathcal{E}} b_{i,t}^k$$

où $b_{i,t}^k$ est une distribution de probabilité sur la variable X_k à l'instant t qui représente les croyances de l'agent i à propos de la variable X_k.

$$b_{i,t}^k = \bigotimes_{\forall x_p \in DOM(X_k)} P(X_k = x_p)$$

Deux propriétés sont prises en compte pour la définition d'un degré de pertinence : sa nouveauté et à quelle point elle est *précisante*. Nous considérons dans cette partie que toute information pour laquelle nous calculons un degré de pertinence est correcte. Certifier si une information est correcte ou non est un problème qui sera considéré dans la section 0.3.2.

Le terme *information* est ambigu quant à son utilisation : il est généralement associé à une grandeur indénombrable, mais souvent utilisé comme une entité dénombrable. Afin de résoudre cette ambiguïté, nous appellerons par la suite *observation* tout atome d'information qu'un agent peut envoyer ou recevoir et le terme *information* sera réservé à l'ensemble indénombrable des faits et éléments d'intérêt.

Cette observation est-elle nouvelle ?

On considère un agent i possédant à un instant t des croyances $\mathcal{B}_{i,t}$ et recevant une observation o. Les croyances de l'agent sont mises à jour et deviennent $\mathcal{B}_{i,t+1}$. L'observation o est considérée comme nouvelle pour l'agent i si elle modifie de façon significative les croyances de l'agent, c'est à dire si $\mathcal{B}_{i,t}$ et $\mathcal{B}_{i,t+1}$ sont significativement éloignés l'un de l'autre. Pour mesurer l'éloignement de deux distributions de probabilité, nous utilisons la distance de Hellinger [Nikulin, 2015], donnée par la formule

$$D_H(P,Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{j=1}^{n} \left(\sqrt{P(j)} - \sqrt{Q(j)} \right)^2} \tag{4}$$

À partir de la distance de Hellinger, nous définissons le degré de nouveauté d'une observation de la façon suivante :

Définition 0.1 (Degré de nouveauté)**.** Le degré de nouveauté d'une observation o reçue par un agent i à l'instant t est définie par

$$\begin{aligned}
nov_{i,t}(o) &= D_H(\mathcal{B}_{i,t}^{\mathcal{E}} || \mathcal{B}_{i,t+1}^{\mathcal{E}}) \\
&= \sum_{X_k \in \mathcal{E}} D_H(b_{i,t}^{k} || b_{i,t+1}^{k}) \\
&= \sum_{X_k \in \mathcal{E}} \frac{1}{\sqrt{2}} \sqrt{\sum_{x_p \in DOM(X_k)} \left(\sqrt{b_{i,t}^{k}(x_p)} - \sqrt{b_{i,t+1}^{k}(x_p)} \right)^2}
\end{aligned} \tag{5}$$

avec $\mathcal{B}_{i,t+1}^{\mathcal{E}} = update(\mathcal{B}_{i,t}^{\mathcal{E}}, o)$ l'état de croyance de l'agent i après la mise à jour avec l'observation o, $b_{i,t}^{k}$ est la distribution de probabilité de l'agent i concernant X_k, et $b_{i,t}^{k}(x_p)$ est la probabilité dans la distribution précédente que $X_k = x_p$.

Cette observation est-elle précisante ?

De même que précédemment, on considère un agent i possédant à un instant t des croyances $\mathcal{B}_{i,t}$ et recevant une observation o. Ses croyances sont mises à jour et deviennent $\mathcal{B}_{i,t+1}$. Une observation est dite précisante si elle permet à l'agent d'améliorer son état de croyances, c'est à dire si $\mathcal{B}_{i,t+1}$ est plus précis que $\mathcal{B}_{i,t}$. L'état de croyance d'un agent étant un vecteur de distributions de probabilité, pour mesurer sa précision il faut mesurer la précision de chaque distribution le composant. La mesure de précision d'une distribution de probabilité la plus connue et utilisée est l'entropie de Shannon, donnée par :

$$H_b(X) = -\sum_{j=1}^{n} P(j) \log_b P(j) \tag{6}$$

où X est une variable aléatoire possédant n valeurs possibles et P la distribution de probabilité associée à cette variable. À partir de l'entropie de Shannon, nous pouvons définir le degré de précision d'une observation pour un agent selon la définition 0.2.

Définition 0.2 (Degré de précision)**.** Soit un agent i possédant un état de croyance $\mathcal{B}_{i,t}^{\mathcal{E}}$ à l'instant t, et une observation o reçue par cet agent i. Le degré de précision de l'observation

o pour l'agent i est donné par :

$$sound_{i,t}(o) = H(\mathcal{B}_{i,t}^{\mathcal{E}}) - H(\mathcal{B}_{i,t+1}^{\mathcal{E}}) \tag{7}$$

où $\mathcal{B}_{i,t+1}^{\mathcal{E}} = update(\mathcal{B}_{i,t}^{\mathcal{E}}, o)$ est l'état de croyance de l'agent i après mise à jour avec l'observation o, et $H(\mathcal{B}_{i,t}^{\mathcal{E}})$ est l'entropie de l'état de croyance $\mathcal{B}_{i,t}^{\mathcal{E}}$ donné par la formule :

$$H(\mathcal{B}_{i,t}^{\mathcal{E}}) = \sum_{X_k \in \mathcal{E}} H(b_{i,t}^k) \tag{8}$$

Un degré de pertinence orienté agent

À partir du degré de nouveauté et du degré de précision définis précédemment, nous pouvons définir un degré de pertinence orienté agent.

Définition 0.3 (Degré de pertinence). Le degré de pertinence d'une observation o pour un agent i, noté $rel_i(o)$, est donné par

$$rel_{i,t}(o) = (1 - \delta)\frac{nov_{i,t}(o)}{|\mathcal{E}|} + \delta\frac{sound_{i,t}(o)}{H_{max}} \tag{9}$$

où $nov_{i,t}(o)$ - respectivement $sound_{i,t}(o)$ - est le degré de nouveauté - respectivement degré de précision - de l'observation o, $\delta \in [0, 1]$ est un poids permettant de modéliser la dynamique de l'environnement, et $H_{max} = \sum_{X_k \in \mathcal{E}} \log(|DOM(X_k)|)$ est l'entropie maximum de l'état de croyance, atteinte lorsque toutes les variables suivent une distribution uniforme.

$|\mathcal{E}|$ et H_{max} sont utilisées pour normaliser le degré de pertinence. Le paramètre δ dépend de l'application considérée et de la dynamique de l'environnement. Dans des environnements très dynamiques, δ sera proche de 0 tandis que dans des environnements plus statiques il sera proche de 1. δ est directement relié à la probabilité qu'un changement se produise dans le système : $\delta = 1 - ProbabilityOfChange$.

3.2 Le modèle MAPING

Le modèle Multi-Agent Planning for INformation Gathering (MAPING) permet l'exploration d'événements avec un système multi-agents tout en limitant le nombre de communications. Il est composé d'une partie hors-ligne et d'une partie en-ligne. La partie hors-ligne correspond au modèle de décision décentralisé basé sur les POMDPs et permet la création d'une politique d'exploration et de communication. La partie en-ligne est constituée du module d'exécution de la politique et du module de mise à jour de l'état de croyance de l'agent. Cette mise à jour inclut un mécanisme d'oubli des connaissances anciennes. En effet, l'exploration d'événements est un problème théoriquement infini et les agents doivent sans cesse remettre à jour leurs croyances. Pour ce faire, ils doivent oublier leur connaissances les plus anciennes afin d'explorer de nouveau les différents points d'intérêt de l'environnement.

Dans MAPING , chaque agent dispose de croyances sur l'environnement mais également de croyances sur les croyances des autres agents. Nous utilisons un état de croyance étendu défini par :

$$\mathcal{B}_{i,t} = \langle \mathcal{B}_{i,t}^{\mathcal{E}}, \mathcal{B}_{i,t}^{j,\mathcal{E}} \rangle \tag{10}$$

où $\mathcal{B}_{i,t}^{\mathcal{E}} = \langle b_{i,t}^{i,k} \rangle_{\forall X_k \in \mathcal{E}}$ est l'état de croyance de l'agent i sur l'environnement $\mathcal{E}$, et $\mathcal{B}_{i,t}^{j,\mathcal{E}} = \langle b_{i,t}^{j,k} \rangle_{\forall X_k \in \mathcal{E}}$ représente les croyances de l'agent i sur les croyances des autres agents du système sur l'environnement. Notons que ces dernières sont des approximations des croyances réelles des autres agents sur l'environnement.

Le POMDP de MAPING

Le cœur du module hors-ligne de MAPING est le processus de décision dont dispose chaque agent : un Processus Décisionnel de Markov Partiellement Observable. Un POMDP est un tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \Omega, \mathcal{R}, b_0 \rangle$ où $\mathcal{S}$ est l'ensemble d'états possibles, $\mathcal{A}$ l'ensemble des actions possibles, $\mathcal{O}$ l'ensemble des observations possibles, $\mathcal{T}$ la fonction de transition, Ω la fonction d'observation, $\mathcal{R}$ la fonction de récompense et b_0 l'état de croyance initial.

L'ensemble des états correspond à l'ensemble des instanciations jointes des variables $X_k \in \mathcal{E}$. Deux types d'actions sont considérés : les agents peuvent soit *explorer* leur environnement, soit *communiquer* une observation à un autre agent. À ces deux types s'ajoute une action *Idle*, qui correspond à ne rien faire. L'exploration d'un agent consiste à percevoir la valeur courant de l'une des variables X_k. Après cette perception, l'agent reçoit une observation qui le renseigne sur la valeur courante de cette variable. Pour des raisons de simplicité, nous ne considérons dans nos travaux que la communication d'une unique observation à un unique agent, bien que le modèle soit généralisable.

$$\mathcal{A} = \mathcal{A}_{Explore} \cup \mathcal{A}_{Communicate} \cup \{Idle\}$$
$$\mathcal{A} = \{Explore(X_k), \forall X_k \in \mathcal{E}\} \cup \{Communicate(o, ag), \forall o \in \mathcal{O}, \forall ag \in \mathcal{AG}\} \cup \{Idle\}$$

La fonction de transition décrit la dynamique du système et la fonction d'observation décrit les observations qu'un agent peut recevoir et la fiabilité de ses capteurs. Ces deux fonctions dépendent de l'application et des capacités des agents considérés.

À chaque fois qu'une action est effectuée, l'agent doit mettre à jour son état de croyance étendu. Dans MAPING , celui-ci est mis à jour dans trois cas : 1. lorsque l'agent explore une variable 2. lorsque l'agent communique une observation à un autre agent 3. lorsque l'agent reçoit une communication d'un autre agent. Dans le premier cas, l'agent met à jour ses propres croyances sur l'environnement – $\mathcal{B}_{i,t}^{i,\mathcal{E}}$. Dans le second cas, l'agent met à jour uniquement ses croyances sur les croyances de l'agent a qui il communique – $\mathcal{B}_{i,t}^{j,\mathcal{E}}$. En effet, l'agent qui reçoit la communication va mettre à jour ses croyances en utilisant l'observation reçue. Afin que l'expéditeur de l'observation conserve une approximation correcte des croyances du récepteur, il doit la mettre à jour comme le fera le récepteur. Dans le troisième cas, l'agent met à jour à la fois ses propres croyances et celles de l'agent qui lui a envoyé la communication. En effet, l'observation est nouvelle pour l'agent qui la reçoit et il doit donc la prendre en compte. Il peut également supposer que l'agent qui lui a envoyé la communication a déjà pris en compte cette observation dans son état de croyance et doit donc mettre à jour l'approximation qu'il a de l'agent expéditeur.

Lorsque l'agent reçoit une observation, il doit mettre à jour ses croyances en tenant compte des caractéristiques de l'expéditeur et non de ses propres caractéristiques – fonction de croyance et de transition. Dans le cadre d'agents homogènes, les caractéristiques sont identiques pour tous les agents et le problème ne se pose pas. Cependant, dans le cas où les agents sont hétérogènes les fonctions de transition et d'observation sont propres à chaque agent et doivent être partagées. Ce problème a été résolu en pratique mais reste ouvert théoriquement. Nous discutons plus en détails du cas des agents hétérogènes dans la partie 5.6.

La fonction de récompense est le cœur du POMDP. Elle permet de définir les buts de l'agent. Dans notre cas, l'agent doit être récompensé s'il explore des variables qui peuvent lui donner des observations pertinentes et s'il envoie des observations pertinentes aux autres agents. La fonction de récompense se base donc sur les états de croyance et non sur les états réels du système. Étant donné que trois types d'actions sont possibles, la fonction est divisées en trois parties : $\mathcal{R}(\mathcal{B}_{i,t}, Idle)$, $\mathcal{R}(\mathcal{B}_{i,t}, Explore(X_k))$ and $\mathcal{R}(\mathcal{B}_{i,t}, Communicate(o,j))$. Le cas $\mathcal{R}(\mathcal{B}_{i,t}, Idle)$ est le plus simple : puisque l'agent ne fait rien, il ne reçoit aucune récompense et $\mathcal{R}(\mathcal{B}_{i,t}, Idle) = 0$. Pour les deux autres cas, l'agent doit déterminer la pertinence de l'observation qu'il souhaite recevoir ou envoyer. Comme évoqué dans la section 0.3.1, une observation doit être correcte pour être pertinente. Or, l'agent n'ayant par définition pas de vue complète du système, il ne peut définir avec certitude si une observation est correcte ou non. Il ne peut que croire en son exactitude. Pour cela nous définissons un degré de croyance qui représente la certitude qu'un agent a concernant l'exactitude d'une observation.

Définition 0.4 (Degré de croyance). Le degré de croyance d'un agent i concernant une observation o à un instant t est défini par

$$bel_{i,t}(o) = \sum_{s \in \mathcal{S}} \sum_{\alpha \in \mathcal{A}_{Explore}} \mathcal{B}_{i,t}(s) \Omega(o, s, \alpha)$$

où $\mathcal{B}_{i,t}(s)$ est la croyance de l'agent i que s soit l'état courant du système.

En utilisant ce degré de croyance, on peut définir une fonction de récompense pour les actions de type exploration et les actions de type communication.

$$\mathcal{R}(\mathcal{B}_{i,t}, Explore(X_k)) = \sum_{o \in \mathcal{O}} bel_{i,\alpha,t}(o) rel_i(o) - C_{Explore(X_k)} \tag{11}$$

où $C_{Explore(X_k)}$ est le coût associé à l'exploration.

$$\mathcal{R}(\mathcal{B}_{i,t}, Communicate(o,j)) = bel_{i,t}(o) rel_i(o) + (D_H(\mathcal{B}_{i,t}^{\mathcal{E}}||\mathcal{B}_{i,t}^{j,\mathcal{E}}) - D_H(\mathcal{B}_{i,t+1}^{\mathcal{E}}||\mathcal{B}_{i,t+1}^{j,\mathcal{E}})) \\ - C_{Communicate(o,j)} \tag{12}$$

où $C_{Communicate(o,j)}$ est le coût de communication de l'observation o à l'agent j et $\mathcal{B}_{i,t+1}^{j,\mathcal{E}} = update(\mathcal{B}_{i,t}^{j,\mathcal{E}}, o)$ est l'état de croyances de l'agent i mis à jour avec l'observation o.

Résoudre le POMDP

La fonction de récompense de MAPING étant définie sur les états de croyances et non sur les états réels, le POMDP ne peut pas être résolu à l'aide des techniques habituelles de résolution. De plus, la fonction de récompense n'est pas prouvée complexe. Il est donc impossible d'utiliser les techniques du ρPOMDP présenté dans [Araya-Lopez et al., 2010]. Cependant, il est possible de transformer le POMDP en un Belief-MDP équivalent et de résoudre ce Belief-MDP pour obtenir une politique basée sur les états de croyance. Le Belief-MDP obtenu est un tuple $< \Delta, \mathcal{A}, \tau >$ où :

- Δ est l'ensemble d'états de croyance, correspondant dans le POMDP à $\mathcal{B}_{i,t}$

- $\mathcal{A}$ est l'ensemble d'actions, identique à celui du POMDP

- $\tau : \Delta \times \mathcal{A} \to \mathbb{R}$ est la fonction de transition

La fonction de transition τ peut être directement obtenue à partir de la fonction de transition $\mathcal{T}$ et de la fonction d'observation Ω du POMDP :

$$\tau(\mathcal{B}_{i,t}, \alpha, \mathcal{B}_{i,t+1}) = \begin{cases} \sum_{s \in \mathcal{S}} \sum_{o \in U_t} \Omega(o, s, \alpha) \mathcal{B}_{i,t}(s) & \text{if } U_t \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

où $U_t = \{o \in \mathcal{O} \text{ tel que } \mathcal{B}_{i,t+1} = update(\mathcal{B}_{i,t}, o)\}$ est l'ensemble des observations permettant la transition depuis l'état $\mathcal{B}_{i,t}$ vers l'état $\mathcal{B}_{i,t+1}$ et où $\mathcal{B}_{i,t}(s)$ est la croyance de l'agent i que l'état courant du système est s. On peut résoudre ce Belief-MDP en discrétisant l'ensemble des états de croyance et en utilisant les algorithmes classiques de résolution. Cependant cette méthode de résolution n'est pas passable à l'échelle étant donné que la taille de l'espace des états de croyance augmente exponentiellement avec le nombre d'agents et le nombre de variables. Pour permettre la résolution de plus grands modèles, nous proposons donc d'utiliser une technique de séparation des variables indépendantes. Pour cela, nous définissons deux types d'indépendance :

1. L'indépendance des variables : la valeur d'une variable donnée ne dépend que des valeurs d'un sous groupe – possiblement vide – d'autres variables. Mathématiquement, on note cette hypothèse $\forall X, Y \in \mathcal{P}(\mathcal{E}), X \neq Y, X$ et Y sont indépendants, $\mathcal{P}(\mathcal{E})$ étant une partition de l'ensemble de variables $\mathcal{E}$

2. L'indépendance des observations : la probabilité de recevoir une observations donnée après avoir exécuté une action donnée ne dépend que des valeurs d'un sous-groupe de variables du système. Mathématiquement, on note cette hypothèse $\forall O \in \mathcal{P}(\mathcal{O})$, il existe un unique ensemble $X \in \mathcal{P}(\mathcal{E})$ tel que $\forall o \in O, P(o|s, a) = P(o|X, a)$.

Ces deux hypothèses permettent de décomposer le POMDP initial en un groupe de sous-POMDP où chaque sous-POMDP est un tuple $\langle \mathcal{S}_\ell, \mathcal{A}, \mathcal{O}_\ell, \mathcal{T}_\ell, \omega_\ell, \mathcal{R}_\ell, b_{\ell,0} \rangle$ où

- $\mathcal{S}_\ell$ est l'ensemble des instanciations jointes des variables $X_\ell \in \mathcal{P}(\mathcal{E})$

- $\mathcal{A}$ est le même espace d'actions que celui du POMDP global

- $\mathcal{O}_\ell \in \mathcal{P}(\mathcal{O})$ est l'ensemble des observations dépendant des variables X_ℓ, selon la propriété d'indépendance des observations

- $\mathcal{T}_\ell$ est la fonction de transition appliquée aux variables X_ℓ selon la propriété d'indépendance des variables

- ω_ℓ est la fonction d'observation appliquée aux variables X_ℓ

- $\mathcal{R}_\ell$ est la fonction de récompense appliquée aux variables X_ℓ

- $b_{\ell,0}$ est l'état de croyance initial

Chaque sous-POMDP peut être résolu indépendamment des autres en le transformant en un Belief-MDP tel que décrit précédemment. En plus de la politique optimale, les algorithmes de résolution devront stocker la fonction de valeur optimale $V^*(\mathcal{B}_{\ell,i,t})$ pour chaque sous-POMDP. Lors de l'exécution, l'agent pourra calculer l'action à effectuer avec l'algorithme 0.1. Dans cet algorithme, l'agent compare le gain espéré pour chaque action de chaque sous-POMDP. Ce gain est la somme du gain obtenu en effectuant cette action de ce sous-POMDP à cet instant et des gains obtenus en n'effectuant rien sur les autres sous-POMDPs. L'agent choisit ensuite une action au hasard parmi celle ayant le gain espéré le plus grand.

Data: sous politiques π_ℓ et fonctions de valeur associées V_{π_ℓ}, état de croyance courrant $\mathcal{B}_{i,t}$

Result: action à exécuter α_{opt}

1 $V_{max} = -Infinity$;

2 $\alpha_{opt} = null$;

3 $gainEspere = 0$;

4 $listeActionsOptimales = \{\}$;

5 foreach $\mathcal{B}_{\ell,i,t}$ *composant* $\mathcal{B}_{i,t}$ **do**

6 $gainEspere = V_{\pi_\ell}(\mathcal{B}_{\ell,i,t})$;

7   **foreach** $\mathcal{B}_{\ell',i,t}, \ell' \neq \ell$ **do**

8 $gainEspere = gainEspere + V_{Idle}(\mathcal{B}_{\ell',i,t})$;

9   **end**

10   Calculer le coût d'exécuter l'action $\pi_\ell(\mathcal{B}_{\ell,i,t})$, noté $C_{\pi_\ell(\mathcal{B}_{\ell,i,t})}$;

11 $gainEspere = gainEspere - C_{\pi_\ell(\mathcal{B}_{\ell,i,t})}$;

12   **if** $gainEspere > V_{max}$ **then**

13 $empty(listeActionsOptimales)$;

14 $listeActionsOptimales = \{\pi_\ell(\mathcal{B}_{\ell,i,t})\}$;

15 $V_{max} = gainEspere$;

16   **end**

17   **if** $gainEspere == V_{max}$ **then**

18 $listeActionsOptimales = listeActionsOptimales \cup \{\pi_\ell(\mathcal{B}_{\ell,i,t})\}$;

19   **end**

20 end

21 $\alpha_{opt} = choisirAleatoirementDepuis(listeActionsOptimales)$;

22 return α_{opt} ;

Algorithm 0.1: Choisir l'action à effectuer depuis l'ensemble des sous-politiques

Oublier les anciennes connaissances

Lorsqu'un agent explore un environnement dynamique, les observations qu'il reçoit peuvent être rapidement contredites par des changements dans l'environnement. Il est donc important qu'il révise ses croyances régulièrement en ré-explorant des variables déjà connues pour détecter d'éventuels changements. Pour cela nous avons mis en place un mécanisme d'oubli de croyance basé sur une application contractante. Après l'avoir mis à jour, un agent applique cette application contractante sur certaines distribution de son état de croyance afin de les rapprocher d'une distribution uniforme. Les distributions sur lesquelles est appliquée l'application contractante sont choisies en fonction de la fonction de transition. En effet, les variables dont la probabilité de changement est la plus forte sont contractées plus souvent. Ainsi, à chaque mise à jour de l'état de croyance, chaque variable a une probabilité d'être contractée égale à sa probabilité de changement. Nous appelons cette étape *lissage*. Toute fonction de lissage de MAPING respecte la définition 0.5.

Définition 0.5 (Fonction de lissage). Une fonction $f : [0, 1] \rightarrow [0, 1]$ est appelée fonction de lissage pour une variable X_k qui a pour domaine $DOM(X_k)$ si elle satisfait l'ensemble des contraintes suivantes :

$$\begin{cases} f(b_{i,t}^{j,k}(x_p)) \geq 0, \forall x_p \in DOM(X_k) & (13) \\ \sum_{x_p \in DOM(X_k)} f(b_{i,t}^{j,k}(x_p)) = 1 & (14) \\ f(\dfrac{1}{n_k}) = \dfrac{1}{n_k} & (15) \\ \left| f(b_{i,t}^{j,k}(x_p)) - \dfrac{1}{n_k} \right| \leq \left| b_{i,t}^{j,k}(x_p) - \dfrac{1}{n_k} \right|, \forall x_p \in DOM(X_k) & (16) \\ (b_{i,t}^{j,k}(x_p) - \dfrac{1}{n_k})(f(b_{i,t}^{j,k}(x_p)) - \dfrac{1}{n_k}) \geq 0, \forall x_p \in DOM(X_k) & (17) \end{cases}$$

où n_k correspond à $|DOM(X_k)|$ et $b_{i,t}^{j,k}(x_p)$ est la probabilité que la variable X_k prenne la valeur x_p dans l'état de croyance de l'agent.

Le choix de la fonction de lissage dépend de l'application considérée, mais une fonction linéaire correspond pour la plupart des applications. Nous ne considérons donc que ce cas dans cette thèse.

Théorème 0.1. Soit X_k une variable du POMDP et $n_k = |DOM(X_k)|$ le nombre de valeurs possibles de X_k. Soit $\mathcal{P}^k$ l'ensemble de toutes les distributions de probabilité possibles sur X_k. Soit $(f_{X_k})_{X_k \in \mathcal{X}}$ une famille de fonction linéaires. Chaque f_{X_k} est une fonction de lissage si et seulement si $\exists a_k \in [0, 1]$ tel que :

$$\begin{aligned} f_{X_k} : \quad & \mathcal{P}^k \rightarrow \mathcal{P}^k \\ & p \mapsto a_k \times p + \frac{1 - a_k}{n_k} \end{aligned}$$

Dans MAPING , lorsque l'agent met à jour son état de croyance, il commence par utiliser l'observation reçue ou envoyée puis utilise sa fonction de lissage tel que présenté dans l'algorithme 0.2.

Data: Croyances de l'agent i, notées $\mathcal{B}_{i,t}$, observation o, ensemble de fonctions de lissage $\{f_{X_k} \forall X_k \in \mathcal{E}\}$

Result: croyances de l'agent i, notées $\mathcal{B}_{i,t+1}$

1   **foreach** $X_k \in \mathcal{E}$ **do**

2     $\mathcal{B}_{i,t+1}^k = \left\langle update(b_{i,t}^{j,k}, o) \right\rangle_{j \in \mathcal{AG}}$;

3     $probabiliteDeChangement = \displaystyle\sum_{x_i, x_j \in DOM(X_k), x_i \neq x_j} T_{X_k}(x_{i,t}, x_{j,t+1})$;

4     $nombreAleatoire = choisirNombreAleatoire()$;

5   **if** $nombreAleatoire < probabiliteDeChangement$ **then**

6       $\mathcal{B}_{i,t+1}^k = \left\langle f_{X_k}(b_{i,t+1}^{j,k}) \right\rangle_{j \in \mathcal{AG}}$;

7   **end**

8   **end**

Algorithm 0.2: Mise à jour et lissage des croyances

4   Expériences et résultats

Nous avons validé notre modèle à l'aide de diverses simulations, toutes basées sur un scénario commun mais faisant varier les environnements. Nous avons considéré quatre environnements, chacun divisé en n zones. Dans chaque zone, deux événements peuvent se produire : la zone peut être vide ou non, et la zone peut être en feu ou non. Le système contient trois robots : un robot capable de détecter les visages, un robot capable de détecter les couleurs et un robot capable de détecter les températures. Pour chaque environnement, deux simulations ont été réalisées. La première simulation présente un environnement statique : les zones vides et en feu sont décidées au lancement de la simulation et ne varient pas. Dans la deuxième simulation, les zones vides et en feu sont décidées au lancement de l'application mais peuvent varier au cours du temps, selon la fonction de transition du système. Enfin, trois séries de simulations ont été faites : la première avec un coût de communication faible, la deuxième avec un coût de communication moyen et la troisième avec un fort coût de communication.

Les quatre environnements considérés sont :

- une maison : la carte a été basée sur une carte réelle de maison de plain-pied.

- un bâtiment dont certaines transitions entre les zones ne peuvent être réalisées que dans un sens (cas de portes coupe-feu par exemple)

- un environnement extérieur, où les transitions d'une zone à l'autre peuvent être très coûteuse en fonction de la topographie du lie

- le Palais de l'Élysée, qui permet de tester le passage à l'échelle du système

Nous avons défini quatre mesures d'évaluation pour tester l'efficacité de notre modèle. La première mesure est le nombre de croyances correctes. Ceci nous permet de vérifier que le système parvient à détecter les événements. La deuxième mesure est la distance entre les états de croyance des agents et un état de croyance dit *parfait*, dans lequel la croyance représentant la valeur courante de la variable est à 1 et toutes les autres sont à 0. Cette mesure nous permet de nous assurer que le système est capable d'avoir des croyances précises. La troisième mesure est le nombre de communications envoyées par agent. En effet, le but de notre système est d'explorer efficacement tout en réduisant le nombre de ses communications. Enfin, notre quatrième mesure

est la différence entre les approximations qu'un agent a des croyances des autres agents et les croyances réelles de ces autres agents. Cette mesure nous permet de nous assurer que notre système de mise à jour des approximations est efficace et que chaque agent peut calculer la pertinence d'une information à partir de bases correctes.

Les expériences ont montrées que notre modèle était capable d'effectuer une exploration efficace tout en limitant ses communications dans le cas où la communication est moyenne ou forte. Cependant, dans le cas d'une faible communication, le système a tendance à communiquer tout ce qu'il reçoit y compris des observations incorrectes dues à des erreurs de détection. Ces erreurs de détection ont donc tendance à se répandre dans le système et à dégrader ses performances globales. L'une des grosses limitations du système semble être son passage à l'échelle. En effet, dans le Palais de l'Élysée, le système composé de seulement trois agent ne parvient pas à obtenir des états de croyances satisfaisants dus à la taille de l'environnement. Cependant, il ne nous a pas été possible de tester le système avec plus d'agents dans la mesure où le nombre d'états augmente exponentiellement avec le nombre d'agents et que les POMDPs pour des systèmes à 4 ou 5 agents étaient impossibles à résoudre dans un temps raisonnable avec notre algorithme et nos ressources.

5   Conclusion

Nous présentons dans cette thèse un modèle de planification multi-agents décentralisé pour l'exploration d'événements. Ce modèle est basé sur le modèle des Processus de Markov Partiellement Observables et utilise un degré de pertinence orienté agent que nous avons défini. Dans la version complète du manuscrit (rédigé en anglais), nous présentons dans une première partie les travaux effectués dans le domaine de la perception active, de la représentation de l'incertitude et de la planification sous incertitude. La seconde partie présente l'ensemble de nos contributions et la troisième partie présente les expériences effectuées et les résultats obtenus. Le manuscrit se termine par une conclusion sur les travaux effectués et sur de possibles évolutions et travaux futurs.

Preamble

6   Introduction

Multi-robot systems have been proven useful in more and more applications, and especially in active sensing problems. In this specific kind of problems, information gathering is not only a means to reach a goal, but the goal itself. In this thesis, we are interested in multi-agent active sensing in dynamic environments under strong communication constraints. We consider that in any type of mission, an agent - human or robot - is interested to gather information about certain features of the environment, such as the presence of a foe, the moves of a target, the state of a pipe, etc. We call these features *features of interest*. We formally define *event exploration* as the process of travelling across a topologically known environment with the goal of detecting events happening in this environment. An event is defined as a change affecting the environment's features of interest. To refer to previous examples, changes could be an intrusion in a building, the move of a target or a pipe suddenly breaking. The concept of event exploration is not expressed in the literature, even if it can be related to the perception and the understanding parts of the Situation Assessment, as defined by Endsley in [Endsley, 1995, and, 2000]. The perception part tries to answer to the question "What are the facts ?" while the understanding part tries to answer to the question "What is going on ?". For both, active sensing is a major tool. A system performing active sensing acts in a way to optimize its input and collect as much interesting information as possible. Using multi-robot systems is an intuitive way to automatise efficiently the event exploration process. However, some problems remain open. During this work, we studied two open problems: (1) the problem of decentralized coordination under constrained communication (2) the problem of efficient active sensing with heterogeneous robots

Various multi-agent systems have been developed for various applications. Most of those systems have in common the fact that they do not consider the communication as a problem. However, in a lot of real application, the communication is not free and can be extremely constrained. This is for instance the case during exploration near enemy borders where the communication can be intercepted, or in crisis situations where the bandwidth is reduced. Considering constrained communication is therefore essential to be able to coordinate the robots in these kind of situations. During this thesis, we focused on developing a fully-decentralized system able to perform event exploration with a limited amount of communications.

The second problem we tackled in this work is the the problem of efficient event exploration with heterogeneous robots. The sensing capabilities of each robot affect a lot the way they will explore and their efficiency. However, from the system point of view, it is important that all the data collected are fused together in order to get a complete and accurate view of the environment. This problem is illustrated in the example described in the next session. In this work, we also addressed the problem of heterogeneous data fusion in an active sensing system.

7   Running example

Along this manuscript, we will refer to a single example to illustrate the concepts. We consider a problem of event exploration in an office building. A building is made up of 5 rooms. All the workers in this building need to leave before $8:00$pm. After this time, a system of heterogeneous robots[1] travels among the environment to ensure that all the rooms are empty and that there is no start of fire in the rooms. We consider three types of robots:

- *face-detection robots*: these robots are equipped with a camera and a face detection module. They are able to detect a face in a room with a probability p_f to be right.

---

[1]That is no say robots with different sensing capabilities

- *color-detection robots*: these robots are equipped with a camera and a color detection module. They are able to detect the colors corresponding to a fire with a probability p_c to be right.

- *temperature-detection robots*: these robots are equipped with a thermometer and a temperature detection module. They are able to detect any hotspot in a room with a probability p_t to be right. We assume that these robots can only detect if there are objects with a temperature higher than a given threshold without being able to detect the shape or the exact temperature of the object.

We can intuitively see that these robots need to work together and to fuse their information to be able to detect humans or fire. Indeed, the face-detection robot alone cannot detect fire and cannot make the difference between a real human and a picture of a human. Similarly, the color-detection robot alone cannot detect human and cannot make the difference between a real fire and a picture of a fire. The temperature-detection robot can make the difference between real human/fire and pictures of a human/fire thanks to their temperature, but, due to the assumption we made, it cannot discriminate between a human, a fire, a heater or a computer. However, fusing information from the different robots can create higher-level interesting information. If a face-detection robot detects a face in a room, a color-detection robot detects no fire color in the same room and a temperature-detection robot detects a hotspot in the same room, the system can globally concludes that a human is probably in the room. This conclusion could not be made without information fusion from heterogeneous robots.

The main advantage of this example is that it is rather simple while presenting all the aspects we wish to explore in this thesis: dynamics of the environment, heterogeneous agents, information fusion and uncertainty. It is obvious that the assumption made for the temperature-detection robots is not realistic at all, but it enables us to keep the system simple for the examples and the experimentation. In real-case applications, the robots will obviously be more sophisticated and able to infer more information, but the principle remains identical: several sensing capabilities can be combined to obtain higher-level and more accurate information.

8 Overview of our approach

Global overview

This dissertation proposes a framework called MAPING , for Multi-Agent Planning for INformation Gathering . This framework is made up of two parts: one offline and one online, both presented on Figure 1.

The offline part is the solver of MAPING . It takes a POMDP as input, which describes the dynamics of the environment and the capacities of the agents. From this POMDP, the solver produces a policy file, which describe the behavior of the agent. Details about the POMDP and policy can be found in Chapter 5. As indicated by its name, the offline part of the MAPING is performed before the start of the mission and the file produced remains untouched during the execution of the mission.

The online part is the execution module of MAPING . Each agent in the system is provided with one execution module, which is the same for all agents, regardless of their capacities. Given a policy file produced by the solver, the agent computes its next action. If the action involve some communication, the agent generates a message from the action and send this message to the agent dictated by the action. Then the agent get some information from its environment, if any, and updates its beliefs. If it received a message from another agent, this message is also
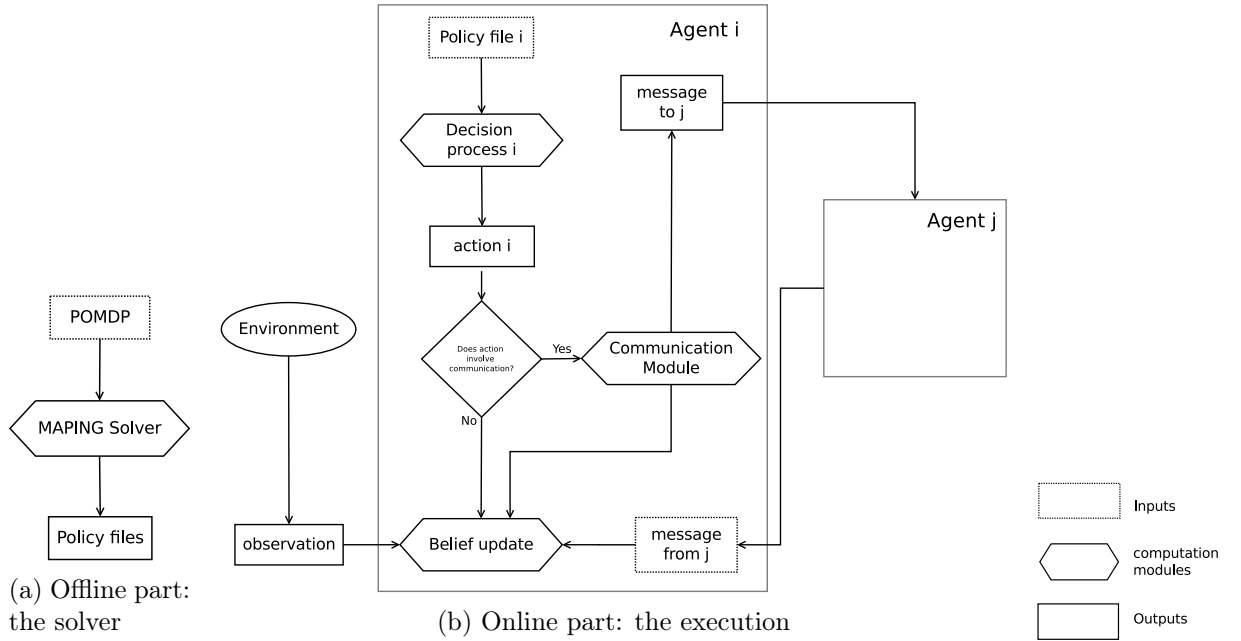
Figure 1: The offline and online part of the MAPING framework

considered during the beliefs update step. This process is then repeated again until the end of the mission.

Innovative aspects

This thesis contributed to advances in the fields of multiagent active sensing and multiagent planning under uncertainty. Principally, our main contributions are:

- the definition of a *degree of relevance* that quantifies the relevance of a given piece of information for a given agent. This degree of relevance is based on the agent's beliefs, expressed as probability distributions. This degree gives the opportunity to order different pieces of information regarding their relevance for a given agent at a given time.

- the Multi-Agent Planning for INformation Gathering framework (MAPING), a theoretical framework to perform fully decentralized multiagent event exploration. In MAPING , no central agent is assumed and all the agents compute their policy independently. MAPING enables the agents to compute a communication and an exploration strategy to explore efficiently their environment.

This work also led to implementations:

- a generic and reusable Java solver for MAPING POMDPs. The solver used in the experiments has been developed to be as generic as possible and can be re-used for various applications of the MAPING framework.

- a generic C++ ROS node to control the agents in MAPING applications. As for the solver, the ROS node used in the experiment has been developed to be as generic as possible. It takes the files outputted by the Java solver as inputs and can be re-used for all kinds of robots. Only the interface between the ROS node and the robot low-level control needs to be re-done, as it is specific to the robot.

Dissemination

The major points of our approach have been published in various national and international conferences:

- Renoux, J., Mouaddib, A.-I., and Le Gloannec, S. A distributed decision-theoretic model for multiagent active information gathering. *Multi-agent Sequential Decision Making under Uncertainty 2014 (MSDM'14)*.

- Renoux, J., Mouaddib, A.-I., and Le Gloannec, S. A distributed decision-theoretic model for multiagent active information gathering. In *Modeling Decisions for Artificial Intelligence (MDAI) 2014*. Springer.

- Renoux, J., Mouaddib, A.-I., and Le Gloannec, S. A distributed decision-theoretic model for multiagent active information gathering. *Workshop on Multi-Agent Coordination in Robotic Exploration (MACOREX'14)*.

- Renoux, J., Mouaddib, A.-I., and Le Gloannec, S. Un modèle de décision distribué pour la collecte d'information active multiagents. In *Actes de la conférence RFIA 2014*, France.

- Renoux, J., Mouaddib, A.-I., and Le Gloannec, S. A decision-theoretic planning approach for multi-robot exploration and event search. In *International Conference on Intelligent Robots and Systems (IROS'15)*, Hamburg.

and an international patent:

- Renoux, J., Mouaddib, A.-I., and Le Gloannec, S. Method for obtaining a system for active, decentralized multi-agent situation control. International Patent, WO-2015-079191-A2.

9   Outline of the document

The remaining of the thesis will be organized as follows. Part I presents a review of the literature concerning the fields of interest, that is to say *active sensing* (chapter 1), information theory (chapter 2) and planning under uncertainty (chapter 3).

After this review, part II presents the core theoretical contributions of this thesis. It is divided in two chapters. The first one presents a study of quantitative relevance and a formalization of a degree of relevance (chapter 4). The second chapter focuses on planning under uncertainty in multiagent information gathering and presents MAPING , a framework for fully-distributed multiagent planning for information gathering.

Part III is devoted to the practical implementation of MAPING and its evaluation through various experiments. The experiments implemented are based on the example described in section 0.7.

Part IV concludes this document with a review of the contributions, a summary of the open issues and suggestion for future research.

Part I

Review of the Literature

Chapter 1

Active Sensing

Contents

To be able to perform any mission, an agent needs to collect information about its environment. Environments becoming more and more complex, an agent cannot settle for sensing everything neither randomly. It needs to control its sensing in order to collect the most relevant information. This behavior is called *active sensing*[2]. Active sensing is currently studied in two major problems: patrolling and map-building. Both problems are useful in many application types such as search and rescue, intrusion detection, industrial maintenance... This chapter presents an overview of the studies existing for active sensing in patrolling and map-building.

---

[2]Also called sometimes *active perception*

1.1 Definition of Active Sensing

[Bajcsy, 1988] defined active sensing as a "problem of controlling strategies applied to the data acquisition process which will depend on the current state of the data interpretation and the goal or the task of the process". Then the active sensing problem have been deeply studied [Floreano and Mondada, 1994, Weyns et al., 2004] and applied to various domain such as surveillance and exploration.

Formally, [Mihaylova et al., 2002] defined active sensing as "the process of determining the inputs by optimizing a criterion, function of both costs and utilities" and formulated as following :

$$x_{k+1} = f(x_k, u_k, \eta_k) \tag{1.1}$$

$$z_{k+1} = h(x_{k+1}, s_{k+1}, \xi_{k+1}) \tag{1.2}$$

where x is the system state vector, f and h are nonlinear system and measurement functions, z is the measurement vector, η and ξ are respectively system and measurement noises. u stands for the input vector of the state function, s stands for a sensor parameter vector as input of the measurement function. The subscripts k and $k+1$ stand for steps. Based on this definition, the active sensing problem is markovian: the state at step $k+1$ only depends on the state at step k and not on the history $\{0..k-1\}$. The system's states are influenced by the input u (an example of input is the action taken by the agent) and the system's noises η (modeling for instance the possible failure of the action taken by the agent). The measurements depend on the system's states x, the parameters of the measurement function s (for instance a camera's focal length) and the measurement's noises ξ (modeling for instance the precision of the sensors).

The active sensing problem is very challenging for various reasons [Mihaylova et al., 2002] :

1. The robot and sensor models are nonlinear, and even if some methods linearize the models, all problems cannot be treated that way.

2. The task solution depends on an optimality criterion which is a multi-objective function weighting the information gain to some other utilities and costs. Multi-objective optimization is a challenging problem that often leads to computational explosion in time and number of operations.

3. Uncertainties in the robot model, the environment model and the sensor data need to be dealt with.

4. The system is often partially observable, that is to say that the sensors give information about some variables but not all.

In multi-agent active sensing, sensors in the system may be heterogeneous and need to cooperate to achieve the best active sensing possible. This kind of application raises new challenges such as robustness, communication complexity and performance requirements [Chung et al., 2004]. In the following section, we will explore the literature in different cases. Section 1.2 presents the patrolling problem. Section 1.3 presents a very well studied problem in which active sensing is capital : map exploration. Finally section 1.4 presents other less studied types of exploration in which active sensing is also used through different forms.

1.2 Adversarial settings : the patrolling problem

1.2.1 What is "patrolling" ?

Literally, to patrol is "keep watch over (an area) by regularly walking or traveling around or through it" [Oxford,]. Basilico et al. defined the patrolling task as a task in which an agent needs to perceive portions of a known environment in order to detect intrusion [Basilico et al., 2010].

Machado et al. described in [Machado et al., 2003] the patrolling problem in term of movement in a discrete graph, in which the nodes represent the zones to visit. In the case of continuous terrains, techniques such as skeletonization [Russell and Norvig, 2009] or Voronoi diagrams [Aurenhammer, 1991] are suggested to build a representing graph. Given this graph, the patrolling task consists in continuously visiting all the graph nodes so as to minimize the time lag between two visits. The authors also claimed that the patrolling task can exhibit slightly different characteristics according to the nature of the environment and the domain of application. For instance, for an environment containing mobile obstacles, the edges of the graph may change among time. Edges with different weights or nodes with different priorities can be considered to model a path more of less difficult and to visit more often some sensitive nodes. However, they focused their study on graphs with static and uniform weighted edges, and without nodes priority. Guo and Qu presented in [Guo and Qu, 2004] a mathematical formulation of the patrolling problem, in which the patrolling problem is considered as a problem of path planning.

Both representation (graphical and path-planning) have been widely reused and studied. However, and since path planning is not our main concern here, we will focus more on studies that adopt a decision-theoretic approach and so represent the patrolling problem as visiting all the nodes in a graph. Two major settings are usually considered in patrolling : non-adversarial setting, in which the patrolling agents need to visit the nodes and report changes, and adversarial settings, where the patrolling agents face a rational intruder that they have to detect.

A lot of algorithms have been suggested in the literature to find good patrolling strategies. However, the problem of evaluating a patrolling strategy is not straightforward since it may depend on the considered application. Given a discrete graph as described in their work [Machado et al., 2003], Machado et al. introduced three measures of patrolling quality : *idleness*, *worst idleness* and *exploration time*. The idleness of a graph is the average number of time steps between visits to all vertices. The worst idleness is the highest number of time steps that occurs during a simulation. The exploration time is the number of time steps necessary for an agent to visit at least once each node in the graph. These criteria, and particularly the idleness criterion, has been reused and adapted in more recent works. For instance, Elmaliach et al. suggested in [Elmaliach et al., 2009] three new criteria built on idleness and taking into account the frequency in which points in the area are visited.

Some studies [Agmon et al., 2008b,Agmon et al., 2011,Sak et al., 2008], argued that minimizing the time-lag between each visit of a node was not sufficient in adversarial settings. Indeed, the optimal deterministic patrolling path will fail miserably against an opponent that has knowledge about this path (for instance able to observe the patrolling agent for a while). Therefore it is mandatory to introduce some unpredictability in the patrolling and build non-deterministic patrolling strategies.

1.2.2 Single-agent patrolling

Studies about non-adversarial setting does not take into account a model for the intruders. They only try to optimize the patrolling path to reduce the time between each visit. In [Martins-Filho

and Macau, 2007], Martins et al. considers patrolling on arbitrary graphs, but the goal is to patrol edges[3] with the same frequency. They also introduced chaotic motion behavior through the use of an area-preserving chaotic map to ensure a high unpredictability of robot's trajectories and give the impression of erratic motion from an observer's point of view. In [Ruan et al., 2005], the authors considered a graph in which nodes have different priorities, based on incident rates, and a patrolling scheme that visits the node depending on their importance. The solution is generated thanks to a learning algorithm designed under a Markov Decision Process (MDP).

In adversarial settings, in which a model of the intruder is considered, Games Theory has gained a lot of interest recently, its main advantage being the possibility to take into account a model of the intruder in developing a patrolling strategy. Bayesian Games are commonly used to model scenarios in adversarial settings. In a Bayesian Game each agent may belong to one or more type, that determines its possible actions and payoffs. Usually, these games are analyzed in order to find a Bayesian equilibrium[4] [Fudenberg and Tirole, 1991]. However, this requires that all the players choose their strategy simultaneously. This assumption is not possible in many applications, for instance when the intruder have time to observe the patrolling agent and to adapt its strategy. For these cases, Stackleberg Games [Fudenberg and Tirole, 1991] are more appropriate. In a Stackleberg Game a leader, the patrolling agent, commits its strategy first and then a group of followers, the adversaries, optimize their own strategy considering the action chosen by the leader. The problem of choosing an optimal strategy for the leader in a Stackleberg Game has been proven to be NP-hard [Conitzer and Sandholm, 2006]. Paruchuri et al. proposed a heuristic to find exact solutions in such problems [Paruchuri et al., 2007] and an efficient algorithm to solve Stackleberg Games [Paruchuri et al., 2008]. However, in [Gatti, 2008], Gatti pointed out some inconsistencies in Paruchuri et al.'s work and suggested another model, consistent with game-theory, and an algorithm to solve it. Basilico et al. also considered the leader-followers strategies [Basilico et al., 2009] but applied it to infinite-horizon problems. They also defined the notion of Patrolling Security Games (PSG) [Basilico et al., 2012] to address this specific type of problems. A PSG is defined as a two-player game played by a patroller and a intruder on a graph-represented environment. The game is organized in turns during which the players act simultaneously. They theoretically studied this subclass of security games and provided algorithms to solve large instances with a single patroller and single intruder.

Randomization has been introduced in single-agent patrolling strategies to avoid predictability, for instance in [Lewis et al., 2004, Carroll et al., 2005]. However, no specific algorithm has been presented for the generation of randomized policies. Paruchuri et al. provided such an algorithm by using randomization in Fully Observable and Partially Observable Markov Decision Processes [Paruchuri et al., 2006, Paruchuri et al., 2009] and in Bayesian Stackleberg games [Paruchuri et al., 2009].

1.2.3 Multi-agent patrolling

Multiagent patrolling adds new challenges to the patrolling task. Indeed, the agents need to coordinate their decision-making with the purpose of achieving optimal group performance. One of the first work in this field is described in [Machado et al., 2003] where several architecture of multi-agent patrolling were presented and evaluated. The authors considered mainly homogeneous groups of agents, that is to say all the agents share the same architecture. Some parameters have been considered, presented in Table 1.1. This table presents only the architectures that have been chosen by the authors as the most appropriate to the task by combining all the features.

---

[3]Unlike the usual patrolling problem which considers patrolling vertices

[4]The extension of Nash equilibrium for Bayesian Games

Architecture Name	Basic Type	Communication	Next node choice	Coordination strategy
Random reactive	reactive	none	locally random	emergent
Conscientious reactive			locally individual idleness	
Reactive with flags		flags	locally shared idleness	
Conscientious cognitive	cognitive	none	globally individual idleness	
Blackboard cognitive		blackboard	globally shared idleness	
Random Coordinator		messages	globally random	central
Idleness Coordinator			globally shared idleness	

Table 1.1: Resume of the main features of the agents. From [Machado et al., 2003]
.

The first parameter considered is the type of agent : reactive or cognitive. Reactive agents can only perceive adjacent nods while cognitive agents can perceive a depth $d > 1$ of the graph, allowing them to plan paths. In their study, [Machado et al., 2003] considered a global perception : cognitive agents have a complete view of the graph. Different types of communication have been considered : in the *flags* type, agents put flags in their environment ; in the *blackboard* type, agents share a common database ; in the *messages* type, agents exchange messages with a coordinator. The decision-making part, or how to chose the next node to visit, takes into account the type of agents (local perception or global perception) and the criteria chosen to to evaluate the patrolling. Finally the last feature is the mean of coordinating the agents : a centralized coordination where a coordinator chooses the goal for each agent, or a decentralized coordination, that emerges from the interactions between the agents. The experiments conducted in [Machado et al., 2003] enabled to see three groups of architecture emerging : the *random group*, the *non-coordinated group* and the *top group*. Table 1.2 shows how the architectures are divided into the groups.

Groups	Agents
Random Group	Random Reactive Agent
	Random Coordinator
Non-coordinated Group	Reactive Agent with Flags
	Blackboard Cognitive Agent
Top Group	Conscientious Reactive Agent
	Conscientious Cognitive Agent
	Idleness Coordinator

Table 1.2: MAS architecture groups. From [Machado et al., 2003]

The Random Group performed unsurprisingly the worst with small population, while the Top Group performed better. However, when increasing the size of the population, the results of the Random Group tended to be equivalent to those of the Top Group. This can be explained by the highly unpredictable behavior of the agents in the Random Group. For the Non-coordinated

Group, all the agents tended to go to the same place at the same moment. This study, even though being an important piece of work in the beginning of multi-agent patrolling, presents several weaknesses already pointed out by [Portugal and Rocha., 2011], the most important being that the edges are unweighted, which means that the agents travel from a vertex to another regardless of the distance between them.

In [Chevaleyre, 2004], Chevaleyre proved that the patrolling problem could be solved with a Travelling Salesman approach and extended this approach to more than one agent. This multi-agent approach is called cyclic strategies. They compared this approach to partitioning strategies, in which the area to patrol is separated in different zones, then distributed among the agent. The cyclic strategies are showed to be better suited in highly connected graphs while the partitioning strategies show better performances in environment with long corridors separating the zones. Other partitioning methods have been developed after Chevaleyre's study, as [Portugal and Rocha, 2010]. In [Elmaliach et al., 2009], the authors suggested an algorithm for patrolling inside a closed area that guarantees to visit each point in the graph with an optimal frequency. This algorithm is robust by ensuring the optimality as soon as one robot works properly. Finally, they suggested an algorithm to handle events, that is to say changes in the environment that requires treatment from the robot, while maintaining the patrolling path. Still in the frequency-based techniques, Elmaliah et al. extended the state of the art by suggesting a realistic model of motion that considers velocity uncertainties [Elmaliach et al., 2008]. Those works are centralized, as the optimal path and zone partitioning is computed globally for all agents.

In [Glad et al., 2008], Glad et al. used the ant paradigm to coordinate the agents in a decentralized way. Each agent can only mark and move according to its local perception of the environment. This work is based on the EVAP algorithm [Chu et al., 2007], that enables patrolling even in unknown environments. The agents drops pheromone when they move to a cell, and this pheromone evaporates according to a certain rate. Sensing the remaining pheromone enable the agents to decide where to go next. The authors proved theoretically that the patrolling task is achieved with this setting. In [Menezes et al., 2006] and [Hwang et al., 2009], auction mechanisms are proposed to determine the patrolling area for each agent. With this mechanism, the agents need no pre-computed path.

All the work presented so far deals with non-adversarial setting. As for single-agent, non-deterministic strategies are widely used to solve the problem of multi-agent patrolling in adversarial setting. For instance, in [Agmon et al., 2008a], several robots are patrolling around a closed area with randomized movements. The authors suggested a polynomial-time algorithm such that the minimal probability of penetration detection is maximized. They assumed a strong adversarial model in which the adversary knows the position of the robots and the patrol scheme. In [Agmon, 2010], Agmon extended the patrolling state of the art by considering that the agents should be granted different reward depending on when they detect an intrusion[5]. He also considered that an intrusion can be detected from distance, the probability of detection depending both on the distance from the robot and the state of the intrusion. His algorithm incorporates Markovian assumptions in dynamic programming inspired algorithms. The work presented in [Paruchuri et al., 2006] for single agent also been carried out for multi-agent setting in the same study, using distributed POMDPs.

---

[5]For instance, a intrusion detected very lately will be less rewarded that an intrusion detected as soon as the intruder penetrates the area

1.3   Active Sensing in Map exploration

Robotic exploration can be defined as a process that discovers unknown features in an environment by means of mobile robots [Amigoni et al., 2012]. In the case of Map Exploration, the unknown feature is the map of the environment itself. The problem of map-building have not been much addressed as a standalone problem but widely explored under the problem of Simultaneous Localization And Mapping (SLAM).

1.3.1   Single agent map exploration

The major piece of work concerning map-building as a single-agent standalone problem has been produced by Yamauchi, in [Yamauchi, 1997]. Yamauchi introduced the *frontier-based exploration*, based on the idea that the borders between known and unknown world are the places one that provides the most new information. Frontiers are regions at this boundary and when robots reach these frontiers, they can see unexplored areas and expand the map territory, pushing back the boundary. To detect the boundary, Yamauchi used evidence grids, that are made up of cells that stores the probability that the corresponding region is occupied [Moravec and Elfes, 1985]. This work has been widely reused in multi-agent map exploration, but very little in single-agent setting. Recently, Bachrach et al. suggested in [Bachrach et al., 2009] an exploration planner that used a modified frontier-based strategy to determine the best frontier to reach and optimize the map-building.

Simultaneous Localization And Mapping (SLAM) is the problem of building a map of the environment while simultaneously determining the location of the robot within this map [Aulinas et al., 2008]. At the beginning, both the map and the robot's location are unknown, but the robot has known kinematic model. The SLAM problem is usually addressed in several steps :

1. Initialization : Define the robot's initial position or work with some pre-existing features with high uncertainty on the robot's position

2. Prediction : When the robot moves, the motion model provides new estimates of its new position with the associated uncertainty

3. Measurement : New features are added to the map and previously added features are re-measured

4. Loop : repeat step 2 and 3

In addition to map-building related difficulties, Step 3 raises the issue of data association : due to uncertainty and sensors' error, the same feature may be sensed several times and give slightly different measures. It is important that the SLAM algorithm can detect that those measures are produced by the same feature. This data association problem is very hard to solve and a key in SLAM problems. However, this more related to data fusion and object recognition than map exploration. That's why we will not detail this problem in this thesis. However, a survey of techniques used for data association can be found in [Aulinas et al., 2008].

The SLAM problem is characterized by uncertainty and sensor noise, and it is not surprising that probabilistic techniques have naturally gain popularity to solve it, the dominant techniques being filter-based. Extensions of Kalman Filter (Extended Kalman Filter, Unscented Kalman Filter) have been successfully used in SLAM algorithms. Indeed, the Kalman Filter can only be applied to linear systems, which is obviously not the case in real-world. The Extended Kalman Filter, used for instance in [Davison and Murray, 2002, Jensfelt et al., 2006], deals

with nonlinearities by linearizing the dynamic equations. This can be viewed as a first-order approximation of the nonlinear system. The propagation during the update step is also conducted through the approximated equations, which can introduce large error and can lead sometimes to the divergence of the filter. The Unscented Kalman Filter is an improvement of the Extended Kalman Filter, as it can capture mean and covariance up to the second order and propagates through the true non-linear system. The Unscented Kalman Filter has also been widely used to solve the SLAM problem [Sunderhauf et al., 2007, Holmes et al., 2009]. Other types of Kalman Filters have been used (Information Filter, Compressed Extended Kalman Filter), each with advantages and drawbacks. Particles Filters are other type of filter used for SLAM, that can handle highly nonlinear sensors. However, it involves a growth in computational complexity and are not suitable for real-time applications [Montemerlo et al., 2002]. They are therefore used only for the localization part of the SLAM, but not for map-building, in combination with other techniques, as in [Montemerlo and Thrun, 2007].

Another technique used in SLAM, but not filter-based, is the Expectation Maximization, as in [Burgard et al., 1999]. Expectation Maximization is a statistical algorithm that is able to build a map when the robot's pose is known. The algorithm iterates two steps : an Expectation step where the posterior over robot poses is calculated for a given map, and a Maximization step where the most likely map is calculated given these pose expectations. The accuracy of the computed map increases during the execution of the algorithm, but the need to process the same data several times makes it inefficient as a complete framework and not suitable for real-time applications. However, it is often used in combination with Particle Filter, which can evaluation the robot's pose and so reduce or even suppress the expectation step [Cappé, 2009, Corff et al., 2011].

The Table 1.3, from [Aulinas et al., 2008], summarizes the different techniques as well as their pros and cons.

The usual SLAM problem is passive : the influence of the robot on the SLAM algorithm performance is ignored. In *active SLAM*, the robot needs to optimize an objective function related to the SLAM algorithm. Usually, it means planning efficient paths to perform SLAM efficiently. Active SLAM has first been tackled in [Feder et al., 1999], with a greedy approach where the next action is chosen to maximize the information gain in the next measurement. This approach is myopic and no long-horizon planning is achieved. Active SLAM has then been studied by number of authors, with one problem being to chose a correct objective function, and so a good optimality criterion. This optimality criterion should reflect the uncertainty of the system about its state. Two kind of criteria are usually considered, both introduced in [Fedorov, 1972], the A-optimality criterion and the D-optimality criterion. Both are based on the covariance matrix of the probability distribution of a state. Some studies, such as [Mihaylova et al., 2003] and [Sim and Roy, 2005], argued that the A-optimality criterion performed better than the D-optimality criterion and is a meaningful metric for active SLAM. However, recent studies, such as [Carillo et al., 2012], claimed that computing the D-optimality criterion as in the previous studies leads to wrong results and proved that, by computing the D-optimality criterion as in [Kiefer, 1974], this criterion produced results as good as the A-optimality criterion, and even better in some cases. Other uncertainty measures have been proposed, such as the entropy or change in entropy of the probability distribution [Stachniss et al., 2005] and the Kullback-Leibler distance [Carlone et al., 2010].

