Actes des 14es Journées Francophones Planification, Décision et Apprentissage

Emmanuel Rachelson, Caroline P C Chanel

AfIA

Association française
pour l'Intelligence Artificielle

JFPDA

Journées Francophones Planification, Décision et Apprentissage

PFIA 2019

Table des matières

Éditorial

Les Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes (JFPDA) ont pour but de rassembler la communauté de chercheurs francophones travaillant sur les problèmes d'intelligence artificielle, d'apprentissage par renforcement, de programmation dynamique et plus généralement dans les domaines liés à la prise de décision séquentielle sous incertitude et à la planification. Les travaux présentés traitent aussi bien d'aspects purement théoriques que de l'application de ces méthodes à la conduite de systèmes virtuels (jeux, simulateurs) et réels (robots, drones). Ces journées sont aussi l'occasion de présenter des travaux en cours de la part de doctorants, postdoctorants et chercheurs confirmés dans un cadre laissant une large place à la discussion constructive et bienveillante.

Les articles présentés ici correspondent à des versions de soumission. Certains de ces articles ont par la suite été publiés dans d'autres conférence en version finale.

Emmanuel Rachelson et Caroline Chanel

Comité de programme

Présidents

— Emmanuel Rachelson (ISAE-SUPAERO)
— Caroline Chanel (ISAE-SUPAERO)

Membres

— Olivier Buffet (LORIA)
— Nicolas Drougard (ISAE-SUPAERO)
— Alain Dutech (LORIA)
— Humbert Fiorino (Université de Grenoble - LIG)
— Raphaël Fonteneau (Université de Liège)
— Frédérick Garcia (INRA)
— Matthieu Geist (Google AI)
— Malik Ghallab (LAAS-CNRS)
— Alain Haït (ISAE-SUPAERO)
— Guillaume Infantes (Jolibrain)
— Félix Ingrand (LAAS-CNRS)
— Erwan Lecarpentier (ISAE-SUPAERO)
— Charles Lesire (ONERA)
— Frédéric Maris (Université Toulouse 3 - IRIT)
— Laetitia Matignon (LIRIS CNRS)
— Alexandre Niveau (Université de Caen - GREYC)
— Damien Pellier (Université de Grenoble - LIG)
— Nicolas Perrin (CNRS - ISIR)
— Mathieu Picard (IRT Saint Exupéry - Airbus DS)
— Cédric Pralet (ONERA)
— Philippe Preux (Université de Lille - INRIA)
— Serge Rainjonneau (IRT Saint Exupéry - Thalès Alenia Space)
— Régis Sabbadin (INRA)
— Nicolas Schneider (Airbus CRT)
— Olivier Sigaud (Université Paris Sorbonne - ISIR)
— Jonathan Sprauel (Thalès Alenia Space)
— Florent Teichteil (Airbus CRT)
— Vincent Thomas (LORIA)
— Paul Weng (UM-SJTU Joint Institute)
— Bruno Zanuttini (Université de Caen - GREYC)

Étude de la motivation intrinsèque en apprentissage par renforcement

A. Aubret[1] L. Matignon[1] S. Hassas[1]

[1] Univ Lyon, Université Lyon 1, CNRS, LIRIS, F-69622, Villeurbanne, France

arthur.aubret@liris.cnrs.fr

Résumé

Malgré les nombreux travaux existants en apprentissage par renforcement (AR) et les récents succès obtenus notamment en le combinant avec l'apprentissage profond, l'AR fait encore aujourd'hui face à de nombreux défis. Certains d'entre eux, comme la problématique de l'abstraction temporelle des actions ou la difficulté de concevoir une fonction de récompense sans connaissances expertes, peuvent être adressées par l'utilisation de récompenses intrinsèques. Dans cet article, nous proposons une étude du rôle de la motivation intrinsèque en AR et de ses différents usages, en détaillant les intérêts et les limites des approches existantes. Notre analyse suggère que la notion d'information mutuelle est centrale à la plupart des travaux utilisant la motivation intrinsèque en AR. Celle-ci, combinée aux algorithmes d'AR profond, permet d'apprendre des comportements plus complexes et plus généralisables que ce que permet l'AR traditionnel.

Mots Clef

Apprentissage par renforcement, motivation intrinsèque, curiosité, acquisition de connaissances, *empowerment*, *options*, génération d'objectifs, méta-récompense.

Abstract

Despite many existing works in reinforcement learning (RL) and the recent successes obtained by combining it with deep learning, RL is facing many challenges. Some of them, like the ability to abstract the action or the difficulty to conceive a reward function without expert knowledge, can be addressed by the use of intrinsic motivation. In this article, we provide a survey on the role of intrinsic motivation in RL and its different usages by detailing interests and limits of existing approaches. Our analysis suggests that mutual information is central to most of the work using intrinsic motivation in RL. The combination of deep RL and intrinsic motivation enables to learn more complicated and more generalisable behaviours than what enables standard RL.

Keywords

Reinforcement learning, intrinsic motivation, curiosity, knowledge acquisition, empowerment, options, generation of objectives, meta-reward.

1 Introduction

En apprentissage par renforcement (AR), un agent apprend par essais-erreurs à maximiser l'espérance des récompenses reçues suite aux actions effectuées dans son environnement [Sutton and Barto, 1998].

Traditionnellement, pour apprendre une tâche, un agent maximise une récompense définie selon la tâche à accomplir : cela peut être un score lorsque l'agent doit apprendre à gagner à un jeu ou une fonction de distance lorsque l'agent apprend à atteindre un objectif. Nous parlons alors de récompense extrinsèque (ou *feedback*) car la fonction de récompense est fournie de manière experte spécifiquement pour la tâche. Avec une récompense extrinsèque, plusieurs résultats spectaculaires ont été obtenus sur les jeux Atari [Bellemare *et al.*, 2015] avec le Deep Q-network (DQN) [Mnih *et al.*, 2015] ou sur le jeu du go avec AlphaGo Zero [Silver *et al.*, 2017]. Cependant, ces approches se révèlent le plus souvent infructueuses lorsque les récompenses sont trop éparses dans l'environnement, et l'agent est alors incapable d'apprendre le comportement recherché pour la tâche [François-Lavet *et al.*, 2018]. D'autre part, les comportements appris par l'agent sont difficilement réutilisables, aussi bien au sein d'une même tâche que pour plusieurs tâches [François-Lavet *et al.*, 2018] : il est difficile pour un agent de généraliser ses compétences de manière à prendre des décisions abstraites dans l'environnement. Par exemple, une décision abstraite (ou de haut-niveau) pourrait être *aller jusqu'à la porte* en utilisant des actions primitives (ou de bas-niveau) consistant à se déplacer dans les quatre directions cardinales ; ou encore de *se déplacer en avant* en contrôlant les différentes articulations d'un robot humanoïde comme dans le simulateur de robotique MuJoCo [Todorov *et al.*, 2012]. Ces actions abstraites sont souvent appelées *options* [Sutton *et al.*, 1999].

Contrairement à l'AR, l'apprentissage développemental [Piaget and Cook, 1952; Cangelosi and Schlesinger, 2018; Oudeyer and Smith, 2016] s'inspire de la tendance qu'ont les bébés, ou plus généralement tout organisme, à explorer spontanément leur environnement [Gopnik *et al.*, 1999; Georgeon *et al.*, 2011] : c'est ce que nous appelons une motivation intrinsèque, laquelle peut être issue d'une récompense intrinsèque. Ce type de motivation permet d'acquérir de manière autonome de nouvelles connaissances ou

compétences, lesquelles facilitent alors l'apprentissage de nouvelles tâches [Baldassarre and Mirolli, 2013]. Ce paradigme offre une plus grande flexibilité d'apprentissage, de part l'utilisation d'une fonction de récompense plus générale, permettant d'adresser les problèmes soulevés précédemment dans le cas d'une récompense extrinsèque. Typiquement, nous verrons que la motivation intrinsèque peut permette d'inciter l'agent à explorer son environnement ou d'apprendre des *options* indépendantes de sa tâche principale.

Dans cet article nous proposons une étude de l'usage de la motivation intrinsèque dans le framework de l'apprentissage profond par renforcement, plus particulièrement nous souhaitons répondre aux questions suivantes :

- Comment caractériser la motivation intrinsèque ?
- Comment la motivation intrinsèque peut-elle s'intégrer au framework d'AR ?
- Quel rôle joue-t-elle vis-à-vis des défis énoncés ci-dessus ?
- Quel lien existe-t-il entre la motivation intrinsèque et la théorie de l'information ?
- Quelles sont les limites actuelles de l'utilisation de récompenses intrinsèques en AR ?

Nous ne prétendons pas faire une étude exhaustive mais plutôt donner les axes de recherches courants et des perspectives à exploiter.

Dans un premier temps, nous définirons les éléments clés de l'article qui sont les processus de décision markovien, les bases de la théorie de l'information et la motivation intrinsèque (Partie 2). Ensuite nous mettrons en avant les problématiques de l'AR et expliquerons comment combiner l'AR et la motivation intrinsèque (Partie 3). Nous aurons alors les éléments pour détailler les trois différents types de travaux intégrant l'AR et la motivation intrinsèque (Partie 4). Puis, nous prendrons du recul et analyserons les points communs entre les différents travaux (Partie 5). Pour terminer, nous mettrons en avant les défis actuels des modèles intégrant la motivation intrinsèque à l'AR (Partie 6).

2 Définitions

2.1 Processus de décision markovien

L'objectif d'un processus de décision markovien (MDP) est de maximiser l'espérance de récompense reçue via une suite d'interaction. Il est défini par :

- S l'ensemble des états possibles du système.
- A l'ensemble des actions possibles.
- P est la fonction de transition $P : S \times A \times S \to \mathbb{R}$.
- R est la fonction de récompense $R : S \times A \to \mathbb{R}$.
- $\gamma \in [0, 1]$ est le facteur d'atténuation.
- $\rho_0 : S \to \mathbb{R}$ est la distribution initiale de l'état du système.

L'agent démarre dans un état s_0 donné par ρ_0 puis effectue une action a_0. Il attend ensuite la réponse de l'environnement qui lui renverra un nouvel état s', donné par la fonction de transition P, et une récompense r évaluée par la fonction de récompense R. L'agent peut répéter la boucle d'interactions jusqu'à la fin d'un épisode.

L'objectif du MDP est de maximiser la récompense sur le long terme :

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]. \tag{1}$$

Un algorithme de renforcement permet d'associer des actions a aux états s via une politique π. L'objectif de l'agent est alors de trouver la politique π^* maximisant la récompense :

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))\right]. \tag{2}$$

De manière à trouver l'action maximisant la récompense sur le long-terme dans un état s, il est courant de maximiser l'espérance de gain atténuée depuis cet état, noté $V(s)$, ou l'espérance de gain atténuée depuis le couple (état,action) $Q(s, a)$ (c.f équation 3). Cela permet d'adresser le *credit assignment problem* en mesurant le rôle du couple (état,action) dans l'obtention de la récompense cumulée [Sutton and Barto, 1998].

$$Q_\pi(s, a) = E_{a_t \sim \pi(s_t)}\left(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|_{s_0=s, a_0=a}\right). \tag{3}$$

Pour calculer ces valeurs, il est possible d'utiliser l'équation de Bellman [Sutton and Barto, 1998] :

$$Q(s_t, a_t) = R(s_t, a_t) + \gamma Q(P(s_t, a_t), a_{t+1}) \tag{4}$$

Q et/ou π sont souvent approximés via des réseaux de neurones lorsque l'espace d'état est continu ou de très grande taille [Mnih *et al.*, 2016; Lillicrap *et al.*, 2015] .

2.2 Théorie de l'information

L'entropie de Shannon quantifie l'information nécessaire moyenne pour déterminer la valeur d'une variable aléatoire. Soit X une variable aléatoire de probabilité $p(X)$ satisfaisant la condition de normalisation et de positivité, on définit son entropie par :

$$H(X) = -\int_X p(x) \log p(x). \tag{5}$$

L'entropie est maximale lorsque X suit une distribution uniforme, et minimale lorsque $p(X)$ vaut zero partout sauf en une valeur, ce qui est typiquement le cas dans une distribution de Dirac.

On peut aussi définir l'entropie conditionnelle sur la variable aléatoire S. Elle quantifie l'information nécessaire moyenne pour trouver X sachant la valeur d'une autre variable aléatoire S :

$$H(X|S) = -\int_S p(s) \int_X p(x|s) \log p(x|s). \tag{6}$$

L'information mutuelle conditionnelle permet de quantifier l'information que contient une variable aléatoire sur une autre, sachant la valeur d'une troisième variable aléatoire. Elle s'écrit de plusieurs manières :

$$
\begin{aligned}
I(X;Y|S) &= H(X|S) - H(X|Y,S) && (7)\\
&= H(Y|S) - H(Y|A,S) && (8)\\
&= H(X|S) + H(Y|S) - H(X,Y|S)\\
&= \int_{Y,X} p(x,y|s) \log \frac{p(x,y|s)}{p(x|s)p(y|s)} && (9)\\
&= \int_X p(x|s) \int_Y p(y|x,s) \log \frac{p(y|x,s)}{p(y|s)} && (10)\\
&= D_{KL}\Big[p(x,y|s)||p(x|s)p(y|s)\Big] && (11)\\
&= \int_X p(x|s) D_{KL}\Big[p(y|x,s)||p(y|s)\Big] && (12)
\end{aligned}
$$

On voit avec les équations (7) et (8) que l'information mutuelle est symétrique et qu'elle caractérise la baisse d'entropie sur X apportée par Y (ou inversement). L'équation (11) permet de voir l'information mutuelle conditionnelle comme l'écart entre la distribution $P(Y,X|S)$ et cette même distribution si Y et X étaient des variables indépendantes (cas où $H(Y|X,S) = H(Y|S)$). Pour plus de détails sur ces notions, nous renvoyons le lecteur vers [Tishby *et al.*, 2000; Ito, 2016; Cover and Thomas, 2012].

2.3 Motivation intrinsèque

L'idée de la motivation intrinsèque est d'inciter un agent à avoir un certain type de comportement sans que l'environnement intervienne directement. Plus simplement, il s'agit de faire quelque chose pour son inhérente satisfaction plutôt que pour une récompense assignée par l'environnement [Ryan and Deci, 2000]. Ce type de motivation renvoie à l'apprentissage développemental, lequel s'inspire de la tendance des bébés à explorer leur environnement [Gopnik *et al.*, 1999]. Historiquement, la motivation intrinsèque est issue de la tendance des organismes à jouer ou explorer leur environnement sans qu'aucune récompense externe ne leur soit attribuée [White, 1959; Ryan and Deci, 2000].

Plus rigoureusement, Oudeyer [Oudeyer and Kaplan, 2008] explique qu'*une situation est intrinsèquement motivée pour une entité autonome si son intérêt dépend principalement de la collecte ou comparaison d'information depuis différents stimulus indépendamment de leur sémantique.* Le point principal est que l'agent ne doit avoir aucun *a priori* sur la sémantique des observations qu'il reçoit. On remarque que le terme de comparaison d'information renvoie directement à la théorie de l'information ci-dessus. Berlyne [Berlyne, 1965] et Oudeyer [Oudeyer and Kaplan, 2008] proposent plusieurs types de motivation pouvant être caractérisées comme intrinsèques :

TABLE 1 – Types d'apprentissage. Le *feedback* fait ici référence à une supervision experte.

	Avec *feedback*	Sans *feedback*
Actif	Renforcement	Motivation intrinsèque
Passif	Supervisé	Non supervisé

- la nouveauté et la complexité comme étant quelque chose que l'agent ne connaît pas ;
- la surprise et l'incongruité peuvent attirer l'agent car cela remet en question ses précédents apprentissages ;
- l'ambiguïté et l'indistinction renvoient à l'incompréhension de l'agent vis-à-vis des observations.

Typiquement, un étudiant qui fait ses devoirs de mathématique car il les trouve intéressants est intrinsèquement motivé tandis que son camarade qui les fait pour avoir une bonne note est extrinsèquement motivé. De même, jouer avec des jouets pour s'amuser est une motivation intrinsèque tandis que jouer à un jeu télévisé pour gagner de l'argent est une motivation extrinsèque. La notion d'**intrinsèque/extrinsèque** renvoie au *pourquoi de l'action*, à ne pas confondre avec l'internalité/externalité qui renvoie à la localisation de la récompense [Oudeyer and Kaplan, 2008].

La table 1 montre la différence entre l'apprentissage par renforcement et l'usage de motivation intrinsèque. L'apprentissage par renforcement est un apprentissage actif puisque l'agent apprend de ses interactions avec l'environnement, contrairement à des méthodes de classification ou de régression supervisées. L'apprentissage non supervisé est quant à lui un apprentissage passif qui n'utilise pas de labels prédéfinis, donc sans *feedback*. Enfin, le remplacement du *feedback* par une récompense intrinsèque permet de s'affranchir de la supervision experte.

3 Intégrer la motivation intrinsèque à l'AR

Dans cette partie, nous détaillons tout d'abord les deux principaux défis que cherchent à résoudre les travaux combinant récompenses intrinsèques et AR. Ensuite, nous présentons le framework général permettant d'intégrer des récompenses intrinsèques à l'AR.

3.1 Problématiques de l'AR

Les récompenses éparses. Les algorithmes classiques d'AR fonctionnent dans des environnements où les récompenses sont **denses**, *i.e.* que l'agent reçoit une récompense après presque chaque action réalisée. Dans ce type d'environnements, des politiques d'explorations naïves telles que l'exploration ϵ-greedy [Sutton and Barto, 1998] ou l'ajout de bruit gaussien [Lillicrap *et al.*, 2015] sont efficaces. Des méthodes plus élaborées peuvent aussi être utilisées comme l'exploration Boltzmann [Cesa-Bianchi *et al.*, 2017; Mnih *et al.*, 2015], une exploration dans l'espace des

paramètres [Plappert *et al.*, 2017; Rückstiess *et al.*, 2010; Fortunato *et al.*, 2017] ou l'AR bayésien [Ghavamzadeh *et al.*, 2015].

Dans les environnements à récompenses **éparses**, l'agent reçoit un signal de récompense seulement après avoir exécuté une longue séquence spécifique d'actions. Le jeu *Montezuma's revenge* [Bellemare *et al.*, 2015] est un environnement de référence pour illustrer le cas des récompenses éparses. Dans ce jeu, un agent doit se déplacer de salles en salles en y récupérant des objets (clés pour ouvrir les portes, torches, ...). L'agent reçoit une récompense uniquement lorsqu'il trouve des objets ou la sortie d'une salle. Plusieurs actions spécifiques doivent donc être réalisées avant l'obtention d'une récompense. Ce type d'environnements à récompenses éparses est pratiquement impossible à résoudre avec les méthodes d'exploration mentionnées ci-dessus, l'agent parvenant difficilement à apprendre une bonne politique vis-à-vis de la tâche [Mnih *et al.*, 2015].

Plutôt que de travailler sur la politique d'exploration, il est courant de construire une récompense intermédiaire dense qui s'ajoute à celle de la tâche pour faciliter l'apprentissage de l'agent [Su *et al.*, 2015]. Cependant, la construction d'une fonction de récompense fait souvent apparaître des erreurs inattendues [Ng *et al.*, 1999; Amodei *et al.*, 2016] et nécessite le plus souvent des compétences expertes. Par exemple, il est difficile de concevoir une récompense locale pour des tâches de navigation. En effet, il faudrait être capable de calculer le plus court chemin entre l'agent et son objectif, ce qui revient à résoudre le problème de navigation. D'un autre côté l'automatisation de la construction d'une récompense locale (sans faire appel à un expert) demande de trop grandes capacités de calcul [Chiang *et al.*, 2019].

L'abstraction temporelle des actions. L'abstraction temporelle des actions consiste à utiliser des actions de haut niveau, aussi appelées *options*, pouvant avoir des durées d'exécution différentes [Sutton *et al.*, 1999]. A chaque *option* est associée une politique intra-option qui définit les actions (de bas-niveau ou d'autres *options*) à réaliser dans chaque état lorsque l'*option* est exécutée. Abstraire les actions est un élément clé pour accélérer l'apprentissage. En effet, le nombre de choix à réaliser pour atteindre un objectif peut être fortement diminué si des *options* sont utilisées. Cela facilite aussi le renforcement des actions qui sont déterminantes pour l'obtention de la récompense (c'est le *credit assignment problem* [Sutton and Barto, 1998]). Par exemple supposons qu'un robot essaye d'accéder à un gâteau sur une table. Si le robot a une *option* `se rendre à la table` et qu'il la suit, il ne lui restera qu'à prendre le gâteau. Il sera alors facile d'assimiler l'obtention du gâteau à l'option `se rendre à la table`. À l'inverse, si le robot doit apprendre à gérer chacune de ses articulations (actions de bas-niveau), il aura du mal à déterminer quelles actions de bas-niveau lui ont permis d'obtenir le gâteau, parmi toutes celles qu'il a réalisées.

Utiliser des *options* peut par ailleurs faciliter l'exploration lorsque les récompenses sont éparses. Pour illustrer cela, supposons que l'agent ait accès à l'*option* `Aller chercher la clé` dans *Montezuma's revenge*. Le problème devient trivial car une seule action d'exploration peut mener à une récompense là où il faudrait sans *option* une séquence spécifique d'actions de bas-niveau.

Concernant la politique intra-option, elle peut être définie manuellement [Sutton *et al.*, 1999], mais cela nécessite des compétences expertes, ou être apprises avec la fonction de récompenses [Bacon *et al.*, 2017; Riemer *et al.*, 2018], mais les *options* ne seront alors pas réutilisables pour d'autres tâches.

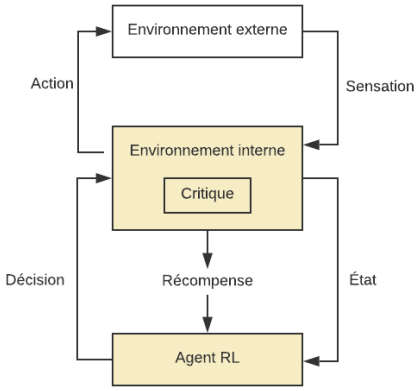


FIGURE 1 – Nouvelle modélisation du MDP. Adaptée de [Singh *et al.*, 2010]

Pour résumer, l'utilisation de la motivation intrinsèque peut aider d'une part, à définir des politiques d'explorations plus élaborées permettant d'améliorer l'AR dans le cas de récompenses éparses ; et d'autre part, à établir une abstraction des décisions ne dépendant pas de l'objectif.

3.2 Nouvelle modélisation de l'AR

L'apprentissage par renforcement est issu du courant behavioriste [Skinner, 1938] et utilise des récompenses extrinsèques [Sutton and Barto, 1998]. Cependant [Singh *et al.*, 2010] et [Barto *et al.*, 2004] ont reformulé le framework de l'AR pour y incorporer la motivation intrinsèque. Plutôt que de considérer l'environnement MDP comme étant l'environnement de l'agent, ils suggèrent que l'environnement MDP peut être constitué d'une partie interne à l'agent et d'une partie externe. La partie externe correspond à l'environnement réel de l'agent et la partie interne est celle qui calcule le signal de récompense total via les observations, actions et récompenses extrinsèques. Dès lors, nous pouvons accepter une récompense intrinsèque comme une récompense de l'environnement MDP. La figure 1 résume le nouveau framework.

L'évolution permet, d'après [Singh *et al.*, 2010], de trouver une fonction de récompense intrinsèque générale qui

maximise une fonction de fitness. Nous pensons que cette motivation intrinsèque peut être une méta-compétence facilitant l'apprentissage d'autres comportements. La curiosité, par exemple, ne génère pas d'avantage sélectif immédiatement mais permet l'acquisition de compétences étant elles-mêmes des avantages sélectifs. Plus largement, l'utilisation de motivation intrinsèque peut permettre d'obtenir des comportements intelligents pouvant servir des objectifs plus efficacement qu'avec du renforcement classique [Lehman and Stanley, 2008] (voir partie 4).

En pratique, la récompense r est souvent calculée comme une somme pondérée de récompense intrinsèque r_{int} et extrinsèque r_{ext} : $r = \alpha r_{int} + \beta r_{ext}$ [Burda *et al.*, 2018; Gregor *et al.*, 2016; Vezhnevets *et al.*, 2017]. Nous pouvons aussi parler de la récompense intrinsèque comme un bonus intrinsèque. Notons que la fonction de récompense évolue au cours du temps, cela ne respecte pas la propriété d'invariance d'un MDP.

4 Principales motivations intrinsèques

Dans cet article, nous proposons de catégoriser les différentes motivations intrinsèques utilisées en AR en trois classes de méta-compétences : l'acquisition de connaissances, l'*empowerment* et la génération d'objectifs. Nous n'adressons pas les motivations à base sociales ou émotionnelles, un état de l'art récent étant déjà disponible [Moerland *et al.*, 2018]. De plus ces motivations font rarement office de méta-compétences mais jouent plutôt le rôle de récompenses auxiliaires [Perolat *et al.*, 2017; Yu *et al.*, 2015; Hughes *et al.*, 2018; Sequeira *et al.*, 2011; 2014; Williams *et al.*, 2015; Moerland *et al.*, 2016] adressant le problème de la fonction de récompense optimale dans l'espace des fonctions de récompenses [Singh *et al.*, 2010].

4.1 Acquisition de connaissances

Les motivations intrinsèques basées sur l'acquisition de connaissances sont les plus utilisées en AR, de part leur capacité à rendre accessible des récompenses éparses. Nous allons étudier les trois principales méthodes existantes pour implémenter une récompense d'acquisition de connaissances. La première utilise le gain d'information, la seconde l'erreur de prédiction des états et la troisième l'évaluation de la nouveauté d'un état. Dans les deux cas, cette récompense intrinsèque permet de compléter la politique d'exploration. Nous focalisons notre étude sur des travaux récents et renvoyons le lecteur vers [Schmidhuber, 2010] pour une revue sur des travaux plus anciens concernant l'acquisition de connaissances.

Le gain d'information. Le gain d'information est une récompense basée sur la réduction de l'incertitude sur la dynamique de l'environnement suite à une action [Oudeyer and Kaplan, 2009; Little and Sommer, 2013], ce qui peut aussi être assimilé au progrès d'apprentissage [Oudeyer and Kaplan, 2009; Schmidhuber, 1991; Frank *et al.*, 2014] ou à la surprise bayésienne [Itti and Baldi, 2006; Schmidhuber, 2008]. Cela permet d'une part de diriger l'agent vers les zones déterministes qu'il ne connaît pas, d'autre part de l'empêcher d'aller dans les zones fortement stochastiques. En effet, si la zone est déterministe, les transitions de l'environnement sont prédictibles et son incertitude à propos des dynamiques de la zone peut baisser. Au contraire, lorsque les transitions sont stochastiques, l'agent se révèle incapable de prédire les transitions et ne réduit pas son incertitude. La stratégie d'exploration VIME [Houthooft *et al.*, 2016] formalise le progrès d'apprentissage de manière bayésienne, l'intérêt de ces approches étant de pouvoir mesurer l'incertitude sur le modèle appris [Blundell *et al.*, 2015]. L'agent apprend donc la dynamique de l'environnement via un réseau de neurone bayésien [Graves, 2011], et utilise la réduction d'incertitude sur la dynamique de l'environnement comme bonus intrinsèque. Autrement dit, l'agent essaye de faire des actions qui sont informatives sur le modèle des dynamiques. Similairement [Achiam and Sastry, 2017] propose de remplacer le modèle bayésien par un réseau de neurones classique suivi d'une distribution de probabilité gaussienne factorisée. Deux approches sont proposées : la première (NLL) utilise comme bonus intrinsèque l'entropie croisée de la prédiction et la seconde (AKL) l'amélioration de prédiction entre un instant t et après k améliorations à $t+k$. Bien que ces deux méthodes soient plus simples que VIME, les bénéfices en terme de performance sont mitigés.

L'erreur de prédiction. L'idée est ici de diriger l'agent vers des zones pour lesquelles la prédiction de l'état suivant est difficile. Plutôt que de considérer la réduction d'erreur dans le modèle des dynamiques, l'agent utilise l'erreur directement comme récompense intrinsèque. Ainsi, au lieu d'utiliser des modèles probabilistes comme précédemment, Dynamic AE [Stadie *et al.*, 2015] calcule la distance entre l'état prédit et l'état réel dans un espace compressé via un auto-encodeur [Hinton and Salakhutdinov, 2006]. Cette distance fait ensuite office de récompense intrinsèque. Cependant cette approche est incapable de gérer la stochasticité locale de l'environnement [Burda *et al.*, 2019]. Par exemple, il s'avère qu'ajouter une télévision affichant aléatoirement des images dans un environnement 3D attire l'agent ; il va alors regarder passivement la télé puisqu'il sera incapable de prédire la prochaine observation. Ce problème est aussi appelé le *problème du bruit blanc* [Pathak *et al.*, 2017; Schmidhuber, 2010]. Une solution serait de s'assurer que les transitions entre états soient apprenables, *i.e.* que la transition n'est pas trop stochastique, mais cette problématique est difficile à résoudre en pratique [Lopes *et al.*, 2012].

Le module de curiosité intrinsèque (ICM) [Pathak *et al.*, 2017] apprend la dynamique de l'environnement dans un espace de caractéristiques. Il construit d'abord une représentation des états en apprenant à prédire les actions réalisées dans un état à partir de cet état et de l'état d'ar-

rivée. Cela restreint la représentation à ce qui peut être contrôlé par l'agent. Il prédit ensuite dans cet espace l'état suivant. L'erreur de prédiction est alors utilisée comme récompense intrinsèque, ainsi l'erreur n'incorpore pas le bruit blanc puisque celui-ci ne dépend pas des actions. ICM permet notamment à un agent d'explorer son environnement dans les jeux *VizDoom* et *Super Mario Bros*. Dans *Super Mario Bros*, l'agent franchit 30% du premier niveau sans récompense extrinsèque. Finalement, en considérant toujours l'erreur de prédiction comme bonus intrinsèque, [Burda *et al.*, 2019] propose un résumé des différentes manières de définir un espace de caractéristiques et montre, d'une part, qu'utiliser des caractéristiques aléatoires peut être performant mais peu généralisable lorsque l'environnement change, d'autre part qu'utiliser l'espace brut d'états (e.g. les pixels) est inefficace. AR4E [Oh and Cavallaro,] réutilise le modèle ICM, mais encode l'action dans un grand espace avant de l'ajouter à l'état courant lors de la prédiction de l'état suivant. Cette astuce améliore ICM, mais il manque une analyse expliquant leurs résultats.

Nouveauté d'un état. Il existe une large littérature sur la mesure de la nouveauté d'un état comme motivation intrinsèque. Une première idée a été d'ajouter un bonus intrinsèque lorsque l'agent se dirige dans un état dans lequel il ne va jamais [Brafman and Tennenholtz, 2002; Kearns and Singh, 2002] ; ces méthodes sont dites basées sur le comptage. Au fur et à mesure qu'il visite un état, la récompense intrinsèque liée à cet état baisse. Bien que cette méthode soit efficace dans un environnement tabulaire (avec un espace d'états discretisé), elle est difficilement applicable lorsque les états sont très nombreux ou continus puisqu'on ne retourne alors jamais dans un même état.

Une première solution proposée par [Tang *et al.*, 2017], nommée TRPO-AE-hash, est de faire un hashage de l'espace d'états lorsqu'il est continu ou très grand. Cependant, les résultats ne sont que légèrement meilleurs que ceux d'une politique d'exploration classique. D'autres tentatives d'adaptation à un très grand espace d'états ont été proposées, comme DDQN-PC [Bellemare *et al.*, 2016], A3C+ [Bellemare *et al.*, 2016] ou DQN-PixelCNN [Ostrovski *et al.*, 2017], qui reposent sur des modèles de densité [Van den Oord *et al.*, 2016; Bellemare *et al.*, 2014]. Ces modèles permettent de calculer le *pseudo-count* [Bellemare *et al.*, 2016], adaptation du comptage permettant la généralisation du décompte d'un état auprès des états avoisinants. Bien que ces algorithmes fonctionnent sur des environnements avec des récompenses éparses, les modèles de densité rajoutent une importante couche de complexité calculatoire [Ostrovski *et al.*, 2017]. Par ailleurs, même si ces modèles gèrent des espaces d'états de très haute dimension (e.g. pixels), ils ne peuvent pas être utilisés avec des espaces d'états continus.

Afin de diminuer la complexité calculatoire induite par les modèles de densité, ϕ-EB [Martin *et al.*, 2017] propose de ne pas modéliser la densité sur l'espace d'états brut, mais sur un espace de caractéristiques dénombrable induit par le calcul de $V(s)$. Plus indirectement, DQN+SR [Machado *et al.*, 2018] utilise la norme de la représentation successeure [Kulkarni *et al.*, 2016b] comme récompense intrinsèque. Pour justifier ce choix, les auteurs expliquent que ce bonus est corrélé au décompte.

Astucieusement, DORA l'exploratrice [Fox *et al.*, 2018] utilise un autre MDP ne contenant aucune récompenses. La valeur d'un état dans ce MDP est biaisée de manière optimiste, de sorte qu'elle baisse au fur et à mesure que l'agent la met à jour. La valeur calculée est utilisée comme approximateur du décompte d'état. Bien que l'approche s'utilise naturellement dans un espace d'états continus, il manque des expérimentations pour la comparer aux approches existantes [Bellemare *et al.*, 2016].

