HAL
open science

Contrôle du niveau de collaboration Humain-Robot sous un environnement complexe, via un modèle de l'autonomie ajustable

Nicolas Côté, Arnaud Canu, Maroua Bouzid, Abdel-Illah Mouaddib

To cite this version:

Nicolas Côté, Arnaud Canu, Maroua Bouzid, Abdel-Illah Mouaddib. Contrôle du niveau de collaboration Humain-Robot sous un environnement complexe, via un modèle de l'autonomie ajustable. Journées Francophones sur la planification, la décision et l'apprentissage pour le contrôle des systèmes - JFPDA 2012, May 2012, Villers-lès-Nancy, France. 12 p. hal-00736305

HAL Id: hal-00736305

https://inria.hal.science/hal-00736305

Submitted on 28 Sep 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Contrôle du niveau de collaboration Humain-Robot sous un environnement complexe, via un modèle de l'autonomie ajustable.

Nicolas Côté, Arnaud Canu, Maroua Bouzid, Abdel-Illah Mouaddib

GREYC - CNRS (UMR 6072), Université de Caen, Basse-Normandie, ENSICAEN
Boulevard du Maréchal Juin 14032 Caen CEDEX, France

Résumé : De nombreux travaux, issus de la communauté planification, permettent de rendre autonome un agent évoluant en environnement complexe. Il arrive cependant que des situations imprévues arrivent, qui rendent inutile le plan pré-calculé d'un agent. Dans cet article, nous étudions un problème dans lequel plusieurs agents sont en interaction avec plusieurs humains, au sein d'un environnement complexe. Nous présentons notre modèle intitulé HHP-MDP (Human Help Provider for Markovian Decision Processes, ou « processus de décision markovien assisté par des humains »), qui permet aux agents de demander de l'aide à des humains lorsqu'ils rencontrent des situations non prévues. Pour cela, nous expliquons comment les agents peuvent détecter les situations compliquées pour eux, et envoyer différents types de requêtes à l'ensemble des humains. HHP-MDP décrit comment un contrôleur peut gérer les différentes requêtes d'un agent et les assigner à un humain, tout en prenant en compte les capacités de l'humain à gérer ces requêtes. Les capacités d'un humain à gérer les requêtes seront apprises lors de la résolution du problème. Ce contrôleur est conçu pour réduire le coût de dérangement d'un humain. De plus, nous expliquons comment un humain peut prendre en compte rapidement l'état d'un agent, et ainsi limiter les inconsistances.

1 Introduction

L'usage de plus en plus fréquent de la robotique implique de parfaire l'autonomie et l'efficacité des agents. Malgré tout, calculer un comportement optimal pour un agent évoluant en environnement complexe reste une tâche difficile, notamment à cause de capacités de calcul limitées. Dans cet article, nous étudions le problème de la planification en environnement incertain et dynamique, afin notamment, de réduire la complexité du processus de décision et d'améliorer les performances des comportements résultants. Nous intégrons pour cela des humains dans le processus de décision des agents : le problème résultant, impliquant plusieurs agents et plusieurs humains, nécessite une collaboration Humain-Robot efficace.

Afin d'illustrer cette collaboration, nous étudions ici un problème de navigation au sol (sur aéroport) pour une flotte de véhicules aériens sans pilote (UAV). Le problème *UAV-an* implique un ensemble de drones autonomes (UAV) qui évoluent sur un ensemble de voies de circulation. Ces drones doivent remplir des objectifs simples comme atteindre un hangar, ou atteindre la piste de décollage. Cependant, les voies de circulation peuvent être obstruées par des facteurs environnementaux (par exemple de la pluie ou de la neige). Lorsqu'un agent rencontre un obstacle, il devient difficile pour lui d'agir comme il l'avait planifié. Ses capacités étant limitées, il ne peut pas détecter les différents problèmes l'empêchant d'atteindre son objectif, et ne peut donc pas recalculer son plan en prenant en compte les modifications de l'environnement. L'agent doit alors demander de l'aide, et un humain sera sélectionné afin de le guider.

Un humain peut aider un agent en le téléopérant, ou en lui donnant des recommandations (comme nous l'avons introduit précédemment dans Cote *et al.* (2011)). La téléopération est utilisée lorsque l'agent rencontre une situation critique (les actions sont limitées). Dans les autres cas (l'agent a des problèmes de planification, mais la situation n'est pas critique), l'humain utilise des recommandations et donne à l'agent des sous-buts à atteindre avant d'atteindre son but principal, de façon à éviter les difficultés. La téléopération a pour défaut de demander plus de temps à l'humain que les recommandations : le temps requis d'un humain pour aider un agent détermine la qualité de l'autonomie d'un agent, et nous pouvons déduire de cette autonomie le nombre d'agents qu'un humain peut gérer.

Dans cet article, nous décrivons nos trois contributions répondant à des problématiques importantes :

1. **Détecter les problèmes :** un agent doit détecter le fait qu'il se trouve dans une situation difficile, comprendre si cette situation est critique (à cause de ses capacités limitées) ou juste compliquée (problème de planification) et finalement envoyer une requête aux humains.
2. **Allouer les requêtes :** en premier, les agents envoient leurs requêtes à un contrôleur. Ce contrôleur analyse les requêtes et les capacités de chacun des humains disponibles pour confier l'aide d'un agent à un opérateur humain. L'objectif est de trouver l'allocation qui maximise les performances du système, et donc l'aide apportée aux agents.
3. **Maximiser la compréhension de la situation :** l'agent doit détecter et donner les informations qui seront importantes à l'humain afin qu'il puisse comprendre les difficultés de l'agent à remplir son objectif. Une bonne compréhension de la situation permet de réduire les inconstances avec le comportement de l'agent.