Based on an optimality criterion, different algorithms have been suggested to compute efficient exploration policies. In [Leung et al., 2006], Leung et al. suggested to use local planning strategies as Model Predictive Control [Morari et al., 2014]. In [Kollar and Roy, 2008], the authors framed the problem as a constrained optimization and solved it using the Policy Search

Pros	Cons
Kalman Filter and Extended KF (KF/EKF)	
- high convergence	- Gaussian assumption
- handle uncertainty	- slow in high dimensional maps
Compressed Extended KF (CEKF)	
- reduced uncertainty	- require very robust features
- reduction of memory usage	- data association problem
- handle large areas	- require multiple map merging
- increase map consistency	
Information Filter (IF)	
- stable and simple	- data association problem
- accurate	- max need to recover a state estimates
- fast for high dimensional maps	- in high-D is computationally expensive
Particle Filter (PF)	
- handle nonlinearities	- growth in complexity
- handle non-Gaussian noise	
Expectation Maximization (EM)	
- optimal to map building	- inefficient, cost growth
- solve data association	- unstable for large scenarios
	- only successful in map building

Table 1.3: List of advantages and disadvantages of filtering approaches applied into the SLAM framework. From [Aulinas et al., 2008]

Dynamic Programming algorithm suggested in [Bagnell et al., 2003]. Rao-Blackwellized Particle Filters have also been explored [Stachniss et al., 2005, Carlone et al., 2014].

1.3.2 Multi agent map exploration

Using several robots for map exploration has a lot of advantages : faster exploration, redundancy that implies a better fault tolerance and better accuracy. However, it also raises new issues, two major being (1) coordinate the agents so that they explore simultaneously different regions of the environment and (2) merging the maps built by each agent. A lot of studies are based on [Yamauchi, 1997], and focus on splitting the frontier regions between the robots. Yamauchi himself suggested in [Yamauchi, 1998] a multi-robot exploration in which the robots share the gathered information to build the map and the list of frontiers. Each robot then moves to its closest frontier. Here, the coordination is implicit and performed by sharing information. More recent studies also assume implicit coordination, as in [Bautin et al., 2011], in which the robots choose their target location by reasoning on positions instead of distances : each robot will choose the frontier having less robots closer than it. Both algorithms share a common drawback which is that the several robots can choose the same frontier. To overcome this issue, an explicit coordination is usually needed. This coordination can be centralized - one coordination component affects the frontier to the robots - or decentralized - the robots communicate with each others to decide how they affect the frontiers.

Most of centralized approaches suggested to use a trade-off between the cost to reach a given target location and the utility of reaching this target location. In [Simmons et al., 2000], the robots send a bid to a central executive containing their cost of reaching different location and their estimated information gain. Then, the central executive uses a greedy algorithm to assign the locations by maximizing the team utility, expressed as the information gain minus the cost. In [Burgard et al., 2002], Burgard et al. the utility is expressed in a different way since it depends on the probability that the target location was visible from target locations affected to other robots. In both studies, the local robots' maps are sent to a central server that combines them to a global map and robots' initial positions are also assumed approximately known. Ko et al. were the first to tackle the problem of map merging with completely unknown initial location [Ko et al., 2003] by using particle filters. In this approach, a robot tried to localize itself in another robot's map to confirm their relative position before merging maps. This map-merging problem has also been studied in [Zhou and Roumeliotis, 2006] using landmarks and [Wang et al., 2007] using meeting points.

Despite their efficiency in coordinating the robots, centralized algorithms encounter several problems, such as :

1. a computational cost that increases dramatically with the number of robots

2. a communication problem : the robots all need to be in communication range with the central server

To overcome these issues, decentralized algorithms have quickly been studied. In [Zlot et al., 2002], Zlot et al. extended the utility method by using approach based on Market Economy. The robots bid on their target and then negotiate to affect the target locations. In this work, the robots communicate only with other robots that are in communication range and their relative initial positions are known. Burgard et al. also considered the problem of limited communication in [Burgard et al., 2005]. They extended their previous work [Burgard et al., 2002] and applied the target selection algorithm to each sub-team of robots that can communicate with each others.

Some studies considered other ways, not frontier based, for map-building. In [Wurm et al., 2008], the authors considered instead the notion of segment and used the Hungarian [Kuhn, 1955] method to assign to each robot a segment to explore. In [Matignon et al., 2012], the authors suggested a decision-theoretic approach based on a Decentralized Markov Decision Process (Dec-MDP) in which each robot computes locally a strategy that minimizes the interactions between the robots and maximizes the coverage of the team. In [Kontitsis et al., 2013], Kontitsis et al. based their work on motion planning and used Extended Kalman Filter and Relative Entropy optimization to compute efficient paths.

1.4 Other type of exploration

Since exploration considers all type of unknown features, it can be applied to other problems than map exploration. In several studies, Lilienthal et al. applied exploration strategies to Gas Distribution Mapping [Lilienthal et al., 2007, Loutfi et al., 2009]. In this problem, the purpose is to explore the environment and to combine heterogeneous information from different sources (gas detection, wind detection...) to find the source of a gas emission or to build a map of the gas distribution. This kind of work is also referred as *robot olfaction*. Chanel et al. applied in several studies POMDP-based techniques for multi-target detection [Chanel et al., 2012, Chanel et al., 2013]. In [Singh et al., 2009], Singh et al. defined Multi-robot Informative Path Problem, in which robots have to optimize their path in order to gather as much information possible about some dynamics phenomena.

1.5 Conclusion

Active sensing is acting in order to maximize the information gain. In this chapter, we presented different contexts in which active sensing is a key. We first presented the patrolling problem, in which agents explore their environment indefinitely to detect any intrusion and/or abnormalities. Then, we presented the map-building problem, in which the agents need to coordinate in order to create an accurate map of their environment.

Both problems share the need of exploring unknown features, and as does the event exploration problem. However, the nature of the features to be explored and the mission to perform is very different and need adapted frameworks. The patrolling and event exploration problem both concern dynamic environments, topologically known, in which *events* need to be detected. The main different between both remains in the type of solutions required. The patrolling problem requires to find an optimal path to detect intrusions. The behavior of the robots after the intrusion is detected is rarely discussed in the literature. In the event exploration problem, we are interested in finding adaptable long-term strategies, not only to detect events but also to maintain a good knowledge about those event among time. Moreover the event exploration problem shares with the map-building problem the fact that it needs to take uncertainty into account. In the event exploration problem, this uncertainty is due to the robots' sensors but also to the dynamicity of the system.

We face then the issue of representing the agents' knowledge and the uncertainty about this knowledge. The next chapter will present some basis of knowledge representation and the most common ways to deal with uncertainty.

Chapter 2

Representing uncertainty and sharing knowledge

Contents

To reason and to perform some tasks, an agent need a good representation of the world surrounding it. By nature, the real world is uncertain in different ways. Therefore a good uncertainty representation is a key for operating in the real world. In addition, several agents that cooperate in the same world need to exchange their information. This chapter focuses on ways to represent uncertainty in an agent and on a crucial concept when sharing information : the relevance.

2.1 Knowledge Representation

Knowledge Representation is the kernel of all intelligent systems : without knowledge, a system is not able to reason and to perform any task. A good representation is a key for good reasoning and decision-making, and depends on the application considered. Logics are a common and useful way to represent knowledge and were studied far before the computer's extend. Frege defined the purpose of logic as "to express a content through written signs in a more precise and clear way than it is possible to do through words" [Heijenoort, 1967]. The basic logic used in computer programming is the *propositional logic*. Propositional logic manipulates atoms, which can be interpreted as the base vocabulary of the logic, and formulas, which are made up of atoms and some operators. Each formula can be interpreted as *true* or *false*. This logic has been widely used and advanced to express more and more complicate things. The predicate logics use variables and quantifiers ($\exists$ "there exists" and $\forall$ "for all") to manipulate wider concepts. The modal logic introduces operators such as $\diamond$ "it is possible" and $\square$ "it is necessary". Epistemic logic manipulates knowledge by using operators as B "I believe that" and K "I know that. All those logics (and a lot of others not listed here) are monotonic : adding a new formula never reduces the set of consequences: learning a new piece of knowledge will not reduce what is already known. However, those logics cannot deal with belief revision, that is to say when a new knowledge contradicts what is already known. Non-monotonic logic has been introduced to solve this problem, and to enable other kind of reasoning such as reasoning by default. Other formalisms have been introduced to represent knowledge and make inference such and Bayesian networks, that will be introduce in Section 2.2.2.

2.2 Representing uncertainty

As soon as one need to reason about the real world, uncertainty is unavoidable. In [Dubois, 2007], the authors distinguishes three different types of uncertainty arising from different origins : aleatory uncertainty, epistemic uncertainty and inconsistent uncertainty (Figure 2.1).
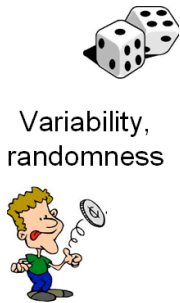
Type of Uncertainty	Aleatory Uncertainty	Epistemic Uncertainty	Inconsistent Uncertainty
Origins	Variability, randomness	Ignorance, Incompletness	Conflict between sources

Figure 2.1: Origins of Aleatory, Epistemic, Inconsistent Uncertainty. From [Bellenger, 2013]

Aleatory uncertainty is due to the outcomes of random experiments, as throwing a dice, and

to the variability of phenomena, like weather forecast. Aleatory uncertainty can be estimated objectively through statistical data. Epistemic uncertainty is due to a lack of knowledge. Unlike aleatory uncertainty, epistemic uncertainty cannot be estimated through statistics. An example given in [Dubois, 2007] is the knowledge of the Brazilian President's birth date: knowing the birth date of other presidents will not help to figure out the birth date of the Brazilian President. Inconsistent uncertainty is due to conflicting reports from different sources. The more sources, the more likely the inconsistency.

To deal with uncertainty of any kind, an agent need a formal representation of this uncertainty. Almost all uncertainty representations are based on a set of *possible worlds*, also called *possible states*. These are the worlds or the outcomes that an agent considers possible. The problem of dealing with uncertainty is to select the possible worlds. The more worlds are considered possible, the more uncertain is the agent.

In the following we will present different ways to represent uncertainty. We will focus on numeric representations and finite set of possible worlds. The first and the most obvious is the probability measures (Section 2.2.1), which will be used in this thesis. Related to probabilities, we introduce the Bayesian Networks to represent dependencies between variables (Section 2.2.2). Then, for a sake of completeness, we also present the Dempster-Shafer theory of evidence (Section 2.2.3) and the possibility measures (Section 2.2.4). We will finally give an overview of some other less common methods (Section 2.2.5).

2.2.1 Probability measures

Let us consider the set of possible worlds $W = \{w_1, \ldots, w_n\}$. A probability measure assigns to each world $w_i \in W$ a number - the probability - that described the likelihood that this world is the actual world. A probability measure on possible worlds shall respect some properties, including:

$$\forall w_i \in W, P(w_i) \in [0,1] \tag{2.1}$$

$$\forall w_i, w_j \in W, w_i \neq w_j, P(w_i \cup w_j) = P(w_i) + P(w_j) \tag{2.2}$$

$$\sum_{w_i \in W} P(w_i) = 1 \tag{2.3}$$

The *Principle of Indifference* states that, in absence of specific information, there is no reason to consider one possible world more likely that another. So, if we consider n possible worlds, the probability associated to each world would be $\frac{1}{n}$. Then, statistical studies allow to attribute probabilities to the possible worlds. As an example, let us consider the patients of a doctor. There is three possible worlds : w_1 = *the next patient is over 60* and w_2 = *the next patient is between 20 and 60* and w_3 = *the next patient is under 20*. If we know that 60 percent of these patients are over 60, we can conclude that $P(w_1) = 0.6$ and $P(w_2 \cup w_3) = 0.4$. However we cannot conclude about $P(w_2)$ and $P(w_3)$. The matter on how statistics are created is beyond the scope of this thesis.

We call the belief state of an agent [6] the probability measure an agent associates to the set of possible worlds. This belief state is a basis for reasoning and should be updated each time the agent receives new information. The most straightforward and common way to update agent's beliefs is the Bayes' rule [Bayes, 1763] : $P(w_i|E) = \frac{P(E|w_i) \cdot P(w_i)}{P(E)}$, where

- w_i is the possible world considered, whose probability may be affected by new information

---

[6]Sometimes shortened as *agent's beliefs*

- E is an evidence, that is to say new information that may affect the probability of the possible world considered

- $P(w_i)$ is the *prior probability*, the probability of w_i before E is observed

- $P(w_i|E)$ is the *posterior probability*, the probability of w_i knowing E

- $P(E|w_i)$ is the *likelihood* of E, that is to say the compatibility of the evidence given the possible world considered

- $P(E)$ is the *marginal likelihood*

This update, also called *conditioning*, is a very useful tool but also suffers from some problems when dealing with possible worlds with probability 0. In that case, $P(w_i|E)$ is undefined. This leads to some philosophical questions regarding worlds with probability 0 : are they really impossible ? How unlikely should they be before being assigned probability 0 ? In practice, this kind of case usually occurs when the probability distributions are discretized and so to approximations. Theoretically, according to the principle of Indifference and in absence of other information, the initial probability distribution should be the uniform distribution. If the Bayes rule is applied from this distribution, the exact values of probabilities could never be 0. If some prior information states that a world in truly impossible, then it seems reasonable to assume that this world should not be represented as a possible world. Therefore, the case of probability equal to 0 due to discretization should be handled when designing the system, with solutions depending on the problem.

2.2.2 Bayesian Networks

Bayesian network, introduced by Pearl in [Pearl, 1988] and Neapolitan in [Neapolitan, 1990], is a practical way to represent dependencies between random variables. A Bayesian network is a directed acyclic graph (DAG) whose nodes are labeled by random variables and whose edges represent causal influence. Let us consider as an exemple the following situation. Two events can cause some grass to be wet : rain or use of sprinkler. It is common knowledge that it usually rains more in winter, and that sprinklers are usually not useful in winter. We consider in addition that rain provokes slippery roads. This situation is represented in Figure 2.2 as a Bayesian network.

The nodes of the graph are labeled with the variables considered in the situation and the edges represent the causal dependency. Each variable has two possible values : *true* or *false*. A table is associated with each node in the network, containing conditional probabilities of that node given its parents. For instance, we can see that the probability of *Rain* being *true* while *Winter* is *true* is 0.8, and the probability of *Wet Grass* while *Sprinkler* is *true* and *Rain* is *true* is 0.95.

A Bayesian network over variables X specifies a unique probability distribution over its variables, defined as follows [Pearl, 1988]:

$$P(X = x) := \prod_{\theta_{x|u}:xu\sim x} \theta_{x|u}$$

where $\theta_{x|u}$ is the conditional probability of x given an instantiation u of its parents and $xu \sim x$ means that instantiations xu and x agree on the values of their common variables.

Bayesian networks enables to make inference and answer to two fundamental queries :
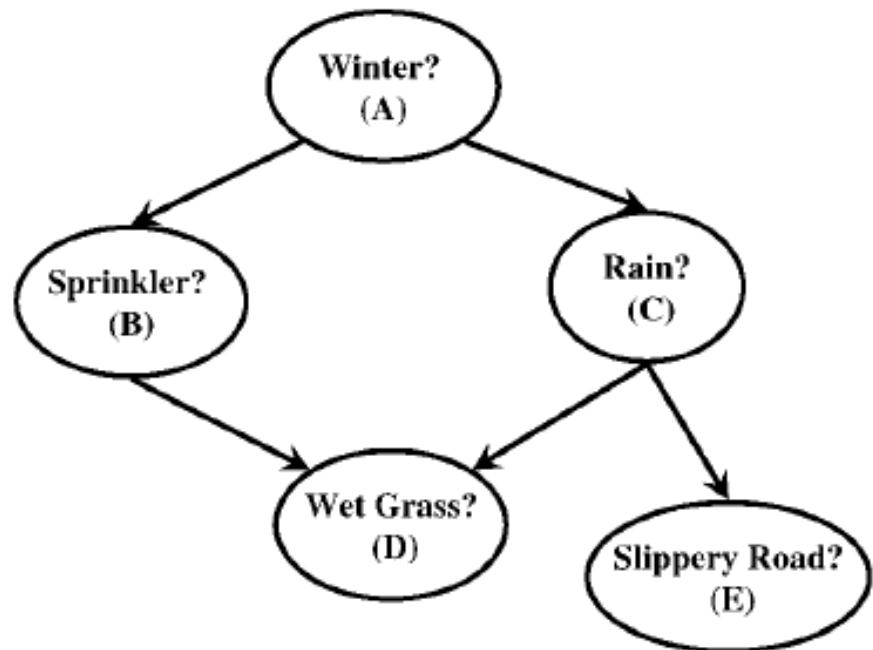
A	Θ_A
true	0.6
false	0.4

A	B	$\Theta_{B|A}$
true	true	0.2
true	false	0.8
false	true	0.75
false	false	0.25

A	C	$\Theta_{C|A}$
true	true	0.8
true	false	0.2
false	true	0.1
false	false	0.9

B	C	D	$\Theta_{D|B,C}$
true	true	true	0.95
true	true	false	0.05
true	false	true	0.9
true	false	false	0.1
false	true	true	0.8
false	true	false	0.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E|C}$
true	true	0.7
true	false	0.3
false	true	0
false	false	1

Figure 2.2: A bayesian network over five propositional variables. From [Harmelen et al., 2008].

Most Probable Explanation (MPE) : What's the most likely instantiation of network variables X given some evidence e ? $MPE(e) = \operatorname{argmax}_x P(x|e)$.

Probability of Evidence (PR) : What's the probability of evidence e $P(e)$?

Those problems are difficult, since finding the Most Probable Explanation has been proven to be NP-hard [Cooper, 1990] and finding the Probability of an Evidence has been proven to be PP-hard [Littman et al., 2001]. Exact and approximate algorithms have been developed to answer these questions.

Exact algorithms are based on the network topology and use variable elimination (as in [Dechter, 1999]), joint trees (as in [Shenoy and Shafer, 1986, Lauritzen and Spiegelhalter, 1988, Jensen et al., 1990]), or recursive conditioning (as in [Darwiche, 2001]). Those algorithms have been refined to exploit the parametric structure of a Bayesian network, for instance by adopting non-tabular representation of the conditional probabilities (as in [Poole and Zhang, 2003, Larkin and Dechter, 2003]) or local structures (as in [Allen and Darwiche, 2002, Chavira and Darwiche, 2005]). Approximate algorithms are based on the idea that the best answer may be too hard to compute, but it may be sufficient to compute a *good answer*. Unlike exact algorithms, approximate algorithms are generally not sensitive to tree width. The main techniques used in approximate algorithms are stochastic sampling (as in [Shachter and Peot., 1989, Cheng and Druzdzel, 2000]), belief propagation (as in [Yedidia et al., 2005]) and variational methods (as in [Jordan et al., 1999]). Most of the approximate algorithms in the literature have bounded errors.

2.2.3 Dempster-Shafer belief functions

The Dempster-Shafer theory of evidence has been introduced by Arthur Dempster [Dempster, 1967] and developed by Glenn Shafer [Shafer, 1976]. It allows the combination of distinct evidence from different sources in order to calculate a global amount of belief for a given hypothesis. Given the set W of possible worlds, to each subset $U \subseteq W$ is associated a belief function and a plausibility function.

The belief function, denoted $Bel(U)$ associates to each U a number in the interval $[0, 1]$ corresponding to the belief in U. This belief function must satisfy the following properties :

1. $Bel(\emptyset) = 0$

2. $Bel(W) = 1$

3. $Bel(\bigcup_{i=1}^{n} U_i) \geq \sum_{i=1}^{n} \sum_{I \subseteq \{1,\ldots,n\}} (-1)^{|I|+1} Bel(\bigcap_{j \in I} U_j)$

Property 3 is a general formulation of the well known inclusion-exclusion principle [van Lint and Wilson, 2001], but with the $=$ replaced by a $\geq$. We can note that any probability measure defined on the power set $\mathcal{P}(W)^7$ is a belief function, but the converse does not hold. For instance, if $W = \{w_1, w_2\}$, $Bel(w_1) = 0.5$, $Bel(w_2) = 0$, $Bel(W) = 1$ and $Bel(\emptyset) = 0$, then Bel is a belief function but not a probability measure. A second important difference between belief functions and probability measures is that probability measures are entirely characterized by their behavior on the singleton sets, which is not the case for belief functions.

The plausibility function is defined as $Plaus(U) = 1 - Bel(\bar{U})$ and satisfies the following properties :

---

[7]That is to say the set of all subsets of W

4. $Plaus(\emptyset) = 0$

5. $Plaus(W) = 1$

6. $Plaus(\bigcap_{i=1}^{n} U_i) \geq \sum_{i=1}^{n} \sum_{I \subseteq \{1,...,n\}} (-1)^{|I|+1} Plaus(U_{j \in I} U_j)$

It is easy to derive from property 3 of the belief functions that $Bel(U) < Plaus(U)$. For an event U, the interval $[Bel(U), Plaus(U)]$ can be viewed as describing the range of possible values of the likelihood of U.

The beliefs are supported by evidences to a certain degree. Let us take the following example, explained in [Halpern, 2003a]. A bag contains 100 marbles. 30 are known to be red and the remainder are known to be either blue or yellow. We are interested in the color of a marble taken out of the bag. The information that there is 30 red marbles provides support in degree 0.3 for *red*. The information that there is 70 blue and yellow marbles does not provide any support for either *blue* or *yellow*, but does provide support in degree 0.7 for the set $\{blue, yellow\}$.

As for probabilistic belief states, the belief functions need to be updated when a new evidence arrives. This update is made as follows [Halpern, 2003a]:

$$Bel(V|U) = \frac{Bel(V \cup \bar{U}) - Bel(\bar{U})}{1 - Bel(\bar{U})}$$

$$Plaus(V|U) = \frac{Plaus(V \cap U)}{Plaus(U)}$$

It has been pointed out, for instance in [Halpern, 2003a] that updating belief functions can give counter-intuitive results on some applications. This problem stresses out the need to be extremely careful about the underlying interpretation of a belief function when trying to take into account new information.

2.2.4 Possiblity measures

Possibility theory, introduced in [Zadeh, 1999], is a generalization of the fuzzy logic. Rather than defining a degree of belief or verity, the possibility theory models the credit an agent has for a given hypothesis. Let us consider once again the set W of all possible worlds. A possibility measure $Poss$ associates to each subset of W a number in $[0, 1]$ and satisfies the following properties [Halpern, 2003a] :

9. $Poss(\emptyset) = 0$

10. $Poss(W) = 1$

11. $Poss(U \cup V) = max(Poss(U), Poss(V)) \ \forall \ U, V \subset W, U$ and V disjoint

Like the probability measure, the possibility measure can be determined by its behavior on singletons : $Poss(U) = \max_{w \in U} Poss(w), \forall U \subset W$. The dual of possibility is called necessity and is defined as $Nec(U) = 1 - Poss(\bar{U})$. There are four extreme cases of necessity and possibility to understand the relation between them.

- $Nec(U) = 1$ means that U is necessary, and so certainly true. It implies $Poss(U) = 1$

- $Poss(U) = 0$ means that U is impossible, and so certainly false. It implies $Nec(U) = 0$

- $Poss(U) = 1$ means that U is completely possible, and so it would not be surprising if U occurs. But it leaves $Nec(U)$ unconstrained

- $Nec(U) = 0$ means that U is unnecessary, and so it would not be surprising if U doesn't occur. But it leaves $Poss(U)$ unconstrained

Once more, possibilistic beliefs need to be updated with new evidence. There are two approaches in the literature for updating beliefs based on possiblity measures [Halpern, 2003a]. The first one, but not the most common, considers possibility measure as a special case of Dempster-Shafer's plausibility function and so defines the update as :

$$Poss(V|U) = \frac{Poss(V \cap U)}{Poss(U)}$$

The second and most common definition considers that the operation min should play the same role for possibility measures than the multiplication does for probability measures. In this case, the update is performed as follows:

$$Poss(V|U) = \begin{cases} Poss(V \cap U) & \text{if } Poss(V \cap U) < Poss(U) \\ 1 & \text{if } Poss(V \cap U) = Poss(U) \end{cases}$$

These two approaches may be useful in different context. In [Dubois and Prade, 1998], Dubois and Prade suggested that in infinite spaces and for numeric representations of uncertainty it may be more appropriate to use the first approach, while the second one is more appropriate for qualitative, nonnumeric representations.

2.2.5 Other methods

A bunch of other methods have been developed to represent uncertainty. Ranking functions [Spohn, 1988], sharing the same intuition than possibility measures, associates a natural number (or infinity) to an subset $U \subset W$. This number described the degree of surprise, that is to say how surprised an agent would be if the actual world was in U. The 0 denotes *unsurprising* and the higher the number, the more surprising. The infinity denotes "so surprising as to be impossible". A ranking function is characterized by its behavior on singletons and follows the following properties :

1. $\kappa(\emptyset) = \infty$

2. $\kappa(W) = 0$

3. $\kappa(U \cup V) = min(\kappa(U), \kappa(V)), \forall U, V \subset W, U, V \, disjoint$

The last approach we will consider is a generalization of all the approaches mentioned so far: plausibility measures[8] [Friedman and Halpern, 1995, Friedman and Halpern, 2001]. As a probability measure maps sets in an algebra $\mathcal{F}$ over a set W of worlds to $[0, 1]$, a plausibility measure maps sets in $\mathcal{F}$ to some arbitrary partially ordered set. For any $U, V \subset W$, having $Pl(U) \leq Pl(V)$ means that V is at least as plausible as U. Formally, a plausibility space is a tuple $S = \langle W, \mathcal{F}, Pl \rangle$, where W is a set of possible worlds, $\mathcal{F}$ is an algebra over W and Pl maps sets in $\mathcal{F}$ to some set D of plausibility values, partially ordered by a reflexive, transitive and antisymmetric relation $\leq_D$. D is also assumed to contain two special elements $\top_D$ and $\bot_D$, such as $\forall d \in D, \bot_D \leq_D d \leq_D \top_D$. A plausibility measure respects the following properties :

---

[8]Not to be confused with the plausibility functions of the Dempster-Shafer theory

1. $Pl(\emptyset) = \perp$

2. $Pl(W) = \top$

3. If $U \subset V$, then $Pl(U) \leq Pl(V)$

Probability measures, Dempster-Shafer belief and plausibility functions, and possibility and necessity measures are instances of plausibility measures where $D = [0, 1]$, $\perp = 0$, $\top = 1$ and $\leq_D$ is the standard ordering on reals. Ranking functions are instances of plausibility measures where $D = \mathbb{N}^*$, $\perp = \infty$, $\top = 0$ and $\leq_D$ is the opposite of the standard ordering on $\mathbb{N}^*$.

The same belief updates steps exist for these different representations. An overview can be found in [Halpern, 2003b]

2.3 Information relevance

The relevance of information is a concept that we, as humans, intuitively manipulate in our everyday communications and interactions with others. Communicating relevant information and knowing that somebody else is communicating relevant information is a key to recognition, understanding and knowledge inference. Studying relevance is part of the pragmatics, defined by Moeschler as the study of language use: its object is "meaning in use" [Moeschler, 2007]. In this field, Grice suggested a Cooperation Principle [Grice, 1975] that all cooperative agents engaged in a conversational interaction should respect. This principle can be decomposed in four categories. The first one is the category of Quantity, which relates to the quantity of information to be provided, contains two maxims :

1. Make your contribution as informative as is required (for the current purpose of the exchange)

2. Do not make your contribution more informative that is required

Grice argued that the second maxim in disputable since being overinformative is not really a transgression of cooperation, but mostly a waste of time. However, in the case of costly communication, being overinformative is costly and so this maxim is of prior importance.
The second category is the category of Quality. The main message of this category is to try to make a contribution that is true. Two maxims are set by Grice :

1. Do not say what you believe to be false

2. Do not say that for which you lack adequate evidence

The third category is the category of Relation, in which only one maxim holds : Be relevant. Grice admits himself that this terse formulation includes a high number of problems about the different kinds and focuses of relevance, their impact in the conversation and so on. The fourth and last category is the category of Manner, which states that the contribution should be easy to understand.

Following this Cooperation principle, Sperber and Wilson developed a Theory of Relevance [Wilson and Sperber, 2002], sharing the intuition of a need for relevance but following a different approach. For Sperber and Wilson, intuitively, a contribution is relevant for a given agent if and only if it produces positive cognitive effects, that is to say if it implies a correct modification of the mental state of the person which receives the information. Therefore, false contribution cannot be considered as relevant. They also state that different contributions can have different

relevance level. The more positive cognitive effects a contribution brings to an agent, the more relevant it is for it. However, the harder a contribution is to understand for an agent, the less relevant it is for it. Therefore, relevance is seen as a combination between its informative value and the difficulty to integrate it in a mental state. For a better understanding, let's take an example given in [Wilson and Sperber, 2002]. Mary, who dislikes most meat and is allergic to chicken, rings her dinner party host to find out what is the menu. He could tell her any of three things :

1. We are serving meat

2. We are serving chicken

3. Either we are serving chicken or $(7^2 - 3)$ is not 46

The three utterances are true, and all would be relevant to Mary. However, 2 would be more relevant to Mary than 1 and 3. Indeed, 2 is more informative that 1 and so implies more positive effects to Mary. 3 implies the same positive effects to Mary than 2, but it requires an additional effort to understand.

These philosophical considerations of relevance can be easily reused to design intelligent systems dealing with relevance. We will consider in this section two type of intelligent systems using relevance : the Information Retrieval Systems and Multi-agent systems.

2.3.1   Relevance in Information Retrieval Systems

The main purpose of Information Retrieval (IR) is to retrieve all the documents relevant to a request and as few of the non-relevant as possible [van Rijsbergen, 1979]. This relates to Grice's Quantity category : to be interesting, a contribution should only contain the useful information and no more. However, the multidimensional aspect of relevance already stressed out by Grice makes this concept very different dependent on the problem and the user, and explains why it is so hard to reach a consensus. First of all, relevance in intelligent systems is usually used with a wider meaning. In the Grice's category of Relation, the term relevant can be interpreted in its first meaning : related to a subject. In intelligent systems usually, the term relevant means not only "related to a subject", but also "interesting, useful for the receiver".

In [Borlund, 2003], Borlund defined two classes of relevance : the objective, or system-based relevance, and the subjective, or user-based relevance. Borlund also classified the different types of relevance highlighted by Sarajevic [Saracevic, 1996] in these classes. On one hand, the system-based relevance is computed using different criteria and contains Sarajevic's algorithmic relevance, which analyzes the number of matches between the query and the documents in the corpus to be searched in. On the other hand, the user-based relevance is assessed by a user, depending on its need and its current beliefs. The main types of relevance belonging to this class are :

- the topical-like relevance : the user assesses the match between the provided documents and its request. This kind of relevance uses the term relevant with its basic meaning.

- the cognitive relevance : the document is relevant is it answers a cognitive need from the user.

- the situational relevance : the document is relevant if it contains information useful for the user to perform a task.

2.3.2 Toward a theory of relevance in multi-agent systems

The concept of relevance as described and studied for Information retrieval is unfortunately not suitable for information exchange in multiagent systems. Indeed, most of the time, agents don't express any specific request but information should be exchanged to improve the quality of agents' knowledge. It can be related to Borlund's user-based relevance as the relevance of some information depends on an agent's need and beliefs. Floridi suggested in [Floridi, 2008] a base of epistemic relevance. He defined epistemic relevance as a relevance that takes into account the agent to whom the information is sent. He defined the degree of relevance as being a function of the accuracy of some semantic information i understood as an answer to a question q, given the probability that q might be asked by the agent a receiving information i. There is in Floridi's work several concepts already encountered in the pragmatic definition of relevance:

- Information is relevant if it meets a need, the question q, but it may not meet it completely. That's why Floridi defined several degree's of relevance.

- False information cannot be relevant. This corresponds to Grice's category of Quality.

- Information is relevant according to a context. This corresponds to Grice's category of Relation.

In addition, some elements more related to information exchange between agents are developed, such as the need of the agent receiving the information. To model this need, Floridi considers the probability that the agent a receiving the information answering the question q may actually ask it. This underlines the fact that the agents may not be aware of all their needs and that questions may arise or disappear according to their beliefs' evolution. Even if Floridi's work is much more formal than those presented in 2.3, no concrete element is suggested on how to define these probabilities and they seem very hard to evaluate and manipulate in real-type applications.

In several studies, Roussel and Cholvy deepened Floridi's work to propose a formal definition of relevance based on BDI (Belief-Desire-Intention) agents architecture. Roussel first noticed in [Roussel, 2010] that the agent that will communicate the relevant information is hardly ever taken into account. However this agent needs to evaluate the relevance of the information for the agent who will receive it. Roussel extracted some characteristics an information should have to be relevant, in the sense of useful :

- Relevant information for an agent is connected to a need of this agent. The more the information meets the need, the more relevant it is.

- Relevant information is true.

- The more some information is easy to understand, the more relevant it is.

- Relevance of information for a given agent depends on this agent's beliefs. The relevance of an information may change among time if the agent's beliefs are modified.

From these characteristics, Roussel extracted also characteristics of cooperative agents when dealing with information exchange :

- An agent is cooperative if it considers that information it communicates meets a need of the agent receiving it.

- An agent is cooperative if it communicates information that it considers true.

- An agent is cooperative if it anticipates other agent's needs.

- An agent is even more cooperative if it communicates information easy to process.

We notice that an agent is cooperative depending on what it estimates about the other agent, even if this estimation is false. It matches the way we intuitively manage cooperation as human : we can never be certain of the needs of the person we are communicating with, except if they express them directly. Otherwise, we need to guess those needs depending on what we know about the other and the current situation. The problem is the same in multiagent systems, where agents need to model somehow the beliefs of the other agents to be able to decide if some information may be relevant for them and if it should be communicated. From those characteristics, they used BDI logic define the relevance[9] of some information, modeled as formulas, as follows [Roussel and Cholvy, 2009]:

Definition 2.1 (Relevance). Let a be some agent of $\mathcal{A}$, ϕ a formula and Q a request. ϕ is said to be relevant (or useful) for agent a concerning request Q iff the following formula is true :

$$I_a Bif_a Q \land (B_a(\phi \to Q) \otimes B_a(\phi \to \neg Q)) \land \phi$$

where I_a describes the intention of agent a, $Bif_a Q$ holds for "agent a wants to know if Q is true or $\neg Q$ is true", and $B_a(X)$ describes the fact that a believes that X is true.

This definition includes three elements :

- **Agent's information need** $I_a Bif_a Q$: The agent's information need is modeled in a simple way: "agent a wants to know if Q or $\neg Q$".

- **Agent's beliefs** $(B_a(\phi \to Q) \otimes B_a(\phi \to \neg Q))$: Using its current beliefs and the piece of information ϕ, the agent must be able to answer its request and so to determine if Q or $\neg Q$.

- **The piece of information's truth value** ϕ : A false piece of information is considered not relevant.

Therefore, the previous definition can be naturally explained by : a piece of information is relevant for an agent if it is true and enables the agent to answer a request it previously had. Roussel then introduced the notion of potential utility [Roussel, 2010], which corresponds to the previous relevant definition, minus the truth value. Indeed, even if it's acknowledged in the literature that false information cannot be relevant, it may be useful in some specific cases to assess the relevance of some specific piece of information for a given agent, without knowing if this piece of information is true or not.

To our knowledge this work presents the first formal theory of relevance. Despite very interesting models and progresses to manipulate relevance in agent, this work presents some flaws and limitations which renders it unusable in the field of event exploration. The most important of these limitations, already risen by the authors in [Roussel, 2010], is the fact that the model does not consider incorrect beliefs : an agent's beliefs are always considered as true and an agent never need to check if what it already knows is correct. However, checking if an agent's beliefs are true is key in event exploration. The authors suggested to introduce graded beliefs as suggested by Laverny in [Laverny and Lang, 2005a, Laverny and Lang, 2005b]. However, to our knowledge, this step has not been done yet.

---

[9]Also called utility in some of their work

2.4 Conclusion

We presented in this chapter different way to model uncertainty and to update an agent's beliefs. Each model presents advantages and drawbacks, and choosing to use one or the other depends mostly on the application one wants to deal with. Belief functions are proven useful as a model of evidence, while possibility measures and ranking functions deal well with default reasoning and counterfactual reasoning. Plausibility measures, as a general approach, is appropriate for proving general results about representing uncertainty but not for being applied in real problems. Probability is a very powerful tool, very well studied and many technical results have been proven that makes it easy to use. Many justifications for probability can be found in the literature, and particularly in [Mises, 1957], and more recently in [van Lambalgen, 1987]. Some arguments, for instance in [Savage, 1972], claimed that probability is the only way to represent uncertainty for a rational agent. Savage defined an agent as rational according to some axioms, and showed that this agent can be viewed as acting as if its beliefs were characterized by probability measures. The relation between probabilities and statistics makes probability the measure the most adapted for real applications, where frequency of events' occurrences can be objectively measured. For those reasons, we chose to model uncertainty in agent's beliefs with probabilities.

We also presented the existing frameworks to deal with relevance in several contexts. Relevance is still a new concept in multi-agent systems, and the framework presented can only be applied to modal logic. However, we believe that modal logic is not easily applicable in the context of event exploration in dynamic systems. Indeed, modal logic can not express degrees of belief and how those beliefs changes according to new and relevant information. An agent can only believe or not believe. However, in dynamic systems and when dealing with uncertainty, an agents' beliefs need to be quantified (for instance depending on the world's change rate, the robots' sensors' precision, etc.). A formal definition of a relevance degree based on quantified beliefs is still missing in the literature.

Reasoning about uncertainty is important in the context of event exploration, but not enough. As already mentioned, we need to build long-term strategies in order to maintain a good knowledge. Such strategies can only been built through task planning. The next chapter will present the different frameworks for task planning, while keeping in mind the importance of dealing with uncertainty is the planning process.

Chapter 3

Planning under uncertainty

Contents

We can define intuitively planning as the process of deciding how to perform a task before really performing it. In an agent-based frame of mind, it is the process of finding a way to go from an initial state to a goal state. By planning, an agent can decide what is the best way to reach a goal or fulfill a task and anticipate and prevent negative outcomes. To do so, the agent needs to know at least partially the results of its actions and the way they impact the system[10]. Thanks to this knowledge, the agent simulate internally the evolution of the system until it finds a satisfying plan - that is to say a succession of actions - to reach its goal. This chapter presents fundamental concept for agent-planning, focusing on how to deal with uncertainty while planning. In this chapter and later in this thesis, we only consider planning as task planning. Path and motion planning are beyond our concern.

---

[10]the system being defined by the environment and the agents in it, including the planning agent

3.1 Overview of planning

3.1.1 Seven restrictive assumptions of planning

In [Ghallab et al., 2004], Ghallab et al. defined seven assumptions to distinguish different types of planning. The more assumptions are released, the more complex is the problem. Those assumptions are :

1. (*Finite system*) The system has a finite set of states.

2. (*Fully Observable system*) The system is fully observable, i.e. the agent has complete knowledge about the current state of the system.

3. (*Deterministic system*) The system is deterministic, i.e. for every state and every action, there exists at most one successor of the state through the action.

4. (*Static system*) The system is static, i.e. no event can occur. The system keeps the same state until the controller applies some actions.

5. (*Restricted goals*) The goals handled by the planner correspond to goal states to reach. The planner doesn't handle goals such as states to be avoided, constraints, utility function etc.

6. (*Sequential plans*) A solution plan to a planning problem is a linearly ordered finite sequence of actions.

7. (*Implicit time*) Actions and events have no duration; they are instantaneous state transitions.

8. (*Offline planning*) The planner is not concerned with any change that may occur in the system while it is planning: it plans for the given initial and goal states, regardless of the current dynamic, if any.

A system considering all the assumption is called a *restricted model* and is the main subject of *Classical Planning* (Section 3.1.2). Probabilistic planning (Section 3.1.4) usually releases the assumptions 3 and 5. Some techniques of probabilistic planning also consider planning in partially observable environment, releasing the assumption 2.

3.1.2 Classical planning

Classical planning deals with restricted state-transition systems, i.e. ones that meet the 7 assumptions. In [Ghallab et al., 2004], Ghallab et al. defined formally the classical planning problem for a system as follows : given a system $\Sigma = (S, A, \gamma)$, S being the set of states, A being the set of actions (also called operators) and γ being the progression function (that gives the state produced by applying an action a to a state s) ; a planning problem is a triple $\mathcal{P} = (\Sigma, s_0, g)$ where s_0 is an initial state and g a set of goal states. A solution to $\mathcal{P}$ is a sequence of actions $(a_1, a_2, ..., a_k)$ such as $s_1 = \gamma(s_0, a_1), ..., s_k = \gamma(s_{k+1}, a_k)$. The purpose of planning is to find the optimal solution to a planning problem $\mathcal{P}$, that is to say to minimize the sequence of actions.

The General Problem Solver (GPS) [Newell et al., 1959] was one of the first and the most well known automatic planner for restricted state-transition systems. It can solve simple problems as the well known Tower of Hanoï (Figure 3.1) but is very limited to solve higher problems due to combinatorial explosion. Indeed, GPS uses a means-end search process (Figure 3.2) : the

solver determines what is to change in the environment to get closer to the goals, and chooses an operator to perform this change. This operator can contain another sub-goal to achieve in its preconditions. The solver chooses another operator to achieve this sub-goal and so on.
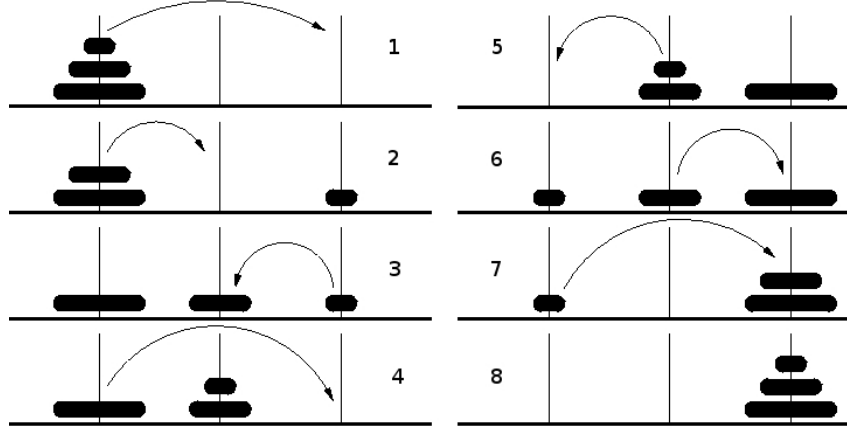

Figure 3.1: A solution to the 3-disk Tower of Hanoï puzzle. In the Tower of Hanoï puzzle, the player must move a set of disk from an initial peg to a final one, adhering to three basic rules : 1/ Move only one disk at a time. 2/ A larger disk may not be placed on top of a smaller disk. 3/ All disks, except the one being moved, must be on a peg. [Smyth, 2014]

The STRIPS planner [Fikes and Nilsson, 1972] is the second important planner to be developed, also using means-end analysis. STRIPS works under the closed world assumption which states that a statement that is true is also known to be true, or in other words anything that is not explicitly stated is considered false. As the General Problem Solver, STRIPS is a linear planner, which means that it tries to satisfy one goal completely before working on another goal and cannot undo a satisfied goal. This technique assumes that the sub-goals to be solved are independent and that the solver cannot change the plan made for a sub-goal later. Sussman showed in [Sussman, 1974] that there exists simple problems where this type of planning can not provide a solution, as illustrated on Figure 3.3. Three blocks are placed on a table with the configuration depicted on Figure 3.3a. An agent must move the blocks to place them as depicted on Figure 3.3b. The agent can move only one block at a time. Linear planners will decompose the goal into two subgoals:

1. Place block A on top of block B

2. Place block B on top of block C

Suppose that the planner starts by satisfying the sub-goal 1, it will move block C out of the way, then move block A on top of block B, as depicted on Figure 3.3c. According to the principles of linear planning, the sub-goal 1 is satisfied and cannot be undone. However, to satisfy the sub-goal 2, the agent needs to remove block A from block B, undoing the sub-goal 1. If the planner decide to start by satisfying the sub-goal 2, it will place block B on top of block C, as depicted on Figure 3.3d. Once again, the agent is not able to satisfy the sub-goal 1 without undoing the sub-goal 2.

Partial order planners [Sacerdoti, 1975, Chapman, 1987, Penberthy and Weld, 1992] have been developed to overcome this issue. Partial order planners are based on the principle of
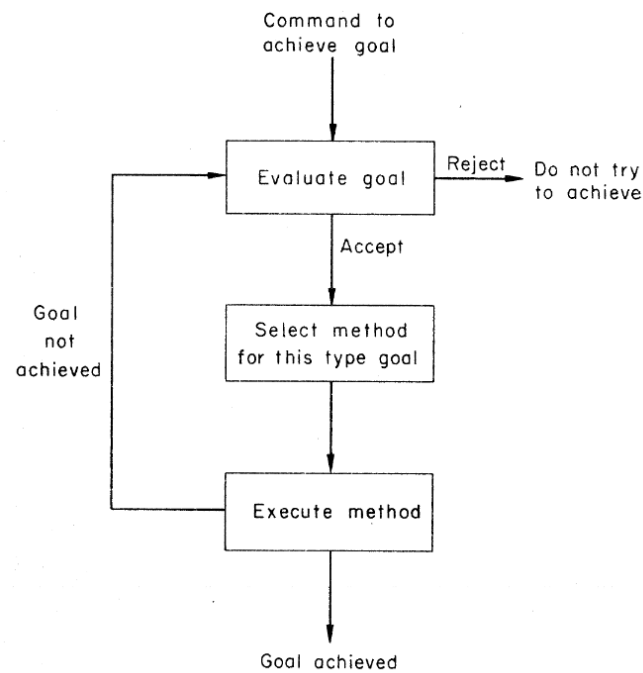
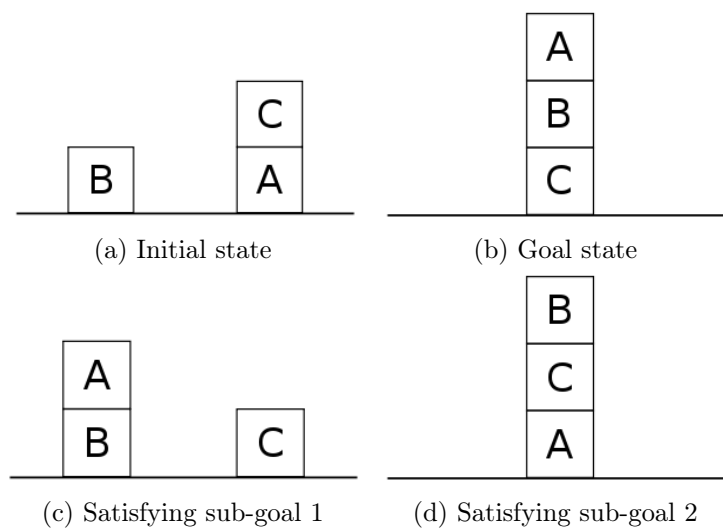Figure 3.2: Executive Organization of GPS. [Newell et al., 1959]

(a) Initial state

(b) Goal state

(c) Satisfying sub-goal 1

(d) Satisfying sub-goal 2

Figure 3.3: Sussman Anomaly

least commitment, which states that decision should be deferred as long as possible before being taken, so that when they are taken the probability of their correctness is maximized [Stefik, 1981]. Partial order planners are still concerned with the restricted state-transition systems. However all those hypotheses are impossible to assume as soon as we consider a real-like application. Indeed, as we already mentioned, real applications are characterized by the presence of uncertainty. It is so mandatory that a planner is able to deal with uncertainty.

3.1.3 Classical planning and uncertainty

Several logics have been developed to deal with uncertainty. According to [Halpern, 2003a], three criteria can influence the choice of a logic: (1) the underlying representation of uncertainty (2) the degree of significance of quantitative reasoning (do we need to talk about the probability of events or could we only rank these events by order of likelihood) (3) the notions being reasoned about.

Epistemic modal logic, introduced in [Wright, 1951] and [Hintikka, 1962], provides a way to reason about Epistemic uncertainty[11]. Epistemic logic has been widely studied, and extends the propositional logic with several new operators [Fagin et al., 1995] :

- $K_a\phi$, read "agent a knows that ϕ"

- $E_G\phi$, read "every agent in group G knows that ϕ"

- $C_G\phi$, read "it is common knowledge to every agent in G that ϕ"

- $D_G\phi$, read "it is distributed knowledge to every agent in G that ϕ"

Common knowledge is defined as : ϕ is a common knowledge in a group of agents G if all the agents in G know that they know ϕ, all the agents in G know that they all know that they know ϕ, and so ad infinitum. Distributed knowledge is defined as : ϕ is a distributed knowledge in a group of agents G if an agent which knows everything that each member of G knows knows ϕ.

Multi-valued logic [Béziau, 1997], and more precisely fuzzy logic [Zadeh, 1988] are another way to deal with Epistemic uncertainty. In multi-valued logic, variables have more than two truth values. In fuzzy logic, the truth value is a degree between 0 and 1. It is so possible to represent notions as "old" and "young", instead of "25 years old".

In addition to the logics, several planners have also been developed to handle uncertainty. In [Petrick and Bacchus, 2002], Petrick and Bacchus introduced PKS, a planner that reasons at the knowledge level and is able to produce conditional plans in the presence of incomplete knowledge. In this planner, based on STRIPS, the agent's knowledge is represented by a set of databases and actions are represented as updates to these databases. Thus, actions are defined at a knowledge level instead of a state level. The authors extended this model in [Petrick and Bacchus, 2004] with new mechanisms. This type of knowledge-level reasoning has been reused in more recent works [Pistore et al., 2005, Palacios and Geffner, 2009].

Some studies also focused on uncertainty in the result of actions and so dealt with Aleatory uncertainty. In [Smith and Weld, 1998], Smith and Weld extended the planner Graphplan [Blum and Furst, 1997] to handle uncertainty in the initial state and in action effects. It assumes that no information at all is available at run-time. The resulting plan chooses robust actions that cover all eventualities. In [Bertoli et al., 2001], Bertoli et al. considered environment with partial observability and suggested an algorithm that generates plans guaranteed to achieve the goal despite of the uncertainty in the initial condition and the effects of actions.

---

[11]As presented in Chapter 2, Figure 2.1

However, due to the uncertainty in the results of actions, a plan[12] doesn't always exist and a policy[13] may be preferred. For such cases, probabilistic planning may be preferred to classical planning.

3.1.4 Probabilistic planning

In a non-deterministic system, an action may succeed, fail totally or fail partially. This uncertainty involves that the plan we need to compute may not be a linearly ordered sequence of actions but a function that gives the action to perform given a input. On top of that, in non-deterministic systems, the goals specification should take into account this non-determinism and cannot be restricted to goal states. Therefore, we consider in this section systems that release the hypothesis of Deterministic systems 3, Sequential plans 6 and Restricted goals 5. In the following, we will refer the set *environment + agents situated in this environment* as *system*. If an event is described as affecting the system, it can affects the environment or/and the agents. In this section, we will introduce the basic concepts of probabilistic planning for one agent. The multi-agent case will be considered in section 3.3.

State

A state describes a possible situation of the system. It is usually made up of a certain number of descriptors whose values can change among time. In most of the applications, the system is closed, which means that all the possible states are described in a set of states. It is not possible for the system to end in a state which is not in this set. On the contrary, an open system can be in a state which is not included in the initial description.

Actions

The actions describe how the agent can influence the system. Some actions have direct effect on the system (provoking a change of states) and some actions only influence the agent's internal representation. Those latest are called epistemic actions.

Transition function

The transition function is a function that takes as input a starting state, a final state and an action and give as an output the probability for the system to reach the final state from the starting state when the agent acts with the given action. It describes the way the system can evolve through the actions of the agent. The uncertainty on the actions' result is modeled in the transition function. An action is said deterministic if it always leads to the same final state given the starting state. In other word, the probability associated to the tuple $< startingstate, action, finalstate >$ is 1 for one final state and 0 for all others. A transition function is called stationary if does not depend on the time.

Observability

A system can be totally observable or partially observable. In totally observable systems, the agent has a perfect knowledge of the state of the system at each time step. In partially observable systems, the agent receives observations providing indications about the current state of the

---

[12]That is to say a sequence of actions that guarantees the goal achievement
[13]That is to say a function that outputs the action to perform given some input

system. The agent needs to maintain an internal representation of the possible states and their likelihood. This internal representation is called the agent's *belief state*. The *observation function* gives the probability for an agent to receive a specific observation when doing a given action in a given state.

Belief state

The belief state of the agent is used when the system is partially observable. It describes the agent's internal representation of the current state of the system. This belief state is updated each time an observation is received. The most common way to manipulate an agent's beliefs is to use a probability distribution over the space of states. However other measures are possible like the Dempster-Shafer theory and the Possibility measures presented in Section 2.2

Policy

A policy describes the strategy the agent follows to behave. It is a function that gives the action to process given a certain input. This input can be the current state of the system, the current belief state of the agent, a sequence of observations or events... The purpose of the planning problem is to find the optimal policy given a set of states, a set of actions, a transition function, a belief state and observability properties.

3.2 Probabilistic planning for single agent : Markov Decision Processes frameworks

3.2.1 Full-Observability

Markov Decision Processes (MDP) [Puterman, 1994] are the most common framework to deal with decision making under uncertainty in fully observable domains. A Markov Decision Process is defined as a controlled stochastic process satisfying the Markov porperty and assigning reward values to state transitions [Sigaud and Buffet, 2010]. The Markov property [Markov, 1960] assumes that the probability to reach a state s_{t+1} from a state s_t after performing an action a_t depends neither on the previous states $s_0 \ldots s_{t-1}$ nor on the previous actions $a_0 \ldots a_{t-1}$. More formally, for each $n \geq 0$ and for each sequence of states $(s_0 \ldots s_{n-1})$, the Markov property states that

$$P(s_{n+1}|s_0, a_0, s_1, a_1, \ldots s_n, a_n) = P(s_{n+1}|s_n, a_n)$$

Definition 3.1 (MDP). A Markov Decision Process (MDP) is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, H, T, \mathcal{R} \rangle$ where,

- $\mathcal{S}$ is a finite set of possible states

- $\mathcal{A}$ is a finite set of possible actions

- H is a set of time steps where decisions need to be made.

- $T : \mathcal{S}, \mathcal{A}, \mathcal{S} \to [0, 1]$ is the transition function with $T(s, a, s') = P(s'|s, a)$

- $R : \mathcal{S}, \mathcal{A} \to \mathbb{R}$ is the reward function

---

The set H is the *horizon* of the MDP. It is discrete but can be either finite or infinite. In some cases, the process should run until a goal state[14] is reached. In that case, the horizon is said to be indefinite: the number of steps is finite but unknown. The reward function gives for each couple state-action $\langle s, a \rangle$ the reward earned by the agent if it performs action a while being in state s. In a lot of cases, this reward function is simplified in $R : \mathcal{S} \to \mathbb{R}$ and gives the reward of an agent being in a given state, as illustrated in figure 3.4.

-0.01	-0.01	-0.01	-0.01	1
-0.01				-0.01
-0.01		-0.01		-1
-0.01	-0.01	-0.01		-0.01
-0.01 START	-0.01	-0.01	-0.01	-0.01

Figure 3.4: An example of an environment with a reward function defined on states

Policies and optimality

A policy $\pi : \mathcal{S} \to \mathcal{A}$ is a function that maps the set of states $\mathcal{S}$ with the set of actions $\mathcal{A}$ and gives for each $s \in \mathcal{S}$ the action $a \in \mathcal{A}$ to perform. A policy is said stationary if it does not depend on time.

The purpose of planning in an MDP is to find the policy π^* that gives the maximum expected reward within the horizon H. To do so, an optimality criterion needs to be defined. Four criteria are usually considered depending on the problem [Sigaud and Buffet, 2010] :

1. the finite criterion: $E[r_0 + r_1 + \ldots + r_{N-1}|s_0]$.

   A prior assumption linked to this criterion implies that the agent has to control the system withing N steps, N being finite.

2. the γ-discounted criterion: $E[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots + \gamma^t r_t + \ldots |s_0]$.

   This criterion is the most commonly used in the infinite horizon problems. γ represents the difference of important between immediate rewards and future rewards.

3. the total-reward criterion: $E[r_0 + r_1 + \ldots + r_t + \ldots |s_0]$.

   This criterion corresponds to the γ-discounted criterion with $\gamma = 1$, which is meaningful in some specific cases.

4. the average-reward criterion: $\lim_{n \to \infty} \frac{1}{n} E[r_0 + r_1 + \ldots + r_{n-1}|s_0]$.

---

[14]also called terminal state

This criterion is usually used when decisions have to be made frequently with a discount factor close to one or when it appears impossible to value directly the rewards.

Based on the chosen optimality criterion, a value function $V^\pi : \mathcal{S} \to \mathbb{R}$ can be defined, which gives for each $s \in \mathcal{S}$ the expected gain when starting from state s and applying policy π. The value functions for the usual criteria are :

1. the finite criterion: $\forall s \in \mathcal{S} \ V_N^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{N-1} r_t | s_0 = s, \pi\right]$

2. the γ-discounted criterion: $\forall s \in \mathcal{S} \ V_N^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \pi\right]$

3. the total-reward criterion: $\forall s \in \mathcal{S} \ V_N^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} r_t | s_0 = s, \pi\right]$

4. the average-reward criterion: $\forall s \in \mathcal{S} \ V_N^\pi(s) = \lim_{n\to\infty} \mathbb{E}\left[\frac{1}{n}\sum_{t=0}^{n-1} r_t | s_0 = s, \pi\right]$

In the following we will only consider the γ-discounted criterion, which is the most used for infinite horizon. An optimal value function V^{π^*} is a value function such as $\forall \pi \in \Pi \ \forall s \in \mathcal{S}$ $V^\pi(s) \leq V^{\pi^*}(s)$, where Π is the set of all possible policies. Therefore, optimal policies π^* are those which maximize the value function : $\pi^* \in \underset{\pi \in \Pi}{\operatorname{argmax}} V^\pi$. Dynamic programming enable to compute an optimal policy for a given MDP. Different algorithms exist in the literature. In this section, we will detail the two basis algorithms : Value Iteration and Policy Iteration. Both relies on the Bellman equation.

Bellman equation

Bellman and Dreyfus showed in [Bellman and Dreyfus, 1962] that a dynamic optimization problem in discrete time can be stated in a recursive step-by-step form. This involves to express the relationship between the value function at one time step and the value function at the next time step. This relationship is called the Bellman equation. The Bellman equation for the value function associated to the γ-discounted criterion is :

$$\forall s \in \mathcal{S}, V_t(s) = \max_{a \in \mathcal{A}} \left[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s')V_{t-1}(s') \right]$$

Given $\gamma \in [0,1[$, the optimal value function is the unique solution of the Bellman equation.

Compute the optimal policy : Value Iteration

The Value Iteration algorithm [Bellman, 1957] computes the optimal value function V^* thanks to backward induction. It is detailed on algorithm 3.1. The algorithm starts with an arbitrarily defined value function (line 1), and then compute for each time step, for each possible state, the best value over all the possible actions (lines 3 to 8). For an infinite horizon, this refinement of the value function will never ends. Therefore an threshold ϵ is used to determine when the value function is good enough (Line 8). Finally, the policy is computed based on the optimal value function and returned (lines 9 to 12). The Value Iteration algorithm has been proved to converge in [Puterman, 1994] and its complexity is $\mathcal{O}(|\mathcal{S}|^2||\mathcal{A}|)$.

Input: $\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \epsilon$
Result: the optimal policy $\pi*$

1 Assign V_0 arbitrarily for all $s \in \mathcal{S}$;
2 $k \leftarrow 0$;
3 repeat
4 $k \leftarrow k + 1$;
5   **foreach** *state* $s \in \mathcal{S}$ **do**
6 $V_k(s) = \max\limits_{a \in \mathcal{A}} \left[R(s,a) + \gamma \sum\limits_{s' \in \mathcal{A}} T(s,a,s') \times V_{k-1}(s') \right]$
7   **end**
8 until $\max\limits_{s \in \mathcal{S}} (|V_k(s) - V_{k-1}(s)|) < \epsilon$;
9 foreach *state* $s \in \mathcal{S}$ **do**
10 $\pi*(s) = \operatorname*{argmax}\limits_{a \in \mathcal{A}} \left[R(s,a) + \gamma \sum\limits_{s' \in \mathcal{S}} T(s,a,s') \times V_k(s') \right]$
11 end
12 return $\pi*$

Algorithm 3.1: The Value Iteration algorithm. In this version, only the value function $V(s)$ is stored. Other versions store the function $Q(s,a) = \mathcal{R}(s) + \gamma \sum_{s' \in \mathcal{A}} T(s,a,s') \times V(s')$. In these versions, the storage needs increases, but it is then easier to build the optimal policy.

Compute the optimal policy : Policy Iteration

The Policy Iteration algorithm [Howard, 1960] starts with an arbitrarily defined policy and iteratively improves it. To do so, the algorithm relies on the value function V^{π_i}, expressed as a set of $|\mathcal{S}|$ linear equations in $|\mathcal{S}|$ unknown variables. The unknown variables are the values of $V^{\pi_i}(s)$. Three steps are followed to improve the policy :

1. Policy evaluation : determine $V^{\pi_i}(s)$. The linear equations can be solved iteratively or by a linear equation solution method such as Gaussian elimination. This step is often time-consuming.

2. Policy improvement : choose $\pi_{i+1}(s) = \operatorname*{argmax}\limits_{a \in \mathcal{A}} \left[R(s,a) + \gamma \sum\limits_{s' \in \mathcal{S}} T(s,a,s') V^{\pi_i}(s') \right]$

3. Test the stopping condition : compare the updated policy π_{i+1} to the previous π_i. The algorithm stops if $\forall s \in \mathcal{S}, \pi_{i+1}(s) = \pi_i(s)$.

The policy iteration is detailed on algorithm 3.2. The Policy Iteration algorithm always halts, is has also been proved to converge in [Puterman, 1994], and its complexity is $\mathcal{O}(|\mathcal{S}^2||\mathcal{A}|)$.