RND [Burda *et al.*, 2018] mesure la nouveauté d'un état en distillant un réseau de neurones aléatoire (dont les poids sont figés) dans un autre réseau de neurones apprenant. Pour chaque état, le réseau de neurones aléatoire génère des caractéristiques aléatoires. Le second réseau apprend à reproduire la sortie du réseau aléatoire pour chaque état. L'erreur de prédiction fait office de récompense. Cela revient à récompenser la nouveauté d'un état puisque l'erreur sera importante tant que le second réseau aura peu visité l'état en question, et sera faible lorsqu'il aura beaucoup appris dessus. L'agent n'arrive cependant pas à apprendre une exploration à long terme. Par exemple dans *Montezuma's revenge*, l'agent utilise ses clés pour ouvrir les premières portes qu'il voit mais n'arrive pas à accéder aux deux dernières portes. De plus, les caractéristiques aléatoires peuvent être insuffisantes pour représenter la richesse d'un environnement.

Nouveauté comme écart aux autres états. Une autre manière d'évaluer la nouveauté d'un état est de l'estimer comme la distance aux états habituellement parcourus. L'exploration informée de [Oh *et al.*, 2015] utilise un modèle de l'environnement pour prédire quelle action lui permettra d'aller dans un état différent des d derniers états visités. Les auteurs utilisent pour cela un noyau gaussien. Cependant ils n'utilisent pas cette distance comme récompense intrinsèque mais comme moyen de choisir l'action en lieu et place de l'aléatoire dans la stratégie ϵ-greedy. Il serait intéressant de l'évaluer comme récompense intrinsèque. EX^2 [Fu *et al.*, 2017] apprend un discriminateur pour différencier les états entre eux : lorsque le discriminateur n'arrive pas à différencier l'état courant de ceux d'un buffer, cela veut dire qu'il est peu allé dans cet état et l'agent recevra un bonus intrinsèque, et inversement lorsqu'il arrive à différencier l'état.

Le module de curiosité épisodique ECO [Savinov *et al.*, 2018] approfondit cette idée en s'inspirant de la mémoire épisodique. Le modèle proposé contient un module de comparaison capable de renvoyer un bonus si l'agent est proche ou loin des états contenus dans un buffer. Ainsi, il calcule la probabilité que le nombre d'actions nécessaire pour aller de l'état sélectionné dans le buffer à l'état

courant soit inférieure à un seuil. En stockant des états éparses dans le buffer, l'agent pose des points de repères dans l'environnement et essaye de s'éloigner de ceux-ci, cela revient à partitionner l'environnement. Le probabilité que l'agent soit écarté de chaque état du buffer est utilisée pour calculer la récompense intrinsèque. Ce modèle a été appliqué sur des environnements 3D comme *DMLab* [Beattie *et al.*, 2016] ou *VizDoom* [Kempka *et al.*, 2016] et permet à l'agent d'explorer l'ensemble de son environnement. Cependant pour calculer la récompense intrinsèque, l'agent doit comparer son observation courante à tous ses états en mémoire. Un passage à l'échelle de cette méthode risque d'être difficile lorsque la richesse de l'espace d'état deviendra plus grande, il serait en effet plus compliqué de partitionner efficacement l'espace d'état. D'un autre côté, l'avantage de cette méthode est que l'agent ne subit pas l'effet du bruit blanc (cf. §4.1).

Dans ces méthodes basées sur le calcul de la nouveauté d'un état, [Stanton and Clune, 2018] distingue la nouveauté inter-épisodes, utilisée par A3C+ [Bellemare *et al.*, 2016] et EX^2 [Fu *et al.*, 2017], et la nouveauté intra-épisodes, que l'on retrouve dans le module de curiosité épisodique ECO [Savinov *et al.*, 2018] et dans l'exploration informée [Oh *et al.*, 2015]. Typiquement, une nouveauté intra-épisodes remettra à zéro les décomptes d'états au début de chaque épisode. Cela pourrait être une piste pour pallier à la difficulté de RND [Burda *et al.*, 2018] concernant l'exploration à long terme.

4.2 *Empowerment*

L'*empowerment* a été développé pour répondre à la question suivante : existe-t-il une fonction d'utilité locale qui rende compte de la survie d'un organisme [Klyubin *et al.*, 2005; Salge *et al.*, 2014b] ? Cette hypothétique fonction devrait être locale dans le sens où elle n'affecte pas le comportement de l'organisme sur le très long terme (la mort en elle-même n'affecte pas cette fonction par exemple) et les comportements induits doivent favoriser la survie de l'espèce. Typiquement, cette fonction peut expliquer la tendance d'un animal à vouloir dominer sa meute, et plus généralement l'envie d'un humain d'acquérir un statut social, d'avoir plus d'argent ou d'être plus fort, le besoin d'avoir un important taux de sucre ou la peur d'être blessé [Klyubin *et al.*, 2005; Salge *et al.*, 2014a]. Chacune de ces motivations permet d'élargir les possibilités d'action de l'agent, et par là son influence : une personne riche pourra faire plus de choses qu'une personne pauvre. Ces motivations sont par ailleurs locales, dans le sens où la récompense est presque immédiate. [Klyubin *et al.*, 2005] nomme cette capacité de contrôle de l'environnement l'*empowerment* d'un agent.

Définition. L'*empowerment* est défini avec la théorie de l'information. Il interprète la boucle d'interaction comme un envoi d'information dans l'environnement : une action est un signal envoyé et l'observation est le signal reçu. Plus l'action est informative sur les observations suivantes, plus l'*empowerment* est élevé. L'*empowerment* est mesuré comme la capacité d'un canal reliant les actions et les observations de l'agent. Soit $a_t^n = (a_t, a_{t+1}, ..., a_{t+n})$ les actions réalisées par l'agent de l'instant t à $t+n$ et s_{t+n} l'état de l'environnement à l'instant $t+n$. L'*empowerment* de l'état s_t, noté $\Sigma(s_t)$, est alors défini par :

$$\Sigma(s_t) = \max_{p(a_t^n)} I(a_t^n; s_{t+n}|s_t) \tag{13}$$

$$= \max_{p(a_t^n)} H(a_t^n|s_t) - H(a_t^n|s_{t+n}, s_t). \tag{14}$$

Maximiser l'*empowerment* revient à rechercher l'état dans lequel l'agent a le plus de contrôle sur l'environnement. Typiquement, le second terme de l'équation 14 permet à l'agent d'être sûr de là où il va, tandis que le premier terme insiste sur la diversité des états accessibles. Pour une vue d'ensemble sur les manières de calculer l'*empowerment*, le lecteur peut se référer à [Salge *et al.*, 2014b]. Nous nous focalisons dans la suite sur l'utilisation de l'*empowerment* dans le cadre de l'AR. Les travaux utilisant l'*empowerment* en dehors de ce contexte, e.g. [Karl *et al.*, 2017; Guckelsberger *et al.*, 2016; Capdepuy *et al.*, 2007; Salge *et al.*, 2014b], ne sont pas détaillés dans cet article.

L'*empowerment* en tant que récompense intrinsèque. En AR, l'agent maximisant l'*empowerment* est donc récompensé s'il se dirige dans des zones où il contrôle son environnement. Comme l'objectif de l'agent est de maximiser la fonction de récompense intrinsèque, celle-ci est définie par :

$$\begin{aligned} R_{int}(s, a, s') &= \Sigma(s') \\ &\approx -\mathbb{E}_{\omega(a|s)} \log \omega(a|s) \\ &\quad + \mathbb{E}_{p(s'|a,s)\omega(a|s)} \log p(a|s, s'). \end{aligned} \tag{15}$$

où $\omega(a|s)$ est la distribution choisissant les actions a_t^n. Dans l'idéal, $\omega(a|s)$ est la distribution maximisant l'équation 15 conformément à l'équation 14.
Le problème est que $p(a|s, s')$ est difficile à obtenir car il nécessite $p(s'|a, s)$ ce qui implique l'utilisation d'un modèle de densité.

[Mohamed and Rezende, 2015] propose de calculer l'*empowerment* en approximant l'équation 15. Pour cela, il calcule une borne inférieure à cette information mutuelle, cette borne sera ensuite reprise par plusieurs travaux :

$$I(a; s'|s) \geq H(a|s) + \mathbb{E}_{p(s'|a,s)\omega(a|s)} \log q_\xi(a|s, s'). \tag{16}$$

Son idée est de faire apprendre l'approximateur q_ξ de la distribution de probabilité $p(a|s, s')$ de manière supervisée sur les données que reçoit l'agent de l'environnement en utilisant la méthode du maximum de vraisemblance. Son approche permet de généraliser le calcul de l'*empowerment* à des observations continues. Dans ces travaux, les expérimentations montrent que la

maximisation de l'*empowerment* est notamment utile dans des environnements dynamiques, c'est-à-dire des environnements qui modifient l'état de l'agent même s'il fait une action stationnaire. Il prend l'exemple de l'environnement classique proie-prédateur : la proie est l'apprenant et a intérêt à éviter de se faire attraper car si elle meurt, elle n'aura plus aucun contrôle sur ses états suivants. Implicitement, la proie évite donc de mourir en maximisant son *empowerment*. Au contraire d'un environnement dynamique, un environnement statique admet une politique optimale statique (l'agent ne bouge plus lorsqu'il a trouvé le meilleur état) rendant l'*empowerment* comme récompense intrinsèque moins intéressant vis-à-vis d'une tâche. Les expérimentations proposées dans [Mohamed and Rezende, 2015] utilisent cependant la planification pour estimer l'*empowerment* et non des interactions avec l'environnement pour récupérer les données, ce qui implique l'utilisation d'un modèle de l'environnement.

[Gregor *et al.*, 2016] essaie de maximiser l'*empowerment* via des interactions avec l'environnement en utilisant $\omega(a|s) = \pi(a|s)$. La récompense devient alors :

$$R(a, h) = -\log \pi(a|h) + \log \pi(a|s', h) \qquad (17)$$

où h est l'historique d'observation (dont l'observation courante) et d'actions. Ses expérimentations sur divers environnements montrent que les trajectoires apprises mènent à des zones diverses et qu'un pré-entrainement via l'*empowerment* aide à apprendre une tâche. Les tâches apprises restent cependant relativement simples.

L'*empowerment* peut aussi se révéler intéressant en AR multi-agents. L'AR multi-agents fonctionne similairement à l'AR mono-agent, sauf que plusieurs agents apprennent simultanément à résoudre une tâche et doivent se coordonner. [Jaques *et al.*, 2018] a montré que dans un jeu non coopératif de type dilemme social [Leibo *et al.*, 2017], la récompense intrinsèque d'un agent pouvait stabiliser l'apprentissage en compensant la baisse de récompense individuelle due à une politique maximisant la récompense long-terme de l'ensemble des agents.

Conclusion. La principale difficulté de l'utilisation de l'*empowerment* en AR est donc son calcul. La plupart des approches utilisent un modèle de l'environnement pour calculer la récompense intrinsèque basée sur l'*empowerment* [Mohamed and Rezende, 2015; de Abril and Kanai, 2018]. Cependant le coeur même de l'AR est que l'agent ne connaît pas *a priori* la dynamique de l'environnement ou la fonction de récompense. Les travaux existants dans ce contexte restent donc limités et ne suffisent pas à montrer le potentiel de l'*empowerment* pour aider l'apprentissage.
Il est intéressant de noter que l'*empowerment* peut pousser un agent à apprendre des comportements même dans un environnement *a priori* statique. En effet, supposons que l'agent choisisse non pas des actions primitives directement, mais des *options*. S'il n'a pas encore appris les *options*, il est incapable de les discerner, c'est donc comme si l'agent n'avait aucun contrôle sur son environnement. Si au contraire, ses *options* sont parfaitement distinguées dans l'espace d'état, l'agent a le contrôle de son environnement. Or il s'agit là non pas de choisir les états maximisant l'*empowerment* mais de définir des *options* qui augmentent l'*empowerment*. Nous revenons sur ce point dans la partie 4.3.

4.3 Génération d'objectifs

La génération d'objectifs est la capacité d'un agent à apprendre des compétences diverses de manière non supervisée. Les compétences ou objectifs générés par l'agent sont des *options* (cf. §3.1). Comparé à l'AR multi-objectifs [Liu *et al.*, 2015], les compétences sont ici générées de manière non supervisée. Dans les travaux utilisant la génération d'objectifs, l'agent apprend généralement sur deux échelles de temps : il génère des *options* et apprend les politiques intra-option associées en utilisant une récompense intrinsèque ; si un objectif global existe, il apprend à utiliser ces compétences pour réaliser cet objectif global, appelé tâche, en utilisant la récompense extrinsèque associée à la tâche. L'intérêt est d'une part d'apprendre des compétences intéressantes pouvant servir à plusieurs tâches, or elles le seront d'autant plus qu'elles sont décorrélées de la tâche apprise [Heess *et al.*, 2016]. D'autre part, l'abstraction temporelle des actions réalisée via les compétences facilite l'apprentissage. Prenons l'exemple de MuJoCo [Todorov *et al.*, 2012] qui est un environnement souvent utilisé dans les travaux sur la génération d'objectifs. Dans cet environnement, les articulations d'un robot peuvent être contrôlées par un agent pour accomplir par exemple des tâches de locomotion. L'idée de certains travaux est donc de générer des compétences de type *avancer* ou *reculer* avec une récompense intrinsèque. Ces compétences peuvent alors servir à une tâche de navigation.

Classiquement, l'AR a un seul objectif et n'apprend pas à réaliser plusieurs objectifs. Une manière de généraliser l'AR profond à l'apprentissage de plusieurs objectifs, voir à tous les objectifs possibles dans l'espace d'état, est d'utiliser l'approximateur de fonction de valeur universelle (UVFA) [Schaul *et al.*, 2015]. UVFA intègre la représentation de l'état objectif dans l'observation de l'agent. La politique trouvée est alors conditionnée sur l'objectif : $\pi(s)$ devient $\pi(s, g)$ où g est un objectif. La même idée peut être retrouvée avec la recherche de politiques contextuelles [Fabisch and Metzen, 2014]. Ainsi, les travaux apprenant les *options* avec une motivation intrinsèque apprennent des politiques $\pi(s, g)$. Bien que l'espace d'exploration augmente alors, [Andrychowicz *et al.*, 2017] améliore l'efficacité des données en apprenant sur plusieurs objectifs à la fois via une seule interaction.

En effet, via une interaction (s, s', r_g, a, g), il est possible de créer une nouvelle interaction avec un nouvel objectif (lequel serait réussi) $(s, s', r_{g'}, a, g')$ tant qu'une fonction de récompense $R(s, a, s', g)$ est accessible, ce qui est généralement le cas lorsque la récompense est intrinsèque.

Dans la suite, nous allons présenter plusieurs travaux incorporant des récompenses expertes dans un algorithme hiérarchique. Ensuite nous étudierons les deux principaux ensembles de travaux portant sur l'auto-génération d'objectifs. La première approche utilise l'espace d'états pour générer les objectifs et calculer la récompense intrinsèque ; la seconde utilise la théorie de l'information.

Entre récompenses expertes et récompenses intrinsèques. Il existe quelque travaux précurseurs montrant l'intérêt de la décomposition hiérarchique des actions. Parmi ceux-ci, [Kulkarni *et al.*, 2016a] présente le modèle *hierarchical-DQN* dans lequel la représentation des objectifs est définie de manière experte via des tuples $(entite, relation, entite2)$. Une entité peut être un objet à l'écran ou l'agent, et la relation renvoie notamment à une distance. Ainsi l'objectif peut être que l'agent atteigne un objet. La récompense experte vaut simplement un si l'objectif est atteint, zéro sinon. Il montre que cela peut aider l'apprentissage notamment lorsque les récompenses sont éparses comme dans *Montezuma's revenge*. Cependant, s'affranchir de l'apprentissage de la représentation des compétences revient à esquiver le problème principal : il est en effet difficile de choisir quelles caractéristiques sont assez intéressantes pour être des objectifs dans un grand espace d'état. D'autres travaux démontrent le potentiel de l'approche en utilisant des objectifs auxiliaires spécifiques à la tâche [Riedmiller *et al.*, 2018] ou plus abstraits [Dilokthanakul *et al.*, 2019; Rafati and Noelle, 2019]. Plus particulièrement, une heuristique régulièrement utilisée pour générer une compétence est la recherche d'états faisant office de goulots d'étranglements [McGovern and Barto, 2001; Menache *et al.*, 2002]. Il s'agit d'identifier des états charnière quant aux prochains états visités (par exemple, une porte). De récents travaux [Zhang *et al.*, 2019; Tomar *et al.*, 2018] utilisent la représentation successeure [Kulkarni *et al.*, 2016b] pour généraliser l'approche à des espaces d'états continus. Le désavantage de ce type de travaux est que les récompenses ne sont pas suffisamment générales pour s'appliquer à tous les environnements et nécessitent une expertise.

Distance entre objectifs. Certains travaux utilisent l'espace d'états pour créer un espace d'objectif, l'intérêt est de pouvoir se servir de chaque état comme objectif. Ainsi, *Hierarchical Actor-Critic* (HAC) [Levy *et al.*, 2019] se sert directement de l'espace d'états comme espace d'objectif pour apprendre trois niveaux d'options (les *options* du second niveau sont choisies de manière à répondre à l'*option* du troisième niveau). Une distance entre l'objectif et l'état final fait donc office de récompense intrinsèque. Au contraire, HIRO [Nachum *et al.*, 2018] utilise comme objectif la différence entre l'état initial et l'état à la fin de l'objectif ; la récompense intrinsèque est alors une distance entre la direction prise et l'objectif. Les objectifs permettent ainsi d'orienter les compétences vers certaines zones spatiales. Cependant, il y a deux problèmes dans l'utilisation de l'espace d'états comme espace d'objectifs. D'une part une distance (comme L2) a peu de sens dans un espace très grand comme une image composée de pixels, d'autre part il est difficile de faire fonctionner un algorithme d'AR sur un espace d'action trop grand. Concrètement, un algorithme ayant comme espace d'objectif des images peut impliquer pour la politique d'*options* un espace d'action de 84x84 dimensions. Un espace d'action aussi large est actuellement inconcevable, aussi, ces algorithmes ne fonctionnent que sur des espaces d'états de faible dimension. Pour pallier ce problème, FuN [Vezhnevets *et al.*, 2017] utilise comme récompense intrinsèque la direction prise dans un espace de caractéristiques d'états. Les caractéristiques utilisées sont celles qui sont utiles à la tâche, elles sont donc construites par la rétro-propagation de la récompense extrinsèque. L'agent apprend donc uniquement des compétences liées à la tâche et a besoin d'un accès à la récompense extrinsèque.

Le problème est finalement de construire une bonne représentation de l'espace d'états [Schwenker and Palm, 2019] faisant office d'espace d'objectifs, i.e. choisir les informations à ne pas perdre lors de la compression des états en une nouvelle représentation. Ainsi, la distance entre deux objectifs aurait un sens et serait une bonne récompense intrinsèque. Pour construire un espace de caractéristiques, RIG [Nair *et al.*, 2018] se sert d'un auto-encodeur variationel (VAE) [Kingma and Welling, 2013], mais ce type d'approche peut-être très sensible à des distracteurs (i.e. des caractéristiques inutiles à la tâche ou l'objectif, présentes dans les états) et ne permet donc pas de pondérer les caractéristiques correctement. [Zhou *et al.*, 2019] utilise quant à lui des méthodes non supervisées comme l'algorithme de *slow features analysis* [Wiskott and Sejnowski, 2002] et le réseau *growing when required* [Marsland *et al.*, 2002] pour construire une carte topologique. Un agent hiérarchique se sert ensuite des noeuds de la carte comme espace d'objectif. Ils ne se comparent cependant pas aux autres approches. Sub-optimal representation learning [Nachum *et al.*, 2019] essaye de borner la sous-optimalité de la représentation des objectifs, offrant des garanties théoriques. L'agent s'avère capable d'apprendre à aller partout en sélectionnant les caractéristiques importantes pour la tâche. Cependant, comme FuN [Vezhnevets *et al.*, 2017], l'agent a besoin d'une récompense dense. [Florensa *et al.*, 2019] reformule l'équation de Bellman et présente des perspectives intéressantes en apprenant une représentation d'états pour laquelle la distance L2 entre deux états correspond au nombre d'actions à effectuer pour aller d'un état à l'autre. Il manque cependant des expérimentations montrant son intérêt.

Information mutuelle entre objectif et trajectoire. Une deuxième approche ne nécessite pas de fonction de distance mais consiste à maximiser l'information mutuelle entre un objectif et sa trajectoire associée. Informellement, il s'agit d'apprendre des compétences selon la capacité de l'agent à les distinguer entre elles à partir de la trajectoire (i.e. les états parcourus) de la politique de la compétence choisie. Dans cette section, nous appelons $I(g;c)$ l'information mutuelle entre g, un objectif, et c, une partie (changeante selon les travaux) de la trajectoire.

SNN4HRL [Florensa *et al.*, 2017] apprend des compétences en maximisant l'information mutuelle $I(g;c)$ où c est un agrégé de la trajectoire. Chaque objectif est généré de manière uniforme donc maximiser l'information mutuelle revient à minimiser $H(g|c)$ (cf. équation 7). Or, c'est équivalent de maximiser la récompense intrinsèque $\log q(g|c)$ (équation 16), où q est la probabilité prédite par un modèle. Pour calculer cet élément, il discrétise l'espace d'états et calcule la probabilité d'accomplir son objectif courant dans son état courant. Pour calculer cette probabilité, l'agent compte le nombre de fois où il a parcouru cette partition pour chaque objectif. Ensuite, l'agent ayant appris les compétences est intégré dans une structure hiérarchique dans laquelle un manager choisit les objectifs à accomplir. Notons que l'espace d'objectifs est ici discret.

VALOR [Achiam *et al.*, 2018] et DIAYN [Eysenbach *et al.*, 2018] reprennent la même idée, mais se distinguent des travaux précédents en utilisant un réseau de neurone plutôt qu'une discrétisation pour calculer $\log q(g|c)$ et en choisissant c comme une partie de la trajectoire de la compétence dans l'environnement. Ils réussissent à faire apprendre à un agent des tâches de locomotion dans des espaces d'état ayant plus de 100 degrés de liberté. De plus, ils montrent l'intérêt de cette méthode utilisée comme pré-entrainement pour de l'apprentissage hiérarchique et comme initialisation pour l'apprentissage d'une tâche. DIAYN choisit c comme un état de la trajectoire et calcule la récompense intrinsèque à chaque itération de la trajectoire. VALOR se distingue en considérant c comme un agrégé de la trajectoire d'états et en assignant la récompense à la fin de la trajectoire. Avec VALOR, l'agent parvient à apprendre jusqu'à 10 compétences différentes et jusqu'à 100 en augmentant peu à peu le nombre d'objectifs via un curriculum [Achiam *et al.*, 2018]. VIC [Gregor *et al.*, 2016] avait déjà fait quelques expériences avec la même approche, mais sur des environnements plus simples et sans exhiber la même diversité de comportements.

Deux principales limites à cette approche peuvent être distinguées :

1. l'agent est incapable d'apprendre à générer des objectifs sans désapprendre ses compétences. Ainsi, la distribution d'objectifs générée par l'agent doit rester uniforme [Gregor *et al.*, 2016; Eysenbach *et al.*, 2018].
2. Le calcul de $\log q(g|c)$ implique par ailleurs que l'espace d'objectifs soit discrétisé. Il est donc impossible d'utiliser des *options* continues. DISCERN [Warde-Farley *et al.*, 2018] s'attaque à ce problème en considérant l'espace d'objectifs comme l'espace d'états. Ensuite il fait une approximation de $\log q(g|c)$ en essayant de classifier l'état final de la trajectoire auprès du bon objectif parmi d'autres objectifs tirés de la même distribution que le vrai objectif. Cela revient à apprendre à trouver l'objectif le plus proche de l'état final depuis une liste d'objectifs.

Nous avons énoncé au §4.2 que l'*empowerment* d'un agent s'améliorait au fur à et mesure que ses compétences se distinguaient. Les travaux présentés ici augmentent implicitement l'*empowerment* d'un agent, du point de vue de la politique d'options. En effet ils maintiennent une entropie sur les objectifs élevées et associent une direction dans l'espace d'état à un objectif. Ainsi, $H(a|s)$ est maximale, puisque la distribution de probabilité est uniforme, et $H(a|s,s')$ se réduit au fur et à mesure que l'agent apprend à distinguer les *options*.

Choisir la compétence à apprendre en estimant le progrès d'apprentissage. D'autres travaux sont à l'intersection entre l'apprentissage de compétences et le progrès d'apprentissage, l'idée est de choisir l'objectif qui ne soit ni trop difficile, ni trop facile pour faciliter l'apprentissage de l'agent. Goal GAN [Florensa *et al.*, 2018] apprend à générer des objectifs de plus en plus complexes (via un GAN [Goodfellow *et al.*, 2014]) de manière à ce que l'agent sache aller partout par la suite, mais utilise une fonction de récompense manuelle : il suppose l'accès aux coordonnées de l'agent pour calculer une récompense binaire. Nous ne détaillerons pas plus ces travaux qui supposent pour la plupart que la représentation de l'objectif et la fonction de récompense associée sont donnés [Fabisch and Metzen, 2014; Deisenroth *et al.*, 2013].

Conclusion. Pour résumer, il existe deux principaux ensembles de travaux portant sur l'auto-génération d'objectifs. Le premier ensemble apprend ses objectifs à l'aide de l'espace d'états, l'avantage est alors d'avoir un espace continu et lisse d'objectifs permettant l'interpolation. Le désavantage est qu'il faut trouver la bonne métrique de comparaison et la juste manière de compresser l'espace d'états. Le second ensemble de travaux utilise la théorie de l'information, cela force l'utilisation d'un espace discret d'objectifs mais évite les complications associées à un espace continu qui est originellement de haute dimension.

4.4 Conclusion

D'autres approches incorporant la motivation intrinsèque dans l'AR ne sont pas directement liées à ces trois grandes catégories de travaux et ne sont donc pas détaillées dans cet article [Hester and Stone, 2017; Machado *et al.*, 2017a; Lakshminarayanan *et al.*, 2016; Machado *et al.*, 2017b; Stanton and Clune, 2018; Still and Precup, 2012; Little and Sommer, 2013; Frank *et al.*, 2014; Montúfar *et al.*, 2016].

De même que les méthodes essayant de combiner plusieurs motivations intrinsèques [Hester and Stone, 2017; de Abril and Kanai, 2018], qui nécessiteraient cependant des expérimentations sur des environnements plus compliqués.

5 Analyse des travaux

Dans cette section, nous allons étudier les points communs entre les travaux présentés afin de mettre en avant des perspectives de recherche.

5.1 Information mutuelle comme point commun

Une redondance semble apparaître tout au long de notre étude des différents travaux, que ce soit via l'acquisition de connaissance, l'*empowerment* ou l'apprentissage d'objectifs. L'information mutuelle apparait comme centrale pour élargir les capacités d'un agent.

Utilisation directe de l'information mutuelle. Nous avons d'abord vu que l'*empowerment* est entièrement défini via l'information mutuelle (cf. §4.2). VIME [Houthooft *et al.*, 2016] et AKL [Achiam and Sastry, 2017] maximisent le gain d'information, c'est-à-dire l'information que contient le prochain état sur le modèle de l'environnement. Plusieurs travaux sur la génération d'objectifs [Eysenbach *et al.*, 2018; Achiam *et al.*, 2018; Florensa *et al.*, 2017] utilisent directement l'information mutuelle entre la trajectoire issue d'un objectif et l'objectif en lui-même pour récompenser l'agent. [Still and Precup, 2012] suggère que l'agent doit maximiser l'information mutuelle entre son action et les état suivants pour améliorer la politique d'exploration.

Fonctions équivalentes à l'information mutuelle. L'erreur de prédiction [Nachum *et al.*, 2019; 2018; Pathak *et al.*, 2017] est aussi relative à l'information mutuelle [de Abril and Kanai, 2018], puisque la prédiction de l'état suivant un couple (état,action) peut être correcte seulement si l'action contient de l'information sur l'état suivant. Notons toutefois que les états suivants possibles doivent être suffisamment divers. Par ailleurs, [Nachum *et al.*, 2019] explique que sa méthode apprend une représentation d'états maximisant l'information mutuelle entre l'état en question et les états suivants. Finalement, [Bellemare *et al.*, 2016] a montré que la récompense issue du *pseudo-count* [Bellemare *et al.*, 2016; Ostrovski *et al.*, 2017] est proche de celle du gain d'information.

Généralisation des modèles utilisés. Globalement, les modèles étudiés ont souvent comme point commun de contenir deux modules :

1. Le premier est un module cherchant une fonction d'évaluation entre les actions et états parcourus par l'agent (sa dernière trajectoire [Eysenbach *et al.*, 2018; Achiam *et al.*, 2018], sa dernière action [Stadie *et al.*, 2015; Pathak *et al.*, 2017; Burda *et al.*, 2019], le nombre de fois où chaque état a été parcouru [Bellemare *et al.*, 2016; Ostrovski *et al.*, 2017] ou les dernières trajectoires [Savinov *et al.*, 2018; Fu *et al.*, 2017; Oh *et al.*, 2015] ...) et une autre source de données (un objectif [Eysenbach *et al.*, 2018; Achiam *et al.*, 2018], les états suivants [Pathak *et al.*, 2017; Mohamed and Rezende, 2015] ...). Cette fonction est souvent une fonction de causalité.
2. Le deuxième est une politique maximisant une récompense intrinsèque issue de la fonction d'évaluation.

5.2 Motivation intrinsèque comme compression de l'information

Compression de l'information. Schmidhuber propose que l'organisme est guidé par le désir de compresser l'information qu'il reçoit [Schmidhuber, 2008]. Ainsi, plus nous arrivons à compresser les données reçues de l'environnement, plus la récompense intrinsèque reçue est élevée. Il note toutefois que c'est l'amélioration de la compression qui est importante et non le degré de compression en lui même, sous peine qu'un agent soit inactif devant du bruit ou devant une obscurité uniforme. Or la compression de données est fortement liée à l'observation de régularités dans ces mêmes données. Par exemple ce que nous appelons visage est, dans notre environnement, un ensemble, apparaissant de manière récurrente, composé d'une forme ovale contenant deux yeux, un nez et une bouche. De même, un état de l'environnement peut être décrit avec quelques-unes des caractéristiques les plus pertinentes. Cela implique que la motivation intrinsèque se traduit par une recherche de nouvelles régularités dans l'environnement.

Lien avec les travaux. Il a été montré dans [Schmidhuber, 2008; Houthooft *et al.*, 2016] que les travaux sur le gain d'information sont directement liés au progrès de la compression d'information ; le module de curiosité épisodique [Savinov *et al.*, 2018] essaye d'encoder l'environnement en sauvegardant les états les plus diversifiés possibles ; et les modèles prédictifs [Burda *et al.*, 2019] encodent les dynamiques de l'environnement dans un modèle paramétré (souvent un réseau de neurone). L'*empowerment* est similaire, rappelons qu'il s'agit de diriger l'agent vers une zone dans laquelle il a du contrôle, i.e. que les états sont déterminés par les actions de l'agent. Il est possible de reformuler l'*empowerment* comme l'intérêt d'un agent pour des zones où ses actions sont une compression des états suivants. En effet, l'*empowerment* est maximal si chaque trajectoire mène à ses propres états (toujours les mêmes dans le même ordre) distincts de ceux des autres trajectoires ; tandis qu'il est minimal si toutes les trajectoires mènent à un même état. Les travaux sur la génération d'objectifs cherchent explicitement à compresser des trajectoires dans un espace d'objectifs. Enfin, une partie des travaux [Vezhnevets *et al.*, 2017; Nachum *et al.*, 2019; Pathak *et al.*, 2017] repose sur la qualité d'une compression

de l'espace d'état.

5.3 Conclusion

Pour résumer, l'ensemble des travaux s'attache à compresser les nouvelles régularités détectées dans les trajectoires de l'agent. Pour cela, la théorie de l'information est un puissant outil de mesure.

6 Limites et challenges

Plusieurs travaux sont limités par des problématiques sortant du cadre de l'AR, telles que les performances des modèles de densité [Bellemare *et al.*, 2016; Ostrovski *et al.*, 2017], la difficulté d'approximation de l'information mutuelle entre deux variables aléatoires continues [Gregor *et al.*, 2016] ou les performances des modèles prédictifs [Nachum *et al.*, 2018; Nair *et al.*, 2018]. Ces limites dépassent le cadre de cet article. Aussi, malgré l'hétérogénéité des travaux sur la motivation intrinsèque en AR, et les limitations propres à chacune de ses méthodes, nous avons identifié et présentons dans cette section quatre problématiques majeures qui sont communes à l'ensemble des approches.

6.1 Stochasticité de l'environnement

Nous avons vu dans la partie précédente qu'il était intéressant de maximiser le progrès de compression et que la plupart des travaux étaient relatifs à la compression d'information, et non au progrès de la compression. Cet écart explique la difficulté de plusieurs travaux [Burda *et al.*, 2019] à gérer l'effet du bruit blanc [Schmidhuber, 2010] ou plus généralement la stochasticité de l'environnement. Certains travaux de l'état de l'art gèrent cette problématique [Savinov *et al.*, 2018; Pathak *et al.*, 2017; Burda *et al.*, 2018; Houthooft *et al.*, 2016], mais chacun avec ses défauts.