Nous introduisons notre modèle HHP-MDP qui permet de décrire comment un agent formalise ses requêtes et comment le contrôleur les gère, de façon à les assigner aux humains disponibles. Ces assignations sont effectuées en fonction des préférences et des capacités de l'humain. De plus, ce modèle est flexible et optimise le système global, qui fonctionne même lorsque le contrôleur reçoit plus de demandes que d'humains disponibles. Dans un premier temps, nous décrivons les travaux ayant proposé des solutions à certains des problèmes adressés par notre approche. Dans un second temps, nous développons les trois contributions qui composent notre modèle. Pour finir, nous présentons des résultats expérimentaux obtenus grâce à notre modèle HHP-MDP, sur le problème *UAV-an*.

2 Etat de l'art

Notre travail est basé sur un ensemble de résultats venant des communautés de l'interaction homme-robot (HRI) et des processus de décision markoviens (MDP). Dans cette section, nous présentons les travaux issus de ces deux communautés, relatifs à notre modèle.

2.1 Interactions homme-robot

Les interactions homme-robot ont été largement étudiées dans la littérature. Notre modèle est basé sur l'autonomie ajustable, telle que Bradshaw *et al.* (2005) l'ont décrite, qui nous permet de changer dynamiquement l'autonomie d'un agent. Dorais *et al.* (1999) ont démontré les avantages de l'autonomie ajustable comparée aux approches traditionnelles. Brooks (1986) a introduit plusieurs types d'interactions homme-robot, impliquant la mise en place d'une autonomie ajustable (dont quatre ont été étudiées plus en détails par Goodrich *et al.* (2001)). Dans cet article, nous décrivons un modèle dans lequel les agents évoluent parmi trois niveaux d'autonomie :

- agents autonomes,
- agents télé-opérés (tels que Goldberg *et al.* (2000) l'ont décrit),
- agents « Goal Biased » (que nous avons introduit dans Cote *et al.* (2011)).

Ce dernier niveau est un compromis entre la téléopération et l'autonomie complète, dans lequel les humains donnent des recommandations à l'agent. Un système impliquant des humains doit, de plus optimiser trois paramètres cruciaux :

1. **Le temps de négligence** d'un agent, (ou « neglect time », introduit par Crandall *et al.* (2005)). Ce paramètre représente le temps pendant lequel un agent peut agir de façon complètement autonome, avant que ses performances n'atteignent un seuil critique. D'après cette définition, il faut minimiser le délai durant lequel l'agent attend de l'aide, afin de ne pas dépasser son temps de négligence. Ce paramètre permet théoriquement de calculer le nombre d'humains nécessaires au bon fonctionnement.
2. **La compréhension de la situation** d'un agent par un humain (ou « situation awareness », tel qu'introduit par Endsley (1988)). Ce paramètre permet de décrire comment l'humain perçoit l'état dans lequel se situe l'agent, aussi bien dans le temps que dans l'espace, mais également le comportement de l'agent (son évolution récente et son objectif). Il est nécessaire de maximiser cette compréhension, comme l'a montré Yanco (2004). En effet, lorsque l'humain ne comprend pas la situation dans laquelle l'agent se trouve, l'efficacité du système est fortement réduite.

3. **Le coût de l'interruption** (ou « bother cost »). Ce paramètre décrit la fréquence d'interruption d'un humain, lorsqu'il effectue une tâche. Cohen *et al.* (2011) ont montré que de telles interruptions peuvent avoir des effets néfastes sur le système. Il est donc nécessaire de minimiser cet aspect.

La collaboration entre l'homme et le robot, telle qu'on l'adresse dans cet article, n'est qu'un seul des types possibles d'interaction. Certains travaux, impliquant la collaboration entre l'homme et le robot, posent par exemple l'hypothèse que le robot est un outil au service de l'humain. Ainsi, Bechar & Edan (2003) ont étudié le problème d'un opérateur humain travaillant avec plusieurs robots et les aidant lorsqu'il pense qu'ils sont en difficulté (en utilisant plusieurs niveaux d'autonomie). D'autre travaux se basent sur des dialogues entre l'homme et le robot. Fong *et al.* (2003, 1999) par exemple présentent un robot capable de poser des questions à un seul opérateur, lorsque son autonomie est limitée. Dans cet article, nous utilisons un **contrôleur de collaboration**, qui a pour objectif de compenser les limitations des agents.

Ishikawa & Suzuki (1997) ont étudié un problème dans lequel les robots exécutent leurs tâches dans un environnement complexe, pendant que les opérateurs les observent. Lorsqu'un robot rencontre une difficulté, l'humain le télé-opère. De la même façon, Lee *et al.* (2010); Wang *et al.* (2009) ont décrit un problème impliquant plusieurs agents et plusieurs humains, où les humains observent les agents et aident ceux qui sont dans des situations compliquées. Il est alors de la responsabilité des humains de détecter ces agents en difficultés. Ohba *et al.* (1999) ont montré que le délai durant lequel un agent attend un humain a un effet négatif sur le système multi-opérateurs/multi-agents. Dans notre approche, nous minimisons ce délai d'attente (afin de ne pas dépasser le temps de négligence) ainsi que le coût de l'interruption, et nous maximisons la compréhension de la situation. L'optimisation de ces trois paramètres nous permet d'exploiter au mieux l'aide que peuvent apporter les humains au groupe d'agents.

2.2 Processus de décision makoviens

Les processus de décision markoviens (ou MDP), introduit par Puterman (1994), forment un modèle élégant permettant de décrire les problèmes de décision d'un agent autonome évoluant en environnement complexe. Un MDP est un n-uplet $\langle S, A, T, R\rangle$ où :

- S est un ensemble fini d'états dans lesquels l'agent peut se trouver,
- A est un ensemble fini d'actions, applicables en tout état,
- $T : S \times A \times S \rightarrow [0; 1]$ est une fonction de transition, donnant la probabilité pour l'agent de passer d'un état $s \in S$ à un état $s' \in S$, lorsqu'il a appliqué l'action $a \in A$,
- $R : S \times A \rightarrow \mathbb{R}$ est une fonction de récompense, représentant l'intérêt immédiat qu'a l'agent à exécuter l'action $a \in A$ lorsqu'il se trouve dans l'état $s \in S$.