Factored MDP

The MDP as described previously doesn't use structured representation and requires the enumeration of all the possible states in the problem. The complexity of this enumeration grows exponentially with the number of features to be considered in the problem, involving a big issue to solve large problems. Factored Markov Decision Processes (FMDP), proposed by Boutilier et al. [Boutilier et al., 1995, Boutilier et al., 1999] are extensions to MDP that exploits problems structures to represent the transition and reward functions compactly. In FMDPs, the environment is described as a set of features of interest, each feature being described by a random

---

    Input: $\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}$
    Result: the optimal policy $\pi*$

1 repeat

2 $noChange \leftarrow true$;

3   Solve $V(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s, \pi(s), s') \times V(s')$;

4   **foreach** $s \in \mathcal{S}$ **do**

5 $QBest = V(s)$;

6   **foreach** $a \in \mathcal{A}$ **do**

7 $Qsa = R(s) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s') \times V(s')$;

8   **if** $Qsa > QBest$ **then**

9 $\pi(s) \leftarrow a$;

10 $QBest \leftarrow Qsa$;

11 $noChange \leftarrow false$;

12   **end**

13   **end**

14   **end**

15 until $noChange$;

16 return π

Algorithm 3.2: The Policy Iteration algorithm

variable X_i which can take values in its domain $DOM(X_i)$. $X = (X_1, \ldots, X_n)$ is the multivariate variable describing a state. Then, a possible state is an instantiation of each random variable X_k and can be written as a vector $x = (x_1, \ldots, x_n)$, such as $\forall 1 \geq i \leq n$, $x_i \in DOM(X_i)$. $DOM(X)$ is the set of all possible instantiations for the multivariate variable X. Therefore, the state space $\mathcal{S}$ is defined by $\mathcal{S} = DOM(X)$.

In real problems, the transition of a variable often depends on a small number of other variables. FMDPs exploit this dependence properties to represent the transition function as a Dynamic Bayesian Network (DBN) [Dean and Kanazawa, 1989], where the nodes are the variables X_i and the edges the dependencies between variables. Therefore, the transition from a state x to another state x' is written :

$$P(x'|x) = \prod_i P(x_i'|Parents(x_i'))$$

with x_i' being the value of the variable X_i' in state x', and $Parents(x_i')$ being the values of the variables that are parents of X_i' in the DBN.

A similar representation can be used for the reward function: the reward function can be factored additively into a set of localized reward functions, each of which only depending on a small set of variables. Let $W_1 \ldots W_p, \forall i$, $W_i \subset \{X_1, \ldots, X_n\}$ be a set of clusters of variables. Let $R_1 \ldots R_p$ be a set of reward functions with $\forall i$, $R_i : DOM(W_i) \to \mathbb{R}$. The reward function associated to a state x and an action a is defined as

$$R(x, a) = \sum_{i=1}^{p} R_i(x[W_i], a)$$

with $x[W_i]$ being the instantiation of the variables of the cluster W_i in the state x.

Several techniques enable to solve FMDP efficiently, such as the Structured Value Iteration and Structured Policy Iteration algorithms [Boutilier et al., 2000] that used decision trees to represent the different functions of the problem (transition, reward, policy and value functions). Linear programming has also been suggested in [Koller and Parr, 2000] to solve FMDP, but their scalability is limited in large problems. In [Guestrin et al., 2003], Guestrin et al. used approximation algorithms and function-specific independence to decompose the constraints of the initial problem into a set of constraints with a complexity depending on the structure of the problem rather than on its size.

3.2.2   Partial Observability

The Markov Decision Processes rely on the very strong hypothesis that the agent has a perfect knowledge of the current state of the system at each time step. Unfortunately, this assumption is not real in a lot of problems and particularly while dealing with event exploration where, by definition, the state of the system is not known and need to be explored. Partially Observable Markov Decision Processes, introduced in [Sondik, 1978], are an extension of MDPs to deal with the the incomplete information on the state of the system. A classic example of POMDP is the Tiger Problem, depicted in Figure 3.5. One agent is in front of two doors. Behind one door is a great reward, behind the other door is a tiger which will attack the agent if the door is opened. The agent can open one door or listen to hear some noise and gain information about the position of the tiger. However, listening is not a fully reliable action and can sometimes produce incorrect outputs like not hearing a noise from the left door even if the tiger is behind.
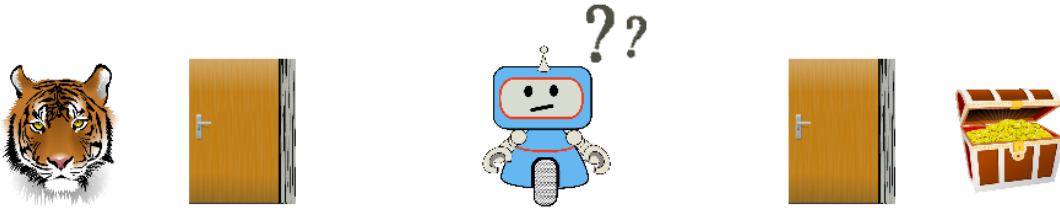

Figure 3.5: The Tiger Problem

Definition 3.2 (POMDP)**.** A Partially Observable Markov Decision Process (POMDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \Omega, H, \mathcal{T}, \mathcal{O}, \mathcal{R}, b_0 \rangle$ where,

- $\mathcal{S}$ is a finite set of possible states

- $\mathcal{A}$ is a finite set of possible actions

- Ω is a finite set of possible observations

- H is a set of time steps where decisions need to be made

- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function with $T(s, a, s') = P(s'|s, a)$

- $\mathcal{O} : \Omega \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the observation function with $\mathcal{O}(\omega, s, a) = P(\omega|s, a)$

- $R : \mathcal{S}, \mathcal{A} \to \mathbb{R}$ is the reward function

- b_0 is the initial belief state

The set of states $\mathcal{S}$, the set of actions $\mathcal{A}$, the set of time steps H, the transition function T and the reward function R are defined as in MDPs. In the Tiger Problem, the set of state would be $\{tiger-left, tiger-right\}$ and the set of actions $\{listen-left, listen-right, open-left, open-right\}$. The reward is highly positive if the action $open-left$ (respectively $open-right$) is executed in the state $tiger-right$ (respectively $tiger-left$) and highly negative otherwise.

In a POMDP, the agent doesn't know the exact state of the system but receives observations which give it information about the current state of the system, as illustrated on Figure 3.6. In
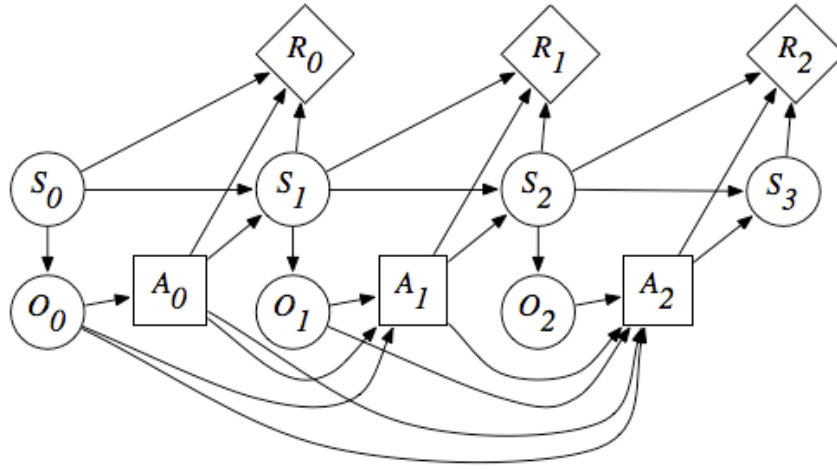
Figure 3.6: A POMDP as a dynamic decision network. From [Poole and Mackworth, 2010]

general, an observation can correspond to more than one states which creates ambiguity in the possible current state. The set Ω represents all the possible observations an agent can receive. The observation function $\mathcal{O}$ gives the probability to receive a specific observation $\omega \in \Omega$ when a certain action a_t is done in a certain state s_t. In the Tiger Problem, the possible observations are $noise$ and $no-noise$. The observation $noise$ is very likely if the action $listen-left$ (respectively $listen-right$) is executed in the state $tiger-left$ (respectively $tiger-right$) and the observation $no-noise$ is very likely in the other configurations.

Belief states

Since the agent doesn't know the exact state of the system, it needs to act according to the available information. This available information is summed up in the agent's belief state. A belief state is a probability distribution on the state space. The set of all belief states is written $\mathcal{B}$. The agent's belief state b is updated after each transition, that is to say after executing the action a and receiving the observation o. The most used mechanism is the Bayesian update, in which the new belief state becomes $b_o^a(s') = P(s'|b, a, o)$, which can be developed as [Cassandra et al., 1997]:

$$b_o^a(s') \;=\; \frac{O(\omega, s', a) \sum_{s \in \mathcal{S}} T(s, a, s') b(s)}{\sum_{s' \in \mathcal{S}} O(\omega, s', a) \sum_{s \in \mathcal{S}} T(s, a, s') b(s)}$$

$$\;=\; \frac{O(\omega, s', a) \sum_{s \in \mathcal{S}} T(s, a, s') b(s)}{P(\omega | b, a)}$$

Other types of belief updates have been suggested to improve the efficiency of the POMDP is some cases. For instance Izadi and Precup considered in [Izadi and Precup, 2005] that obtaining a reward can be a useful information to distinguish the current state and suggested a belief update step that uses this reward. In [Seymour and Peterson., 2009], Seymour and Peterson included trust mechanisms to improve the performance of Interactive-POMDPs[15]

Finding policies in POMDP

Due to partial observability, it is not possible to define a policy on the state space for POMDPs. The agent has only access to the observations it received. However, policies based on the last observation received have been proved to be non-optimal [Littman, 1994a]. Therefore the agent needs to consider the history of observations received, which breaks the Markov Property. To overcome this issue, it is possible to base the policy on the belief states, that sum up all information received from the observations until the current decision step. Therefore a policy in a POMDP is a function which maps the belief space to the action space : $\pi_{POMDP} : \mathcal{B} \to \mathcal{A}$. The belief state being Markovian, it is possible to transform a POMDP into a *belief-MDP*, which is an MDP where the states are the belief states. The belief-MDP is defined as a tuple $\langle \mathcal{B}, \mathcal{A}, \tau, r \rangle$ where :

- $\mathcal{B}$ is the set of belief states of the POMDP

- $\mathcal{A}$ is the same finite state of actions as for the POMDP

- $\tau : \mathcal{B} \times \mathcal{A} \times \mathcal{B} \to [0, 1]$ is the new transition function defined on the belief state space

- $r : \mathcal{B} \times \mathcal{A} \to \mathbb{R}$ is the new reward function defined on the belief state space

The new functions τ and r can be easily derived from the old functions T and R. Indeed,

$$\tau(b, a, b') = \sum_{\omega \in \Omega} P(b'|b, a, o) \sum_{s \in \mathcal{S}} O(\omega, s, a) b(s)$$

were

$$P(b'|b, a, o) = \begin{cases} 1 \text{ if the update of the belief state } b \text{ with action a and observation } o \text{ is } b' \\ 0 \text{ otherwise} \end{cases}$$

r can be written $r(b, a) = \sum_{s \in \mathcal{S}} b(s) R(s, a)$

The Bellman equation applied to this belief-MDP is :

$$V_t(b) = \max_{a \in \mathcal{A}} \left[\sum_{s \in \mathcal{S}} b(s) R(s, a) + \gamma \sum_{\omega \in \Omega} P(\omega | b, a) V_{t-1}(b_o^a) \right]$$

---

[15]I-POMDPs will be dealt more precisely in section 3.3.2.

where b_o^a is the updated belief state, obtained by updating b with the observation o and the action a. And so, the optimal policy is defined for each b as

$$\pi_t * (b) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left[\sum_{s \in \mathcal{S}} b(s) R(s, a) + \gamma \sum_{\omega \in \Omega} P(\omega | b, a) V_{t-1}(b_o^a) \right]$$

The value function of a POMDP has been proved to be piecewise linear and convex for finite horizons [Sondik, 1978] and so V^* corresponds to the upper surface of a set of value functions for the all the possible policies, that is to say hyperplanes through the belief-space, as depicted on Figure 3.7. Those hyperplane also defined a partition through the belief state space. Each hyperplane is represented by a $|\mathcal{S}|$-dimensional vector, called α-vector, which contains the coefficients of the equation of the hyperplane. Therefore, a value function for a finite horizon can be represented as a set of α-vectors. [Smallwood and Sondik, 1973] proved that there is an optimal action for each partition.
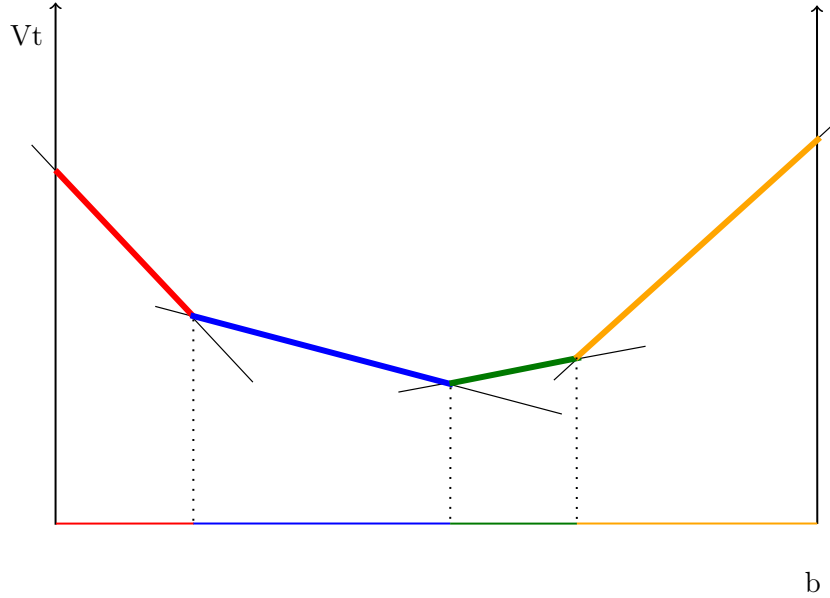
Figure 3.7: The value function for a POMDP with 2 states and its associated belief partition. In that case, the belief state space is one-dimensional since $b(s_1) = 1 - b(s_2)$. The optimal value function $V*$ corresponds to the upper surface of all the possible value functions.

Exact Value Iteration algorithm [Sondik, 1978, Cassandra et al., 1997] exploits the α vectors representation to solve a POMDP optimally. It rewrites the steps of the Value Iteration for MDP using the α-vectors. The new value function is computed as follows:

$$
\begin{aligned}
V' &= \bigcup_{a \in \mathcal{A}} V^a \\
V^a &= \bigoplus_{\omega \in \Omega} V^{a,o} \\
V^{a,o} &= \left\{ \frac{1}{|\Omega|} R_a + \alpha^{a,o} : \alpha \in V \right\} \\
\alpha^{a,o}(s) &= \sum_{s' \in \mathcal{S}} O(o, s', a) T(s, a, s') \alpha(s')
\end{aligned}
$$

with R_a is the vector representation of the reward function, $R_a(s) = R(s, a)$. The complexity of this algorithm is $O(|V| \times |\mathcal{A}| \times |\Omega| \times |\mathcal{S}|^2 + |\mathcal{A}| \times |\mathcal{S}| \times |V|^{|\Omega|})$ [Shani et al., 2013]. Therefore, it is only scalable to small domains. In [Littman, 1994b], Littman presented Witness, an algorithm to solve efficiently POMDPs. Witness uses Q-values to generate the elements of the optimal value function directly. The Q-value $Q_t^a(b)$ is the value of taking action a from belief state b and then following the optimal $(t-1)$-step policy. This algorithm is still exponential in the worst case, but polynomial for almost all realistic sample problems. Improvements have been made to exact algorithms by using gradient-based techniques [Meuleau et al., 1999], dynamic programming [Poupart and Boutilier, 2003], or constrained linear programs [Amato et al., 2006]. Even if those algorithms scale better, they are still not applicable to large problems.

A important breakthrough has been made by [Lovejoy, 1991] which proposed to use point-based method to make a good approximation of the value function by discretizing the belief state space. Then the value function can be maintained on the subset of selected points thanks the linear program below:

$$
\begin{aligned}
&\text{Minimize} && \sum_{i=1}^{|\bar{V}|} w_i \cdot v_i \\
&\text{Subject to} && b = \sum_{i=1}^{|\bar{V}|} w_i \cdot b_i \\
& && \sum_{i=1}^{|\bar{V}|} w_i = 1 \\
& && w_i \in [0, 1]
\end{aligned}
$$

Hauskrecht improved in [Hauskrecht, 2000] the method by selecting for the discretization only belief states that can be reached from the initial state. From that point, different fast and efficient algorithms have been suggested to solve approximately POMDPs like PBVI [Pineau et al., 2003], PERSEUS [Spaan and Vlassis, 2005], SARSOP [Kurniawati et al., 2008] or GarMin [Poupart et al., 2011]. Other approximate methods have been developed to solve POMDPs like a combination between the Bellman equation and shortest path algorithms [Geffner and Bonet, 1998], online algorithms [Ross and Chaib-Draa, 2007] and Monte Carlo planning [Silver and Veness, 2010, Bai et al., 2011].

3.3 Multi-agent concerns : coordination and cooperation

Dealing with multi-agent settings brings a lot of new challenges, depending on the relations between the agents in the system. Pool and Mackworth defined two extreme type of relations [Poole and Mackworth, 2010]:

- fully-cooperative: the agents share the same utility function and so work for a system-goal

- fully-competitive: an agent can only win when another loses.

Between those two extrema, intermediate systems can be created, when agents' share some goals but also have some personal objectives.

In information gathering and event exploration, agents are usually considered fully-cooperative : the team's goal is to gain as much information as possible about the environment. Therefore coordination has to be taken into account so that the exploration is efficient.

3.3.1 Decentralized POMDP and equivalent models

Without explicit communication

Decentralized POMDP [Bernstein et al., 2002] is a model able to deal with multiagent decision making which principle is depicted on figure 3.8. In a Dec-POMDP, the planning is made offline and so is completely centralized : a single computer computes the joint plan and distributes it to the agents. The online execution is completely decentralized : each agent executes its own plan and receives its own observations.
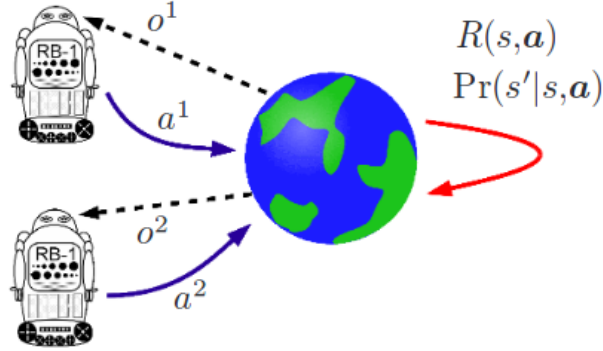

Figure 3.8: Execution of a decentralized POMDP : At each stage, the agents independently take an action. The environment undergoes a state transition and generates a reward depending on the state and the actions of both agents. Finally each agent receives an individual observation of the new state. From [Oliehoek, 2012]

Definition 3.3 (Dec-POMDP). A Decentralized Partially Observable Markov Decision Process (Dec-POMDP) is defined as a tuple $\langle \mathcal{D}, \mathcal{S}, \mathcal{A}, T, R, \Omega, \mathcal{O}, h \rangle$ where

- $\mathcal{D} = \{1 \ldots n\}$ is a set of n agents

- $\mathcal{S}$ is a finite set of states s

- $\mathcal{A} = \times_{i \in \mathcal{D}} A^i$ is a finite set of joint actions, each A^i being a set of actions available to agent i

- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function

- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function

- $\Omega = \times_{i \in \mathcal{D}} O^i$ is a finite set of joint observations, each O^i being a set of observations available to agent i

- $\mathcal{O} : \Omega \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the observation function

- h is the horizon of the problem

During execution, the agents are assumed to act based on their own individual actions and no explicit communication is assumed. However, the Dec-POMDP model can deal with commu-

nication through the state of environment, for instance with an action modifying something in the environment and an observation reporting this modification.

Planning in Dec-POMDP means computing a joint-policy $\pi = \times_{i \in \mathcal{D}} \pi^i$, each π^i being the policy for agent i. Since the agents only have access to their own actions and observations during the execution, and the transition and observation functions are defined with joint actions and joint observations, it is impossible in a Dec-POMDP to maintain an individual belief state as in POMDP. Therefore, the policies π^i cannot be defined on a Markovian signal as is the belief state in POMDP. It means that planning in a Dec-POMDP requires to match individual histories to actions. The most complete history an agent can maintain is the action-observation history, that is to say the sequence of actions taken by and observations received. However, when dealing with deterministic policies[16], it is sufficient to consider only the observation history.

Definition 3.4 (Observation history [Oliehoek, 2012])**.** The observation history (OH) for agent i, written $\bar{\omega}^i$, is defined as the sequence of observations an agent has received. At a specific time step t, this is :
$$\bar{\omega}^i_t = \left(\omega^i_1, \ldots, \omega^i_t \right)$$
The joint observation history is the OH for all agents $\bar{\omega}_t = \left\langle \bar{\omega}^1_t, \ldots, \bar{\omega}^n_t \right\rangle$. The set of observation histories for agent i at time t is denoted $\bar{\Omega}^i_t$. The set of all possible observations histories for an agent i is written $\bar{\Omega}^i$ and the set of all possible joint observation histories is written $\bar{\Omega}$.

Therefore, the individual policies are defined as $\pi^i : \bar{\Omega}^i \to \mathcal{A}_i$ and the joint policy is defined as $\pi : \bar{\Omega} \to \mathcal{A}$. The value function of a Dec-POMDP is given by :

$$V^\pi(s_t, \bar{\omega}_t) = \begin{cases} R(s_t, \pi(\bar{\omega}_t)) \text{ for the last stage } t = h - 1 \\ R(s_t, \pi(\bar{\omega}_t)) + \displaystyle\sum_{s_{t+1} \in \mathcal{S}} \sum_{\omega \in \Omega} P(s_{t+1}, \omega | s_t, \pi(\bar{o}_t)) V^\pi(s_{t+1}, \bar{\omega}_{t+1}) \text{ for } t \neq h - 1 \end{cases}$$

Generating a joint policy for a Dec-POMDP is NEXP-complete [Bernstein et al., 2002]. However, a Dec-POMDP can be reduced to a single agent MDP if a free-communication is assumed, that is to say agents are able to communicate at every timestep during execution. In that case, the policy can be computed in polynomial time [Pynadath, 2002]. Though this assumption is extremely strong and cannot hold in most of real cases.

The Multiagent Team Decision Problem (MTDP) [Pynadath, 2002] is another framework for decision-making in multiagent teams, very similar to Dec-POMDPs. Both models have been proved to be equivalent, that is to say that their corresponding decision problems are complete for the same complexity class. For a complete description of the models and proof of their equivalence, see [Seuken and Zilberstein, 2008].

With explicit communication

The Decentralized Partially Observable Markov Decision Process with Communication (Dec-POMDP-COM) framework [Goldman and Zilberstein, 2003] is an extension of the Dec-POMDP that allows explicit communication between the agents.

---

[16]That is to say policies that associate a single action to an input signal. Those are the only policies we consider in this thesis.

Definition 3.5 (Dec-POMDP-COM). A Dec-POMDP-COM is a tuple $\langle \mathcal{D}, \mathcal{S}, \mathcal{A}, T, \Omega, \mathcal{O}, \Sigma, C_{\Sigma}, R_A, R, h \rangle$ where

- $\mathcal{D}, \mathcal{S}, \mathcal{A}, T, \Omega, \mathcal{O}, h$ are defined as in Dec-POMDPs

- Σ is an alphabet of communication messages. $\sigma_i \in \Sigma$ is an atomic message sent by agent i and $\vec{\sigma} = \langle \sigma_1, \ldots, \sigma_n \rangle$ is a joint message, that is to say a tuple of all messages sent by the agents in one time step. The set of all possible joint messages is written $\vec{\Sigma}$.

- $C_{\Sigma} : \Sigma \rightarrow \mathbb{R}$ is the message cost function. $C_{\Sigma}(\sigma_i)$ is the cost for transmitting atomic message σ_i

- $R_A : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, identical to the reward function in Dec-POMDPs

- $R : \mathcal{S} \times \mathcal{A} \times \vec{\Sigma} \rightarrow \mathbb{R}$ is the total reward function, defined as $R(s', a, \vec{\sigma}) = R(s', a) - \sum_{i \in \mathcal{D}} C_{\Sigma}(\sigma_i)$

In Dec-POMDP-COM, two types of policies are computed : an *action* policy and a *communication* policy. A local action policy for an agent i, written π_i^A, is a mapping from local histories of observations and histories of messages received to control actions : $\pi_i^A : \Omega_i \times \vec{\Sigma} \rightarrow \mathcal{A}_{\rangle}$. A local communication policy for an agent i, written π_i^{Σ}, is a mapping from local histories of observations and histories of messages received to communication actions : $\pi_i^{\Sigma} : \Omega_i \times \vec{\Sigma} \rightarrow \Sigma$. A joint policy for a Dec-POMDP-COM $\pi = \langle \pi_1, \ldots, \pi_n \rangle$ is a tuple of local policies where each π_i is composed of the communication and action policies for agent i. It has been shown that it is possible to transform any Dec-POMDP-COM into an equivalent Dec-POMDP [Goldman and Zilberstein, 2004].

The COM-MTDP [Pynadath, 2002] is an extension to MTDP including explicit communication, as Dec-POMDP-COM is for Dec-POMDP. The principles are very similar to those explained for Dec-POMDP-COM : a joint policy has to be computed, which is composed of a communication policy and an action policy. As for Dec-POMDP and MTDP, Dec-POMDP-COM and COM-MTDP have been proved to be equivalent [Seuken and Zilberstein, 2008]. Since any Dec-POMDP-COM can be transformed into an equivalent Dec-POMDP, the four models (Dec-POMDP, Dec-POMDP-COM, MTDP and COM-MTDP) are equivalent [Seuken and Zilberstein, 2008].

Several extensions of Dec-MDP and Dec-POMDP have been suggested to overcome the complexity of resolution, each of them assuming different hypothesis. In [Beynier and Mouaddib, 2005], Beynier and Mouaddib considered the case where the agents in the system have limited interactions. Each agent is given a task to perform and temporal constraints are defined between the tasks. They suggested a polynomial algorithm to solve this problem. In [Becker, 2004], Becker suggested an algorithm to solve transition-independent Dec-MDPs. This class of problem is characterized by two or more cooperative agents solving (mostly) independent local problems. The actions taken by one agent cannot affect any other agents' observation or local state. Observation independence and more generally locality of interaction [Oliehoek et al., 2008] have also been used to solve Dec-MDP and Dec-POMDP.

3.3.2   I-POMDP

The Interactive POMDP framework [Gmytrasiewicz and Doshi, 2005] considers extending the POMDP network by considering not only the belief over the underlying system state but also the belief over the other agents. To do so, Gmytrasiewicz and Doshi defined the *frame*, the *type* and the *model* of an agent.

Definition 3.6 (Agent's frame). A frame of an agent i is $\hat{\theta}_i = \langle \mathcal{A}_i, \Omega_i, T_i, O_i, R_i, OC_i \rangle$ where :

- $\mathcal{A}_i$ is a set of actions agent i can execute

- Ω_i is a set of observations the agent i can receive

- $T_i : \mathcal{S} \times \mathcal{A}_i \times \mathcal{S} \to [0,1]$ is a transition function

- $O_i : \Omega \times \mathcal{S} \times \mathcal{A}_i \to [0,1]$ is an observation function

- $R_i : \mathcal{S} \times \mathcal{A}_i \to \mathbb{R}$ is a reward function

- OC_i is the agent's optimality criterion that specifies how rewards acquired over time are handled.

Definition 3.7 (Agent's type). A type of an agent i is $\theta_i = \left\langle b_i, \hat{\theta}_i \right\rangle$ where :

- b_i is agent i's state of belief

- $\hat{\theta}_i$ is agent i's frame

Definition 3.8 (Model of an agent). An agent's model $m_j \in M_j$ is a triple $\langle h_j, f_j, O_j \rangle$ where

- $h_j \in H_j$ is a possible history of agent j's observations

- $f_j : H_j \to \Delta(A_j)$ is agent j's function, assumed computable, which maps possible histories of j's observation to distribution over its actions

- O_j is a function describing the way the environment supplies the agent j with its input (function related to the observation function of the POMDPs)

Based on these new notions, an I-POMDP is defined as follows [Gmytrasiewicz and Doshi, 2005] .

Definition 3.9 (I-POMDP). An Interactive Partially Observable Markov Decision Process (I-POMDP) is a tuple $\langle IS_i, \mathcal{A}, T_i, \Omega_i, \mathcal{O}_i, R_i \rangle$ where

- IS_i is a set of interactive states defined as $IS_i = \mathcal{S} \times_{j=1}^{n-1} M_j$, where $\mathcal{S}$ is the set of states of the physical environment and M_j is the set of possible models of agent j

---

- $\mathcal{A} = \mathcal{A}_i \times \mathcal{A}_j$ is a set of joint actions of all agents

- T_i is a transition function. An assumption called *Model non manipulability assumption* is made, stating that agents' actions do not change the other agents' model directly. Given this assumption, the transition function is defined as $T_i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$

- Ω_i is a set of agent i's possible observations

- $\mathcal{O}_i$ is an observation function. An assumption called *Model non observability* is made, stating that agents cannot observe other's model directly. Given this assumption, the observation function is defined as $\mathcal{O}_i : \Omega_i \times \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$

- $R_i : IS_i \times \mathcal{A} \rightarrow \mathbb{R}$ is a reward function. Usually, this function is simplified by considering only physical states and so defined as $R_i : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

Under the Model Non Manipulability and the Model Non Observability assumptions, the belief update function for an I-POMDP given the interactive state $is^t = \left\langle s^t, m_j^t \right\rangle$ is:

$$
b_i^t(is^t) = \beta \sum_{is^{t-1} : m_j^{\hat{t}-1} = \hat{\theta}_j^t} b_i^{t-1}(is^{t-1}) \sum_{a_j^{t-1}} P(a_j^{t-1} | \theta_j^{t-1}) \mathcal{O}_i(\omega_i^t, s^t, a^{t-1})
$$
$$
\cdot T_i(s^{t-1}, a^{t-1}, s^t) \sum_{\omega_j^t} \tau_{\theta_j^t}(b_j^{t-1}, a_j^{t-1}, \omega_j^t, b_j^t) \mathcal{O}_j(s^t, a^{t-1}, \omega_j^t)
$$

where:

- β is a normalizing constant

- $\sum_{is^{t-1} : m_j^{\hat{t}-1} = \hat{\theta}_j^t}$ is the summation over all interactive states where agent j's frame is $\hat{\theta}_j^t$

- b_j^{t-1} and b_j^t are the belief elements of agent j's type θ_j^{t-1} and θ_j^t

- $P(a_j^{t-1} | \theta_j^{t-1})$ is the probability that action a_j was taken by agent j during the last time step, given its type.

- $\mathcal{O}_j$ is agent j's observation function in m_j^t

- $\tau_{\theta_j^t}(b_j^{t-1}, a_j^{t-1}, \omega_j^t, b_j^t)$ represents the update of agent j's belief

As in POMDP, each belief state in an I-POMDP has an associated value reflecting the maximum payoff an agent can expect in that belief state:

$$
U(\theta_i) = \max_{a_i \in \mathcal{A}_i} \left\{ \sum_{is} ER_i(is, a_i) b_i(is) + \gamma \sum_{\omega_i \in \Omega_i} P(\omega_i | a_i, b_i) \cdot U \left(\left\langle SE_{\theta_i}(b_i, a_i, \omega_i), \hat{\theta}_i \right\rangle \right) \right\}
$$

where $ER_i(is, a_i) = \sum_{a_j} R_i(is, a_i, a_j) P(a_j | m_j)$ is the expected reward for state is taking action a_i.

In I-POMDPs, an agent i's belief state includes beliefs over another agent j's belief state. However, this agent j's belief state includes also belief over agent i's belief state. Therefore, the update of agent i's beliefs includes the update of agent j's beliefs, which in turn includes the update of agent i's beliefs and so on. This nested belief update is an obvious issue for computing

the optimal solution. To deal with that infinite nesting, Gmytrasiewicz and Doshi suggested to use a *strategy level l*, which enables to terminate the recursion when computing the belief update and the value function. However, Seuken and Zilberstein noticed that it is doubtful that the finitely nested I-POMDP model allows for approximate optimality and this problem remains an open research question [Seuken and Zilberstein, 2008].

In [Doshi et al., 2009] Doshi et al. proposed a graphical representation for I-POMDPs, called *Interactive Influence Diagram* (I-ID) and *Interactive Dynamic Influence Diagram* (I-DID), which generalize the Influence Diagrams (ID) and Dynamic Influence Diagrams (DID) representations for POMDP, introduced in [Boutilier and Poole, 1996], to multiagent settings.

The I-POMDP model is more expressive than the Dec-POMDP model since it allows non-cooperative systems. However, to our knowledge, there is no optimal algorithm for solving infinitely nested I-POMDP. Moreover, even if some approximations and algorithms have been suggested to solve finitely-nested I-POMDP [Rathnasabapathy et al., 2006, Doshi and Perez, 2008, Doshi and Gmytrasiewicz, 2009], this task remains extremely challenging and hard to scale.

3.4   Decision models for event exploration

Event exploration is a subtype of active sensing problems : agents need to act in order to gather information about their environment, to improve their beliefs. The goal of the system is to optimize the information gathering. Some kind of active sensing problems can be modeled using POMDP by rewarding, if successful, a final action consisting in expressing the system's best guess of classification. In [Guo, 2003] Guo described a POMDP framework in which the actions are using a specific sensor or outputting a classification label. The agent is rewarded only if the classification label is correct, and penalized otherwise. This reward system force the agent to wait until enough information has been collected so that the agent can be sure of the label it outputs. By using rewards on states and not on belief states, the authors also remain in the standard POMDP framework. In [Ji and Carin, 2007], Ji and Carin used a similar system, but coupled it with the training of a Hidden Markov Model. Later, in [Spaan et al., 2010], Spaan et al. applied this classifying approach to multi-agent settings in order to perform active cooperative perception using classifying actions. In this model, the cooperative perception lies in the observation model, as the false positive and false negative rates are different depending on the position and characteristics of sensors. The agents are rewarded the first time they correctly classified an event as interesting, and penalized the first time they don't classify it correctly.

However, all active sensing problems cannot be defined using classifying actions and in some applications, the objective needs to be explicitly described as reducing the uncertainty about the state. In those cases, the standard POMDP framework is not adapted, since the reward function needs to be expressed on the belief states instead of system states. In [Spaan, 2008], Spaan presented the problem of rewards based on belief states, raised some relevant issues both on modeling the problem as well as solving the underlying POMDP. This study discussed the first steps toward a decision-theoretic approach to cooperative perception, but without providing general solution. The paper also focuses on a single decision-maker and the problem of cooperation and coordination between the agents is not really addressed. In [Eidenberger and Scharinger, 2010], Eidenberger and Scharinger presented an active perception POMDP-based framework where a single agent has to modify the position of its sensors to recognize objects. In this study, the reward function is based on the expected belief state after applying the action, balanced with the cost of doing this action. This POMDP is not solved using literature algorithm since the

value function is not piecewise linear. The resolution technique suggested provides a myopic strategy. In [Araya-Lopez et al., 2010] Araya-Lopez et al. introduced the ρPOMDP, which is an extension of the POMDP in which the reward function ρ can be defined on the belief states. They showed that the ρPOMDP's value function keeps its convexity property if the reward function ρ is convex. They also showed that algorithms from the literature can be easily adapted to ρPOMDPs if the reward function is piecewise linear, and that all convex reward functions can be approximated by a piecewise linear convex function. In [Satsangi et al., 2015] (extended version in [Satsangi et al., 2014]), Satsangi et al. suggested a POMDP-based framework and a new point-based method for dynamic sensor selection. Using theoretical results from [Araya-Lopez et al., 2010] and the negative belief entropy reward, they demonstrated the submodularity of value functions and bounded the error of greedy maximization. However, this work considered a single decision-maker to select the sensors.

3.5 Conclusion

Planning consists in reasoning before acting. In this chapter, we presented the two major subfield of planning : classical planning and probabilistic planning. Classical planning usually deals with restricted models and produces a plan, that is to say a sequence of actions that guarantees the correct output. However when considering uncertainty on actions' result and external events, a plan may not exist. In such cases, policies are usually more adapted. Probabilistic planning, and more specifically Markov Decision Processes and extensions, is an appropriate framework to produce policies in different settings. However, the classical (PO)MDP framework is not suitable to solve active sensing problems due to the reward function which is based on real states. Extensions have been suggested to enable a reward function on belief states, usually based on Entropy measures. However, those extensions address the problem of active sensing in single-agent settings and the multi-agent case remains opened.

In this thesis, we propose a complete decentralized decision-theoretic framework for multi-agent active sensing. In this framework, each agent will need to quantify its uncertainty about the state of the world and the state of other agents, and to plan actions in order to reduce this uncertainty. However, since we are concerned with dynamic systems, the agent need to check and revise its beliefs among time, by exploring again features it already discovered. All the agents in the system also need to coordinate themselves in a decentralize way, and so to exchange their information. Since exchanging all information would be counter-productive, they will need to select relevant information to communicate.

Part II

Contributions

Chapter 4

Theory of relevance

Contents

An agent evolving in an environment perceives this environment through its sensors and acquire partial information. The agent must then gather all information it receives into a internal representation, that is usually referred as the agent's *knowledge*. When several agents cooperate in a single environment, they must exchange information they gathered to be efficient. However exchanging all information all the time is not possible in some real applications because of many reasons such as limited resources of communication and performance. The agents must communicate only interesting information to the other agents. In this chapter, we consider the notion of information relevance to assess its usefulness. Then we define a degree of relevance to quantify the relevance that a piece of information has for a given agent.

4.1   What does it mean to be relevant?

As seen in section 2.3, the relevance of some information for a given agent is usually defined as how much this information answers the agent's request. But how to characterize relevance when the agents don't express their requests explicitly? In these cases, the need for information has to be inferred based on the agent's current knowledge.

In this section we gathered some important properties that a piece of information should satisfy to be considered as relevant for a given agent.

Property 4.1. Relevant information is correct.

In this thesis' context, information being *correct* means that it reflects the real state of the environment. Since the environment is dynamic, correct information may become incorrect later. However, to be considered relevant it is important that at the information is correct at the moment it is considered. Indeed, as already mentioned in section 2.3, incorrect information would degrade agent's knowledge and so cannot be considered as useful for the agent.

Property 4.2. Information is relevant for a specific agent at a specific time.

Since agents in a system may have different states of knowledge about the environment, information that is relevant for a given agent may be less relevant for another one with a different state of knowledge, for instance if the second agent already knows the information. This property states that the relevance of information should only be considered regarding the agent which receives the information. In addition the environment and the knowledge of the agent are modified among time. Therefore, some information, relevant for a given agent at a certain time, may not be relevant anymore later. Indeed, the agent may have got the information in between or the information may not be true anymore.

Property 4.3. Relevant information is new.

"New" means here that the agent didn't know this information previously. The goal of agents in event exploration is to detect events, that is to say changes in the environment. Therefore, any new information is relevant since it corresponds to an event. Similarly to Property 4.2, information is new only for a given agent.

Property 4.4. Relevant information improves the precision of agent's knowledge.

Sometimes, information may not be new for a given agent, but may confirms what an agent already believed. Indeed passing time and sensors' precision affect the quality of information an agent had received. Therefore information that confirms previously acquired uncertain knowledge is relevant since it improves the agent's knowledge's precision. In the remaining of this thesis we will call this kind of information *precising information* (by analogy with new information).

4.2   Agents observe and believe

4.2.1   Environment and states

In this section we describe the framework we used to model the environment and the agent's knowledge. Let us assume an agent i situated in an environment $\mathcal{E}$. The environment is modeled

as a set of features of interest. Each feature is described using a discrete variable X_k which takes values in its domain $DOM(X_k)$.

$$\mathcal{E} = \bigotimes_{\forall k} X_k$$

Let $DOM(\mathcal{E})$ be the set of all possible values for all $X_k \in \mathcal{E}$.

$$DOM(\mathcal{E}) = \langle DOM(X_k) \rangle_{\forall X_k \in \mathcal{E}}$$

Example 4.1 (Patrol in a working building) Using the running example defined in section 0.7, we consider two variables per room: *RoomEmpty* and *RoomOnFire*. Each variable has two possible values: *true* or *false* . Considering the 5 rooms of the example, we have:

$$\begin{aligned}
\mathcal{E} = \langle &\langle RoomEmpty, RoomOnFire \rangle_1 \\
&\langle RoomEmpty, RoomOnFire \rangle_2, \\
&\langle RoomEmpty, RoomOnFire \rangle_3, \\
&\langle RoomEmpty, RoomOnFire \rangle_4 \\
&\langle RoomEmpty, RoomOnFire \rangle_5 \rangle
\end{aligned}$$

and

$$DOM(\mathcal{E}) = \{true, false\}$$

4.2.2 Agent's beliefs

Agent i has some beliefs $\mathcal{B}_i^{\mathcal{E}}$ concerning the features of interest modeled as probability distributions over the $X_k \in \mathcal{E}$.

$$\mathcal{B}_{i,t}^{\mathcal{E}} = \bigotimes_{\forall X_k \in \mathcal{E}} b_{i,t}^k$$

with $b_{i,t}^k$ being a probability distribution over the variable X_k at time t, representing agent i's knowledge about the variable X_k.

$$b_{i,t}^k = \bigotimes_{\forall x_p \in DOM(X_k)} P(X_k = x_p)$$

For the sake of readability the beliefs of agent i can also be represented by a vector:

$$\mathcal{B}_{i,t}^{\mathcal{E}} = \begin{pmatrix} b_{i,t}^1 \\ \vdots \\ b_{i,t}^k \\ \vdots \\ b_{i,t}^{|\mathcal{E}|} \end{pmatrix}$$

In most applications the agents have no prior knowledge about the system. All distributions in its belief state follow the uniform distribution in which all the possible values are equiprobable,

that is to say

$$\mathcal{B}_{i,0}^{\mathcal{E}} = \begin{pmatrix} \frac{1}{|X_1|} \\ \vdots \\ \frac{1}{|X_k|} \\ \vdots \\ \frac{1}{|X_{|\mathcal{E}|}|} \end{pmatrix}$$

Example 4.2 (Patrol in a working building) The initial belief state for each agent in the system is:

$$\mathcal{B}_{a,0}^{\mathcal{E}} = \begin{pmatrix} (0.5, 0.5) \\ (0.5, 0.5) \\ (0.5, 0.5) \\ (0.5, 0.5) \\ (0.5, 0.5) \\ (0.5, 0.5) \\ (0.5, 0.5) \\ (0.5, 0.5) \\ (0.5, 0.5) \\ (0.5, 0.5) \end{pmatrix} \begin{matrix} \%RoomEmpty_1 \\ \%RoomOnFire_1 \\ \%RoomEmpty_2 \\ \%RoomOnFire_2 \\ \%RoomEmpty_3 \\ \%RoomOnFire_3 \\ \%RoomEmpty_4 \\ \%RoomOnFire_4 \\ \%RoomEmpty_5 \\ \%RoomOnFire_5 \end{matrix}$$

Information is an uncountable concept which is not possible to manipulate as it is in a multi-agent system and in belief update mechanisms. We need to consider atomic pieces of information that we call *observations*.

Definition 4.1 (Observation). An *observation* is a high-level piece of information, that the agent receives and that gives an indication about the state of the environment, that is to say about the current values of variables X_k. An observation shall be related to at least one variable.

Example 4.3 (Patrol in working building) **Room 1 is empty**, **Room 3 is not empty**, **4 rooms are empty** are three different examples of observations.

By receiving new observations, the agents in the system will update their belief state by modifying the probabilities associated to each value. The way to do this modification depends on the application. Bayesian networks and Bayes conditioning are often used [Darwiche, 2008]. If some observation o is received and does not have a probability of 0 according to the current beliefs, then the beliefs are updated as follows:

$$\forall X_k \in \mathcal{E}, \forall x \in DOM(X_k), P(X_k = x | o) = \frac{P(X_k = x \wedge o)}{P(o)}$$

4.3   How much relevant is this observation? The degree of relevance

In event-exploration, the agents need to communicate the most relevant observations to each others. To allow relevance comparison, we need to define an agent-based degree of relevance

which quantifies the relevance of a specific observation at a time t, given an agent receiving it and its belief state at that time t.

In this section, we will assume that the observation considered is correct. Indeed, and as said previously, we assume that an observation cannot be relevant if it is incorrect. It is therefore useless to compute the degree of relevance of an incorrect observation. The problem of assessing the correctness of an observation in the decision making process will be discussed in Chapter 5, Section 5.3.4.

To define a degree of relevance, we need to formalize the novelty of an observation (Section 4.3.1) and how much this observation is sound (Section 4.3.2). Then we combine those measures in an agent-based degree of relevance (Section 4.3.3).

4.3.1 Measuring novelty: the Hellinger Distance

Let us consider an agent i at time t, with a belief state $\mathcal{B}_{i,t}^{\mathcal{E}}$. This agent receives an observation o and updates its belief state, which becomes $\mathcal{B}_{i,t+1}^{\mathcal{E}}$. Intuitively, we consider that the observation is new for agent i if it modifies significantly its belief state. To quantify this modification, we need to compare $\mathcal{B}_{i,t}^{\mathcal{E}}$ and $\mathcal{B}_{i,t+1}^{\mathcal{E}}$, which are two vectors of probability distributions over all the variables X_k. Therefore we need to compare pairs of probability distributions for each variable X_k, that is to say $b_{i,t}^k$ and $b_{i,t+1}^k$.

Several measures exists to compare probability distributions, the most well-known and used being the Kullback-Leibler divergence [Kullback and Leibler, 1951], shortened $\mathcal{KL}$ divergence. Let us consider two probability distribution P and Q over a discrete random variable X which can take n values. The $\mathcal{KL}$ divergence, denoted $D_{KL}(P||Q)$, measures the information lost when Q is used to approximate P and is defined as follows:

$$D_{KL}(P||Q) = \sum_{j=1}^{n} P(j) \ln \frac{P(j)}{Q(j)}. \tag{4.1}$$

The $\mathcal{KL}$ divergence has some interesting properties: it is non negative, additive for independent distributions and admits an upper bound [Sayyareh, 2011]. These properties and its popularity encouraged us to consider the $\mathcal{KL}$ divergence as our novelty measure. We defined the degree of novelty of a given observation received by a given agent as the $\mathcal{KL}$ divergence between its belief state before receiving the observation and its belief state after update with the received observation. The $\mathcal{KL}$ divergence between two belief states is defined as the sum of the $\mathcal{KL}$ divergences of the probability distributions in these belief states.

$$D_{KL}(\mathcal{B}_{i,t_1}^{\mathcal{E}}||\mathcal{B}_{i,t_2}^{\mathcal{E}}) = \sum_{X_k \in \mathcal{E}} \sum_{p=1}^{n} b_{i,t_1}^k(x_p) \ln \frac{b_{i,t_1}^k(x_p)}{b_{i,t_2}^k(x_p)} \tag{4.2}$$

The degree of novelty of an observation o received by an agent a is so defined as follows:

$$nov_i(o) = D_{KL}(\mathcal{B}_{i,t}^{\mathcal{E}}||\mathcal{B}_{i,t+1}^{\mathcal{E}}) \tag{4.3}$$

where $\mathcal{B}_{i,t+1}^{\mathcal{E}} = update(\mathcal{B}_{i,t}^{\mathcal{E}}, o)$ is agent i's belief state after update with the observation o. As a sum of convex non negative function, this degree of novelty is still convex non negative.

Despite its interesting properties, the $\mathcal{KL}$ divergence is not symmetric: the $\mathcal{KL}$ divergence between P and Q is generally not equal to the $\mathcal{KL}$ divergence between Q and P. This involves that the $\mathcal{KL}$ divergence is not a metric, which is a major drawback to its use as a measure of novelty. Indeed, our degree of novelty aims at measuring how different two probability distributions are.

It is important that the difference between $\mathcal{B}_{i,t}^k$ and $\mathcal{B}_{i,t+1}^k$ is the same as the difference between $\mathcal{B}_{i,t+1}^k$ and $\mathcal{B}_{i,t}^k$.

Another measure of divergence can be considered, presenting the same interesting properties as the $\mathcal{KL}$ divergence, but being symmetric: the Hellinger distance [Nikulin, 2015]. Given two probability distributions P and Q over a discrete random variable which can take n values, the Hellinger distance is defined as follows:

$$D_H(P,Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{j=1}^{n} \left(\sqrt{P(j)} - \sqrt{Q(j)} \right)^2} \tag{4.4}$$

As the $\mathcal{KL}$ divergence, the Hellinger distance is convex, non negative and bounded: $0 \leq D_H(P,Q) \leq 1$. Its symmetry also ensure that the distance between $\mathcal{B}_{i,t}^k$ and $\mathcal{B}_{i,t+1}^k$ is the same as the distance between $\mathcal{B}_{i,t+1}^k$ and $\mathcal{B}_{i,t}^k$.

Similarly to what we did with the $\mathcal{KL}$ divergence, we can define the degree of novelty for a given observation received by a given agent using the Hellinger distance, as presented in Definition 4.2.

Definition 4.2 (Degree of novelty). The degree of novelty of an observation o received by an agent i at time t is defined by

$$
\begin{aligned}
nov_{i,t}(o) &= D_H(\mathcal{B}_{i,t}^{\mathcal{E}} || \mathcal{B}_{i,t+1}^{\mathcal{E}}) \\
&= \sum_{X_k \in \mathcal{E}} D_H(b_{i,t}^k || b_{i,t+1}^k) \\
&= \sum_{X_k \in \mathcal{E}} \frac{1}{\sqrt{2}} \sqrt{\sum_{x_p \in DOM(X_k)} \left(\sqrt{b_{i,t}^k(x_p)} - \sqrt{b_{i,t+1}^k(x_p)} \right)^2}
\end{aligned}
\tag{4.5}
$$

where $\mathcal{B}_{i,t+1}^{\mathcal{E}} = update(\mathcal{B}_{i,t}^{\mathcal{E}}, o)$ is agent i's belief state after update with the observation o, $b_{i,t}^k$ is the probability distribution representing the belief of agent i concerning the variable X_k, $b_{i,t}^k(x_p)$ is the single probability in this distribution corresponding to $X_k = x_p$.

Property 4.5. The degree of novelty is convex.

Proof. The Hellinger Distance is convex, so $D_H(b_{i,t}^k || b_{i,t+1}^k)$ is convex. The degree of novelty being the sum of convex function is also convex. $\qquad \square$

Property 4.6. The degree of novelty is bounded: $\forall i, \forall t$

$$0 \leq D_H(\mathcal{B}_{i,t}^{\mathcal{E}} || \mathcal{B}_{i,t+1}^{\mathcal{E}}) \leq |\mathcal{E}|$$

Proof. The Hellinger distance is bounded, $0 \leq D_H(b_{i,t}^k || b_i i, t+1^k) \leq 1$ By summing over all the variables $X_k \in \mathcal{E}$, we obtain $0 \leq D_H(\mathcal{B}_{i,t}^{\mathcal{E}} || \mathcal{B}_{i,t+1}^{\mathcal{E}}) \leq |\mathcal{E}|$ $\qquad \square$

4.3.2 Measuring the precision: the Shannon entropy

As previously, we consider an agent i at a time t with a belief state $\mathcal{B}_{i,t}^{\mathcal{E}}$ receiving an observation o. We stated in property 4.4 that an observation is relevant if it improves the precision of agent i's belief state. A very common way to measure precision is to use the Shannon entropy. Initially, the Shannon entropy measure the information contained in a source. This source is represented by a discrete random variable X which has n possible values and which is associated to a probability distribution P. The Shannon entropy is defined as follows:

$$H_b(X) = -\sum_{j=1}^{n} P(j) \log_b P(j) \tag{4.6}$$

where b is the base of the logarithm used. In this thesis, we consider only $b = 10$ and simplify the notation as $H(X) = -\sum_{j=1}^{n} P(j) \log P(j)$.

The Shannon entropy is continuous, symmetric and concave. It is also bounded: $0 \leq H_b(X) \leq \log_b(n)$, the maximum being reached when all the possible values are equiprobable.

Using the Shannon entropy, we define the entropy of a belief state as presented in definition 4.3.

Definition 4.3 (Entropy of a belief state). Given an agent i and its belief state at time t $\mathcal{B}_{i,t}^{\mathcal{E}}$, the entropy of the belief state of agent i is given by:

$$\begin{aligned} H(\mathcal{B}_{i,t}^{\mathcal{E}}) &= \sum_{X_k \in \mathcal{E}} H(b_{i,t}^k) \\ &= -\sum_{X_k \in \mathcal{E}} \sum_{x_p \in DOM(X_k)} b_{i,t}^k(x_p) log(b_{i,t}^k(x_p)) \end{aligned} \tag{4.7}$$

where $b_{i,t}^k$ is the probability distribution representing the belief of agent i concerning the variable X_k and $b_{i,t}^k(x_p)$ is the single probability in this distribution corresponding to $X_k = x_p$.

Using this definition we can assess how precising an observation o is for an agent i by comparing the entropy of agent i's belief state before receiving the observation $H(\mathcal{B}_{i,t}^{\mathcal{E}})$ with the entropy of agent i's belief state after update with the observation $H(\mathcal{B}_{i,t}^{\mathcal{E}})$.

Definition 4.4 (Degree of soundness). Given an agent i and its belief state at time t $\mathcal{B}_{i,t}^{\mathcal{E}}$ and an observation o received by agent i, the degree of soundness of the observation o for the agent i, describing how much the observation is sound, is given by:

$$sound_{i,t}(o) = H(\mathcal{B}_{i,t}^{\mathcal{E}}) - H(\mathcal{B}_{i,t+1}^{\mathcal{E}}) \tag{4.8}$$

where $\mathcal{B}_{i,t+1}^{\mathcal{E}} = update(\mathcal{B}_{i,t}^{\mathcal{E}}, o)$ is agent i's belief state after update with the observation o.

Property 4.7. The degree of soundness is bounded: $\forall i, \forall t$

$$- \sum_{X_k \in \mathcal{E}} \log(|DOM(X_k)|) \le sound_{i,t}(o) \le \sum_{X_k \in \mathcal{E}} \log(|DOM(X_k)|)$$

Proof. The Shannon entropy is bounded: $\forall X_k, 0 \le H(X_k) \le \log(|DOM(X_k)|)$. Therefore, we have

$$0 \le H(\mathcal{B}_{i,t}^{\mathcal{E}}) \le \sum_{X_k \in \mathcal{E}} \log(|DOM(X_k)|)$$

and

$$0 \le H(\mathcal{B}_{i,t+1}^{\mathcal{E}}) \le \sum_{X_k \in \mathcal{E}} \log(|DOM(X_k)|)$$

so

$$- \sum_{X_k \in \mathcal{E}} \log(|DOM(X_k)|) \le -H(\mathcal{B}_{i,t+1}^{\mathcal{E}}) \le 0$$

We can conclude

$$- \sum_{X_k \in \mathcal{E}} \log(|DOM(X_k)|) \le H(\mathcal{B}_{i,t}^{\mathcal{E}}) - H(\mathcal{B}_{i,t+1}^{\mathcal{E}}) \le \sum_{X_k \in \mathcal{E}} \log(|DOM(X_k)|)$$

$\square$

Unfortunately, we can not conclude about the concavity property of the degree of soundness. Indeed, the Shannon entropy is concave, so $H(\mathcal{B}_{i,t}^{\mathcal{E}})$ and $H(\mathcal{B}_{i,t+1}^{\mathcal{E}})$ are concave. Therefore $-H(\mathcal{B}_{i,t+1}^{\mathcal{E}})$ is convex and it is difficult to conclude anything about the convexity of the sum.

4.3.3 An agent-based degree of relevance

The relevance degree of an observation is a combination between the novelty of this observations and soundness. In some cases both can be satisfied[17], but in some other cases a compromise need to be found. For instance, an observation which results from a change in the environment concerning a feature that has already been detected is new - and so very relevant - but degrades agent's beliefs.

We defined the degree of relevance of an observation for an agent as follows:

Definition 4.5 (Degree of relevance)**.** The degree of relevance of an observation o for an agent i, noted $rel_i(o)$, is given by

$$rel_{i,t}(o) = (1 - \delta) \frac{nov_{i,t}(o)}{|\mathcal{E}|} + \delta \frac{sound_{i,t}(o)}{H_{max}} \tag{4.9}$$

where $nov_{i,t}(o)$ - respectively $sound_{i,t}(o)$ - is the degree of novelty - respectively soundness - of the observation o received by the agent i, $\delta \in [0, 1]$ is a weight to model how dynamic the environment is, and $H_{max} = \sum_{X_k \in \mathcal{E}} \log(|DOM(X_k)|)$ is the maximum belief state's entropy - reached when all variables follow the uniform distribution.

---

[17]which is the case when the agent has no knowledge about a variable: the received observation is both new and precising

The values $|\mathcal{E}|$ and H_{max} are used to normalize the degree of relevance. The parameter δ depends on the application considered and enables to tune the degree of relevance. Indeed, in highly dynamic systems - δ close to 0 - the values of the variables X_k may often change. In this case, it is important that novelty is strongly rewarded so that the agent updates its beliefs to the new values frequently. In the opposite case - static system and δ close to 1 - each variable X_k is likely to keep its value among time and novelty is not crucial. It is however important that the agent reaches a very accurate belief state, and so a reduction of entropy is strongly rewarded. δ is directly related to a physical parameter which is the probability that a change occurs in the system: $\delta = 1 - ProbabilityOfChange$.

Property 4.8. The degree of relevance is bounded: $\forall i, \forall o$

$$-\delta \leq rel_{i,t}(o) \leq 1$$

Proof. From property 4.7, we know that $\forall i, \forall o$:

$$-H_{max} \quad \leq \quad sound_{i,t}(o) \quad \leq \quad H_{max}$$

Since $H_{max} > 0$,

$$-1 \quad \leq \quad \frac{sound_{i,t(o)}}{H_{max}} \quad \leq \quad 1$$

Since $\delta \geq 0$,

$$-\delta \quad \leq \quad \delta\frac{sound_{i,t(o)}}{H_{max}} \quad \leq \quad \delta \tag{4.10}$$

From property 4.6, we know that $\forall i, \forall o$:

$$0 \quad \leq \quad nov_{i,t}(o) \quad \leq \quad |\mathcal{E}|$$
$$0 \quad \leq \quad \frac{nov_{i,t}(o)}{|\mathcal{E}|} \quad \leq \quad 1$$

Since $0 \leq \delta \leq 1$, we know $1 - \delta \geq 0$,

$$0 \quad \leq \quad (1-\delta)\frac{nov_{i,t}(o)}{|\mathcal{E}|} \quad \leq \quad 1-\delta \tag{4.11}$$

From inequalities 4.10 and 4.11, we can conclude that

$$-\delta \quad \leq \quad rel_i(o) \quad \leq \quad 1$$

$\square$

Since the degree of soundness $sound_{i,t}(o)$ is not proved to be either convex or concave, we can unfortunately not conclude concerning the convexity of the degree of relevance.

4.4 Conclusion

Assessing the relevance of information is very important for efficient communication. In this chapter, we studied some properties that information should have to be relevant. It appeared that relevance includes novelty and soundness of information. Those two properties are concurrent and a compromise must be found. Based on this analysis, we defined a degree of relevance that combines a degree of novelty and a degree of soundness. The degree of novelty uses the Hellinger distance to measure the difference between the agent's beliefs before and after receiving

an observation. The degree of soundness uses the Shannon entropy to measure the accuracy of the agent's beliefs. The degree of relevance defined is bounded. A parameter δ can be tuned to adapt the degree to different types of application. δ models the dynamicity of the environment. In highly dynamic environments, the novelty should be favored over the soundness since the information is likely to change. However, in little dynamic environments, information should be sounder since it is less likely to change.

Chapter 5

The MAPING Model

Contents

A team work cannot be efficient without cooperation, coordination and communication. In critical scenarios such as military conflicts, search and rescue missions, border or protected area surveillance, etc., the communication may be limited for safety or technical reasons. In those situations, it is also unsafe to rely on only one agent to separate the missions among other agents. A fully decentralized decision framework is necessary. In this chapter we define a complete offline-online framework to enable a team of agents to perform fully decentralized event exploration under limited communication constraints.

5.1 The MAPING framework: general overview

In multiagent event exploration problems, the agents of the system should coordinate their activities in order accomplish efficiently their mission. The coordination can be centralized or decentralized. In centralized coordination, a central decision process distributes the goals among the agents and ensure a good coordination. In these systems, the communication between the agents is usually not considered as a problem and is reduced to the goals distribution and observations report to the central agent. In decentralized systems, each agent uses a policy - often computed offline - to decide internally its next action. The most well-known framework for decentralized coordination is the Dec-POMDP framework, presented in section 3.3, in which the communication is usually considered free and full. Both cases include a major drawback: the reliability of the system depends on the communication to the central agent or on the full and free communication assumption. However, in real applications and specifically in crisis situations, relying on only one central agent is unsafe and the communication is neither full nor free. It is therefore important to develop decentralized techniques that are able to reduce their communication to send only the necessary information.

In this thesis, we consider a set of fully cooperative agents with heterogeneous sensors. All the agents in the system share the same goal, but they are able to collect different types of information.