6.2 Acquisition de connaissance sur le long-terme

A notre connaissance, aucune approche existante n'est capable de gérer la recherche d'information long-terme [Burda *et al.*, 2018]. Dans *Montezuma's revenge*, il s'agit d'éviter d'utiliser une clé trop rapidement pour pouvoir l'utiliser plus tard. Dans la vie de tous les jours, il peut s'agir d'éviter de dépenser son argent trop vite. Cette difficulté pourrait être résolue avec une approche utilisant la planification [Hafner *et al.*, 2018]. L'apprentissage hiérarchique de compétences pourrait apporter une solution, en transformant le long terme en court terme via une hiérarchie de compétences multi-niveaux [Riemer *et al.*, 2018]. Finalement, il manque d'autres environnements que *Montezuma's revenge* mettant vraiment en avant cette problématique.

6.3 Construire une représentation des états

Nous avons vu que construire de bonnes caractéristiques d'état est important dans la découverte d'objectifs afin de travailler sur un espace d'objectifs réduit. C'est aussi primordial dans les travaux sur l'acquisition de connaissances pour avoir une erreur de prédiction significative. Le module de curiosité intrinsèque ICM [Pathak *et al.*, 2017] propose une représentation des états intéressante, restreinte à ce qui peut être contrôlé par l'agent, mais sa limite est que le module apprend une partie suffisante des caractéristiques permettant de déterminer l'action, et non l'ensemble des caractéristiques déterminées par l'action. Il manque encore un moyen de compresser parfaitement l'espace d'états dans l'espace des caractéristiques contrôlées par l'agent. [Florensa *et al.*, 2019] est allé dans ce sens en apprenant une représentation d'états pour laquelle la distance L2 entre deux états correspond au nombre d'actions à effectuer pour aller d'un état à l'autre, mais il manque des expérimentations. Par ailleurs, [Lesort *et al.*, 2018] présente plusieurs pistes pouvant améliorer la représentation des états d'un agent.

6.4 Décorréler les objectifs de la tâche

L'avantage de décorréler l'apprentissage des objectifs de l'apprentissage d'une tâche est de favoriser l'exploration et le transfert d'apprentissage. On parle alors d'apprentissage *bottom-up* car on apprend les compétences avant la tâche. Si cet apprentissage a fait des progrès significatifs, il est encore impossible d'apprendre des tâches spécifiques en même temps que les compétences permettant de les réaliser sans subir d'oubli catastrophique [McCloskey and Cohen, 1989; Florensa *et al.*, 2018]. En effet, lorsque l'agent apprend séquentiellement des tâches, il oublie les premières tâches en apprenant les suivantes. Des travaux adressant le problème d'oubli catastrophique existent déjà [Kirkpatrick *et al.*, 2017; Parisi *et al.*, 2019] mais ils n'ont, à notre connaissance, pas été évalués avec la motivation intrinsèque et un large nombre de tâches.

7 Conclusion

L'AR fait face à plusieurs défis, comme l'apprentissage avec des récompenses éparses ou l'abstraction des actions de l'agent en décisions de plus haut niveau. Nous avons vu que la motivation intrinsèque pouvait être utilisée en AR et que ses nombreuses applications pouvait résoudre partiellement ces problématiques. Plusieurs types de motivations intrinsèques existent comme méta-compétences, chacune avec leur littérature. Parmi celles-ci, l'acquisition de connaissances est effectuée via des modèles prédictifs, des modèles bayésiens ou des modèles de densité pour inciter l'agent à explorer l'environnement. L'*empowerment* est une motivation universelle poussant l'agent à avoir des comportements divers tels que la survie. La génération d'objectifs est directement liée à l'*empowerment* et permet d'abstraire les actions de l'agent, aidant à résoudre le *credit assignment problem*. Lorsque l'apprentissage est de type *bottom-up*, l'abstraction des décisions facilite l'exploration et le transfert d'apprentissage. Plusieurs défis restent cependant à adresser : les méchanismes d'acquisition de connaissance gèrent difficilement la stochasticité de l'environnement et ont encore des difficultés à générer des trajectoires utiles à l'exploration sur le long terme ; avoir des

représentations d'état plus significatives pourrait ouvrir de nouvelles perspectives pour la génération d'objectifs ; les travaux apprenant des compétences multi-tâches souffrent encore de l'oubli catastrophique. Notre analyse suggère que la théorie de l'information, en permettant de compresser l'information contenue dans les séquences d'interactions de l'agent, devrait jouer un rôle proéminent dans la résolution des défis mentionnés.

Références

[Achiam and Sastry, 2017] Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv :1703.01732*, 2017.

[Achiam *et al.*, 2018] Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv :1807.10299*, 2018.

[Amodei *et al.*, 2016] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv :1606.06565*, 2016.

[Andrychowicz *et al.*, 2017] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[Bacon *et al.*, 2017] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, pages 1726–1734, 2017.

[Baldassarre and Mirolli, 2013] Gianluca Baldassarre and Marco Mirolli. Intrinsically motivated learning systems : an overview. In *Intrinsically motivated learning in natural and artificial systems*, pages 1–14. Springer, 2013.

[Barto *et al.*, 2004] Andrew G Barto, Satinder Singh, and Nuttapong Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning*, pages 112–19, 2004.

[Beattie *et al.*, 2016] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv :1612.03801*, 2016.

[Bellemare *et al.*, 2014] Marc Bellemare, Joel Veness, and Erik Talvitie. Skip context tree switching. In *International Conference on Machine Learning*, pages 1458–1466, 2014.

[Bellemare *et al.*, 2015] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment : An evaluation platform for general agents (extended abstract). In *IJCAI*, pages 4148–4152. AAAI Press, 2015.

[Bellemare *et al.*, 2016] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[Berlyne, 1965] Daniel E Berlyne. Structure and direction in thinking. 1965.

[Blundell *et al.*, 2015] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv :1505.05424*, 2015.

[Brafman and Tennenholtz, 2002] Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct) :213–231, 2002.

[Burda *et al.*, 2018] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv :1810.12894*, 2018.

[Burda *et al.*, 2019] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations*, 2019.

[Cangelosi and Schlesinger, 2018] Angelo Cangelosi and Matthew Schlesinger. From babies to robots : the contribution of developmental robotics to developmental psychology. *Child Development Perspectives*, 12(3) :183–188, 2018.

[Capdepuy *et al.*, 2007] Philippe Capdepuy, Daniel Polani, and Chrystopher L Nehaniv. Maximization of potential information flow as a universal utility for collective behaviour. In *2007 IEEE Symposium on Artificial Life*, pages 207–213. Ieee, 2007.

[Cesa-Bianchi *et al.*, 2017] Nicolò Cesa-Bianchi, Claudio Gentile, Gábor Lugosi, and Gergely Neu. Boltzmann exploration done right. In *Advances in Neural Information Processing Systems*, pages 6284–6293, 2017.

[Chiang *et al.*, 2019] Hao-Tien Lewis Chiang, Aleksandra Faust, Marek Fiser, and Anthony Francis. Learning navigation behaviors end-to-end with autorl. *IEEE Robotics and Automation Letters*, 4(2) :2007–2014, 2019.

[Cover and Thomas, 2012] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[de Abril and Kanai, 2018] Ildefons Magrans de Abril and Ryota Kanai. A unified strategy for implementing curiosity and empowerment driven reinforcement learning. *arXiv preprint arXiv :1806.06505*, 2018.

[Deisenroth *et al.*, 2013] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2) :1–142, 2013.

[Dilokthanakul *et al.*, 2019] Nat Dilokthanakul, Christos Kaplanis, Nick Pawlowski, and Murray Shanahan. Feature control as intrinsic motivation for hierarchical reinforcement learning. *IEEE transactions on neural networks and learning systems*, 2019.

[Eysenbach *et al.*, 2018] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need : Learning skills without a reward function. *CoRR*, abs/1802.06070, 2018.

[Fabisch and Metzen, 2014] Alexander Fabisch and Jan Hendrik Metzen. Active contextual policy search. *The Journal of Machine Learning Research*, 15(1) :3371–3399, 2014.

[Florensa *et al.*, 2017] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[Florensa *et al.*, 2018] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*, pages 1514–1523, 2018.

[Florensa *et al.*, 2019] Carlos Florensa, Jonas Degrave, Nicolas Heess, Jost Tobias Springenberg, and Martin Riedmiller. Self-supervised learning of image embedding for continuous control. *arXiv preprint arXiv :1901.00943*, 2019.

[Fortunato *et al.*, 2017] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv :1706.10295*, 2017.

[Fox *et al.*, 2018] Lior Fox, Leshem Choshen, and Yonatan Loewenstein. DORA the explorer : Directed outreaching reinforcement action-selection. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[François-Lavet *et al.*, 2018] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4) :219–354, 2018.

[Frank *et al.*, 2014] Mikhail Frank, Jürgen Leitner, Marijn Stollenga, Alexander Förster, and Jürgen Schmidhuber. Curiosity driven reinforcement learning for motion planning on humanoids. *Frontiers in neurorobotics*, 7 :25, 2014.

[Fu *et al.*, 2017] Justin Fu, John Co-Reyes, and Sergey Levine. Ex2 : Exploration with exemplar models for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2577–2587, 2017.

[Georgeon *et al.*, 2011] Olivier L Georgeon, James B Marshall, and Pierre-Yves R Ronot. Early-stage vision of composite scenes for spatial learning and navigation. In *2011 IEEE International Conference on Development and Learning (ICDL)*, volume 2, pages 1–6. IEEE, 2011.

[Ghavamzadeh *et al.*, 2015] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. Bayesian reinforcement learning : A survey. *Foundations and Trends® in Machine Learning*, 8(5-6) :359–483, 2015.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[Gopnik *et al.*, 1999] Alison Gopnik, Andrew N Meltzoff, and Patricia K Kuhl. *The scientist in the crib : Minds, brains, and how children learn.* William Morrow & Co, 1999.

[Graves, 2011] Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

[Gregor *et al.*, 2016] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv :1611.07507*, 2016.

[Guckelsberger *et al.*, 2016] Christian Guckelsberger, Christoph Salge, and Simon Colton. Intrinsically motivated general companion npcs via coupled empowerment maximisation. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2016.

[Hafner *et al.*, 2018] Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *CoRR*, abs/1811.04551, 2018.

[Heess *et al.*, 2016] Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv :1610.05182*, 2016.

[Hester and Stone, 2017] Todd Hester and Peter Stone. Intrinsically motivated model learning for developing curious robots. *Artificial Intelligence*, 247 :170–186, 2017.

[Hinton and Salakhutdinov, 2006] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786) :504–507, 2006.

[Houthooft *et al.*, 2016] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime : Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[Hughes *et al.*, 2018] Edward Hughes, Joel Z Leibo, Matthew G Philips, Karl Tuyls, Edgar A Duéñez-Guzmán,

Antonio García Castañeda, Iain Dunning, Tina Zhu, Kevin R McKee, Raphael Koster, et al. Inequity aversion resolves intertemporal social dilemmas. *arXiv preprint arXiv :1803.08884*, 2018.

[Ito, 2016] Sosuke Ito. *Information thermodynamics on causal networks and its application to biochemical signal transduction*. Springer, 2016.

[Itti and Baldi, 2006] Laurent Itti and Pierre F Baldi. Bayesian surprise attracts human attention. In *Advances in neural information processing systems*, pages 547–554, 2006.

[Jaques *et al.*, 2018] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro A Ortega, DJ Strouse, Joel Z Leibo, and Nando de Freitas. Intrinsic social motivation via causal influence in multi-agent rl. *arXiv preprint arXiv :1810.08647*, 2018.

[Karl *et al.*, 2017] Maximilian Karl, Maximilian Soelch, Philip Becker-Ehmck, Djalel Benbouzid, Patrick van der Smagt, and Justin Bayer. Unsupervised real-time control through variational empowerment. *arXiv preprint arXiv :1710.05101*, 2017.

[Kearns and Singh, 2002] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3) :209–232, 2002.

[Kempka *et al.*, 2016] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom : A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2016.

[Kingma and Welling, 2013] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv :1312.6114*, 2013.

[Kirkpatrick *et al.*, 2017] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13) :3521–3526, 2017.

[Klyubin *et al.*, 2005] Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Empowerment : A universal agent-centric measure of control. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 128–135. IEEE, 2005.

[Kulkarni *et al.*, 2016a] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning : Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[Kulkarni *et al.*, 2016b] Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv :1606.02396*, 2016.

[Lakshminarayanan *et al.*, 2016] Aravind S Lakshminarayanan, Ramnandan Krishnamurthy, Peeyush Kumar, and Balaraman Ravindran. Option discovery in hierarchical reinforcement learning using spatio-temporal clustering. *arXiv preprint arXiv :1605.05359*, 2016.

[Lehman and Stanley, 2008] Joel Lehman and Kenneth O Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336, 2008.

[Leibo *et al.*, 2017] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 464–473. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[Lesort *et al.*, 2018] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control : An overview. *Neural Networks*, 2018.

[Levy *et al.*, 2019] Andrew Levy, Robert Platt, and Kate Saenko. Hierarchical reinforcement learning with hindsight. In *International Conference on Learning Representations*, 2019.

[Lillicrap *et al.*, 2015] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv :1509.02971*, 2015.

[Little and Sommer, 2013] Daniel Ying-Jeh Little and Friedrich Tobias Sommer. Learning and exploration in action-perception loops. *Frontiers in neural circuits*, 7 :37, 2013.

[Liu *et al.*, 2015] Chunming Liu, Xin Xu, and Dewen Hu. Multiobjective reinforcement learning : A comprehensive overview. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, 45(3) :385–398, 2015.

[Lopes *et al.*, 2012] Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in Neural Information Processing Systems*, pages 206–214, 2012.

[Machado *et al.*, 2017a] Marios C Machado, Marc G Bellemare, and Michael Bowling. A laplacian framework for option discovery in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2295–2304. JMLR. org, 2017.

[Machado *et al.*, 2017b] Marlos C Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through

the deep successor representation. *arXiv preprint arXiv :1710.11089*, 2017.

[Machado *et al.*, 2018] Marlos C Machado, Marc G Bellemare, and Michael Bowling. Count-based exploration with the successor representation. *arXiv preprint arXiv :1807.11622*, 2018.

[Marsland *et al.*, 2002] Stephen Marsland, Jonathan Shapiro, and Ulrich Nehmzow. A self-organising network that grows when required. *Neural networks*, 15(8-9) :1041–1058, 2002.

[Martin *et al.*, 2017] Jarryd Martin, Suraj Narayanan Sasikumar, Tom Everitt, and Marcus Hutter. Count-based exploration in feature space for reinforcement learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2471–2478, 2017.

[McCloskey and Cohen, 1989] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks : The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[McGovern and Barto, 2001] Amy McGovern and Andrew G Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. 2001.

[Menache *et al.*, 2002] Ishai Menache, Shie Mannor, and Nahum Shimkin. Q-cut—dynamic discovery of sub-goals in reinforcement learning. In *European Conference on Machine Learning*, pages 295–306. Springer, 2002.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540) :529, 2015.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[Moerland *et al.*, 2016] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Fear and hope emerge from anticipation in model-based reinforcement learning. In *IJCAI*, pages 848–854, 2016.

[Moerland *et al.*, 2018] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Emotion in reinforcement learning agents and robots : a survey. *Machine Learning*, 107(2) :443–480, 2018.

[Mohamed and Rezende, 2015] Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 2125–2133, 2015.

[Montúfar *et al.*, 2016] Guido Montúfar, Keyan Ghazi-Zahedi, and Nihat Ay. Information theoretically aided reinforcement learning for embodied agents. *arXiv preprint arXiv :1605.09735*, 2016.

[Nachum *et al.*, 2018] Ofir Nachum, Shixiang (Shane) Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3303–3313. 2018.

[Nachum *et al.*, 2019] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2019.

[Nair *et al.*, 2018] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9209–9220, 2018.

[Ng *et al.*, 1999] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations : Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

[Oh and Cavallaro,] Changjae Oh and Andrea Cavallaro. Learning action representations for self-supervised visual exploration.

[Oh *et al.*, 2015] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*, pages 2863–2871, 2015.

[Ostrovski *et al.*, 2017] Georg Ostrovski, Marc G Bellemare, Aaron van den Oord, and Rémi Munos. Count-based exploration with neural density models. *arXiv preprint arXiv :1703.01310*, 2017.

[Oudeyer and Kaplan, 2008] Pierre-Yves Oudeyer and Frederic Kaplan. How can we define intrinsic motivation ? In *Proceedings of the 8th International Conference on Epigenetic Robotics : Modeling Cognitive Development in Robotic Systems, Lund University Cognitive Studies, Lund : LUCS, Brighton*. Lund University Cognitive Studies, Lund : LUCS, Brighton, 2008.

[Oudeyer and Kaplan, 2009] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation ? a typology of computational approaches. *Frontiers in neurorobotics*, 1 :6, 2009.

[Oudeyer and Smith, 2016] Pierre-Yves Oudeyer and Linda B Smith. How evolution may work through curiosity-driven developmental process. *Topics in Cognitive Science*, 8(2) :492–502, 2016.

[Parisi *et al.*, 2019] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter.

Continual lifelong learning with neural networks : A review. *Neural Networks*, 2019.

[Pathak *et al.*, 2017] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017, 2017.

[Perolat *et al.*, 2017] Julien Perolat, Joel Z Leibo, Vinicius Zambaldi, Charles Beattie, Karl Tuyls, and Thore Graepel. A multi-agent reinforcement learning model of common-pool resource appropriation. In *Advances in Neural Information Processing Systems*, pages 3643–3652, 2017.

[Piaget and Cook, 1952] Jean Piaget and Margaret Cook. *The origins of intelligence in children*, volume 8. International Universities Press New York, 1952.

[Plappert *et al.*, 2017] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv :1706.01905*, 2017.

[Rafati and Noelle, 2019] Jacob Rafati and David C Noelle. Unsupervised methods for subgoal discovery during intrinsic motivation in model-free hierarchical reinforcement learning. 2019.

[Riedmiller *et al.*, 2018] Martin A. Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 4341–4350, 2018.

[Riemer *et al.*, 2018] Matthew Riemer, Miao Liu, and Gerald Tesauro. Learning abstract options. In *Advances in Neural Information Processing Systems*, pages 10445–10455, 2018.

[Rückstiess *et al.*, 2010] Thomas Rückstiess, Frank Sehnke, Tom Schaul, Daan Wierstra, Yi Sun, and Jürgen Schmidhuber. Exploring parameter space in reinforcement learning. *Paladyn, Journal of Behavioral Robotics*, 1(1) :14–24, 2010.

[Ryan and Deci, 2000] Richard M Ryan and Edward L Deci. Intrinsic and extrinsic motivations : Classic definitions and new directions. *Contemporary educational psychology*, 25(1) :54–67, 2000.

[Salge *et al.*, 2014a] Christoph Salge, Cornelius Glackin, and Daniel Polani. Changing the environment based on empowerment as intrinsic motivation. *Entropy*, 16(5) :2789–2819, 2014.

[Salge *et al.*, 2014b] Christoph Salge, Cornelius Glackin, and Daniel Polani. Empowerment–an introduction. In *Guided Self-Organization : Inception*, pages 67–114. Springer, 2014.

[Savinov *et al.*, 2018] Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. *arXiv preprint arXiv :1810.02274*, 2018.

[Schaul *et al.*, 2015] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.

[Schmidhuber, 1991] Jürgen Schmidhuber. Curious model-building control systems. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pages 1458–1463. IEEE, 1991.

[Schmidhuber, 2008] Jürgen Schmidhuber. Driven by compression progress : A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. In *Workshop on Anticipatory Behavior in Adaptive Learning Systems*, pages 48–76. Springer, 2008.

[Schmidhuber, 2010] Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3) :230–247, 2010.

[Schwenker and Palm, 2019] Friedhelm Schwenker and Guenther Palm. Artificial development by reinforcement learning can benefit from multiple motivations. *Frontiers in Robotics and AI*, 6 :6, 2019.

[Sequeira *et al.*, 2011] Pedro Sequeira, Francisco S Melo, Rui Prada, and Ana Paiva. Emerging social awareness : Exploring intrinsic motivation in multiagent learning. In *2011 IEEE International Conference on Development and Learning (ICDL)*, volume 2, pages 1–6. IEEE, 2011.

[Sequeira *et al.*, 2014] Pedro Sequeira, Francisco S Melo, and Ana Paiva. Learning by appraising : an emotion-based approach to intrinsic reward design. *Adaptive Behavior*, 22(5) :330–349, 2014.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676) :354, 2017.

[Singh *et al.*, 2010] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning : An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2) :70–82, 2010.

[Skinner, 1938] B. F. Skinner. The behavior of organisms. In *New York : Appleton*, 1938.

[Stadie *et al.*, 2015] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv :1507.00814*, 2015.

[Stanton and Clune, 2018] Christopher Stanton and Jeff Clune. Deep curiosity search : Intra-life exploration can improve performance on challenging deep reinforcement learning problems. 2018.

[Still and Precup, 2012] Susanne Still and Doina Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3) :139–148, 2012.

[Su *et al.*, 2015] Pei-Hao Su, David Vandyke, Milica Gasic, Nikola Mrksic, Tsung-Hsien Wen, and Steve Young. Reward shaping with recurrent neural networks for speeding up on-line policy learning in spoken dialogue systems. *arXiv preprint arXiv :1508.03391*, 2015.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Reinforcement learning : An introduction*, volume 1. MIT press Cambridge, 1998.

[Sutton *et al.*, 1999] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps : A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2) :181–211, 1999.

[Tang *et al.*, 2017] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration : A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2753–2762, 2017.

[Tishby *et al.*, 2000] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[Todorov *et al.*, 2012] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco : A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[Tomar *et al.*, 2018] Manan Tomar, Rahul Ramesh, and Balaraman Ravindran. Successor options : An option discovery algorithm for reinforcement learning. 2018.

[Van den Oord *et al.*, 2016] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

[Vezhnevets *et al.*, 2017] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3540–3549, 2017.

[Warde-Farley *et al.*, 2018] David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv :1811.11359*, 2018.

[White, 1959] Robert W White. Motivation reconsidered : The concept of competence. *Psychological review*, 66(5) :297, 1959.

[Williams *et al.*, 2015] Henry Williams, Christopher Lee-Johnson, Will N Browne, and Dale A Carnegie. Emotion inspired adaptive robotic path planning. In *2015 IEEE congress on evolutionary computation (CEC)*, pages 3004–3011. IEEE, 2015.

[Wiskott and Sejnowski, 2002] Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis : Unsupervised learning of invariances. *Neural computation*, 14(4) :715–770, 2002.

[Yu *et al.*, 2015] Chao Yu, Minjie Zhang, Fenghui Ren, and Guozhen Tan. Emotional multiagent reinforcement learning in spatial social dilemmas. *IEEE transactions on neural networks and learning systems*, 26(12) :3083–3096, 2015.

[Zhang *et al.*, 2019] Jingwei Zhang, Niklas Wetzel, Nicolai Dorka, Joschka Boedecker, and Wolfram Burgard. Scheduled intrinsic drive : A hierarchical take on intrinsically motivated exploration. *arXiv preprint arXiv :1903.07400*, 2019.

[Zhou *et al.*, 2019] Xiaomao Zhou, Tao Bai, Yanbin Gao, and Yuntao Han. Vision-based robot navigation through combining unsupervised learning and hierarchical reinforcement learning. *Sensors*, 19(7) :1576, 2019.

Construction d'un système d'aide au diagnostic de maladies rares

M. Rémi Besson[1] M. Erwan Le Pennec[1] Mme. Stéphanie Allassonnière[2]

[1] CMAP, Ecole Polytechnique, Route de Saclay, 91128 Palaiseau
[2] Ecole de médecine, Université Paris-Descartes, 15 Rue de l'Ecole de Médecine, 75006 Paris

remi.besson@polytechnique.edu

Résumé

Dans ce travail, nous proposons une méthode pour construire un outil d'aide à la décision pour le diagnostic de maladie rare. Nous cherchons à minimiser le nombre de tests médicaux nécessaires pour atteindre un état où l'incertitude concernant la maladie du patient est inférieure à un seuil prédéterminé. Ce faisant, nous tenons compte de la nécessité dans de nombreuses applications médicales, d'éviter autant que possible, tout diagnostic erroné. Pour résoudre cette tâche d'optimisation, nous étudions plusieurs algorithmes d'apprentissage par renforcement et les rendons opérationnels pour notre problème de très grande dimension : les stratégies apprises se révèlent bien plus performantes que des stratégies gloutonnes classiques. Nous présentons également une façon de combiner les connaissances d'experts, exprimées sous forme de probabilités conditionnelles, avec des données cliniques. Il s'agit d'un aspect crucial car la rareté des données pour les maladies rares empêche toute approche basée uniquement sur des données cliniques. Nous montrons, tant théoriquement qu'empiriquement, que l'estimateur que nous proposons est toujours plus performant que le meilleur des deux modèles (expert ou données) à une constante près.

Mots Clef

Aide au diagnostic, Arbre de décision, Mélange expert/données.

Abstract

In this work, we propose a method to build a decision support tool for the diagnosis of rare diseases. We aim to minimize the number of medical tests necessary to achieve a state where the uncertainty regarding the patient's disease is less than a predetermined threshold. In doing so, we take into account the need in many medical applications, to avoid as much as possible, any misdiagnosis. To solve this optimization task, we investigate several reinforcement learning algorithm and make them operable in our high-dimensional : the strategies learned are much more effective than classic greedy strategies. We also present a way to combine expert knowledge, expressed as conditional probabilities, with clinical data. This is crucial because the scarcity of data in the field of rare diseases prevents any approach based solely on clinical data. We show, both empirically and theoretically, that our proposed estimator is always more efficient than the best of the two models (expert or data) within a constant.

Keywords

Symptom Checker, Decision Tree, expert/data mixture.

1 Introduction

1.1 Contexte et objectifs

Pendant la grossesse, plusieurs échographies fœtales sont effectuées pour évaluer l'anatomie, la croissance et le bien-être du fœtus. Lors de chaque examen échographique, le médecin effectue un certains nombre de mesures et d'observations standardisées. Néanmoins, en cas d'anomalie, éventuellement liée à un trouble génétique, il n'y a pas de consensus sur la façon de procéder à l'échographie pour établir le diagnostic. C'est un problème puisqu'il existe beaucoup de symptômes (220 dans notre cas) qui peuvent être difficiles à détecter et peuvent mener à un grand nombre de diagnostics possibles alors que les médecins n'ont pas le temps ni la connaissance encyclopédique pour les envisager tous.

Dans ce travail, nous cherchons à systématiser la procédure de diagnostic prénatal afin d'aider le praticien à poser un diagnostic, fiable avec grande probabilité, tout en minimisant le nombre moyen de questions, c'est-à-dire les symptômes à vérifier.

Pour cela, nous cherchons un algorithme qui propose les symptômes les plus intéressant à rechercher à chaque étape de l'examen médical (état des connaissances sur l'état du patient) tout en affichant de manière dynamique la probabilité de chaque maladie possible. Ce type d'algorithme est parfois appelé "symptom checker" dans la littérature. Enfin, notre algorithme doit être utilisable et interprétable en direct, au chevet du patient.

Il est à noter que notre approche s'applique à tout problème visant à établir un symptom checker pour les maladies rares et ne se limite pas, bien entendu, à notre cas d'étude auquel nous ferons référence tout au long de ce papier : le diagnostic prénatal.

1.2 Données disponibles

Nous disposons d'une liste de maladies et pour chaque maladie d'une liste de symptôme généralement associés (que nous appellerons symptômes typiques). Nous écrivons :

$$B_i = \begin{cases} 1 & \text{si le patient a le symptôme de type i} \\ 0 & \text{sinon .} \end{cases}$$

Nous notons D la variable aléatoire associée à la maladie du patient : $D \in \{d_1, ..., d_k\}$. Nous faisons l'hypothèse que le patient ne présente qu'une maladie à la fois ce qui est une hypothèse raisonnable pour les maladies rares. $\{D = d_k\}$ représente l'évènement pour lequel le patient est sain ou est atteint d'une maladie qui n'est pas dans la base de données.

Données expertes. Nous disposons d'une estimation de la probabilité de chaque maladie, ainsi que de la probabilité de présenter chaque symptôme typique sachant la maladie. Nous connaissons donc $\mathbb{P}[D = d_j]$ abrégé en $\mathbb{P}[D_j]$ et également $\mathbb{P}[B_i = 1 \mid D = d_j]$ abrégé en $\mathbb{P}[B_i \mid D_j]$ lorsque B_i est un symptôme typique de D_j. Ajoutons qu'il est possible d'avoir une maladie et de présenter un symptôme atypique mais nous n'avons pas d'estimation pour de tels évènements. Nous supposerons que les symptômes atypiques se manifestent avec une faible probabilité (fixée à 10^{-5}) et indépendamment des autres symptômes (conditionnellement à la maladie).
Notez que nous ne disposons pas des lois jointes des symptômes sachant la maladie mais seulement des marginales. Nous abordons cette difficulté dans la section 3.
Toutes ces informations, que nous désignons dans la suite sous le terme de données expertes, nous ont été fournies par des médecins de l'hôpital Necker en se basant sur la littérature disponible. Nous avons mis les symptômes de la base obtenue en correspondance avec ceux de la base Human Phenotype Ontology (HPO) [20]. HPO est un travail récent qui fournit une terminologie standardisé des anomalies phénotypiques retrouvées dans les pathologies humaines. Nous nous en sommes servis pour harmoniser la terminologie. Nous avons ensuite mis cette liste de symptômes par maladie en correspondance avec OrphaData [1] qui est une base de données de référence pour les maladies rares (et n'est pas limitée à l'obstétrique). OrphaData a été utile pour imputer les valeurs manquantes des prévalences des symptômes dans les maladies. Nous restreignons nos analyses au sous-ensemble des symptômes pouvant être détectés en utilisant l'échographie fœtale.
Actuellement, notre base de données référence 81 maladies et 220 symptômes différents. La maladie avec le plus grand nombre de symptômes typiques associés est le syndrome de VACTERL avec 19 symptômes associés.

Données cliniques. Comme dans de nombreuses applications médicales l'accès à des données cliniques est difficile. Notre outil d'aide à la décision a cependant vocation à collecter des données au fur et à mesure de son utilisation. Ces données devront nous permettre de mieux inférer la loi de notre environement initialisé par les données expertes.
De manière générale nous espérons que ce travail constituera une opportunité de bâtir une base de données globalisée permettant de mieux connaître les maladies rares diagnosticables par échographie foetale.

1. Orphanet. INSERM 1997. Une base de données pour les maladies rares et orphelines. Disponible sur http ://www.orpha.net. Consulté le [02/10/2018].

1.3 Principales contributions

Nous distinguons deux principales contributions relativement indépendantes l'une de l'autre.
Tout d'abord dans la section 2 nous formulons de manière originale le problème d'optimisation associé à la recherche d'un bon symptom checker. Nous proposons une manière de réduire notre problème global de très haute dimension et donc intractable à une somme de sous-tâches de plus petites dimensions résolubles par des algorithmes de planification avec approximation (de type DQN [28]). Nous montrons que l'usage adéquat de sous-tâches déjà résolues permet d'augmenter sensiblement la vitesse d'apprentissage de sous-tâches de plus grande dimension en tirant partie de leurs intersections. Enfin nous montrons que la stratégie apprise surpasse largement les performances d'algorithmes gloutons classiques de construction d'arbre de décision (que nous appellerons dans la suite algorithme de Breiman [25, 34]).
Par ailleurs, dans la section 3 nous détaillons une méthode pour mélanger efficacement données expertes et données cliniques afin de faire face à une difficulté fréquemment rencontrée en médecine (et d'autant plus lorsqu'il s'agit de maladies rares) : le faible nombre de données cliniques. Nous montrons, tant théoriquement qu'empiriquement, que l'estimateur que nous proposons est toujours plus performant que le meilleur des deux modèles (expert ou données) à une constante près
Tous nos codes informatiques ont été écrit dans le langage R. Afin d'assurer la reproductibilité de nos expériences numériques, nous avons mis ce code en libre accès sur GitHub [33].

2 Le problème de planification

2.1 Formulation du problème d'optimisation

Ce que nous cherchons à optimiser. Notre problème de prise de décision séquentielle peut être formulé sous la forme d'un processus décisionnel de Markov (PDM). Soit $\mathbb{S}$ l'espace d'état, en utilisant la base ternaire nous codons 1 si le symptôme considéré est présent, 0 s'il est absent, 2 si non encore observé. Nous écrivons $\mathbb{S} = \big\{(2, \ldots, 2), (1, 2, \ldots, 2), \ldots, (0, \ldots, 0)\big\}$. Un élément $s \in \mathbb{S}$ est un vecteur de longueur 220 (le nombre de symptômes possibles), il résume notre état de connaissance sur l'état du patient : le i-ème trigit de s code l'information sur le symptôme dont l'identifiant est i. Soit $\mathbb{A}$ l'état des actions : $\mathbb{A} = \big\{a_1, ..., a_{220}\big\}$. Une action est un symptôme que nous suggérons à l'obstétricien de rechercher.