Résoudre un MDP signifie calculer une politique $\pi : S \rightarrow A$ qui associe, à tout état, l'action à exécuter. On peut calculer cette politique grâce à l'équation de Bellman qui donne, pour tout état s, sa valeur $V(s)$:

$$V(s) = \max_a R(s, a) + \gamma \sum_{s'} T(s, a, s')V(s')$$

Grâce à cette équation, on calcule la politique π^* qui maximise la valeur espérée en chaque état (c'est-à-dire l'espérance de gain à long terme). Dans cet article, nous utilisons les MDP orientés but, dans lesquels seul un état contient une récompense positive, tandis que les autres apportent une récompense nulle ou négative. Lorsque ce but est atteint, l'agent en reçoit un nouveau et doit recalculer sa politique.

Mouaddib *et al.* (2010) ont proposé une approche basée MDP pour détecter lorsque l'agent doit faire des demandes de téléopération. Le modèle décrit est, toutefois, assez limité (il ne permet, par exemple, que de gérer un robot par humain). L'approche OPOMDP de Armstrong-Crews & Veloso (2007) a été introduite de façon à aider les agents dont le modèle est un processus de décision markovien partiellement observable (POMDP). Dans cette approche, l'humain donne des indications à l'agent de façon à l'aider à déterminer l'état dans lequel il se trouve, réduisant ainsi l'aléa sur les connaissances de l'agent afin d'augmenter la qualité de son comportement. Ce modèle repose toutefois sur l'hypothèse qu'un humain est toujours disponible. Ainsi, le modèle OP-POMDP (introduit par Rosenthal & Veloso (2011)) est similaire aux OPOMDP, mais prend en compte une probabilité que l'humain ne soit pas disponible, (supprimant ainsi la contrainte de la disponibilité totale de l'humain mais augmentant la complexité du modèle).

Dans cet article, nous utilisons une approche basée MDP : ce modèle permet non-seulement aux agents d'évoluer de façon autonome dans leur environnement, mais dispose également de propriétés intéressantes permettant à l'agent de déterminer seul l'aide dont il a besoin, sans aucun dialogue avec l'humain. L'usage d'un MDP facilite également la compréhension de la situation par l'humain, lorsqu'il doit aider l'agent.

Pour finir, notre approche intègre les techniques du « Goal Biased », telles qu'elles ont été introduites dans Cote *et al.* (2011). L'humain peut donc donner des recommandations, sous la forme d'un ensemble de sous-buts que l'agent doit atteindre avant son objectif principal. Nous avons ainsi un niveau intermédiaire d'autonomie, situé entre la téléopération et l'autonomie complète qui permet de réduire le temps que l'humain doit consacrer à l'agent.

3 Détection des états critiques (respect du temps de négligence)

Il est crucial de détecter les états critiques de façon efficace : plus vite un agent peut détecter les états dans lesquels il doit demander de l'aide, moins il risque de dépasser son temps de négligence. Deux approches complémentaires sont décrites dans cette partie. Dans la première, nous expliquons comment détecter un ensemble d'états critiques, durant le calcul de la politique. Dans la seconde, nous montrons comment identifier des états critiques additionnels, durant l'exécution de cette politique.

3.1 États absorbants

Un ensemble d'états absorbants est un « piège » dont l'agent a une probabilité nulle d'échapper (c'est-à-dire d'atteindre un autre état, non contenu dans cet ensemble). Plus formellement, avec g l'objectif d'un agent, un état s est absorbant si et seulement si $P(g|s, \pi) = 0$ (la probabilité d'atteindre l'état but g depuis l'état s après une infinité de pas de temps est nulle, lorsque l'on applique la politique π dans chaque état rencontré). Cet ensemble peut être calculé grâce à une analyse du graphe de la politique (algorithme 1).

Algorithm 1: `absorbingStates()`.

Input: S, T (du MDP) et π (la politique)
Output: un ensemble d'états absorbants
$D \leftarrow$ nouveau dictionnaire;
foreach $s \in S$, $s \neq g$ **do**
  $D(s) \leftarrow \{s' \in S | T(s, \pi(s), s') > 0)$;
$L \leftarrow \{g\}$;
while *L n'est pas vide* **do**
  $s' \leftarrow L.pop()$;
  `// pour chaque clé s dans le dictionnaire:`
  foreach $s \in D$ **do**
    `// si s' est associé à une clé s:`
    if $s' \in D(s)$ **then**
      retirer s de D;
      ajouter s in L;
`// les clés restantes sont absorbantes`
return $\{s | D(s) \neq \emptyset\}$;

Si l'agent se trouve dans un état absorbant, alors ses capacités ne seront pas suffisantes pour atteindre l'objectif, et il devra envoyer une requête à un humain. Dans le problème *UAV-an*, un agent peut être dans un état absorbant si il se trouve dans une voie de circulation dans laquelle il n'a pas assez d'espace pour manœuvrer par exemple. Il demandera alors l'aide d'un humain tant qu'il ne sera pas arrivé dans des voies de circulation plus grandes. Cette approche ne permet de détecter que les états critiques qui sont prévisibles : un agent peut atteindre un état critique qui n'a pas été prévu à cause d'un nouvel obstacle par exemple. Comment, dans ce cas, l'agent peut-il détecter cet état critique ?

3.2 Détection en temps réel

On calcule, grâce à l'équation de Bellman, la politique optimale qui maximise la valeur espérée en chaque état. Ainsi, lors de l'exécution d'une politique optimale π^*, l'agent se déplace vers les états associés aux plus hautes valeurs espérées (car il se déplace vers le but). Cependant, les agents évoluent dans un environnement stochastique, donc leurs actions peuvent parfois échouer et amener l'agent vers des états indésirables. Nous posons l'hypothèse que les valeurs espérées rencontrées lors de l'exécution vont, en moyenne, augmenter.

La seule exception est lorsqu'un but est atteint : un nouveau but est alors assigné à l'agent, et la valeur espérée de l'état où il se trouve va temporairement diminuer (ce nouveau but étant généralement loin du but précédent). La figure 1 présente un exemple d'un historique de valeurs espérées lors d'une exécution.