Example 5.1 (Patrol in a working building) We consider three different types of robots: (1) robots equipped with a camera and a face detection module (2) robots equipped with a camera and a color detection module (3) robots equipped with a thermo-camera and temperature detection module. The robots of type 1 and 3 can detect the presence of humans in a room. The robots of type 2 and 3 can detect the presence of a fire.

All the robots share the same representation of the environment and their belief states cover the same variables X_k.

The Multi-Agent Planning for INformation Gathering (MAPING) framework we propose in this thesis is a framework for multi-agent event exploration with reduced communications that includes an offline part and an online part. It used an extended belief state, presented in section 5.2, that contains agent's beliefs about the environment but also about other agents' beliefs. The offline part is made up of the decision-theoretic model based on a Partially Observable Markov Decision Process (POMDP), presented in section 5.3. This POMDP can be solved using a technique that we present in section 5.4.

The online part of the framework includes the execution module, which executes actions and receives the observations, and a belief update module, which updates the agents' belief with the received observation. This belief update module also include a new step to implement a forgetting mechanism in the framework. Indeed, event exploration aims, by definition, at detecting dynamic events and is theoretically an open-ended problem. The more time passes after an agent discovered a feature's value, the more probable it is that this value has changed. Therefore the agents need to modify their belief state to take this raising uncertainty into account, and forget gradually what they have learned. The belief update module is presented in section 5.5.

5.2   Estimating what others know and need

Each agent has beliefs about the environment which are model, as in Chapter 4, as a set of probability distributions over the variables X_k. During their mission, they acquire knowledge and collect observations about these variables. To coordinate themselves and select the relevant piece of information they should communicate, they need to know about the belief state of the other agents. However it is impossible for an agent to know exactly at each decision step the belief state of other agents, as it is unreasonable to ask for this belief state wherever the agent needs it. As humans for instance, we don't know the exact beliefs of our fellow humans and we don't ask for their belief each time we want to tell them something. We decide internally if the information we want to communicate may be relevant for them according to what we believe they believe. This decision is sometimes accurate - if our beliefs about their beliefs are accurate - sometimes not - if our beliefs about their beliefs are outdated or wrong - but in any case it limits the amount of communication that could is necessary. We formalize this principle in the MAPING framework. To do so, we use an *extended belief state*, which includes not only the agent's own beliefs concerning the environment but also the agent's belief about other agent's beliefs. The formal description is given in definition 5.1.

Definition 5.1 (Extended belief state)**.** The *extended belief state* of the agent i at time t is defined as

$$\mathcal{B}_{i,t} = \langle \mathcal{B}_{i,t}^{\mathcal{E}}, \mathcal{B}_{i,t}^{j,\mathcal{E}} \rangle \tag{5.1}$$

with $\mathcal{B}_{i,t}^{\mathcal{E}} = \langle b_{i,t}^{i,k} \rangle_{\forall X_k \in \mathcal{E}}$ being the beliefs of agent i concerning the environment $\mathcal{E}$ and $\mathcal{B}_{i,t}^{j,\mathcal{E}} = \langle b_{i,t}^{j,k} \rangle_{\forall X_k \in \mathcal{E}}$ being the beliefs of agent i concerning the environment beliefs of agent j.

Let us note that $\mathcal{B}_{i,t}^{j,\mathcal{E}}$ is an approximation of $\mathcal{B}_{j,t}^{\mathcal{E}}$. We can also use a matrix representation, in which rows represent the different random variables describing the environment and columns represent agent i's beliefs on each agent's beliefs, including itself :

$$\mathcal{B}_{i,t} = \begin{pmatrix} b_{i,t}^{1,1} & \cdots & b_{i,t}^{j,1} & \cdots & b_{i,t}^{i,1} & \cdots & b_{i,t}^{|\mathcal{AG}|,1} \\ \vdots & & \vdots & & \vdots & & \vdots \\ b_{i,t}^{1,k} & \cdots & b_{i,t}^{j,k} & \cdots & b_{i,t}^{i,k} & \cdots & b_{i,t}^{|\mathcal{AG}|,k} \\ \vdots & & \vdots & & \vdots & & \vdots \\ b_{i,t}^{1,|\mathcal{E}|} & \cdots & b_{i,t}^{j,|\mathcal{E}|} & \cdots & b_{i,t}^{i,|\mathcal{E}|} & \cdots & b_{i,t}^{|\mathcal{AG}|,|\mathcal{E}|} \end{pmatrix} \tag{5.2}$$

In this extended belief state, each agent has approximate beliefs about the beliefs of other agents and can decide if a piece of information is relevant for another agent. It is clear that each agent needs to update its extended belief state to keep not only an accurate belief on the variables but also an accurate belief on the other agents' beliefs. The way this belief state is updated is described in section 5.3.3.

5.3   Observing the system and assessing the mission

The core of the offline decision module is the Partially Observable Markov Decision Process (POMDP) used to compute each agent's policy. A POMDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \Omega, \mathcal{R}, b_0 \rangle$ where

- $\mathcal{S}$ is the set of states

- $\mathcal{A}$ is the set of epistemic actions

- $\mathcal{O}$ is the set of observations

- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the transition function

- $\Omega : \mathcal{O} \times \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the observation function

- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function

- b_0 is the initial belief state

In this section we describe the elements making up the POMDP and specific to MAPING .

5.3.1 What can be seen and done: states, observations and actions

The set of states consists of all the joint instantiations of the variables $X_k \in \mathcal{E}$. The size of the set of states is

$$|\mathcal{S}| = \prod_{X_k \in \mathcal{E}} n_k$$

n_k being the number of possible values for variable X_k.

As in chapter 4, we consider that observations are an atomic piece of information an agent may receive after it performed an action, that is to say a feedback of the environment. Those observations give an indication about the current state of the system and are used by the agents to update their belief states. The set of observations fully depends on the application considered. The most general case involves that an observation can be related to several variables, as the observation **4 rooms are empty** in example 4.3. However, in our case we consider that the robots explore the different areas in order to collect observation about the area it explored. Therefore, each observation is related to one area, and so one variable.

We consider in MAPING only epistemic actions, which means actions that don't imply a change in the environment and don't modify the current state of the POMDP but only the agent's internal belief state. We define two types of actions : look for the value of a particular random variable (*Explore*-type actions) and communicate a set of observations to a set of agents (*Communicate*-type actions). To these types, we add another action named *Idle*, which corresponds to doing nothing. The set of actions is so:

$$\mathcal{A} = \mathcal{A}_{Explore} \cup \mathcal{A}_{Communicate} \cup \{Idle\}$$
$$\mathcal{A} = \{Explore(X_k), \forall X_k \in \mathcal{E}\} \cup \{Communicate(O, Ag), \forall O \subset \mathcal{O}, \forall Ag \subset \mathcal{AG}\} \cup \{Idle\}$$

The size of the action set is :

$$|\mathcal{A}| = |\mathcal{A}_{Explore}| + |\mathcal{A}_{Communicate}| + 1$$

There is as many *Explore*-type actions as the number of variables, so $|\mathcal{A}_{Explore}| = |\mathcal{E}|$. Each agent can communicate any subset of observations except the empty set, that is to say $2^{|\mathcal{O}|} - 1$ possible subsets. The agent can communicate to any subset of agents except subsets that includes itself, that is to say $(2^{|\mathcal{AG}|-1})$ possible subsets. So the size of the action set is:

$$|\mathcal{A}| = |\mathcal{E}| + (2^{|\mathcal{O}|} - 1) \times (2^{|\mathcal{AG}|-1}) + 1 \tag{5.3}$$

It is obvious that the size of the set of states and the size of set of actions increase exponentially with the number of variables and create a combinatorial explosion even with small systems, and makes the policy impossible to compute. It is nevertheless possible to reduce this size with two reasonable hypothesis. Let us consider first the set of states. The most general case requires to consider possible that the variables may have different and possibly high numbers of possible values. However, in event exploration we are interested in knowing if an event happened or not. In a lot of cases it is then possible to assume that each variable represents a event that can occur and has only two possible values: *true* or *false*, if the event occurred or not. With this assumption, the size of the set of states is reduced to $2^{|\mathcal{E}|}$.

Second we may assume that, instead of communicating any subset of observations to any subset of agents, each agent can communicate one observation to one agent. In that case, the agent can communicate only $|\mathcal{O}|$ observations to $|\mathcal{AG}|-1$ agents and the size of the set of actions becomes $|\mathcal{A}| = |\mathcal{E}| + |\mathcal{O}| \times (|\mathcal{AG}|-1) + 1$.

In the remainder of this thesis we will consider both assumptions. However let us note that the model is still valid if the hypotheses need to be relaxed.

Example 5.2 (Patrol in a working building) Let us consider an instantiation of the the patrolling example with 5 rooms and 3 agents, one of each type. There are 10 variables in $\mathcal{E}$ corresponding to the state of the rooms:

$$\{RoomEmpty_{R_k}, RoomOnFire_{R_k} | \forall R_k \in ListOfRooms\}$$

Each variable has two possible values: *true* or *false*. The possible observations describe what the robots may receive, depending on their sensors. The robots equipped with a face detection module can receive the observations

$$\{HumanInRoom_{R_k}, NoHumanInRoomK_{R_k} | \forall R_k \in ListOfRooms\}$$

The robots equipped with a color detection module can receive the observations

$$\{FireColorInRoom_{R_k}, NoFireColorInRoom_{R_k} | \forall R_k \in ListOfRooms\}$$

And finally the robots with a thermocamera can receive the observations

$$\{HotSpotInRoom_{R_k}, NoHotSpotInRoom_{R_k} | \forall R_k \in ListOfRooms\}$$

We consider that the agents never receive an observation after sending a message.

The corresponding POMDP has 1024 states, 30 possible observations and 101 possible actions if we consider the simplifying assumption, 4294967303 otherwise.

5.3.2 The dynamic of the system: transition and observation functions

The transition function describes the dynamic of the system. The transition $T(s', \alpha, s)$ is the probability that the system makes a transition from the state s to the state s' when action α is performed: $T(s, \alpha, s') = P(s'|\alpha, s)$. We only consider epistemic actions, that is to say actions that only affect the agents' belief states and not the environment. Therefore the evolution of the environment is completely independent from the agents, which can be modeled by: $T(s, \alpha, s') = T(s, s') = P(s'|s)$.

The observation function describes the reliability of the sensors. It is the probability to

receive an observation o after action α has been performed in state s: $\Omega(o, s, \alpha) = P(o|s, \alpha)$. In this case, the observation function remains action-dependent since the actions considered aim at gathering information. When an agent executes an *Explore*-type action, it explores the value of a variable and receives an observation corresponding to this variable. When the agent executes a *Communicate*-type action, it may receive an observation concerning the status of the sent message - if it has been sent and/or received properly - or no observation at all. When the agent performs *Idle*, it doesn't receive any observation.

Example 5.3 (Patrol in a working building) In the patrolling example there are two types of events: the presence of a human in the room and the presence of a start of fire in a room. The transition function should so emphasize these two types. We consider that the presence of a human and the presence of a start of fire in a room are completely independent. We also consider that the presence of a human in room is independent of the presence of a human in other rooms. However, we can consider that the probability to find a start of fire in a room is 0.9 if there is a start of fire in the room next to it, and 0.1 otherwise.

The observation function should reflect the capacities of the robot and the precision of its sensors. The probability to receive each observation defined in example 5.2 strictly depends on the room explored by the robot and the sensors it possesses. Robot equipped with a face detection module will have an observation function shaped as follows:

$$\Omega(HumanInRoom_{R_k}, RoomEmpty_{R_k} = true, Explore(RoomEmpty_{R_k})) = 0.1$$
$$\Omega(HumanInRoom_{R_k}, RoomEmpty_{R_k} = false, Explore(RoomEmpty_{R_k}) = 0.9$$
$$\Omega(NoHumanInRoom_{R_k}, RoomEmpty_{R_k} = true, Explore(RoomEmpty_{R_k}) = 0.9$$
$$\Omega(NoHumanInRoom_{R_k}, RoomEmpty_{R_k} = false, Explore(RoomEmpty_{R_k})) = 0.1$$

All the other combinations of observation - state - action have a probability equals to 0. Indeed, the robot will never be able to receive an observation $FireInRoom_{X_k}$ since it's not equipped with the right sensors, and will never receive an observation $HumanInRoom_{X_k}$ if it is not performing an action $Explore(X_k)$. Since the presence of a start of fire in the room does not influence the probability to receive the observation $HumanInRoom_{R_k}$, the state of the variable $RoomOnFire_{R_k}$ does not appear in the observation function. Similarly, robot equipped with a color detection module will have an observation function shaped as follows:

$$\Omega(FireColorInRoom_{R_k}, RoomOnFire_{R_k} = true, Explore(RoomOnFire_{R_k})) = 0.9$$
$$\Omega(FireColorInRoom_{R_k}, RoomOnFire R_k = false, Explore(RoomOnFire_{R_k})) = 0.1$$
$$\Omega(NoFireColorInRoom_{R_k}, RoomOnFire_{R_k} = true, Explore(RoomOnFire_{R_k})) = 0.1$$
$$\Omega(NoFireColorInRoom_{R_k}, RoomOnFire_{R_k} = false, Explore(RoomOnFire_{R_k})) = 0.9$$

For robot equipped with a thermocamera the observation function is a bit more complicated since both events can trigger the observation $HotSpotInRoom_{R_k}$. Therefore both variables

---

need to be taken into account in the observation function.

$$\Omega(HotSpotInRoom_{R_k}, RoomEmpty = true \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomEmpty_{R_k})) = 0.8$$

$$\Omega(HotSpotInRoom_{R_k}, RoomEmpty = true \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomOnFire_{R_k})) = 0.8$$

$$\Omega(HotSpotInRoom_{R_k}, RoomEmpty = true \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomEmpty_{R_k})) = 0.05$$

$$\Omega(HotSpotInRoom_{R_k}, RoomEmpty = true \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomOnFire_{R_k})) = 0.05$$

$$\Omega(HotSpotInRoom_{R_k}, RoomEmpty = false \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomEmpty_{R_k})) = 0.95$$

$$\Omega(HotSpotInRoom_{R_k}, RoomEmpty = false \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomOnFire_{R_k})) = 0.95$$

$$\Omega(HotSpotInRoom_{R_k}, RoomEmpty = false \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomEmpty_{R_k})) = 0.7$$

$$\Omega(HotSpotInRoom_{R_k}, RoomEmpty = false \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomOnFire_{R_k})) = 0.7$$

$$\Omega(NoHotSpotInRoom_{R_k}, RoomEmpty = true \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomEmpty_{R_k})) = 0.2$$

$$\Omega(NoHotSpotInRoom_{R_k}, RoomEmpty = true \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomOnFire_{R_k})) = 0.2$$

$$\Omega(NoHotSpotInRoom_{R_k}, RoomEmpty = true \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomEmpty_{R_k})) = 0.95$$

$$\Omega(NoHotSpotInRoom_{R_k}, RoomEmpty = true \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomOnFire_{R_k})) = 0.95$$

$$\Omega(NoHotSpotInRoom_{R_k}, RoomEmpty = false \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomEmpty_{R_k})) = 0.05$$

$$\Omega(NoHotSpotInRoom_{R_k}, RoomEmpty = false \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomOnFire_{R_k})) = 0.05$$

$$\Omega(NoHotSpotInRoom_{R_k}, RoomEmpty = false \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomEmpty_{R_k})) = 0.3$$

$$\Omega(NoHotSpotInRoom_{R_k}, RoomEmpty = false \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomOnFire_{R_k})) = 0.3$$

5.3.3 Stay informed: maintaining a belief state

To keep an accurate representation of the current state of the system an agent has to update its beliefs regularly. An update will occur in three cases :

1. the agent receives a new observation from its sensors after an *Explore*-type action. It

updates its own beliefs concerning the environment : $\mathcal{B}^{\mathcal{E}}_{i,t+1}$, or in the matrix representation:

$$\mathcal{B}_{i,t+1} = \begin{pmatrix} b^{1,1}_{i,t} & \cdots & b^{j,1}_{i,t} & \cdots & \mathbf{b^{i,1}_{i,t+1}} & \cdots & b^{|\mathcal{AG}|,1}_{i,t} \\ \vdots & & \vdots & & \vdots & & \vdots \\ b^{1,k}_{i,t} & \cdots & b^{j,k}_{i,t} & \cdots & \mathbf{b^{i,k}_{i,t+1}} & \cdots & b^{|\mathcal{AG}|,k}_{i,t} \\ \vdots & & \vdots & & \vdots & & \vdots \\ b^{1,|\mathcal{E}|}_{i,t} & \cdots & b^{j,|\mathcal{E}|}_{i,t} & \cdots & \mathbf{b^{i,|\mathcal{E}|}_{i,t+1}} & \cdots & b^{|\mathcal{AG}|,|\mathcal{E}|}_{i,t} \end{pmatrix}$$

2. the agent receives a new observation from agent j. It updates its own beliefs $\mathcal{B}^{\mathcal{E}}_{i,t+1}$. It also consider that agent j updated its own beliefs with this observation before sending it, so agent i also updates its beliefs concerning agent j to reflect agent j's latest belief state: $\mathcal{B}^{j,\mathcal{E}}_{i,t+1}$. In the matrix representation, it updates:

$$\mathcal{B}_{i,t+1} = \begin{pmatrix} b^{1,1}_{i,t} & \cdots & \mathbf{b^{j,1}_{i,t+1}} & \cdots & \mathbf{b^{i,1}_{i,t+1}} & \cdots & b^{|\mathcal{AG}|,1}_{i,t} \\ \vdots & & \vdots & & \vdots & & \vdots \\ b^{1,k}_{i,t} & \cdots & \mathbf{b^{j,k}_{i,t+1}} & \cdots & \mathbf{b^{i,k}_{i,t+1}} & \cdots & b^{|\mathcal{AG}|,k}_{i,t} \\ \vdots & & \vdots & & \vdots & & \vdots \\ b^{1,|\mathcal{E}|}_{i,t} & \cdots & \mathbf{b^{j,|\mathcal{E}|}_{i,t+1}} & \cdots & \mathbf{b^{i,|\mathcal{E}|}_{i,t+1}} & \cdots & b^{|\mathcal{AG}|,|\mathcal{E}|}_{i,t} \end{pmatrix}$$

3. the agent sends an observation to agent j with a *Communicate*-type action. We consider that agent i already updated its own beliefs with the observation when it has received it. Since agent j has the same update policy, it will update its own beliefs when it will receive the observation, so agent i only updates its beliefs concerning agent j : $\mathcal{B}^{j,\mathcal{E}}_{i,t+1}$, which updates the matrix representation as follows:

$$\mathcal{B}_{i,t+1} = \begin{pmatrix} b^{1,1}_{i,t} & \cdots & \mathbf{b^{j,1}_{i,t+1}} & \cdots & b^{i,1}_{i,t} & \cdots & b^{|\mathcal{AG}|,1}_{i,t} \\ \vdots & & \vdots & & \vdots & & \vdots \\ b^{1,k}_{i,t} & \cdots & \mathbf{b^{j,k}_{i,t+1}} & \cdots & b^{i,k}_{i,t} & \cdots & b^{|\mathcal{AG}|,k}_{i,t} \\ \vdots & & \vdots & & \vdots & & \vdots \\ b^{1,|\mathcal{E}|}_{i,t} & \cdots & \mathbf{b^{j,|\mathcal{E}|}_{i,t+1}} & \cdots & b^{i,|\mathcal{E}|}_{i,t} & \cdots & b^{|\mathcal{AG}|,|\mathcal{E}|}_{i,t} \end{pmatrix}$$

In all cases, $b^{j,k}_{i,t+1} = update(b^{j,k}_{i,t}, o)$ is the update of the previous probability distribution with the observation o and $\mathcal{B}^{j,\mathcal{E}}_{i,t+1} = update(\mathcal{B}^{j,\mathcal{E}}_{i,t}) = \{b^{j,k}_{i,t+1}, \forall X_k \in \mathcal{E}\}$

For exploring and sending an observation, the update is made as usual in Partially Observable Markov Decision Processes :

$$b^{j,k}_{i,t+1}(s') = \frac{\Omega(o, s', \alpha) \sum_{s \in \mathcal{S}} p(s'|s, \alpha) \mathcal{B}_{i,t}(s)}{\sum_{s \in \mathcal{S}} \sum_{s'' \in \mathcal{S}} \Omega(o, s'', \alpha) p(s''|s, \alpha) b^{j,k}_{i,t}} \tag{5.4}$$

where α is the action, $\mathcal{B}_{i,t}(s)$ is the probability that the state s is true, and Ω is the current agent's observation function.

The case of the agent receiving an observation is a bit more complex. In the first steps of our models, all the agents had the same sensors and so the same observation function. Therefore we

processed the Bayesian update by using this observation function when receiving an observation. However with heterogeneous robots, each type of robot has its own observation function and the one used in the update should be the one of the agent sending the observation. Practically, we solved it during the system initialization. When the system is initialized, all the agents send their observation function to all other agents. Then each agent store in its internal memory the observation function of all the other agents. The case of heterogeneous agents will be discussed more in details in section 5.6.

5.3.4 What is the goal ? The reward function

The reward function usually defines the reward an agent may receive by performing action α in state s. However in event exploration we are not interested in bringing the system to a specific state but in improving the accuracy of the agents' belief states. Agents should be rewarded when they improve their belief state and when they help others to improve theirs as well, and so when they collect and communicate relevant observations for them and the others. The reward function of MAPING is not defined on the states as usually but on the belief states of the agent: $\mathcal{R} : \mathcal{B} \times \mathcal{A} \to \mathbb{R}$. The reward function is split into three parts: $\mathcal{R}(\mathcal{B}_{i,t}, Idle)$, $\mathcal{R}(\mathcal{B}_{i,t}, Explore(X_k))$ and $\mathcal{R}(\mathcal{B}_{i,t}, Communicate(o, j))$. The case $\mathcal{R}(\mathcal{B}_{i,t}, Idle)$ is the easiest, since the agent does nothing. Therefore it receives no reward: $\mathcal{R}(\mathcal{B}_{i,t}, Idle) = 0$.

As stated in Chapter 4, a relevant observation is correct. However the agents can observe the state of the world only partially and have only beliefs about it. They cannot state with certainty if an observation is true or not. They should rely on their beliefs to decide if a given observation is *likely* to be true or if it is *likely* to be false.

Definition 5.2 (Degree of belief)**.** The degree of belief an agent i has concerning an observation o at a time t is given by:

$$bel_{i,t}(o) = \sum_{s \in \mathcal{S}} \sum_{\alpha \in \mathcal{A}_{Explore}} \mathcal{B}_{i,t}(s) \Omega(o, s, \alpha)$$

where $\mathcal{B}_{i,t}(s)$ is the probability that s is the current state.

A special case of this degree of belief is the *action dependent degree of belief*.

Definition 5.3 (Action dependent degree of belief)**.** The action dependent degree of belief an agent i has concerning an observation o when performing an action α is given by:

$$bel_{i,\alpha,t}(o) = \sum_{s \in \mathcal{S}} \mathcal{B}_{i,t}(s) \Omega(o, s, \alpha)$$

where $\mathcal{B}_{i,t}(s)$ is the probability that s is the current state.

The degree of beliefs represents the probability an agent would have to receive the observation if it explored all the variables in a state s, considering the belief of the agent that s is the current state. The actions-dependent degree of belief limits the degree of belief to a single action.

When performing an Explore-type action, the agent is rewarded if it explores a variable that can provide it with relevant observations. Therefore the agent chooses among all observations the one which would be the most relevant and the variable it needs to explore to get this observation.

However exploring has a cost which must be taken into account. This cost may represent battery loss, distance to travel, etc. The reward function for an *Explore*-type action is:

$$\mathcal{R}(\mathcal{B}_{i,t}, Explore(X_k)) = \sum_{o \in \mathcal{O}} bel_{i,\alpha,t}(o)rel_i(o) - C_{Explore(X_k)} \qquad (5.5)$$

where $C_{Explore(X_k)}$ is the cost associated to the exploration.

When performing a Communicate-type action, the agent is rewarded if it communicates an observation which is relevant for the agent to which it communicates. In addition to this, the agents should try to get homogeneous belief states, which means belief states close to each others. This is also due to the fact that the agents only have beliefs about the environment and, by definition, cannot state if those beliefs are correct. Once again, this is inspired by our human way of thinking. If all of my colleagues and I think that the meeting is at 2:00pm, this belief is likely to be true. If I am the only one thinking it is at 4:00pm, I am likely to be wrong and I should check again. If after checking I am still convinced that the meeting is at 4:00pm, I should tell my colleagues so that they can check in turn, in case a change occurred in the meeting time. If we all have different times in mind for the meeting, we should all check to get the right time. We assume that agents have the same behavior in the system: if all of them have close beliefs, those are likely to be close to the truth. However if there is one agent which think differently or if all agents have different beliefs, it means that at least one agent is wrong. This fault may result to a sensor failure or a change in the environment. For this reason, we include in the reward function for *Communicate*-type action a term to encourage the agents to communicate relevant observation that reduces the difference between their beliefs and the approximation they have about other agents' beliefs. The reward function for a *Communicate*-type action is thus defined as follows:

$$\mathcal{R}(\mathcal{B}_{i,t}, Communicate(o,j)) = bel_{i,t}(o)rel_i(o) + (D_H(\mathcal{B}_{i,t}^{\mathcal{E}}||\mathcal{B}_{i,t}^{j,\mathcal{E}}) - D_H(\mathcal{B}_{i,t+1}^{\mathcal{E}}||\mathcal{B}_{i,t+1}^{j,\mathcal{E}}))$$
$$- C_{Communicate(o,j)} \qquad (5.6)$$

where $C_{Communicate(o,j)}$ is the cost of communicating observation o to agent j and $\mathcal{B}_{i,t+1}^{j,\mathcal{E}} = update(\mathcal{B}_{i,t}^{j,\mathcal{E}}, o)$ is the belief state of agent i updated with the observation o.

For the sake of simplicity, we considered in the first part of our work that $C_{Explore(X_k)}$ and $C_{Communicate(o,j)}$ were constant. However in real applications the cost of exploring varies with the position of the robot and the cost of communicating varies with the size of the message to send. Considering varying costs when computing the policy implies that the model includes the parameters that makes the costs vary such as the position of the robots, causing the model grows dramatically. In section 5.4 we present a technique to compute a satisfactory policy taking into account the variation of the costs.

The reward function being defined on the belief states instead of the system state, it is impossible to use literature techniques mentioned in section 3.1.4. Moreover the degree of relevance being not convex, the reward function is not convex either, which makes impossible the use of the ρPOMDP suggested by Araya-Lopez et al. [Araya-Lopez et al., 2010]. The next section presents a technique we used to solve the MAPING POMDP.

5.4 Planning algorithm of MAPING

5.4.1 General case

As explained in the previous section, the MAPING model cannot be solved by using classical literature techniques. However, the fact that we consider only epistemic actions makes it possible to solve it by transforming it into a Belief-MDP and rely on classic MDP algorithms. A Belief-MDP is defined as a tuple $< \Delta, \mathcal{A}, \tau >$ where :

- Δ is the set of states

- $\mathcal{A}$ is the set of actions

- $\tau : \Delta \times \mathcal{A} \rightarrow \mathbb{R}$ is the transition function

The state space of the Belief-MDP corresponds directly to the belief states space in the MAPING POMDP: $\Delta = \mathcal{B}_{i,t}$. Considering only epistemic actions, the set of actions in MAPING only affects the Belief states. Therefore, the set of actions in the Belief-MDP is the same as the set of actions in the MAPING POMDP. The transition function τ can be obtained from the transition and observation T and Ω functions of the MAPING POMDP and is defined as follows:

$$\tau(\mathcal{B}_{i,t}, \alpha, \mathcal{B}_{i,t+1}) = \begin{cases} \sum\limits_{s \in \mathcal{S}} \sum\limits_{o \in U_t} \Omega(o, s, \alpha) \mathcal{B}_{i,t}(s) & \text{if } U_t \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

where $U_t = \{o \in \mathcal{O}, \text{ such as, } \mathcal{B}_{i,t+1} = update(\mathcal{B}_{i,t}, o)\}$ is the set of all observations enabling the transition from state $\mathcal{B}_{i,t}$ to state $\mathcal{B}_{i,t+1}$ and $\mathcal{B}_{i,t}(s)$ is the belief of agent i that the current state is s.

Proof. If there is no observation such as $\mathcal{B}_{i,t+1} = update(\mathcal{B}_{i,t}, o)\}$ - that is to say $U_t = \emptyset$ - it is not possible to transfer from belief state $\mathcal{B}_{i,t}$ to belief state $\mathcal{B}_{i,t+1}$. Therefore, $\tau(\mathcal{B}_{i,t}, a, \mathcal{B}_{i,t+1}) = 0$. If there exists at least one observation o such as $\mathcal{B}_{i,t+1} = update(\mathcal{B}_{i,t}, o)\}$ - that is to say $U_t \neq \emptyset$ - we have the following equations :

$$\begin{aligned} \tau(\mathcal{B}_{i,t}, \alpha, \mathcal{B}_{i,t+1}) &= P(\mathcal{B}_{i,t+1} | \mathcal{B}_{i,t}, \alpha) \\ &= \sum_{o \in U_t} P(o | \mathcal{B}_{i,t}, \alpha) \\ &= \sum_{s \in \mathcal{S}} \sum_{o \in U_t} P(o | s, \alpha) \mathcal{B}_{i,t}(s) \\ &= \sum_{s \in \mathcal{S}} \sum_{o \in U_t} \Omega(o, s, \alpha) \mathcal{B}_{i,t}(s) \end{aligned}$$

$\square$

The value function corresponding to this Belief MDP is defined as follows:

$$V(\mathcal{B}_{i,t}) = \mathcal{R}(\mathcal{B}_{i,t}) + \max_{\alpha \in \mathcal{A}} \int_{\mathcal{B}_{i,t+1}} \tau(\mathcal{B}_{i,t}, \alpha, \mathcal{B}_{i,t+1}) V(\mathcal{B}_{i,t+1}) \tag{5.7}$$

	5 elements		11 elements	
	2 agents	3 agents	2 agents	3 agents
2 variables	625	7776	14641	1771561
3 variables	7776	1953125	1771561	2.36×10^9
4 variables	390625	2.44×10^8	2.14×10^7	3.14×10^{12}
5 variables	9765625	3.05×10^{10}	2.59×10^{10}	4.18×10^{15}
6 variables	2.44×10^8	3.81×10^{12}	3.14×10^{12}	5.56×10^{18}

Table 5.1: Number of states with 2 possible values for each variable

5.4.2 Discretized solving

Equation 5.7 corresponds to continuous model. Using discretization techniques[18] we may transform equation 5.7 in :

$$V(\mathcal{B}_{i,t}) = \mathcal{R}(\mathcal{B}_{i,t}) + \max_{\alpha \in \mathcal{A}} \sum_{\mathcal{B}_{i,t+1} \in Samples} \tau(\mathcal{B}_{i,t}, \alpha, \mathcal{B}_{i,t+1}) V(\mathcal{B}_{i,t+1}) \qquad (5.8)$$

where $Samples$ is the set of belief states in the discretized space. Then, any technique from the literature may be used to solve the Belief-MDP.

This technique faces a huge computational problem since the size of the belief states space grows exponentially in the number of variables and agents, which leads to major difficulties for solving the model for real cases. Tables 1 presents the number of states obtained for different numbers of variables - all with two possible values - and different numbers of agents, using two different sizes of discretization. We considered a naive discretization in which the probability distributions that make up the belief states are set finite, and the set of all probability distributions can have 5 or 11 elements. The set with 5 elements is: $((0,1) ; (0.2, 0.8) ; (0.5, 0.5) ; (0.8, 0.2) ; (1,0))$. The set with 11 elements is: $((0,1) ; (0.1, 0.9) ; (0.2, 0.8) ; (0.3, 0.7) ; (0.4, 0.6) ; (0.5, 0.5) ; (0.6, 0.4) ; (0.7, 0.3) ; (0.8, 0.2) ; (1,0))$. It is obvious that even the best algorithms cannot solve such large MDPs. That's why we need to find a way to reduce the number of states.

To do so, we considered two independence assumptions:

- Variable independence : the value of a given random variable does not depend on all the other variables in the system but only on a subset of variables.

- Observation independence : the probability of receiving a given observation when performing a given action depends only on the values of a given subset of random variables.

Example 5.4 (Patrol in a working building) In example 5.3, we stated that the probability that a human is present in the room is independent of the probability that a human is present in another room. All the variables $RoomEmpty_{R_k}$ are independent from each others. We could make extend the assumption to the variables $RoomOnFire_{R_k}$. Both assumptions are simplification of the reality, but the loss involved by this simplification is balanced by the possibility to compute a satisfactory policy. This is the variable independence assumption.

We also stated in example 5.3 that the probability for an agent to receive an observation about a variable $HumanInRoom_{R_k}$, $FireColorInRoom_{R_k}$ or $HotSpotInRoom_{R_k}$ only de-

---

[18]For instance by discretizing the probability distributions

pends on the value of variables about R_k - that is to say $RoomEmpty_{R_k}$ and $RoomOnFire_{R_k}$ - and not on the other variables. For instance the probability of receiving the observation $HumanInRoom_{R_1}$ does not depend on the value of $RoomEmpty_{R_3}$. This is the observation independence assumption.

Using those two assumptions, we suggest to split the initial very large POMDP into a set of smaller independent POMDPs that can be solved in a reasonable time. To explain how to split the POMDP, we first need to define the independence of sets.

Definition 5.4 (Independence of sets)**.** Two sets A and B are called independent if and only if :
$$\forall a \in A, \forall b \in B, P(a, b) = P(a) \times P(b)$$

Then, we denote as usual $\mathcal{E}$ the set of all the random variables describing the environment and $\mathcal{O}$ the set of all the observations. We assume a partition of the set $\mathcal{E}$, written $\mathcal{P}(\mathcal{E})$, and a partition of the set $\mathcal{O}$, written $\mathcal{P}(\mathcal{O})$ which respect the variable independence and the observation independence assumption:

1. (variable Independence) $\forall X, Y \in \mathcal{P}(\mathcal{E}), X \neq Y$, X and Y are independent.

2. (observation Independence) $\forall O \in \mathcal{P}(\mathcal{O})$, there is a unique set $X \in \mathcal{P}(\mathcal{E})$ so that $\forall o \in O, P(o|s, a) = P(o|X, a)$.

Such a decomposition is guaranteed to exist, take $\mathcal{P}(\mathcal{E}) = X$ and $\mathcal{P}(\mathcal{O}) = O$, and should be as fine as possible for the decomposition to be efficient. The variable independence enables us to rewrite the transition function:
$$T(s, \alpha, s') = \prod_{X \in \mathcal{P}(\mathcal{E})} T(X, \alpha, X')$$

The observation independence enables us to rewrite the observation function:
$$\Omega(o, s, \alpha) = P(o|s, \alpha) = P(o|X^o, \alpha)$$

with X^o being the set of variables that influences the probability of receiving observation o.

Using these partitions, we can build a set of $|\mathcal{P}(\mathcal{E})|$ sub-POMDPs, each sub-POMDP being a tuple $\langle \mathcal{S}_\ell, \mathcal{A}, \mathcal{O}_\ell, \mathcal{T}_\ell, \omega_\ell, \mathcal{R}_\ell, b_{\ell,0} \rangle$ where

- $\mathcal{S}_\ell$ is all the possible joint instantiations of the variables $X_\ell \in \mathcal{P}(\mathcal{E})$

- $\mathcal{A}$ is the same set of actions as in the global POMDP

- $\mathcal{O}_\ell \in \mathcal{P}(\mathcal{O})$ being the set of observations depending on the set X_ℓ, as defined previously

- $\mathcal{T}_\ell$ is the transition function applied to variables X_ℓ

- ω_ℓ is the observation function applied to variables of X_ℓ

- $\mathcal{R}_\ell$ is the reward function applied to variables of X_ℓ

- $b_{\ell,0}$ is the initial belief state

Each belief state used in the sub-POMDPs, written $\mathcal{B}_{\ell,i,t}$, is one part of the belief state of the global POMDP : $\mathcal{B}_{i,t} = \langle \mathcal{B}_{\ell,i,t} \rangle_{\forall \ell}$.

Example 5.5 (Patrolling in a working building) The matrix notation of one of the agent's belief state is:

$$\mathcal{B}_{1,t} = \begin{pmatrix} b_{1,t}^{1,1} & b_{1,t}^{2,1} & b_{1,t}^{3,1} \\ \vdots & \vdots & \vdots \\ b_{1,t}^{1,k} & b_{1,t}^{2,k} & b_{1,t}^{3,k} \\ \vdots & \vdots & \vdots \\ b_{1,t}^{1,10} & b_{1,t}^{2,10} & b_{1,t}^{3,10} \end{pmatrix}$$

As stated in Example 5.3, all the variables are independent from each others. Therefore we can build 10 sub-POMDPs, each corresponding to one variable. The belief state of the agent in the sub-POMDP ℓ will be a single row on the variable ℓ:

$$\mathcal{B}_{1,t} = \begin{pmatrix} b_{1,t}^{1,\ell} & b_{1,t}^{2,\ell} & b_{1,t}^{3,\ell} \end{pmatrix}$$

Then each of these sub-POMDPs can be solved independently by transforming it into a Belief-MDP as explained previously. Techniques using state-space discretization can be applied since the state space is less complex and so the number of cells is tractable. While solving the belief-MDPs, we store the optimal value function $V^*(\mathcal{B}_{\ell,i,t})$ for each sub-POMDP in addition to the optimal policy π_ℓ^*. The agent is provided with the sub-policies π_ℓ^* instead of the global one and with the associated value functions V_{π_ℓ}. It can retrieve the action to perform during execution by following algorithm 5.1.In this algorithm the agent compares the expected value of each locally optimal action π_ℓ for each sub-belief state making up the current belief state and select the action that gives with the best expecetd value. This comparison is possible since the rewards of the sub-POMDP are commensurable, which implies that the computed values are also commensurable.

Data: sub-policies π_ℓ and associated value functions V_{π_ℓ}, current belief state $\mathcal{B}_{i,t}$
Result: action to execute α_{opt}
1 $V_{max} = -Infinity$;
2 $\alpha_{opt} = null$;
3 **foreach** $\mathcal{B}_{\ell,i,t}$ *composing* $\mathcal{B}_{i,t}$ **do**
4   **if** $V_{\pi_\ell}(\mathcal{B}_{\ell,i,t}) \geq V_{max}$ **then**
5         $V_{max} = V_{\pi_j}(\mathcal{B}_{\ell,i,t})$;
6         $\alpha_{opt} = \pi_\ell(\mathcal{B}_{\ell,i,t})$;
7   **end**
8 **end**
9 return α_{opt} ;

Algorithm 5.1: Getting the action to execute from local optimal policies

---

This algorithm presents two drawbacks, which makes the action computed not optimal and hard to use in real-type applications: 1. the sub-policies π_ℓ are optimal and computed with infinite horizon, but the global action is taken considering only the next step 2. the cost of the actions is still considered constant and contained in the sub-POMDP's reward functions. To improve the first point slightly, we considered that the agent should take into account the fact that choosing one locally optimal actions at a time t - and so one subset of variables - means

not doing anything on the other subsets of variables, that is to say performing the action *Idle* on the other subsets. To deal with the second drawback, we suggest to remove the cost from the reward function and consider it while choosing the global action. This makes it possible to use costs that depends on physical parameters, such as the position of the robot, without including those parameters in the POMDP.

The updated algorithm is presented in algorithm 5.2. The agent compares the expected value of the locally optimal action π_ℓ corresponding to the current subset of variables plus the expected value of the *Idle* action for all other subsets of variables. This modification does still not provide

---

Data: sub-policies π_ℓ and associated value functions V_{π_ℓ}, current belief state $\mathcal{B}_{i,t}$
Result: action to execute α_{opt}
1 $V_{max} = -Infinity$;
2 $\alpha_{opt} = null$;
3 $expectedValue = 0$;
4 **foreach** $\mathcal{B}_{\ell,i,t}$ *composing* $\mathcal{B}_{i,t}$ **do**
5 $expectedValue = V_{\pi_\ell}(\mathcal{B}_{\ell,i,t})$;
6   **foreach** $\mathcal{B}_{\ell',i,t}$, $\ell' \neq \ell$ **do**
7 $expectedValue = expectedValue + V_{Idle}(\mathcal{B}_{\ell',i,t})$;
8   **end**
9   Compute the cost of performing action $\pi_\ell(\mathcal{B}_{\ell,i,t})$, noted $C_{\pi_\ell(\mathcal{B}_{\ell,i,t})}$;
10 $expectedValue = expectedValue - C_{\pi_\ell(\mathcal{B}_{\ell,i,t})}$;
11   **if** $expectedValue > V_{max}$ **then**
12 $V_{max} = expectedValue$;
13 $\alpha_{opt} = \pi_\ell(\mathcal{B}_{\ell,i,t})$;
14   **end**
15 **end**
16 return α_{opt} ;

Algorithm 5.2: Getting the action to execute from local optimal policies – version 1

---

optimal actions but enables to plan one step ahead. When several actions have the same value, algorithm 5.2 will always return the first one. This could affect the behavior of the system. It would so be preferable to chose randomly an action among those with the highest value. To do so, algorithm 5.2 should be modified. Modifications are presented in algorithm 5.3

5.5   Online belief update: forgetting information if it is too old

When exploring a dynamic environment, agents' beliefs can be quickly contradicted by changes in the values of the features they have to explore. Therefore belief revision is key to maintaining a correct knowledge about the world. Agents need to check the values of the features repeatedly even if they don't receive any information that contradict their beliefs. To do so, they need to gradually forget what they learned and take the right action to verify the features. To our knowledge, there is currently no work investigating this kind of forgetting mechanism in POMDPs.

In this section we describe a transformation which is applied online, after the usual belief update step and which aims at bringing the probability distributions in the agent's belief state close to the uniform distribution. We call this operation *smoothing*. To smooth an agent's belief

---

Data: sub-policies π_ℓ and associated value functions V_{π_ℓ}, current belief state $\mathcal{B}_{i,t}$
Result: action to execute α_{opt}

1 $V_{max} = -Infinity$;
2 $\alpha_{opt} = null$;
3 $expectedValue = 0$;
4 $listOfBestActions = \{\}$;
5 **foreach** $\mathcal{B}_{\ell,i,t}$ *composing* $\mathcal{B}_{i,t}$ **do**
6 $expectedValue = V_{\pi_\ell}(\mathcal{B}_{\ell,i,t})$;
7   **foreach** $\mathcal{B}_{\ell',i,t}$, $\ell' \neq \ell$ **do**
8 $expectedValue = expectedValue + V_{Idle}(\mathcal{B}_{\ell',i,t})$;
9   **end**
10   Compute the cost of performing action $\pi_\ell(\mathcal{B}_{\ell,i,t})$, noted $C_{\pi_\ell(\mathcal{B}_{\ell,i,t})}$;
11 $expectedValue = expectedValue - C_{\pi_\ell(\mathcal{B}_{\ell,i,t})}$;
12   **if** $expectedValue > V_{max}$ **then**
13 $empty(listOfBestActions)$;
14 $listOfBestActions = \{\pi_\ell(\mathcal{B}_{\ell,i,t})\}$;
15 $V_{max} = expectedValue$;
16   **end**
17   **if** $expectedValue == V_{max}$ **then**
18 $listOfBestActions = listOfBestActions \cup \{\pi_\ell(\mathcal{B}_{\ell,i,t})\}$;
19   **end**
20 **end**
21 $\alpha_{opt} = chooseRandomFrom(listOfBestActions)$;
22 return α_{opt} ;

Algorithm 5.3: Getting the action to execute from local optimal policies – version 2

state, we need to apply a transformation f to all the probabilities of the probability distributions in the belief state.

Definition 5.5 (Smoothing function). A function $f : [0,1] \to [0,1]$ is called a *smoothing function* for a variable X_k with domain $DOM(X_k)$ if it satisfies the following constraints:

$$
\begin{cases}
f(b_{i,t}^{j,k}(x_p)) \geq 0, \forall x_p \in DOM(X_k) & (5.9) \\[2mm]
\displaystyle\sum_{x_p \in DOM(X_k)} f(b_{i,t}^{j,k}(x_p)) = 1 & (5.10) \\[2mm]
f(\dfrac{1}{n_k}) = \dfrac{1}{n_k} & (5.11) \\[2mm]
\left| f(b_{i,t}^{j,k}(x_p)) - \dfrac{1}{n_k} \right| \leq \left| b_{i,t}^{j,k}(x_p) - \dfrac{1}{n_k} \right|, \forall x_p \in DOM(X_k) & (5.12) \\[2mm]
(b_{i,t}^{j,k}(x_p) - \dfrac{1}{n_k})(f(b_{i,t}^{j,k}(x_p)) - \dfrac{1}{n_k}) \geq 0, \forall x_p \in DOM(X_k) & (5.13)
\end{cases}
$$

where n_k denotes $|DOM(X_k)|$ and $b_{i,t}^{j,k}(x_p)$ is the probability that variable X_k takes the value x_p in the agent's belief state.

Constraints 5.9 and 5.10 define the fact that the result of the transformation must be a probability distribution, which means each term is positive and the sum of all terms equals 1.

Constraints 5.11 and 5.12 represent the fact that we want to smooth the probability distribution until a fixed point, which is the uniform distribution with each probability equals $\dfrac{1}{n_k}$. Those two constraints describe a contraction mapping admitting $\dfrac{1}{n_k}$ as a fixed point.

Finally, Constraint 5.13 formalizes that the transformation must keep the shape of the beliefs, that is to say that if a probability is greater than $\dfrac{1}{n_k}$, the result of the transformation should still be greater than or equal to $\dfrac{1}{n_k}$.

The choice of the shape of the smoothing function (linear, logarithmic, etc) depends on the problem considered and the type of belief update one wants to implement. In most cases however, beliefs should be revised linearly at each time step. Except in some very specific applications, we believe that there is no reason for the agent to forget quickly something it just learned and then to slow down, or on the contrary to keep beliefs almost unchanged at the beginning and then to suddenly forget about them. Therefore, we think that in most of the applications a linear function should be the most simple and efficient smoothing function to implement. That's why we will focus on this type of function in the remainder of the paper. The method to find a smoothing function based on another function shape remains similar to the one presented in this section.

Theorem 5.1. Let X_k be a random variables of the POMDP and $n_k = |DOM(X_k)|$ the number of possible values for X_k. Let $\mathcal{P}^k$ be the set of all the possible probability distributions over X_k. Let $(f_{X_k})_{X_k \in \mathcal{X}}$ be an indexed family of linear functions. Then each f_{X_k} is a

---

smoothing function if and only if $\exists a_k \in [0, 1]$ such that:

$$f_{X_k}: \quad \mathcal{P}^k \to \mathcal{P}^k$$
$$p \mapsto a_k \times p + \frac{1 - a_k}{n_k}$$

Proof. Let us consider a linear function defined by $f_{X_k}(p) = a \times p + b$. Let us note $p^i = p(X_k = x_i)$. The constraints system can be written :

$$
\begin{cases}
a\,p^i + b \geq 0, \forall p \in \mathcal{P}^k, \forall x_i \in DOM(X_k) & (5.14) \\[2mm]
\displaystyle\sum_{x_i \in DOM(X_k)} \left(a\,p^i + b\right) = 1, \forall p \in \mathcal{P}^k & (5.15) \\[2mm]
a\,\dfrac{1}{n_k} + b = \dfrac{1}{n_k} & (5.16) \\[2mm]
\left| a\,p^i + b - \dfrac{1}{n_k} \right| \leq \left| p^i - \dfrac{1}{n_k} \right|, \forall p \in \mathcal{P}^k, \forall x_i \in DOM(X_k) & (5.17) \\[2mm]
\left(a\,p^i + b - \dfrac{1}{n_k}\right)\left(p^i - \dfrac{1}{n_k}\right) \geq 0, \forall p \in \mathcal{P}^k, \forall x_i \in DOM(X_k) & (5.18)
\end{cases}
$$

Constraint 5.16 enables the derivation $b = \dfrac{1 - a}{n_k}$. Therefore we can replace it in the other constraints.

Constraints 5.15 becomes

$$\sum_{x_i \in DOM(X_k)} \left(a\,p^i + \frac{1 - a}{n_k}\right) = 1, \forall p \in \mathcal{P}^k$$
$$\Leftrightarrow \sum_{x_i \in DOM(X_k)} \left(a\,p^i\right) + 1 - a = 1, \forall p \in \mathcal{P}^k$$
$$\Leftrightarrow a\left[\sum_{x_i \in DOM(X_k)} \left(p^i\right) - 1\right] + 1 = 1, \forall p \in \mathcal{P}^k$$

p being a probability distribution, $\sum_{x_i \in DOM(X_k)} \left(p^i\right) = 1$. Therefore this constraint is true whatever the value of a.

Constraint 5.17 becomes

$$\left| a\,p^i + \frac{1 - a}{n_k} - \frac{1}{n_k} \right| \leq \left| p^i - \frac{1}{n_k} \right|, \forall p \in \mathcal{P}^k, \forall x_i \in DOM(X_k)$$
$$\Leftrightarrow \left| a\left(p^i - \frac{1}{n_k}\right) \right| \leq \left| p^i - \frac{1}{n_k} \right|, \forall p \in \mathcal{P}^k, \forall x_i \in DOM(X_k)$$
$$\Leftrightarrow |a|\left| p^i - \frac{1}{n_k} \right| \leq \left| p^i - \frac{1}{n_k} \right|, \forall p \in \mathcal{P}^k, \forall x_i \in DOM(X_k)$$
$$\Leftrightarrow |a| \leq 1 \forall p \in \mathcal{P}^k, \forall x_i \in DOM(X_k)$$

Constraint 5.18 becomes

$$\left(a\,p^i + \frac{1 - a}{n_k} - \frac{1}{n_k}\right)\left(p^i - \frac{1}{n_k}\right) \geq 0, \forall p \in \mathcal{P}^k, \forall x_i \in DOM(X_k)$$
$$\Leftrightarrow a\left(p^i - \frac{1}{n_k}\right)\left(p^i - \frac{1}{n_k}\right) \geq 0, \forall p \in \mathcal{P}^k, \forall x_i \in DOM(X_k)$$
$$\Leftrightarrow a\left(p^i - \frac{1}{n_k}\right)^2 \geq 0, \forall p \in \mathcal{P}^k, \forall x_i \in DOM(X_k)$$
$$\Leftrightarrow a \geq 0 \forall p \in \mathcal{P}^k, \forall x_i \in DOM(X_k)$$

Therefore we obtain $a \in [0, 1]$.

Constraint 5.14 becomes $a\, p^i + \dfrac{1 - a}{n_k} \geq 0$. Knowing that $a \in [0, 1]$, then $\dfrac{1 - a}{n_k} \geq 0$. Since p^i is a probability, $p^i \geq 0$. We conclude that Constraint is true for $a \in [0, 1]$.

We can conclude that any f_{X_k} respecting the previous constraints is written : $f_{X_k}(p) = a \times p - \dfrac{1 - a}{n_k}, a \in [0, 1]$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\square$

In the MAPING framework, when an agent receives a new observation, it first updates its belief states as usual using a Bayesian update. Second, it applies the smoothing function on all the elements of the probability distributions composing its belief state, depending on the transition function. Indeed, it seems reasonable that the beliefs concerning variables are likely to change are smoothed more frequently than beliefs concerning variables that are known to be rather stable. Therefore each variable has a probability to be smoothed equals to its probability of change. The complete belief update step is described in algorithm 5.4, where $T_{X_k}(x_{i,t}, x_{j,t+1})$ is the probability that the variable X_k has value x_i at decision step t and value x_j at decision step $t + 1$.

Data: agent i's beliefs $\mathcal{B}_{i,t}$, observation o, set of smoothing functions $\{f_{X_k} \forall X_k \in \mathcal{E}\}$
Result: agent i's beliefs $\mathcal{B}_{i,t+1}$

1 **foreach** $X_k \in \mathcal{E}$ **do**

2 $\quad \mathcal{B}_{i,t+1}^k = \left\langle update(b_{i,t}^{j,k}, o) \right\rangle_{j \in \mathcal{AG}}$;

3 $\quad probabilityOfChange = \displaystyle\sum_{x_i, x_j \in DOM(X_k), x_i \neq x_j} T_{X_k}(x_{i,t}, x_{j,t+1})$;

4 $\quad randomNumber = chooseRandomNumber()$;

5 $\quad$ **if** $randomNumber < probabilityOfChange$ **then**

6 $\quad\quad \mathcal{B}_{i,t+1}^k = \left\langle f_{X_k}(b_{i,t+1}^{j,k}) \right\rangle_{j \in \mathcal{AG}}$;

7 $\quad$ **end**

8 **end**

Algorithm 5.4: Belief update step with smoothing

$$\mathcal{B}_{i,t+1} = \left\langle f_{X_k}(update(b_{i,t}^{j,k}, o)) \right\rangle_{j \in \mathcal{AG}, X_k \in \mathcal{E}}$$

where $update(b_{i,t}^{j,k}, o)$ is the result of the updating process of the belief over the variable X_k with observation o and f_{X_k} is the smoothing function.

5.6 The MAPING framework for heterogeneous agents

In this thesis, our purpose is to combine mobile agents with different capabilities, called heterogeneous agents, to perform an efficient exploration, as we did in the running example with the three agent types. This way, agents can complete each others and fuse their results to obtain more complete information. In the example, if the color-detection agent detects a fire color in a room and the temperature-detection agent detects a high temperature in the same room, the information provided by both agents is fused to increase the belief that this room is on fire. This could not be possible only color-detection agents or only temperature-detection agents. In the first case, a picture of a fire could be responsible for the fire color, in the second the presence of

a human or a heater could be responsible for the high temperature. By combining agents with different sensing capabilities, we are able to increase the amount of information we get from the environment. This is the basis of data fusion.

For simplicity reason, we developed MAPING and performed the first experiments with homogeneous agents. With homogeneous agents, all the agents share the same observation function. The update after receiving an observation, presented in section 5.5, is easy since the agent can use its own observation function to update its beliefs. However, in the case of heterogeneous agents, each agent has its own observation function, depending on its capabilities. The agent which receives the observation should therefore know the observation function of the emitter to be able to update correctly its beliefs. Though it is not desirable that the agent sends its observation function with the observation since it would increase needed dramatically the size of the messages and so the communication resources. During the thesis, we didn't manage to find a good theoretical solution to this problem. We solved it practically by giving all the observations functions to all agents at the system initialization. This enables to apply MAPING to heterogeneous agents. Nevertheless the problem of updating the agents' beliefs after a communication with heterogeneous agents remains open.

Chapter 6

Conclusion of part II

Agents working as a group with a common goal needs to cooperate, to coordinate, and to exchange information about what they find and what they know. When the communication is limited, agents need to select the information to send in order to limit the exchange to the minimum required to complete their mission. The purpose of this part was to present a complete framework enabling a multiagent system to perform event exploration in a completely decentralized way while limiting its communication. We first presented in Chapter 4 a degree of relevance which quantifies the relevance of an observation for a given agent. This relevance is assessed regarding the beliefs the agent has about the environment. If the observation is correct and can improve the agent's beliefs by giving new information or by rendering it more precise, then the observation is considered relevant. To our knowledge, this is the first piece of work in the literature suggesting such a degree. Previous works presented in Chapter 2 investigated the agent-based relevance problem. Most of the studies explore the properties of agent-based relevance, Floridi's work especially [Floridi, 2008]. Roussel and Cholvy [Roussel and Cholvy, 2009] suggested a definition of relevance for BDI architecture. Their work lays the foundations for an agent-based theory of relevance, but is limited to epistemic logic. In addition, it does not make it possible to compare the relevance of two observations for a given agent, what our degree of relevance does.

Based on this degree of relevance, we presented in chapter 5 a offline-online framework named MAPING (Multi-Agent Planning for INformation Gathering) that performs event exploration in a complete decentralized way. This framework includes a decision-theoretic model, based on POMDPs, in which each agent has beliefs about the environment but also about other agents' beliefs. Using those beliefs, each agent is able to assess the relevance of an observation for another agent. In MAPING , the agents are not rewarded when they reach a specific state but when they gather and exchange relevant observations. The reward function not being convex has been considered as a problem, casting doubt on our degree of relevance. Indeed, in [Araya-Lopez et al., 2010], Araya-Lopez et al. states that convexity is a natural property for uncertainty measures since the purpose is to assign higher rewards to beliefs that give higher probabilities of being in a certain state. However our degree of relevance does not contain only uncertainty measures. Indeed, the main task of our agent is to collect observations that give information about the system to reach quickly a satisfactory belief state and send observations to other agents - which will render their belief state even more precise. Therefore instead of being only interested in reaching a very precise belief state, we are more interested in collecting informative observations. This is the purpose of the degree of soundness $sound_{i,t}(o)$, which is not proved to be convex or concave.

The idea of representing other agents' knowledge in a decision process has also been used in [Gmytrasiewicz and Doshi, 2005] by Gmytrasiewicz and Doshi with the Interactive-POMDP (I-POMDP). The I-POMDP extends the classic POMDP framework by including new beliefs about other agents, such as their state or policy. The I-POMDP model is more expressive than MAPING since it includes more information about other agents and can deal with cooperative and non-cooperative settings. However, as classical POMDPs, its reward function is based on the system states and the mechanisms used seem hard to adapt to belief states based reward function. In addition, its expressiveness renders it complex to manipulate and to solve. Since the event exploration problem is more restrictive - fully cooperative agents, epistemic actions, etc. - it seemed more interesting to us to consider a less expressive model but simpler to use in real type applications.

To overcome the scalability problem of solving this POMDP of MAPING , we suggested a resolution technique based on variable and observation independence. The POMDP is split into a set of smaller sub-POMDP which are solved offline independently. Then, during execution, the agent compare the value functions of the sub-POMDPs to choose the next action to perform. This technique does not provide optimal policies, but the results presented in the next part tends to show that the policies are satisfactory. The online part of the framework also includes a smoothing module, which is applied after the beliefs update step. This module aims at smoothing gradually the beliefs of the agent. This is motivated by the fact that we consider dynamic environments and that the probability of a change - and so of previous beliefs becoming incorrect - increases among time.

Part III present experimental protocol and results to validate the MAPING model.

Part III

Experiments

Chapter 7

Protocol description

Contents

In this chapter, we describe the scenario and the environments used for the evaluation, the way we implemented the model and the measures we used to evaluate our system. We were very careful to choose environments that reflects real life applications and real life problems. Four environments are considered: three to present different configurations and problems that can occur in real-like scenarios, and one on a bigger environment to evaluate the scalability of the model.

7.1 Scenario description

The scenario we used for evaluation is an instantiation of the scenario described in the preamble. We considered indoor and outdoor locations, divided in n different areas. For each area, two events are explored: the area can be *empty* or not, and the area can be *on fire* or not. The system is made up of robots from three different types: 1. a *face-detection* type, equipped with a camera and able to detect faces 2. a *color-detection* type, equipped with a camera and able to detect 3. a *temperature-detection* type, equipped with a thermocamera and able to detect hotspots. The *face-detection* and the *temperature-detection* robots can obtain information about the event *empty*. The *color-detection* and the *temperature-detection* robots can obtain information about the event *on fire*.

First we evaluate our model with a static scenario: the events, in each room, are assigned at the beginning and doesn't change in time. This is done to evaluate the ability of the system to actually explore an environment and build correct beliefs about it. Second we evaluate our model with a dynamic scenario: the events, in each room, are assigned at the beginning but may change during the simulation according to the transition function described in Section 7.3.

7.2 Environments description

We consider four different environments reflecting real cases and problems that may arise in such environments:

- a *house* environment

- a *one-way transition* environment

- an *outdoor* environment

- an *Élysée Palace* environment.