Notez qu'ainsi définie la dynamique de notre environnement est clairement Markovienne en ce sens que : $\mathbb{P}[s_{t+1} \mid a_t, s_t, a_{t-1}, s_{t-1}, ...a_0, s_0] = \mathbb{P}[s_{t+1} \mid a_t, s_t]$ où s_t (respectivement a_t) est l'état (resp. l'action) visité (resp. prise) au temps t.
Nous cherchons à apprendre une stratégie de diagnostic qui associe à chaque état de connaissance (liste de présence/absence de symptômes) une action à prendre (un symptôme à rechercher) :

$$\pi : \mathbb{S} \to \mathbb{A}. \tag{1}$$

Que devrait être une bonne stratégie de diagnostic ? La plupart des travaux visant une application médicale proposent d'optimiser un compromis entre le coût des tests médicaux à réaliser (que cela soit mesuré en temps, en argent ou en prise de risque pour le patient) et le coût d'un éventuel mauvais diagnostic [6], [7], [8].
Dans notre cas, le coût des tests médicaux (c'est-à-dire aller observer un symptôme potentiel supplémentaire) est négligeable devant le coût potentiel d'une erreur de diagnostic. En théorie les obstétriciens doivent vérifier tous les symptômes possibles afin de s'assurer que le fœtus ne présente aucune maladie.
Par conséquent, nous ne prendrons pas le risque de proposer un diagnostic erroné dans le but de poser moins de questions. Cependant, si le médecin observe un nombre suffisant de symptômes, il peut arrêter l'examen échographique et prescrire des examens supplémentaires, comme une amniocentèse par exemple, pour confirmer ses hypothèses.
C'est pourquoi nous pouvons définir certains états comme terminaux : ce sont ceux où l'entropie de la variable aléatoire maladie D est tellement faible que nous n'avons aucun doute sur le diagnostic. Dans ce contexte, notre objectif est de minimiser le nombre moyen de question à poser avant d'atteindre un état final :

$$\pi^* = \arg\min_{\pi} \; E_{\mathcal{P}}\big[I \big| s_0, \pi\big], \tag{2}$$

où $s_0 = (2, ..., 2)$ est l'état initial, $\mathcal{P}$ la loi de l'environnement actuellement utilisée, π la stratégie de diagnostic, et I le nombre aléatoire de questions à poser avant d'atteindre un état final, i.e :

$$I = \inf\{t \in \mathbb{N}^\star \mid H(D \mid S_t) \leq \epsilon\}$$

où

$$\begin{aligned} H(D \mid S_t) &= \sum_{s_t} \mathbb{P}[S_t = s_t] H(D \mid S_t = s_t) \\ &= -\sum_{s_t} \mathbb{P}[S_t = s_t] \sum_{d} \mathbb{P}[D = d \mid s_t] \log \mathbb{P}[D = d \mid s_t] \end{aligned}$$

est l'entropie de la variable aléatoire maladie D sachant toutes les informations disponibles sur les symptômes du patient à l'instant t : S_t.

s_t est une réalisation de S_t, ce n'est rien de plus que l'état atteint pour un examen sur un patient donné alors que S_t est la variable aléatoire associée. Pour un état initial donné s_0 et une stratégie π de nombreux états différents peuvent être atteints puisque les réponses sont stochastiques.
Notez que nous ne sommes pas assurés que pour tout t nous ayons $H(D \mid s_{t+1}) \leq H(D \mid s_t)$. Cependant cette inégalité est vraie lorsque l'on prend la moyenne $H(D \mid S_{t+1}) \leq H(D \mid S_t)$, voir le théorème 2.6.5 de [3], "information can't hurt". Pour le dire autrement, lorsque nous considérons que l'entropie est suffisamment faible pour arrêter l'examen et proposer un diagnostic, nous savons qu'en moyenne, l'incertitude relative à la maladie du patient n'aurait pas augmenté si nous avions continué à vérifier l'absence/présence d'autres symptômes.
En définissant une fonction de récompense comme suit : $r_t := r(s_t, a_t) = -1$, $\forall s_t, a_t$ nous pouvons réécrire (2) sous la forme classique d'un problème d'apprentissage par renforcement épisodique [21] :

$$\pi^* = \arg\max_{\pi} \; E_{\mathcal{P}}\left[\sum_{t=0}^{I} r_t \Big| s_0, \pi\right]. \tag{3}$$

Dans la communauté de l'apprentissage par renforcement, une telle manière de modéliser le signal de récompense est appelée "action-penalty representation", puisque l'agent est pénalisé pour chaque action qu'il exécute [24]. Le problème d'optimisation (3) est un problème de planification puisque nous supposons connaître la loi de notre environnement $\mathcal{P}$ (voir la figure 10) et visons à résoudre le PDM associé. Néanmoins, dans la pratique, comme la dimension de notre problème est élevée, nous résolvons le PDM en échantillonnant des parties à partir d'un simulateur de notre environnement construit dans la section 3 .

Travaux apparentés. De nombreux systèmes experts ont été développés pour le diagnostic des maladies rares (en particulier en obstétrique) comme par exemple Orphamizer voir [32] et [20]. La plupart de ces systèmes experts prennent en entrée une liste de symptômes observés et fournissent en sortie une liste correspondante de maladies possibles. Néanmoins, nous pensons qu'un algorithme utilisable pendant l'examen médical serait plus utile qu'un système expert conçu pour une utilisation rétrospective. C'est pourquoi nous visons à proposer à chaque étape le symptôme le plus intéressant à vérifier.
Très peu de travaux récents s'attaquent à cette question. Nous avons référencé [8] qui propose un algorithme de type A* afin de trouver le plus court chemin dans le graphe mais ce genre d'algorithme ne peut pas faire face aux problèmes de grande dimension.
Notez qu'en un certain sens, notre problème peut être assimilé à celui de l'optimisation d'un arbre de décision où les variables explicatives sont les symptômes et la maladie est la variable objectif. En effet, une stratégie associée à un PDM est une généralisation d'un arbre, une stratégie étant moins rigide en ce sens qu'elle peut encore proposer

la prochaine action à prendre même lorsque le médecin a fait un choix différent de celui que nous avions proposé. Les algorithmes classiques d'arbre de décision, voir [25] ou [34], reposent sur l'optimisation d'une fonction d'impureté (l'entropie ou l'indice de Gini de la variable aléatoire cible) de manière gloutonne et sont donc soumis à l'effet de l'horizon [30] bien connu dans les jeux : une action peut paraître intéressante dans l'immédiat mais se révéler très mauvaise à un horizon plus lointain. Ainsi, les travaux récents cherchant une procédure d'optimisation globale des arbres de décision tels que [29] peuvent être considérés comme pertinents. Cependant, encore une fois, ces algorithmes utilisant des solveurs MIO (Mixed Integer Optimization) ne peuvent pas faire face à notre problème de haute dimension. En effet, la complexité de ces algorithmes est en $n \times 2^D$ où n est le nombre de données et D la profondeur maximale de l'arbre. Or, dans notre cas, nous ne pouvons pas limiter aussi facilement la profondeur maximale autorisée de l'arbre car dans le pire des cas, le médecin n'observera aucun symptôme et devra alors les vérifier tous.

Des travaux plus récents tels que [6], [7] et [31] s'intéressent eux aussi à l'optimisation de stratégie de diagnostic en utilisant des algorithmes d'apprentissage par renforcement. Néanmoins, notre approche est significativement différente de ces travaux précédents, tant dans notre façon de formuler l'objectif (et donc dans notre manière de définir la récompense) que dans les solutions que nous proposons (nos façons de casser la dimension).

Dans tous ces papiers [6, 7, 31] le problème d'optimisation est formulé comme un compromis entre poser moins de questions et faire le bon diagnostic, tandis que nous le formulons comme la tâche d'atteindre le plus rapidement possible, en moyenne, un degré élevé prédéterminé de certitude concernant la maladie du patient. En pratique, dans notre cas, le seul paramètre ϵ à régler est le degré de certitude que nous désirons avoir à la fin de l'examen : nous devons nous arrêter lorsque l'entropie de la maladie tombe sous ce seuil. Plus ϵ est petit, plus notre algorithme aura besoin de vérifier de symptômes avant de considérer que l'examen est terminé.

[6] fait usage d'un terme de discount $\gamma \in [0, 1]$ dans la modélisation du signal de récompense. Ils définissent la récompense associée à chaque question comme étant nulle jusqu'à la formulation d'un diagnostic (qui est une action supplémentaire possible) où la récompense est égale à γ^q (si la proposition était correcte, 0 dans le cas contraire), q étant le nombre de questions qui ont été posées avant de proposer le diagnostic. Dans ce contexte, γ fait le compromis entre poser moins de questions et faire le bon diagnostic. Plus γ est petit, plus l'algorithme est susceptible de faire un mauvais diagnostic en essayant de poser moins de questions.

Notez que [6] doit exécuter son algorithme d'apprentissage en essayant plusieurs valeurs différentes de γ. A l'inverse, nous pouvons déterminer la valeur de ϵ à prendre avant d'exécuter tout algorithme d'apprentissage. Nous pouvons en effet, dans un premier temps, interagir avec le médecin, en lui présentant un échantillon d'états où notre algorithme s'arrêterait et formulerait un diagnostic. Si le médecin considère que l'algorithme s'arrête trop tôt, nous devrons diminuer ϵ, sinon nous devrions augmenter ϵ. Il s'agit d'un avantage non négligeable puisque le principal goulet d'étranglement en terme de temps de calcul est la phase d'apprentissage.

Un problème de grande dimension. Notre problème (3) est de très haute dimension puisqu'il y a 220 symptômes et donc théoriquement 3^{220} états possibles. Une approche tabulaire classique de type Q-learning est donc impossible. D'après nos expériences, un algorithme d'apprentissage classique de type Deep-Q-learning est lui aussi numériquement intractable. Afin de casser la dimension, nous tirons d'abord parti du fait que les médecins utilisent notre algorithme principalement après avoir vu un premier symptôme. Dans ce cas, nous supposons que ce symptôme initial est typique. Il peut être possible en effet d'avoir une maladie qui présente également un symptôme atypique mais cela se produit avec une très faible probabilité, suffisamment négligeable pour les cliniciens. Quoi qu'il en soit, dans ce cas, nous nous retrouverions avec une entropie élevée et aucune identification de la maladie. Cela nous conduirait à passer à une autre stratégie. Avec une telle hypothèse, la dimension diminue considérablement puisque nous ne considérons plus que les maladies pour lesquelles ce symptôme initial est typique, les seuls symptômes pertinents étant ceux qui sont typiques de ces autres maladies. Nous avons donc créé 220 tâches $\mathcal{T}_i$ à résoudre, $\forall i$, $s_{(i)} = (2, ..., 2, 1, 2, ..., 2)$:

$$\pi^*_{(i)} = \arg\max_{\pi} E_{\mathcal{P}} \left[\sum_{t=0}^{I} r_t \mid s_{(i)}, \pi \right]. \qquad (\mathcal{T}_i)$$

Les dimensions des différents sous-problèmes sont présentées sur la figure 3. Décomposer à ce point notre problème en sous-tâches représente également l'avantage de nous donner une stratégie très performante sur plusieurs parties de notre arbre décisionnel qui auraient été sinon sous-optimisées (car ces parties de l'arbre ne sont pas souvent visitées). Bien sûr, l'optimisation des parties de l'arbre qui ne sont pas souvent visitées n'est pas très utile pour réduire notre fonction de perte globale, mais il est important de fournir, dans tous les cas, une proposition raisonnable au médecin si nous voulons qu'il ait confiance en nous. Cette approche nous obligera à choisir un algorithme d'apprentissage capable de résoudre différents sous-problèmes sans avoir à régler trop d'hyper-paramètres.

Pour faire face aux problèmes de dimensions, Tang et al. [6] ont proposé dans leur premier papier d'apprendre une stratégie différente pour chaque région anatomique préalablement construite (qui sont au nombre de 11). Comme ils l'ont reconnu dans leur second papier [7] cette approche est problématique. En effet un symptôme peut être associé à plusieurs régions anatomiques différentes. Comment

choisir le modèle à utiliser lorsque l'on reçoit l'information du symptôme initial observé ? Dans leur premier papier, lorsqu'un patient donne son symptôme d'appel, ils choisissent le modèle le plus performant sur leur jeu d'entraînement et suivent cette stratégie jusqu'à la fin du processus. Cependant, comme explicité dans [7], il est possible que la maladie du patient n'appartiennent pas à l'ensemble des maladies de la région anatomique choisie. C'est pourquoi ils proposent dans ce nouveau papier [7] d'apprendre une autre stratégie, appelée "master model", qui choisit à chaque étape le modèle (parmi les 11 disponibles) à utiliser.

En ce qui concerne leur papier le plus récent [31], l'idée principale est d'utiliser les techniques de reward-shaping [47] pour faire face au caractère fortement sporadique des récompenses. La plupart des actions ne donnent, en effet, pas de récompense immédiate significative dans notre problème et il faut attendre la fin de l'examen pour obtenir enfin un signal positif ou négatif. Dans [31] la découverte de la présence d'un nouveau symptôme donne droit à un bonus, appelé récompense auxiliaire. L'idée est de guider la stratégie vers des objectifs auxiliaires, considérés comme désirables, ici le fait d'observer la présence d'un symptôme. Il est cependant bien connu que le reward-shaping est équivalent à une initialisation différente des Q-valeurs, voir [38]. Ainsi initialiser nos algorithmes avec des stratégies raisonnables (par exemple celle qui minimise l'entropie de manière gloutonne [34]) à la place d'une stratégie aléatoire donnerait les mêmes avantages que ceux qui sont observés dans [31].

2.2 Deux approches différentes pour résoudre notre problème.

En apprentissage par renforcement, il existe plusieurs manières de résoudre un problème comme ($\mathcal{T}_i$). Si la dimension est suffisamment petite, il est possible de trouver la solution optimale explicitement en utilisant un algorithme de programmation dynamique [21], par exemple en utilisant l'algorithme de "value iteration". Si le nombre d'états est trop élevé, nous devons paramétrer la stratégie (approche centrée sur la stratégie : "policy-based") ou paramétrer les Q-valeurs ("value-based"). Nous avons exploré ces deux approches pour résoudre notre problème.

Une approche centrée sur la stratégie pour faire office de référence. Dans notre application, il peut être intéressant de proposer à l'utilisateur plusieurs symptômes à rechercher, chacun avec son score correspondant (l'intérêt à le vérifier), au lieu d'un seul. En effet, les médecins pourraient être réticents à utiliser un outil d'aide à la décision qui ne leur laisse pas une part de liberté dans leur choix. C'est pourquoi nous considérons une formulation energy-based, un choix populaire comme dans [27] :

$$\pi_\theta(s,a) = e^{\theta^T \phi(s,a)} \Big/ \sum_b e^{\theta^T \phi(s,b)}$$

où $\pi_\theta(s,a)$ désigne la probabilité de prendre l'action a en l'état s, $\phi(s,a)$ est un vecteur résumant un certains nombre de mesures liées à l'intérêt de prendre l'action a lorsque nous sommes en l'état s. Plus précisèment : $\phi(s,a) = \big(H(D \mid s) - E[H(D \mid s,a)], \mathbb{P}[S_a \mid s], \mathbb{1}_{A_a \in S_{max}(s)}\big)$ où $S_{max}(s)$ is est l'ensemble des symptômes typiques de la maladie la plus probable à l'état s et $H(D \mid s)$ l'entropie de la variable aléatoire maladie en l'état s. Ainsi $\phi(s,a)$ regroupe trois manières raisonnables de jouer à notre jeu :

- Poser la question qui minimise l'entropie moyenne de la variable aléatoire maladie. Il s'agit de la manière classique de construire les arbres de décision [25, 34].
- Poser la question dont la probabilité de réponse positive est maximale. Ceci est spécifique à notre problème où une réponse positive est beaucoup plus informative qu'une réponse négative (cela ne serait pas le cas dans un jeu classique de type "20 questions").
- Poser une question sur un symptôme typique de la maladie actuellement la plus plausible.

Ces caractéristiques représentent différentes façons de penser et dilemmes rencontrés lors d'un examen médical : quand j'observe un symptôme, dois-je penser aux symptômes habituellement observés conjointement ou dois-je penser à la maladie la plus plausible et rechercher les symptômes correspondants ?

Notez que cette fonction paramétrée π_θ n'est rien de plus qu'un réseau de neurones sans couche cachée conçue avec des fonctionnalités faites main. Lorsqu'elle est correctement optimisée, cette politique surpasse, par construction, l'algorithme classique d'optimisation d'arbre de décision [25].

Notre objectif est d'apprendre les paramètres optimaux θ pour chacun des 220 sous-problèmes : $\theta^*_{(i)} = \arg\min_\theta L_i(\theta) := E_{\pi_\theta}[I \mid s_{(i)}]$. Ce type de problème d'optimisation, a été intensément étudié par la communauté de l'apprentissage par renforcement, voir [9] ou [10] pour une analyse générale et [27] pour le cas particulier d'une stratégie energy-based. Nous avons entraîné notre stratégie en utilisant un algorithme de type REINFORCE [39]. Comme nous avons cassé la dimension et que le nombre de paramètres à apprendre est limité, cet algorithme est parfaitement adapté et présente des performances similaires à celles d'un algorithme Acteur-Critique.

Une approche centrée sur les Q-valeurs.

Entraîner des Q-réseaux

Rappelons que les Q-valeurs sont définies par : $Q_\pi(s,a) = E\big[\sum_{t'=t}^{I} r_{t'} \mid s_t = s, a_t = a, \pi\big]$, il s'agit de la quantité moyenne de récompense obtenue lorsque l'on part de l'état s, que l'on prend l'action a et que l'on suit la stratégie π. Les Q-valeurs optimales, sont définies comme $Q^\star(s,a) = \max_\pi Q_\pi(s,a)$ et satisfont l'équation de Bellman : $Q^\star(s,a) = E_{s' \sim \mathcal{P}}[r + \max_{a'} Q^\star(s',a')]$.

La stratégie optimale $\pi^\star$, est directement obtenue à partir de $Q^\star$: $\pi^\star(s) = \arg\max_a Q^\star(s,a)$. Nous avons donc "seulement" besoin d'évaluer $Q^\star(s,a)$, $\forall s, a$. Cela peut-être fait par un algorithme de value-iteration qui fait un usage itératif de l'équation de Bellman : $Q_{i+1}(s,a) = E[r + \max_{a'} Q_i(s',a')|s,a]$. Il peut-être prouvé, voir [21], que $Q_i \to Q^\star$ lorsque $i \to \infty$.
Comme la dimension du problème est trop élevée pour stocker/évaluer toutes les Q-valeurs, nous les paramétrons par un réseau de neurones : $Q(s,a) \approx Q_w(s,a)$.
Le célèbre algorithme Deep Q-Network (DQN [28]) a rendu possible l'utilisation des réseaux de neurones pour paramétrer les Q-valeurs (appelé Q-réseau) pour l'algorithme de "value iteration" avec approximation. Le Q-réseau, à l'itération i, est entraîné pour minimiser la fonction de perte $L_i(w_i) = E_{s,a}\left[\left(y_i - Q_{w_i}(s,a)\right)^2\right]$ où $y_i = E_{s' \sim \mathcal{P}}[r + \max_{a'} Q_{w_{i-1}}(s',a')|s,a]$ est la cible. Cette dernière opération peut être réalisée par un algorithme standard de rétropropagation du gradient. En pratique, pour combiner l'apprentissage profond et l'apprentissage par renforcement, l'idée principale est de stocker les résultats des interactions avec l'environnement (les transitions s, a, r, s') dans ce qui sera appelé l'experience-replay. Il s'agit ensuite d'échantilloner des transitions de ce replay-memory pour faire les pas de gradient à l'image de technique de mini-batch. Cette procédure doit permettre de casser les corrélations entre les données. Une autre astuce est de "geler" le réseau cible pendant un certains nombre d'itérations afin de réduire l'instabilité de l'apprentissage.
En procédant ainsi la prise d'action est dissociée de l'apprentissage, la stratégie utilisée pour jouer (appelée stratégie comportementale : "behavior policy") est différente de celle formée à partir des transitions échantillonnées (la stratégie cible : "target policy"). En RL, ces algorithmes sont alors dit "off-policy". Il s'agit d'une propriété désirable pour un algorithme d'apprentissage par renforcement car il est ainsi permis d'explorer davantage ce qui est nécessaire pour éviter de rester bloquer dans des extremas locaux.
La figure 1 montre le schéma général simplifié des algorithmes d'apprentissage par renforcement utilisant des réseaux de neurones : un agent interagit avec son environnement et collecte des données (des transitions s_t, a_t, r_t, s_{t+1}) qui sont incorporés au replay memory dans lequel nous échantillonnons pour former la stratégie cible. La stratégie comportementale est mise à jour de manière périodique en la remplaçant par la stratégie apprise. Dans notre cas, nous effectuons cette mise à jour de la stratégie comportementale dès que nous avons fait un pas de montée de gradient.

Quelques remarques sur la politique comportementale

Notre politique comportementale est une version ϵ-gloutonne de la politique couramment apprise afin de forcer l'algorithme à explorer. Nous utilisons cette stratégie pour simuler des parties, c'est-à-dire les transitions (s_t, a_t, s_{t+1}, r_t).

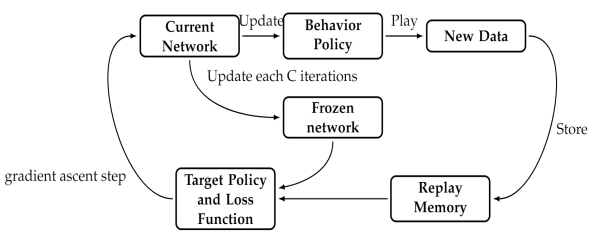


FIGURE 1 – Schéma général des algorithmes d'apprentissage par renforcement utilisant des réseaux de neurones.

Pour cela, nous avons besoin d'un modèle de l'environnement, un modèle de transition qui nous donne la probabilité d'atteindre un état s_{t+1} lorsque l'on prend l'action a_t en l'état s_t. Notre modèle de l'environnement est composé des distributions des combinaisons de symptômes typiques pour chaque maladie (voir la section 3). Nous stockons $(\mathbb{P}[B_1, ..., B_{K_D} \mid D])_{B_1,...,B_{K_D}}$ les probabilités de toutes les combinaisons possibles de symptômes typiques sachant la maladie, K_D désigne le nombre de symptôme typique de la maladie D. Nous ajoutons l'hypothèse qu'un patient peut également présenter un symptôme non typique mais avec faible probabilité et indépendamment des autres symptômes.
Ainsi pour simuler des transitions nous avons besoin de déterminer pour chaque maladie et pour chaque symptôme de la liste renseigné dans l'état courant ceux qui sont typiques de ceux qui ne le sont pas. Cela nous permet de trouver la bonne combinaison à extraire de $(\mathbb{P}[B_1, ..., B_{K_D} \mid D])_{B_1,...,B_{K_D}}$. Ce calcul peut se révéler coûteux. Une manière d'accélérer ces simulations est de démarrer des parties de l'état initial s_0 jusqu'à un état terminal s_I : nous mémorisons de quelles maladies les symptômes déjà observé sont typiques et nous n'avons ainsi plus qu'à chaque itération à déterminer si le dernier symptôme obtenu est typique pour chaque maladie.
Notez que, à chaque étape de notre jeu, nous devons calculer la probabilité de chaque combinaison de symptôme pour chaque maladie, puis d'utiliser la formule Bayes, afin de déterminer si nous devons nous arrêter ou non. Une autre observation importante est qu'il est tout aussi coûteux de calculer directement $p(s_I \mid s_0)$ que de jouer de manière incrémentale (comme décrit précédemment) la totalité de la partie de l'état s_0 jusqu'à s_I.
Ces deux observations devraient suffire à convaincre le lecteur qu'une approche d'apprentissage asynchrone, comme dans [41], ne serait pas appropriée pour notre problème. D'un point de vue computationnel, il est raisonnable de jouer des parties du début à la fin.

La mise à jour de la cible : algorithmes de Temporal-difference et de Monte-Carlo

Une question demeure concernant la manière de mettre à jour la cible, devons-nous utiliser un algorithme de type Monte Carlo ou faire du bootstrap en utilisant les estimées

courantes des Q-valeurs ?
Nous rappelons (en suivant la définition donnée dans [21]) qu'un algorithme est une méthode de bootstrapping s'il base sa mise à jour en partie sur une estimée courante. Ceci est le cas de l'algorithme de Temporal-Difference (TD) défini comme suit :

$$Q_{k+1}(s,a) \leftarrow (1-\alpha) \underbrace{Q_k(s,a)}_{\text{ancien estimé}} + \alpha \underbrace{\left(r(s,a,s') + \max_{a'} Q_k(s',a')\right)}_{\text{mise à jour}}$$

où nous avons échantillonné s, a, s' en utilisant la stratégie comportementale et le modèle de l'environnement $\mathcal{P}$. Q_k est l'estimé à l'itération k, α le learning rate. A l'inverse une méthode de Monte-Carlo ne fait pas usage de bootstrap :

$$Q_{k+1}(s,a) \leftarrow (1-\alpha)Q_k(s,a) + \alpha G$$

où G est la récompense obtenue après avoir simulé une partie.
Le choix entre une méthode de TD et de MC pour calculer la cible y_i n'est pas clair au premier abord. Cette question est le sujet d'un travail récent [35] qui montre que les approches MC peuvent être une alternative viable au TD dans l'ère moderne de l'apprentissage par renforcement. Habituellement, les méthodes TD sont vues comme de meilleurs alternatives face au Monte-Carlo qui est souvent mis de côté à cause de la grande variance des récompenses.
Cependant, notre contexte est particulier : nous cherchons à résoudre une tâche à horizon fini avec une récompense finale : le signal de récompense n'est pas très informatif avant d'atteindre un état terminal. De plus, pour les sous-problèmes de dimension intermédiaires, nous sommes assurés que les parties ne dureront pas trop longtemps et qu'il y a donc une variance relativement faible des récompenses pour les épisodes Monte-Carlo.
Nous avons implémenté les deux solutions, appelées respectivement DQN-TD et DQN-MC. A chaque étape de DQN-MC, nous échantillonnons, en suivant la politique comportementale, 100 parties commençant par l'état initial $s_{(i)}$ and terminant lorsqu'un état terminal est atteint. Toutes les transitions s, a, s' de toutes ces parties se voient associées leur récompense (le nombre de questions qu'il a été nécessaire de poser pour atteindre un état terminal durant la partie concernée) et sont incorporées au Replay-Memory. Nous échantillonnons ensuite des transitions de ce replay memory (un vingtième) et réalisons un pas de montée de gradient avec un algorithme de rétropropagation du gradient (nous avons utilisé pour ce dernier point la librairie Keras [26]).
Concernant DQN-TD, nous avons gardé toutes les caractéristiques principales de DQN-MC afin de faciliter leur comparaison. Nous jouons 100 parties, toujours avec la stratégie comportementale, et toutes les transitions s, a, s' de toutes ces parties reçoivent une récompense de -1 lorsque s' n'est pas terminal et 0 sinon. Le learning rate est initialisé avec une valeur plus faible que pour l'algorithme DQN-MC mais il est diminué à la même vitesse dans les deux cas : on le divise par deux toutes les 300 itérations. Une autre différence est le frozen network que nous utilisons comme cible pour DQN-TD et qui n'est pas nécessaire dans DQN-MC. Nous mettons à jour ce frozen network toutes les 2 itérations (nous avons également essayé de le mettre à jour moins fréquemment mais nous n'avons pas observé de différence significative avec les résultats présentés ici).

Nous comparons ces deux algorithmes, DQN-MC and DQN-TD, sur plusieurs sous-tâches (voir les figures 4 et 5). Nous n'avons pas observé de grande différence sur les sous-tâches de petite et moyenne dimension : les deux algorithmes convergent à la même vitesse vers des solutions de la même qualité. Cependant DQN-TD apparaît beaucoup plus sensible au learning rate. En effet comme cela peut être vu sur la figure 4, DQN-TD converge sur cette tâche où il reste 29 symptômes suspects et 8 maladies possibles, lorsque le learning rate est initialisé à 0.0001. Cependant si le learning rate est choisi un peu plus grand, 0.001, DQN-TD diverge. Au contraire, DQN-MC converge lorsque le learning rate est initialisé à 0.001 mais également lorsqu'il est initialisé à 0.01 même si les variations de l'algorithme sont moins stable dans ce dernier cas. Ces observations doivent être combinées avec celles de la figure 5 où il reste 104 symptômes suspects et 18 maladies possibles. Nous pouvons voir que DQN-TD avec un learning rate initial de 0.0001 diverge. Diminuer le learning rate à 0.00001 ne change pas cette observation. A l'inverse nous n'avons pas besoin de réduire le learning rate initial de DQN-MC (nous le prenons égale à 0.001) afin de le faire converger vers une bonne solution. Comme nous devons entraîner autant de réseaux de neurones qu'il y a de sous-tâche, nous avons besoin d'un algorithme robuste capable de résoudre des tâche de difficulté variable sans avoir à changer à chaque fois tous les hyper-paramètres.

C'est pourquoi nous avons choisi d'utiliser DQN-MC plutôt que DQN-TD. Il est, en effet, bien connu que combiner la paramétrisation des Q-valeurs, l'apprentissage off-policy et le bootstrap pour calculer la cible (ce que fait DQN-TD) peut diverger : il s'agit du "deadly triad" [21]. Nous montrons que DQN-MC atteint de bonnes performances sur les sous-tâches de petite et de moyenne dimension. Les tâches de dimension plus grande sont plus difficiles à résoudre parce que les parties durent en moyenne plus longtemps ce qui est une difficulté tant en terme de temps de calcul que pour la stabilité de l'apprentissage (il y a une plus grande variance des récompenses obtenues lors des épisodes). Pour passer à l'échelle sur de tels problèmes, nous divisons l'espace des états en une partition et utilisons les sous-tâches déjà résolues comme des méthodes de bootstrap.

Algorithm 1 DQN-MC with Bootstrapping on already solved sub-tasks.

```
Start with low dimensional tasks.
for i such that the task T_i has not been yet optimized do
  if |B_i| ≤ 30 then
    while the budget for the optimization of this task has not been reached do
      Play 100 games (ε-greedy) from the start s_(i) to a terminal state.
      Integrate all the obtained transitions to the Replay-Memory
      Throws part of the Replay-Memory away (the oldest transitions of the replay)
      Sample 1/20 of the Replay-Memory
      Perform a gradient ascent step (backpropagation algorithm) on the sample
    end while
  end if
end for

Continue with higher-dimension tasks.
while there are still tasks to be optimized do
  Choose the easiest task to optimize : the one with the highest proportion
  of already solved sub-tasks (weighted by their probability to be faced)
  while the budget for the optimization of this task has not been reached do
    Play 100 games (ε-greedy) from the start s_(i) to a terminal state (condition (1))
    or to a state that was yet encountered in an already solved task (condition (2))
    if we stopped a game because of condition (2) then
      Bootstrap i.e use the network of the sub-tasks to predict
      the average number of question to reach a terminal state
    end if
    Integrate all the obtained transitions to the Replay-Memory
    Throws part of the Replay-Memory away (the oldest transitions of the replay)
    Sample 1/20 of the Replay-Memory
    Perform a gradient ascent step (backpropagation algorithm)
  end while
end while
```

Résoudre les sous-tâches de plus grande dimension en faisant du bootstrap avec des sous-tâches déjà résolues

Nous notons $\mathbb{B}_i = (B_{i_1}, ..., B_{i_k})$ l'ensemble des symptômes liés au symptôme i, c'est-à-dire l'ensemble des symptômes qui sont toujours suspects après avoir observé la présence du symptôme i. Lorsque $|\mathbb{B}_i|$ est suffisamment petit (disons $|\mathbb{B}_i| < 11$), nous pouvons apprendre la stratégie optimale π^* par un simple algorithme de value-iteration, voir [21].

Pour ce qui est des problèmes de dimension intermédiaire (disons $11 < |\mathbb{B}_i| < 31$) nous pouvons utiliser l'algorithme DQN-MC qui obtient rapidement de bonnes performances sur ces problèmes (voir les expériences de la section 2.3). Pour les problèmes de haute-dimension ($|\mathbb{B}_i| > 30$) utiliser directement l'algorithme DQN serait trop coûteux en terme de puissance de calcul. Une façon simple d'accélérer la phase d'apprentissage de ces gros réseaux et de faire usage des plus petit réseaux préalablement optimisés. En effet, si B_i est un symptôme pour lequel $|\mathbb{B}_i|$ est grand, il existe probablement $B_j \in \mathbb{B}_i$ tel que $|\mathbb{B}_j|$ est suffisamment petit et donc tel que les Q-valeurs de $\pi^\star_{(j)}$ ont déjà été calculé ou au moins approché. Pour le dire autrement, lorsque nous cherchons à apprendre le Q-réseau optimal d'un problème donnée, nous connaissons déjà, pour certains inputs, les Q-valeurs que devrait renvoyer un Q-réseau quasi-optimal.