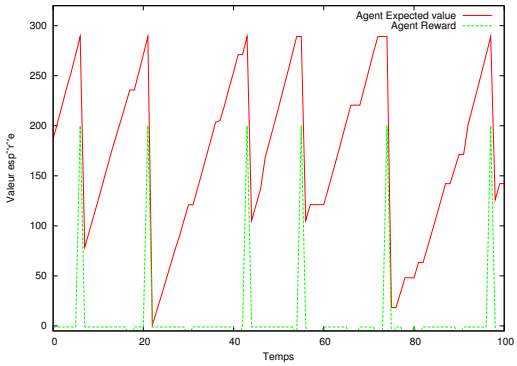

FIGURE 1 – L'historique des valeurs espérées et des récompenses

Nous pouvons donc déduire que, si la valeur espérée n'augmente pas aussi vite que nous l'avons prévu, alors l'agent rencontre probablement une difficulté. L'augmentation peut être trop lente (dans le problème *UAV-an*, l'agent peut détecter de l'eau sur une voie de circulation par exemple, qui le forcera à ralentir), ou critique (de la neige sur une voie de circulation peut empêcher complètement l'agent de bouger). Afin de détecter ces problèmes (augmentation lente ou critique), l'agent garde un historique des valeurs espérées récentes, et compare leur augmentation avec l'augmentation moyenne sur l'ensemble de l'historique. Lorsque la valeur espérée diminue ou augmente trop lentement (pour une autre raison que l'accomplissement d'un objectif), alors l'agent demande de l'aide.

Formellement, avec s_i un état dans lequel l'agent était à l'instant i, t l'instant actuel et w la taille de la fenêtre d'observation, nous pouvons calculer la moyenne *Avg* et la moyenne de la fenêtre *WAvg* :

$$Avg = \frac{1}{t}\sum_{i=0}^{t}(V(s_i) - V(s_{i-1})) \cdot \mathbb{1}[R(s_{i-1}) \leq 0]$$

$$WAvg = \frac{1}{w}\sum_{i=(t-w)}^{t}(V(s_i) - V(s_{i-1})) \cdot \mathbb{1}[R(s_{i-1}) \leq 0]$$

Ici, $\mathbb{1}[R(s_{i-1}) \leq 0]$ est égal à 0 if $R(s_{i-1})$ est positif, 1 sinon. Ce facteur est utilisé pour enlever de l'équation les pas de temps suivant l'accomplissement d'un objectif (car la valeur espérée diminue fortement à chaque fois qu'un objectif est atteint). Finalement, nous calculons le niveau critique C d'un agent avec $C = WAvg/Avg$. Lorsque C est assez bas (par exemple, $C < 0.5$), alors l'agent demande au contrôleur de l'aide en lui fournissant la valeur de C dans la requête. Si $C \leq 0$, l'agent demande une aide critique.

4 Le contrôleur (minimisation du coût de l'interruption)

Nous ajoutons au problème un contrôleur centralisé, qui reçoit les requêtes envoyées par les agents et qui les assigne aux opérateurs humains disponibles. La figure 2 décrit l'architecture associée à ce contrôleur.

Dans le cas simple d'un problème impliquant un humain et un agent, un tel contrôleur ne serait pas nécessaire. Cependant, la plupart du temps, plusieurs agents sont impliqués et le contrôleur doit répartir d'avantage de requêtes que d'opérateurs disponibles (parfois, à l'inverse, on disposera de plus d'humains que de requêtes en attente). Un tel contrôleur doit alors optimiser l'efficacité de la collaboration pour le système global : nous voulons non-seulement assigner en priorité les requêtes les plus urgentes, mais également prendre en compte les capacités des humains lors de la distributions de ces requêtes (de façon à ce qu'ils les traitent le plus rapidement possible).

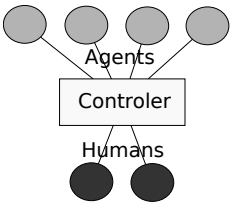

FIGURE 2 – HHP-MDP architecture

4.1 Types de requêtes

Il existe trois types de requêtes différentes qu'un agent peut envoyer, chacune associée à une priorité :

1. **État absorbant :** cette requête est la plus critique. L'agent sait qu'il ne peut pas atteindre son but avec ses propres capacités. En général, une telle requête est envoyée lorsque l'agent tombe en panne et requièrt une intervention de l'humain (le plus souvent une téléopération, voire une intervention physique sur le robot).
2. **Évolution des valeurs critiques :** ces requêtes sont également critiques ($C \leq 0$). L'agent n'évolue plus vers son but. Le plus souvent, l'agent est dans une situation difficile pour laquelle il ne peut pas calculer une solution lui permettant d'atteindre son objectif (au vu de ses capteurs et effecteurs limités). L'intervention de l'humain sera une téléopération, jusqu'à ce que l'agent redevienne autonome.
3. **Évolution des valeurs faibles** : cette requête n'est pas critique. Malgré ses difficultés, l'agent évolue vers le but, mais son évolution est plus lente que la normale ($0 < C \leq 0.5$). En général, l'humain donnera une (ou des) recommandation à l'agent afin de l'aider à trouver une autre solution pour atteindre son but.

4.2 Allocation des requêtes

Lorsque de nombreux agents sont impliqués, le contrôleur reçoit beaucoup de requêtes au même instant. Le plus souvent, il en reçoit plus que le nombre d'opérateurs disponibles. Afin de minimiser le coût de l'interruption, nous avons simplement choisi de ne pas interrompre un humain qui est entrain d'aider un agent. Le contrôleur doit donc choisir la meilleure allocation possible des requêtes aux opérateurs, afin de maximiser l'impact de la collaboration. Dans l'objectif de calculer la meilleure allocation, le contrôleur maintient un modèle des capacités de chaque opérateur.

4.2.1 Modèle d'un opérateur

Le contrôleur maintient un modèle de chaque opérateur. Il reçoit également un ensemble de requêtes. Pour un opérateur donné, le contrôleur choisit la meilleure requête en prenant en compte :
– le temps requis pour gérer une requête,
– la valeur espérée que l'agent aura gagnée.