TThe *home* environment uses the map of a one-storey house. The maps of those two environments are presented on Figures 7.1. Those environments don't present any specific difficulty as all openings between rooms can be used in both directions and moving from a room to another adjacent one presents no specific cost. The *one-way transition* environment presents the case when some transitions between areas can only be done in one-way. This can be the case for instance with firedoors that can be used only to exit a room, or with some kind of doors that the robot can only cross in one way for technical reasons (for instance if the robot may push the door but not pull it). The map of this environment is presented on Figure 7.2a. The *outdoor* environment represents a case where the transition between different areas is costly due to a hill, a muddy ground, etc. We based our map on the topographic map of a mountain location at the border between France and Switzerland and the map is presented on figure 7.2b. In these scenarios we considered that all robots have the same difficulties to move through one-way doors or across costly grounds. It is nevertheless perfectly possible to adapt the costs for each robot according to its capacities. The four environments described previously are divided into 10 areas. Though they are inspired from real applications, they are small. The *Élysée Palace* environment is used to demonstrate the scalability of our model on medium-size environments. Figure 7.3 presents the map of the first floor of the Élysée Palace and figure 7.3 presents the division we made in 28 areas. For simplicity reasons, we considered only one transition between two rooms, even if the map shows that several openings are present. We also considered no specific costs for transitions despite the stairs.
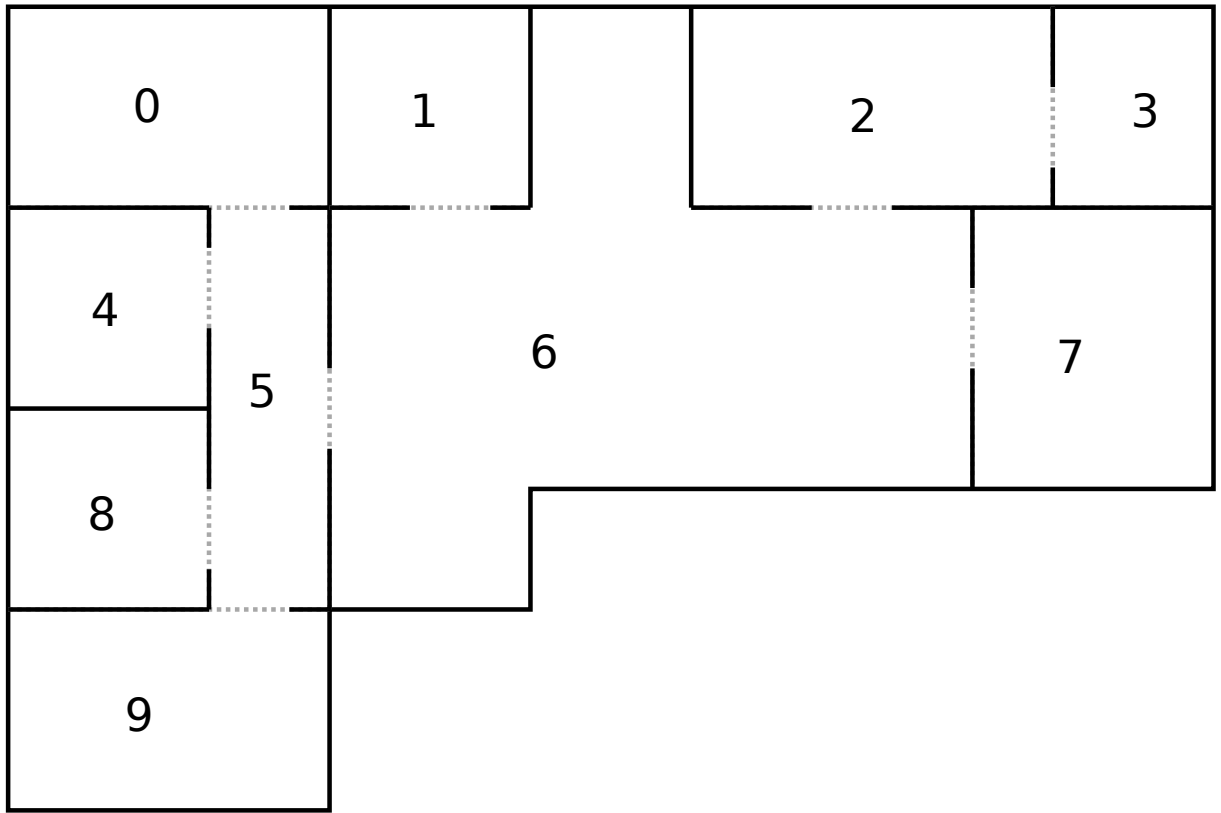
Figure 7.1: The house environment
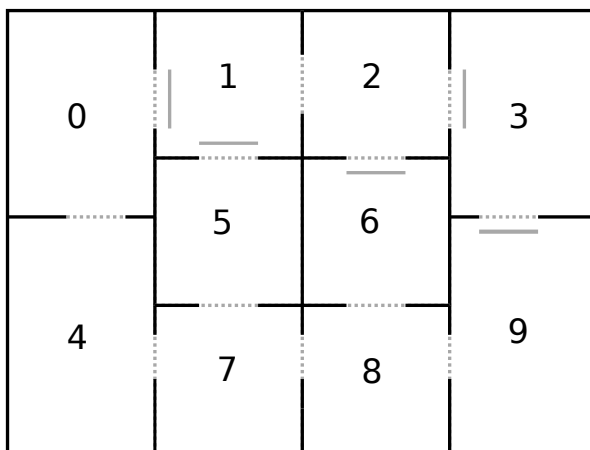
(a) The one-way transitions environment
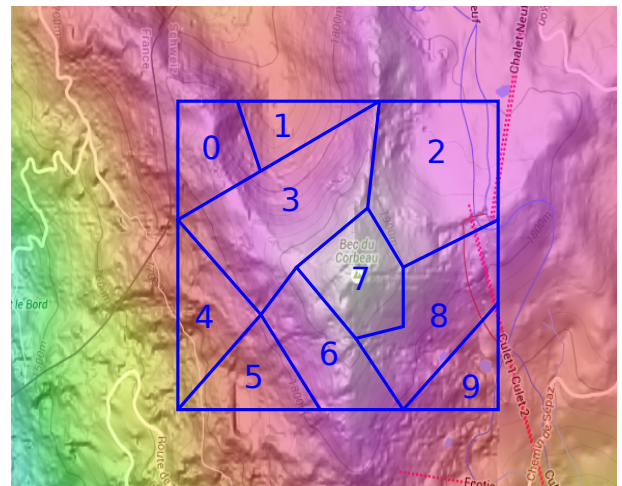
(b) The outdoor environment

Figure 7.2: The costly environments
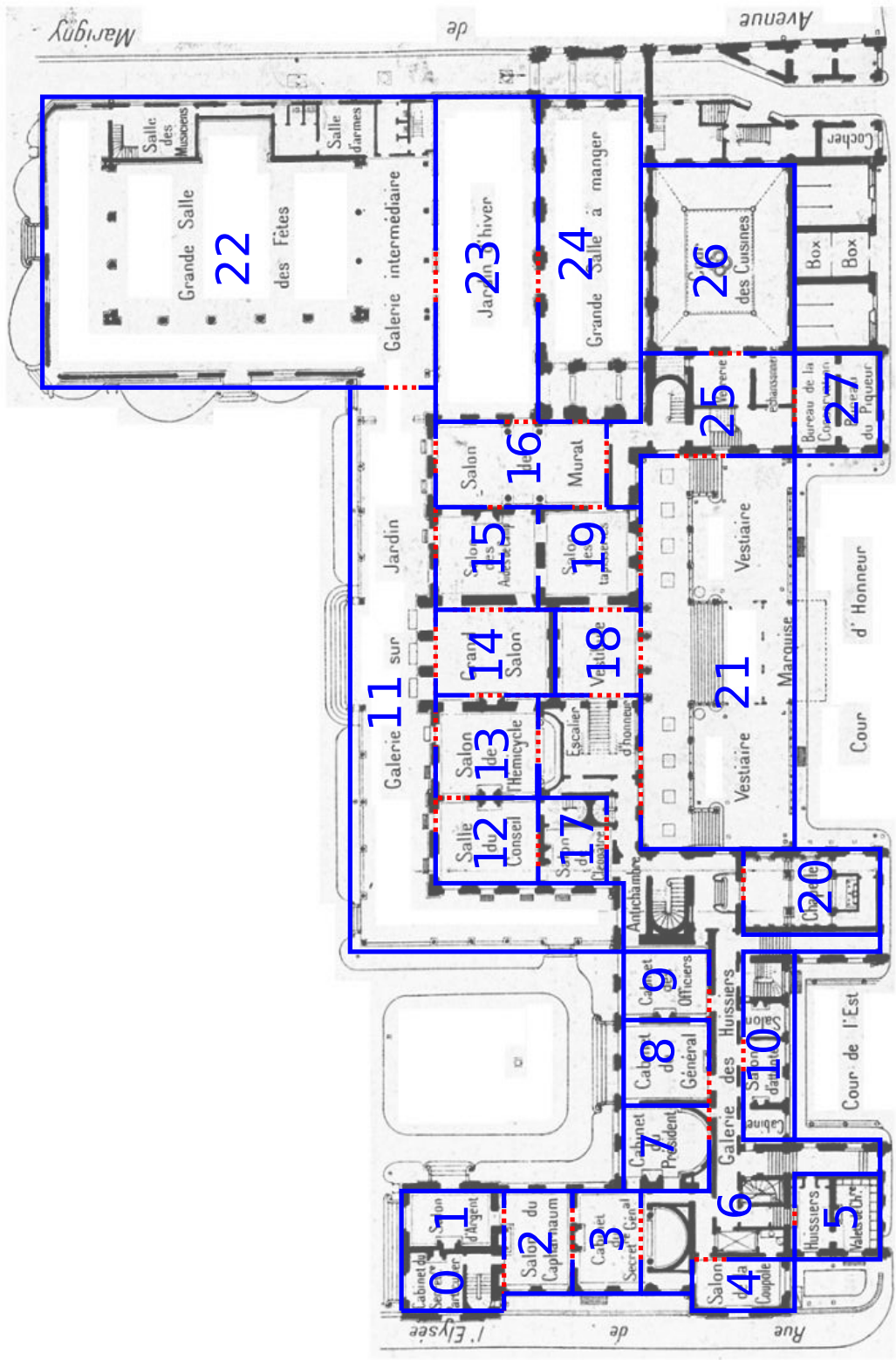
Figure 7.3: The Élysée Palace environment

7.3   System modeling and implementation parameters

Model of the POMDP

The topology of the environment is represented by a directed weighted graph, in which the nodes represent the areas. A node A is connected to a node B if the robot can move from the area A to the area B. The weight associated to the edges represents the cost of moving from the area A to the area B. Figures 7.4, 7.5 and 7.6 present the directed graphs for the four environments. For the sake of readability, the graph of the Élysée Palace environment has been simplified: all the edges are bi-directional and their weight is 1. Using these graphs, the cost of an *Explore*-type
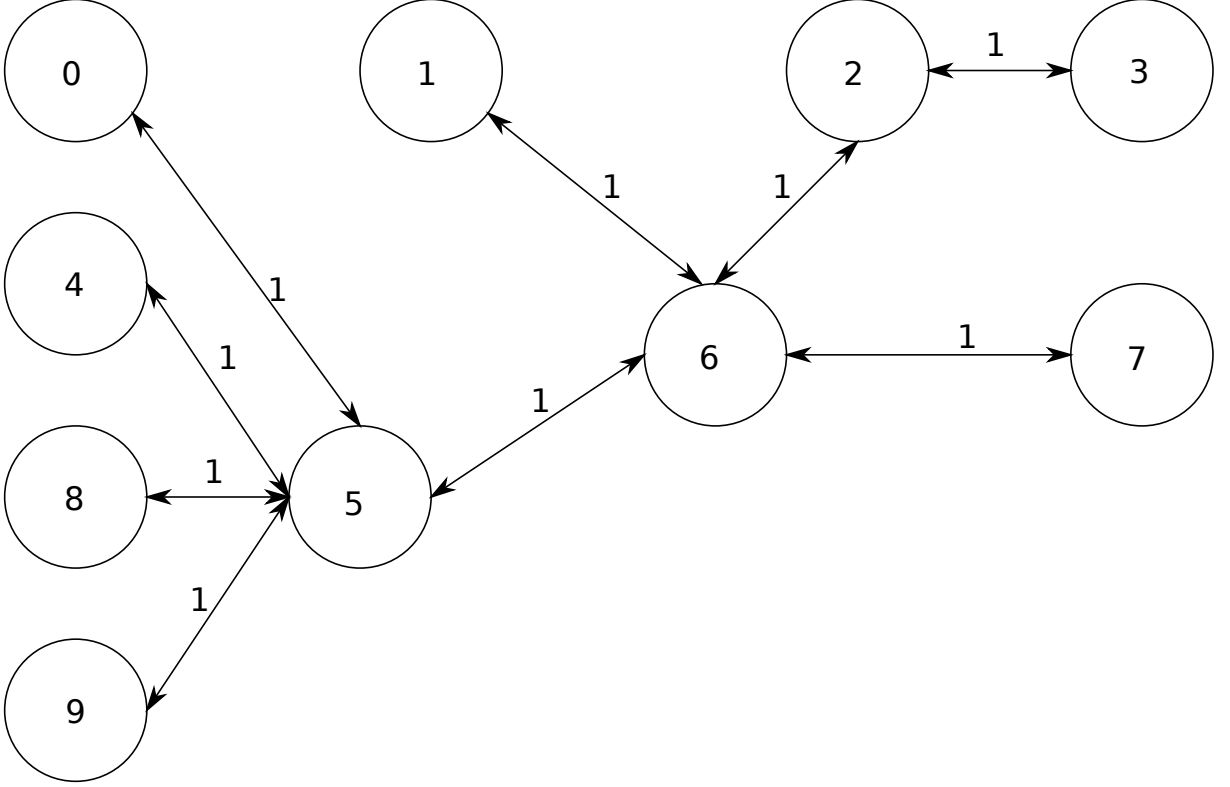
Figure 7.4: Weighted directed graph of the house environment

action is computed thanks a classic A^* algorithm between the current position of the robot and the area to explore. The cost of a *Communicate*-type action is constant. We experimented with two communication costs: a cost of 2 and a cost of 3, which means that communicating is as expensive as moving among a path with a cost of 2 (respectively 3). The robots' starting points are defined randomly when launching the system.

Since we are considering three types of robots (*face-detection*, *color-detection* and *temperature-detection*) that has different sensing capabilities, we need to model and to solve three different POMDPs. Those POMDPs will share the same set of states, the same set of actions, and the same transition and reward functions. However, they differ by their set of observations and their observation function, which depends on the capabilities of the robot considered. The variables of all POMDP for the small environments are:

$$\mathcal{E} = \{RoomEmpty_k, RoomOnFire_k, \forall k \in [1, n]\}$$

(a) One way environment

(b) Outdoor environment
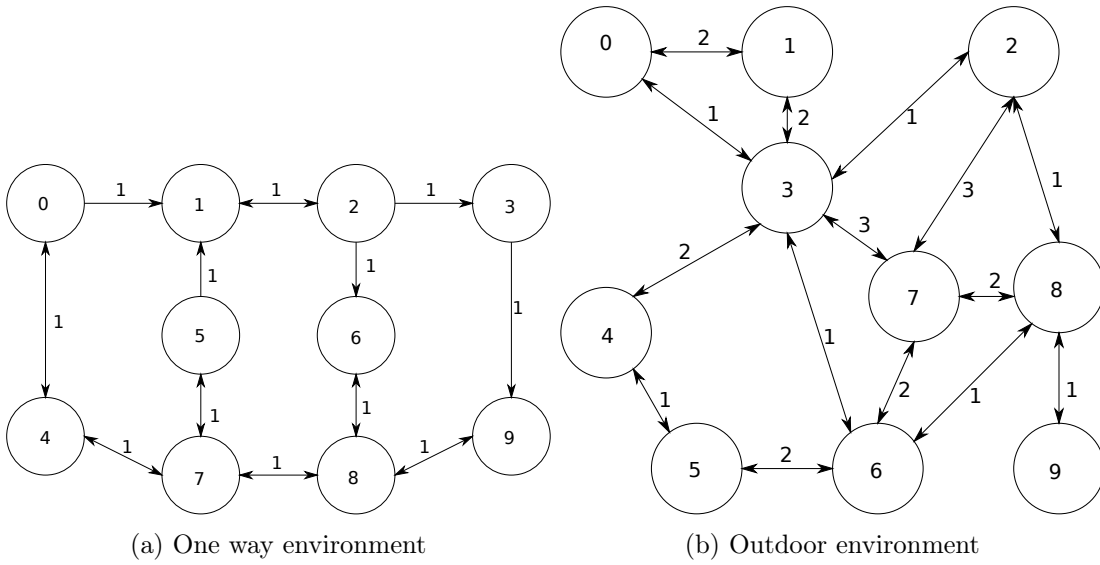
Figure 7.5: The weighted directed graphs of the costly environments
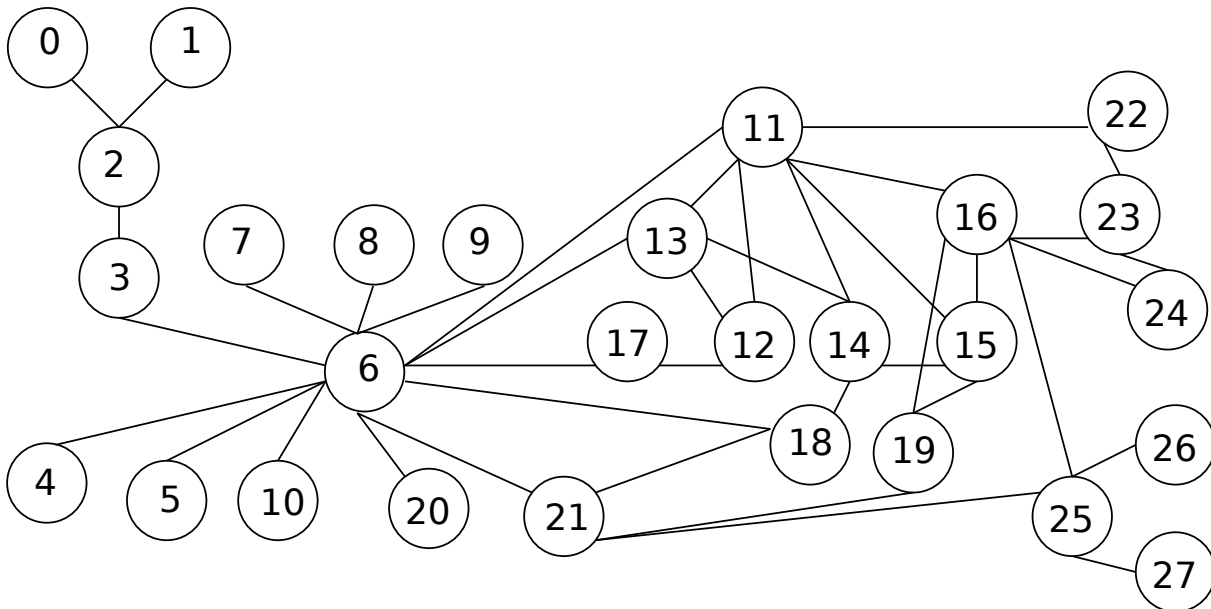
Figure 7.6: Weighted directed graph of the Élysée Palace environment

n being the number of rooms ($n = 10$ for the small environments and $n = 28$ for the Élysée Palace environment). Each variable has two possible values: *true* or *false*.

Each robot can receive two observations which depend on the robot's type. The *face-detection* robots can receive the observations

$$\mathcal{O}_{face} = \{FaceDetected, noFaceDetected\}$$

The *color-detection* robots can receive the observations

$$\mathcal{O}_{color} = \{FireColorDetected, NoFireColorDetected\}$$

The *temperature-detection* robots can receive the observations

$$\mathcal{O}_{temperature} = \{HotSpotDetected, NoHotSpotDetected\}$$

Since our main purpose is neither the localization nor the sensor data processing, we assumed that the robots' position are exactly known, and we implemented a noisy simulator that sends directly the high-level observations described previously, depending on the robot's position and type. We considered that the simulator will send the correct observation 98% of the time.

We considered that the transition for each variable is independent from other variables, that is to say $P(RoomEmpty_{R_k,t+1} = true|s_t) = P(RoomEmpty_{R_k,t+1} = true|RoomEmpty_{R_k,t})$, and similarly $P(RoomOnFire_{R_k,t+1} = true|s_t) = P(RoomOnFire_{R_k,t+1} = true|RoomOnFire_{R_k,t})$. Therefore the transition function is based on the following probabilities:

$$P(RoomEmpty_{R_k,t+1} = true|RoomEmpty_{R_k,t} = true) = 0.9$$
$$P(RoomEmpty_{R_k,t+1} = false|RoomEmpty_{R_k,t} = true) = 0.1$$
$$P(RoomEmpty_{R_k,t+1} = true|RoomEmpty_{R_k,t} = false) = 0.7$$
$$P(RoomEmpty_{R_k,t+1} = false|RoomEmpty_{R_k,t} = false) = 0.3$$
$$P(RoomOnFire_{R_k,t+1} = true|RoomOnFire_{R_k,t} = true) = 0.99$$
$$P(RoomOnFire_{R_k,t+1} = false|RoomOnFire_{R_k,t} = true) = 0.01$$
$$P(RoomOnFire_{R_k,t+1} = true|RoomOnFire_{R_k,t} = false) = 0.1$$
$$P(RoomOnFire_{R_k,t+1} = false|RoomOnFire_{R_k,t} = false) = 0.9$$

Then the transition function is computed using the joint instantiation of the variables. For instance, if

$$
\begin{aligned}
s_t = \{&RoomEmpty_1 = true, RoomOnFire_1 = false \\
&RoomEmpty_2 = true, RoomOnFire_2 = false \\
&RoomEmpty_3 = true, RoomOnFire_3 = false \\
&RoomEmpty_4 = false, RoomOnFire_4 = false \\
&RoomEmpty_5 = true, RoomOnFire_5 = false \\
&RoomEmpty_6 = false, RoomOnFire_6 = false \\
&RoomEmpty_7 = true, RoomOnFire_7 = false\}
\end{aligned}
$$

and

$$s_{t+1} = \{RoomEmpty_1 = true, RoomOnFire_1 = false$$
$$RoomEmpty_2 = false, RoomOnFire_2 = false$$
$$RoomEmpty_3 = true, RoomOnFire_3 = false$$
$$RoomEmpty_4 = true, RoomOnFire_4 = false$$
$$RoomEmpty_5 = true, RoomOnFire_5 = false$$
$$RoomEmpty_6 = false, RoomOnFire_6 = false$$
$$RoomEmpty_7 = true, RoomOnFire_7 = false\}$$

we have

$$T(s_t, s_{t+1}) = 0.9 * 0.1 * 0.9 * 0.7 * 0.9 * 0.3 * (7 * 0.9) \simeq 0.096$$

To determine the value of δ, we know that $\delta = 1 - ProbabilityOfChange$. But

$$ProbabilityOfChange = 1 - \sum_{s_i \in \mathcal{S}} P(s_{i,t+1}|s_{i,t}) \times P(s_{i,t})$$

where $s_{i,t}$ is an instantiation of all the variables $X_k \in \mathcal{E}$ and $\mathcal{S}$ is the set of all possible instantiations of the variables $X_k \in \mathcal{E}$. Therefore,

$$\delta = \sum_{s_i \in \mathcal{S}} P(s_{i,t+1}|s_{i,t}) \times P(s_{i,t})$$

However, since $P(s_{i,t})$ depends on $P(s_{i,t-1})$, and so on until s_0, the value of δ would change at each time step and so become a variable of the POMDP, which is not desirable. To deal with this issue, we used for δ the average probability that a state remains the same:

$$\delta = \frac{\sum_{s_i \in \mathcal{S}} P(s_{i,t+1}|s_{i,t})}{|\mathcal{S}|}$$

By considering the transition function described previously, $delta \simeq 0.602$.

The observation function is dependent on the robot's type. Robot *face-detection* robots will have an observation function shaped as follows:

$$\Omega(FaceDetected, RoomEmpty_{R_k} = true, Explore(RoomEmpty_{R_k})) = 0.2$$
$$\Omega(FaceDetected, RoomEmpty_{R_k} = false, Explore(RoomEmpty_{R_k}) = 0.8$$
$$\Omega(NoFaceDetected, RoomEmpty_{R_k} = true, Explore(RoomEmpty_{R_k}) = 0.8$$
$$\Omega(NoFaceDetected, RoomEmpty_{R_k} = false, Explore(RoomEmpty_{R_k})) = 0.2$$

Color-detection robots will have an observation function shaped as follows:

$$\Omega(FireColorDetected, RoomOnFire_{R_k} = true, Explore(RoomOnFire_{R_k})) = 0.9$$
$$\Omega(FireColorDetected, RoomOnFire_{R_k} = false, Explore(RoomOnFire_{R_k})) = 0.1$$
$$\Omega(NoFireColorDetected, RoomOnFire_{R_k} = true, Explore(RoomOnFire_{R_k})) = 0.1$$
$$\Omega(NoFireColorDetected, RoomOnFire_{R_k} = false, Explore(RoomOnFire_{R_k})) = 0.9$$

Temperature-detection robots will have an observation function shaped as follows:

$$\Omega(HotSpotDetected, RoomEmpty = true \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomEmpty_{R_k})) = 0.2$$
$$\Omega(HotSpotDetected, RoomEmpty = true \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomEmpty_{R_k})) = 0.05$$
$$\Omega(HotSpotDetected, RoomEmpty = false \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomEmpty_{R_k})) = 0.6$$
$$\Omega(HotSpotDetected, RoomEmpty = false \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomEmpty_{R_k})) = 0.15$$

$$\Omega(HotSpotDetected, RoomEmpty = true \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomOnFire_{R_k})) = 0.25$$
$$\Omega(HotSpotDetected, RoomEmpty = true \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomOnFire_{R_k})) = 0.05$$
$$\Omega(HotSpotDetected, RoomEmpty = false \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomOnFire_{R_k})) = 0.6$$
$$\Omega(HotSpotDetected, RoomEmpty = false \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomOnFire_{R_k})) = 0.1$$

$$\Omega(NoHotSpotDetected, RoomEmpty = true \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomEmpty_{R_k})) = 0.25$$
$$\Omega(NoHotSpotDetected, RoomEmpty = true \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomEmpty_{R_k})) = 0.6$$
$$\Omega(NoHotSpotDetected, RoomEmpty = false \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomEmpty_{R_k})) = 0.05$$
$$\Omega(NoHotSpotDetected, RoomEmpty = false \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomEmpty_{R_k})) = 0.1$$

$$\Omega(NoHotSpotDetected, RoomEmpty = true \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomOnFire_{R_k})) = 0.2$$
$$\Omega(NoHotSpotDetected, RoomEmpty = true \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomOnFire_{R_k})) = 0.6$$
$$\Omega(NoHotSpotDetected, RoomEmpty = false \cap RoomOnFire_{R_k} = true,$$
$$Explore(RoomOnFire_{R_k})) = 0.05$$
$$\Omega(NoHotSpotDetected, RoomEmpty = false \cap RoomOnFire_{R_k} = false,$$
$$Explore(RoomOnFire_{R_k})) = 0.15$$

The discretization of the belief state is the one described in Section 5.4. We discretized the probability distributions by making distributions with 11 elements:

$$b_{i,t}^k \in \{(0.1, 0.9); (0.2, 0.8); (0.3, 0.7); (0.4, 0.6); (0.5, 0.5); (0.6, 0.4); (0.7, 0.3); (0.8, 0.2); (0.9, 0.1)\}$$

Separation into sub-POMDPs

As explained in the previous section, we consider transition independence: $P(RoomEmpty_{R_k,t+1} = true|s_t) = P(RoomEmpty_{R_k,t+1} = true|RoomEmpty_{R_k,t})$ and $P(RoomOnFire_{R_k,t+1} = true|s_t) = P(RoomOnFire_{R_k,t+1} = true|RoomOnFire_{R_k,t})$. We can build n independent sets of variables, n being the number of areas.

$$X_\ell = \{RoomEmpty_{R_\ell}, RoomOnFire_{R_\ell}\}, \ell \in [0, n]$$

The observation set stays unchanged and includes 2 elements depending on the type of the robot. Based on those sets of variables and observations, we can divide each of the three POMDPs into n sub-POMDPs as described in Section 5.4 and solve them independently. Since the Explore actions set is build upon the variables of the POMDP, it is also possible to consider only a reduced set of actions:

$$\mathcal{A}_\ell = \{Explore(X_\ell)\forall \ell \in [0, n]\} \cup \{Communicate(o, i)\forall o \in \mathcal{O}, \forall i \in \mathcal{AG}\}$$

On top of that, all sub-POMDP for an agent are similar: their variables represent the two possible states of a given room. It is so possible to solve only one sub-POMDP and combine n times this sub-POMDP to obtain the global policy. Therefore, we only have 3 sub-POMDP to solve. The solving has been made using a simple Value Iteration algorithm.

7.4 Evaluation criteria

In this section we describe the criteria we used to evaluate the efficiency of our system. The main purpose of the system is to reach quickly an correct and accurate belief state and to maintain it among time.

To measure the correctness and accuracy of the agents' belief states, we considered a *perfect belief state*, which is the one associating 1 to the true state and 0 to the others, and we recorded the evolution of the Hellinger distance between the agents' belief states and this perfect belief state. Though it is included in the Hellinger distance, we also recorded a score for each agent related to the number of incorrect and correct beliefs. An agent is considering having incorrect belief if it's belief concerning a variable is inverted compared to the perfect belief state. For instance, if the perfect belief state for a variable X_k is given by $(0, 1)$ and the agent has the belief state $(0.7, 0.3)$ for the same variable, it is considered as being wrong. To compute this score, we compare the belief of the agent for each variable. Each correct belief is worth 1 point, each incorrect belief is worth -1 point and each undefined belief (which means a belief $(0.5, 0.5)$) is worth 0 point.

The second purpose of our system is to perform efficient event exploration while reducing the number of messages sent. Therefore we tracked the number of message send for each scenario by each agent. Since the communication makes it possible for the agents to approximate the beliefs of other agents, we will also evaluate the quality of this approximation by computing the Hellinger distance between the real belief of an agent and the approximation another agents made of this belief. In parallel, we recorded the path followed by the agents and the average distance traveled by the agent at each decision step. This is done to ensure that the agents are following reasonable paths to explore their environment and that they are not jumping between rooms far from each others.

Chapter 8

Low communication cost

Contents

In this chapter we present the results of the experiments developed with a low communication cost. We will expose the results for each environment. We so evaluate the efficiency of the exploration by assessing that the system is able to maintain a correct and accurate belief state. Then we evaluate the communication strategy by tracking the number of communications sent an the quality of the belief approximation of each agent. Finally, we evaluate the quality of the paths travelled by the agents and the average cost they pay for each explore action. We will expose in details the results in the house environment for both static and dynamic cases. For the other environments, we will only expose the dynamic case and describe briefly the graphics when there is nothing special to mention and when the behavior is similar to those observed in the house environment. We will stress out the points of difference and interest. As a conclusion we will analyze the differences between the environments and conclude about the global efficiency of MAPING under medium communication constraints.

8.1 Simple configuration: the house environment

The house environment is a classic indoor configuration corresponding to real-type application. It is used to evaluate the efficiency of the system without special constraints.

8.1.1 Evaluating the exploration efficiency

The system is considered efficient in exploring the environment if it manages to maintain a correct and accurate belief states among time, despite the environment changes. To measure the correctness of the belief state, we tracked the number of correct beliefs and incorrect beliefs among time. A belief is considered correct if it matches the shape of the perfect belief.

Example 8.1 If we consider only one room of the environment, which is empty and not on fire, the perfect belief state represented the environment's state is

$$\mathcal{B}^{*,\mathcal{E}} = \begin{pmatrix} (1,0) \\ (0,1) \end{pmatrix} \quad \begin{array}{l} \%RoomEmpty = true \\ \%RoomOnFire = false \end{array}$$

If we consider the agent a which has the following belief state:

$$\mathcal{B}_a^{\mathcal{E}} = \begin{pmatrix} (0.2, 0.8) \\ (0.3, 0.7) \end{pmatrix}$$

Then its belief concerning the variable **RoomEmpty** is incorrect and its belief concerning the variable **RoomOnFire** is correct.

Figure 8.1 presents the results of this tracking in the static and the dynamic case. The blue plain curve shows evolution the average number of correct beliefs. Since the environment contains 10 rooms and 20 variables, the maximum number of correct or incorrect states is 20. The yellow dotted curve shows the number of variables that changed value at this decision step. The probability of changes in the environment follows the transition function described in section 7.3 and is implemented with algorithm 8.1 Using this algorithm, the bigger the environment is,

---

```
1  foreach decision step do
2      foreach X_k ∈ E do
3          select a random number ;
4          if X_{k,t} = true & randomNumber < P(X_{k,t+1} = false|X_{k,t} = true) then
5              X_{k,t+1} = false ;
6          end
7          if X_{k,t} = false & randomNumber < P(X_{k,t+1} = true|X_{k,t} = false) then
8              X_{k,t+1} = true ;
9          end
10     end
11 end
```

Algorithm 8.1: Performing the changes in the environments

---

the most probable is the change. That is why in our environments, at almost every time step, one or more variables change. The red dots shows the number of incorrect observations sent to all agents at each decision step. When an agent perform an explore-type action, it asks the

simulator for an observation corresponding to its type and its position. The simulator sends the correct observation 98% of the time. We implemented this to model the margin of error of real sensors and detection algorithms.

We can see in the static case from figure 8.1a that the maximum number of correct beliefs is quickly reached (around 150 decision steps), which means that all agents in the system reach quickly a correct belief state. We also notice that, as expected, the average number of correct beliefs varies only when incorrect observations are sent by the simulator. However, the system is able to correct itself quickly. In the dynamic case, the curve varies much more, as expected. Indeed, each time the state of the environment changes all the correct beliefs become incorrect. However the system manages to keep an average number of correct beliefs around 14, which means that in average the agents are correct about 70% of the environment. We notice between decision steps 300 and 350 a big decrease in the number of correct beliefs states. This is obviously due to the 2 incorrect observations send at iteration 301 and associated to big changes. However in less than 50 decision steps the system is able to reach again its average number of correct beliefs.

Those results confirm that the system is able to keep a correct representation of the environment, but doesn't show the accuracy of this representation. Indeed, in this tracking beliefs $(0.6, 0.4)$ and $(0.9, 0.1)$ are treated equally, though in reality they show a very different knowledge. In the first case, the agent could be considered as "having an idea" about the current state of the environment while in the second case the agent is pretty sure about the current state of the environment. To evaluate the accuracy of the belief state, we tracked the evolution of the Hellinger distance between the current belief state of the agents and the perfect belief state representing the environment. Results are presented on figure 8.2.

In this experiment, we used as reference four levels of belief states. For each level, we computed the Hellinger distance between this belief state and the perfect belief state and plotted it to evaluate the accuracy of the agents' belief state. The level 1 is the less precise belief state. It corresponds to a belief state where all beliefs are $(0.6, 0.4)$ or $(0.4, 0.6)$. This is the agent "having an idea" mentioned previously. In the level 2, all beliefs in the belief states are $(0.7, 0.3)$ or $(0.3, 0.7)$. Though this belief is still not completely precise, it is more reliable than level 1 beliefs. In the level 3, all beliefs are $(0.8, 0.2)$ or $(0.2, 0.8)$. In this case we consider that the belief state is rather precise and refers it as an acceptable precision. Finally, the level 4 is the most precise and all the beliefs are $(0.9, 0.1)$ or $(0.1, 0.9)$. It is obvious that an agent will rarely have a belief state in which all beliefs belong to a same level, but those levels give an indication about the average knowledge of the agents.

We first notice from figure 8.2a that, in the static case, the Hellinger distance varies even if no incorrect observation is sent. This behavior is counter-intuitive. Indeed, we would expect the Hellinger distance to decrease as the agents confirm their beliefs about the environment. The explanation to this variation is the smoothing step that has been implemented also for the static cases. Indeed, for both static and dynamic cases we used the same transition model and the same smoothing step. The idea behind was to consider that the system doesn't know that the environment is static and to observe its behavior in this situation. This situation may happen is real application. Let us consider for instance that the inhabitants of a house decide to go on vacation for several days. The environment is very likely to become static. It seems reasonable not to re-program the system if this one is able to deal with this new situation using the same model.

In the static case, the system manages to keep the belief states varying between level 2 and level 4 beliefs most of the time, which is very satisfactory. In the dynamic case the performance of the system are worse since the beliefs vary between level 1 and level 2 beliefs. We can clearly
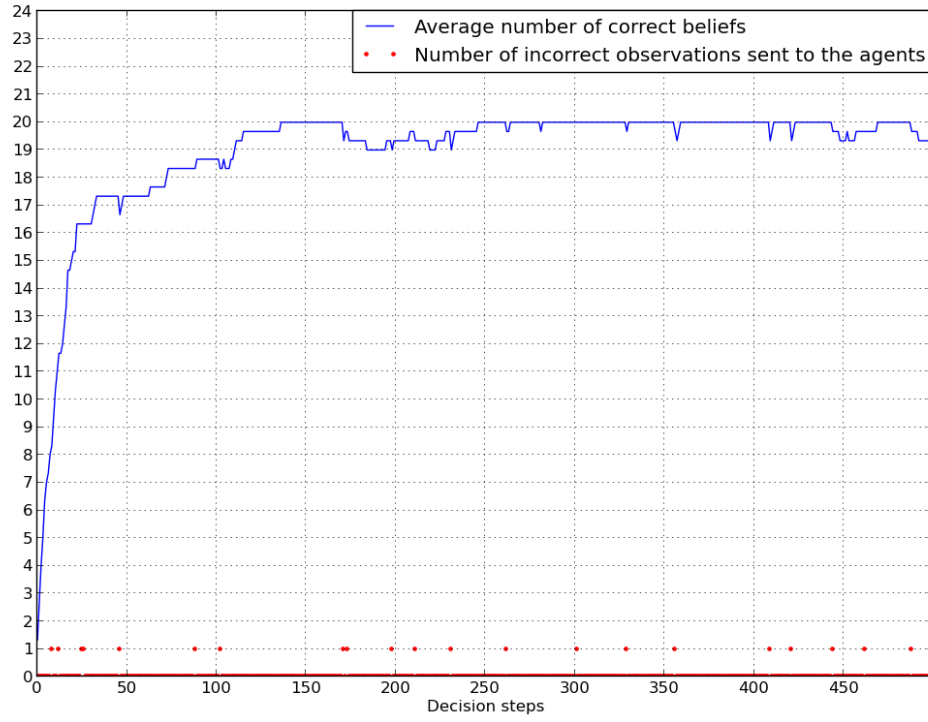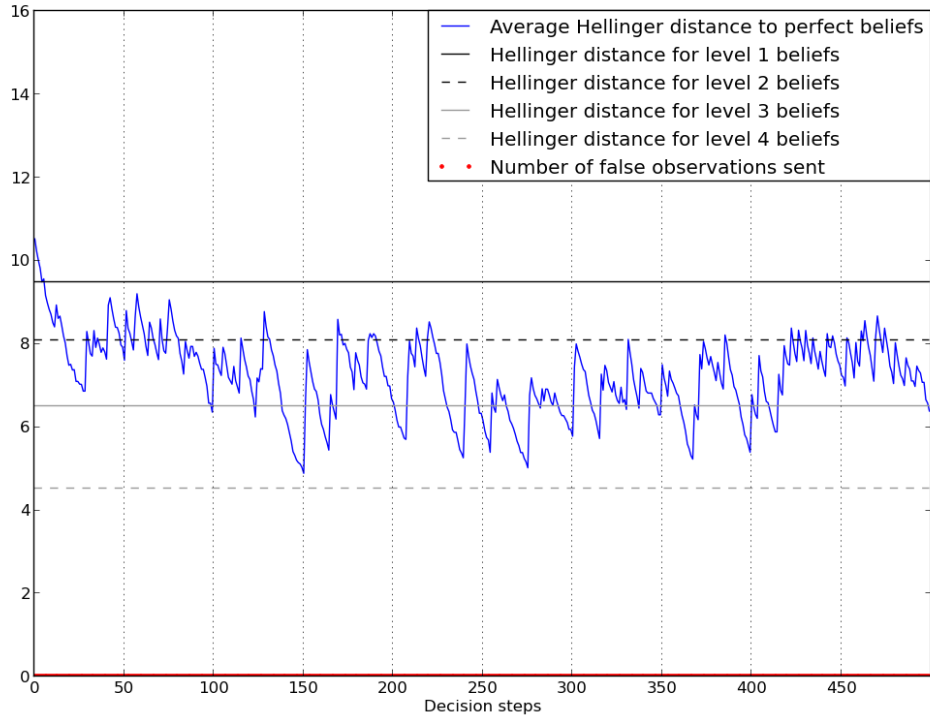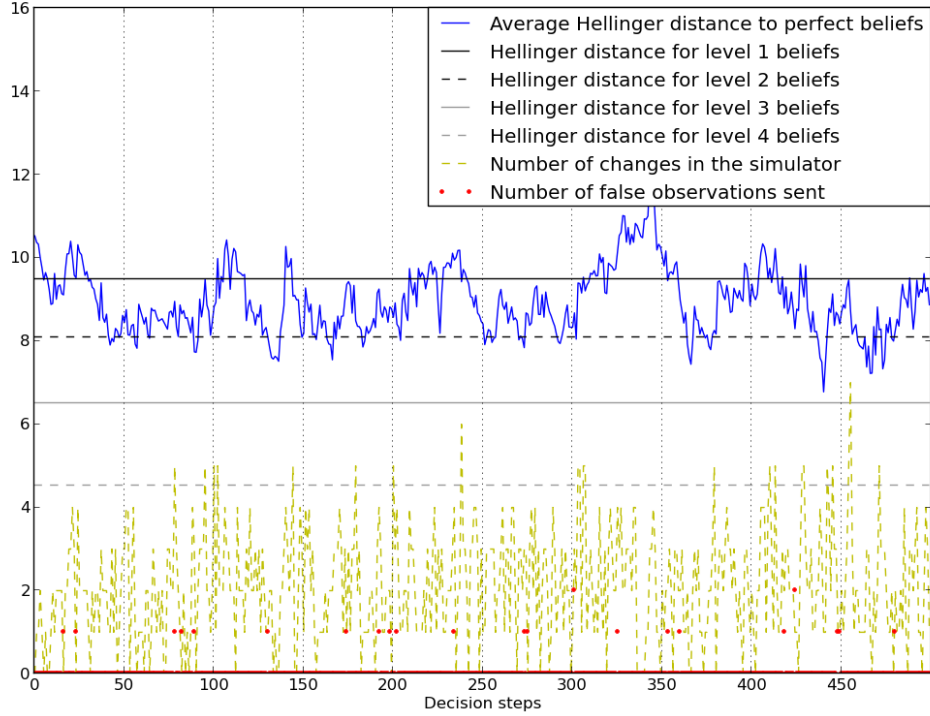
(a) Static case
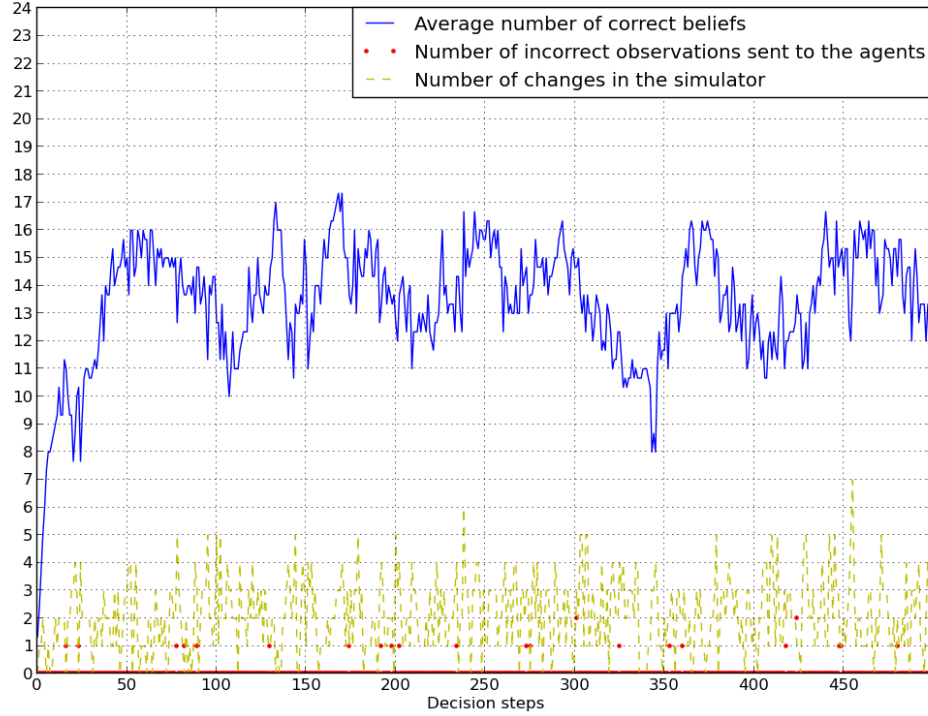
(b) Dynamic case

Figure 8.1: Average number of correct beliefs for the house environment

(a) Static case

(b) Dynamic case

Figure 8.2: Average Hellinger distance to the perfect belief state for the house environment

see that the quality of the beliefs decreases significantly after big changes (for instance around decision steps 240) and big amount of incorrect observations (for instance around decision step 300). We are not completely satisfied by the behavior of the system in the dynamic case for the house environment. Indeed, level 1 beliefs are discriminated beliefs (that is to say beliefs different that $(0.5, 0.5)$) but not discriminated enough to be reliable.

8.1.2   Evaluating the communication

The second purpose of the system is to be able to perform this mission while limiting its number of communications. In this section we evaluate the communication of the system. In MAPING , the role of the communication is to inform the other agents about received observations in order to overcome sensors limitations and to reduce the uncertainty in the system. If the agent in the system didn't communicate, the color-detection robots would have very good information about the **RoomOnFire** variables but would be completely ignorant about the **RoomEmpty** variable (and the opposite for the face-detection robots). The temperature-detection would be able to induce believes about both kind of variables but would be incapable of discriminating between them.

To evaluate the communication in the system, we recorded the number of communications performed by each agent and for each other agent. Results are presented on figure 8.3.
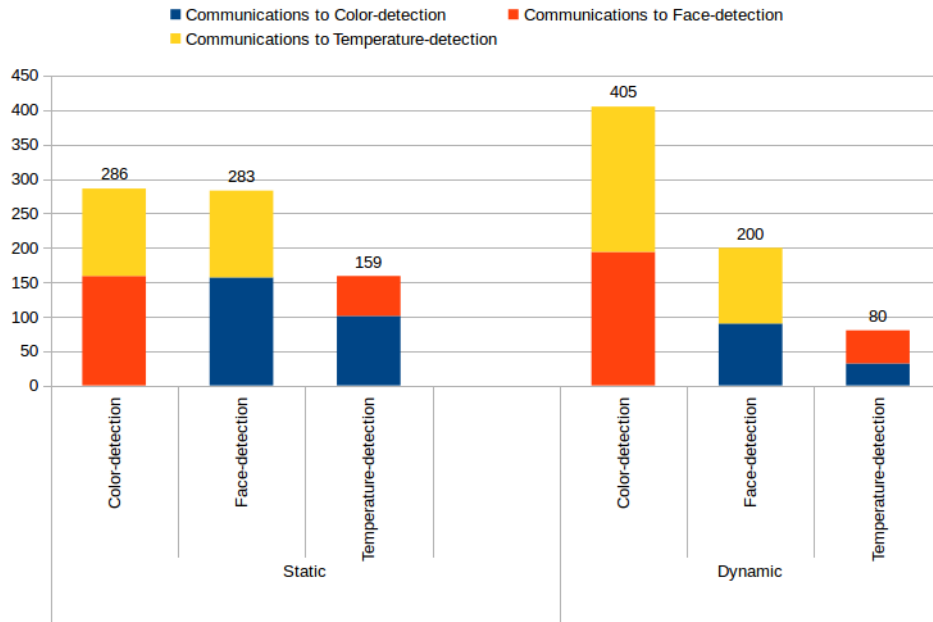
Figure 8.3: Number of communications per robot for the house environment

In the static case, the agents used the communication action 728 times, that is to say 243 times in average per agent. In the dynamic case, they used the communication action 685 times, that is to say 228 time in average per agent. This number are summarized in table 8.1. It is very surprising to see that the agents communicate more in the static case than in the dynamic case. We would expect the agents to communicate less in the static case. Indeed, the agents communicate to inform other agents that don't have discriminated beliefs or when their beliefs are incoherent with other agents' beliefs. The second case can only happen if an incorrect observation

	Static		Dynamic	
	Total	Average	Total	Average
House environment	728	243	685	228

Table 8.1: Total and average number of communication in the house environment in static and dynamic cases

is received or when the simulator changed. Since there is no change in the static case, the agents only have incoherent beliefs when they received an incorrect observation. It seems reasonable to assume that agents would communicate less in the static case than in the dynamic case. However in most of the environments, the difference between the amount of communications in the static and dynamic cases is not big enough to be significant and confirms this scenario.

We also notice that, in the dynamic case, the color-detection agent communicates significantly more than the two others This behavior has been encountered in all environments.

8.1.3 Evaluating the homogeneity of the beliefs

As mentioned in section 5.3.4, agents cannot know by definition if their belief states are correct regarding the state of the system. However if all agents share the same shape of belief state, it is more likely that they are correct. Since the system is fully decentralized, the agents don't know the exact belief state of other agents but have approximate representation of them that they update when they send and receive observations. It is therefore important that these representations are accurate so that the system can work properly. To measure the accuracy of the approximation, we used once more the Hellinger distance. For each pair of agents $< a_1, a_2 >$ we computed the Hellinger distance between agent a_1's approximation of agent a_2's beliefs and agent a_2's real beliefs and we tracked this measure among time. The results are presented in figure 8.4. To evaluate the accuracy of the approximation, we used the same system of level as in section 8.1.1. An approximation of level 4 is an approximation that differs by only 0.1 from the real belief.

Example 8.2 If we consider an agent that has the following belief

$$b_{a_2}^k = \big((0.7, 0.3)\big)$$

Then the belief

$$b_{a_1}^{a_2,k} = \big((0.8, 0.2)\big)$$

is a level 4 approximation of $b_{a_2}^k$

Figure 8.4 shows that the approximation of others' beliefs is very good. Indeed in both static and dynamic cases, the approximation remains under a level 4 approximation. This means that most of the approximations of the agents are equals to the real belief.

8.1.4 Evaluating the paths

The last criteria we want to evaluate is the path the agents use to explore the environment. A really efficient exploration should be able to travel across the environment in an organized way and no jump from a room to another room far away. Figure 8.5 presents the actions made by
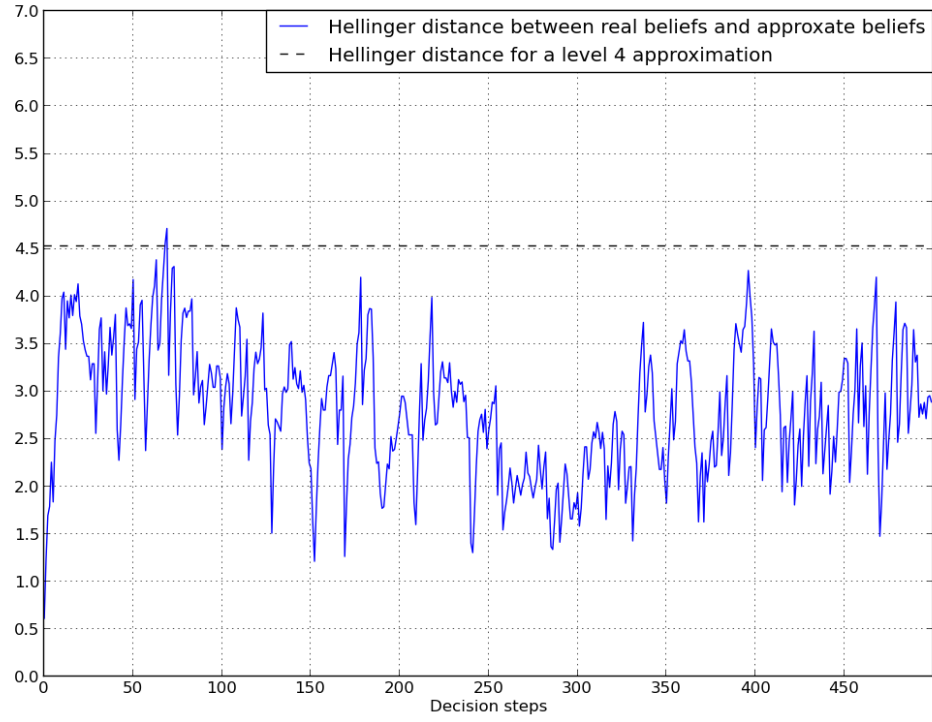
(a) Static case
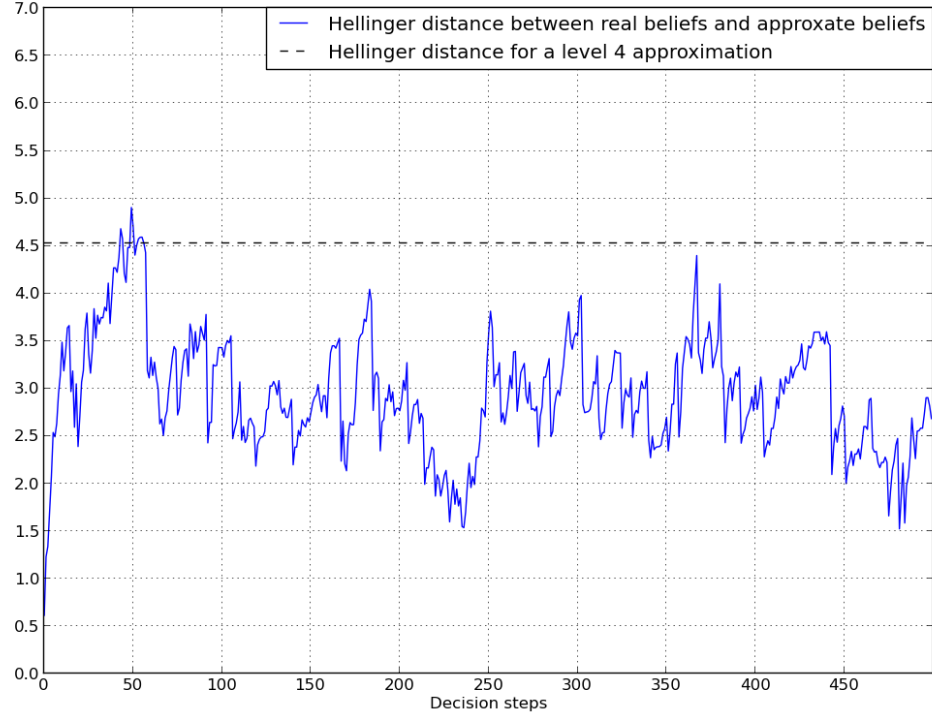
(b) Dynamic case

Figure 8.4: Average Hellinger distance to the real belief states for the house environment

each agent during the 15 first decisions steps and the room they were in. We added the graph of the environment as reminder.

From a general point of view, the agents travel among the environment following a reasonable path and don't jump between distant rooms. This expectation is confirmed by the average cost of the used path, that we recorded during the simulation. The color detection robot has an average cost of 2.11 per exploration action, the face detection has an average cost of 2.13 and the temperature detection has an average cost of 2.31.

8.1.5 Analysis

In this section we evaluated the performance of the system in the house environment. The results show that the system is able to explore its environment and gather correct beliefs while limiting the number of communications. However, the beliefs reached in the dynamic cases are not accurate enough to be completely reliable. The very high number of communications in both static and dynamic cases could be partially responsible for this lack of reliability. Indeed, since the communication is cheap, the agents communicate each time they notice what they believe is a change, without waiting for their beliefs to be more reliable. Therefore, changes are spread very quickly, but incorrect observations are also spread quickly, which leads to a decrease in the accuracy of the beliefs.

The approximations made by each agent about other agents' beliefs are very accurate, which gives some guarantee about the correctness of all the belief states and force the agents to explore or communicate again when the accuracy decreases. The paths explored showed interesting behaviors and strategies to cover the whole area. In the next sections, we present the same evaluations for the other environments. We will briefly comment the parts that are similar to what we observed in the house environment and stress out the differences.

8.2 Constraints in the navigation: the one-way environment

The one-way environment presents a particular topology, in which some transitions between rooms can only be done in one way. This constrained environment can be encounter in building with fire doors or with some doors that a robot can push but not pull. This constrained environment is expected to make the exploration more complex and is used to evaluate the adaptability of MAPING .

8.2.1 Evaluating the exploration efficiency

Figure 8.8 presents the evolution of the number of correct states and figure 8.7 presents the evolution of the Hellinger distance between the agents' belief states and the perfect belief state in the dynamic case.

The results obtained are better than those obtained in the house environment since number of correct belief states in the dynanic case is around 75%, which is extremely satisfactory.