L'idée est donc d'entraîner le réseau en jouant des parties commençant par l'état initial $s^{(i)}$ et faisant du bootstrap avec un réseau déjà optimisé lorsque l'on atteint un état qui appartient à l'espace des états où il existe déjà un réseau optimisé. Nous avons en pratique une fonction qui est appelée à chaque fois que l'on reçoit une réponse positive afin de vérifier s'il n'existe pas déjà un réseau optimisé partant d'un tel symptôme. Si c'est effectivement le cas, la partie courante est arrêtée et le réseau optimisé correspondant est utilisé pour prédire le nombre moyen de question à poser pour atteindre un état terminal. Le schéma de cette procédure est détaillé dans l'algorithme 1.

Notez qu'en procédant ainsi nous n'optimisons pas le réseau sur la totalité de la tâche. Il sera donc nécessaire de changer de réseau pour faire les recommandations durant l'examen lorsque nous changerons d'élément de la partition de l'espace des états. L'avantage et que nous n'avons pas besoin d'utiliser une architecture plus complexe pour les tâches de plus grande dimension.

Finalement, notez que nous apprenons les Q-réseaux les uns après les autres et qu'il y a un ordre préférable aux autres pour les optimiser. Nous choisissons à chaque étape d'optimiser le Q-réseau qui a le plus fort taux de sous-problème déjà résolus (où chaque sous-tâche a un poids égale à sa probabilité d'être rencontrée).

2.3 Résultats numériques

Pour toutes les expériences mettant en jeu des réseaux de neurones, nous utilisons toujours la même architecture détaillée dans le tableau 1. Nous utilisons tout d'abord un embedding layer puisque les entrées que reçoit le réseau ne

devraient pas être traitées comme des valeurs numériques. Nous utilisons ensuite deux couches cachées avec une activation de type ReLu et une couche finale avec une fonction d'activation linéaire qui donne en sortie les Q-valeurs des différentes actions possibles. Le paramètre ϵ de notre critère d'arrêt est fixé à 10^{-6} pour toutes les expériences.

TABLE 1 – Architecture du réseau de neurone pour la tâche $\mathcal{T}_i$. $|\mathbb{B}_i|$ le nombre de symptôme suspects restant.

Name	Type	Input Size	Output Size
L1	Embedding Layer	$\lvert\mathbb{B}_i\rvert$	$3 \times \lvert\mathbb{B}_i\rvert$
L2	ReLu	$3 \times \lvert\mathbb{B}_i\rvert$	$2 \times \lvert\mathbb{B}_i\rvert$
L3	ReLu	$2 \times \lvert\mathbb{B}_i\rvert$	$\lvert\mathbb{B}_i\rvert$
L4	Linear	$\lvert\mathbb{B}_i\rvert$	$\lvert\mathbb{B}_i\rvert$

Ces expériences ont été réalisé sur un ordinateur portable sans faire usage de GPU et devrait donc être facilement reproductibles en utilisant notre simulateur d'environnement ou un simulateur similaire.

Notre référence stratégie-centrée a des performances quasi-optimale sur les sous-problèmes de petite dimension. Nous pouvons comparer les performances de nos stratégies entraînées par un algorithme REINFORCE (celles de la section 2.2), avec une stratégie classique d'arbre de décision que nous appelons stratégie de Breiman : choisir l'action qui minimise l'entropie de la cible à l'horizon 1 (voir [25]). Nous comparons également ces stratégies avec la stratégie optimale lorsque celle-ci a pu être calculé, c'est-à-dire lorsque la dimension est suffisamment petite.
Les résultats sur une partie de nos sous-tâches sont présentés sur la figure 2. Notre stratégie energy-based apparaît clairement supérieure à un algorithme classique de Breiman et ce d'autant plus que la dimension augmente : le nombre moyen de question à poser peut-être divisé par deux dans certains cas. Pour les sous-tâches de faible dimension où nous avons pu calculer la stratégie optimale par un algorithme de programmation dynamique, notre stratégie energy-based apparaît très proche des performances de la stratégie optimale.

DQN-MC vs notre référence. Nous avons exécuté un algorithme DQN-MC sur nos sous-tâches. Il est attendu que cet algorithme trouve un meilleur chemin que la stratégie energy-based de la section 2.2 puisqu'un réseau de neurones a beaucoup plus de paramètre et peut donc faire face à des situations beaucoup plus diverses que notre référence. Cependant entraîner une fonctionnelle avec autant de paramètre plutôt que notre référence et ses trois paramètres a un coût. De combien d'itérations a besoin DQN-MC pour dépasser notre référence ?
Nous rappelons ici qu'une itération de DQN-MC consiste à jouer 100 parties qui sont ajoutées au replay memory dont nous échantillonnons ensuite un vingtième et réalisons un pas de montée de gradient. En comparaison, notre référence a été entraînée avec un algorithme REINFORCE,

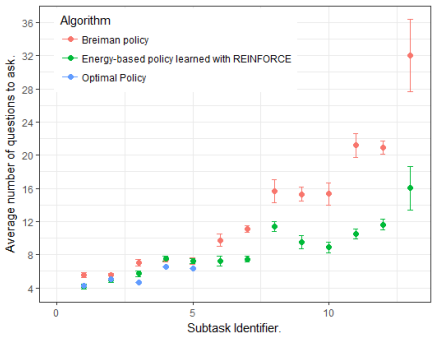


FIGURE 2 – Nombre moyen de questions à poser pour plusieurs sous-tâches.

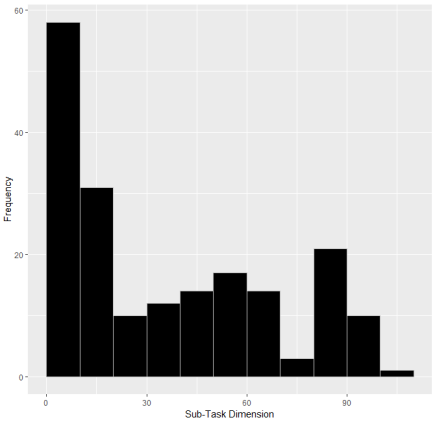


FIGURE 3 – Histogramme des dimensions des différentes sous-tâches.

chaque itération consiste à jouer une partie et à réaliser un pas de montée de gradient, nous arrêtons la phase d'apprentissage après 1000 itérations.
Les figures 6 et 7 montre, comme attendu, que DQN-MC a besoin de jouer plus de parties que notre référence. En effet sur ces deux sous-tâches, DQN-MC a besoin de respectivement 40 et 200 itérations pour atteindre notre référence, c'est-à-dire $40 \times 100 = 4000$ et $200 \times 100 = 20000$ parties au lieu des 1000 qui ont entraîné notre référence. Sur la figure 6, pour une sous-tâche de dimension 10, nous pouvons voir que DQN-MC a besoin d'une quantité raisonnable de parties pour dépasser notre référence. Dans ce cas, DQN-MC trouve une très bonne stratégie mais n'atteint pas tout à fait la stratégie optimale, l'algorithme est probablement bloqué dans un extremum local (bien que nous utilisions un paramètre d'exploration).
Sur la figure 7, nous pouvons voir que DQN-MC semble converger vers la référence. Cela est probablement dû au fait que, dans ces tâches de dimension intermédiaire (il reste 26 symptômes suspects et 8 maladies), notre référence est déjà une très bonne solution proche de l'optimale. Ainsi DQN-MC qui n'est pas assuré de converger vers une stratégie optimale peut rester bloquer dans un extremum

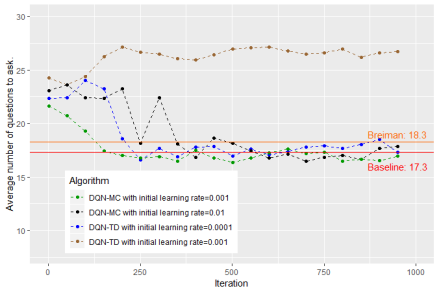


FIGURE 4 – Comparaison de DQN-TD et DQN-MC. Dimension de la tâche : 29.

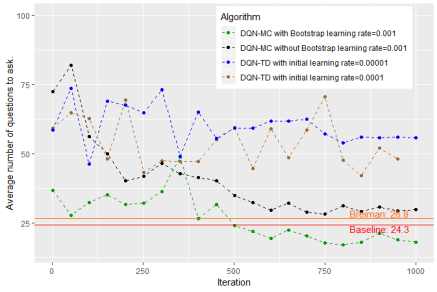


FIGURE 5 – Comparaison de DQN-TD, DQN-MC et DQN-MC-Bootstrap. Dimension de la tâche : 104.

local au niveau de la référence.

Finalement, comme nous pouvions nous y attendre en considérant la différence d'itérations nécessaires pour converger entre la figure 6 et la figure 7, nous verrons dans la prochaine section que l'idée d'utiliser les sous-tâches préalablement résolue sera importante pour faire face aux tâches de grande dimension.

Le bootstrapping en utilisant des sous-tâches déjà résolues aide (beaucoup) pour les tâches de grande dimension. Dans ces expériences, nous comparons les performances d'un algorithme DQN-MC face à celle de DQN-MC-Bootstrap sur certaines de nos tâches. Nous utilisons la même architecture pour le réseau des deux algorithmes (voir le tableau 1). Les deux algorithmes utilisent exactement les mêmes hyper-paramètres, la seule différence étant l'usage de l'astuce du bootstrap pour DQN-MC-Bootstrap.

Les figures 8 et 5 montrent les bénéfices de l'usage du bootstrap sur les sous-tâches déjà résolues. Dans ces deux cas, un algorithme DQN-MC basique n'est pas capable de trouver une bonne solution alors que DQN-MC-Bootstrap dépasse plutôt rapidement notre référence. Notez que le réseau de neurones entraîné avec DQN-MC-Bootstrap commence avec une stratégie qui n'est pas si mauvaise. Cela est appréciable car cela réduit, dès le début de la phase d'apprentissage, les longueurs des épisodes et donc le coût en calcul associé.

Pour l'expérience de la figure 8 il reste 70 symptômes suspects, 9 maladies possibles incluant la maladie "autre", et

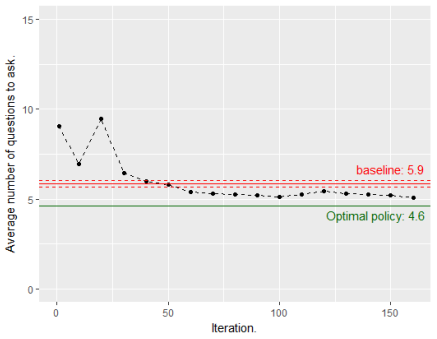


FIGURE 6 – Evolution de la performance du réseau de neurone durant la phase d'apprentissage avec DQN-MC. Dimension de la tâche : 10.

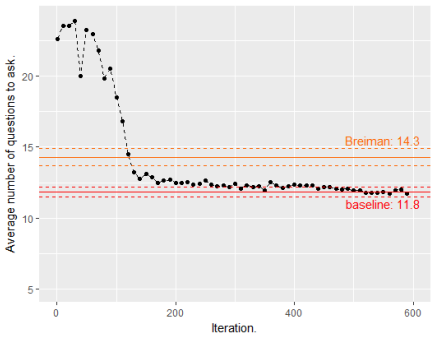


FIGURE 7 – Evolution de la performance du réseau de neurone durant la phase d'apprentissage avec DQN-MC. Dimension de la tâche : 26.

20 sous-tâches ont déjà été résolues. Enfin, les probabilités de présence des symptômes initiaux de chaque sous-tâche sachant la présence du symptôme initial de la tâche principal sont les suivantes : (0.01 ; 0.44 ; 0.01 ; 0.15 ; 0.15 ; 0.01 ; 0.03 ; 0.02 ; 0.11 ; 0.01 ; 0.26 ; 0.01 ; 0.03 ; 0.01 ; 0.15 ; 0.01 ; 0.15 ; 0.24 ; 0.16 ; 0.06).

Pour l'expérience de la figure 5 il reste 104 symptômes suspect, 18 maladies possibles incluant la maladie "autre", et 103 sous-tâches ont déjà été résolues.

Pour finir nous avons été capable d'apprendre une bonne stratégie pour la tâche principale (2) où il reste 220 symptômes suspects, 82 maladies possibles incluant la maladie "autre" et où toutes les sous-tâches ont déjà été résolues. DQN-MC-Bootstrap commence avec une bonne stratégie qui a seulement besoin de 45 questions en moyenne pour atteindre un état terminal. Quelques itérations lui permette de s'améliorer jusqu'à n'avoir plus besoin que de 40 questions. A l'inverse l'expérience que nous avons conduit pour DQN-MC essayant de résoudre la tâche principale en partant de rien a besoin de 117 questions, en moyenne, pour atteindre un état terminal et ne s'améliore pas de manière significative durant les 1000 iterations. Nous avons également évalué les performances de la stratégie de Breiman sur la tâche générale, celui-ci a besoin de 89 questions en

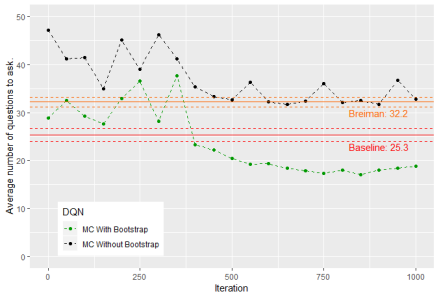


FIGURE 8 – Evolution de la performance du réseau de neurone durant la phase d'apprentissage. Dimension de la tâche : 70.

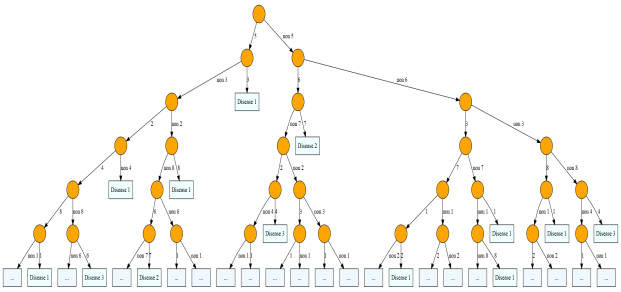


FIGURE 9 – Arbre de décision optimal pour une sous-tâche ayant 8 symptômes possibles restant.

moyenne pour atteindre un état terminal (avec une variance de 10 questions).

Une analyse qualitative pour une sous-tâche de faible dimension. Nous analysons ici la politique obtenue en utilisant un algorithme de type "value-iteration" sur une sous-tâche de faible dimension (dans cet exemple, il reste 8 symptômes pertinents à vérifier) afin d'illustrer certains des dilemmes auxquels un médecin peut faire face durant un examen. Nous commençons par la présence du symptôme 9. Les trois maladies qui ont le symptôme 9 dans leur liste de symptômes typiques sont affichées dans le tableau 2. Supposons, pour cette seule expérience, que les symptômes sont conditionnellement indépendants de la maladie. Une autre information importante est la prévalence de chaque maladie, nous avons $\mathbb{P}[D = d_1] = 0.042$, $\mathbb{P}[D = d_2] = 0.0083$ et $\mathbb{P}[D = d_3] = 0.0083$. Enfin, il n'y a pas de relation d'ascendant/descendant entre les symptômes de 9 considérés dans cet exemple. La stratégie optimale obtenue induit un arbre de décision qui est affiché dans la Figure 9.

TABLE 2 – Liste des maladies probables et leur liste de symptômes typiques avec leur probabilité pour la sous-tâche commençant par la présence du symptôme 9.

Maladie 1		Maladie 2		Maladie 3	
Id Symptôme	Probabilité	Id Symptôme	Probabilité	Id Symptôme	Probabilité
1	0.50	6	0.90	2	0.90
2	0.55	7	0.50	4	0.90
3	0.50	9	0.90	6	0.50
5	0.90			9	0.50
8	0.50				
9	0.50				

La première question est compréhensible, il s'agit du symptôme le plus probable de la maladie la plus probable : le symptôme 5. Si la réponse est positive, la stratégie continue avec un symptôme typique de la première maladie qui n'est pas typique des autres maladies : le symptôme 3. La combinaison de ces deux symptômes (en plus du symptôme 9 donc) est suffisante pour diagnostiquer la maladie 1. Le reste de l'arbre est moins évident. Par exemple, lorsque nous obtenons un "oui" pour le symptôme 5 et un "non" pour le symptôme 3, devrions-nous continuer à poser des questions relatives à des symptômes typiques de la maladie 1 ou devrions-nous changer de cible et poser des questions typiques d'autres maladies ? Le chemin trouvé choisi un symptôme lié à la fois à la maladie 1 et à la maladie 3 (le symptôme 2), probablement parce qu'il est ensuite simple (et rapide) d'écarter l'éventualité de la maladie 3 en posant une question sur le symptôme 4 (notez que la maladie 3 n'a que 3 symptômes typiques).

Une autre partie intéressante de l'arbre est lorsque nous recevons une réponse négative à notre première question sur le symptôme 5. Alors, la maladie initialement la plus probable (la maladie 1) devient moins probable, mais il n'est pas clair que cette probabilité ai suffisamment diminué pour que nous en arrivions à changer de cible et à décider de vérifier des symptômes typiques d'autres maladies. Dans ce cas la stratégie optimale est de changer de cible et de s'orienter vers des symptômes typiques de la maladie 3 qui a moins de symptômes typiques et doit être (dans cette partie de l'arbre) plus probable que la maladie 1.

Du fait de limites de taille, nous n'avons pas pu représenter la totalité de l'arbre de décision, nous nous sommes limités à une profondeur maximale de 6 nœuds et nous écrivons "..." pour les feuilles où la stratégie de diagnostic continue de proposer de nouveaux symptômes à aller observer.

3 Apprendre un modèle de l'environnement.

Comme décrit dans la section 2 notre agent est entraîné en jouant des parties obtenues lors d'interactions avec son environnement. Les transitions (état de départ, action, état atteint, récompense)=(s_t, a_t, s_{t+1}, r_t) utilisées pour améliorer la stratégie de diagnostic peuvent être obtenues en interagissant directement avec l'environnement dans le monde réel (model-free RL) ou par simulation en utilisant un modèle approché de l'environnement (model-based RL).

Dans notre problème, comme dans la plupart des cas d'applications pratiques de l'apprentissage par renforcement, une approche model-free n'est pas envisageable. Il est, en effet, souvent impossible dans un contexte médical comme industriel, de déployer dans la vie réelle un algorithme non encore optimisé du fait des coûts engendrés par de mau-

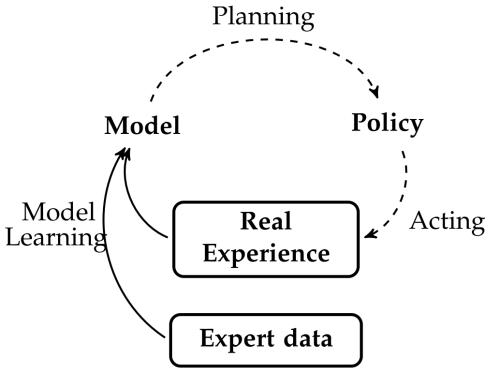


FIGURE 10 – L'architecture globale.

vaises décisions.
De plus le faible nombre de données disponibles, caractéristique typique des maladies rares, nous empêche d'envisager toute approche reposant uniquement sur des données et se passant des connaissances expertes disponibles. Il nous faut donc construire un simulateur. Notez que dans un certain nombre d'applications populaires de l'apprentissage par renforcement un tel modèle de l'environnement n'est pas nécessaire : citons les jeux adversariaux où la solution classique est de jouer contre soi-même [14, 15].
Pour construire notre modèle, nous disposons tout d'abord de données expertes (voir section 1.2) ainsi que de données cliniques collectées au fur et à mesure de l'utilisation de l'algorithme. Il s'agit de tirer profit au mieux de ces deux sources d'information pour inférer la loi de notre environnement.

3.1 Mélanger experts et données

L'objectif Soit $p^\star = (p_1^\star, ..., p_K^\star)$ la distribution que nous cherchons à estimer. Dans notre cas précis, nous nous plaçons à l'échelle d'une maladie et cherchons à estimer les probabilités des différentes combinaisons de symptômes typiques conditionnellement au fait de présenter la maladie en question.
Ainsi, soient B_1,..., B_L les symptômes typiques de la maladie D nous cherchons les probabilités des 2^L différentes combinaisons $\mathbb{P}[B_1, ..., B_L \mid D]$ alors que nous n'avons de données que sur les marginales $\mathbb{P}[B_i \mid D]$.
Bien sûr une première idée serait de faire l'hypothèse que les symptômes sont conditionnellement indépendant sachant la maladie. Nous nous attendons cependant à des corrélations complexes entre les symptômes d'une maladie donnée. En effet nous pouvons imaginer deux symptômes individuellement très plausibles mais qui surviennent rarement ensemble (ou même jamais dans le cas de symptômes incompatibles comme par exemple microcéphalie et macrocéphalie).
De plus, notons que l'hypothèse d'indépendance conditionnelle permettrait de présenter une maladie sans avoir aucun des symptômes liés à cette maladie dans la base de données (quand il n'y a pas B_i tel que $\mathbb{P}[B_i \mid D] = 1$), ce qui devrait être impossible.

Idée générale Notons X_1,..., X_n un échantillon i.i.d de $p^\star$. La distribution empirique $p_n^{\text{emp}} = (p_{n,i}^{\text{emp}})_{i=1}^K$ est donnée par :

$$p_{n,i}^{\text{emp}} = \frac{1}{n} \sum_{j=1}^{n} \mathbb{1}_{\{X_j = i\}}. \tag{4}$$

Nous pouvons également définir la distribution des experts comme :

$$p^{\text{maxent}} = \underset{p/p \in \tilde{\mathcal{C}}}{\arg\max}\, H(p) \tag{5}$$

où $\tilde{\mathcal{C}} = \mathcal{C} \bigcap \mathcal{C}^{\text{expert}}$. $\mathcal{C} = \{p / \sum_i p_i = 1, p_i \geq 0\}$ est le simplexe des probabilités et $\mathcal{C}^{\text{expert}}$ est l'ensemble des contraintes fixées par les experts. Dans notre cas, les contraintes prennent la forme de marginales fixées ou bien encore de combinaisons censurées, c'est-à-dire de combinaisons que l'on force à être de probabilité nulle car l'on sait que ces évènements sont impossibles : par exemple le fait de présenter la maladie sans avoir aucun de ses symptômes ou encore d'avoir simultanément deux symptômes exclusifs. Toutes les combinaisons de contraintes sont imaginables tant que $\tilde{\mathcal{C}}$ reste un espace fermé convexe, et ce afin d'assurer l'existence et l'unicité de p^{maxent}.
Nous construisons donc la distribution des experts en prenant la distribution d'entropie maximale vérifiant les contraintes imposées par les experts. Ainsi parmi l'infinité des distributions qui vérifient les contraintes imposées par les experts nous choisissons la distribution p^{maxent} la moins informative, autrement dit la plus proche de la distribution d'indépendance conditionnelle.
Il nous faut en effet ajouter de l'information pour passer des indications fournies par les experts à la distribution finale et nous souhaitons en ajouter aussi peu que possible sur ce que nous ne connaissons pas. Cette approche est désignée sous le terme de maxent (maximum entropy) et a été largement étudiée dans la littérature [2], [3], [4].
Nous sommes désormais prêts à mélanger experts et données. L'idée générale est que si nous avions suffisamment de données issues d'expériences directes de l'algorithme, nous n'aurions plus besoin des experts. À l'inverse, sans données expérimentales, notre modèle doit s'appuyer entièrement sur les données expertes.
Définissons $\mathcal{L}$ une mesure quelconque de dissimilarité entre deux mesures de probabilités. Notre estimateur mélangeant données et experts est alors définit comme suit :

$$\hat{p}_{\epsilon_n}^{\mathcal{L}} = \underset{p \in \mathcal{C} / \mathcal{L}(p_n^{\text{emp}}, p) \leq \epsilon_n}{\arg\min} \mathcal{L}(p^{\text{maxent}}, p) \tag{6}$$

où

$$\epsilon_n := \epsilon_n^\delta = \underset{l}{\arg\min}\, \mathbb{P}[\mathcal{L}(p_n^{\text{emp}}, p^\star) \leq l] \geq 1 - \delta. \tag{7}$$

$\hat{p}_n^{\mathcal{L}}$ est ainsi définie comme étant la distribution la plus proche des experts, au sens de la mesure de dissimilarité $\mathcal{L}$, qui soit consistante avec les données observées.
Pour qu'une telle construction soit possible nous devrons donc choisir une mesure de dissimilarité $\mathcal{L}$ telle que nous disposions d'une concentration de la distribution empirique autour de la vraie distribution pour $\mathcal{L}$.

3.2 Travaux apparentés

Les statistiques bayésiennes. La volonté de tirer simultanément profit de données expertes et de données d'expériences a bien sûr une histoire très ancienne. Il s'agit là de l'essence même des statistiques bayésiennes [48] qui visent à incorporer les données expertes, sous la forme d'un a priori, à des données d'expériences en faisant usage du théorème de Bayes pour obtenir ce qui sera appelé le posterior.
Notez que dans notre cas nous ne disposons pas d'un a priori classique modélisant les paramètres du modèle par des lois de probabilité. Notre a priori porte sur les marginales et sur un certains nombres de contraintes sur la distribution à estimer. L'absence d'a priori évident pour modéliser la distribution des paramètres nous mène naturellement à l'idée de maximum d'entropie théorisé par Jaynes [2]. En effet, si aucun modèle ne nous apparaît plus plausible qu'un autre alors notre choix se portera sur le moins informatif. Il s'agit d'une généralisation du principe d'indifférence souvent attribué à Laplace : "On regarde deux évènements comme également probables, lorsque qu'on ne voit aucune raison qui rende l'une plus probable que l'autre, parce que, quand bien même il y aurait une inégale possibilité entre elles, comme nous ignorons de quel côté est la plus grande, cette incertitude nous fait regarder l'une comme aussi probable que l'autre" [50].
Ce principe prend donc la forme d'un axiome nous permettant de construire une méthode pour choisir un a priori : le moins informatif possible consistant avec ce que nous savons.

Système expert avec raisonnement probabiliste. La création d'un outil d'aide à la décision pour le diagnostic médical a été un objectif depuis le début de l'ère informatique. La plupart des premiers travaux proposaient un système expert basé sur des règles, mais dans les années 80, une partie importante de la communauté a étudié la possibilité de construire un système expert utilisant des raisonnements probabilistes [16]. Les probabilités et les méthodes bayésiennes ont donc été relativement tôt considérées comme de bon moyens de modéliser l'incertitude inhérente au diagnostic médical.
L'hypothèse d'indépendance conditionnelle des symptômes sachant la maladie a été intensivement discutée car elle est d'une importance cruciale pour la complexité computationnelle. Certains chercheurs ont jugé que cette hypothèse était sans conséquence [23] alors que d'autres proposaient déjà une approche de type maximum d'entropie pour répondre à cette question [11], [12] ou encore [13].
Il semble cependant qu'aucun des travaux de l'époque n'ai considéré le compromis expert/données auquel nous faisons face. Dans l'article [17] faisant l'état des lieux de la recherche de l'époque (1990) sur cette question, il est clairement mentionné que ces méthodes ne prennent en entrée que des données sous forme probabiliste. Plus précisément ils supposent disposer d'un a priori sur les marginales mais également sur certaines des combinaisons de symptômes (dans notre cas nous ferions l'hypothèse d'avoir un a piori sur $\mathbb{P}[B_1, B_2 \mid D]$ par exemple) et proposent une approche de type maximum d'entropie où ces données expertes sont traitées comme des contraintes dans le processus d'optimisation. Encore une fois, ce n'est pas notre cas puisque nous n'avons qu'un a priori sur les marginales (et un certains nombre de contraintes) ainsi que des données expérimentales. Ce domaine de recherche a été très actif dans les années 80 puis a progressivement disparu, probablement du fait de l'intractabilité computationnelle des algorithmes proposés pour les moyens informatiques de l'époque.

Les réseaux bayésiens. Les réseaux bayésiens ont ensuite rapidement été considéré comme une alternative prometteuse pour modéliser les relations de dépendance probabiliste entre symptômes et maladies [16]. Ceux-ci sont désormais utilisés dans la majeure partie des systèmes experts notamment en médecine [52].
Un réseau bayésien est généralement défini comme un graphe orienté acyclique. Les nœuds de ce graphe correspondent aux variables aléatoires : les symptômes ou les maladies dans notre cas. Les arêtes relient deux variables aléatoires corrélées en intégrant l'information de la loi conditionnelle du nœud fils sachant le nœud père. Le principal avantage d'un tel modèle est de pouvoir factoriser la loi jointe en utilisant la propriété dite de Markov globale. La loi jointe peut en effet être exprimée comme le produit des distributions conditionnelles de chaque nœud sachant ses parents directs dans le graphe [53].
La construction d'un réseau bayésien suppose tout d'abord d'inférer sa structure, c'est-à-dire de déterminer les nœuds qui doivent être liés par une arête de ceux qui peuvent être considérés comme conditionnellement indépendant au reste du graphe (structure learning). Ensuite, l'apprentissage du réseau suppose d'apprendre les paramètres, c'est-à-dire les probabilités liant les nœuds (parameter learning).
Il est donc naturel de trouver également dans ce pan de la littérature, des travaux visant à mélanger des données expertes et des données empiriques. Dans [55] les indications des experts prennent une forme particulière puisque ceux-ci indiquent à la main des corrélations, positives ou négatives, entre des variables. L'approche de [54] est également assez éloignée car basée préférentiellement sur des données. [54] ne fait intervenir des indications expertes que pour des variables additionnelles pour lesquelles il n'y a pas de données, typiquement des évènements rares jamais observés dans la base. Un travail plus proche du notre est [51] où les auteurs supposent disposer d'un premier réseau bayésien construit entièrement par les experts, auquel

ceux-ci associent un degré de confiance. Les auteurs utilisent ensuite les données disponibles pour corriger ce réseau expert. Nous nous distinguons de ce travail dans notre effort pour trouver une procédure objective du poids à donner aux experts face aux données (et que ce poids ne soit pas fixé par les experts eux même).
Notez de plus que l'intérêt principal des réseaux bayésiens est de tirer partie de relations d'indépendance conditionnelle connues à l'avance, car pré-remplies par des experts ou inférées à partir d'une quantité suffisante de données. Or, dans notre cas, nous n'avons pas une telle connaissance a priori sur les relations de dépendance entre symptômes et pas assez de données pour les inférer.

Apprentissage par renforcement bayésien. Nous avons ici présenté de manière séparée la partie planification (recherche de politique optimale) de celle d'apprentissage du modèle de transition. Toutefois, une partie de la littérature connue sous le nom d'apprentissage par renforcement bayésien propose de mêler ces deux objectifs, voir [19] pour un aperçu de ce domaine. Il s'agit principalement de reconsidérer dans les algorithmes classiques de renforcement la notion de compromis exploration/exploitation. Il est, en effet, nécessaire d'octroyer un bonus pour explorer davantage, car les probabilités de transition partant d'états peu visités risquent d'être mal estimées. Le risque est alors de passer à côté de bonnes solutions du fait d'un modèle de transition mal connu.
Cette exploration, au sens de l'utilisation en pratique d'une stratégie différente de la politique optimale apprise, existe déjà en quelque sorte dans notre système. En effet, l'utilisateur peut choisir de répondre à n'importe quelle question et non pas à la première que nous ayons listé.
Nous n'avons pas davantage traité cette question et avons fait l'hypothèse que les données cliniques reçues pour estimer le modèle de transition sont complètes. Cela signifie que lorsque le fœtus est atteint d'une pathologie (ce qui est loin d'être le cas le plus fréquent) il faudra prendre le temps de vérifier la présence/absence des symptômes typiques, que ce soit au moment de l'échographie ou plus tard via des examens complémentaires.

Des marginales aux lois jointes. Estimer la distribution jointe à partir des marginales est un problème ancien, qui n'est évidemment pas nécessairement lié aux systèmes experts. Ce problème est parfois connu dans la littérature sous le nom de problème d'estimation des probabilités des cases de tableau de contingence avec marginales fixées ("cell probabilities estimation problem in contingency table with fixed marginals"). Le livre [18] donne un bon aperçu de ce domaine. Nous pouvons remonter au travail de Deming et Stephan [22] qui suppose connaître les marginales et avoir accès à un échantillon de données expérimentales et cherche à estimer la loi jointe. Ils ont proposé dans cet article l'algorithme "iterative proportional fitting procedure" (IPFP), qui est resté très populaire pour résoudre ce problème.
Une hypothèse importante de [22] est que chaque case du tableau de contingence reçoit des données. Dans [42] les auteurs prouvent que l'estimateur asymptotique obtenu par un algorithme IPFP est la distribution qui minimise la divergence de Kullback-Leibler vis-à-vis de la distribution empirique sous contrainte de respecter les marginales expertes.
Cependant un algorithme IPFP n'est pas approprié pour notre problème pour deux raisons principales : tout d'abord nous n'avons pas une confiance absolue dans les marginales données par les experts (nous souhaitons nous donner la possibilité de les modifier au fur et à mesure de l'afflux des données) et ensuite parce que comme nous nous intéressons aux maladies rares nous ne nous attendons pas à avoir un nombre suffisant de données. De fait, beaucoup de case de la table de contingence que nous cherchons à estimer ne recevrons pas de données, mais il serait désastreux dans notre application d'assigner une probabilité nulle à la combinaison de symptôme correspondante. Mentionnons également les travaux liés à notre problème dans des applications des statistiques aux sciences sociales où les chercheurs visent à construire une population synthétique à partir des marginales provenant de plusieurs sources incohérentes [40]. L'appoche proposée fait également appel à des idées de maximum d'entropie mais il n'y a pas non plus ici d'équivalence avec notre problème de compromis expert/expérience puisqu'ils construisent leur modèle sans échantillons.