Le contrôleur calcule donc deux fonctions. La première, $time(r, o)$ donne le temps moyen nécessaire pour qu'un opérateur o gère une requête r (ce temps dépendant du type de requête : état absorbant, évolution des valeurs critique, ou évolution faible). La seconde fonction, $gain(r, o)$ donne la valeur espérée gagnée lorsqu'un opérateur o gère une requête r.

Ces deux fonctions ont des valeurs initiales fixées en fonction de données statistiques. Lors de l'exécution du problème, ces valeurs seront mises à jour. Par exemple, on peut utiliser une approche similaire à l'apprentissage par renforcement : à chaque pas de temps, un opérateur o gère une requête r. On calcule le temps passé à traiter cette requête et la valeur espérée que l'agent a gagné afin de renforcer les fonctions $time(r, o)$ et $gain(r, o)$. Cette partie n'étant pas dans les contributions principales de cet article, nous n'en parlerons pas d'avantage.

4.2.2 Choisir la bonne requête

À chaque fois qu'un opérateur devient disponible, le contrôleur lui envoie la requête qu'il doit gérer et, donc, l'agent qu'il doit aider. On connait, pour chaque requête, le temps nécessaire par un opérateur pour la

traiter et le gain qu'il gagnera. Nous pouvons donc calculer le gain pondéré :

$$G(r, o) = \frac{gain(r, o)}{time(r, o)}$$

De plus, afin de choisir la meilleure requête r^* associée à un opérateur o, nous devons maximiser le gain pondéré pour chaque requête possible.

$$r^*(o) = \underset{r}{\operatorname{argmax}}\, G(r, o)$$

À chaque pas de temps, le contrôleur vérifie si il a reçu de nouvelles requêtes, qui correspondent mieux à un opérateur disponible. De plus si plusieurs opérateurs sont disponibles, le contrôleur affecte les meilleures allocations $r * (o)$ en premier, puis il recalcule les autres affectations jusqu'à ce que les opérateurs ne soit plus disponibles.

5 Informations envoyées à l'humain (maximiser la compréhension)

Nous avons montré, dans la section précédente, la façon dont le contrôleur peut choisir la requête qu'il va assigner à un humain. Toutefois, si cet humain n'a pas assez d'informations sur le problème, son aide ne sera pas optimale. L'humain doit donc comprendre au maximum dans quelle situation se trouve agent. Comme nous l'avons expliqué dans la partie état de l'art, la *compréhension de la situation* est un paramètre crucial pour les performances du système. Une bonne compréhension va permettre à l'humain de prendre des décisions rapidement, et d'éviter les inconsistances avec la politique de l'agent. HHP-MDP prend en compte ce facteur en utilisant des propriétés du modèle MDP.

Lorsqu'un humain aide un agent, les informations inutiles compliquent la compréhension de la situation de l'agent. Nous avons donc essayé de minimiser le nombre d'informations provenant de l'agent, de façon à ne donner que celles qui sont pertinentes à l'humain. Une information pertinente est une information qui permet à l'humain d'identifier le problème de l'agent afin de mieux l'aider. Nous décomposons pour cela les informations envoyées par l'agent en deux parties : les informations globales (concernant la stratégie globale de l'agent) et les informations locales (concernant le voisinage des agents).

5.1 Informations locales

Lorsqu'un humain démarre une téléopération, il a besoin d'informations pour choisir une action qui sera cohérente avec le comportement de l'agent. De plus, l'agent ayant souvent besoin d'une intervention rapide de l'humain, il faut que les informations fournies soient simples à comprendre. Il y a deux types d'informations dont un humain peut avoir besoin :

- **L'historique de l'agent :** l'humain doit comprendre le problème de l'agent. Pour cela, nous présentons les états atteints par cet agent lors des précédents pas de temps et les actions associées.
- **La politique de l'agent :** l'humain doit choisir des actions cohérentes avec la politique de l'agent. Afin de choisir une telle action, l'humain a besoin d'informations sur les états atteignables (tous les états qui ont une probabilité non nulle d'être atteinte en n pas de temps), et sur la politique associée à ces états.

La figure 3 présente un exemple d'informations locales, où la politique de l'agent est de descendre. Il vient de la gauche, et a essayé quatre fois de descendre sans parvenir à bouger.

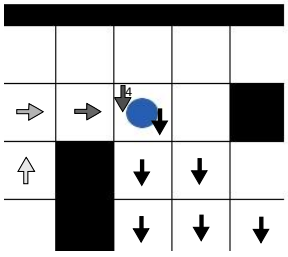

FIGURE 3 – La stratégie d'un agent à court terme

5.2 Informations globales

Les informations concernant l'objectif à long terme d'un agent sont importantes pour identifier ses problèmes de planification. Il existe deux types de données concernant l'objectif à long terme d'un agent, qu'il doit partager avec l'humain : la stratégie globale de l'agent utilisée pour atteindre son objectif, et les informations concernant l'objectif en lui-même. Ces informations peuvent être extraites d'un MDP :

- **Informations concernant l'objectif :** dans cet article, nous avons posé l'hypothèse que seul le but possède une récompense positive, donc on met en avant tous les états s tels que $\exists a \in A, R(s,a) > 0$.
- **Informations concernant la stratégie de l'agent :** nous utilisons le graphe de la politique afin d'extraire la stratégie globale d'un agent. L'objectif est de construire des clusters d'états adjacents, tels que tous les états au sein d'un même cluster auront la même politique (la même action sera effectuée). Cette décomposition en clusters simplifie la compréhension de la stratégie de l'agent, car elle permet de réduire le nombre d'informations à l'écran. Nous pouvons calculer de tels clusters en utilisant une approche gloutonne (algorithme 2).

Algorithm 2: `clusters()`.