The variation of the Hellinger distance to the perfect belief state is also better than in the house environment: in the dynamic case, beliefs vary between level 1 and level 3 and are level 2 in averadge, that is to say $(0.7, 0.3)$ or $(0.3, 0.7)$. Level 2 beliefs, though not the best, are reliable enough to be considered satisfactory.
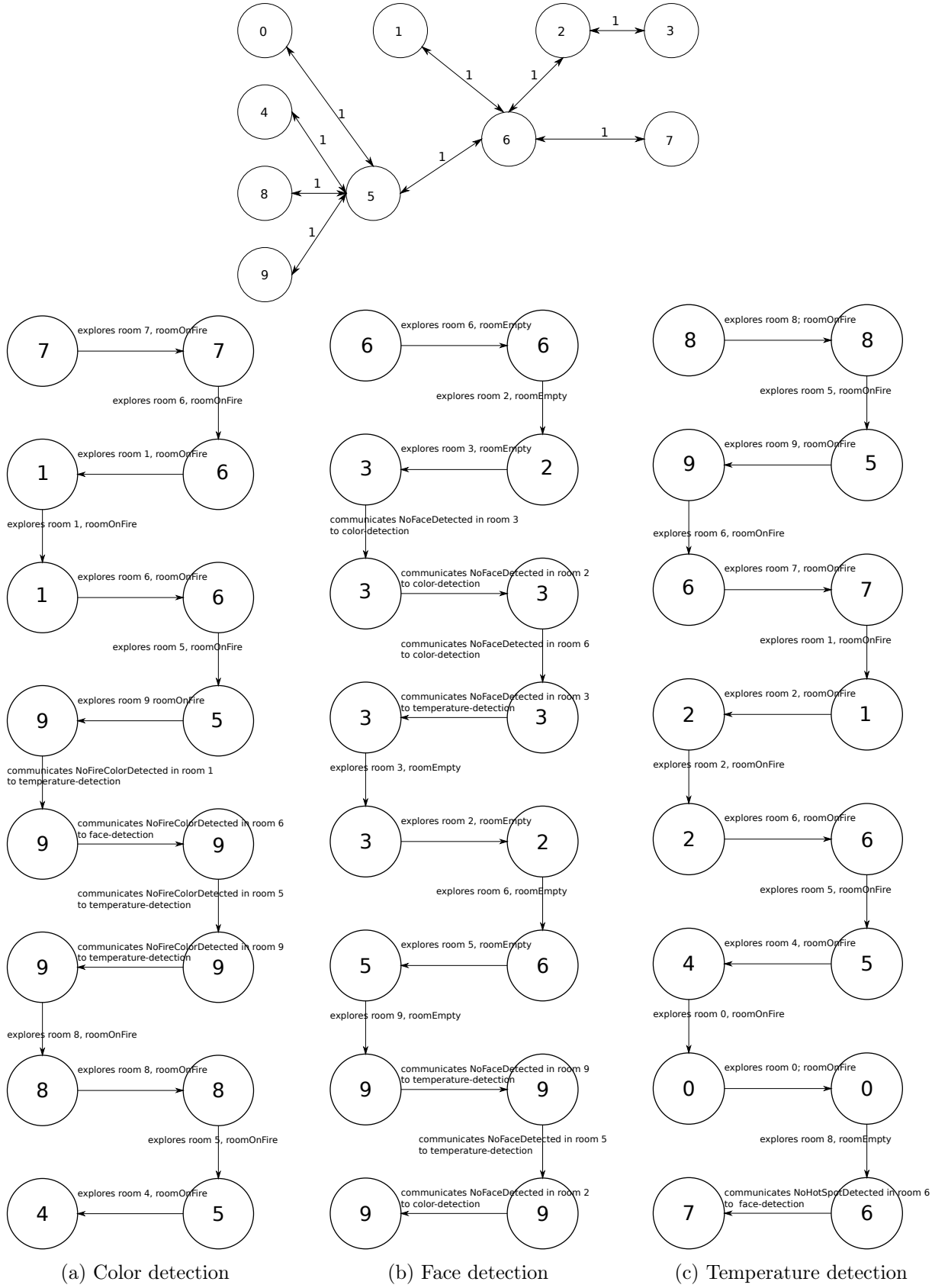
Figure 8.5: Actions performed and path traveled by the three robots during the 15 first iterations in the house environment
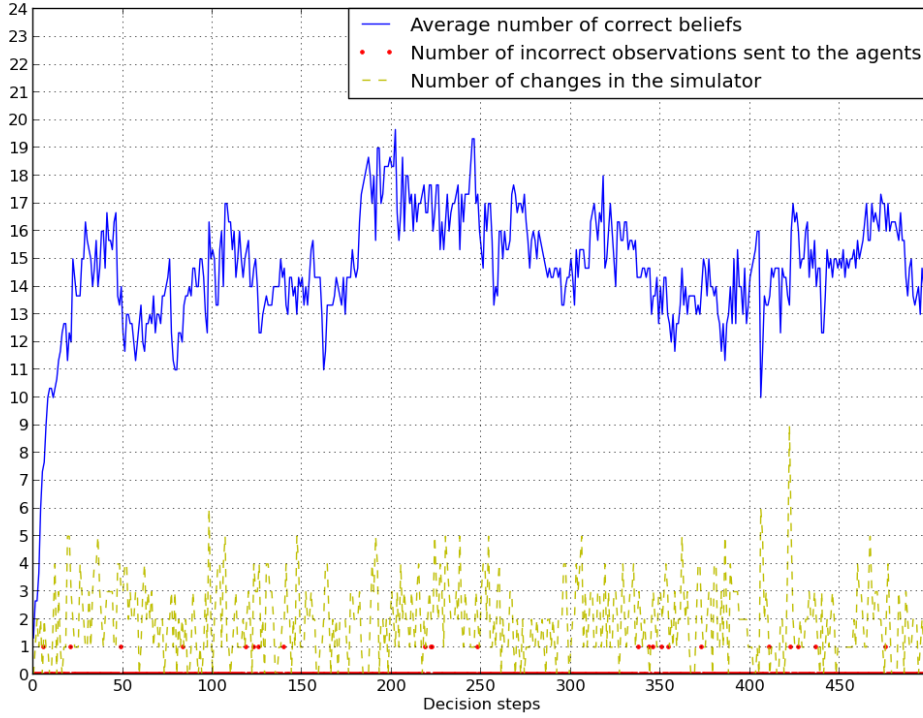
Figure 8.6: Average number of correct beliefs for the one-way environment in the dynamic case

|	Static		Dynamic	
	Total	Average	Total	Average
House environment	728	243	685	228
One-way environment	582	194	664	221

Table 8.2: Total and average number of communication in different environments in static and dynamic cases

8.2.2   Evaluating the communication

Figure 8.8 shows the number of communications sent for each pair emitter-receiver. In total, the agents used a communicate action 582 times in the static case and 664 time in the dynamic case. In average, each agent communicated 194 times in the static case and 221 in the dynamic case. These numbers are summarized and compared to previous environments in table 8.2. The agents communicated less in this environment than they did in the house environment. With this lower communication, the results are better than in the house environment. This tends to strengthen our hypothesis that too much communication is detrimental for the system. Once again, the color-detection robots communicated much more than the other agents. It even spent most of its decision steps to communicate.
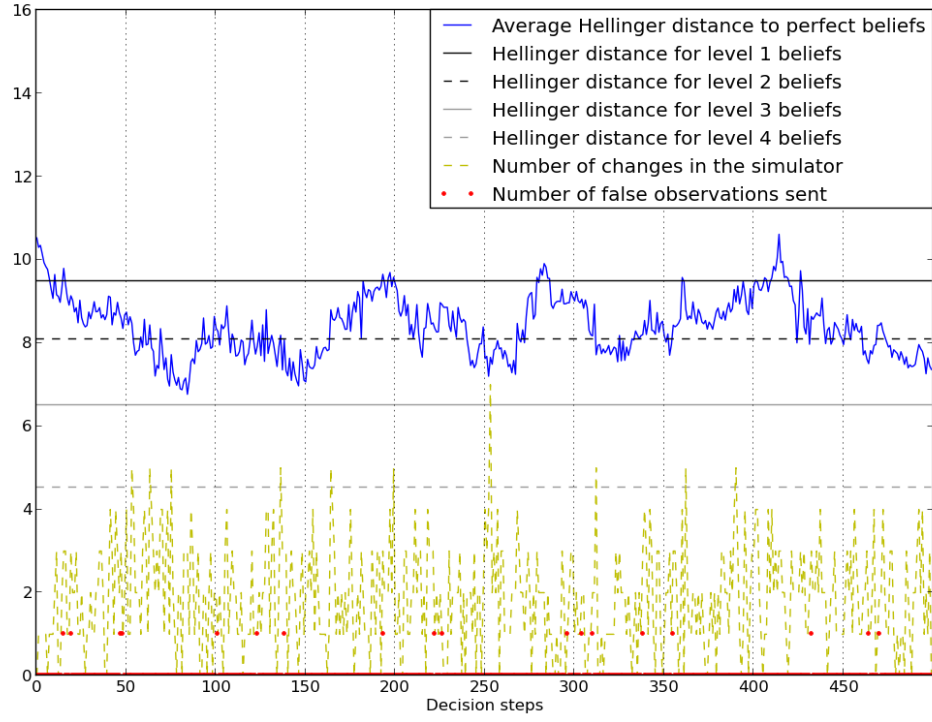
Figure 8.7: Average Hellinger distance to the perfect belief state for the one-way environment in the dynamic case
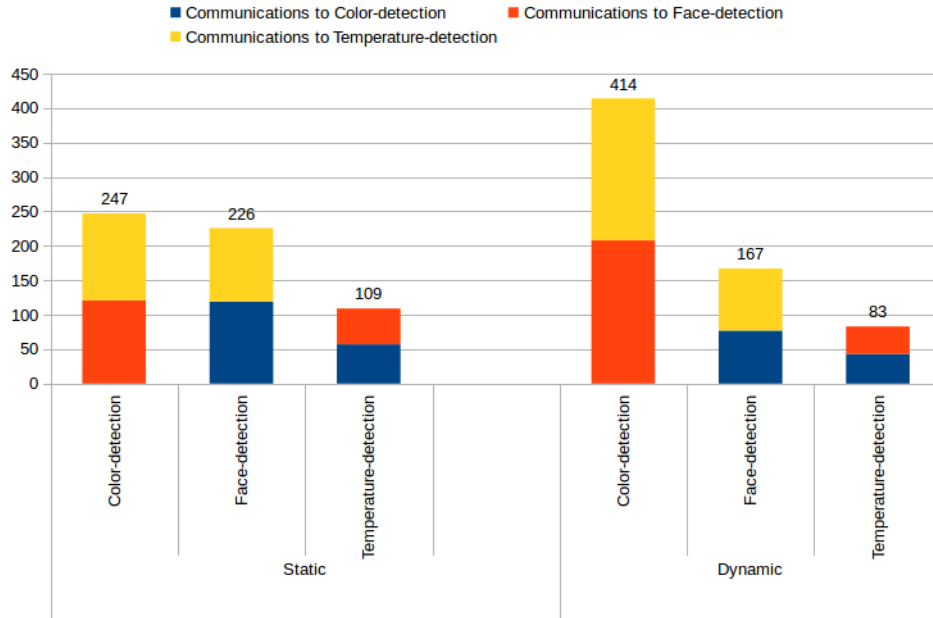
Figure 8.8: Number of communications per robot for the one-way environment

8.2.3 Evaluating the homogeneity of the beliefs

Figure 8.9 presents the evolution of the approximations of agents about other agents' beliefs. Once more, the approximation is usually better than a level 4 approximation, which means that
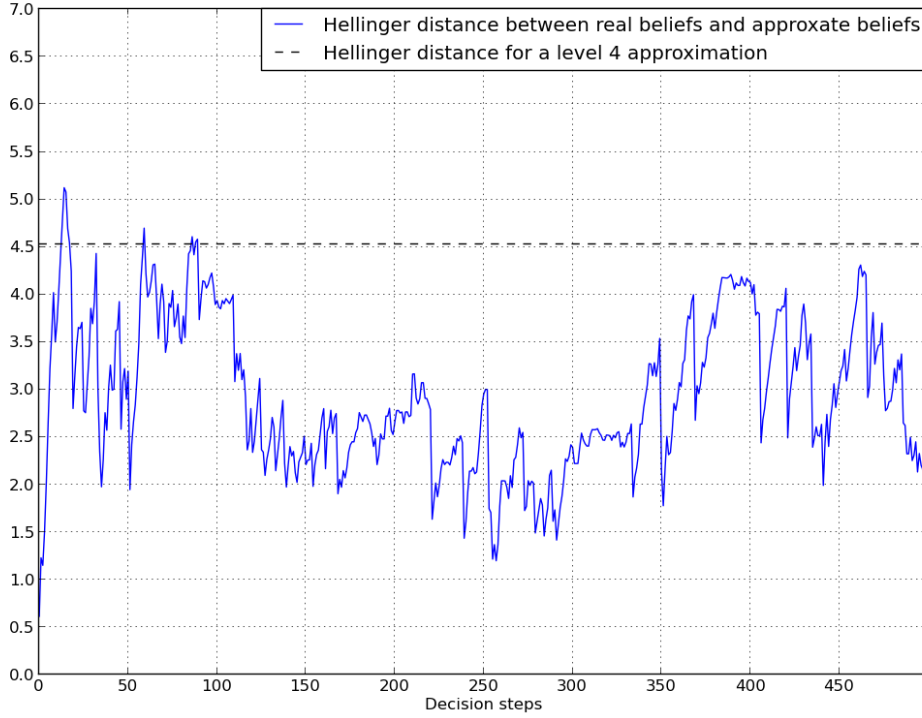
Figure 8.9: Average Hellinger distance to the real belief states for the one-way environment in the dynamic case

the agents approximations are mostly equals to the real beliefs.

8.2.4 Evaluating the paths

Figure 8.10 presents the actions performed and the path travels by the three robot during the 15 first iterations. In this environment, the color detection robot has an average cost of 2.29 per exploration action, the face detection has an average cost of 1.88 and the temperature detection has an average cost of 1.73. Once again, the paths chose by the agents are coherent. We can also observe a behavior in this environments that repeats several time in the simulation: the agents seem to get stuck for several decision steps in a given rooms and performs several times the explore action in this room. This happens for instance in room 5 for the temperature-detection robot. This also happen in other parts of the simulation and in other environments. We identified two reasons to this behavior, that can happen together. The first reason is the possibility to receive contradictory observations, due to a sensor failure or a change in the environment. It is not rare that an agent explores two times the same zone to have a very accurate belief state. If the agent receives two different observations from its exploration, it will explore again and again until it reaches an accurate belief. The second reason is the smoothing function. Let us remind how the smoothing step is performed: after each decision step, each variable has a probability to be smoothed depending on the transition function. This is done to ensure that variables that

(a) Color detection (b) Face detection (c) Temperature detection
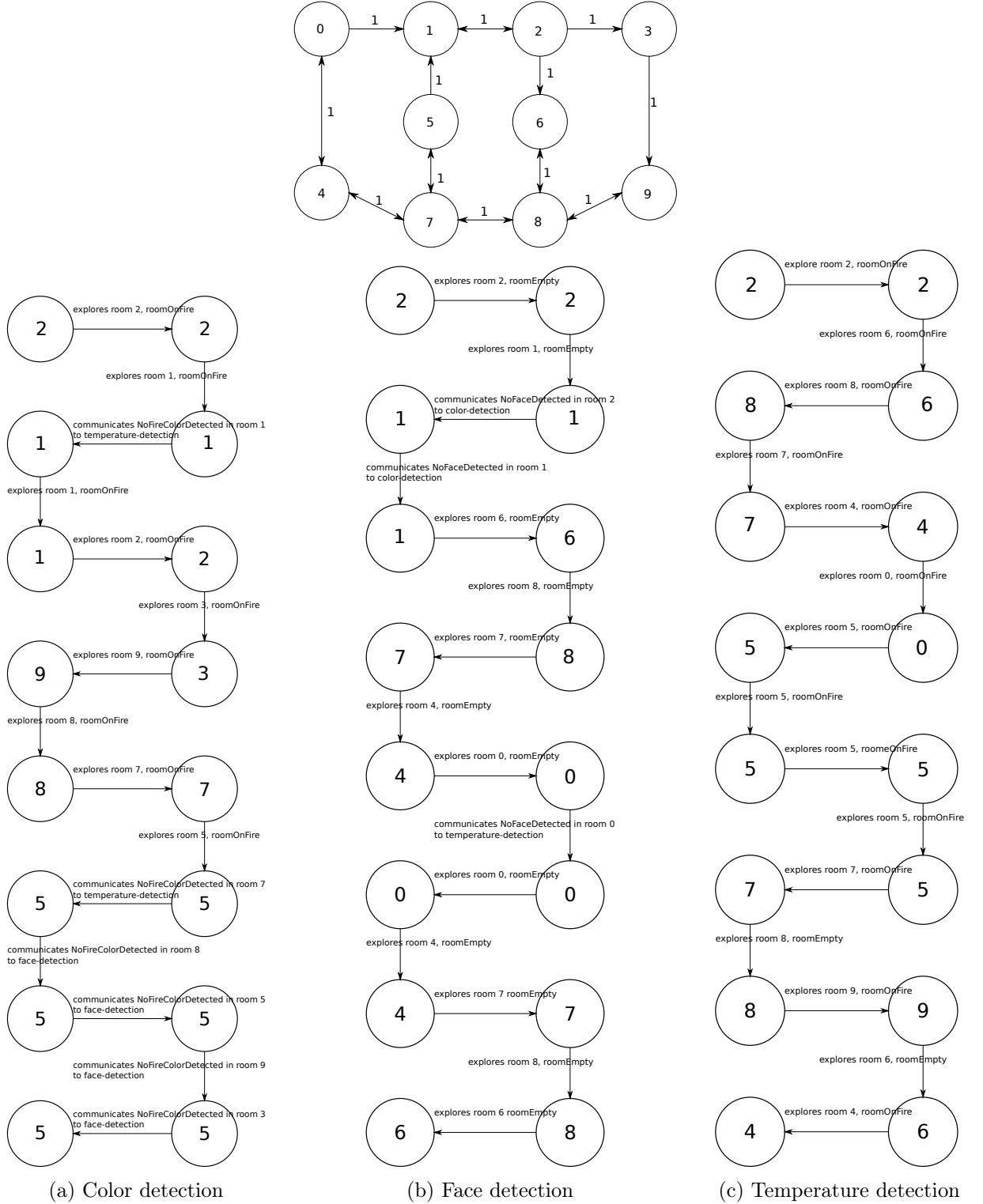
Figure 8.10: Actions performed and path traveled by the three robots during the 15 first iterations in the one-way environment

may change often are smoothed more often than variables that are almost static. A consequence of this is that a variable can be smoothed just after the agent explored this variable. In this case, the agent's belief precision will not be improved enough to be satisfactory and the agent will explore again the area. Both reasons can happen in any room. However, the behavior is more often observed at the beginning of the simulation. This is due to the cost of travelling in another room. This is explained in example 8.3.

Example 8.3 The temperature-detection robot travels from room 0 to room 5 and explores room 5. It receives the observation **NoHotSpotDetected**. Its belief about room 5 is now satisfactory: $(0.7, 0.3)(0.3, 0.7)$. However, it has already explored room 7 previously and received an observation about room 1 from the color-detection robot. Its beliefs about those rooms are so satisfactory and the next interesting room to explore would be room 3, which involves a cost of 3. The decision process states that it is more interesting to confirm the beliefs in room 5 by exploring again than paying the cost and going in one of the other rooms. The agent received contradictory observations from the simulator and so explored again until it reaches a reliable beliefs. Then it can afford to pay a cost for going in room 3.

8.2.5 Analysis

The results in this environment are more satisfactory than in the previous environments. However the communication rate is still very high and the color-detection robot spent most of its time to communicate instead of exploring. This behavior can damage the quality of the beliefs and should be compared to those obtained with higher communication costs.

The analysis of the paths traveled shows that the agents correctly explore the environment and checks their exploration when they receives observations that contradict their current beliefs.

8.3   Costly transitions: the outdoor environment

The outdoor environment is based on real topographic maps from the Alpes, near the city of Chatel. This environment has been designed to observe the behavior of the system in areas where moving from a "room" (in this case we still use the denomination room even if those rooms are only virtual) to another can be very costly to the robots, due for instance to difficult ground.

8.3.1   Evaluating the exploration efficiency

Figure 8.11 presents the evolution of the number of correct states and figure 8.12 presents the evolution of the Hellinger distance between the agents' belief states and the perfect belief state in the outdoor environment.
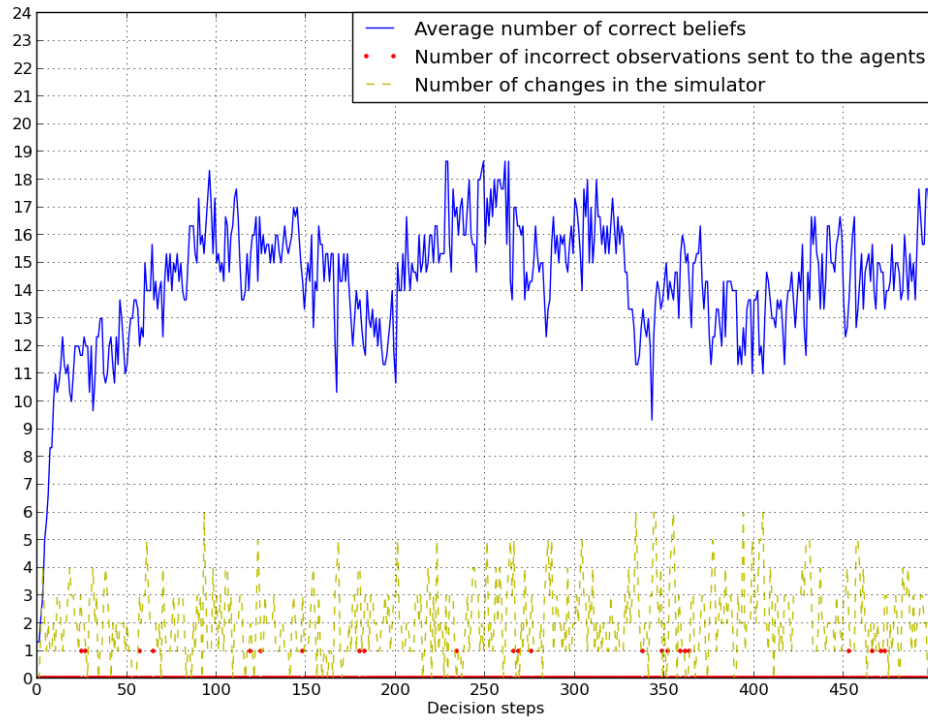
Figure 8.11: Average number of correct beliefs for the outdoor environment in the dynamic case

The system manages to keep around 70% of correct beliefs and an accuracy varying between level 1 and level 3 beliefs. We notice than the three times the beliefs were worse than level 1 beliefs (around decision steps 180, 320 and 390) correspond to three sequences of incorrect observations. But each time the system managed to improve its beliefs up to level 2 beliefs in less than 30 decision steps. Therefore, even if those results are slightly less good than in the one-way environment, they are still better than in the house environment and quite satisfactory.

8.3.2   Evaluating the communication

Figure 8.13 presents the number of communications sent by the agents in the static and dynamic cases. In this environment, the agents communicated 751 times in the static case against 754 times in the dynamic case. The average number of communications for each agent is 250 in
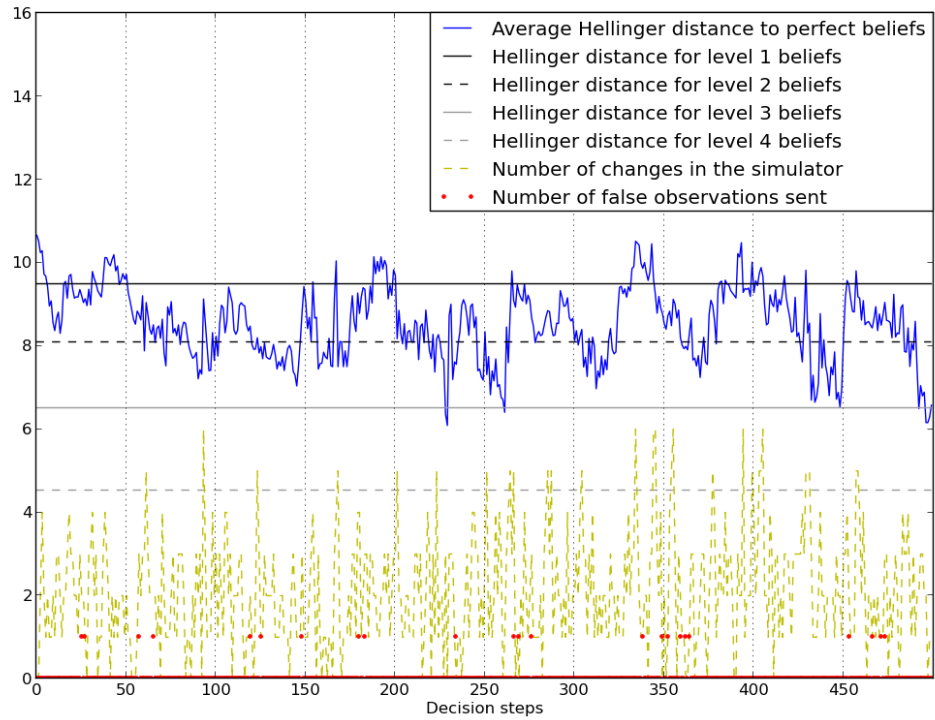
Figure 8.12: Average Hellinger distance to the perfect belief state for the outdoor environment in the dynamic case
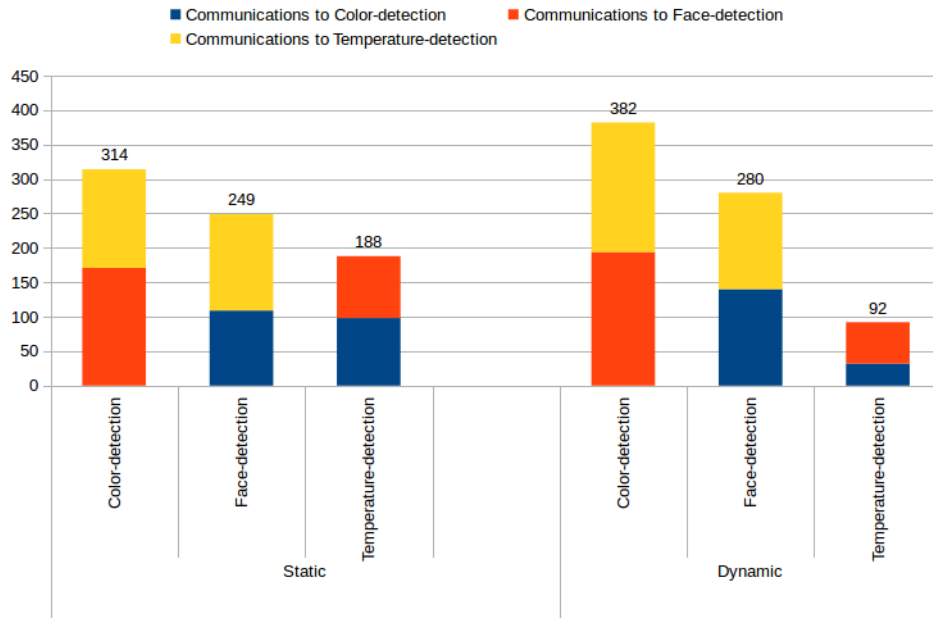
Figure 8.13: Number of communications per robot for the outdoor environment

	Static		Dynamic	
	Total	Average	Total	Average
House environment	728	243	685	228
One-way environment	582	194	664	221
Outdoor environment	751	250	754	251

Table 8.3: Total and average number of communication in different environments in static and dynamic cases

the static case and 251 in the dynamic case. These numbers are summarized and compared to previous environments in table 8.3.

The agents communicate significantly more than in the previous environments. This can be explained by the costly transitions: in this environment, communicating is usually as expensive, even sometimes cheaper, than moving from a room to another. Then the decision processes consider communication as a good alternative to spread beliefs and avoid unnecessary costs involved by exploration. It seems that unlike in the house environment, this high level of communication didn't damage too much the quality of the beliefs. We noticed that in the outdoor environment, the communication is slightly better distributed among the agents. In the home environment, the color-detection robots communicated twice as much as the face-detection and 4 times as much as the temperature-detection robot. Here, the color-detection communicates "only" 1.3 times as much than the face-detection and 4 times as much as the temperature-detection robot. This distribution may explain why the beliefs are not too much impacted by the high communication rate.

Concerning the homogeneity of the beliefs, the homogeneity is also very good in this environment, as shown on Figure 8.14.

8.3.3 Evaluating the paths

Figure 8.15 presents the actions performed and the path travelled by the three robot during the 15 first iterations. In this environment, the color detection robot has an average cost of 2.82 per exploration action, the face detection has an average cost of 2.59 and the temperature detection has an average cost of 3.05. Those average costs are unsurprisingly higher than in the previous environment: the fact than more than half of the transitions have a cost greater than 1 is directly responsible for this increase. We can notice that the paths with the lowest costs are preferred at the beginning of the exploration: all areas can be explored and give a lot of information, it is therefore more interesting to travel across low-cost paths since the ratio information gain / cost is higher. The paths are again coherent with the environment. Only the transition of the color-detection robot from zone 6 to zone 0 could seem incoherent since too expensive. However, when analyzing the graph, we notice that there are only two transitions less expensive than going to zone 0 which are going to zone 8 and going to zone 3. However both zones have been explored previously by the color-detection robot. Therefore, the interesting zones to explore all have a cost of 2, and the algorithm chose among the most interesting zones and selected 0 as the next one to explore. We notice in this environment that the communications are grouped together. We believe this is due to the same reason than the agents exploring again the same room.
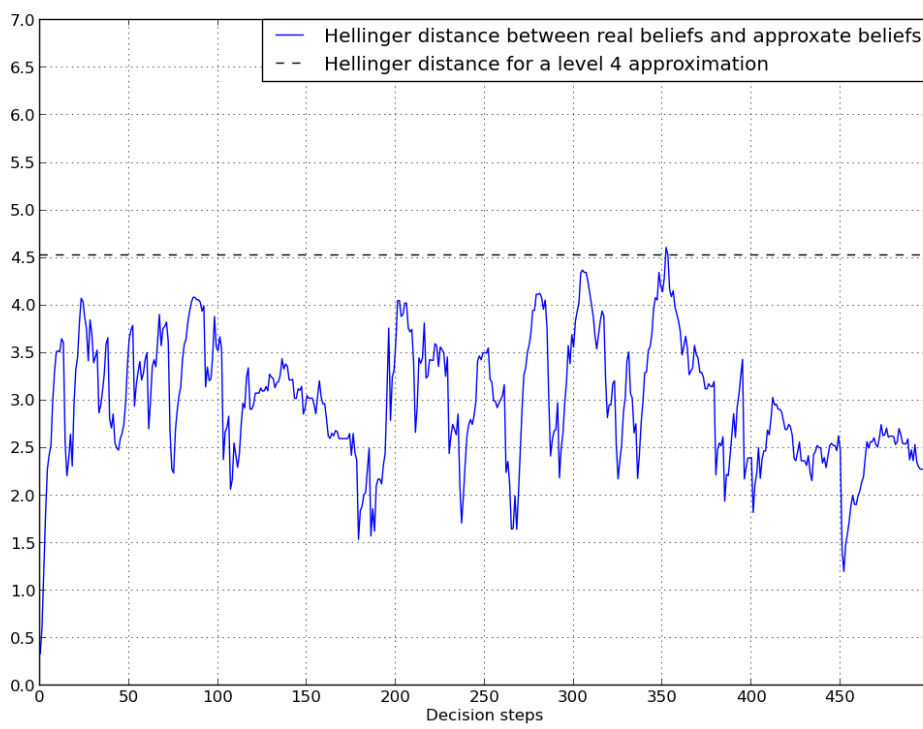
Figure 8.14: Average Hellinger distance to the real belief states for the outdoor environment in the dynamic case
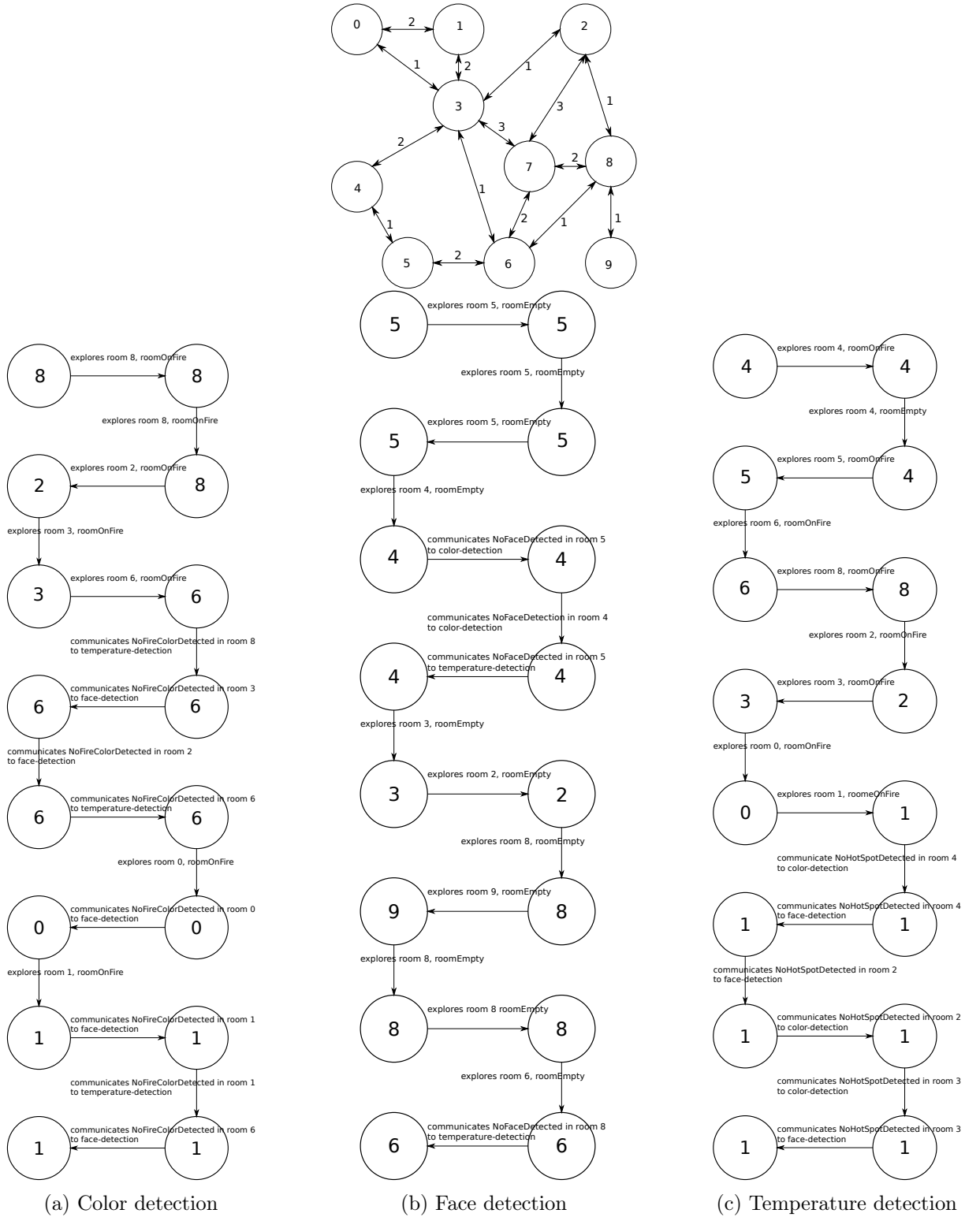
(a) Color detection

(b) Face detection

(c) Temperature detection

Figure 8.15: Actions performed and path traveled by the three robots during the 15 first iterations in the outdoor environment

Example 8.4 The color-detection robot moves from room 3 to room 6 and explores room 6. Its beliefs about room 6 are now satisfactory. Since it already explored rooms 3 and 8, the next interesting rooms would be 4 or 7 which both include a cost of 2. The decision process states that it is more interesting to communicates to inform the other agents of the previous exploration than to move to those rooms.

8.3.4 Analysis

We evaluated in this section the impact of a costly environment for our system. The results tend to show that MAPING is able to adapt to this kind of situation by increasing the number of communication to trade-off the costly exploration and maintain a good belief state. The very high communication rate seems not to impact to much the quality of the beliefs as it did for the house environment. This is probably due to the fact that the communication is slightly better distributed among the agents.

8.4   Scalability: the Élysée Palace environment

The Élysée Palace environment has been chosen to prove the scalability of MAPING in a house-like environment. In this section we will present again the static and the dynamic case. Indeed, the increase of the number of rooms changes a bit the behavior of the system.

8.4.1   Evaluating the exploration efficiency

Figure 8.16 presents the evolution of the number of correct beliefs. We first notice that, as expected, the system is slower to get correct beliefs, the size of the environment being much bigger. Then, we notice that even in the static case, the system don't manage to get all the variables correct, that is to say 56 correct variables. However, the simulation logs show that the robots explore all rooms at least once in less than 50 decision steps and have discriminated beliefs (that is to say beliefs other than $(0.5, 0.5)$) for all variables in less than 100 decision steps thanks to communications. We explain the fact that the maximum amount of correct variable is never reached by the incorrect observations sent by the simulator. Since the environment is big, when the simulator send an incorrect observation, the incorrect beliefs resulting from this incorrect observation is kept for a long time before the variable is explored again or before another agent send a contradictory observation. This is true especially at the beginning of the simulation since incorrect observations cannot contradict prior beliefs and so the corresponding variable will not be explored again by the robot.

Despite this non-optimal result in the static case, we observe that the results in the dynamic case are quite interesting since the system manage to keep a good amount of correct beliefs (about 60%). It is obvious that the problem of incorrect observations in this case may be traded off by environment changes: an observation that was incorrect when it was sent can become correct when the environment changes and the robot's beliefs will be improved even if no new and correct observation has been received. One could say that the system is maybe not that efficient but the changes in the environment give the illusion that it is. However, one should not forget that this impact is also true when an observation has been sent correct and becomes incorrect following a change in this environment. In this case, the robot's beliefs are degraded until it received a new correct observation. This second case appears in average more often that the first one because the simulator sends the correct observation 98% of the time. Therefore we can conclude that the performance of the system are not illusory and that the system really manage to keep a good amount of correct beliefs. The results obtained in the dynamic case are so quite good, even if the system is less efficient than in smaller environments.

It would be interesting to evaluate the results of a bigger system (4, 5 or more agents) in this environment. Unfortunately we were not able to do so due to computational issues. Indeed, the size of the belief states space increase exponentially with the number of agents. For instance, using our discretization with 11 elements, each sub-POMDP to solve would have 214358881 states for 4 agents and 25937424601 for 5 agents. This was impossible to solve using our algorithm and our computing resources.

Figure 8.17 presents the evolution of the Hellinger distance to the perfect belief state. Those results are coherent with the results of figure 8.16: the beliefs varies mostly between level 1 and level 2 in both cases, which means that the exploration is less efficient than in small environments. However, the system still manage to get correct discriminated beliefs for most of the variables, which is a encouraging result. We strongly believe that a system with more robots would be able to maintain a better belief state.
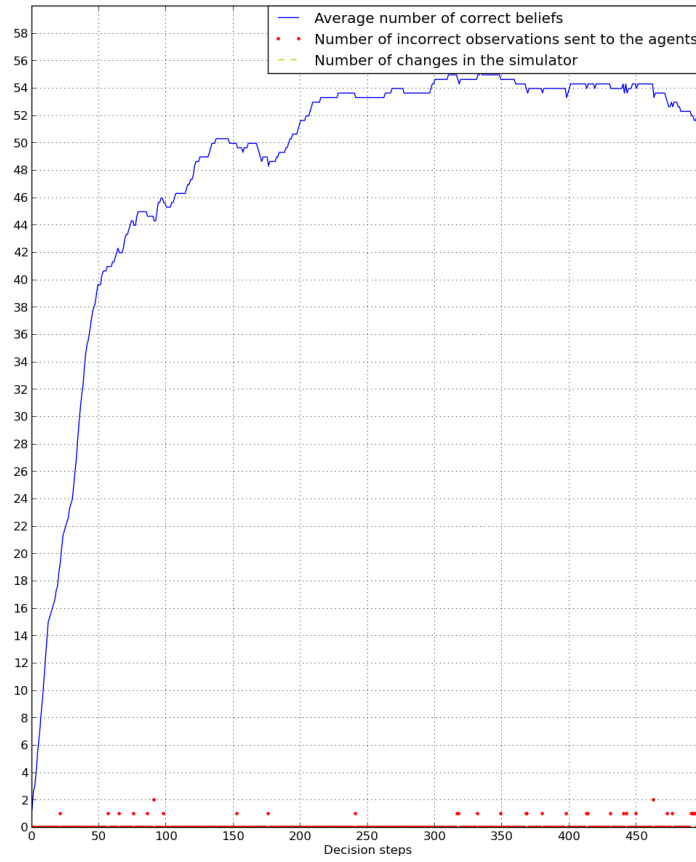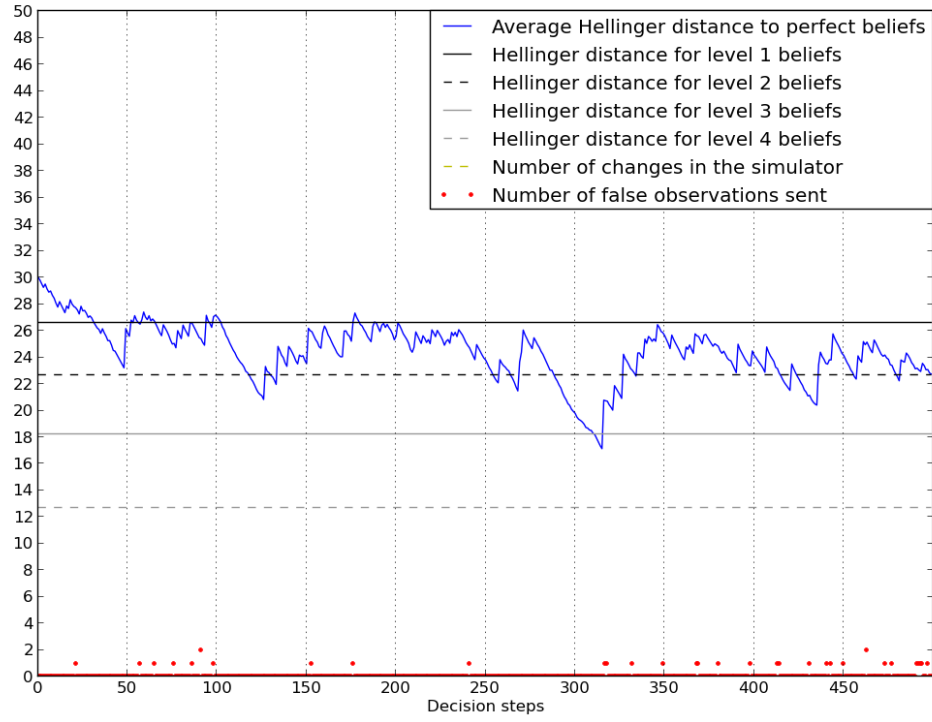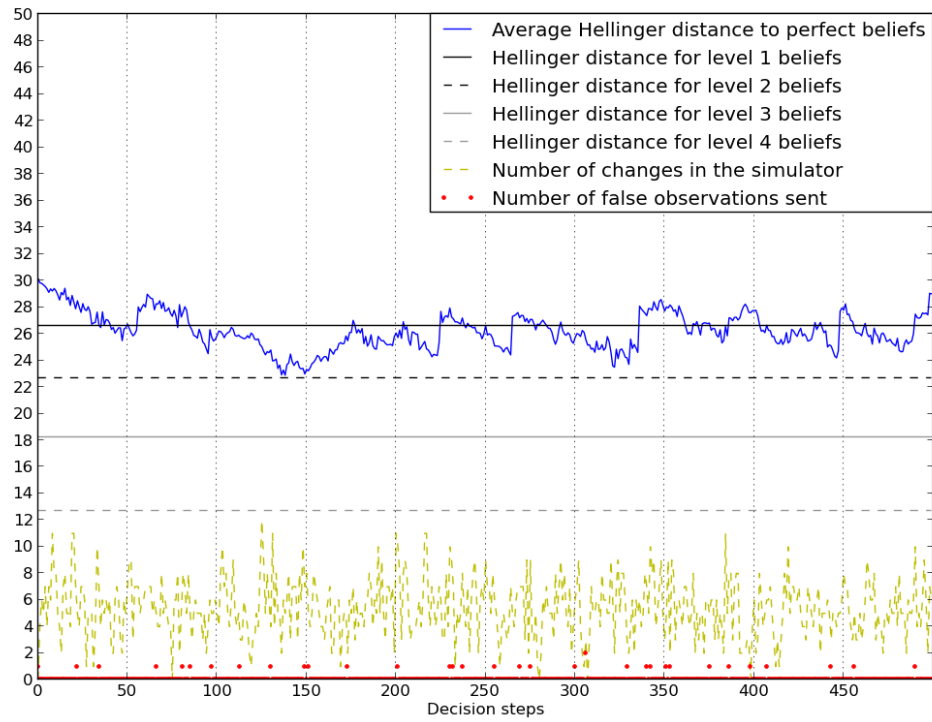
(a) Static case
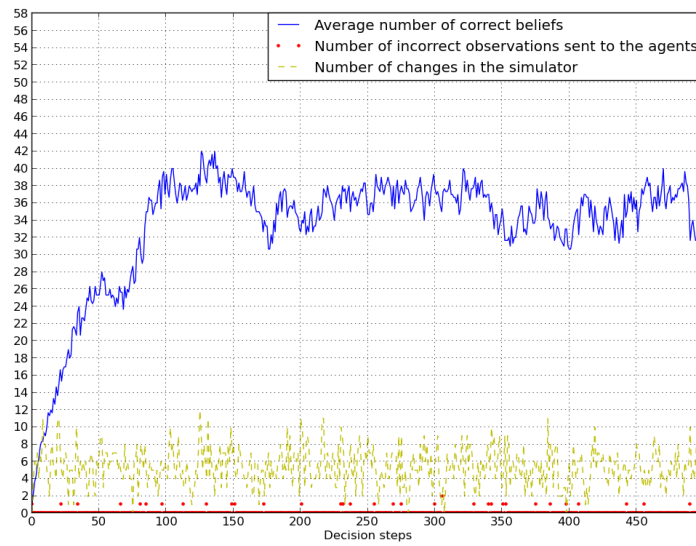
(b) Dynamic case

Figure 8.16: Average number of correct beliefs for the Élysée Palace environment in the dynamic case

(a) Static case

(b) Dynamic case

Figure 8.17: Average Hellinger distance to the perfect belief state for the Élysée Palace environment

	Static		Dynamic	
	Total	Average	Total	Average
House environment	728	243	685	228
One-way environment	582	194	664	221
Outdoor environment	751	250	754	251
Élysée Palace environment	487	162	496	165

Table 8.4: Total and average number of communication in different environments in static and dynamic cases

8.4.2 Evaluating the communication

Figure 8.18 presents the number of communications sent by the agents in the static and dynamic cases. In the static case, the agents used the communicate action 487 times, that is to say 162
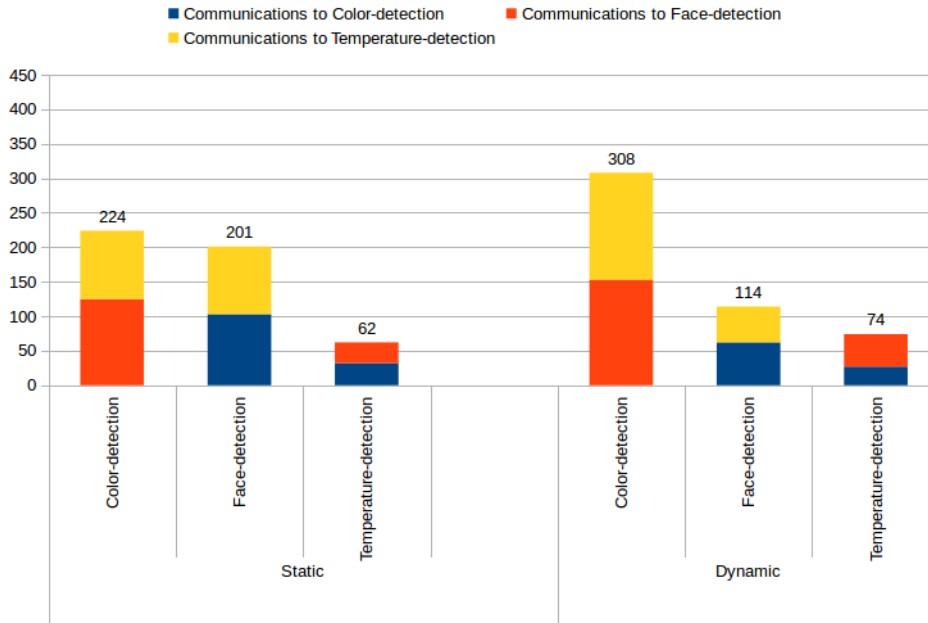
Figure 8.18: Number of communications per robot for the Élysée Palace environment

times per agent in average. In the dynamic case, they used the communicate action 496 times, that is to say 165 times per agent in average. Those numbered are summarized and compared to previous environments in table 8.4

Those numbers are much lower than in any other environments. We believe this is due to the size of the environment. Indeed, since the agents need to reach a certain level of belief for the communication to be interesting and their beliefs are smoothed, they will spend more time to explore the environment and will reach this level less often. Once again, it would be interesting to compare this behavior with a system containing more agents.

8.4.3 Evaluating the homogeneity of the beliefs

Figure 8.19 presents the variation of the Hellinger distance between agents' approximation of beliefs and the real beliefs in the static and the dynamic cases. Due to the low rate of communication, we could expect the approximations to be less good than in previous environment. However this doesn't seem to be the case since the Hellinger distance is still far lower to the Hellinger distance obtained with level 4 approximations. We explain these results by the fact that a given room is explored less often by the agents, and they will be more likely to keep the same beliefs for a long time before changing them. Therefore, as soon as an agent a_1 communicates once to an agent a_2, agent a_2 has a good approximation of agent a_1's beliefs and this approximation will stay good until agent a_1 explores again the room.

8.4.4 Evaluating the paths

Figure 8.20 presents the actions performed and the path travels by the three robot during the 14 first iterations. In this environment, the color detection robot has an average cost of 3.35 per exploration action, the face detection has an average cost of 3.77 and the temperature detection has an average cost of 3.73. Those costs are the highest observed among all environments. This is not so surprising due to the size of the environment: the agents need to travel farther to find a room that can give them interesting information.

The paths in the Élysée Palace environment show much more exploration than in previous environments, which once again was expecting. Indeed, at the beginning of the simulation the agents need to build their beliefs and so explore the environment. We also observe a very high tendency for the agents to follow the same path. Though less obvious, this behavior was also present in other environments and is explained in section 8.5.3.

8.4.5 Analysis

Our system with 3 agents is able to scale to bigger environments and is able to handle discriminated beliefs for most of the variables. Though this discrimination is not high enough and the accuracy of the belief state is not satisfactory. Tests with more agents would be mandatory to analyze deeper the capacity of MAPING in big environment. We believe that more agents would be able to maintain a satisfactory belief state. However it has been impossible for us to compute the sub-policies with a higher number of agents. Indeed, increasing the number of agents increases exponentially the number of belief states in the Beliefs-MDP used to solve the MAPING decision process and our computing resources were not enough to handle this number. A possible solution would have been to reduce the size of the discretization and to use only 5 elements instead of 11 (see table 5.1 in section 5.4). However we would need more time to perform the experiments with this new parameters.
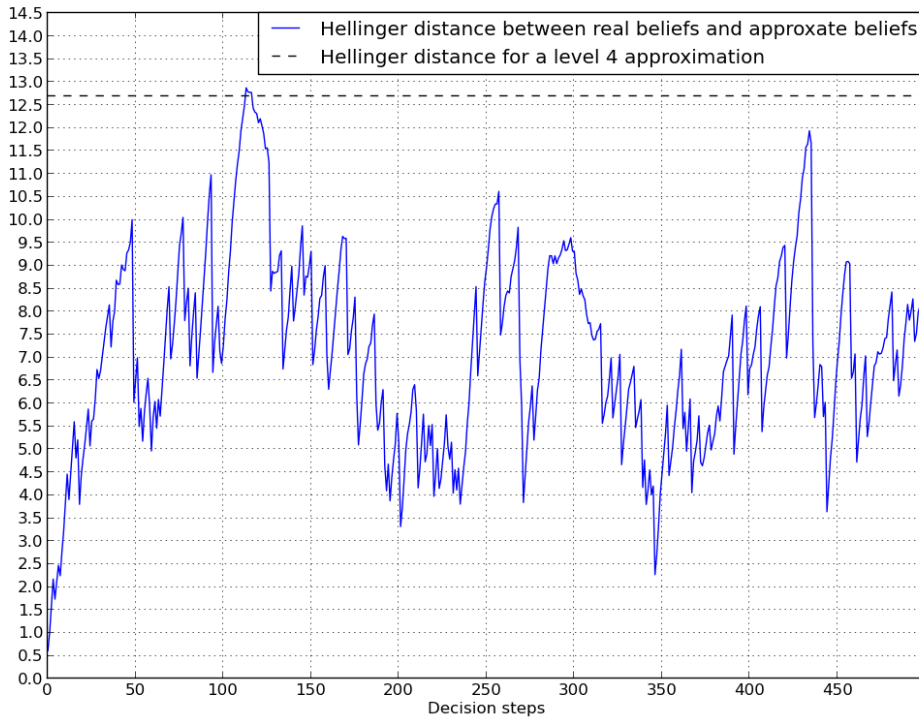
(a) Static case
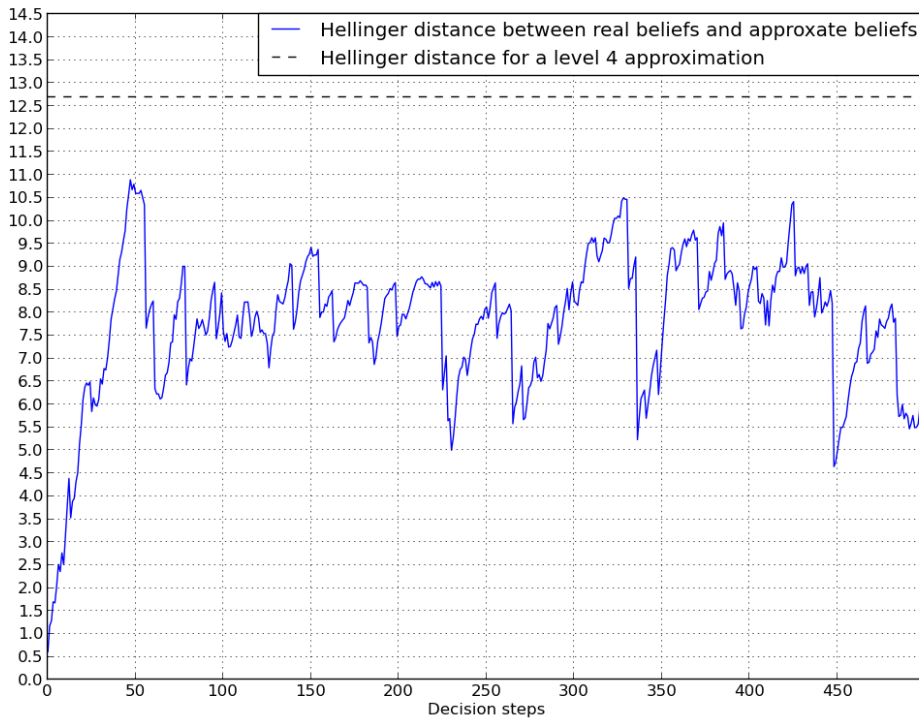
(b) Dynamic case

Figure 8.19: Average Hellinger distance to the real belief states for the Élysée Palace environment
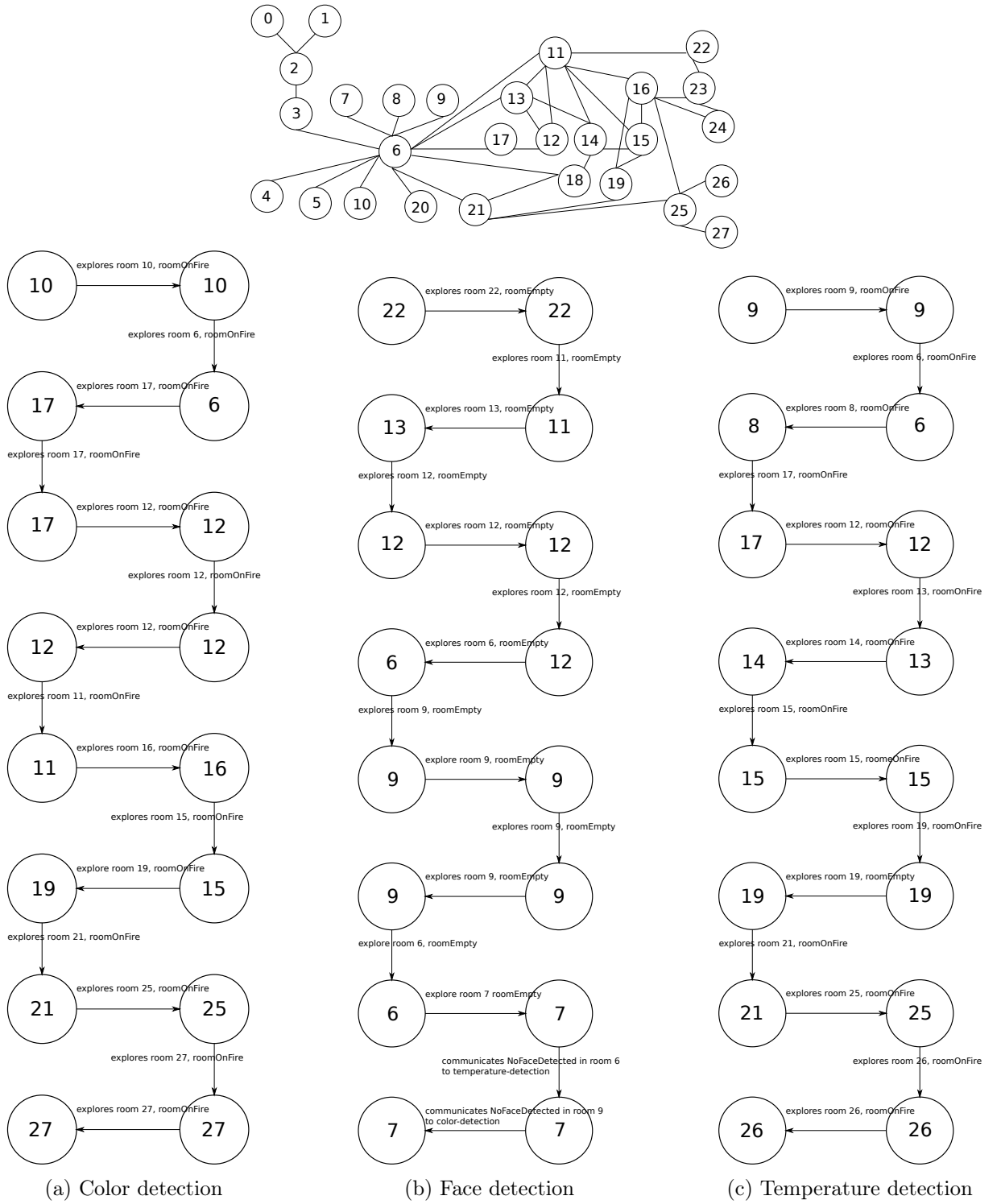
(a) Color detection (b) Face detection (c) Temperature detection

Figure 8.20: Actions performed and path traveled by the three robots during the 14 first iterations in the Élysée Palace environment

8.5 Global analysis and conclusion

8.5.1 Analysis of the exploration efficiency

The system seems to perform better in more difficult environments. Indeed, the beliefs obtained for the house environment are not accurate enough to be reliable. In the outdoor and the one-way environments though, the system manages to maintain reliable beliefs. In the Élysée Palace environment, the accuracy is not very good, but this probably results from the small amount of agents in the system regarding the size of the environment. We believe that the low communication cost could damage the efficiency of the system. Indeed, with a small cost, agents communicates more often, which leads to two impacts. First, agents use their decision steps to communicate and not to explore. Therefore, they spread previous information instead of collecting new information and checking that the previous one is still valid. Second, they may spread more easily incorrect information resulting from incorrect observations sent by the simulator. Next chapters will show if our hypothesis is right and if the results get better with a higher communication cost.

8.5.2 Analysis of the communication policy

The results presented in this chapter regarding the communication shows that MAPING is able to adapt the amount of communications to the situation. Thus when the exploration is more expensive than the communication, the system increases the amount of communication to be more efficient. In all environments, since the communication cost is very low, the agents communicate a lot with the risk to spread incorrect beliefs among the system. We also noticed that for all environments, the color-detection agent was the one which communicated the most, spending sometimes more than 80% of its decision steps to communicate. However, the logs show that all agents have equivalent quality in their beliefs among the simulation.

8.5.3 Analysis of the path traveled

All the experiments show that the agents are traveling in the environment in a intelligent way, favoring paths with small costs. Agents are also able to organize their exploration and communication to spend the lowest possible cost. The communication actions are performed when the exploration is too costly for the gain.

8.5.4 Conclusion

In this chapter we evaluated the MAPING framework in various environments with a low communication costs. All the environments have been designed to reflect real-life situations and to analyze the behavior of the system under different constraints. The house environment modeled classic building, with no specific constraints. The one-way environment modeled a very specific kind of building in which some transitions may be impossible due to the building design or the capabilities of the robots. The outdoor environment modeled an environment where transitions are possible but costly. This is the typical kind of environment a robot may have to deal with in search-and-rescue applications. The system proved to be able to maintain a correct belief state but the accuracy of this belief state is not completely reliable. The agents also have a very high communication rate, which can decrease the efficiency of the system.

The Élysée Palace environment has been designed to test the scalability of the system. In this environment, the performance of the system decreases compared to previous ones. More

tests should be made with more agents to see how the number of agents affect the performance of the system in big environments. Those tests could unfortunately not be performed due to our limited computing resources.

We remind that in all these environments, the model used is the same (described in section 7.3) and only the graph of the environment changes. There is no change in the implementation and no re-computation of the sub policies between environments. The same sub-policies have been used in all cases and the computation of the best action depending on the environment is made online. The whole system has been implemented in order to be as generic as possible. If the user need to create new transition or observation functions, they only need to implement a new POMDP class, the separation in sub-POMDP and the solving being done automatically. If the user need to modify the topology of the environment and the cost of exploration, they only need to create a new xml file containing the description of the new topology.

In the next chapter we will run the same experiments but with a medium communication cost. Our purpose is to observe the changes induced in the system's behavior when communication is costly and to see if this could prevent the agents from communicating uncertain observations and so spreading incorrect beliefs.

Chapter 9

Medium communication cost

Contents

In this chapter we present the results of the experiments realized with a medium communication cost, that is to say a communication cost equivalent to travelling among a path with a cost 3. This chapter mostly follows the same structure as chapter 8. We first evaluate the efficiency of the exploration by assessing that the system is able to maintain a correct and accurate belief state. Then we evaluate the communication strategy by tracking the number of communications sent. However we will not present in details the approximation of the beliefs neither the paths travelled and the average cost of an explore action since those results are really close to the one obtained in the previous chapter. We will expose in details the results in the house environment for both static and dynamic cases. For the other environments, we will only expose the dynamic case and describe briefly the graphics when the behavior is similar to those observed in the house environment. We will stress out the points of difference and interest. As a conclusion we will analyze the differences between the environments and conclude about the global efficiency of MAPING under medium communication constraints.

9.1   Simple configuration: the house environment

9.1.1   Evaluating the exploration efficiency

Figure 9.1 presents the results of tracking the number of correct beliefs in the static and the dynamic cases. The blue plain curve shows the evolution of the average number of correct beliefs. Since the environment contains 10 rooms and 20 variables, the maximum number of correct or incorrect states is 20. The yellow dotted curve shows the number of variables that changed value at this decision step. The probability of changes in the environment follows the transition function described in section 7.3.

The results are very similar to what has been observed with a medium communication cost in the previous chapter. However we notice that when incorrect observations are received and the number of correct beliefs decrease, the system is slower to correct itself and to reach the maximum amount of correct beliefs again. This is obviously a direct consequence of the medium communication costs: as following sections will show it, the agents communicate much less and so are longer to realize that there has been a change, and so longer to check and to correct their beliefs. In the dynamic case however the system seems to be as efficient as in the previous chapter since it manages to keep correct beliefs for about 70% of the environment.

Figure 9.2 shows the evolution of the average Hellinger distance between the agent's beliefs and the perfect belief state representing the real state of the environment. The perfect belief state is a belief state in which all probability distributions are either $(0, 1)$ or $(1, 0)$, 1 being the probability associated to the value which is actually correct in the environment.


Example 9.1 If we consider only one room of the environment, which is empty and not on fire, the perfect belief state represented the environment's state is

$$\mathcal{B}^{*,\mathcal{E}} = \begin{pmatrix} (1,0) \\ (0,1) \end{pmatrix} \quad \begin{matrix} \%RoomEmpty = true \\ \%RoomOnFire = false \end{matrix}$$


We observe that the accuracy of the beliefs is very good for static environment since the beliefs vary between level 2 (that is to say $(0.3, 0.7)$ or $(0.7, 0.3)$) and level 3 (that is to say $(0.2, 0.8)$ or $(0.8, 0.2)$). In the dynamic case, the beliefs are also good since they oscillate between level 1 (that is to say $(0.4, 0.6)$ or $(0.6, 0.4)$) and level 3. We also notice that the Hellinger distance increases dramatically between decision steps 200 and 250, probably due to the high amount of changes just before the decision step 200 and to the sequence of 3 incorrect observations that occurred at the same time. A similar scenario, though less important, is observed around decision steps 350 and 400. In all cases the system manages to improve its beliefs of 1 level in less than 50 decision steps.