Les centroides de Kullback. Notre problème d'optimisation (6) lorsque la mesure de dissimilarité $\mathcal{L}$ est la divergence de Kullback Leibler porte le nom de moment-projection (M-projection) dans la littérature. Les propriétés de ces projections ont été intensément étudié [46].
Notez que le Lagrangien associé à un tel problème d'optimisation n'est alors rien de plus qu'un centroïde de Kullback-Leibler. Ces objets ou des variations/généralisation de ceux-ci (avec des divergences de Jeffrey, de Bregman etc) ont fait l'objet de recherche depuis le papier de Veldhuis [43]. Par exemple les articles [5] et [44] de Nielsen étudie les cas où une formule exacte peut-être obtenue et propose des algorithmes lorsque ce n'est pas le cas.
Nous n'avons cependant pas trouvé d'usage de ces centroïdes pour trouver de bon mélange expert/données comme nous le proposons dans ce papier. Les centroïdes de divergence de Bregman ont été utilisé pour mélanger plusieurs experts potentiellement contradictoires, le lecteur intéressé pourra se référer pour cela à la thèse récente d'Adamcik [45]. Nous pourrions certes considérer que la distribution empirique p_n^{emp} est un deuxième expert et que notre problème revient donc à mélanger deux experts : la littérature et les données. Cependant la question du poids à donner à chaque expert, qui constitue la question qui nous intéresse ici, ne sera pas pour autant réglée. Dans [45] le but est plutôt de synthétiser des avis contradictoires de différents experts en fixant au préalable le poids à donner à chaque expert. Nous proposons, pour notre part, une pro-

cédure objective pour déterminer le poids à donner aux experts face aux données empiriques.

3.3 Etude théorique de notre estimateur pour différentes mesures de dissimilarité

Barycentre dans des espaces normés Dans cette section nous nous plaçons dans les espaces L^p. Rappelons que la norme classique sur l'espace L^p est donnée par : $\|x\|_j = \left(\sum_i |x_i|^j\right)^{\frac{1}{j}}$.
En suivant les idées présentées plus haut nous définissons notre estimateur, $\forall i \geq 1, \forall j \geq 1$ comme :

$$\hat{p}_n^{i,j} = \underset{p \in \mathcal{C}/\|p - p^{\text{maxent}}\|_i \leq \epsilon_n}{\arg\min} \|p - p_n^{\text{emp}}\|_j \tag{8}$$

où

$$\epsilon_n := \epsilon_n^\delta = \arg\min_l \mathbb{P}[\|p_n^{\text{emp}} - p^\star\|_i \leq l] \geq 1 - \delta. \tag{9}$$

Pour contrôler ϵ_n nous utiliserons l'inégalité de concentration obtenue dans le travail récent [1]. Dans la littérature, la plupart des concentrations pour la distribution empirique utilisent la norme L^1. C'est pourquoi, même si nous présenterons dans la suite des résultats en cherchant à généraliser pour le plus de couple (i, j) possible, en pratique seuls les $\hat{p}_n^{1,j}$, pour tout $j \geq 1$, nous intéressent.

Proposition 1. *(existence et unicité) L'estimateur $\hat{p}_n^{i,j}$ définit par* (8) *existe quel que soit $i \geq 1, j \geq 1$.*
$\hat{p}_n^{i,j}$ est unique si et seulement si $i \neq 1$.
Dans la suite $\hat{p}_n^{1,1}$ désigne donc un ensemble de mesure de probabilité.

La prochaine proposition montre qu'une des solutions de (8) s'écrit toujours sous la forme d'un barycentre entre p_n^{emp} et p^{maxent} lorsque $i = j$. Cette propriété nous fourni donc, dans ces cas là, une expression explicite d'une solution de (8) qui par ailleurs n'était pas triviale à obtenir par un calcul direct visant à chercher les points selles du Lagrangien (par exemple dans le cas $i = j = 1$).

Proposition 2. *Soit $\hat{p}_n^{i,j}$ définit par* (8) *alors pour $i = j$, il existe $\tilde{p} \in \hat{p}_n^{i,j}$ tel que $\exists \alpha_n \in [0, 1]$:*

$$\tilde{p} = \alpha_n p^{\textit{maxent}} + (1 - \alpha_n) p_n^{\textit{emp}} \tag{10}$$

où $\alpha_n = \frac{\epsilon_n}{\|p_n^{\textit{emp}} - p^{\textit{maxent}}\|_i}$ si $\epsilon_n \leq \|p_n^{\textit{emp}} - p^{\textit{maxent}}\|_i$ et $\alpha_n = 1$ sinon.

En particulier un des éléments de $\hat{p}_n^{1,1}$ s'écrit sous la forme d'un barycentre. Par soucis de simplicité, nous désignerons dans la suite par $\hat{p}_n^{1,1}$ la solution de (8) pour $i = j = 1$ s'écrivant sous la forme (10) et non plus l'ensemble de toutes les solutions.

Remarque 1. *Noter que la proposition n'est pas valable pour $i = 1$ et $j \neq 1$. C'est pourquoi pour la fin de cette section nous nous concentrerons sur $\hat{p}_n^{1,1}$.*

Il s'agit désormais de dériver un résultat prouvant que mélanger experts et données comme nous le faisons avec $\hat{p}_n^{1,1}$ représente un intérêt par rapport à l'alternative de choisir de manière binaire l'un des deux modèles. Pour cela, nous montrons dans la proposition suivante, qu'avec grande probabilité, notre estimateur $\hat{p}^{1,1}$ est toujours meilleur que le meilleur des modèles à une constante près.

Théorème 1. *Soit $\hat{p}_n^{1,1}$ définit par* (8)*. Alors nous avons avec probabilité au moins $1 - \delta$:*

$$\|p^\star - \hat{p}_n^{1,1}\|_1 \leq 2 \min\{\epsilon_n, \|p^\star - p^{\textit{maxent}}\|_1\} \tag{11}$$

Barycentre en utilisant la divergence de Kullback-Leibler Dans cette section nous étudions le problème général (6) dans le cas particulier où la mesure de dissimilarité $\mathcal{L}$ est la divergence de Kullback-Leibler.
La divergence de Kullback-Leibler entre deux mesures discrètes p et q est définie comme :

$$D(p||q) = \sum_i p_i \log\left(\frac{p_i}{q_i}\right).$$

Rappelons que la divergence de Kullback-Leibler n'est pas une distance puisqu'elle n'est pas symétrique et ne satisfait pas l'inégalité triangulaire, elle est cependant bien définie positive [3].
Nous définissons notre estimateur comme :

$$\hat{p}_n^L = \underset{p \in \mathcal{C}/D(p_n^{\text{emp}}||p) \leq \epsilon_n}{\arg\min} D(p^{\text{maxent}}||p) \tag{12}$$

où

$$\epsilon_n := \epsilon_n^\delta = \arg\min_l \mathbb{P}[D(p_n^{\text{emp}}||p^\star) \leq l] \geq 1 - \delta. \tag{13}$$

Pour calibrer ϵ_n, nous pouvons utiliser l'inégalité de concentration obtenue dans [1]. Plus précisèment nous avons :

$$\epsilon_n = \frac{1}{n}\left(-\log(\delta) + \log\underbrace{\left(3 + 3\sum_{i=1}^{K-2}\left(\sqrt{\frac{e^3 n}{2\pi i}}\right)^i\right)}_{=:G_n}\right). \tag{14}$$

Dans la proposition suivante, nous montrons l'existence et l'unicité de notre estimateur $\hat{p}_n^L$ et le fait que notre estimateur est un barycentre. Il ne semble cependant pas possible cette fois, contrairement au cas de $\hat{p}_n^{1,1}$ de l'équation (8), d'obtenir une formule exacte pour $\hat{p}_n^L$.

Proposition 3. *Soit $\hat{p}_n^L$ définit par* (12) *alors $\hat{p}_n^L$ existe et est unique. De plus $\hat{p}_n^L$ peut s'écrire de la forme suivante :*

$$\hat{p}_n^L = \frac{1}{1 + \tilde{\lambda}} p^{\textit{maxent}} + \frac{\tilde{\lambda}}{1 + \tilde{\lambda}} p_n^{\textit{emp}} \tag{15}$$

où $\tilde{\lambda}$ est un réel positif tel que :

$$\tilde{\lambda} \geq \frac{D(p_n^{emp}||p^{maxent})}{\epsilon_n} - 1. \quad (16)$$

La proposition suivante se veut l'analogue de la proposition 1 lorsque $\mathcal{L}$ est la divergence de Kullback-Leibler. Nous prouvons que le centroïde $\hat{p}_n^L$ est meilleur que les experts (avec grande probabilité). D'autre part, nous obtenons que lorsque $D(p_n^{\text{emp}}||p^\star) > D(p^{\text{maxent}}||p^\star)$, le barycentre $\hat{p}_n^L$ est meilleur que la distribution empirique. Obtenir des garanties lorsque $D(p_n^{\text{emp}}||p^\star) \leq D(p^{\text{maxent}}||p^\star)$ semble moins évident et demande un contrôle sur la quantité $D(p_n^{\text{emp}}||p^{\text{maxent}})$.

Théorème 2. *Soit $\hat{p}_n^L$ définit par* (12) *alors nous avons avec probabilité au moins $1-\delta$:*

$$D(\hat{p}_n^L||p^\star) \leq \min\left\{D(p^{maxent}||p^\star), \epsilon_n\left(L_n+1\right)\right\} \quad (17)$$

où

$$L_n = \frac{D(p^{maxent}||p^\star) - D(p_n^{emp}||p^\star)}{D(p_n^{emp}||p^{maxent})}.$$

3.4 Quelques résultats numériques

Pour chaque expérience de cette section, nous générons une distribution aléatoire $p^\star$ que nous cherchons à estimer. Pour cela nous simulons des réalisations d'une loi uniforme et nous renormalisons afin de sommer à 1.

Nous générons également quatre distributions différentes qui feront office d'a priori pour l'inférence : $p^{\text{maxent},i}, \forall i \in \{1,2,3,4\}$. Les trois premiers priors sont obtenus par une procédure de maximum d'entropie sous contrainte de respecter des marginales de $p^\star$ ayant subies une modification. Nous avons ajouté aux marginales de $p^\star$ un bruit gaussien d'espérance nulle et de variance égale respectivement à $\sigma_1^2 = 0.1$, $\sigma_2^2 = 0.2$ et $\sigma_3^2 = 0.4$. Le dernier prior $p^{\text{maxent},4}$ est choisi égale à la distribution $p^\star$ (les experts nous ont fourni la bonne distribution).

Nous échantillonnons ensuite séquentiellement des données de $p^\star$, i.e nous générons des patients, et mettons à jour pour chaque nouvelle données et chaque a priori différent, le centroïde gauche $\hat{p}_n^L$ (en utilisant un algorithme Uzawa), le barycentre $\hat{p}_n^{1,1}$, la distribution empirique p_n^{emp} ainsi que les divergences $D(\hat{p}_n^L|p^\star)$ et $D(p_n^{\text{emp}}||p^\star)$ et les normes $\|\hat{p}_n^{1,1} - p^\star\|_1$ et $\|p_n^{\text{emp}} - p^\star\|_1$.

Les expériences des figures 11, 12, 13 ont été menées sur le cas d'une maladie ayant 7 symptômes typiques et où il y a donc $K = 2^7 = 128$ combinaisons possibles. L'expérience de la figure 14 a été menée sur le cas d'une maladie ayant 9 symptômes typiques et où il y a donc $K = 2^9 = 512$ combinaisons possibles.

Le seul paramètre que nous pouvons contrôler est le δ utilisé pour la construction de l'intervalle de confiance de la concentration de la distribution empirique autour de la vraie distribution. Rappelons-le, pour le cas du centroïde de Kullback de l'équation (12) :

$$\epsilon_n = \frac{1}{n}\left(-\log(\delta) + \log(G_n)\right) \quad (18)$$

où G_n est défini dans l'équation (14).

Cependant, nos premières expériences numériques montrent que le choix de ϵ_n défini par l'équation (18) est un peu trop conservateur : voir la figure 11. Nous avons besoin de faire converger ϵ_n plus rapidement vers 0 sans pour autant abandonner notre a priori lorsque celui-ci est bon.

Nos expériences suggèrent de prendre un ϵ_n conforme à la concentration proposée en conjecture de [1] pour la Kullback :

$$\epsilon_n = \frac{-\log(\delta) + \frac{n}{2}\log\left(1 + \frac{K-1}{n}\right)}{n}. \quad (19)$$

Notez que nous avons ajouté une constante $\frac{1}{2}$ par rapport à la conjecture de [1]. Concernant le choix de δ, celui-ci apparaît important principalement lorsque n est petit, le prendre suffisamment faible évite une situation de surapprentissage lorsque le nombre de données est encore faible sans pour autant se révéler nocif lorsque n est grand. Nous l'avons pris égale à 10^{-6} dans toutes nos expériences. Les figures 13 et 14 montrent qu'un tel choix pour ϵ_n donne un bon compromis entre expert et données car nous sommes capables de tirer profit de ces deux sources d'information lorsque le nombre de données est faible (typiquement lorsque $n < K$), mais aussi d'abandonner rapidement notre a priori lorsque celui-ci est mauvais (voir les courbes noires) ou de le conserver lorsque celui-ci est bon (les courbes vertes). Enfin les figures 13 et 14 ont été réalisées sur des problèmes de tailles 128 et 512 respectivement et ce choix de ϵ_n apparaît donc relativement robuste aux changements de dimension.

Concernant $\hat{p}_n^{1,1}$, nous avons pris, toujours en suivant les conjectures de [1] :

$$\epsilon_n = \sqrt{\frac{-\log(\delta) + \frac{n}{2}\log\left(1 + \frac{K-1}{n}\right)}{n}}. \quad (20)$$

La figure 12 montre l'erreur commise par notre barycentre en norme L^1 : $\hat{p}_n^{1,1}$ en utilisant un tel ϵ_n. Nous sommes à nouveau capable de nous débarasser relativement rapidemment d'un mauvais a priori pour suivre l'empirique (courbe noire, bleue et jaune) tout en le conservant si celui-ci est bon (courbe verte).

Notez que la manière dont nous simulons les distributions que nous cherchons à estimer (les $p^\star$) produit des distributions assez spécifiques : proches de l'uniforme et denses. Si nous simulons des distributions plus éparses, notre a priori p^{maxent} construit à partir de l'heuristique de maximum d'entropie sera mauvais et les expériences n'auront plus autant d'intérêt.

4 Conclusion et perspectives

Nous avons présenté dans ce travail une nouvelle notion, pour autant que nous le sachions, de ce que devrait être un bon outil d'aide à la décision pour le diagnostic de maladies rares. Nous avons pris en compte la nécessité, en

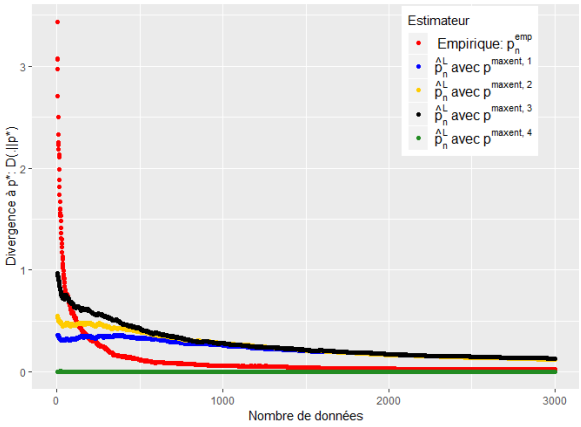


FIGURE 11 – Evolution de la performance de l'estimateur $\hat{p}_n^L$ en fonction du nombre de données disponibles. ϵ_n définit par équation (18)

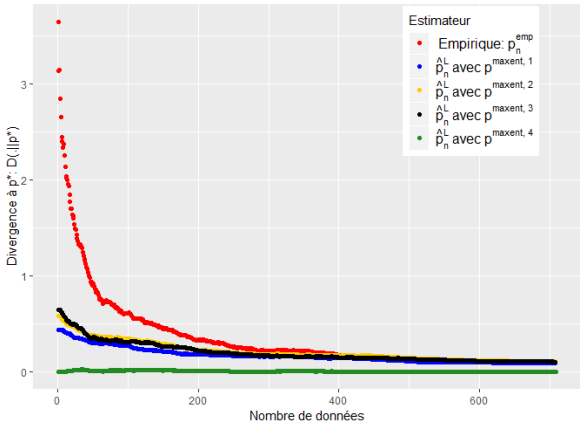


FIGURE 13 – Evolution de la performance de l'estimateur $\hat{p}_n^L$ en fonction du nombre de données disponibles. ϵ_n définit par l'équation (19).

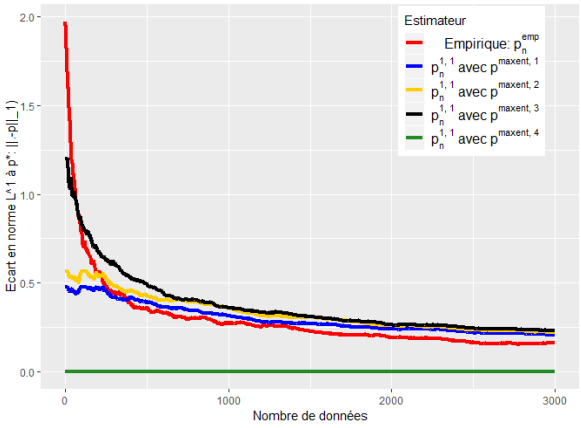


FIGURE 12 – Evolution de la performance de l'estimateur $\hat{p}_n^{1,1}$ en fonction du nombre de données disponibles. ϵ_n définit par équation (20).

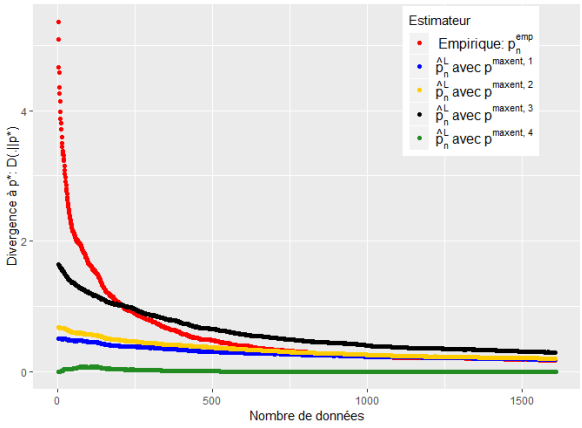


FIGURE 14 – Evolution de la performance de l'estimateur $\hat{p}_n^L$ en fonction du nombre de données disponibles. ϵ_n définit par l'équation (19).

médecine, d'atteindre un haut niveau de certitude au moment de poser un diagnostic. Nous essayons de minimiser le nombre moyen de tests médicaux à effectuer avant d'atteindre ce niveau de certitude. Nous avons étudié plusieurs algorithmes d'apprentissage de renforcement et nous les avons rendus opérationnels dans notre environnement en très haute dimension. Pour ce faire, nous avons divisé la tâche initiale en plusieurs sous-tâches et nous avons appris une stratégie pour chaque sous-tâche. Nous avons prouvé qu'une utilisation appropriée des intersections entre les sous-tâches peut significativement accélérer la procédure d'apprentissage.

De plus, nous avons renoué avec les premiers travaux sur les systèmes experts utilisant des raisonnements probabilistes. Cela s'explique par notre application visée, nous nous intéressons aux maladies rares et nous ne pouvons pas travailler sans les connaissances d'experts (ou de la littérature) qui sont généralement exprimées sous forme de probabilités conditionnelles. Nous avons présenté ici une façon d'intégrer ces connaissances expertes aux données cliniques.

Précisons que nous n'avons pas pris en compte ici un certains nombre de difficultés demandant un travail supplémentaire pour être intégrées au cadre proposé. Ainsi, le caractère binaire des réponses de l'utilisateur peut sembler réducteur. Ne faudrait-il pas plutôt considérer un continuum allant, par exemple, de la valeur 0 pour "très anormal" à la valeur 1 signifiant "sain" ? Par ailleurs, comment prendre en compte la possibilité d'une erreur de l'utilisateur ? Nous avons proposé dans [49] de plutôt se focaliser sur le fait qu'un symptôme peut être décrit à divers niveau de granularité. En effet, l'échographiste sera moins susceptible de se tromper s'il peut sélectionner exactement ce qu'il sait. Cela permet également dans une certaine mesure de traiter la question de la nature binaire des symptômes. Un symptôme est fondamentalement présent ou absent, mis à part

quelques cas limites. Le continuum se situe donc davantage sur la nature du symptôme qui est modélisé par l'ontologie, i.e la description des liens entre symptômes sous une forme arborescente. Pour certaines anomalies l'ajout de la notion de gravité associée à l'anomalie devra tout de même être intégrée. Par ailleurs si le nombre de symptômes typiques d'une maladie devient trop grand (ne serait-ce que pour être stocké) il faudra envisager de relâcher notre modèle de dépendance entre symptôme sachant la maladie, une question déjà abordée dans [49].
Terminons en précisant que cet outil d'aide à la décision a été testé en interne, à l'hôpital Necker, sur des données de fœtopathologie et a donné de très bon résultats (plus de 90% de bon diagnostic).

Annexe

Preuve 1. *L'existence d'une solution de* (8) *pour tout* $i \geq 1$ *et* $j \geq 1$ *est une conséquence du fait que la projection sur un espace de dimension finie existe.*
L'unicité de la solution de (8) *pour* $i \neq 1$ *est dû au fait que l'on cherche à minimiser une fonction strictement convexe sous contrainte convexe. Lorsque* $j = 1$ *la fonction à minimiser n'est plus fortement convexe et des contres-exemples peuvent être exhibés.*
Par exemple si $p^{maxent} = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$, $p_n^{emp} = (\frac{1}{2}, 0, \frac{1}{2}, 0)$ *et* $\epsilon_n = \frac{9}{10}$.
Notez que $\|p^{maxent} - p_n^{emp}\|_1 = 1 > \frac{9}{10}$. *Alors en utilisant la proposition 2 nous savons que*

$$\frac{\epsilon_n}{\|p_n^{emp} - p^{maxent}\|_1} p^{maxent} + \left(1 - \frac{\epsilon_n}{\|p_n^{emp} - p^{maxent}\|_1}\right) p_n^{emp} = \left(\frac{11}{40}, \frac{9}{40}, \frac{11}{40}, \frac{9}{40}\right) =: \hat{p}_n^{1,1}$$

est solution. Mais $\tilde{p} = (\frac{10}{40}, \frac{9}{40}, \frac{12}{40}, \frac{9}{40})$ *est également solution. En effet :*

$$\|\tilde{p} - p^{maxent}\|_1 = \frac{1}{10} = \|\hat{p}_n^{1,1} - p^{maxent}\|_1$$

et :

$$\|\tilde{p} - p_n^{emp}\|_1 = \frac{36}{40} = \frac{9}{10}.$$

Preuve 2. *Soit* $\tilde{p} \in \mathcal{C}$ *tel qu'il existe* $\alpha \in [0, 1]$ *où* $\tilde{p} = \alpha p^{maxent} + (1 - \alpha) p_n^{emp}$ *et tel que* $\|\tilde{p} - p_n^{emp}\|_i = \epsilon_n$. *Nous avons alors :*

$$\|\tilde{p} - p_n^{emp}\|_i = \alpha \|p_n^{emp} - p^{maxent}\|_i = \epsilon_n$$

et donc

$$\alpha = \frac{\epsilon_n}{\|p_n^{emp} - p^{maxent}\|_i}.$$

Notons de plus que nous avons l'égalité suivante puisque $\tilde{p}$ *s'écrit sous la forme d'un barycentre :*

$$\|\tilde{p} - p^{maxent}\|_i + \underbrace{\|\tilde{p} - p_n^{emp}\|_i}_{=\epsilon_n} = \|p_n^{emp} - p^{maxent}\|_i$$

Menons un raisonnement par l'absurde. Supponsons qu'il existe $p' \in \mathcal{C}$ *tel que* $\|p_n^{emp} - p'\|_i \leq \epsilon_n$ *et* $\|p^{maxent} - p'\|_i < \|p^{maxent} - \tilde{p}\|_i$.
Nous aurions alors :

$$\begin{aligned}\|p_n^{emp} - p^{maxent}\|_i &\leq \|p' - p^{maxent}\|_i + \|p' - p_n^{emp}\|_i \\ &< \|\tilde{p} - p^{maxent}\|_i + \|p' - p_n^{emp}\|_i \\ &= \|p_n^{emp} - p^{maxent}\|_i - \epsilon_n + \|p' - p_n^{emp}\|_i \\ &\leq \|p_n^{emp} - p^{maxent}\|_i\end{aligned}$$

ce qui mène à la contradiction désirée.

Preuve 3. *Une simple application de l'inégalité triangulaire nous donne :*

$$\|p^\star - \hat{p}_n^{1,1}\|_1 \leq \|p^\star - p_n^{emp}\|_1 + \|p_n^{emp} - \hat{p}_n^{1,1}\|_1$$

Or $\|p_n^{emp} - \hat{p}_n^{1,1}\|_1 \leq \epsilon_n$ *par construction et* $\|p^\star - p_n^{emp}\|_1 \leq \epsilon_n$ *avec probabilité* $1 - \delta$.
Par ailleurs :

$$\|p^\star - \hat{p}_n^{1,1}\|_1 \leq \|p^\star - p^{maxent}\|_1 + \|p^{maxent} - \hat{p}_n^{1,1}\|_1$$

Or en utilisant la définition de $\hat{p}_n^{1,1}$ *et si* $\|p^\star - p_n^{emp}\|_1 \leq \epsilon_n$ *alors :*

$$\|p^{maxent} - \hat{p}_n^{1,1}\|_1 \leq \|p^{maxent} - p^\star\|_1.$$

On en conclut que si $\|p^\star - p_n^{emp}\|_1 \leq \epsilon_n$, *ce qui arrive avec probabilité* $1 - \delta$, *alors :*

$$\|p^\star - \hat{p}_n^{1,1}\|_1 \leq 2 \min\{\epsilon_n, \|p^\star - p^{maxent}\|_1\} \tag{21}$$

Preuve 4. *L'existence et l'unicité de* $\hat{p}_n^L$ *est une conséquence du fait que* $\mathcal{T} = \{p / p \in \mathcal{C} \text{ and } D(p_n^{emp} || p) \leq \epsilon_n\}$ *est un espace convexe. En effet soit* $p, q \in \mathcal{T}$, $\alpha \in [0, 1]$ *alors en utilisant l'inégalité classique "log-sum inequality" nous avons :*

$$\begin{aligned}D(p_n^{emp} || \alpha p + (1 - \alpha) q) &\leq \alpha D(p_n^{emp} || p) + (1 - \alpha) D(p_n^{emp} || q) \\ &\leq \epsilon_n.\end{aligned}$$

Le Lagrangien associé au problème d'optimisation (12) *est donné par :*

$$\begin{aligned}L(p, \lambda, \mu) = &\sum_i p_i^{maxent} \log\left(\frac{p_i^{maxent}}{p_i}\right) \\ &+ \lambda \left(\sum_i p_i^{emp} \log\left(\frac{p_i^{emp}}{p_i}\right) - \epsilon_n\right) \\ &+ \mu \left(\sum_i p_i - 1\right)\end{aligned}$$

En dérivant nous avons pour tout $i \in [1, K]$ *:*

$$\frac{\partial L(p,\lambda,\mu)}{\partial p_i} = -\frac{p_i^{maxent}}{p_i} - \lambda\frac{p_i^{emp}}{p_i} + \mu$$

En résolvant cette dernière expression égale à 0 *et en utilisant le fait que les mesures de probabilités somment à* 1 *nous trouvons :* $\mu = \lambda + 1$. *Nous avons alors pour tout* $i \in [1, K]$ *:*

$$p_i = \frac{1}{1+\lambda}p_i^{maxent} + \frac{\lambda}{1+\lambda}p_i^{emp}$$

Nous savons que $\hat{p}_n^L$ *existe et est unique et par le théorème de Kuhn-Tucker (dont nous satisfaisons les hypothèses puisque nous minimisons une fonction convexe sous des contraintes d'inégalités convexes) nous savons que le minimum du problème d'optimisation* (12) *est atteint pour le point selle du Lagrangien :* $(\tilde{\lambda}, \tilde{p}) = (\tilde{\lambda}, \hat{p}_n^L)$. *Nous pouvons donc écrire :*

$$\hat{p}_n^L = \frac{1}{1+\tilde{\lambda}}p^{maxent} + \frac{\tilde{\lambda}}{1+\tilde{\lambda}}p_n^{emp} \tag{22}$$

Nous ne pouvons pas obtenir une formule exacte pour $\tilde{\lambda}$ *contrairement au cas de* $\hat{p}^{i,i}$. *Cependant nous savons que par construction* $D(p_n^{emp}||\hat{p}_n) \leq \epsilon_n$.
De plus en utilisant la "log-sum inequality" et notre formule d'interpolation (22) *nous avons :*

$$\begin{aligned} D(p_n^{emp}||\hat{p}_n) &= D\left(p_n^{emp}||\frac{1}{1+\tilde{\lambda}}p^{maxent} + \frac{\tilde{\lambda}}{1+\tilde{\lambda}}p_n^{emp}\right) \\ &\leq \frac{1}{1+\tilde{\lambda}}D(p_n^{emp}||p^{maxent}). \end{aligned}$$

Nous avons donc la condition suivante sur $\tilde{\lambda}$ *:*

$$\frac{1}{1+\tilde{\lambda}}D(p_n^{emp}||p^{maxent}) \leq \epsilon_n \Leftrightarrow \tilde{\lambda} \geq \frac{D(p_n^{emp}||p^{maxent})}{\epsilon_n} - 1$$

Preuve 5. *En utilisant la proposition 3 nous avons :*

$$\begin{aligned} D(\hat{p}_n^L||p^\star) &= D\left(\frac{1}{1+\tilde{\lambda}}p^{maxent} + \frac{\tilde{\lambda}}{1+\tilde{\lambda}}p_n^{emp}||p^\star\right) \\ &\leq \frac{1}{1+\tilde{\lambda}}D(p^{maxent}||p^\star) + \frac{\tilde{\lambda}}{1+\tilde{\lambda}}D(p_n^{emp}||p^\star) \\ &= \frac{1}{1+\tilde{\lambda}}\left(D(p^{maxent}||p^\star) - D(p_n^{emp}||p^\star)\right) \\ &+ D(p_n^{emp}||p^\star) \\ &\leq \epsilon_n\left(\frac{D(p^{maxent}||p^\star) - D(p_n^{emp}||p^\star)}{D(p_n^{emp}||p^{maxent})}\right) \\ &+ D(p_n^{emp}||p^\star) \end{aligned}$$

où nous avons utilisé l'inégalité disponible pour $\tilde{\lambda}$ *(proposition 3) dans la dernière inégalité et on obtient le résultat désiré en supposant que* $D(p_n^{emp}||p^\star) \leq \epsilon_n$ *ce qui arrive avec probabilité* $1 - \delta$.

Par ailleurs, notez que :

$$\begin{aligned} \epsilon_n\left(\frac{D(p^{maxent}||p^\star) - D(p_n^{emp}||p^\star)}{D(p_n^{emp}||p^{maxent})}\right) + D(p_n^{emp}||p^\star) \\ \leq D(p^{maxent}||p^\star) \\ \Leftrightarrow \epsilon_n \leq D(p_n^{emp}||p^{maxent}) \end{aligned}$$

Or si $\epsilon_n \geq D(p_n^{emp}||p^{maxent})$ *nous avons par construction que* $\hat{p}_n^L = p^{maxent}$ *et donc* $D(\hat{p}_n^L||p^\star) = D(p^{maxent}||p^\star)$.
Nous pouvons conclure de tout cela que :

$$D(\hat{p}_n^L||p^\star) \leq D(p^{maxent}||p^\star).$$

Références

[1] Jay Mardia and Jiantao Jiao and Ervin Tánczos and Robert D. Nowak and Tsachy Weissman, Concentration Inequalities for the Empirical Distribution, *arXiv*, 2018.

[2] Edwin T. Jaynes, Information Theory and Statistical Mechanics, *American Physical Society*, 1957.

[3] Cover, Thomas M. and Thomas, Joy A., Elements of Information Theory, *Wiley-Interscience*, 2006.

[4] Berger, Adam L. and Pietra, Vincent J. Della and Pietra, Stephen A. Della, A Maximum Entropy Approach to Natural Language Processing, *Comput. Linguist.*, Vol. 22, pp. 39-71, 1996.

[5] Frank Nielsen and Richard Nock, Sided and Symmetrized Bregman Centroids, *IEEE Transactions on Information Theory*, Vol. 55, pp. 2882-2904, 2009.

[6] Kai-Fu Tang, Hao-Cheng Kao, Chun-Nan Chou and Edward Y. Chang, Inquire and Diagnose : Neural Symptom Checking Ensemble using Deep Reinforcement Learning, *NIPS*, 2016.

[7] Hao-Cheng Kao, Kai-Fu Tang and Edward Y. Chang, Context-Aware Symptom Checking for Disease Diagnosis Using Hierarchical Reinforcement Learning, *AAAI*, 2018.