Input: S, T (du MDP) et π (la politique)
Output: C l'ensemble de clusters
$L \leftarrow \{s \in S\}$;
$C \leftarrow \emptyset$;
while *L n'est pas vide* **do**
  $s \leftarrow L.pop()$;
  $c \leftarrow \{s\}$;
  $C \leftarrow C \cup \{c\}$;
  $toDo \leftarrow \{s\}$;
  while *$toDo$ n'est pas vide* **do**
    $s \leftarrow toDo.pop()$;
    foreach $s' \in L$ **do**
      if $\pi(s) = \pi(s')$ *et* $[T(s,\pi(s),s') > 0$ *or* $T(s',\pi(s'),s) > 0]$ **then**
        $c \leftarrow c \cup \{s'\}$;
        $toDo \leftarrow toDo \cup \{s'\}$;
        $L \leftarrow L \backslash \{s'\}$;
return C;

La figure 4 décrit un exemple d'informations globales présentées à l'humain.

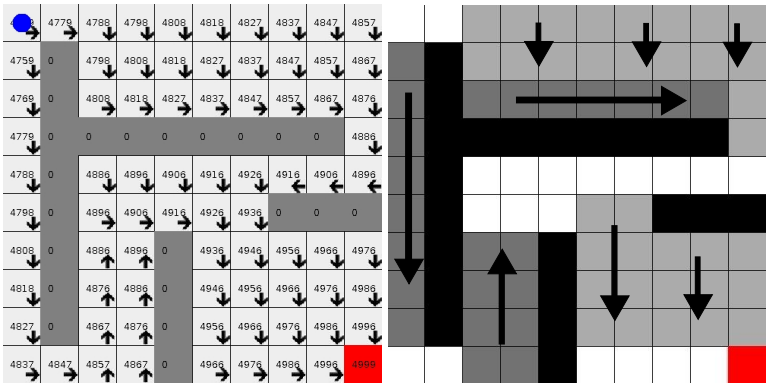

FIGURE 4 – La stratégie d'un agent à long terme

Grâce à cette approche, la vue de l'humain (dans la partie droite de la figure) est beaucoup plus simple à comprendre que la politique complète (dans la partie gauche de la figure). De plus, les clusters trop petits (de taille inférieure à 5% de l'espace d'états) sont ignorés, afin d'alléger l'affichage. Cette représentation permet d'appréhender rapidement la stratégie globale de l'agent.

En utilisant ces deux vues (la vue locale et la vue globale), nous maximisons la compréhension de la situation pour l'humain. Nous évitons également de donner des informations qui seraient inutiles, et nous gardons l'humain concentré sur le problème à résoudre. Le tableau ci-dessous présente les utilisations de notre modèle en fonction des requêtes reçues.

Type de requête	Interfaces	Autonomie
État absorbant	court terme	téléopération
Évolution critique	les deux	les deux
Évolution lente	long terme	recommandations

6 Expérimentations

Nous décrivons, dans cette section, un ensemble de résultats expérimentaux obtenus via notre modèle sur le problème *UAV-an* introduit précédemment.

6.1 Paramètres expérimentaux

Nous avons modélisé l'aéroport de Montpellier (figure 5), afin de réaliser nos expérimentations. Cet aéroport comporte deux voies de circulation principales, connectées à plusieurs voies de circulation proches des hangars. De nouveaux UAV sortent aléatoirement des hangars et doivent atteindre les pistes de décollage. La plupart du temps, les agents sont capables d'atteindre leurs objectifs en utilisant une approche complètement autonome. Cependant, il est possible que les conditions météorologiques (pluie ou neige) perturbent la circulation des UAV sur les voies de circulation. Des opérateurs sont disponibles pour aider les UAV étant dans des situations difficiles. Lorsqu'un agent atteint sa piste de décollage, il reçoit une récompense de 200.

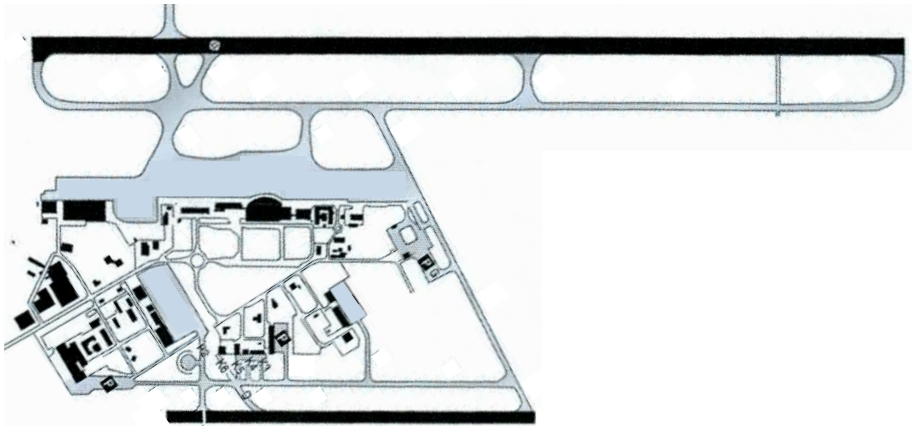

FIGURE 5 – L'aéroport de Montpellier

L'initialisation du modèle des humains est faite via des données précédemment collectées. On utilise les valeurs suivantes, pour les fonctions $time$ et $gain$:

- évolution lente : le temps moyen pour traiter la requête est de 20 pas de temps, le gain associé pour l'agent étant en moyenne de 5,
- évolution critique : le temps moyen est de 15, le gain associé est de 8,
- état absorbant : le temps moyen est de 10, le gain associé est de 12.