9.1.2   Evaluating the communication

Figure 9.3 shows the number of communications sent by each agent during the simulation in the house environment. It is surprising to see that the difference in the number of communication actions with the low communication cost is very high in the static case, but very small in the dynamic case. Indeed, the agents used a total of 325 communication actions in the static case and 644 in the dynamic case. The average number of communication actions per robot is 108 in the static case and 214 in the dynamic case.
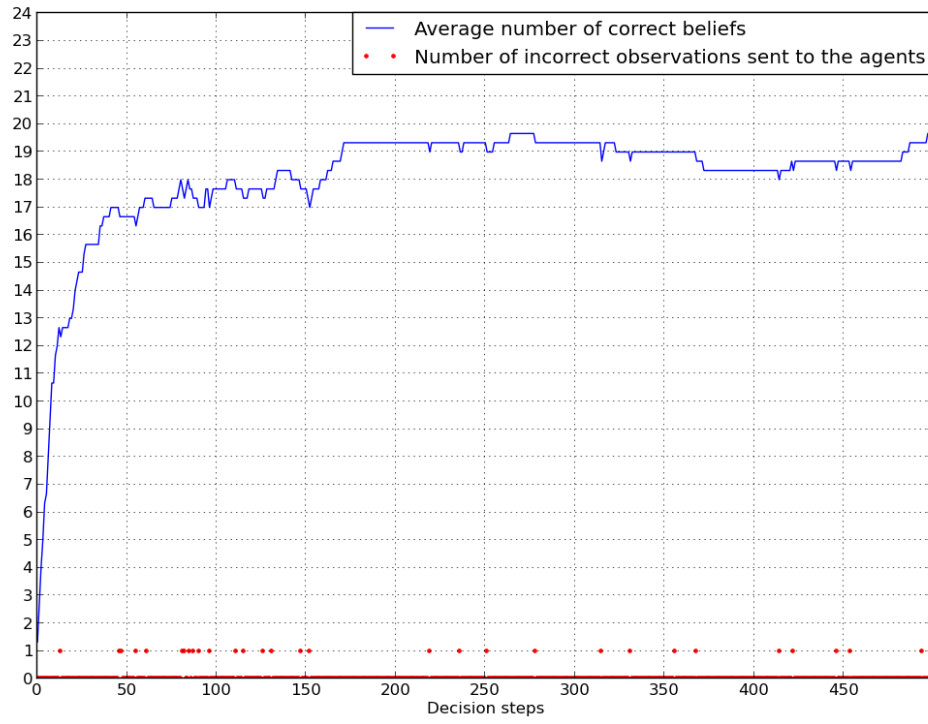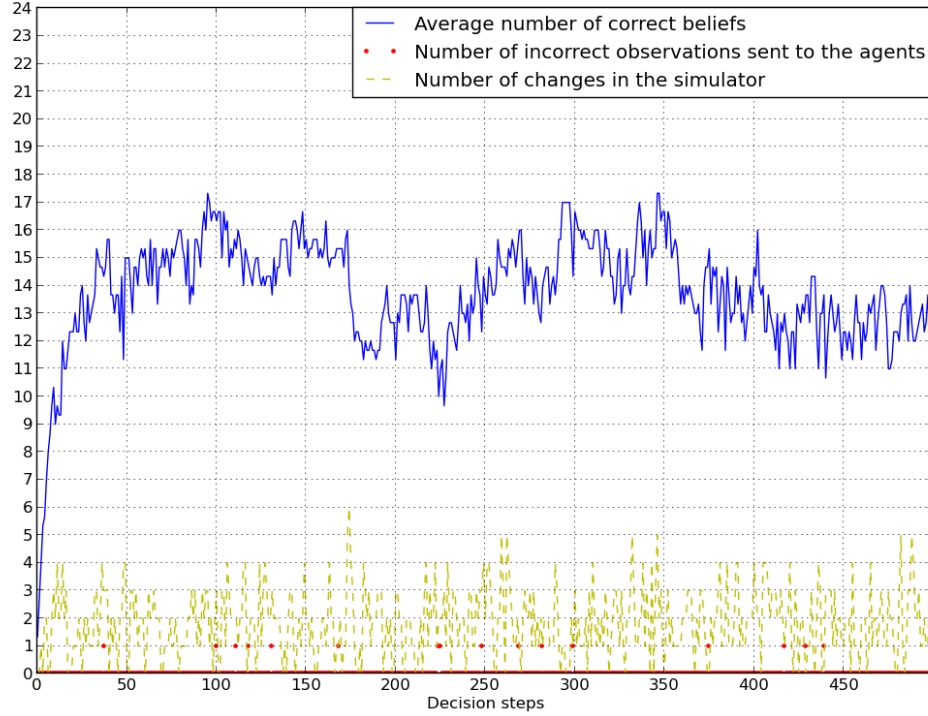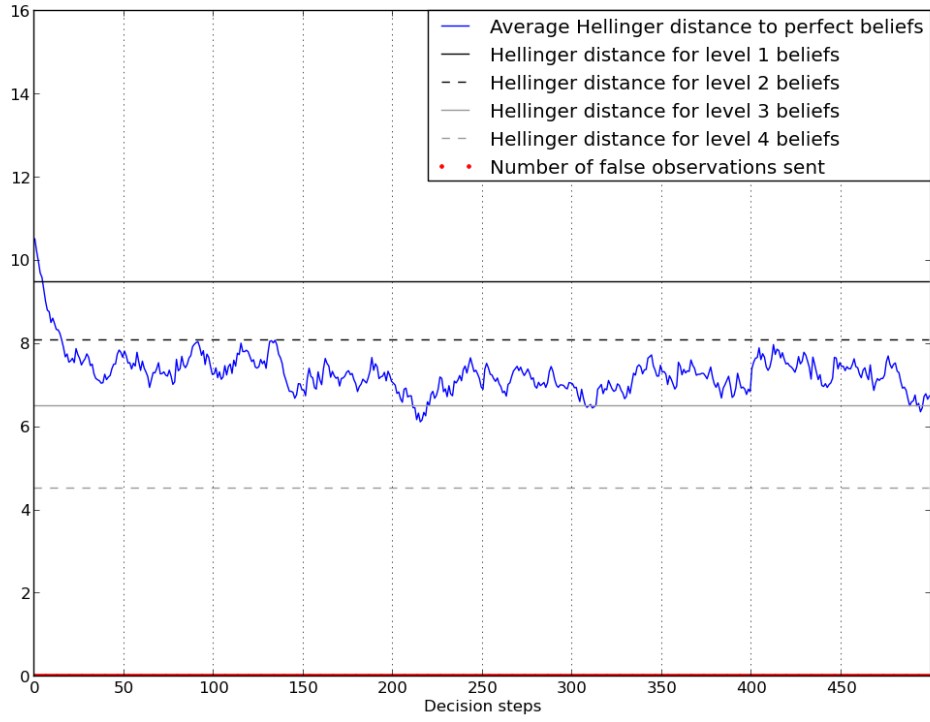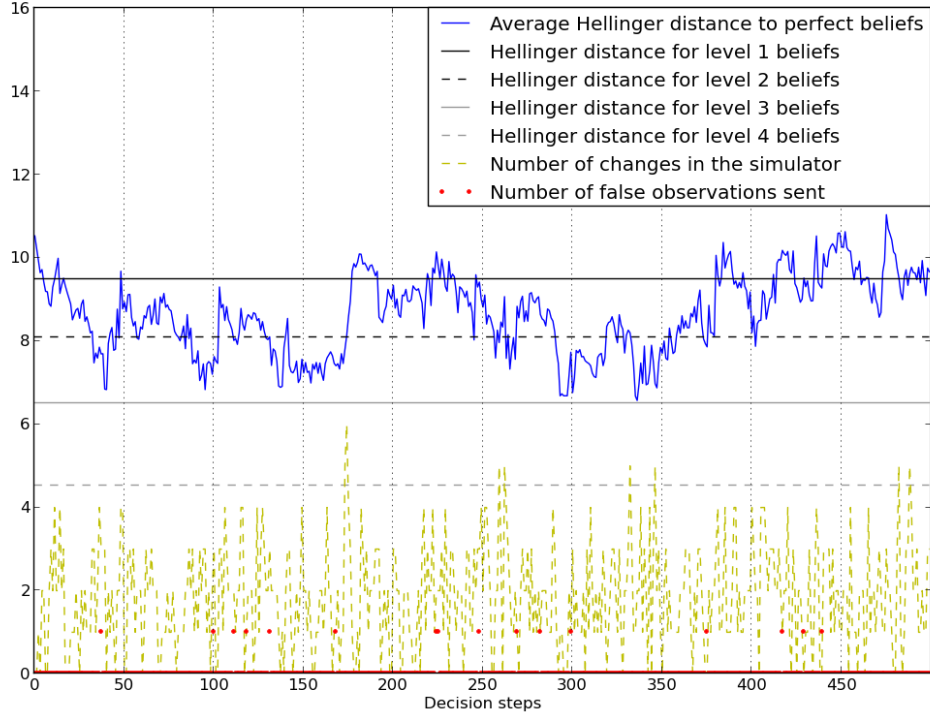
(a) Static case

(b) Dynamic case

Figure 9.1: Average number of correct beliefs for the house environment

(a) Static case

(b) Dynamic case

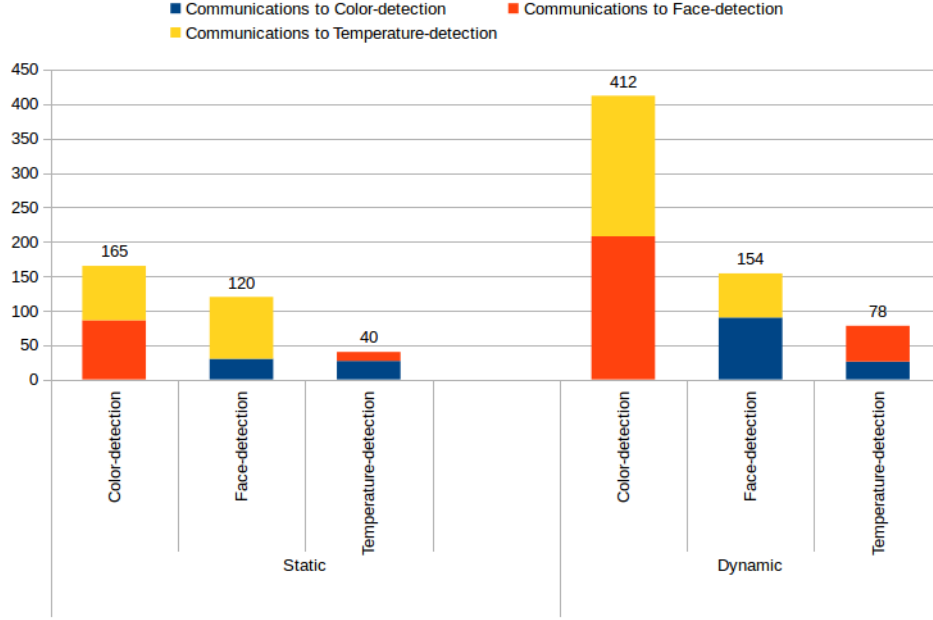Figure 9.2: Average Hellinger distance to the perfect belief state for the house environment

Figure 9.3: Number of communications per robot for the house environment

	Low communication cost				Medium communication cost			
	Static		Dynamic		Static		Dynamic	
	Total	Average	Total	Average	Total	Average	Total	Average
House environment	728	243	685	228	325	108	644	214

Table 9.1: Total and average number of communication in static and dynamic cases, for low and medium communication costs

The robots communicate less with the medium communication cost than with the low communication cost. However the difference is very small in the house environment in the dynamic case. Nevertheless, the quality of the beliefs is much better in this chapter than they were in with the low communication cost. It seems so that, even if the agents communicate as much, their communication is more efficient.

9.1.3   Evaluating the homogeneity of the beliefs

Figure 9.4 presents the evolution of the quality of the agents' approximations of other agents' beliefs for the house environment. To measure this quality, we computed for each pair of agents $< a_1, a_2 >$ the Hellinger distance between agent a_1's approximation of agent a_2's beliefs and agent a_2's real beliefs and we tracked this measure among time.

The quality of the approximation measure is less than the one obtained with the low communication cost. This was expected: since the agents communicate less, they have less opportunity to update their beliefs about other agents. However, in the dynamic case this approximation remains very good since it remains better than a level 4 approximation (that is to say an approximation which is separated from the real belief of 0.1).

> **Example 9.2** If we consider an agent that has the following belief
>
> $$b_{a_2}^k = \big((0.7, 0.3)\big)$$
>
> Then the belief
>
> $$b_{a_1}^{a_2,k} = \big((0.8, 0.2)\big)$$
>
> is a level 4 approximation of $b_{a_2}^k$

9.1.4   Analysis

As expected, the agents communicate less with this medium communication cost but still manage to maintain satisfactory belief states, though the difference of communication is quite small in the house environment. The quality of the belief states is enhanced compared to the previous chapter, which strengthen the idea that too much communication spread incorrect beliefs. It also seems that the agents take longer to detect that their beliefs are not coherent (due to the low amount of communication), which involves that they take longer to detect and acknowledge a change in the environment. The next sections will show if this behavior is enhanced by constraints and if it is harmful for the efficiency of the system.
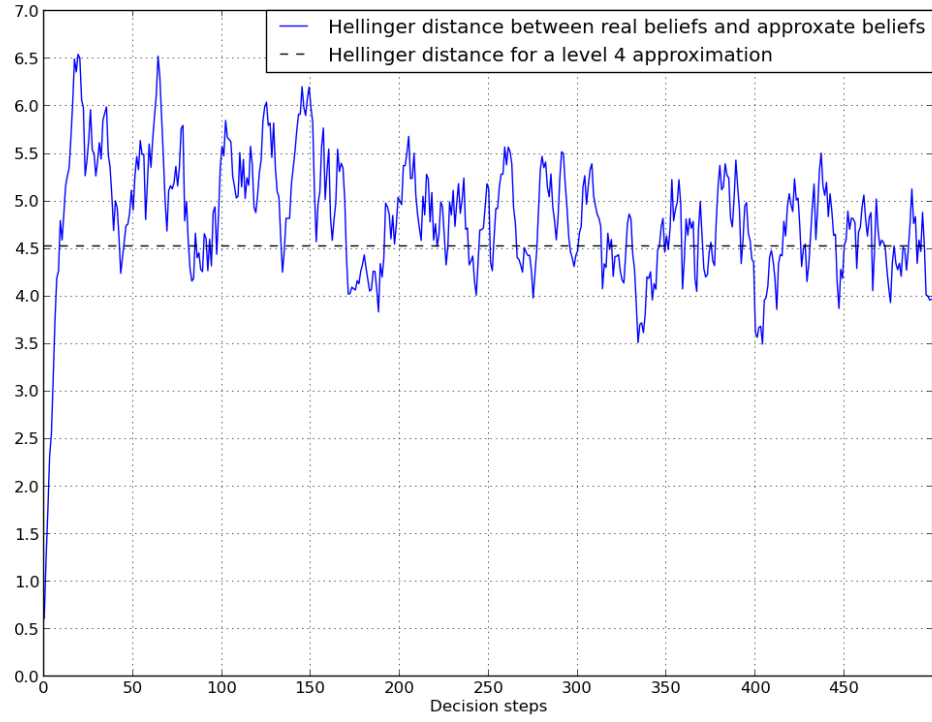
(a) Static case
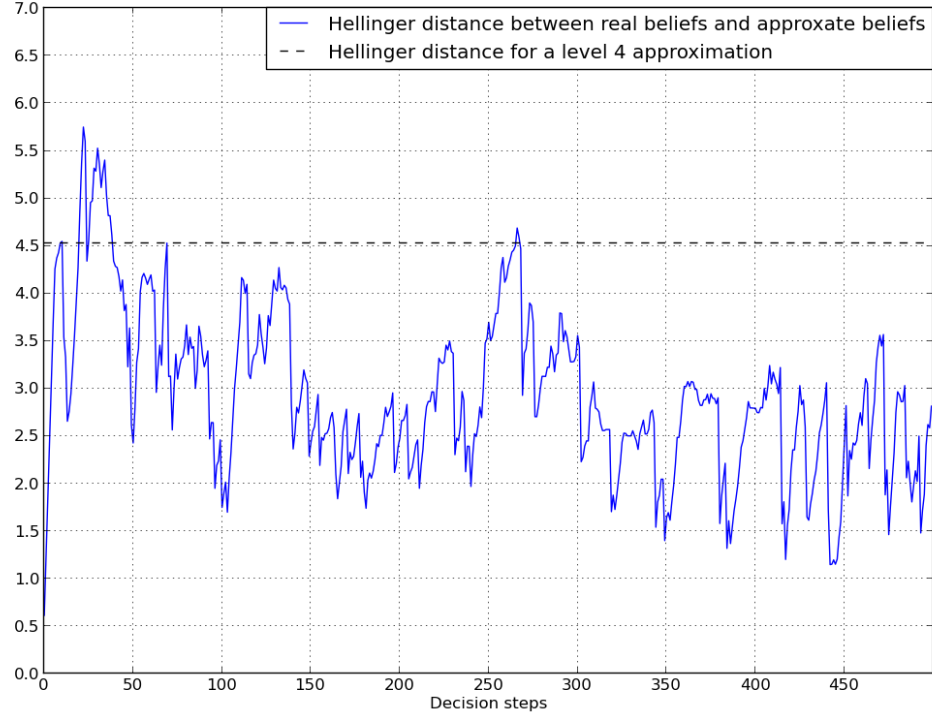
(b) Dynamic case

Figure 9.4: Average Hellinger distance to the real belief states for the house environment

9.2 Constraints in the navigation: the one-way environment

The one-way environment presents a particular topology, in which some transitions between rooms can only be done in one way. This constrained environment can be encounter in building with fire doors or with some doors that a robot can push but not pull.

9.2.1 Evaluating the exploration efficiency

Figure 9.5 presents the evolution of the number of correct states and figure 9.6 presents the evolution of the Hellinger distance between the agents' belief states and the perfect belief state in the dynamic case.
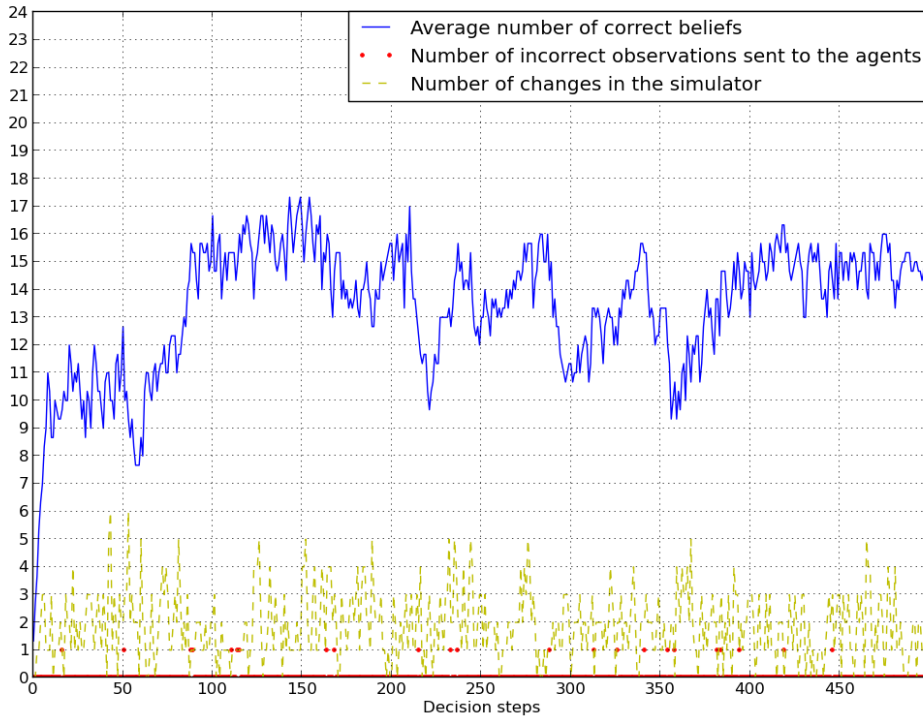
Figure 9.5: Average number of correct beliefs for the one-way environment in the dynamic case

The results are similar to those obtained in the house environment. The agents manage to maintain correct beliefs about 70% of the environment and a good accuracy since their beliefs oscillate around level 2 and the quality remains quite constant.

9.2.2 Evaluating the communication

Figure 9.7 shows the number of communications sent for each pair emitter-receiver.

In total, the agents used a communicate action 353 times in the static case and 588 time in the dynamic case. In average, each agent communicated 118 times in the static case and 196 in the dynamic case. This numbers are summarized and compared to previous environments in table 9.2.
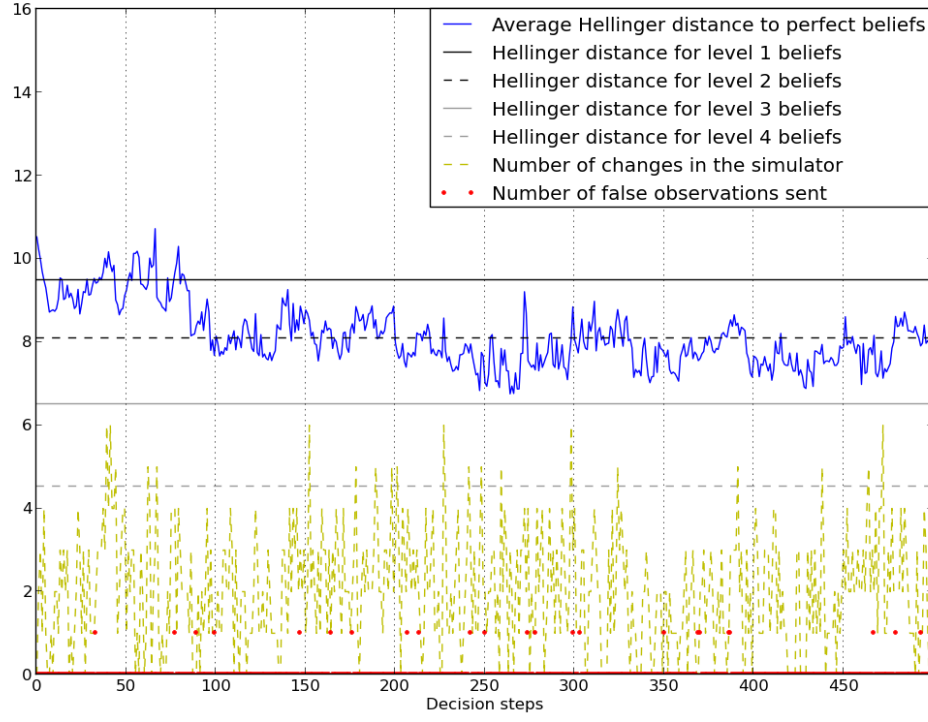
Figure 9.6: Average Hellinger distance to the perfect belief state for the one-way environment in the dynamic case
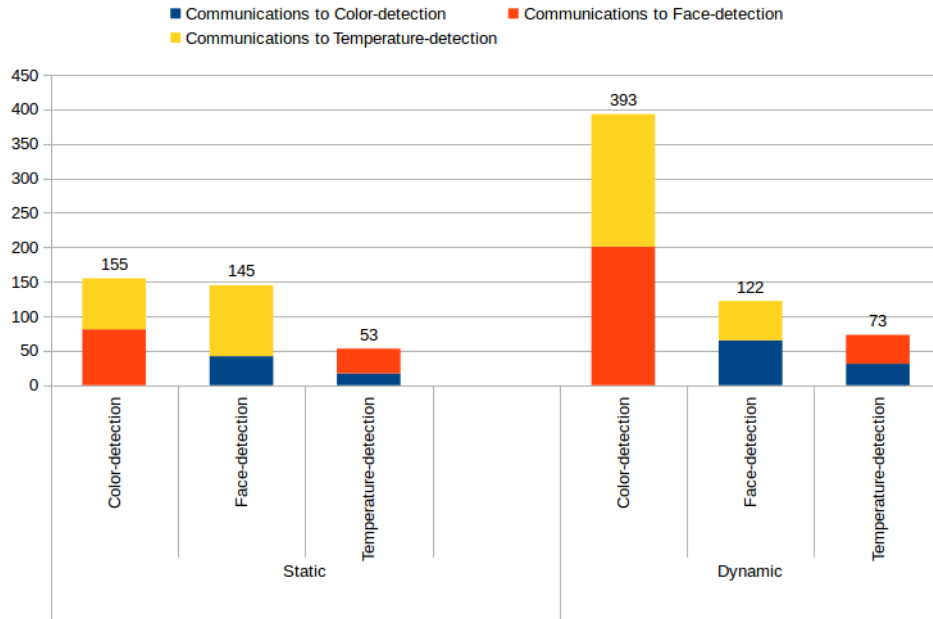

Figure 9.7: Number of communications per robot for the one-way environment

	Low communication cost				Medium communication cost			
	Static		Dynamic		Static		Dynamic	
	Total	Average	Total	Average	Total	Average	Total	Average
House environment	728	243	685	228	325	108	644	214
One-way environment	582	194	664	221	353	118	588	196

Table 9.2: Total and average number of communication in different environments, in static and dynamic cases, for low and medium communication costs

The number of communication actions is much lower than with the low communication cost. But we observe that the color-detection agent communicated again much more than the two others and spent most of its decision steps to communicate with others.

9.2.3   Analysis

It seems that the lower rate of communication doesn't affect the efficiency of the system too much in this environment: the agents still manage to maintain level 2 beliefs (that is to say $(0.8, 0.2)$ or $(0.2, 0.8)$). The quality of the beliefs state is once again better and more steady than with the low communication cost. This strengthen again the idea that a restrained communication is beneficial for the quality of the system since it prevents incorrect beliefs to be spread too much. Changes and incorrect observations also doesn't seem to degrade more the belief state than previously in the dynamic case.

9.3 Costly transitions: the outdoor environment

The outdoor environment is based on real topographic maps from the Alpes, near the city of Chatel. This environment has been designed to observe the behavior of the system in areas where moving from a "room" (in this case we still use the denomination room even if those rooms are only virtual) to another can be very costly to the robots, due for instance to difficult ground.

9.3.1 Evaluating the exploration efficiency

Figure 9.8 presents the evolution of the number of correct states and figure 9.9 presents the evolution of the Hellinger distance between the agents' belief states and the perfect belief state in the outdoor environment.
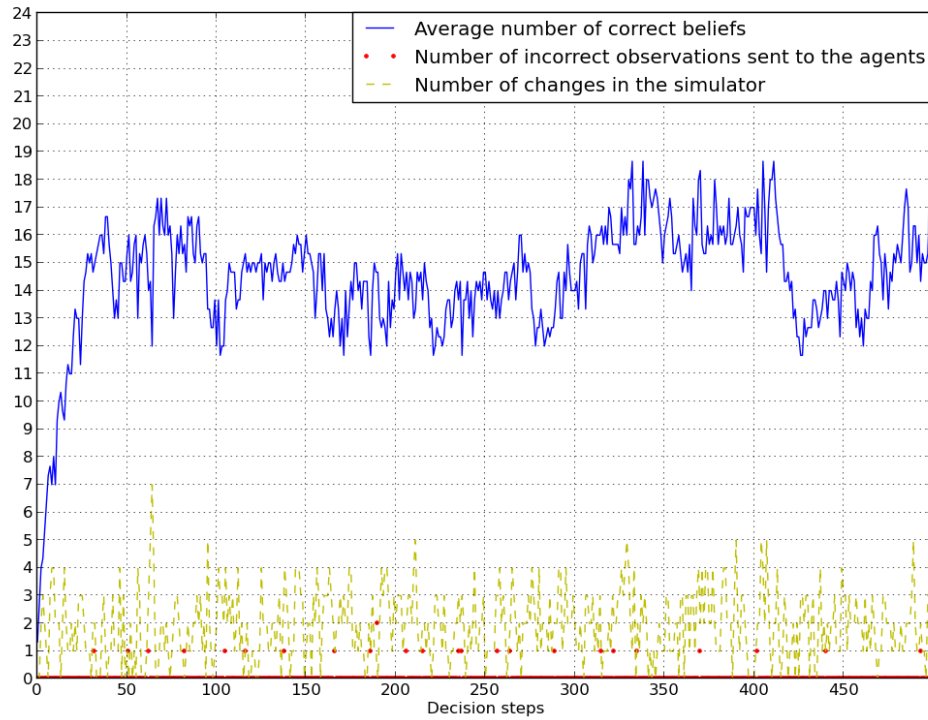

Figure 9.8: Average number of correct beliefs for the outdoor environment in the dynamic case

The system still manage to keep about 70% of correct beliefs and the accuracy oscillates around level 2. We also notice that, as for the one-way environment, the quality of the beliefs varies less frequently than with the low communication.

9.3.2 Evaluating the communication

Figure 9.10 presents the number of communications sent by the agents in the static and dynamic cases. The agents have used a total of 446 communicate actions in the static case, that is to say 149 communications per agent in average. In the dynamic case, the agents have used a total of 562 communicate actions, that is to say 187 communications per agent in average. These numbers are summarized and compared to previous environments in table 9.3. The difference in the number of communication between the low and the medium communication cost is very
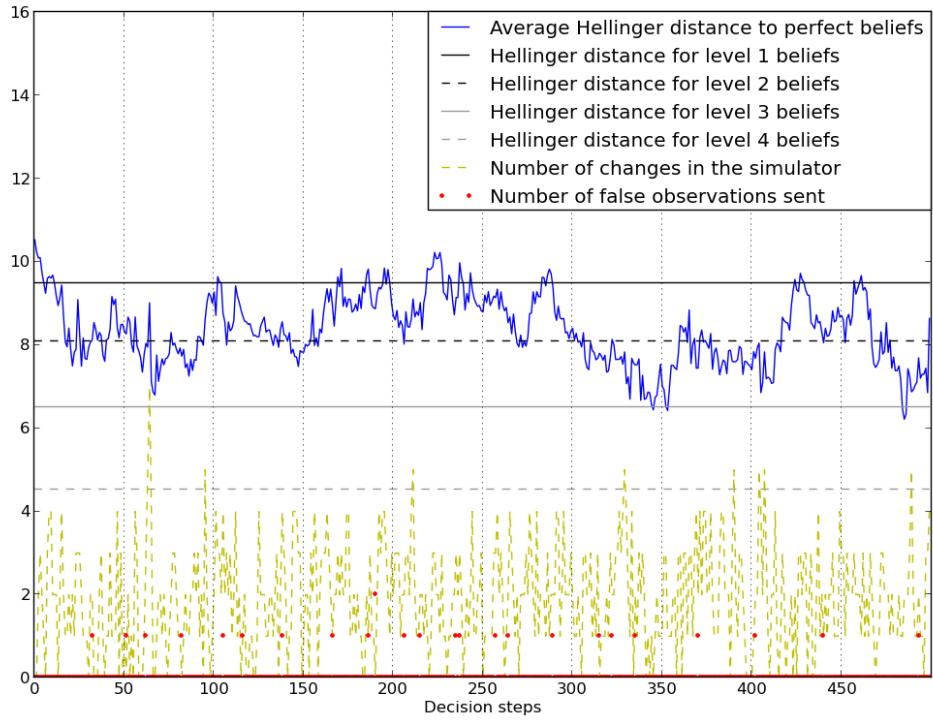
Figure 9.9: Average Hellinger distance to the perfect belief state for the outdoor environment in the dynamic case
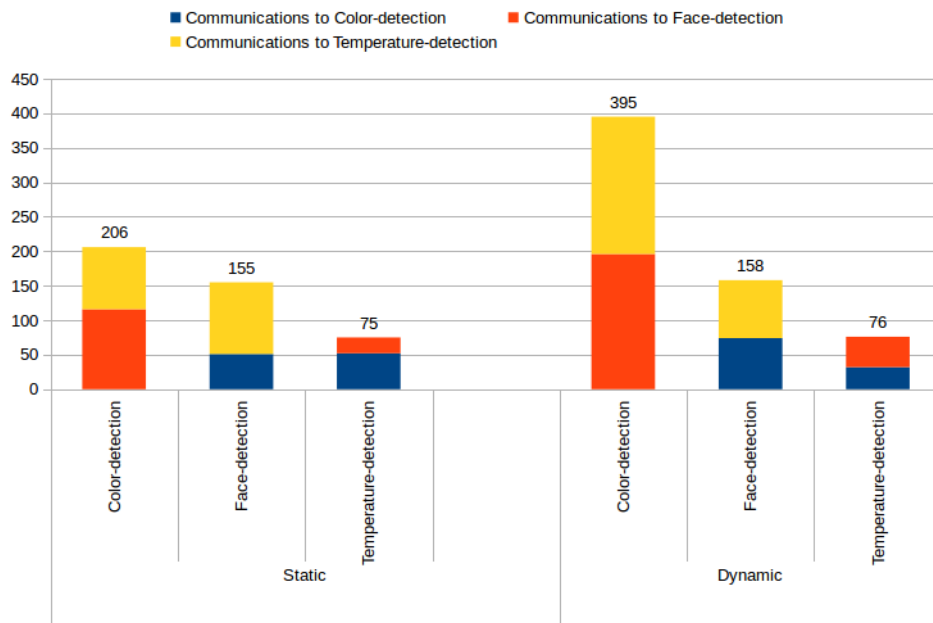

Figure 9.10: Number of communications per robot for the outdoor environment

	Low communication cost				Medium communication cost			
	Static		Dynamic		Static		Dynamic	
	Total	Average	Total	Average	Total	Average	Total	Average
House environment	728	243	685	228	325	108	644	214
One-way environment	582	194	664	221	353	118	588	196
Outdoor environment	751	250	754	251	436	145	629	210

Table 9.3: Total and average number of communication in different environments, in static and dynamic cases, for low and medium communication costs

important in this environment. Since transitions are very costly, a low communication cost could be a replacement for exploration and possibly overused. With this medium communication cost, communicating is as expensive as traveling a path of cost 4 and so more balanced with the exploration cost. The communication is only dedicated to more accurate beliefs and bigger divergence between the agents.

9.3.3 Analysis

The results for the outdoor environment are very satisfactory. The agents manage to keep a good amount of correct beliefs and a good accuracy. The number of communications is reduced regarding to the low communication cost, which involves less variation in the accuracy of the beliefs during the simulation. The system seems to be able to adapt to costly environments very well with this kind of communication. The color-detection agent used once again most of its decision steps to communicate.

9.4 Scalability: the Élysée Palace environment

The Élysée Palace environment has been chosen to prove the scalability of MAPING in a house-like environment. In this section we will present again the static and the dynamic case, as well as the quality of agents' approximations. Indeed, the increase of the number of rooms changes a bit the behavior of the system.

9.4.1 Evaluating the exploration efficiency

Figure 9.11 presents the evolution of the number of correct beliefs. Figure 9.11 shows that the system is slightly less efficient with the medium communication cost than with the low communication cost in both static and dynamic cases.

Figure 9.12 confirms this decrease of efficiency since the beliefs are hardly discriminated in the dynamic case. As previously, it would be interesting to see how a higher number of agents in the system affect the exploration and the efficiency of the system.

9.4.2 Evaluating the communication

Figure 9.13 presents the number of communications sent by the agents in the static and dynamic cases.
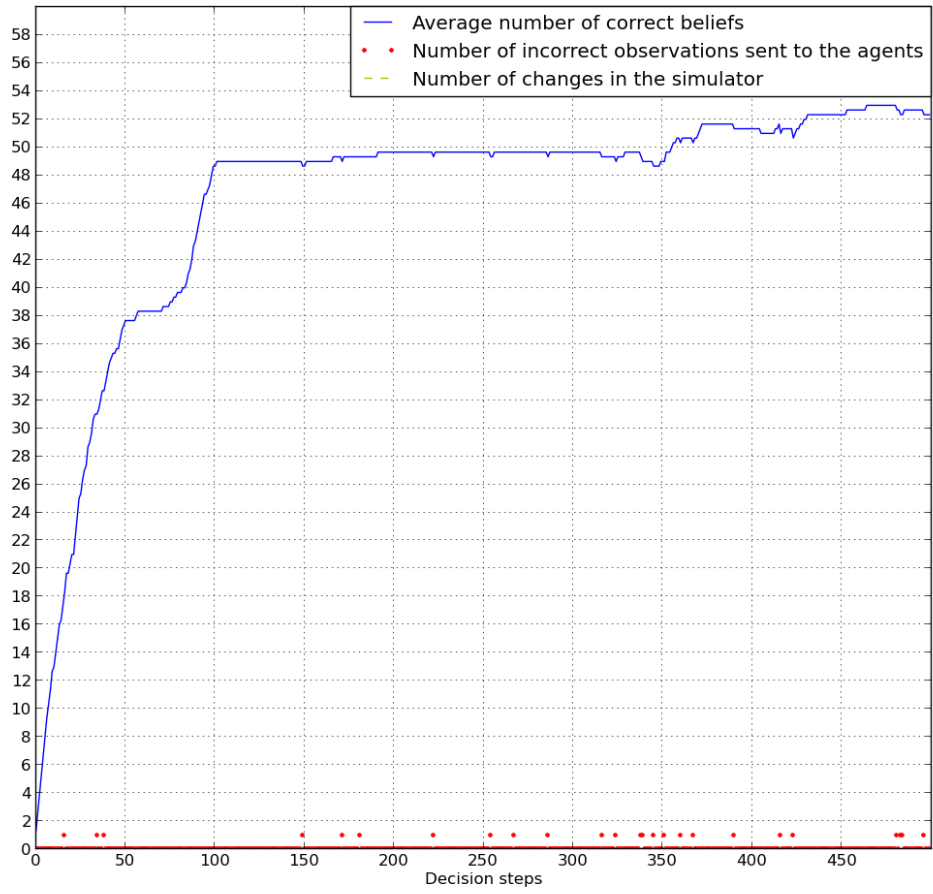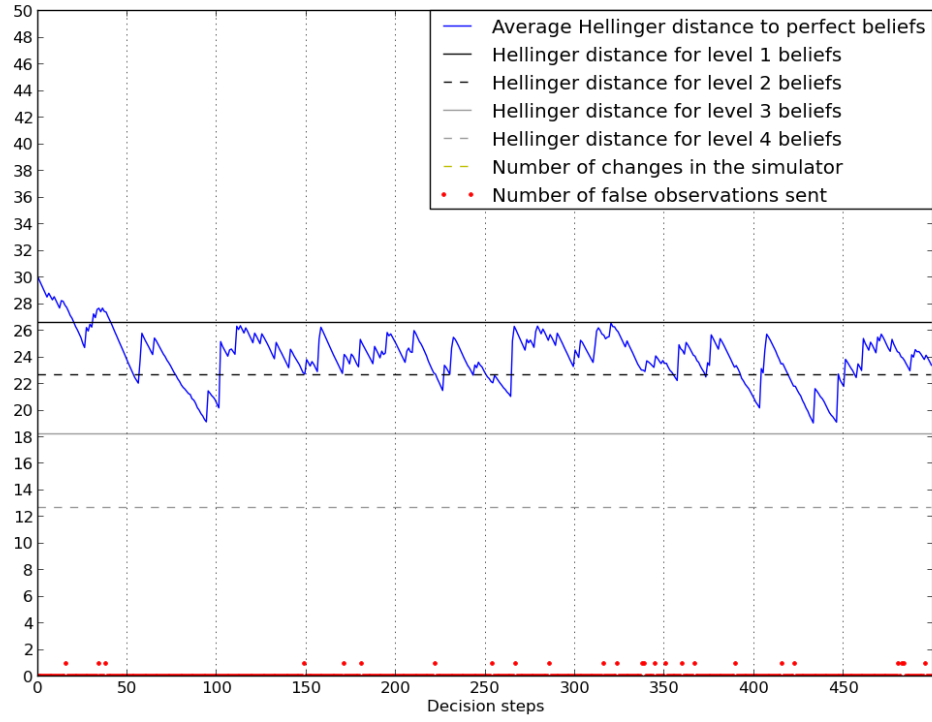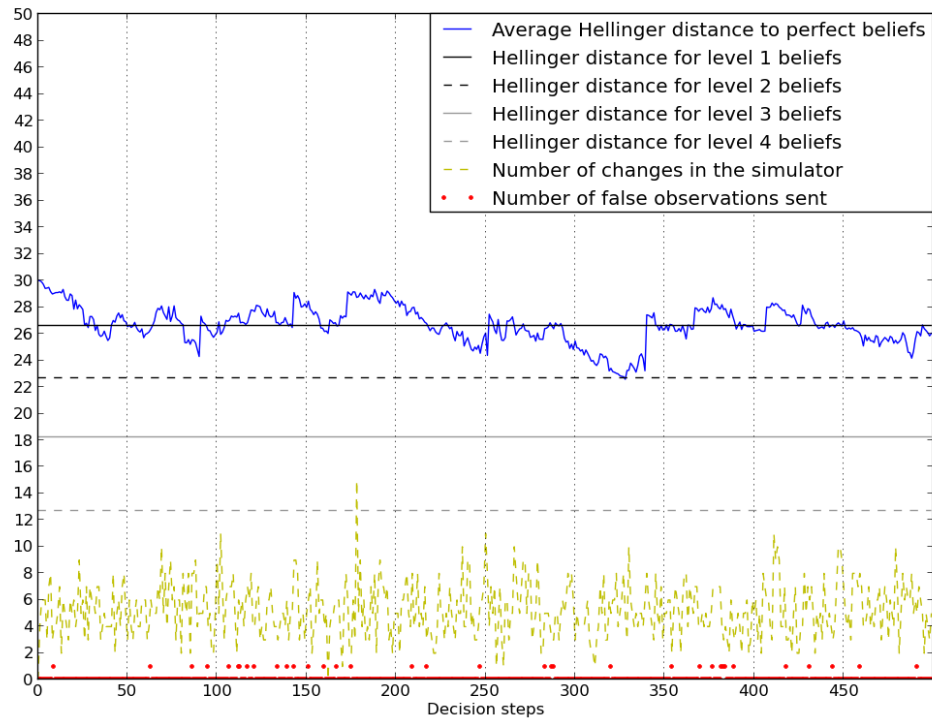
(a) Static case
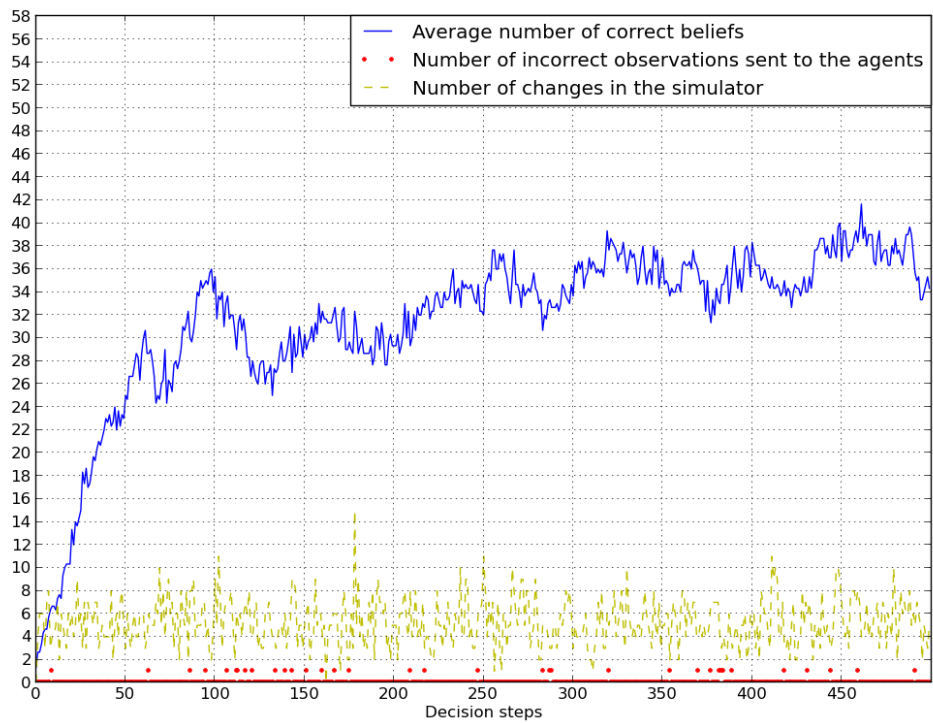
(b) Dynamic case

Figure 9.11: Average number of correct beliefs for the Élysée Palace environment

(a) Static case

(b) Dynamic case

Figure 9.12: Average Hellinger distance to the perfect belief state for the Élysée Palace environment
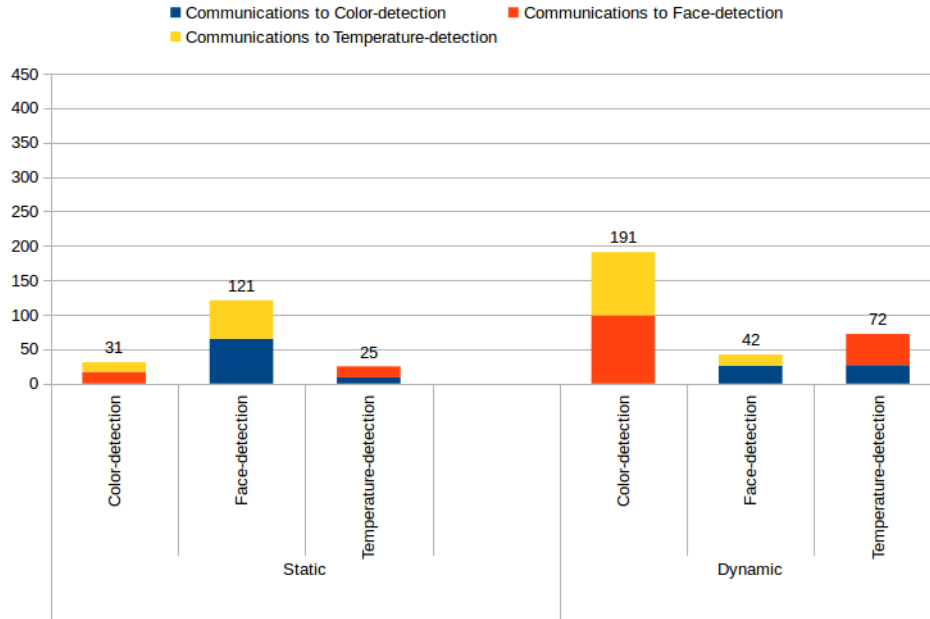
Figure 9.13: Number of communications per robot for the Élysée Palace environment

In the static case, the agents used the communicate action 177 times, that is to say 55 times per agent in average. In the dynamic case, they used the communicate action 305 times, that is to say 102 times per agent in average. Those numbered are summarized and compared to previous environments in table 8.4

	Low communication cost				Medium communication cost			
	Static		Dynamic		Static		Dynamic	
	Total	Average	Total	Average	Total	Average	Total	Average
House environment	728	243	685	228	325	108	644	214
One-way environment	582	194	664	221	353	118	588	196
Outdoor environment	751	250	754	251	436	145	629	210
Élysée environment	487	162	496	165	177	59	305	102

Table 9.4: Total and average number of communication in different environments, in static and dynamic cases, for low and medium communication costs

The agents in the Élysée Palace environment have very low communication rate. It seems obvious that the bad accuracy of the beliefs and this low communication rate are connected. Since the communication is costly, the agents wait to reach a certain accuracy in their beliefs before communicating, accuracy that they may not reach often due to the size of the environment. On the other hand, since they are not communicating a lot, it is more difficult for them to cross their beliefs and reach beliefs accurate enough. Once more, a system with more agents could affect this behavior and presents better results.

9.4.3 Analysis

As in the previous chapter, the results for the Élysée Palace environment are not very satisfactory. The agents hardly manage to discriminate their beliefs in the dynamic case. Once again, more agents in the system could help to obtain better results.

9.5 Global analysis and conclusion

In this chapter we evaluated the MAPING framework in various environments with a medium communication cost. All the environments have been designed to illustrate real applications. The efficiency of the exploration has been revealed slightly higher with the medium communication cost than with the low communication cost. The agents manage to keep the same amount of correct beliefs, but those beliefs are more accurate in this chapter than previously. The amount of communications is also reduced. This strengthen the hypothesis that the communication should be costly so that the agents only communicate important and verified observations.

We conclude from this chapter that MAPING shows promising results with higher communication cost. The next chapter will explore a high communication cost to see if those results are confirmed.

Chapter 10

High communication cost

Contents

In this chapter we present the results of the experiments accomplished with a high communication cost, that is to say a communication cost equivalent to travelling among a path with a cost 4. This cost has been designed to observe the behavior of the system under strong communication constraints. Indeed, for most of the environment used, a path with a cost of 4 already makes it possible to travel across the whole environment. This chapter follows the same structure as Chapter 9. We first evaluate the efficiency of the exploration by assessing that the system is able to maintain a correct and accurate belief state. Then we evaluate the communication strategy by tracking the number of communications sent. We will only expose the dynamic case for all environments, since the static cases are very similar to those obtained in the previous chapter. As a conclusion we will analyze the differences between the environments and conclude about the global efficiency of MAPING under strong communication constraints.

10.1   Simple configuration: the house environment

10.1.1   Evaluating the exploration efficiency

House environment

Figure 10.1 presents the results of tracking the number of correct beliefs in the dynamic case. The blue plain curve shows the evolution of the average number of correct beliefs. Since the environment contains 10 rooms and 20 variables, the maximum number of correct or incorrect states is 20. The yellow dotted curve shows the number of variables that changed value at this decision step. The probability of changes in the environment follows the transition function described in section 7.3.
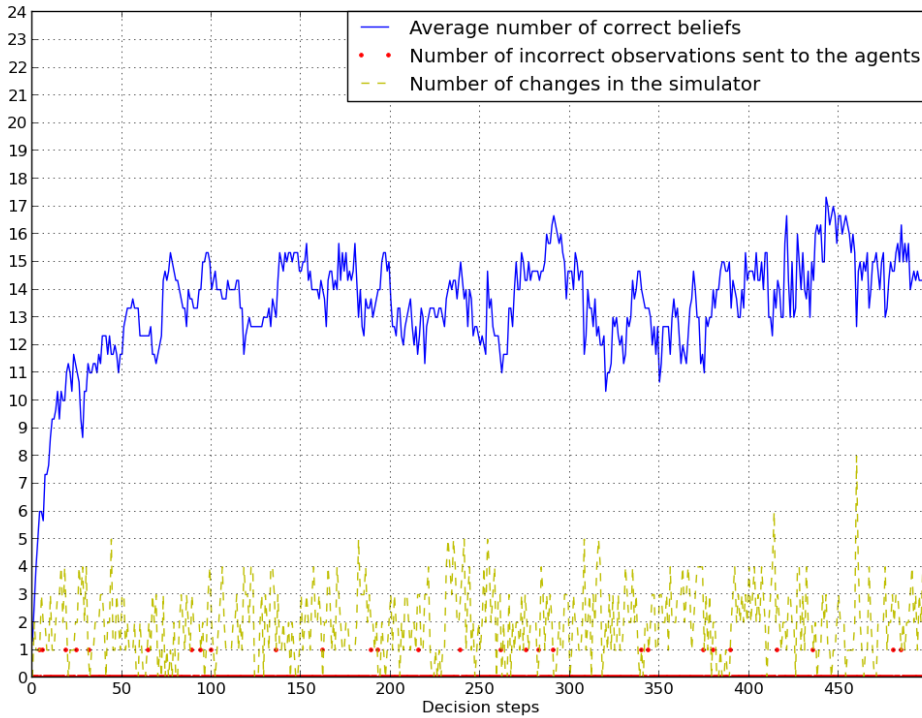

Figure 10.1: Average number of correct beliefs for the house environment

Figure 10.2 shows the evolution of the average Hellinger distance between the agent's beliefs and the perfect belief state representing the real state of the environment.

The results are very similar to those obtained in previous chapters: the system manages to keep around 70% of correct beliefs in the dynamic case and beliefs varying around level 2. The quality of the beliefs is slightly decreased just after decision step 300, but the system is still very fast to detect the incoherence and to improve the beliefs again.

10.1.2   Evaluating the communication

Figure 10.3 shows the number of communications sent by each agents during the simulation in the house environment. The agents communicated 644 times is total in the static case, that is to
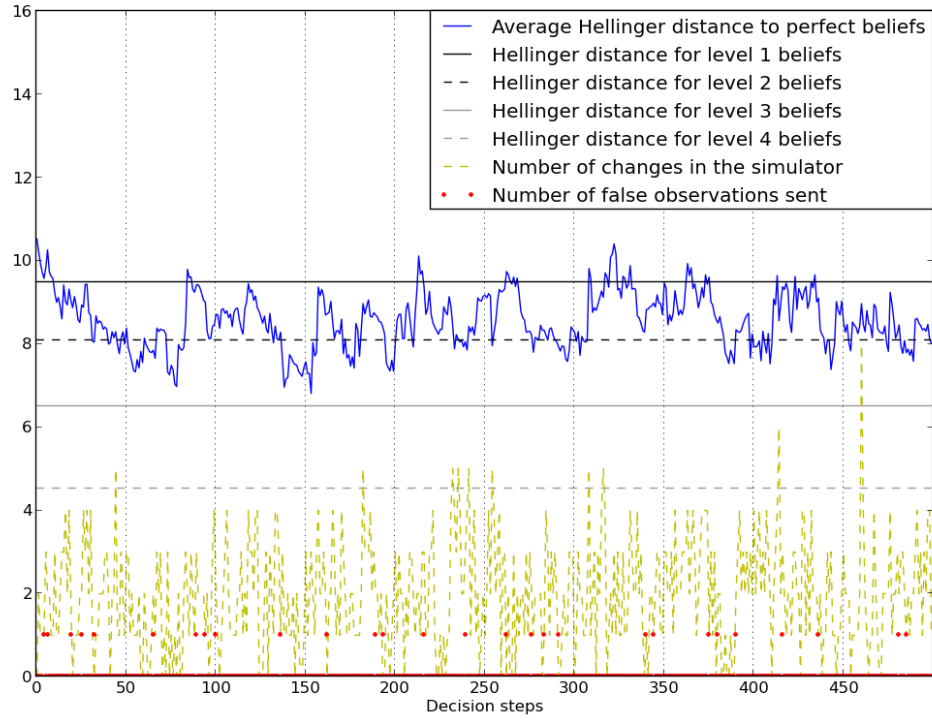
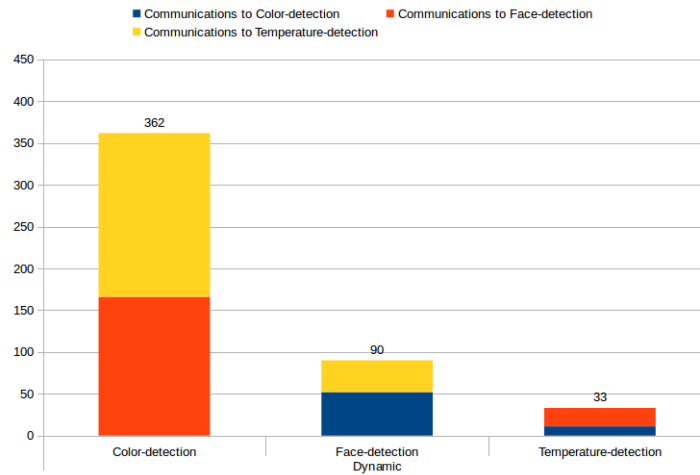Figure 10.2: Average Hellinger distance to the perfect belief state for the house environment

Figure 10.3: Number of communications per robot for the house environment

say 214 in average per agent. In the dynamic case, they communicated 485 times in total, that is to say 162 in average per agent.

	Medium communication cost		High communication cost	
	Total	Average	Total	Average
House environment	644	214	485	162

Table 10.1: Total and average number of communication in static and dynamic cases, for medium and high communication costs

Unsurprisingly, the agents communicated less with the high communication cost than with the medium communication cost. The color detection agent is still the one that communicates the most and the temperature-detection still the one that communicates the least.

10.1.3   Evaluating the homogeneity of the beliefs

Figure 10.4 and presents the evolution of the quality of the agents' approximations of other agents' beliefs for the house environment.
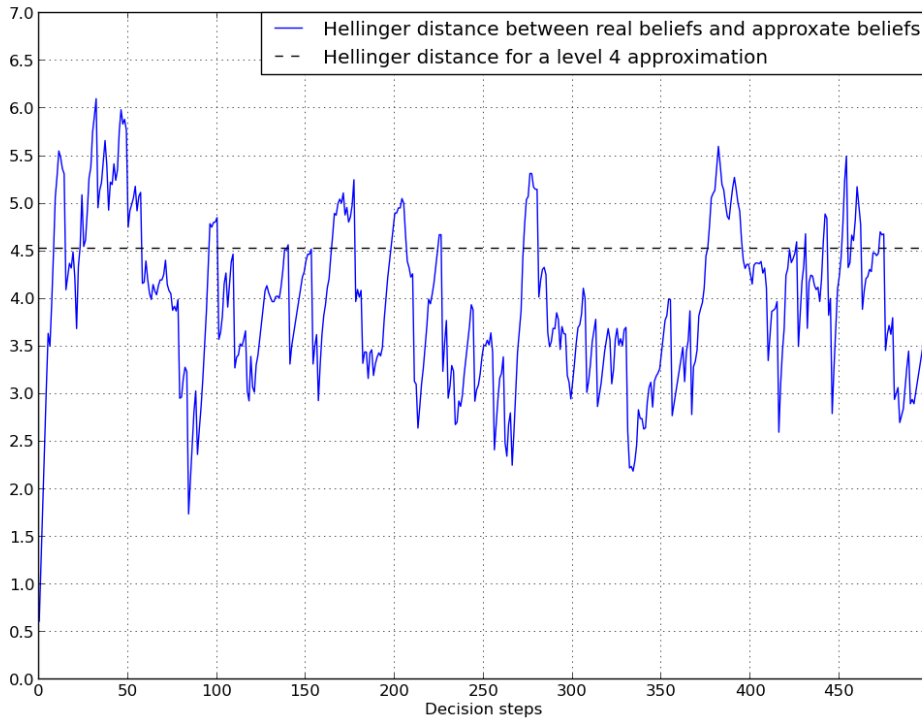

Figure 10.4: Average Hellinger distance to the real belief states for the house environment

The quality of the approximation is less good in this chapter than with the medium communication cost, due to the reduced number of communications. However, the approximation remains good enough to consider that the computation of the relevance is accurate.

10.1.4   Analysis

The efficiency of the system is slightly decreased in the house environment when the communication cost is high. However, the system is still able to get accurate and reliable beliefs and to track changes in the environment. Their approximation of other agent's beliefs remains very good. In the next parts, we will not present the graph for the quality of the approximation since they are extremely similar to what have been presented for the house environment.

10.2   Constraints in the navigation: the one-way environment

10.2.1   Evaluating the exploration efficiency

Figure 10.5 presents the evolution of the number of correct states and figure 10.6 presents the evolution of the Hellinger distance between the agents' belief states and the perfect belief state in the dynamic case.
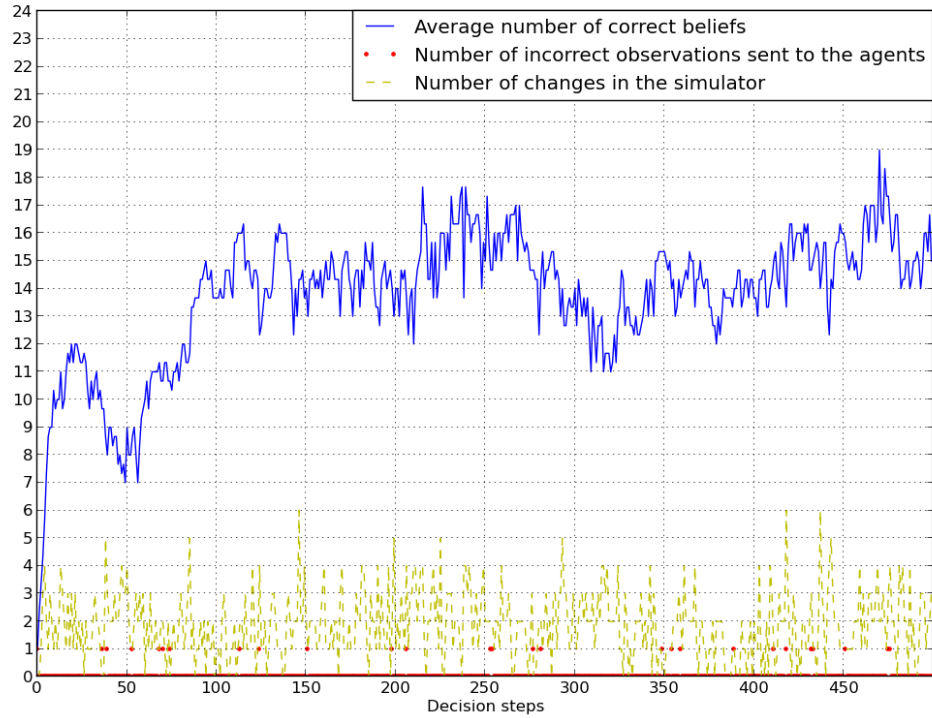
Figure 10.5: Average number of correct beliefs for the one-way environment in the dynamic case

As for previous environments, the system manages to maintain about 70% of correct beliefs and the beliefs vary between level 1 and level 3, which is quite good.