[8] Valentina Bayer Zubek and Thomas G. Dietterich, Integrating Learning from Examples into the Search for Diagnostic Policies, *CoRR*, 2011.

[9] Vijaymohan Konda, Actor-critic algorithms, *NIPS*, 1999.

[10] Richard S. Sutton, David A. McAllester, Satinder P. Singh and Yishay Mansour, Policy Gradient Methods for Reinforcement Learning with Function Approximation, *NIPS*, 1999.

[11] Daniel Hunter, Uncertain Reasoning Using Maximum Entropy Inference, *Proceedings of the First Conference on Uncertainty in Artificial Intelligence*, 1985.

[12] John E. Shore, Relative Entropy, Probabilistic Inference and AI, *CoRR*, 2013.

[13] John W. Miller and Rodney M. Goodman, A Polynomial Time Algorithm for Finding Bayesian Probabilities from Marginal Constraints, *CoRR*, 2013.

[14] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel and Demis Hassabis, Mastering the game of Go with deep neural networks and tree search, *Nature*, Vol. 529, pp. 484-489, 2016.

[15] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, L Robert Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel and Demis Hassabis, Mastering the game of Go without human knowledge, *Nature*, Vol. 550, pp. 354-359, 2017.

[16] Judea Pearl, Probabilistic reasoning in intelligent systems - networks of plausible inference, *Morgan Kaufmann series in representation and reasoning*, 1989.

[17] Radim Jirousek, A survey of methods used in probabilistic expert systems for knowledge integration, *Knowl.-Based Syst.*, Vol. 3, pp. 7-12, 1990.

[18] Yvonne M. M. Bishop, Stephen E. Fienberg, Paul W. Holl, Richard J. Light, Frederick Mosteller and Peter B. Imrey, Discrete Multivariate Analysis : Theory and Practice, 1975.

[19] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau and Aviv Tamar, Bayesian Reinforcement Learning : A Survey, *CoRR.*, 2016.

[20] Sebastian Köhler and al., The Human Phenotype Ontology in 2017, *Nucleic Acids Research*, 2017.

[21] Sutton, Richard S. and Barto, Andrew G., Introduction to Reinforcement Learning, *MIT Press*, 2018.

[22] Deming, W. Edwards and Stephan, Frederick F., On a Least Squares Adjustment of a Sampled Frequency Table When the Expected Marginal Totals are Known, *Ann. Math. Statist.*, Vol. 11, pp. 427–444, 1940.

[23] Eugene Charniak, The Bayesian Basis of Common Sense Medical Diagnosis, *AAAI*, 1983.

[24] Andrew G. Barto, Steven J. Bradtke and Satinder P. Singh, Learning to Act Using Real-Time Dynamic Programming, *Artif. Intell.*, Vol. 72, pp. 81-138, 1995.

[25] Leo Breiman, Joseph H Friedman, R. A. Olshen and C. J. Stone, Classification and Regression Trees, *Chapman and Hall/CRC*, 1984.

[26] Chollet, François and others, Keras, *GitHub*, 2015.

[27] Nicolas Heess, David Silver and Yee Whye Teh, Actor-Critic Reinforcement Learning with Energy-Based Policies, *Proceedings of Machine Learning Research*, Vol. 24, pp. 45–58, 2013.

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra and Martin A. Riedmiller, Playing Atari with Deep Reinforcement Learning, *Nature*, 2013.

[29] Bertsimas, Dimitris and Dunn, Jack, Optimal Classification Trees, *Mach. Learn.*, Vol. 106, pp. 1039–1082, 2017.

[30] Berliner, Hans J., Some Necessary Conditions for a Master Chess Program, *IJCAI'73*, pp. 77–85, 1973.

[31] Yu-Shao Peng, Kai-Fu Tang, Hsuan-Tien Lin and Edward Chang, REFUEL : Exploring Sparse Features in Deep Reinforcement Learning for Fast Disease Diagnosis, *Advances in Neural Information Processing Systems 31*, pp. 7333–7342, 2018.

[32] Sebastian Köhler, Marcel H. Schulz, Peter Krawitz, Sebastian Bauer, Sandra Dölken, Claus E. Ott, Christine Mundlos, Denise Horn, Stefan Mundlos and Peter N. Robinson, Clinical Diagnostics in Human Genetics with Semantic Similarity Searches in Ontologies, *The American Journal of Human Genetics*, Vol. 85, pp. 457 - 464, 2009.

[33] Rémi Besson and Erwan Le Pennec and Stéphanie Allassonnière and Julien Stirnemann and Emmanuel Spaggiari and Antoine Neuraz, Symptom_Checker, *GitHub repository*, 2019.

[34] Quinlan, J. R., Induction of Decision Trees, *Mach. Learn.*, Vol. 1, pp. 81–106, 2019.

[35] Artemij Amiranashvili, Alexey Dosovitskiy, Vladlen Koltun and Thomas Brox, TD or not TD : Analyzing the Role of Temporal Differencing in Deep Reinforcement Learning, *Mach. Learn.*, 2018.

[36] Solomon, B. D., VACTERL/VATER Association, *Orphanet J Rare Dis.*, 2011.

[37] Uzawa, H., Iterative methods for concave programming, *Studies in Linear and Nonlinear Programming*, 1958.

[38] Eric Wiewiora, Potential-Based Shaping and Q-Value Initialization are Equivalent, *CoRR*, 2011.

[39] Ronald J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, *Machine Learning*, Vol. 8, pp. 229-256, 1992.

[40] Johan Barthelemy and Philippe L. Toint, Synthetic Population Generation Without a Sample, *Transportation Science*, Vol. 47, pp. 266-279, 2013.

[41] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver and Koray Kavukcuoglu, Asynchronous Methods for Deep Reinforcement Learning, *ICML*, 2016.

[42] C T Ireland and Solomon Kullback, Contingency tables with given marginals, *Biometrika*, Vol. 55 1, pp. 179-88, 1968.

[43] R Nj Veldhuis, The centroid of the symmetrical Kullback-Leibler distance, *IEEE Signal Processing Letters*, Vol. 9, pp. 96-99, 2002.

[44] Frank Nielsen, The centroid of the Jeffreys Centroids : A Closed-Form Expression for Positive Histograms and a Guaranteed Tight Approximation for Frequency Histograms, *IEEE Signal Processing Letters*, Vol. 20, pp. 657-660, 2013.

[45] Martin Adamcik, Collective Reasoning under Uncertainty and Inconsistency, *Doctoral thesis, Manchester Institute for Mathematical Sciences*, 2014.

[46] Imre Csiszár and Frantisek Matús, Information projections revisited, *IEEE Trans. Information Theory*, Vol. 49, pp. 1474–1490, 2003.

[47] Andrew Y. Ng, Daishi Harada and Stuart J. Russell, Policy Invariance Under Reward Transformations : Theory and Application to Reward Shaping, *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 278–287, 1999.

[48] Andrew Gelman, John B. Carlin, Hal S. Stern, Donald B. Rubin, Bayesian Data Analysis, *Chapman and Hall/CRC*, 2004.

[49] Rémi Besson, Erwan Le Pennec, Stéphanie Allassonnière, Julien Stirnemann, Emmanuel Spaggiari and Antoine Neuraz, A Model-Based Reinforcement Learning Approach for a Rare Disease Diagnostic Task, *arxiv*, 2018.

[50] Pierre-Simon de Laplace, Mémoire sur la probabilité des causes par les évènements, *Mémoires de mathématique et de physique présentés à l'Académie royale des sciences par divers sçavans et lus dans les assemblées*, Tome sixième, 1774.

[51] David Heckerman, Dan Geiger, David Chickering, Learning Bayesian Networks : The Combination of Knowledge and Statistical Data, *IEEE Trans. on Knowl. and Data Eng.*, Vol. 20, pp. 197–243, 1995.

[52] Daphne Koller, Nir Friedman, Probabilistic Graphical Models : Principles and Techniques - Adaptive Computation and Machine Learning, *The MIT Press*, 2009.

[53] David Spiegelhalter, Philip Dawid, Steffen Lauritzen, Robert Cowell, Bayesian Analysis in Expert Systems, *Statistical Science*, Vol. 8, pp. 219–247, 1993.

[54] Anthony Costa Constantinou, Norman Fenton, Martin Neil, Integrating expert knowledge with data in Bayesian networks : Preserving data-driven expectations when the expert variables remain unobserved, *Expert Systems with Applications*, Vol. 56, pp. 197–208, 2016.

[55] Yun Zhou, Norman Fenton, Cheng Zhu, An empirical study of Bayesian network parameter learning with monotonic influence constraints, *Decision Support Systems*, Vol. 87, pp. 69–79, 2016.

Une première approche pour la résolution d'un jeu de sécurité par des MDP augmentés

Romain Châtel[1] Abdel-Illah Mouaddib[1]

[1] GREYC, Université de Caen Normandie

romain.chatel@unicaen.fr

Résumé

Nous proposons une première approche théorique et expérimentale pour la résolution d'un jeu de sécurité. Cette approche se restreint à l'utilisation de MDP augmentés pour le calcul d'une politique optimale d'un attaquant faisant face à un défenseur patrouillant sur une grille. Trois formalisations sont proposées afin d'approcher pas à pas le problème. Les résultats obtenus suite aux simulations des deux premières approches permettent d'envisager de raffiner le modèle et ainsi de combler au mieux ses faiblesses, notamment les difficultés liées au passage à l'échelle.

Mots Clefs

Théorie des jeux, MDP augmentés, Jeux de sécurité, Jeux de Stackelberg.

Abstract

We propose a first theoretical and experimental approach for solving a Stackelberg security game. This approach is restricted to the use of augmented MDPs to compute an optimal policy for an attacker facing a defender patrolling on a grid. Three modelizations are proposed to approach the problem step by step. The results coming from the implementation of the first two models allow to consider refining the model and thus filling its weaknesses, in particular its difficulties to scale to larger problems.

Keywords

Game theory, Augmented MDPs, Security games, Stackelberg games.

1 Introduction

Les jeux de sécurité fournissent un cadre formel ayant récemment émergé dans le domaine des systèmes multi-agents. Ils permettent de modéliser les interactions entre deux types de joueurs, défenseurs et attaquants, et s'appliquent à trouver des stratégies pour les forces de sécurité (défenseurs) aussi bien civile que militaire. La plus grande partie des travaux s'est concentrée sur l'utilisation des jeux de Stackelberg [1] où la défense joue en premier puis la partie adverse choisit un coup après avoir pris connaissance de la stratégie défensive. La résolution de ces jeux du point de vue défensif s'effectue en cherchant un équilibre de Stackelberg fort qui suppose le choix d'une stratégie mixte optimale sachant que l'attaquant l'observera et choisira en réponse une stratégie optimale.

Ces modèles ont notamment été étudiés et résolus avec succès dans les domaines de la sécurité des ports, aéroports, métros, pêcheries, systèmes d'information dans le but d'optimiser les stratégies de gestion du risque d'attaque des équipes de sécurité [2, 3, 4, 5, 6].

Cet article propose une application de ce formalisme à la surveillance militaire, en prenant le point de vue de l'attaquant jusqu'alors moins étudié. En effet, depuis la seconde guerre mondiale, les drones à usage militaire sont intensivement développés par les différentes armées du monde et se voient attribuer des missions de logistique, surveillance, reconnaissance, acquisition de cible, voire d'attaque.

Les modélisations qui suivent adressent le jeu de sécurité suivant : sur une grille de taille $N \times M$, un drone patrouilleur (le défenseur) surveille. Un second drone intrus (l'attaquant) tente de rejoindre une case sur la grille (la cible) sans se faire repérer par le premier. Le défenseur est doté d'un champ de perception dont l'intensité décroît avec la distance. De plus, les agents sont soumis au même environnement stochastique modélisant, par exemple, les dérives dues au vent dans le cas de drones volants. Le but du jeu est de trouver une stratégie pour l'attaquant permettant d'atteindre la cible tout en minimisant le risque de se faire repérer par le patrouilleur. Dans ce travail, nous nous limitons à l'étude et l'utilisation de MDP augmentés afin de bénéficier de la simplicité du formalisme des MDP tout en permettant de l'étendre au cadre multi-agent.

Dans le but d'aborder ce problème pas à pas, celui-ci a été subdivisé en sous-problèmes de difficulté croissante. Dans un premier temps, le défenseur patrouille selon un parcours déterministe sans être soumis à la stochasticité de l'environnement. Ce parcours étant connu de l'intrus. Une seconde variante du problème considère que le patrouilleur est soumis aux aléas de l'environnement et utilise donc une politique de patrouille connue de l'attaquant. Pour ces deux premiers cas, l'état du drone de surveillance est connu à chaque instant par l'attaquant. Enfin, le problème général étend le second : connaissant un ensemble de politiques

possibles suivies par l'agent patrouilleur et disposant d'un oracle fournissant des observations sur sa position au cours du temps, l'intrus doit trouver une politique optimale permettant de résoudre le problème.

2 Modélisation

2.1 Formalisation du risque

La première étape pour la formalisation de ces problèmes est la définition du risque pris par l'intrus vis à vis du patrouilleur. Nous définissons ici ce risque comme l'intensité avec laquelle l'attaquant est perçu par le défenseur. Comme celui-ci dépend de la position de l'intrus par rapport au champ de perception du drone de surveillance, il est raisonnable de le définir dans un repère centré sur le défenseur comme une fonction $D : C \times R \mapsto [0 \dots 1]$, avec C l'ensemble des M colonnes de la grille et R l'ensemble des N lignes de la grille. L'ensemble d'arrivée est choisi de manière à pouvoir être commensurable avec la future fonction de récompense que le risque viendra moduler. De plus, ce risque, centré sur l'agent patrouilleur, est décroissant selon x et y à mesure de l'éloignement à celui-ci. Ces contraintes ont été modélisées à partir de mélange de fonctions logistiques dont la forme est donnée par les formules suivantes :

$$f(t, a, r) = \frac{1}{1 + ae^{-rt}}$$

$$g(t, a, r, L) = \max(f(t, a, r) - f(L, a, r), 0)$$

$$h(t, a, r, L) = \frac{g(t, a, r, L)}{g(0, a, r, L)}$$

$$D_x(x, a_x, r_x, L_x) = h(abs(x), a_x, r_x, L_x)$$

$$D_y(y, a_y, r_y, L_y) = \begin{cases} 0 & \text{si } y \leq 0 \\ h(y, a_y, r_y, L_y) & \text{sinon} \end{cases}$$

$$D_{xy} = \max(D_x + D_y - 1, 0)$$

Les paramètres a et r permettent de modifier les courbures sur x et y tandis que les paramètres L déterminent la profondeur du champ de perception.

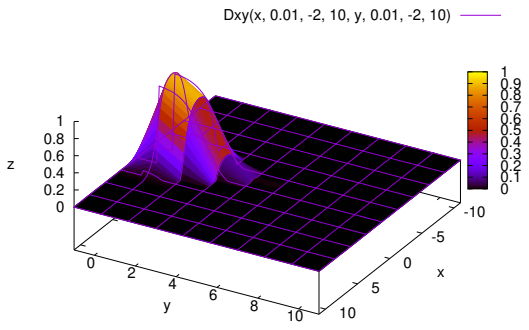


FIGURE 1 – Fonction de risque : mélange de fonctions logistiques

2.2 Formalisation de la récompense

Afin d'inciter l'agent à atteindre la cible, la fonction de récompense R lui délivre une récompense valant 1 si celui ci a atteint la cible, et 0 partout ailleurs :

$$R(x, y) = \begin{cases} 1 & \text{si } (x, y) == (target_x, target_y) \\ 0 & \text{sinon} \end{cases}$$

2.3 Description des actions

Par souci de simplification, l'espace d'actions A est choisi identique pour les deux drones et restera constant dans les trois formalisations qui vont suivre. Chacun des drones peut avancer d'une case dans une des huit directions cardinales :

$$A = \{\text{GO-NORTH, GO-NORTH-EAST,}$$
$$\text{GO-EAST, GO-SOUTH-EAST,}$$
$$\text{GO-SOUTH, GO-SOUTH-WEST,}$$
$$\text{GO-WEST, GO-NORTH-WEST}\}$$

Les possibilités d'actions sont cependant conditionnées à l'orientation des drones, afin de contraindre à des virages moins abrupts, selon le tableau suivant :

Orientation	Actions possibles
NORTH	{GO-NORTH, GO-NORTH-EAST, GO-NORTH-WEST}
NORTH-EAST	{GO-NORTH-EAST, GO-EAST, GO-NORTH}
EAST	{GO-EAST, GO-SOUTH-EAST, GO-NORTH-EAST}
SOUTH-EAST	{GO-SOUTH-EAST, GO-SOUTH, GO-EAST}
SOUTH	{GO-SOUTH, GO-SOUTH-WEST, GO-SOUTH-EAST}
SOUTH-WEST	{GO-SOUTH-WEST, GO-WEST , GO-SOUTH}
WEST	{GO-WEST, GO-NORTH-WEST, GO-SOUTH-WEST}
NORTH-WEST	{GO-NORTH-WEST, GO-NORTH, GO-WEST}

2.4 Patrouilleur non soumis à l'aléa de l'environnement

Le premier problème peut alors être modélisé comme suit :

Le patrouilleur. L'agent patrouilleur est doté d'un espace d'états $S^P = C \times R \times O$ ainsi que d'un automate déterministe de patrouille $next$ tel que $\forall t$, $p^t \in S^P$, $p^{t+1} = next(p^t)$. O désigne ici l'ensemble des huit orientations possibles de l'agent.

L'intrus. L'intrus quant à lui est décrit par un MDP augmenté $\langle S^I, S^P, A, T^I, U^I, next \rangle$, où $S^I = C \times R \times O$ est son espace d'états, S^P est l'espace d'états du patrouilleur, A est l'espace d'actions, T^I est son modèle de transition et $U^I : S^I \times S^P \mapsto \mathbb{R}$ est sa fonction d'utilité telle que :

$$U^I_{ip} = \beta\, R_i - (1 - \beta)\, D_{ip},$$

où β est un facteur compris entre 0 et 1 modulant sa prise de risque, $R_i = R(x_i, y_i)$ représente sa récompense et $D_{ip} = D(x_i, y_i, x_p, y_p, o_p)$ représente le risque pris par l'intrus. Une translation et une rotation sont nécessaires afin d'utiliser la fonction de risque définie précédemment. Enfin, $next$ est l'automate déterministe suivi par le patrouilleur.

Sa fonction de valeur pour le calcul d'une politique sur les états joints $\pi^I : S^I \times S^P \mapsto A$ est alors :

$$v_{ip}^{\pi^I} = U_{ip}^I + \gamma^I \max_{a^I \in A} \sum_{i'} T_{ii'}^{a^I} \, v_{i' next(p)}^{\pi^I}$$

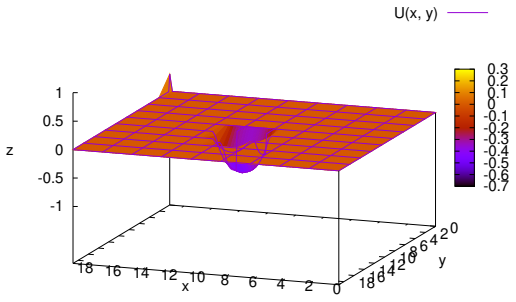


FIGURE 2 – Fonction d'utilité du problème 1, $\beta = 0.3$, cible en $(19,\ 0)$, patrouilleur en $(10,\ 10)$.

2.5 Patrouilleur soumis à l'aléa de l'environnement 1/2

Dans ce second problème, les transitions du drone de surveillance deviennent stochastiques. Il suit alors une politique π^P.
Dans un premier temps, il est possible de formaliser le problème à l'image du précèdent :

Le patrouilleur. Son automate $next$ est remplacé par la politique π^P et un modèle de transition T^P.

L'intrus. Cet agent se trouve alors décrit par le MDP augmenté $\langle S^I, S^P, A, T^I, T^P, U^I, \pi^P \rangle$. L'équation de Bellman servant à sa résolution devient :

$$v_{ip}^{\pi^I} = U_{ip}^I + \gamma^I \max_{a^I \in A} \sum_{i'} \sum_{p'} T_{pp'}^{\pi_p^P} \, T_{ii'}^{a^I} \, v_{i'p'}^{\pi^I}$$

2.6 Patrouilleur soumis à l'aléa de l'environnement 2/2

Une modélisation alternative consiste à considérer, du point de vue de l'intrus, que le patrouilleur poursuit le but de le mettre en danger. Cela donne lieu à un jeu stochastique à somme générale où l'attaquant est récompensé comme précédemment mais où le défenseur se voit également attribuer une récompense : le risque qu'il fait courir à l'attaquant. C'est un point de vue « paranoïaque » sur la situation qu'adopte l'intrus car la politique suivie par le patrouilleur peut ne pas prendre en compte la présence de l'intrus. Nous proposons de représenter ce jeu par deux MDP couplés l'un représentant le comportement du défenseur du point de vue « paranoïaque » de l'attaquant, le second représentant le comportement de l'attaquant.

Le patrouilleur. Il est décrit par le MDP augmenté $\langle S^I, S^P, A, T^I, T^P, R^P, \pi^P \rangle$ où S^I est l'espace d'états de l'intrus, T^I est le modèle de transition de l'intrus, $R_{pi}^P = D_{pi}$ est la fonction de récompense du patrouilleur.
Ce MDP est utilisé pour calculer la valeur d'un couple d'états (p, i) associé à la politique du drone de surveillance :

$$v_{pi}^{\pi^P} = R_{pi}^P + \gamma^P \min_{a^I \in A} \sum_{p'} \sum_{i'} T_{ii'}^{a^I} \, T_{pp'}^{\pi_p^P} \, v_{p'i'}^{\pi^P}$$

Comme ce sont ici les actions de l'intrus qui sont considérées, l'optimisation utilisée est une minimisation. En effet, de son point de vue, l'intrus fait tout pour minimiser le danger.

L'intrus. L'intrus se retrouve alors modélisé par le MDP $\langle S^I, S^P, A, T^I, T^P, R^I, \pi^P, v^{\pi^P} \rangle$. Celui-ci peut être résolu en utilisant la fonction de valeur sur les états joints suivante :

$$v_{ip}^{\pi^I} = \beta \Big[R_i^I + \gamma^I \max_{a^I \in A} \sum_{i'} \sum_{p'} T_{pp'}^{\pi_p^P} \, T_{ii'}^{a^I} \, v_{i'p'}^{\pi^I} \Big] - (1-\beta) v_{pi}^{\pi^P}$$

Bien que ce jeu soit à somme générale, cette proposition de résolution s'approche du *minimax* traditionnellement utilisé pour résoudre les jeux à somme nulle. En effet, chaque camp tente de maximiser son gain tout en prenant en compte la perte maximale occasionnée par la partie adverse.

2.7 Observabilité partielle sur l'état et la politique du patrouilleur

Tandis que précédemment, l'intrus connaissait à chaque instant l'état de son adversaire, cette fois il n'a accès qu'à des observations sur celui-ci. De plus l'adversaire est susceptible d'utiliser une politique choisie parmi un ensemble

$$\Pi^P = \{\pi^{P1}, \ldots, \pi^{Pn}\}$$

La modélisation qui suit est basée sur le principe *pseudo-minimax* de la précédente.

Le patrouilleur. Il suit le MDP augmenté $\langle S^I, S^P, A, T^I, T^P, R^P, \Pi^P \rangle$, associé à un système d'équations de Bellman donné par :

$$\forall j \in [1 \ldots n], \quad v_{pi}^{\pi^{Pj}} = R_{pi}^P + \gamma^{\pi^{Pj}} \min_{a^I \in A} \sum_{p'} \sum_{i'} T_{ii'}^{a^I} \, T_{pp'}^{\pi^{Pj}} \, v_{p'i'}^{\pi^{Pj}}$$

L'intrus. L'incertitude sur l'état du drone de surveillance est prise en compte par l'intrus en maintenant un état de croyance par politique π^{Pj} suivie par le patrouilleur $b^{s^{\pi^{Pj}}} : S^P \mapsto [0 \ldots 1]$ mis à jour à chaque pas de temps t avec la fonction de mise à jour traditionnelle des POMDP. L'incertitude sur la politique suivie par le patrouilleur fait également l'objet d'un état de croyance $b^\pi : \Pi^P \mapsto [0 \ldots 1]$ mis

à jour grâce à l'algorithme *Forward* ou *Viterbi* d'un HMM $H^P = \langle \Pi^P, \Omega, \nu, \Lambda, B \rangle$ tel que Ω est l'ensemble fini des observations sur l'état du patrouilleur, $\nu : \Pi^P \mapsto [0 \dots 1]$ est la loi de distribution initiale sur les politiques suivies par le patrouilleur, $\Lambda : \Pi^P \times \Pi^P \mapsto [0 \dots 1]$ est sa loi de transition et $B : \Omega \times \Pi^P \mapsto [0 \dots 1]$ est sa fonction d'observation.
L'intrus est donc modélisé par le BMDP augmenté $\langle S^I, S^P, A, T^I, T^P, H^P, R^I, \{b^{s^{\pi^{P1}}} \dots b^{s^{\pi^{Pn}}}\}, b^\pi, \{v^{\pi^{P1}}, \dots, v^{\pi^{Pn}}\} \rangle$ dont la fonction de valeur s'écrit :

$$v^{\pi^I}_{ib^s b^\pi} = \beta \Big[R^I_i + \gamma^I \max_{a^I \in A} \sum_{j=1}^{n} \sum_{p} \sum_{i'} \sum_{p'} b^{s^{\pi^{Pj}}}_{p} \, b^{\pi}_{\pi^{Pj}} \, T^{\pi^{Pj}}_{pp'} \, T^{a^I}_{ii'} \, v^{\pi^I}_{i'p'} \Big] - (1-\beta) \sum_{j=1}^{n} \sum_{p} b^{s^{\pi^{Pj}}}_{p} \, b^{\pi}_{\pi^{Pj}} \, v^{\pi^{Pj}}_{pi}$$

3 Premiers résultats

Les politiques de l'attaquant sont calculés par *Value iteration* sur des grilles de taille $N = M = n$ avec comme critère d'arrêt $||V_{n+1} - V_n|| < \epsilon$. Seules les restrictions du problème à l'observabilité complète sur l'état et la politique du patrouilleur ont pour l'instant donné lieu à des expérimentations. L'évaluation des différentes méthodes de résolution proposées porte sur la meilleure politique obtenue pour chacune d'entre elles par recherche (*gridSearch*) des meilleurs hyper-paramètres en regard d'un critère intéressant le cadre applicatif : %interceptions $\succ$ %risque $\succ$ %target $\succ$ temps de calcul. Les valeurs pour le calcul de ce critère sont obtenues par simulation du comportement induit par chaque politique sur la grille. Les valeurs %interception, %risque, %target désignent respectivement le nombre d'interceptions par le patrouilleur (les deux drones sont sur la même case), le nombre de fois que l'intrus a été perçu ainsi que le nombre de fois qu'il a atteint la cible, chacune de ces quantités étant divisée par le nombre de pas de temps de la simulation. Pour les simulations, le patrouilleur est doté d'un champ de perception lui permettant de « voir » jusqu'à 2 cases sur les côtés, 3 cases en avant et 1 case en arrière.

3.1 Patrouilleur non soumis à l'aléa de l'environnement

Les politiques générées sont misent en simulation en affectant au défenseur une routine consistant à parcourir la grille sur la ligne médiane alternativement de gauche à droite et de droite à gauche. La taille de l'espace d'états joints est alors de $8n^2 \times 2n$, le premier terme décrivant la taille de l'espace d'états de l'attaquant, le second celui du défenseur. En considérant un horizon maximal $maxiter$, la complexité de l'algorithme *Value Iteration* utilisé est donnée par $\mathcal{O}(16n^3 \times 3 \times 3 \times maxiter)$.
Deux types de simulations sont effectués : le premier dans lequel les agents décident simultanément (cas synchrone), le second, reflétant une situation plus réaliste dans lequel les prises de décision sont asynchrones.

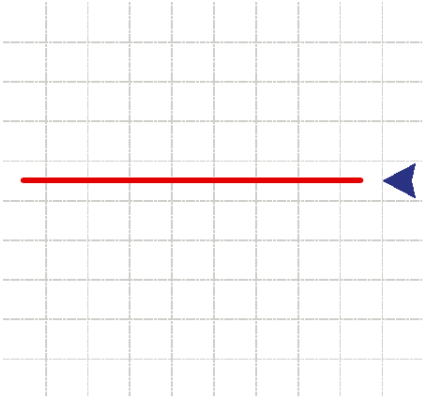


FIGURE 3 – Parcours sur la ligne médiane du patrouilleur

Les figures 4 et 5 montrent l'évolution des différentes métriques d'évaluation en fonction de n. Le cas synchrone témoigne du comportement attendu du modèle dès que l'attaquant a suffisamment de place pour évoluer autour du défenseur ($n \geq 8$). Pour $n > 15$ le patrouilleur n'est un obstacle que sur les premières phases d'avancée vers la cible ce qui explique l'augmentation notable du nombre de passages sur celle-ci. Les traces d'exécution ont quant à elles montré que les trajectoires de l'intrus ont tendance à être tangentes à la zone de perception du patrouilleur, l'optimisation suivie par le MDP de l'intrus l'entraînant à choisir les chemins les plus courts vers la cible. Cela a conduit à rajouter des marges de sécurité sur x et y (2 unités sur chaque axe) pour le cas asynchrone afin d'obtenir les résultats présentés en figure 5. Ces marges permettent d'atteindre des taux d'interception et de prise de risque intéressants pour notre cadre applicatif : respectivement $< 0.004\%$ et $< 0.05\%$ pour $n > 10$. En revanche, le risque moyen par prise de risque reste élevé ($\sim 45\%$) malgré des tailles de grille permettant à l'attaquant d'éviter le défenseur.

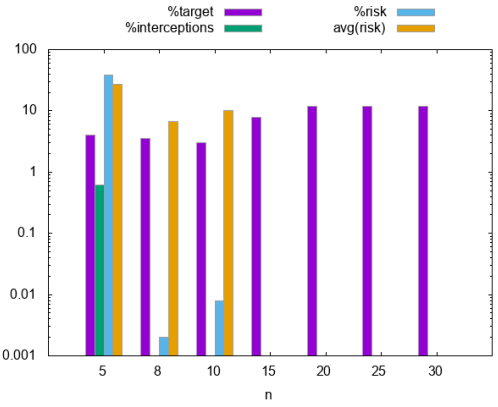


FIGURE 4 – Modèle 1 synchrone, temps d'exécution : 30min

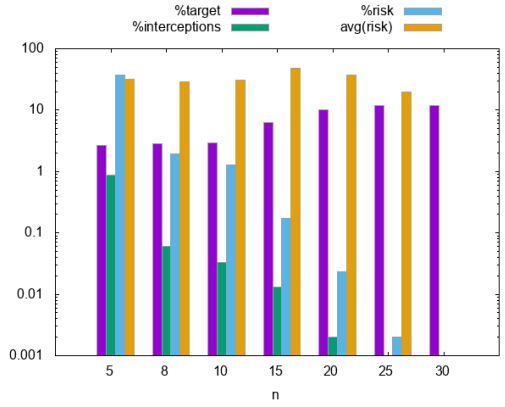


FIGURE 5 – Modèle 1 asynchrone avec marges, temps d'exécution : 30min

3.2 Patrouilleur soumis à l'aléa de l'environnement

Les politiques de ce second problème sont calculées en affectant au patrouilleur une politique consistant à parcourir la grille en zigzags (parcours en boustrophédon). Son espace d'états contient, en plus des variables inhérentes à la position et l'orientation du drone, une variable AR à valeur dans R et désignant la ligne courante à compléter ainsi qu'une variable UD à valeur dans $[0, 1]$ permettant d'alterner entre un parcours vers le haut et un parcours vers le bas de la grille.

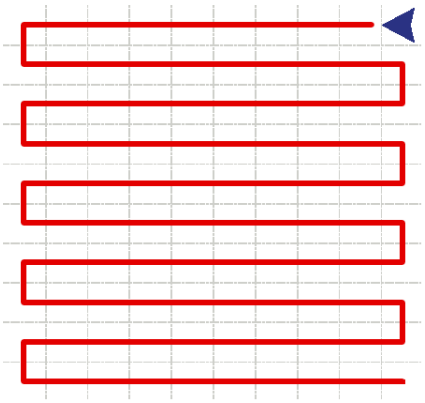


FIGURE 6 – Parcours en boustrophédon du patrouilleur

Version 1/2. Cette première version est caractérisée par un espace d'états joints de taille $8n^2 \times 16n^3$. La complexité de l'algorithme *Value iteration* utilisé est donnée par $\mathcal{O}(128n^5 \times 3 \times 3 \times 3 \times maxiter)$. Ces grandeurs montrent sans ambiguité l'incapacité du modèle à passer à l'échelle, les expérimentations seront donc limitées à de petites grilles ($n \in \{5, 8, 10\}$). Elles permettent toutefois de mettre en évidence les comportements limites de l'attaquant lorsqu'il est dans l'environnement immédiat du défenseur. Les figures 7 et 8 font état de performances similaires au cas déterministe asynchrone pour $n = 10$.