Nous avons donc construit un benchmark basé sur cet aéroport. Nous permettons à l'agent d'avancer tout droit, à gauche ou à droite. Il peut également rester où il se trouve. Dans une situation normale, l'agent a une probabilité de 0,1 que son action échoue (et donc de rester sur place). Lorsque l'agent rencontre une flaque de pluie, il a une probabilité de rester sur place de 0,3 et une probabilité de dévier à gauche ou à droite de 0,1 chacune. Lorsque l'agent évolue sur de la neige, il a une probabilité de dévier à gauche ou droite de 0,4 chacune. La neige ou la pluie tombent aléatoirement durant l'exécution du problème : on remarque, sur la figure 6, des flaques d'eau (tâches bleues) présentes à cause de la pluie. Deux UAV (le rond cyan et

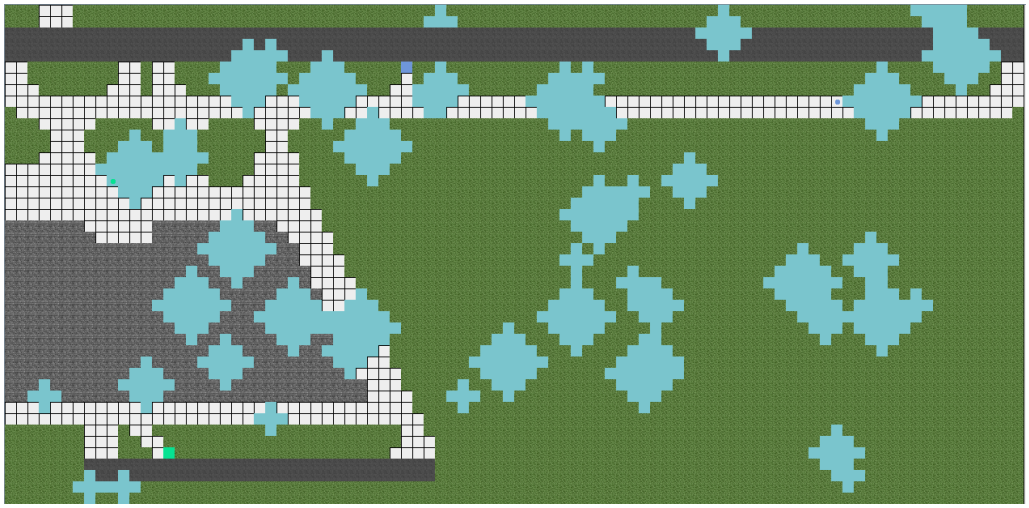

FIGURE 6 – Notre benchmark

bleu) sont en train d'évoluer sur les voies de circulation en direction de leurs buts (carré cyan et carré bleu). L'agent couleur cyan se retrouve dans la neige. Il doit donc demander de l'aide à l'humain.

6.2 Premier test : détection d'un état d'évolution lente ou critique

Nous avons exécuté un ensemble d'expérimentations, avec chacune 3000 pas de temps. Nous avons utilisé une fenêtre d'observation (afin de détecter si les valeurs espérées évoluent trop lentement) de 4 pas de temps. Dans notre première expérience (figure 6), nous n'avons utilisé qu'un seul agent et un seul opérateur. À chaque pas de temps, les agents évoluent sur l'aéroport (soumis aux contraintes météorologiques aléatoires) et vérifient s'ils doivent ou non envoyer des requêtes au contrôleur pour demander de l'aide. On observe que ces requêtes sont distribuées de la façon suivante :

- 11,5% pour des requêtes d'évolution lente.
- 88,5% d'évolution critique.

Cette distribution dépend de la limite à laquelle une évolution est considérée comme critique (dans notre cas, une requête est critique si l'évolution est au moins 50% moins rapide que l'évolution moyenne). D'après nos premiers tests, nous confirmons la capacité des agents à détecter les situations imprévisibles, et demander de l'aide dans ces cas.

6.3 Second test : choisir la bonne requête

Lorsque plus d'un agent est impliqué, plusieurs requêtes peuvent être envoyées en même temps. Le contrôleur doit alors être capable de choisir la plus importante, et de la transférer à un opérateur. La figure 7 présente une situation dans laquelle quatre agents évoluent sur les voies de circulation. Dans ce cas précis, l'agent 4 évolue normalement, tandis que les agents 1, 2 et 3 se déplacent dans la neige. Les agents 1 et 2 viennent d'arriver à des endroits où il y a de la neige. L'agent 3 quant à lui est coincé à cet endroit depuis 3 pas de temps. Le contrôleur reçoit alors trois requêtes : une critique de l'agent 3, et deux non critiques de l'agent 1 et 2. Comme prévu, le contrôleur affecte l'agent 3 à l'opérateur disponible, et ignore ainsi provisoirement les requêtes des agents 1 et 2.

Par la suite, si les agents 1 et 2 se retrouvent coincés eux-aussi dans la neige, ils enverront au contrôleur une requête critique. Celle-ci viendra alors remplacer les requêtes non-critiques envoyées préalablement.

6.4 Troisième test : passage à l'échelle

Nous avons mené d'autres expériences, impliquant de 1 à 15 agents en même temps, sans que le nombre d'opérateurs ne change. Durant ces expériences, nous avons calculé le pourcentage de requêtes critiques ignorées par le contrôleur comme décrit dans la figure 8. Avec moins de 10 agents, un opérateur est capable

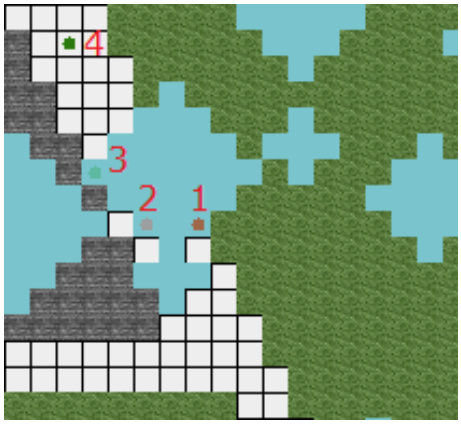

FIGURE 7 – Plusieurs requêtes

de gérer plus de 70% des requêtes critiques. En revanche, avec plus de 10 agents, les performances du système diminuent fortement (la moitié des requêtes sont ignorées avec 15 agents).