10.2.2   Evaluating the communication

Figure 10.7 shows the number of communications sent for each pair emitter-receiver. The agents communicated 574 times in total in the dynamic case, that is to say 191 in average per agent. These numbers are summarized and compared to those obtained with the medium communication cost in table 10.2.
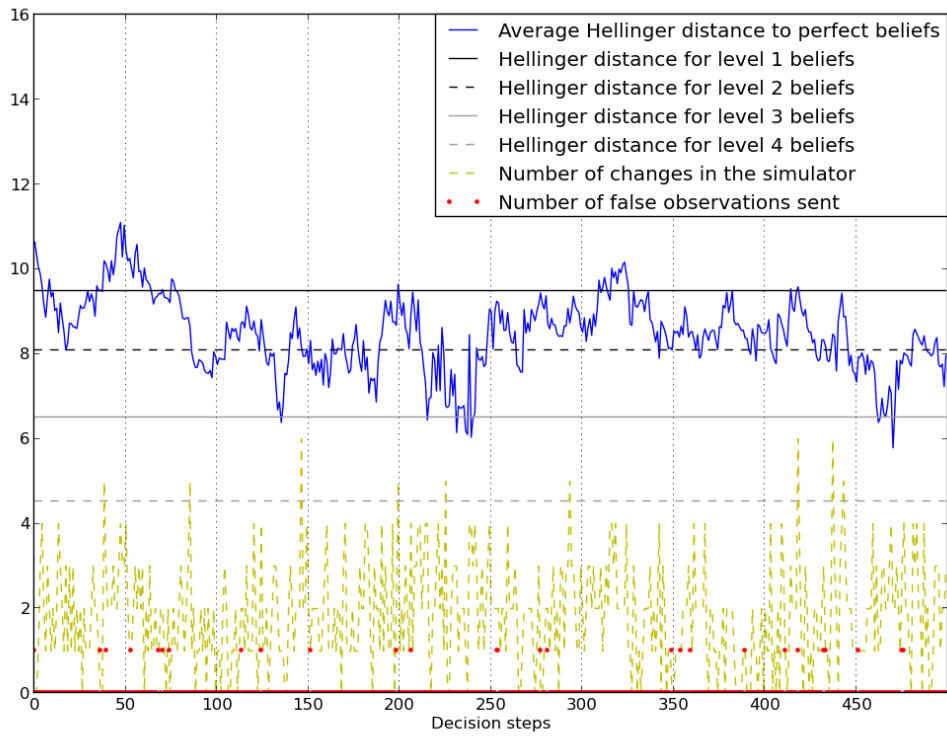
Figure 10.6: Average Hellinger distance to the perfect belief state for the one-way environment in the dynamic case
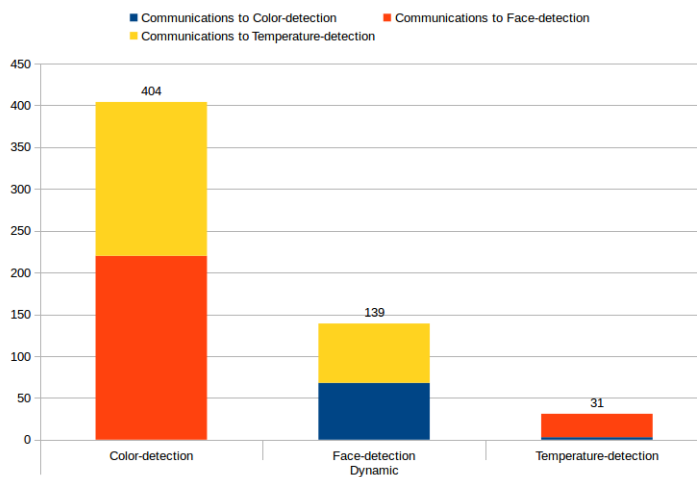
Figure 10.7: Number of communications per robot for the one-way environment

	Medium communication cost		High communication cost	
	Total	Average	Total	Average
House environment	644	214	485	162
One-way environment	588	196	574	191

Table 10.2: Total and average number of communication in different environments, in static and dynamic cases, for medium and high communication costs

The agents communicated slightly less with the high communication cost than with the medium communication cost.

10.2.3 Analysis

It seems that this high communication cost doesn't affect a lot the behavior of the system for the one-way environment. The system is still able to adapt its number of communications to the need by communicating more when the transitions are more difficult. The beliefs are still correct and accurate. This tends to show the adaptability of the system to various environment and various costs.

10.3 Costly transitions: the outdoor environment

10.3.1 Evaluating the exploration efficiency

Figure 10.8 presents the evolution of the number of correct states and figure 10.9 presents the evolution of the Hellinger distance between the agents' belief states and the perfect belief state in the outdoor environment.

The system still manage to keep about 70% of correct beliefs and the accuracy oscillates around level 2. Nothing special is to be noted about this simulation.

10.3.2 Evaluating the communication

Figure 10.10 presents the number of communications sent by the agents in the static and dynamic cases. The agents have used a total 597 communicate actions, that is to say 199 communications per agent in average. These numbers are summarized and compared to previous environments in table 10.3.

	Medium communication cost		High communication cost	
	Total	Average	Total	Average
House environment	644	214	485	162
One-way environment	588	196	574	191
Outdoor environment	629	210	597	199

Table 10.3: Total and average number of communication in different environments, in static and dynamic cases, for medium and high communication costs

The agents communicate less with the high communication cost than with the medium communication cost in the outdoor environment. The difference between the high and the medium
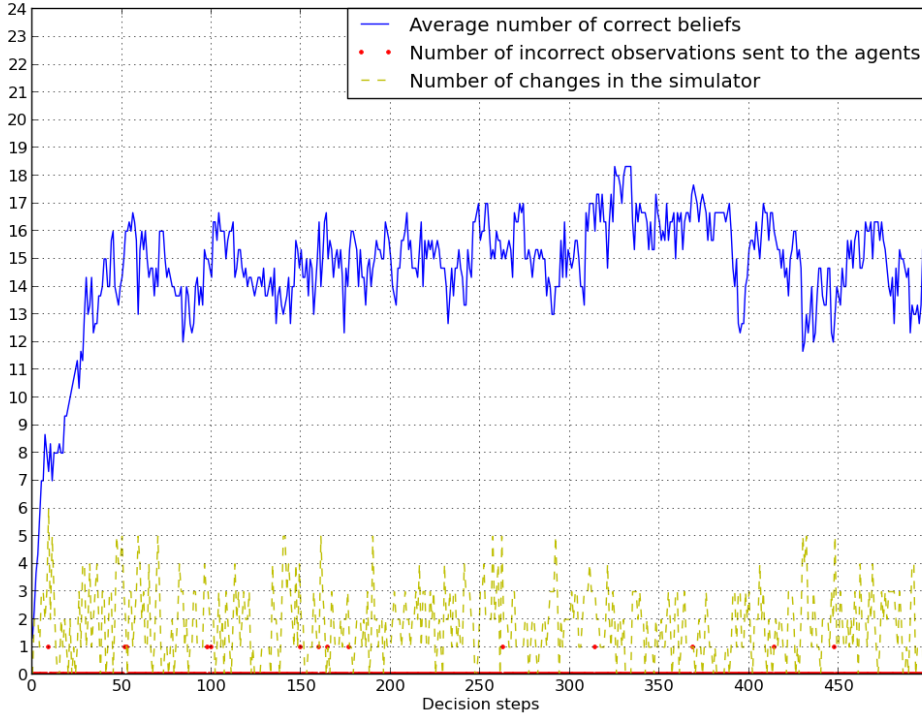
Figure 10.8: Average number of correct beliefs for the outdoor environment in the dynamic case

costs is less important that the difference we observed between the medium and the low cost.

10.3.3 Analysis

It seems that the agents communicate almost as much with the high communication cost than they did with the medium communication cost. The efficiency of the system remains so almost unchanged. This strengthen the idea that the system is able to adapt itself to the need of the environment.

10.4 Scalability: the Élysée Palace environment

10.4.1 Evaluating the exploration efficiency

Figure 10.11 presents the evolution of the number of correct beliefs.

Figure 10.12 presents the evolution of the Hellinger distance between the agents' beliefs and the perfect belief state. Both figures prove that the system is definitely less efficient in larger environments. The number of correct beliefs hardly overcome half of the variables and the beliefs themselves are not discriminated enough. The logs show that at the end of the simulation, all the agents have almost all their beliefs discriminated. Therefore this lack of efficiency is not due to the simulation being too short and will obviously not be solved by running the system longer. Once again, we believe that more agents could make the system more efficient but this hypothesis will be analyzed in future works
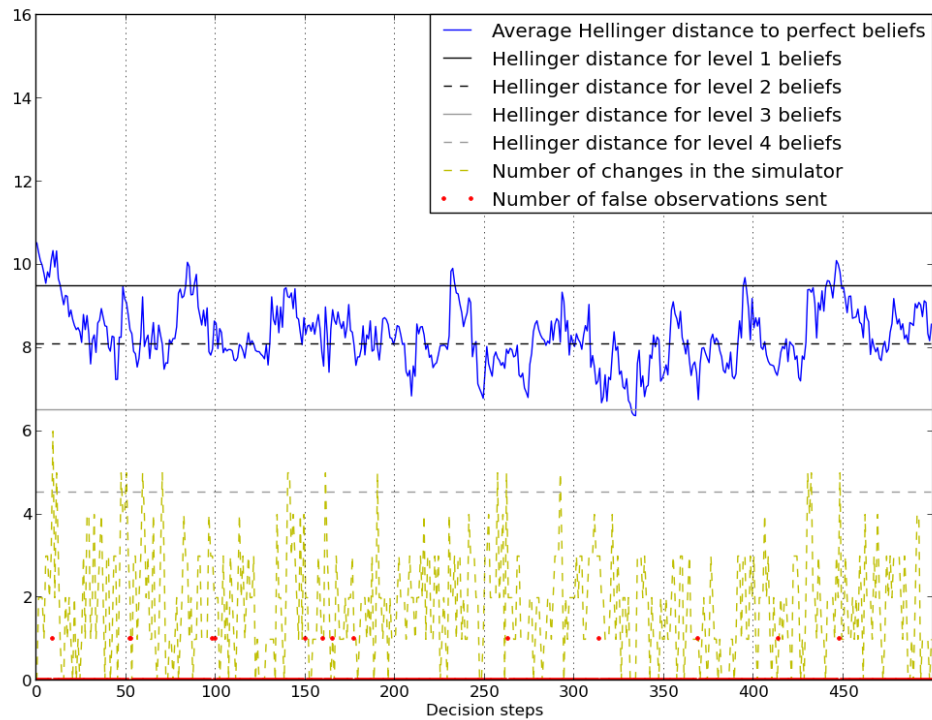
Figure 10.9: Average Hellinger distance to the perfect belief state for the outdoor environment in the dynamic case

Figure 10.10: Number of communications per robot for the outdoor environment
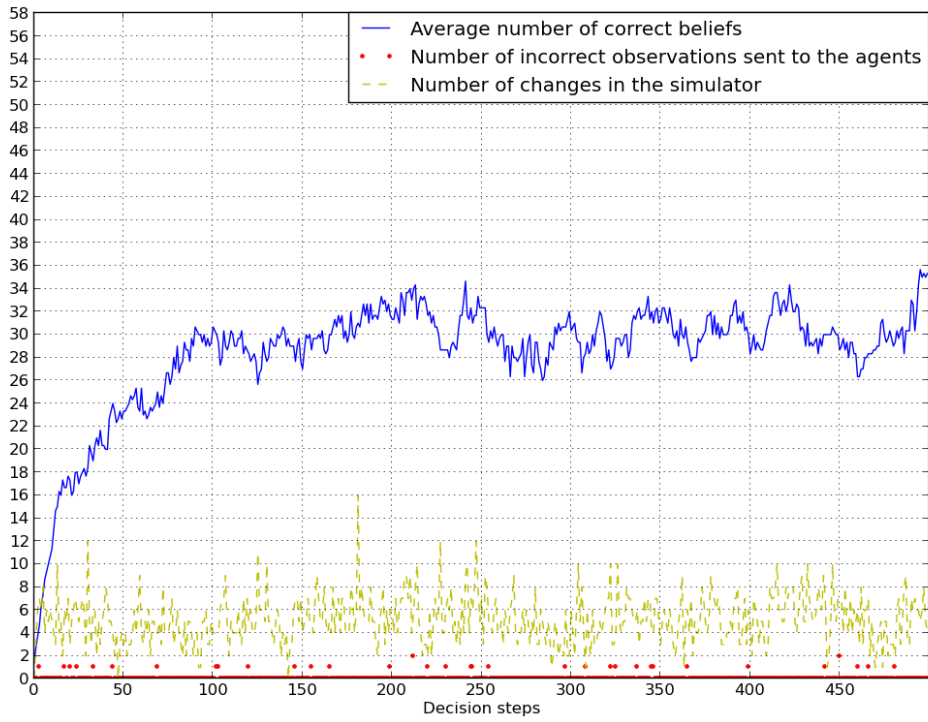
Figure 10.11: Average number of correct beliefs for the Élysée Palace environment

10.4.2 Evaluating the communication

Figure 10.13 presents the number of communications sent by the agents.
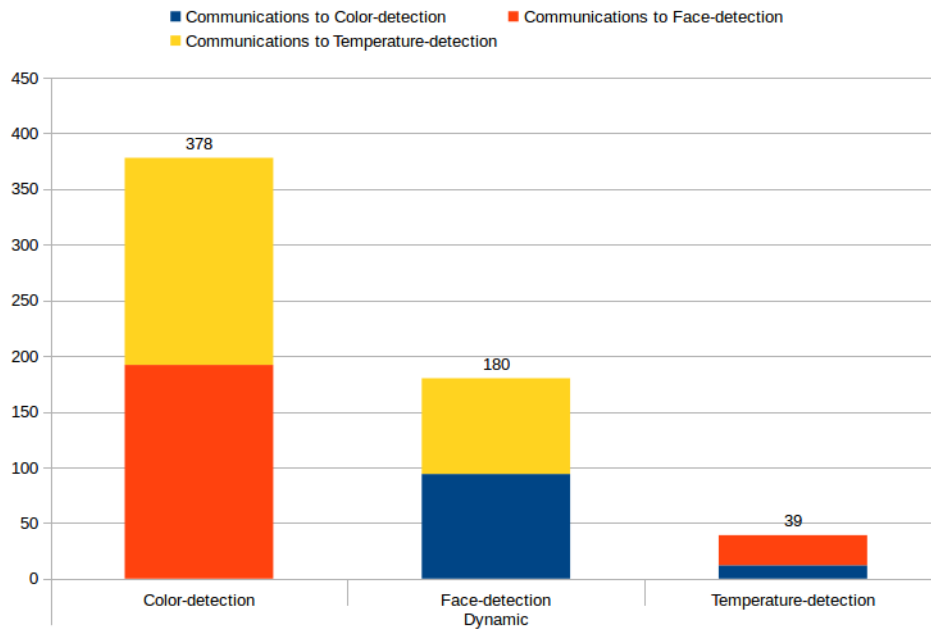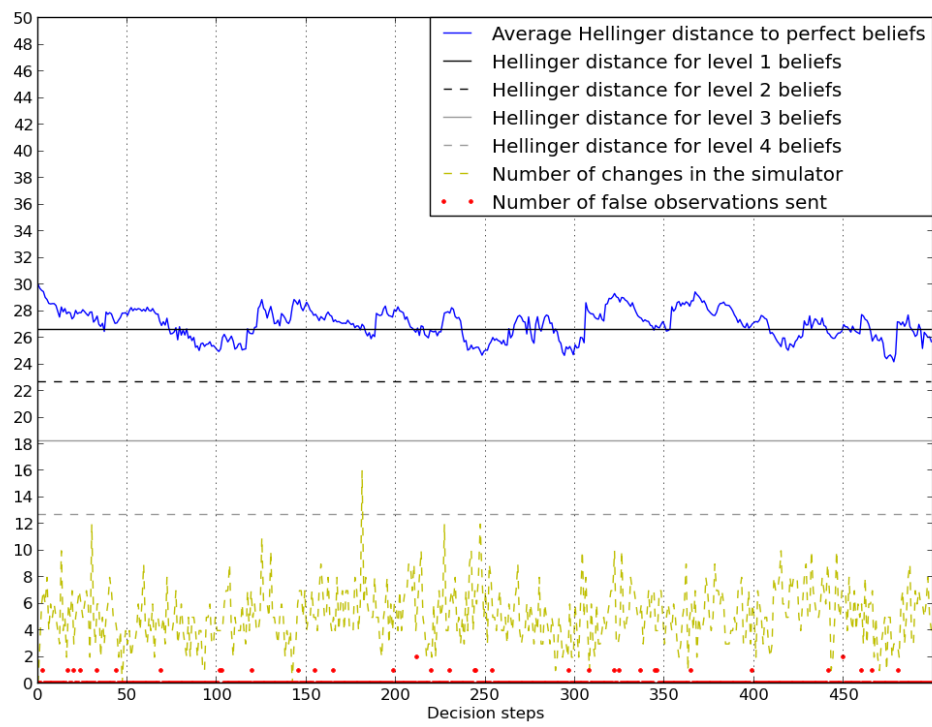
Figure 10.12: Average Hellinger distance to the perfect belief state for the Élysée Palace environment
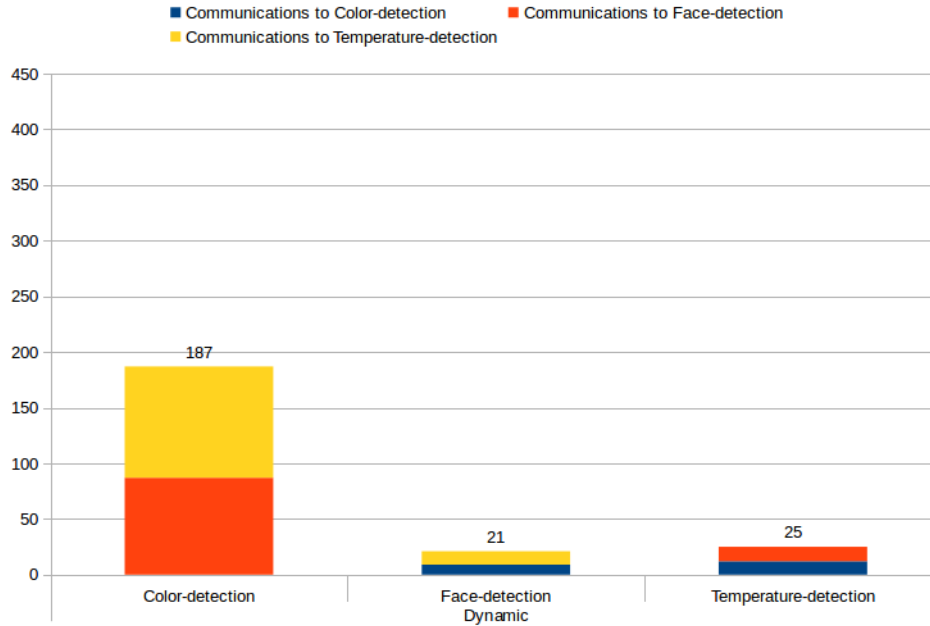
Figure 10.13: Number of communications per robot for the Élysée Palace environment

The agents used the communicate action 233 times, that is to say 78 times per agent in average. These numbers are compared to previous communication cost and environments in table 10.4.

	Medium communication cost		High communication cost	
	Total	Average	Total	Average
House environment	644	214	485	162
One-way environment	588	196	574	191
Outdoor environment	629	210	597	199
Élysée Palace environment	305	102	233	78

Table 10.4: Total and average number of communication in different environments, in static and dynamic cases, for medium and high communication costs

Unsurprisingly, the agents have an extremely low communication rate in this environment. Indeed, the structure of the environment allows easy and cheap exploration.

10.4.3   Analysis

This chapter confirms that the system is not efficient in large environments. A lot of beliefs are incorrect, probably outdated. The agents don't manage to correct them quickly enough, and their beliefs are not precise. As we already mentioned, more experiments would be useful to determine if, as we believe, more agents in the system could produce better results.

10.5   Global analysis and conclusion

In this chapter we evaluated the MAPING framework in various environments with a large communication cost. This experiment has been performed to analyze the impact of a highly constrained communication on the system's behavior and its efficiency. Results show that the efficiency of the system are slightly damaged by the communication cost, but remains very satisfactory and reliable for all the small environments. Even if the system communicates globally less than with the medium communication cost, it has proved being able to adapt itself to the need of the environment and to increase the amount of communications when needed. For instance in the outdoor environment where the transitions are costly, the system communicates almost as much as previously.

This chapter, among with the two previous ones, confirm the work hypothesis that the communication should be reduced for the system to be efficient. It also proves that MAPING is able to be efficient under constrained communication and to maintain reliable beliefs states.

Chapter 11

Conclusion of part III

In this part we evaluated the efficiency of the MAPING framework in various environments. All the environments are based on real applications and have been designed to model real constraints. The evaluation has been performed with three communication costs: a low, a medium and a high communication cost. The system has not been very efficient with the low communication cost due to the communication and the spreading of incorrect beliefs. We concluded that an almost-free communication is not desirable for MAPING to work properly.

Otherwise, MAPING managed to prove its efficiency in various environments with the medium and high communication costs. It is able to reduce the amount of communication depending on the needs: in environments where exploration is easy and cheap, the system reduces the number of communications. On the contrary, in environments where exploration is very costly, the system communicates more to overcome this cost as in outdoor environments.

For all costs, the system had some limitations with big environment and should be reevaluated. More agents may solve the problem. However, with the current implementation, it is very hard to solve the POMDP required for 4 or more agents. Some optimization in the implementation may make it possible to raise the number of agents to 4 or 5, but probably no more.

Part IV

Conclusion and perspectives

Chapter 12

Synthesis

Contents

In this thesis, we addressed the problem of information gathering with a multiagent system, and more specifically *event exploration*. We defined the event exploration as the process of traveling across a topologically known environment with the goal of detecting events happening in this environment. An event is defined as a change in the environment's features of interest. We proposed a framework in which agents assess the relevance of information, estimate other agents' beliefs and decide about the next action to perform. In this chapter we presents a summary of our research and analyze the advantages and drawbacks of our approach.

12.1 Agent-based relevance

Information relevance is a topic well studied in Information Retrieval systems. However, as we figured out in chapter 2, techniques used in information retrieval systems cannot be applied to multiagent systems. We proposed a framework to enable quantitative agent-based relevance by suggesting a bounded degree of relevance. This degree of relevance is a compromise between novelty and soundness of a piece of information.

This degree of relevance has been successfully used in our decision framework, but can be easily applied to other models. We didn't conclude about the convexity of the degree of relevance, which could limit its use in some applications where convexity is required.

12.2 Multiagent planning for information gathering

Event exploration is a very specific kind of exploration in which the agents need to deal with dynamic environments. In this thesis we developed a framework, called MAPING (for Multi-Agent Planning for INformation Gathering), which enables a team of heterogeneous agents to perform efficient event exploration in a fully-decentralized way while limiting their communications. This is to our knowledge the first attempt in the planning under uncertainty community to propose a model that deal with this kind of exploration in a fully decentralized way and with limited communications.

Our model rely on agent's beliefs and the degree of relevance we defined to compute a communication and exploration policy. We also suggested a smoothing mechanism that uses a contraction mapping to artificially degrade the agents' beliefs and force them to explore again features already explored. In that way the system can deal with the dynamicity of the environment.

The MAPING framework has been proved to be efficient in diverse environments with no modification of the offline computation and very few modifications of the online part. It has proved its efficiency in cases where the communication is costly. However since the number of belief states increases exponentially with the number of features and the number of agents, MAPING is not scalable to systems with a high number of agents.

12.3 Possible range of applications

Contributions have led to the implementation of a Java solver for MAPING model and some C++ ROS nodes for the execution. We used this implementation to solve surveillance-type problems, but our approach can be used in various number of situations where information gathering is involved. Instance industrial maintenance is another example of application that require event exploration. Our framework could be for instance combined with the gas-detection model developed by Loufti et al. in [Loutfi et al., 2009] to turn their monoagent system into a multiagent one and detect gas leak in industrial infrastructures.

Autonomous systems are more and more studied for search and rescue applications, and our model can also be proposed for such problems. Indeed, information gathering is a big part of the search and rescue problem and using autonomous robots to explore and collect information could make the work of the rescue workers much easier and safer. However, in this kind of applications, the system will need to operate collaboratively with humans. This topic has not been considered during this thesis and more research is necessary, as discussed in Chapter 13.

Chapter 13

Perspectives

Contents

13.1 Short-term perspectives

The main immediate drawback of our degree of relevance is the fact that it is not proved to be convex. A deeper mathematical study could conclude on this specific point. The results of such a study could guide new researches to solve more easily the MAPING POMDP.

In our implementation of MAPING , we used a single Value Iteration algorithm to perform the offline resolution. We chose this algorithm due to its simple and fast implementation. However this choice limited us to system with a maximum of 3 agents due to the computation time on our resources. To perform the experiments with a higher number of agents, and so to evaluate MAPING on bigger environments, bigger system should be considered. Then other techniques should be considered. By using discretization, we transformed our POMDP into a discretized MDP. A trail could be the studies of Poupart [Poupart, 2005] and Barry et al. [Barry et al., 2011] about solving large MDPs.

13.2 Mid-term perspectives

In this thesis, we considered only Bayesian belief updates. An immediate conclusion of this assumption is that the system is not fault-tolerant. If one or several robots present a failure (typically a faulty sensor), the observations sent by this robot will be taken in the update as observations sent by others. The beliefs of the system will be degraded due to this failure. To limit the impact of such a situation, it could be interesting to study reputation systems, as those presented in [Jøsang et al., 2007], and to integrate them in MAPING . By associating a trust degree to the agents, agents that have sent incorrect observations would see their impact decrease in the update, while trustworthy agents would see their impact increase.

Another evolution to the model would be to consider the gain of information during the travel from a zone to another in addition to the gain of information at the targeted zone. Indeed, in some cases the robots have to travel through some zones in order to reach their target. Since sensing can also be performed while moving it seems logical to integrate informations gathered on the way to the reward in order to improve the paths travelled along. However the robot would not be dedicated to exploration while travelling and the data collected that way shouldn't be granted the same importance as data collected from complete exploration of a zone.

13.3 Long-term perspectives

The biggest issue remaining with our framework concerns heterogeneous agents. To update their beliefs after receiving an observation from another agent, the agents need the observation function of the emitter of the observation. For homogeneous agents we can safely assume that all agents have the same observation function and can use their own. However this assumption is not longer valid with heterogeneous agents, since the observation function depends on the agent's capacities. Therefore it is mandatory to transmit the observation function to the agent that wants to update its beliefs. This issue remains open and deeper studies are needed to find a solution and to integrate it in MAPING .

Different evolutions are possible to extend the MAPING framework. One that we find particularly interesting is to includes Human in the model. In our work, we only considered the system as autonomous and fully detached on any human intervention. However, in a lot of applications cited as possible uses, humans are actually in the system, and they cannot and should not be replaced by robots. In search and rescue applications, rescue workers should be warned when

a victim is found. In surveillance application a human operator should decide of the action to perform is an intrusion or a fire is detected. In addition to this role of referent, humans can also collaborate with robots in the exploration and exchange observations with the robots, as the robots do. For all these situation, the way to incorporate humans in the loop need to be studied and could lead to efficient, complete and very innovating systems.

Another evolution concerns the actions that the robots can perform. In our work, we considered only epistemic actions, that is to say actions that affect the belief states of the robots but not the environment. This reduce the possibilities of exploration and the range of applications. Indeed we considered that a robot cannot open a door itself, that it cannot extinguish a start of fire, etc. Transforming the model to mix epistemic and non-epistemic actions would increase the difficulty of solving it but could make it applicable to more situations.

Finally, research could be carried on to mix MAPING with semantic information fusion. We only considered in this work that each robot has all the features in its belief state. But we could complete the belief state with an probabilistic ontology and use this ontology to infer higher information from the feature detected. This could also enable the robots to have different low level features in their belief state and to combine all those features to different higher level features, usable by another robot or a human operator.

Bibliography

[Agmon, 2010] Agmon, N. (2010). On events in multi-robot patrol in adversarial environments. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[Agmon et al., 2011] Agmon, N., Kaminka, G. A., and Kraus, S. (2011). Multi-robot adversarial patrolling facing a full-knowledge opponent. *Journal of Artificial Intelligence*, pages 887–916.

[Agmon et al., 2008a] Agmon, N., Kraus, S., and Kaminka, G. A. (2008a). Multi-robot perimeter patrol in adversarial settings. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Agmon et al., 2008b] Agmon, N., Sadov, V., Kaminka, G. A., and Kraus, S. (2008b). The impact of adversarial knowledge on adversarial planning in perimeter patrol. In *Proceedings of the 7th international joint conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[Allen and Darwiche, 2002] Allen, D. and Darwiche, A. (2002). New advances in inference by recursive conditioning. In *Proceedings of the 19th conference on Uncertainty in Artificial Intelligence*.

[Amato et al., 2006] Amato, C., Bernstein, D. S., and Zilberstein, S. (2006). Solving pomdps using quadratically constrained linear programs. In *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[Amigoni et al., 2012] Amigoni, F., Basilico, N., and Li, A. Q. (2012). How much worth is coordination of mobile robots for exploration in search and rescue? In *RoboCup 2012: Robot Soccer World Cup XVI*.

[and, 2000] and, M. R. E. (2000). Theoretical underpinnings of situation awareness: A critical review. *Situation awareness analysis and measurement*, pages 3–32.

[Araya-Lopez et al., 2010] Araya-Lopez, M., Buffet, O., Thomas, V., and Charpillet, F. (2010). A pomdp extension with belief-dependent rewards. In *Advances in Neural Information Processing Systems (NIPS)*.

[Aulinas et al., 2008] Aulinas, J., Petillot, Y., Salvi, J., and Lladó, X. (2008). The slam problem: A survey. In *Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*.

[Aurenhammer, 1991] Aurenhammer, F. (1991). Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys*, pages 345–405.

Bibliography

[Bachrach et al., 2009] Bachrach, A., He, R., , and Roy, N. (2009). Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, pages 217–228.

[Bagnell et al., 2003] Bagnell, J. A., Kakade, S. M., Schneider, J. G., and Ng, A. Y. (2003). Policy search by dynamic programming. In *Advances in neural information processing systems*.

[Bai et al., 2011] Bai, H., Hsu, D., Lee, W. S., , and Ngo, V. A. (2011). Monte-carlo value iteration for continuous-state pomdps. In *Algorithmic foundations of robotics IX*.

[Bajcsy, 1988] Bajcsy, R. (1988). Active perception. In *Proceedings of the IEEE*.

[Barry et al., 2011] Barry, J., Kaelbling, L. P., and Lozano-Pérez, T. (2011). Deth*: Approximate hierarchical solution of large markov decision processes. In *Proceedings of the 25th Conference on Artificial intelligence (AAAI)*.

[Basilico et al., 2009] Basilico, N., Gatti, N., and Amigoni, F. (2009). Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[Basilico et al., 2012] Basilico, N., Gatti, N., and Amigoni, F. (2012). Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artificial Intelligence*, pages 78–123.

[Basilico et al., 2010] Basilico, N., Rossignoli, D., Gatti, N., and Amigoni, F. (2010). A game-theoretical model applied to an active patrolling camera. In *Proceedings of the International Conference on Emerging Security Technologies (EST)*.

[Bautin et al., 2011] Bautin, A., Simonin, O., and Charpillet, F. (2011). Towards a communication free coordination for multi-robot exploration. In *Proceedings of the 8th National Conference on Control Architectures of Robots*.

[Bayes, 1763] Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions*, pages 370–418.

[Becker, 2004] Becker, R. (2004). Solving transition independent decentralized markov decision processes. *Computer Science Department Faculty Publication Series*, pages 423–455.

[Bellenger, 2013] Bellenger, A. (2013). *Semantic Decision Support for Information Fusion Applications*. PhD thesis, Institut National des Sciences Appliquées (INSA) de Rouen.

[Bellman, 1957] Bellman, R. (1957). A markovian decision process. Technical report, DTIC Document.

[Bellman and Dreyfus, 1962] Bellman, R. and Dreyfus, S. (1962). Applied dynamic programming. Technical report, RAND Corporation.

[Bernstein et al., 2002] Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. (2002). The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, pages 819–840.

[Bertoli et al., 2001] Bertoli, P., Cimatti, A., Roveri, M., and Traverso, P. (2001). Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*.

[Beynier and Mouaddib, 2005] Beynier, A. and Mouaddib, A.-I. (2005). A polynomial algorithm for decentralized markov decision processes with temporal constraints. In *Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[Blum and Furst, 1997] Blum, A. and Furst, M. (1997). Fast planning through planning graph analysis. *Artificial Intelligence*, pages 281–300.

[Borlund, 2003] Borlund, P. (2003). The concept of relevance in ir. *Journal of the American Society for Information Science and Technology*, pages 913–925.

[Boutilier et al., 1999] Boutilier, C., Dean, T., and Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, pages 1–94.

[Boutilier et al., 1995] Boutilier, C., Dearden, R., and Goldszmidt, M. (1995). Exploiting structure in policy construction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*.

[Boutilier et al., 2000] Boutilier, C., Dearden, R., and Goldszmidt, M. (2000). Stochastic dynamic programming with factored representations. *Artificial Intelligence*, pages 49–107.

[Boutilier and Poole, 1996] Boutilier, C. and Poole, D. (1996). Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of the National Conference on Artificial Intelligence*.

[Burgard et al., 1999] Burgard, W., Fox, D., Jans, H., Matenar, C., and Thrun, S. (1999). Sonar-based mapping with mobile robots using em. In *Proceedings of the International Conference on Machine Learning*.

[Burgard et al., 2002] Burgard, W., Moors, M., and Schneider, F. (2002). Collaborative exploration of unknown environments with teams of mobile robots. In *Advances in plan-based control of robotic agents*.

[Burgard et al., 2005] Burgard, W., Moors, M., Stachniss, C., and Schneider, F. (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, pages 376–386.

[Béziau, 1997] Béziau, J.-Y. (1997). What is many-valued logic? In *Proceedings of the 27th International Symposium on Multiple-Valued Logic*.

[Cappé, 2009] Cappé, O. (2009). Online sequential monte carlo em algorithm. In *Proceedings of the 15th Workshop on Statistical Signal Processing*.

[Carillo et al., 2012] Carillo, H., Reid, I., and Castellanos, J. (2012). On the comparison of uncertainty criteria for active slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Carlone et al., 2010] Carlone, L., Du, J., Ng, M. K., Bona, B., and Indri, M. (2010). An application of kullback-leibler divergence to active slam and exploration with particle filters. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[Carlone et al., 2014] Carlone, L., Du, J., Ng, M. K., Bona, B., and Indri, M. (2014). Active slam and exploration with particle filters using kullback-leibler divergence. *Journal of Intelligent and Robotic Systems*, pages 291–311.

[Carroll et al., 2005] Carroll, D., Nguyen, C., Everett, H., and Frederick, B. (2005). Development and testing for physical security robots. In *Proceedings of the International Society for Optics and Photonics*.

[Cassandra et al., 1997] Cassandra, A., Littman, M. L., and Zhang, N. (1997). Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In *Proceedings of the 13th conference on Uncertainty in artificial intelligence*.

[Cassandra et al., 1994] Cassandra, A. R., Kaelbling, L. P., and Littman, M. L. (1994). Acting optimally in partially observable stochastic domains. In *Proceedings of the 12th National Conference on Artificial intelligence (AAAI)*.

[Chanel et al., 2012] Chanel, C. P. C., Teichteil-Königsbuch, F., and Lesire, C. (2012). Pomdp-based online target detection and recognition for autonomous uavs. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*.

[Chanel et al., 2013] Chanel, C. P. C., Teichteil-Königsbuch, F., and Lesire., C. (2013). Multi-target detection and recognition by uavs using online pomdps. In *Proceedings of the 27th Conference on Artificial intelligence (AAAI)*.

[Chapman, 1987] Chapman, D. (1987). Planning for conjunctive goals. *Artificial Intelligence*, pages 333–377.

[Chavira and Darwiche, 2005] Chavira, M. and Darwiche, A. (2005). Compiling bayesian networks with local structure. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*.

[Cheng and Druzdzel, 2000] Cheng, J. and Druzdzel, M. J. (2000). Ais-bn: An adaptive importance sampling algorithm for evidential reasoning in large bayesian networks. *Journal of Artificial Intelligence Research*, pages 155–188.

[Chevaleyre, 2004] Chevaleyre, Y. (2004). Theoretical analysis of the multi-agent patrolling problem. In *Proceedings of the International Conference on Intelligent Agent Technology (IAT)*.

[Chu et al., 2007] Chu, H.-N., Glad, A., Simonin, O., Sempe, F., Drogoul, A., and Charpillet, F. (2007). Swarm approaches for the patrolling problem, information propagation vs. pheromone evaporation. In *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*.

[Chung et al., 2004] Chung, T. H., andJoel W. Burdick, V. G., and Murray, R. M. (2004). On a decentralized active sensing strategy using mobile sensor platforms in a network. In *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC)*.

[Conitzer and Sandholm, 2006] Conitzer, V. and Sandholm, T. (2006). Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*.

[Cooper, 1990] Cooper, G. F. (1990). The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, pages 393–405.

[Corff et al., 2011] Corff, S. L., Fort, G., and Moulines, E. (2011). Online expectation maximization algorithm to solve the slam problem. In *Statistical Signal Processing Workshop (SSP)*.

[Darwiche, 2001] Darwiche, A. (2001). Recursive conditioning. *Artificial Intelligence*, pages 5–41.

[Darwiche, 2008] Darwiche, A. (2008). *Bayesian Networks*, chapter 11, pages 467–509. Elsevier.

[Davison and Murray, 2002] Davison, A. J. and Murray, D. W. (2002). Simultaneous localization and map-building using active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 865–880.

[Dean and Kanazawa, 1989] Dean, T. and Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, pages 142–150.

[Dechter, 1999] Dechter, R. (1999). Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, pages 41–85.

[Dempster, 1967] Dempster, A. P. (1967). Upper and lower probabilities induced by a multivalued mapping. *The annals of Mathematical statistics*, pages 325–339.

[Doshi and Gmytrasiewicz, 2009] Doshi, P. and Gmytrasiewicz, P. J. (2009). Monte carlo sampling methods for approximating interactive pomdps. *Journal of Artificial Intelligence Research*, pages 297–337.

[Doshi and Perez, 2008] Doshi, P. and Perez, D. (2008). Generalized point based value iteration for interactive pomdps. In *Proceedings of the 23rd Conference on Artificial intelligence (AAAI)*.

[Doshi et al., 2009] Doshi, P., Zeng, Y., and Chen, Q. (2009). Graphical models for interactive pomdps: representations and solutions. *Autonomous Agents and Multi-Agent Systems*, pages 376–416.

[Dubois, 2007] Dubois, D. (2007). Uncertainty theories: a unified view. In *Proceedings of the IEEE Cybernetic Systems Conference*.

[Dubois and Prade, 1998] Dubois, D. and Prade, H. (1998). Possibility theory: qualitative and quantitative aspects. *Quantified representation of uncertainty and imprecision*, pages 169–226.

[Eidenberger and Scharinger, 2010] Eidenberger, R. and Scharinger, J. (2010). Active perception and scene modeling by planning with probabilistic 6d object poses. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[Elmaliach et al., 2009] Elmaliach, Y., Agmon, N., and Kaminka, G. A. (2009). Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, pages 293–320.

[Elmaliach et al., 2008] Elmaliach, Y., Shiloni, A., and Kaminka, G. A. (2008). A realistic model of frequency-based multi-robot polyline patrolling. In *Proceedings of the 7th international Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[Endsley, 1995] Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, pages 32–64.

[Fagin et al., 1995] Fagin, R., Halpern, J. Y., Moses, Y., , and Vardi, M. Y. (1995). *Reasoning about knowledge*. MIT Press Cambridge.

[Feder et al., 1999] Feder, H. J. S., Leonard, J. J., and Smith, C. M. (1999). Adaptive mobile robot navigation and mapping. *he International Journal of Robotics Research*, pages 650–668.

[Fedorov, 1972] Fedorov, V. V. (1972). *Theory of optimal experiments.* Academic Press Inc.

[Fikes and Nilsson, 1972] Fikes, R. E. and Nilsson, N. J. (1972). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, pages 189–208.

[Floreano and Mondada, 1994] Floreano, D. and Mondada, F. (1994). Active perception, navigation, homing, and grasping: an autonomous perspective. In *From Perception to Action Conference.*

[Floridi, 2008] Floridi, L. (2008). Understanding epistemic relevance. *Erkenntnis*, pages 69–92.

[Friedman and Halpern, 1995] Friedman, N. and Halpern, J. Y. (1995). Plausibility measures: a user's guide. In *Proceedings of the 11th conference on Uncertainty in artificial intelligence.*

[Friedman and Halpern, 2001] Friedman, N. and Halpern, J. Y. (2001). Plausibility measures and default reasoning. *Journal of the ACM*, pages 648–685.

[Fudenberg and Tirole, 1991] Fudenberg, D. and Tirole, J. (1991). *Game Theory.* The MIT Press.

[Gatti, 2008] Gatti, N. (2008). Game theoretical insights in strategic patrolling: Model and algorithm in normal-form. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI).*

[Geffner and Bonet, 1998] Geffner, H. and Bonet, B. (1998). Solving large pomdps using real time dynamic programming. In *Proceedings of the 15th AAAI Fall Symposium on POMDPs.*

[Ghallab et al., 2004] Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated planning: theory & practice.* Elsevier.

[Glad et al., 2008] Glad, A., Simonin, O., Buffet, O., and Charpillet, F. (2008). Theoretical study of ant-based algorithms for multi-agent patrolling. In *Proceedings of the 18th European Conference on Artificial Intelligence including Prestigious Applications of Intelligent Systems (PAIS).*

[Gmytrasiewicz and Doshi, 2005] Gmytrasiewicz, P. J. and Doshi, P. (2005). A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, pages 49–79.

[Goldman and Zilberstein, 2003] Goldman, C. V. and Zilberstein, S. (2003). Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the 2th International Conference on Autonomous Agents and Multiagent Systems (AAMAS).*

[Goldman and Zilberstein, 2004] Goldman, C. V. and Zilberstein, S. (2004). Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, pages 143–174.

[Grice, 1975] Grice, H. P. (1975). *Syntax and semantics*, chapter Logic and Conversation, pages 41–58. New York: Academic Press.

[Guestrin et al., 2003] Guestrin, C., Koller, D., Parr, R., , and Venkataraman, S. (2003). Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, pages 399–468.

[Guo, 2003] Guo, A. (2003). Decision-theoretic active sensing for autonomous agents. In *Proceedings of the 2th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[Guo and Qu, 2004] Guo, Y. and Qu, Z. (2004). Coverage control for a mobile robot patrolling a dynamic and uncertain environment. In *Proceedings of the 5th World Congress on Intelligent Control and Automation (WCICA)*.

[Halpern, 2003a] Halpern, J. Y. (2003a). *Reasoning about Uncertainty*. The MIT Press.

[Halpern, 2003b] Halpern, J. Y. (2003b). *Reasoning about Uncertainty*, chapter Belief Revision, pages 97–104. The MIT Press.

[Harmelen et al., 2008] Harmelen, F. V., Lifschitz, V., and Porter, B. (2008). *Handbook of Knowledge Representation (Foundations of Artificial Intelligence)*. Elsevier Science.

[Hauskrecht, 2000] Hauskrecht, M. (2000). Value-function approximations for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, pages 33–94.

[Heijenoort, 1967] Heijenoort, J. V. (1967). *From Frege to Gödel: a source book in mathematical logic*. Harvard University Press.

[Hintikka, 1962] Hintikka, J. (1962). *Knowledge and belief*. Cornell University Press.

[Holmes et al., 2009] Holmes, S. A., Klein, G., , and Murray, D. W. (2009). An o(n2) square root unscented kalman filter for visual simultaneous localization and mapping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1251–1263.

[Howard, 1960] Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. The MIT Press.

[Hwang et al., 2009] Hwang, K.-S., Lin, J.-L., and Huang, H.-L. (2009). Cooperative patrol planning of multi-robot systems by a competitive auction system. In *Proceedings of the International Joint Conference ICCAS-SICE*.

[Izadi and Precup, 2005] Izadi, M. T. and Precup, D. (2005). Using rewards for belief state updates in partially observable markov decision processes. *Lecture Notes in Computer Scienc, Machine Learning: ECML 2005*, 3720:593–600.

[Jensen et al., 1990] Jensen, F., Lauritzen, S., and Olsen, K. (1990). Bayesian updating in recursive graphical models by local computation. *Computational Statistics Quarterly*, pages 269–282.

[Jensfelt et al., 2006] Jensfelt, P., Kragic, D., Folkesson, J., and Bjorkman, M. (2006). A framework for vision based bearing only 3d slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Ji and Carin, 2007] Ji, S. and Carin, L. (2007). Cost-sensitive feature acquisition and classification. *Pattern Recognition*, page 1474–1485.

[Jordan et al., 1999] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul., L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, pages 183–233.

[Jøsang et al., 2007] Jøsang, A., Ismail, R., and Boyd, C. (2007). A survey of trust and reputation systems for online service provision. *Decision support system*, pages 618–644.

[Kiefer, 1974] Kiefer, J. (1974). General equivalence theory for optimum designs (approximate theory). *The annals of Statistics*, pages 849–879.

[Ko et al., 2003] Ko, J., Stewart, B., Fox, D., Konolige, K., and Limketkai, B. (2003). A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[Kollar and Roy, 2008] Kollar, T. and Roy, N. (2008). Trajectory optimization using reinforcement learning for map exploration. *The International Journal of Robotics Research*, page 175–196.

[Koller and Parr, 2000] Koller, D. and Parr, R. (2000). Policy iteration for factored mdps. In *Proceedings of the 16th conference on Uncertainty in artificial intelligence*.

[Kontitsis et al., 2013] Kontitsis, M., Theodorou, E. A., and Todorov, E. (2013). Multi-robot active slam with relative entropy optimization. In *American Control Conference (ACC)*.

[Kuhn, 1955] Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, page 83–97.

[Kullback and Leibler, 1951] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, page 79–86.

[Kurniawati et al., 2008] Kurniawati, H., Hsu, D., and Lee, W. S. (2008). Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. *Robotics: Science and Systems*, pages 65–72.

[Larkin and Dechter, 2003] Larkin, D. and Dechter, R. (2003). Bayesian inference in the presence of determinism. In *Proceedings of the 9+th International Workshop on Artificial Intelligence and Statistics*.

[Lauritzen and Spiegelhalter, 1988] Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, page 157–224.

[Laverny and Lang, 2005a] Laverny, N. and Lang, J. (2005a). From knowledge-based programs to graded belief-based programs, part i: On-line reasoning*. *Synthese*, page 277–321.

[Laverny and Lang, 2005b] Laverny, N. and Lang, J. (2005b). From knowledge-based programs to graded belief-based programs, part ii: off-line reasoning. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*.

[Leung et al., 2006] Leung, C., Huang, S., and Dissanayake, G. (2006). Active slam using model predictive control and attractor based exploration. In *IEEE/RJS International Conference on Intelligent Robots and Systems*.

[Lewis et al., 2004] Lewis, P. J., Torrie, M. R., and Omilon, P. M. (2004). Applications suitable for unmanned and autonomous missions utilizing the tactical amphibious ground support (tags) platform. *Defense and Security*, page 508–519.

[Lilienthal et al., 2007] Lilienthal, A. J., Loutfi, A., Blanco, J. L., Galindo, C., and Gonzalez, J. (2007). A rao-blackwellisation approach to gdm-slam – integrating slam and gas distribution mapping. In *Proceedings of the European Conference on Mobile Robots (ECMR)*.

[Littman, 1994a] Littman, M. L. (1994a). Memoryless policies: Theoretical limitations and practical results. In *From Animals to Animats 3: Proceedings of the third international conference on simulation of adaptive behavior*.

[Littman, 1994b] Littman, M. L. (1994b). The witness algorithm: Solving partially observable markov decision processes. Technical report, Providence, RI, USA.

[Littman et al., 2001] Littman, M. L., Majercik, S. M., and Pitassi, T. (2001). Stochastic boolean satisfiability. *Journal of Automated Reasoning*, page 251–296.

[Loutfi et al., 2009] Loutfi, A., Coradeschi, S., Lilienthal, A. J., and Gonzalez, J. (2009). Gas distribution mapping of multiple odour sources using a mobile robot. *Robotica*, pages 311–319,.

[Lovejoy, 1991] Lovejoy, W. S. (1991). Computationally feasible bounds for partially observed markov decision processes. *Operations research*, page 162–175.

[Machado et al., 2003] Machado, A., Ramalho, G., Zucker, J.-D., and Drogoul., A. (2003). Multi-agent patrolling: An empirical analysis of alternative architectures. *Multi-Agent-Based Simulation II*, pages 155–170.

[Markov, 1960] Markov, A. (1960). *The Theory of Algorithms*. American Mathematical Society Translations.

[Martins-Filho and Macau, 2007] Martins-Filho, L. S. and Macau, E. E. (2007). Patrol mobile robots and chaotic trajectories. *Mathematical problems in engineering*.

[Matignon et al., 2012] Matignon, L., Jeanpierre, L., , and Mouaddib, A.-I. (2012). Distributed value functions for multi-robot exploration: a position paper. In *Proceedings of the Multi-Agent Sequential Decision Making in Uncertain Multi-Agent Domain Workshop (MSDM)*.

[Menezes et al., 2006] Menezes, T., Tedesco, P., and Ramalho., G. (2006). Negotiator agents for the patrolling task. In *Advances in Artificial Intelligence IBERAMIA-SBIA*.

[Meuleau et al., 1999] Meuleau, N., Kim, K.-E., Kaelbling, L. P., , and Cassandra, A. R. (1999). Solving pomdps by searching the space of finite policies. In *Proceedings of the 15th conference on Uncertainty in artificial intelligence*.

[Mihaylova et al., 2003] Mihaylova, L., Lefebvre, T., Bruyninckx, H., Gadeyne, K., and Schutter, J. D. (2003). A comparison of decision making criteria and optimization methods for active robotic sensing. *Numerical Methods and Applications*, page 316–324.

[Mihaylova et al., 2002] Mihaylova, L., Lefebvre, T., Herman Bruyninckx, K. G., and Schutter., J. D. (2002). Active sensing for robotics - a survey. In *Proceedings of the 5 th International Conference On Numerical Methods and Applications*.

[Mises, 1957] Mises, R. V. (1957). *Probability, statistics, and truth*. Courier Dover Publications.

[Moeschler, 2007] Moeschler, J. (2007). *Language and Speech Engineering*, chapter ntroduction to pragmatics, pages 51–68. EPFL Press.

[Montemerlo and Thrun, 2007] Montemerlo, M. and Thrun, S. (2007). *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics.* Springer Tracts in Advanced Robotics.

[Montemerlo et al., 2002] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002). Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the 18th National Conference on Artificial intelligence (AAAI).*

[Morari et al., 2014] Morari, M., Garcia, C. E., and Prett, D. M. (2014). Model predictive control: theory and practice. In *Proceedings of the Workshop on Model Based Process Control.*

[Moravec and Elfes, 1985] Moravec, H. P. and Elfes, A. (1985). High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation.*

[Neapolitan, 1990] Neapolitan, R. E. (1990). *Probabilistic Reasoning in Expert Systems: Theory and Algorithms.* Wiley-Interscience.

[Newell et al., 1959] Newell, A., Shaw, J. C., and Simon, H. A. (1959). Report on a general problem-solving program. In *IFIP Congress.*

[Nikulin, 2015] Nikulin, M. S. (2015). Hellinger distance. Encyclopedia of Mathematics.

[Oliehoek, 2012] Oliehoek, F. A. (2012). *Reinforcement Learning: State of the Art*, chapter Decentralized POMDPs, pages 471–503. Springer Berlin Heidelberg.

[Oliehoek et al., 2008] Oliehoek, F. A., Spaan, M. T., Whiteson, S., and Vlassis, N. (2008). Exploiting locality of interaction in factored dec-pomdps. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems.*

[Oxford,] Oxford. Patrol. The Oxford online dictionary. Accessed: 2015-01-21.

[Palacios and Geffner, 2009] Palacios, H. and Geffner, H. (2009). Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research*, pages 623–675.

[Paruchuri et al., 2008] Paruchuri, P., Pearce, J. P., Marecki, J., Tambe, M., Ordonez, F., and Kraus., S. (2008). Playing games for security: an efficient exact algorithm for solving bayesian stackelberg games. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems.*

[Paruchuri et al., 2009] Paruchuri, P., Pearce, J. P., Marecki, J., Tambe, M., Ordónez, F., and Kraus, S. (2009). Coordinating randomized policies for increasing security of agent systems. *Information Technology and Management*, pages 67–79.

[Paruchuri et al., 2007] Paruchuri, P., Pearce, J. P., Tambe, M., Ordonez, F., and Kraus., S. (2007). An efficient heuristic approach for security against multiple adversaries. In *Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems.*

[Paruchuri et al., 2006] Paruchuri, P., Tambe, M., Ordóñez, F., and Kraus., S. (2006). Security in multiagent systems by policy randomization. In *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems,.*

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers Inc.

[Penberthy and Weld, 1992] Penberthy, J. S. and Weld, D. S. (1992). Ucpop: A sound, complete, partial order planner for adl. In *Proceedings of the 3rd International Conference on the Principles of Knowledge Representation and Reasoning (KR)*.

[Petrick and Bacchus, 2002] Petrick, R. P. and Bacchus, F. (2002). A knowledge-based approach to planning with incomplete information and sensing. In *Artificial Intelligence Planning Systems*.

[Petrick and Bacchus, 2004] Petrick, R. P. and Bacchus, F. (2004). Extending the knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of the 9th International Conference on the Principles of Knowledge Representation and Reasoning (KR)*.

[Pineau et al., 2003] Pineau, J., Gordon, G., and Thrun, S. (2003). Point-based value iteration: An anytime algorithm for pomdps. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*.

[Pistore et al., 2005] Pistore, M., Marconi, A., Bertoli, P., and Traverso., P. (2005). Automated composition of web services by planning at the knowledge level. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*.

[Poole and Zhang, 2003] Poole, D. and Zhang, N. L. (2003). Exploiting contextual independence in probabilistic inference. *Journal of Artificial Intelligence Research*, pages 263–313.

[Poole and Mackworth, 2010] Poole, D. L. and Mackworth, A. K. (2010). *Artificial Intelligence: foundations of computational agents*. Cambridge University Press.

[Portugal and Rocha, 2010] Portugal, D. and Rocha, R. (2010). Msp algorithm: multi-robot patrolling based on territory allocation using balanced graph partitioning. In *Proceedings of the ACM Symposium on Applied Computing*.

[Portugal and Rocha., 2011] Portugal, D. and Rocha., R. (2011). A survey on multi-robot patrolling algorithms. *Technological Innovation for Sustainability*, pages 139–146.

[Poupart, 2005] Poupart, P. (2005). *Exploiting structure to efficiently solve large scale partially observable Markov decision processes*. PhD thesis, University of Toronto.

[Poupart and Boutilier, 2003] Poupart, P. and Boutilier, C. (2003). Bounded finite state controllers. In *Advances in Neural Information Processing Systems (NIPS)*.

[Poupart et al., 2011] Poupart, P., Kim, K.-E., and Kim, D. (2011). Closing the gap: Improved bounds on optimal pomdpsolutions. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS)*.

[Puterman, 1994] Puterman, M. L. (1994). *Markov decision processes: discrete stochastic dynamic programming*. Wiley.

[Pynadath, 2002] Pynadath, D. V. (2002). The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, pages 389–423.

[Rathnasabapathy et al., 2006] Rathnasabapathy, B., Doshi, P., and Gmytrasiewicz., P. (2006). Exact solutions of interactive pomdps using behavioral equivalence. in. In *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems*.

[Ross and Chaib-Draa, 2007] Ross, S. and Chaib-Draa, B. (2007). Aems: An anytime online search algorithm for approximate policy refinement in large pomdps. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*.

[Roussel, 2010] Roussel, S. (2010). *Apports de la logique mathématique pour la modélisation de l'information échangée dans les systèmes multi-agents interactifs*. PhD thesis, Université de Toulouse.

[Roussel and Cholvy, 2009] Roussel, S. and Cholvy, L. (2009). Cooperative interpersonal communication and relevant information. In *Proceedings of the ESSLLI Workshop on Logical Methods for Social Concepts*.

[Ruan et al., 2005] Ruan, S., Meirina, C., Yu, F., Pattipati, K. R., and Popp, R. L. (2005). Patrolling in a stochastic environment. Technical report, Connecticut University.

[Russell and Norvig, 2009] Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach (Third Edition)*. Prentice-Hall.

[Sacerdoti, 1975] Sacerdoti, E. D. (1975). The nonlinear nature of plans. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence (IJCAI)*.

[Sak et al., 2008] Sak, T., Wainer, J., and Goldenstein, S. K. (2008). Probabilistic multiagent patrolling. In *Advances in Artificial Intelligence (SBIA)*.

[Saracevic, 1996] Saracevic, T. (1996). Relevance reconsidered. In *Proceedings of the 2nd Conference on Conceptions of Library and Information Science (CoLIS 2)*.

[Satsangi et al., 2014] Satsangi, Y., Whiteson, S., and Oliehoek, F. (2014). Exploiting submodular value functions for faster dynamic sensor selection: Extended version. Technical report, University of Amsterdam.

[Satsangi et al., 2015] Satsangi, Y., Whiteson, S., and Oliehoek, F. A. (2015). Exploiting submodular value functions for faster dynamic sensor selection. In *Proceedings of the 29th Conference on Artificial intelligence (AAAI)*.

[Savage, 1972] Savage, L. J. (1972). *The foundations of statistics*. Courier Dover Publications.

[Sayyareh, 2011] Sayyareh, A. (2011). A new upper bound for kullback-leibler divergence. *Applied Mathematical Sciences*, page 3303–3317.

[Seuken and Zilberstein, 2008] Seuken, S. and Zilberstein, S. (2008). Formal models and algorithms for decentralized decision making under uncertainty. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[Seymour and Peterson., 2009] Seymour, R. and Peterson., G. L. (2009). A trust-based multi-agent system. In *Proceedings of the International Conference on Computational Science and Engineering (CSE)*.

[Shachter and Peot., 1989] Shachter, R. D. and Peot., M. A. (1989). Simulation approaches to general probabilistic inference on belief networks. In *Proceedings of the 5th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

[Shafer, 1976] Shafer, G. (1976). *A mathematical theory of evidence*. Princeton university press.

[Shani et al., 2013] Shani, G., Pineau, J., and Kaplow, R. (2013). A survey of point-based pomdp solvers. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[Shenoy and Shafer, 1986] Shenoy, P. P. and Shafer, G. (1986). Propagating belief functions with local computations. *IEEE Expert*, pages 43–52.

[Sigaud and Buffet, 2010] Sigaud, O. and Buffet, O. (2010). *Markov decision processes in artificial intelligence*. Wiley.

[Silver and Veness, 2010] Silver, D. and Veness, J. (2010). Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems*.

[Sim and Roy, 2005] Sim, R. and Roy, N. (2005). Global a-optimal robot exploration in slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[Simmons et al., 2000] Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., and Younes., H. (2000). Coordination for multi-robot exploration and mapping. In *Proceedings of the 17th National Conference on Artificial intelligence (AAAI)*.

[Singh et al., 2009] Singh, A., andl Carlos Guestrin, A. K., and Kaiser, W. (2009). Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research*, pages 707–755.

[Smallwood and Sondik, 1973] Smallwood, R. D. and Sondik, E. J. (1973). The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, pages 1071–1088.

[Smith and Weld, 1998] Smith, D. E. and Weld, D. S. (1998). Conformant graphplan. In *Proceedings of the 15th National Conference on Artificial intelligence (AAAI)*.

[Smyth, 2014] Smyth, T. (2014). Rules for tower of hanoï. Accessed: 2014-10-15.

[Sondik, 1978] Sondik, E. J. (1978). The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations Research*, pages 282–304.

[Spaan, 2008] Spaan, M. T. (2008). Cooperative active perception using pomdps. In *Proceedings of the Workshop on advancements in POMDP solvers (Workshop of AAAI)*.

[Spaan et al., 2010] Spaan, M. T., Veiga, T. S., and Lima, P. U. (2010). Active cooperative perception in network robot systems using pomdps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[Spaan and Vlassis, 2005] Spaan, M. T. and Vlassis, N. A. (2005). Perseus: Randomized point-based value iteration for pomdps. *Journal of Artificial Intelligence Research*, pages 195–220.

[Spohn, 1988] Spohn, W. (1988). Ordinal conditional functions: A dynamic theory of epistemic states. In *Proceedings of the Irvine Conference on Probability and Causation*.

[Stachniss et al., 2005] Stachniss, C., Grisetti, G., and Burgard, W. (2005). Information gain-based exploration using rao-blackwellized particle filters. In *Robotics: Science and Systems*.

[Stefik, 1981] Stefik, M. (1981). Planning with constraints. *Artificial Intelligence*, pages 111–139.

[Sunderhauf et al., 2007] Sunderhauf, N., Lange, S., and Protzel, P. (2007). Using the unscented kalman filter in mono-slam with inverse depth parametrization for autonomous airship control. In *IEEE International Workshop of Safety, Security and Rescue Robotics*.

[Sussman, 1974] Sussman, G. J. (1974). The virtuous nature of bugs. *Readings in Planning*, pages 111–117.

[van Lambalgen, 1987] van Lambalgen, M. (1987). *Random Sequences.* PhD thesis, University of Amsterdam.

[van Lint and Wilson, 2001] van Lint, J. H. and Wilson, R. M. (2001). *A course in combinatorics.* Cambridge university press.

[van Rijsbergen, 1979] van Rijsbergen, C. K. (1979). *Information Retrieval (2nd Edition).* Butterworth-Heinemann Newton.

[Wang et al., 2007] Wang, Z., Huang, S., and Dissanayake, G. (2007). Multi-robot simultaneous localization and mapping using d-slam framework. In *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP)*.

[Weyns et al., 2004] Weyns, D., Steegmans, E., and Holvoet, T. (2004). Towards active perception in situated multi-agent systems. *Applied Artificial Intelligence*, pages 867–883.

[Wilson and Sperber, 2002] Wilson, D. and Sperber, D. (2002). *Handbook of pragmatics*, chapter Relevance theory, pages 606–632. Wiley.

[Wright, 1951] Wright, G. H. V. (1951). *An essay in modal logic.* North-Holland Pub Co.

[Wurm et al., 2008] Wurm, K. M., Stachniss, C., and Burgard, W. (2008). Coordinated multi-robot exploration using a segmentation of the environment. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[Yamauchi, 1997] Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*.

[Yamauchi, 1998] Yamauchi, B. (1998). Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents (AGENTS)*.

[Yedidia et al., 2005] Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, pages 1182–2312.

[Zadeh, 1988] Zadeh, L. A. (1988). Fuzzy logic. *Computer*, pages 83–93.

[Zadeh, 1999] Zadeh, L. A. (1999). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, pages 9–34.

[Zhou and Roumeliotis, 2006] Zhou, X. S. and Roumeliotis, S. I. (2006). Multi-robot slam with unknown initial correspondence: The robot rendez-vous case. In *Proceedings of the International Conference on Intelligent Robots and Systems*.

[Zlot et al., 2002] Zlot, R., Stentz, A., Dias, M. B., and Thayer, S. (2002). Multi-robot exploration controlled by a market economy. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.