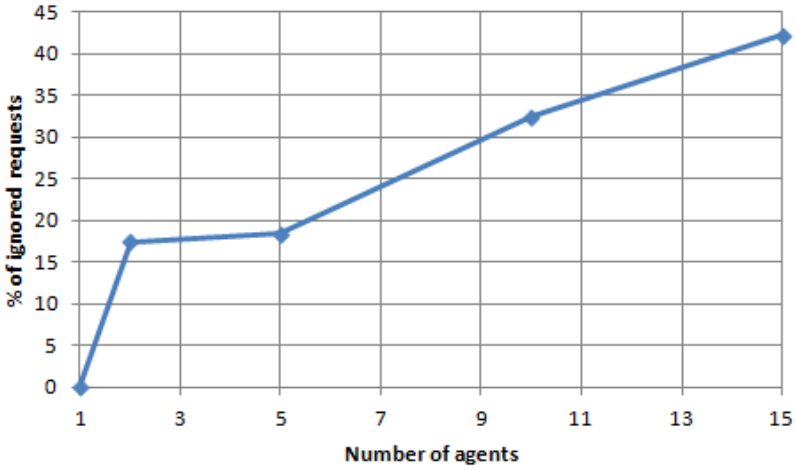

FIGURE 8 – Pourcentage de requête critique ignorés

Afin de garder un système stable, nous pouvons ajouter k opérateurs et diviser par k le nombre de requêtes ignorées. La complexité d'assigner des requêtes reste basse (puisqu'elle est linéaire en nombre d'opérateurs), permettant ainsi le passage à l'échelle avec de nombreux agents.

7 Conclusion

Nous avons décrit, dans cet article, une nouvelle approche permettant de gérer des problèmes de décision dans lesquelles des changements imprévisibles modifient l'environnement des agents. Nous avons notamment proposé une approche collaborative multi-agent/multi-humain, dans laquelle les agents peuvent demander de l'aide aux opérateurs. Nous avons ainsi décrit trois contributions : comment l'agent peut détecter une situation imprévisible et demander de l'aide, comment choisir l'opérateur qui sera le plus apte à aider cet agent et quelle description du problème donner à l'opérateur. Nous avons finalement validé notre approche sur un benchmark représentant le problème *UAV-an* : nous pouvons ainsi passer à l'échelle, avec plusieurs agents qui sélectionneront toujours les requêtes les plus importantes. On dispose donc, grâce au modèle HHP-MDP, d'une méthode efficace permettant d'optimiser l'aide que peuvent apporter les opérateurs humains à un groupe d'agents.

Références

ARMSTRONG-CREWS N. & VELOSO M. (2007). Oracular partially observable markov decision processes : A very special case. *Internationnal Conference on Robotics and Automation*, p. 2477–2482.

BECHAR A. & EDAN Y. (2003). Human-robot collaboration for improved target recognition of agricultural robots. *Industrial Robot : An International Journal*, **30**(5), 432–436.

BRADSHAW J., JUNG H., KULKARNI S., JOHNSON M., FELTOVICH P., ALLEN J., BUNCH L., CHAMBERS N., GALESCU L., JEFFERS R. *et al.* (2005). Kaa : Policy-based explorations of a richer model for adjustable autonomy. *AAMAS*, p. 214–221.

BROOKS R. (1986). *A robust layered control system for a mobile robot*, volume 2. IEEE Journal of Robotics and Automation.

COHEN R., JUNG H., FLEMING M. & CHENG M. (2011). A user modeling approach for reasoning about interaction sensitive to bother and its application to hospital decision scenarios. *User Modeling and User-Adapted Interaction*, p. 1–44.

COTE N., BOUZID M. & MOUADDIB A.-I. (2011). Integrating the human recommendations in the decision process of autonomous agents : A goal biased markov decision process. *Fall Symposium AAAI*.

CRANDALL J., GOODRICH M., OLSEN JR D. & NIELSEN C. (2005). Validating human-robot interaction schemes in multitasking environments. *Systems, Man and Cybernetics, Part A : Systems and Humans, IEEE Transactions on*, **35**(4), 438–449.

DORAIS G., BONASSO R., KORTENKAMP D., PELL B. & SCHRECKENGHOST D. (1999). Adjustable autonomy for human-centered autonomous systems. *Working notes of the Sixteenth International Joint Conference on Artificial Intelligence Workshop on Adjustable Autonomy Systems*, p. 16–35.

ENDSLEY M. (1988). Design and evaluation for situation awareness enhancement. **32**(2), 97–101.

FONG T., THORPE C. & BAUR C. (1999). Collaborative control : A robot-centric model for vehicle teleoperation.

FONG T., THORPE C. & BAUR C. (2003). Robot, asker of questions. *Robotics and Autonomous systems*, **42**(3-4), 235–243.

GOLDBERG K., CHEN B., SOLOMON R., BUI S., FARZIN B., HEITLER J., POON D. & SMITH G. (2000). Collaborative teleoperation via the internet. **2**, 2019–2024.

GOODRICH M., OLSEN D., CRANDALL J. & PALMER T. (2001). Experiments in adjustable autonomy. p. 1624–1629.

ISHIKAWA N. & SUZUKI K. (1997). Development of a human and robot collaborative system for inspecting patrol of nuclear power plants. p. 118–123.

LEE P., WANG H., CHIEN S., LEWIS M., SCERRI P., VELAGAPUDI P., SYCARA K. & KANE B. (2010). Teams for teams performance in multi-human/multi-robot teams. **54**(4), 438–442.

MOUADDIB A.-I., ZILBERSTEIN S., BEYNIER A. & JEANPIERRE L. (2010). A decision-theoretic approach to cooperative control and adjustable autonomy. *European Conference on Artificial Intelligence*, p. 971–972.

OHBA K., KAWABATA S., CHONG N., KOMORIYA K., MATSUMARU T., MATSUHIRA N., TAKASE K. & TANIE K. (1999). Remote collaboration through time delay in multiple teleoperation. **3**, 1866–1871.

PUTERMAN M. (1994). Markov decision processes : Discrete stochastic dynamic programming.

ROSENTHAL S. & VELOSO M. (2011). Modeling humans as observation providers using pomdps. *In Proceedings of International Symposium on Robot-Human Communication*.

WANG H., LEWIS M., VELAGAPUDI P., SCERRI P. & SYCARA K. (2009). How search and its subtasks scale in n robots. p. 141–148.

YANCO H. A. (2004). where am i ? acquiring situation awareness using a remote robot platform. *In IEEE Conference on Systems, Man and Cybernetics*, p. 2835–2840.