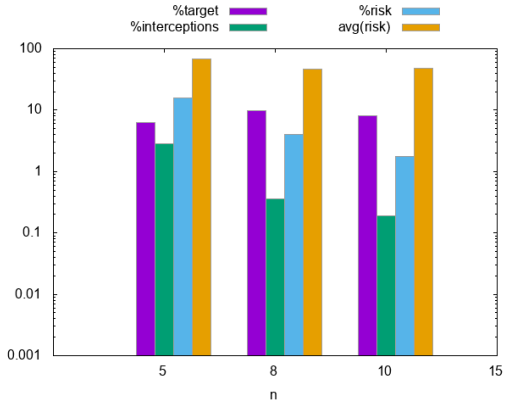


FIGURE 7 – Modèle 2 synchrone version 1, temps d'exécution : 30min

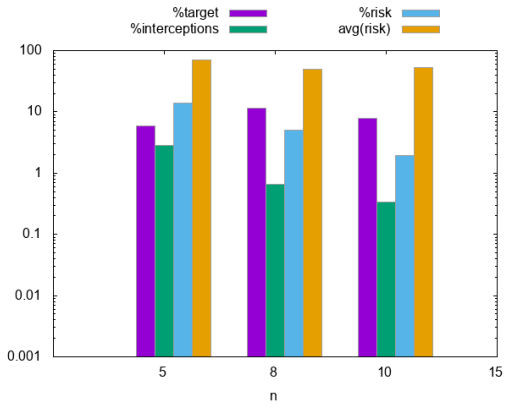


FIGURE 8 – Modèle 2 asynchrone avec marges version 1, temps d'exécution : 30min

Version 2/2. À une constante multiplicative près, la complexité de ce modèle est identique à la précédente, les expérimentations seront donc restreintes de la même manière. Dans l'ensemble, le modèle se montre moins performant que le précèdent avec des taux d'interception et surtout de prise de risque plus élevés aussi bien dans le cas synchrone qu'asynchrone (figures 9 et 10).

4 Conclusion

Nous avons proposé une première approche pour la résolution d'un jeu de patrouille par des MDP augmentés à partir de trois modélisations de complexité croissante. Après implémentation des deux premières modélisations, les résultats montrent des taux d'interception et de prise de risque acceptables pour notre cadre applicatif. Quelques réserves subsistent toutefois quant aux résultats de la seconde version du deuxième modèle : intuitivement, il semblait que celle-ci exhiberait un taux de prise de risque inférieur à la première. Ce point nécessite donc de plus amples inves-

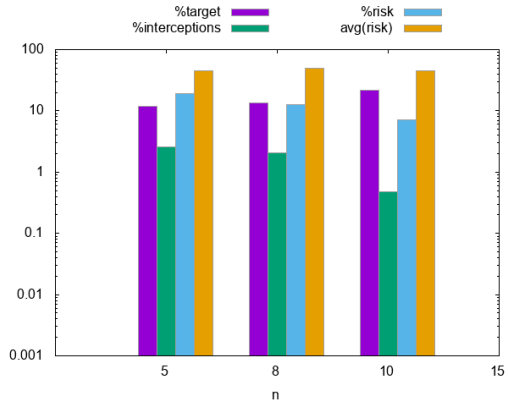


FIGURE 9 – Modèle 2 synchrone version 2, temps d'exécution : 30min

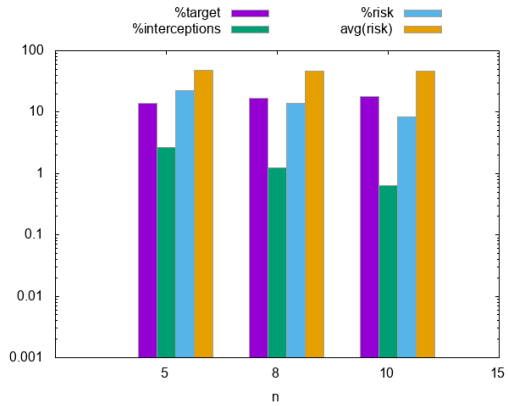


FIGURE 10 – Modèle 2 asynchrone avec marges version 2, temps d'exécution : 30min

tigations. Le passage à l'échelle du modèle est également à améliorer en utilisant des techniques de décompositions et/ou d'approximations. Enfin, il est prévu d'utiliser une approche *Best-response* parallélisée dans une nouvelle formulation de la troisième modélisation.

Références

[1] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordóñez, and M. Tambe, "Computing optimal randomized resource allocations for massive security games," vol. 2, pp. 689–696, AMAAS-2009 Conf, 01 2009.

[2] E. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, and G. Meyer, "Protect : A deployed game theoretic system to protect the ports of the United States," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 13–20, International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[3] J. Pita, M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus, "Deployed ARMOR protection : the application of a game theoretic model for security at the Los Angeles International Airport," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems : industrial track*, pp. 125–132, International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[4] Z. Yin, A. X. Jiang, M. P. Johnson, C. Kiekintveld, K. Leyton-Brown, T. Sandholm, M. Tambe, and J. P. Sullivan, "Trusts : Scheduling randomized patrols for fare inspection in transit systems," in *Twenty-Fourth IAAI Conference*, 2012.

[5] W. Haskell, D. Kar, F. Fang, M. Tambe, S. Cheung, and E. Denicola, "Robust Protection of Fisheries with COmPASS," in *Twenty-Sixth IAAI Conference*, Jun 2014.

[6] S. M. MC Carthy, *Hierarchical planning in security games ; A game theoretic approach to strategic, tactical and operational decision making.* PhD thesis, University of Southern California, 2018.

Temporal Epistemic Gossip Problems

Martin C. Cooper, Andreas Herzig, Frédéric Maris, and Julien Vianey

IRIT, CNRS, Univ. Toulouse, 31062 Toulouse Cedex 9, France
firstname.surname@irit.fr

Abstract. Gossip problems are planning problems where several agents have to share information ('secrets') by means of phone calls between two agents. In epistemic gossip problems the goal can be to achieve higher-order knowledge, i.e., knowledge about other agents' knowledge; to that end, in a call agents communicate not only secrets, but also agents' knowledge of secrets, agents' knowledge about other agents' knowledge about secrets, etc. Temporal epistemic gossip problems moreover impose constraints on the times of calls. These constraints are of two kinds: either they stipulate that a call between two agents must necessarily be made at some time point, or they stipulate that a call can be made within some possible (set of) interval(s). In the non-temporal version, calls between two agents are either always possible or impossible. We investigate the complexity of the plan existence problem in this general setting. Concerning the upper bound, we prove that it is in NP in the general case, and that it is in P when the problem is non-temporal and the goal is a positive epistemic formula. As for the lower bound, we prove NP-completeness for two fragments: problems with possibly negative goals even in the non-temporal case, and problems with temporal constraints even if the goal is a set of positive atoms.

Keywords: epistemic planning, temporal planning, gossip problem, complexity, epistemic logic

1 Introduction

The epistemic gossip problem defined in [22, 12, 11] is a problem in which n agents each have a secret, denoted by s_i for agent i. Agents communicate by calling other agents: during a call the two agents share all their knowledge, not only the secrets they have learned but also epistemic information concerning which agents know which information. The goal of this problem concerns agents' knowledge about other agents' secrets at various epistemic depths. For example, the goal may be shared knowledge of depth 2: all agents know that all agents know all secrets. Such goals can be described as logical formulas in Dynamic Epistemic Logic of Propositional Assignments and Observation DEL-PAO [17, 18]. This generalises the well-known gossip problem [16, 12] which has recently been analysed in the framework of dynamic epistemic logics [6]. We consider it to be an exemplary case of epistemic planning [21, 23, 20, 7, 8] in which communication actions are used to spread knowledge among a network of agents.

Here, we add to this problem the ability to limit any communication to a set of instants (such as an interval during which the two agents involved in the call are both available). For an example, one can think of satellites on different orbits which can only communicate when they 'see' each other. In a more down-to-earth example, the interval during which a mobile communication is available is often limited by the charge capacity of its battery. Another variant we study is when certain calls must occur at given instants (for example for maintenance or security reasons).

What follows applies to either one-way or two-way communication and to either sequential or parallel communication. During a one-way call (such as a letter or email) information only passes in one direction, whereas during a two-way call (such as a telephone conversation) information passes in both directions. In the case of parallel communication, several calls between distinct pairs of agents may take place simultaneously, but an agent can only call one other agent at the same instant. The sequential version, in which only one call can take place at the same time, is useful if the aim is to minimise the total number of calls.

We show that the temporal epistemic gossip problem is in NP even for a complex goal given in the form of a CNF and in the presence of constraints on the instants when calls can or must take place. This positive result, when compared to classical planning which is PSPACE-complete [9], follows from the reasonable assumption that knowledge is never destroyed. Indeed, we show that in the absence of temporal constraints and negative goals, temporal epistemic gossiping is in P. We then show maximality of this tractable subproblem in the sense that the problem becomes NP-complete in the following cases: in the presence of temporal constraints (even as weak as a simple upper bound on the execution time of a plan) and in the presence of negative goals (such as agent i should not learn the secret of agent j).

2 Definitions

Let *Prop* be a countable set of *propositional variables*, *Agt* be a finite set of *agents*.

The set of *visibility operators* is $OBS = \{\mathbf{S}_i \mid i \in Agt\}$. We denote by $OBS^{\leq d}$, for $d \in \mathbb{N}$, the set of sequences of visibility operators of depth at most d. Elements of $OBS^{\leq d}$ are noted σ, σ', etc. An *atom* is any sequence of visibility operators $\mathbf{S}_i$ of length at most d followed by a propositional variable. Formally,

$$ATM = \{\sigma p \mid \sigma \in OBS^{\leq d}, p \in Prop\}$$

Elements of ATM are noted α, α', etc. The *depth of an atom* is the number of visibility operators composing it. Note that if $d = 0$ then atoms are just propositional variables. Moreover, note that if the depth of atom $\alpha \in ATM$ is strictly less than d then $\mathbf{S}_i\alpha$ also belongs to ATM.

The set of *boolean formula* Fml_{bool} is comprised of formulas with the following grammar:

$$\varphi ::= \alpha \mid \neg\varphi \mid (\varphi \wedge \varphi)$$

A *conditional action* is a pair $\mathsf{a} = \langle pre(\mathsf{a}), \mathit{eff}(\mathsf{a})\rangle$ where:

- $pre(\mathsf{a}) \in Fml_{bool}$ is a boolean formula: the *precondition* of a;
- $\mathit{eff}(\mathsf{a}) \subseteq Fml_{bool} \times 2^{ATM} \times 2^{ATM}$ is a set of triples ce of the form

$$\langle cnd(ce), \mathit{ceff}^{+}(ce), \mathit{ceff}^{-}(ce)\rangle,$$

the *conditional effects* of a, where $cnd(ce)$ is a boolean formula (the condition) and $\mathit{ceff}^{+}(ce)$ and $\mathit{ceff}^{-}(ce)$ are sets of atoms (added and deleted atoms respectively).

In the case of the gossip problem, $Prop = \{s_i \mid i \in Agt\}$ where s_i is the secret of the agent i. Furthermore, secrets are constants which cannot be changed by any action. For simplicity of presentation, in the following, we write the visibility operator $\mathbf{S}_i$ as K_i since it represents knowledge by agent i. However, it should be pointed out that K_i is not a modal operator: $K_i s_j$ means that agent i knows the secret of agent j (rather than agent i knows that s_i is true). The actions are calls between two agents leading to an update of the two agents' knowledge. In one-way calls, after a call from agent i to agent j, agent j knows everything that agent i knew before the call and both know that they know these atoms. Indeed, they both know that they both know that they know these atoms, and so on up to the maximum epistemic depth d. More formally, $call(i,j) = \langle pre(call(i,j)), \mathit{eff}(call(i,j))\rangle$ with $pre(call(i,j)) = \top$ and $\mathit{eff}(call(i,j)) = \{\langle K_i\alpha, M^d_{ij}(\alpha), \emptyset\rangle \mid \alpha \in ATM\}$. where $M^d_{ij}(\alpha)$ is the mutual knowledge up to epistemic depth d of α by agents i and j: $M^d_{ij}(\alpha) = \{K_{k_1} \dots K_{k_r}\alpha \mid (k_1, \dots, k_r \in \{i,j\}) \wedge (r + depth(\alpha) \leq d)\}$. Two-way calls have the same effect as two simultaneous one-way calls.

An instance of the temporal epistemic gossip problem (TEGP) is given by a tuple $\mathit{\Pi} = \langle Init, Goal, Agt, I_p, I_n\rangle$:

$$\begin{aligned} Init &\subseteq ATM \\ Goal &\in Fml_{bool} \text{ is a conjunction of clauses} \\ I_p &\subseteq \mathbb{N} \times (\mathbb{N} \cup \{\infty\}) \times Agt \times Agt \\ I_n &\subseteq \mathbb{N} \times Agt \times Agt \end{aligned}$$

where $Init$ is the initial state; $Goal$ is the goal we want to achieve in the form of a CNF formula (that we identify with a set of clauses); I_p is the set of intervals during which two agents can call each other and I_n is the set of instants when two agents must call each other. The set I_n of necessary calls may correspond to calls that have been programmed in the network for some other purpose. We suppose that I_n is included in I_p, in the sense that for every $\langle t, i, j\rangle \in I_n$ there is a $\langle t_1, t_2, i, j\rangle \in I_p$ such that $t_1 \leq t \leq t_2$. In this paper we always consider the initial state $Init = \{K_i s_i \mid i \in Agt\}$ in which all agents know their own secrets.

A set of calls A between agents induces a partial function from 2^{ATM} to 2^{ATM}. For a state $s \in 2^{ATM}$:

$$\mathsf{A}(s) = \begin{cases} \bot & \text{if } \exists \mathsf{a} \in \mathsf{A} : s \not\models \mathit{pre}(\mathsf{a}), \text{ or} \\ & \exists \mathsf{a}_1, \mathsf{a}_2 \in \mathsf{A} : \mathsf{a}_1 = \mathit{call}(i_1, j_1) \text{ and} \\ & \mathsf{a}_2 = \mathit{call}(i_2, j_2) \text{ with } \{i_1, j_1\} \cap \{i_2, j_2\} \neq \emptyset \\ s \cup \bigcup\limits_{\substack{\mathsf{a} \in \mathsf{A} \\ ce \in \mathit{cnd}(\mathsf{a}) \\ \text{and } s \models \mathit{cnd}(\mathsf{a})}} \mathit{ceff}^+(ce) & \text{otherwise} \end{cases}$$

A *plan* is a relation $P \subseteq \mathbb{N} \times Agt \times Agt$. Given a plan P and a natural number t, the set of calls happening at instant t is $P(t) = \{(i, j) : (t, i, j) \in P\}$. We use $|P|$ to denote the number of distinct instants t for which $P(t) \neq \emptyset$. We use $T_P(k)$ to denote the kth instant (in strictly increasing order of time) at which a call happens in P: i.e. $T_P(1) < \ldots < T_P(|P|)$ and $\forall t$, $P(t) \neq \emptyset \Leftrightarrow \exists k \in \{1, \ldots, |P|\}$, $T(k) = t$. Our modelling of time by the natural numbers implicitly imposes a fixed duration of one time unit for each call.

Given a TEGP $\mathit{\Pi} = \langle \mathit{Init}, \mathit{Goal}, \mathit{Agt}, I_p, I_n \rangle$, a plan P *satisfies the temporal constraints* of $\mathit{\Pi}$ if and only if all the necessary calls are in P and every call in P is possible; formally: $I_n \subseteq P$ and for every $\langle t, i, j \rangle \in P$ there is a $\langle t_1, t_2, i, j \rangle \in I_p$ such that $t_1 \leq t \leq t_2$. Moreover, P *solves* the TEGP if and only if it satisfies the temporal constraints and for every s_0 such that $s_0 \models \mathit{Init}$ there is a sequence of states $\langle s_1, \cdots, s_{|P|} \rangle$ such that

- $s_{|P|} \models \mathit{Goal}$
- $s_{k+1} = P(T_P(k+1))(s_k)$ for every k with $0 \leq k < |P|$.

In the sequential version of the TEGP, a solution plan P must also satisfy $\forall t, \mathit{card}(P(t)) \leq 1$.

A TEGP defines in a natural way a call digraph G in which the vertices are the agents and the directed edges the possible calls. In the two-way version, G is a graph.

Example 1. Consider a network of five servers (which we call a, b, c, d and e) where each server can only communicate with a subset of the others. As part of the maintenance program, a, b, c and d send a backup of their data to e every night and these backups can be sent to any server during the day (between 8:00 and 18:00). The others servers can communicate with each other at any moment if there is a communication link between them. The communication graph is depicted in Figure 1.

There is some information on the server b (call it s_b) that needs to be transferred to a, and c must know that the transfer is done. As the servers have different access rights, the information on server a (call it s_a) should not be communicated to c. In the TEGP this can be represented by $\mathit{Goal} = K_c K_a s_b \wedge \neg K_c s_a$. There is a family of solution plans for this problem: $\mathit{call}(b, a)$ at instant t_1, $\mathit{call}(b, d)$ at instant t_2, $\mathit{call}(d, c)$ at instant t_3, where $t_1 < t_2 < t_3$ (together

with the necessary calls $call(a, e)$ at instant 2, $call(b, e)$ at instant 4, $call(c, e)$ at instant 20 and $call(d, e)$ at instant 22).

On the same network, another question that we can ask is whether c can know s_a without a being aware of this. In this case, the goal is $K_c s_a \wedge \neg K_a K_c s_a$. The answer is 'yes' since the following plan establishes the goal: the necessary $call(a, e)$ at instant 2, followed by $call(e, c)$ at an instant $t \in [8, 18]$ (together with the other necessary calls $call(b, e)$ at instant 4, $call(c, e)$ at instant 20 and $call(d, e)$ at instant 22).

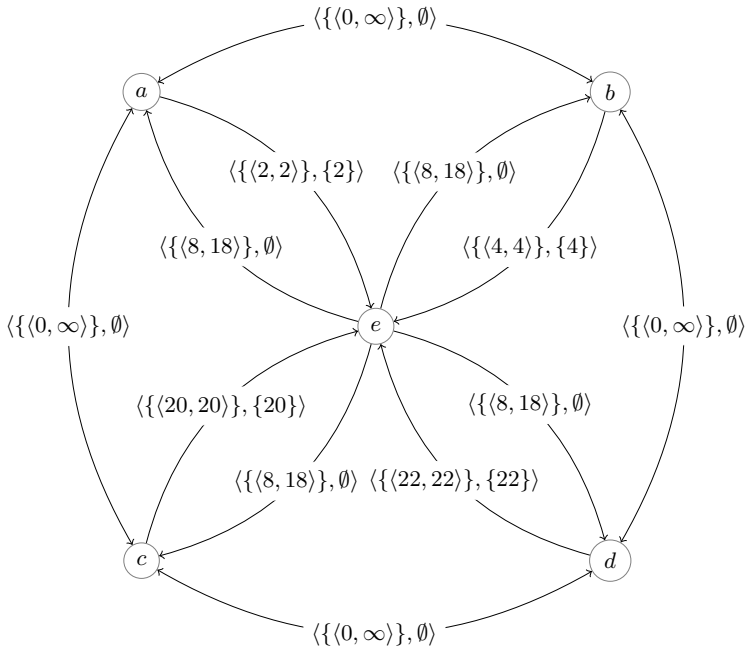


Fig. 1. Call graph for Example 1 involving necessary calls with e. A double-ended arrow represents two directed edges (i.e. the possibility of one-way calls in both directions).

3 Membership in NP

Proposition 1. *Let m be the number of clauses in the CNF of the goal. Let d be the the depth of atoms for the problem. If a plan for an instance of a TEGP exists, then there is a plan with $md(n-1) + |I_n|$ calls or less*

Proof. Let $\alpha \in ATM$ be an atom. $K_i\alpha$ can only be true if there is a path in the graph G between i and some agent who knows α. Without loss of generality, we

can assume that this path is cycle-free. In the worst case the length of this path is $n-1$. Then, for any atom with an epistemic depth d, at most $d(n-1)$ calls are needed for this atom to be true, by the concatenation of d paths of length $n-1$.

The number of calls needed for a disjunction of formulas to be true is the maximum of the number of calls needed for each formula. In a CNF, there are only disjunctions over atoms, so the number of calls needed for a disjunction is $d(n-1)$.

The number of calls needed for a conjunction of formulas is the sum of the number of calls needed for every formula, which here is at most $d(n-1)$. So, with m being the number of conjunctions in the CNF of the goal, at most $md(n-1)$ calls are needed for a problem with only possible calls.

Thus, if a plan P exists, then P contains a subset Q of $md(n-1)$ calls which are sufficient to establish all positive atoms in the goal. For a problem with necessary calls, it can happen that the plan Q does not contain all the necessary calls I_n of P; but adding these necessary calls to Q to form a plan Q' cannot destroy positive goals. All negative atoms in the goal are also valid after execution of Q' since all calls, and in particular the ones in P but not in Q', only establish positive atoms. Thus Q' is a plan of at most $md(n-1)+|I_n|$ calls.

4 A subproblem of the temporal gossip problem in P

We say that a TEGP instance is *positive* if its goal is a CNF containing only positive atoms. A special case of TEGP is the class of positive non-temporally-constrained epistemic gossip problems $\Pi = \langle Init, Goal, Agt, I_p, I_n\rangle$ where $I_p = \{\langle 0, \infty, i, j\rangle : (i,j) \in E\}$, for some E, and $Goal$ is a positive CNF. In this case, E is the set of edges in the call digraph: if a call is possible (as specified by E), it is possible at any instant. On the other hand, there is no restriction on the set of necessary calls I_n.

Proposition 2. *The class of positive non-temporally-constrained epistemic gossip problems can be solved in polynomial time.*

Proof. There is a simple polynomial-time algorithm for positive non-temporally-constrained epistemic gossip problems: make all possible calls in some fixed order and repeat this operation $md(n-1)+|I_n|$ times. Call this sequential plan Q. By the proof of Proposition 1, if a solution plan exists, there is a sequential solution plan P of length at most $md(n-1)+|I_n|$. The actions of P necessarily appear as a subsequence of Q. Since the goal and preconditions of actions contain only positive atoms, the extra actions of Q cannot destroy any goals or preconditions. It follows that if a solution plan exists, then Q is also a solution plan. Thus this simple algorithm solves the class of positive non-temporally-constrained epistemic gossip problems in polynomial time.

Given an arbitrary instance of TEGP, we can construct a positive non-temporally-constrained instance by ignoring negative goals and temporal constraints (specified by I_p). This is a polynomial-time solvable relaxation of the

original TEGP instance. This provides a relaxation which is inspired by the well-known delete-free relaxation of classical planning problems and is orthogonal to the relaxation of temporal planning problems based on establisher-uniqueness and monotonic fluents [13].

5 NP-completeness when execution time is bounded

The simplest temporal constraint is just a time limit on the execution of a plan. In the case of sequential plans this simply corresponds to placing a bound on plan length (which is equal to the number of calls) whereas in the parallel case execution time corresponds to the number of steps. We show in this section that this single constraint (a time limit on plan execution) is sufficient to render the epistemic gossip problem NP-complete. It is worth noting that the PSPACE complexity of classical planning is not affected by the possibility of placing an arbitrary limit on plan length, but the special case of delete-free planning passes from P to NP-hard when a bound is placed on plan length [9]. We show that this remains true for the specific case of gossiping problems.

We begin by studying the sequential case of TEGP.

Proposition 3. *The epistemic gossip problem with no temporal constraints but with a bound on the number of calls is NP-complete, even when the goal is a conjunction of positive atoms.*

Proof. We will exhibit a polynomial-time reduction from the well-known NP-complete problem SAT to the version of the epistemic gossip problem whose question is whether there is a sequential solution plan of length at most L. To do so, for a given set of clauses $\{C_1, \ldots, C_m\}$ we need the following agents:

- an agent S (the source),
- literal agents, i.e., agents for every variable and every negation of a variable (which we name, respectively, x^+ and x^-) for each variable x of the SAT instance,
- clause agents, i.e., agents for every clause (which we name C_i for the ith clause of the SAT instance).

Before performing this construction, we first add a dummy clause $(x \vee \neg x)$ for each SAT variable x. This clearly does not change the semantics of the instance but it does force us to specify the truth value of each variable in a solution of the SAT instance.

The source agent S and clause agents can only communicate with literal agents. The source agent S can communicate with every literal agent. A literal agent can only communicate with S and those clauses it is a member of. The graph G of communications is shown in Figure 2 for a particular SAT instance. In this example, $C_1 = (\neg x \vee y \vee z)$, $C_2 = (\neg y \vee z)$ and the clauses C_3, C_4, C_5 are the dummy clauses $(x \vee \neg x)$, $(y \vee \neg y)$, $(z \vee \neg z)$.

A variable x is considered to be true (false) if the secret s_S passes through x^+ (respectively, x^-) in the solution plan on its way to the agent representing

the dummy clause $(x \vee \neg x)$. The bound on the number of actions will prevent the possibility of s_S passing through both x^+ and x^-. So the choice of whether the secret s_S passes through x^+ or x^- determines an assignment to the variable x in the SAT problem.

The goal of this instance of TEGP is that every clause agent knows the secret of S ($Goal = \bigwedge_{C_i} K_{C_i} s_S$). Now set the bound on plan length to be $L = 2n + m$, where n is the number of variables in the SAT instance and m the number of clauses in the original instance. With the new dummy clauses, the total number of clauses is $n + m$.

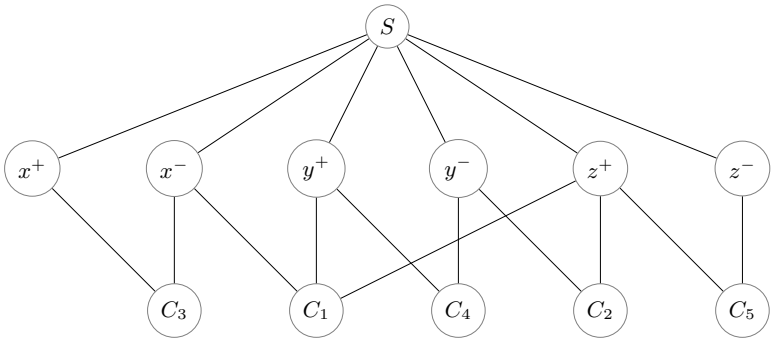


Fig. 2. Representation of the formula $(\neg x \vee y \vee z) \wedge (\neg y \vee z)$ as a temporal epistemic gossip problem in which the question is whether there is a plan using no more than 8 calls.

In a solution plan P we require at least n calls, one to either x^+ or x^-, for the n SAT variables x, in order for the secret s_S to be able to reach the agent corresponding to the dummy clause $x \vee \neg x$. P must also contain at least $n + m$ calls to the clause agents C_i (including the dummy clauses) to establish the goals $K_{C_i} s_S$. A solution plan of length precisely $2n + m$ corresponds to a solution of the corresponding SAT instance since such a plan defines a unique assignment to all variables that satisfies all clauses. For example, the solution $x = false$, $y = false$, $z = true$ to the SAT instance of Figure 2 corresponds to the following solution plan of length 8: S calls x^-; S calls y^-; S calls z^+; x^- calls C_1; z^+ calls C_2; x^- calls C_3; y^- calls C_4; z^+ calls C_5. This reduction from SAT is clearly polynomial.

Proposition 1 proves the existence of a polynomial-length certificate for positive instances of the decision version of TEGP. Such certificates (solutions) can be verified in polynomial time. Thus TEGP $\in$ NP. Since the epistemic gossip problem with no temporal constraints but with a bound on the number of calls is clearly still in NP, this completes the proof of NP-completeness.

The proof of Proposition 3 was given for the case of two-way communications. It is trivial to adapt it to the case of one-way communications (for example, by

only allowing calls from S to literal agents and from literal agents to clause agents).

We now consider the parallel version of the TEGP. Recall that in the parallel version of the TEGP, several calls may take place at each step, provided no agent is concerned by more than one call at each step.

Proposition 4. *The parallel version of the epistemic gossip problem with no temporal constraints except for a bound on the number of steps is NP-complete even when the goal is a conjunction of positive atoms.*

Proof. By the same argument as in the proof of Proposition 1, the problem is in NP. We complete the proof by exhibiting a polynomial reduction from 3SAT which is well known to be NP-complete. Given an instance I_{3SAT} of 3SAT, by introducing sufficiently many new variables x' which are copies of old variables x (together with the clauses $x \vee \neg x'$, $\neg x \vee x'$ to impose equality of x and x') we can transform I_{3SAT} into an equivalent instance in which each literal does not occur in more than three clauses. This is a polynomial reduction since we need to introduce at most one copy of each variable x per clause in which it occurs in I_{3SAT}. Therefore, from now on, we suppose that each literal occurs in at most two clauses in I_{3SAT}.

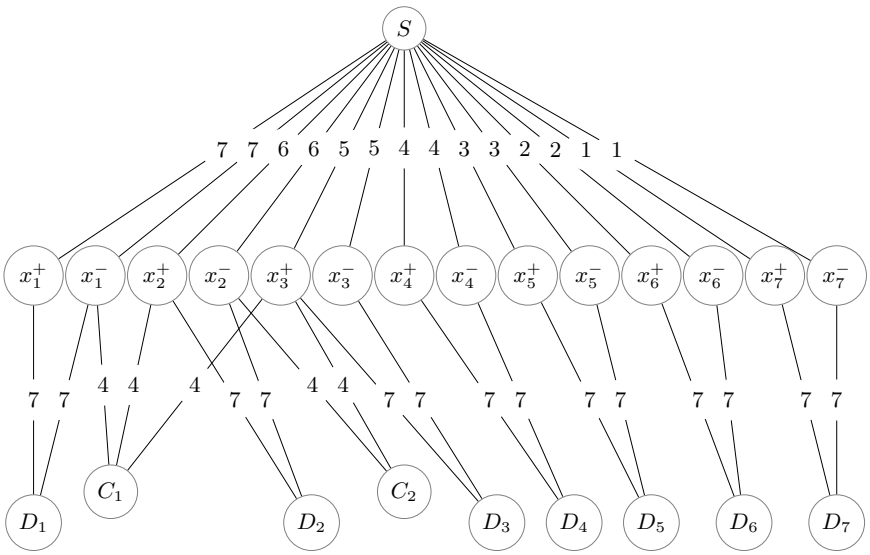


Fig. 3. Representation of the formula $(\neg x \vee y \vee z) \wedge (\neg y \vee z)$ as a temporal epistemic gossip problem in which the question is whether there is a parallel plan using no more than 14 steps.

We construct an instance I of the epistemic gossip problem which has a parallel solution plan of length $2p$ if and only if I_{3SAT} is satisfiable. We choose the value of p to be strictly greater than $n+3$, where n is the number of variables in I_{3SAT}. To be concrete, we can choose $p = n + 4$. We add to I_{3SAT} $p - n$ new dummy variables $x_{n+1}, \ldots, x_p$ none of which occur in the clauses of I_{3SAT}. In I there is an agent S (the source), literal agents x_i^+, x_i^- for each variable x_i $(i = 1, \ldots, p)$, and a clause agent C_j for each of the clauses C_j $(j = 1, \ldots, m)$ of I_{3SAT}. For each variable x_i $(i = 1, \ldots, p)$, we also add a dummy-clause agent D_i which we can consider as representing the dummy clause $x_i \vee \neg x_i$. Instead of linking these basic agents directly, we place paths of new agents between these basic agents. Between agent S and agent x_i^+ we add a path of length $p + 1 - i$. Similarly, we add a new path of the same length between S and agent x_i^-. For $i = 1, \ldots, p$, we add two new paths both of length p between the literal agents x_i^+ and x_i^- and the dummy-clause agent D_i. For each clause C_j of I_{3SAT}, we also add three new paths of length q from the agents corresponding to the literals of C_j to the agent C_j, where $q = p - 2 = n + 1$. The resulting network is shown in Figure 3 for an example instance. The numbers on edges in this figure represent the length of the corresponding path. For example, there are 6 intermediate agents (not shown so as not to clutter up the figure) between the agents S and x_1^+. The goal of I is

$$\left(\bigwedge_{i=1}^{p} K_{D_i} s_S\right) \quad \wedge \quad \left(\bigwedge_{j=1}^{m} K_{C_j} s_S\right)$$

In order to establish the goal $K_{D_i} s_S$, the secret s_S has to follow a path from S to D_i. The shortest paths from S to D_1 are of length $2p$ and pass through either x_1^+ or x_1^-. Recall that our aim is to find a plan whose execution requires at most $2p$ steps. Thus, to establish $K_{D_1} s_S$ in $2p$ steps, during the first step, S must call the first agent on the path to x_1^+ or the first agent on the path to x_1^-. The shortest path from S to D_2 is of length $2p - 1$, so during the second step, S must call the first agent either on the path to x_2^+ or the first agent on the path to x_2^-. By a simple inductive argument, we can see that during step i $(i = 1, \ldots, p)$, S must call the first agent on the path to x_i^+ or x_i^-. We can consider that the choice of whether the secret s_S passes through x_i^+ or x_i^- determines an assignment to the variables x_i. Due to the diminishing lengths of these paths as i increases, the secret s_S arrives simultaneously at the literal agents, either x_i^+ or x_i^-, for $i = 1, \ldots, p$. Another p steps are then required to send in parallel this secret to the dummy-clause agents D_j, for a total number of steps of $2p$. Almost simultaneously (within two time units), the secret s_S arrives at the clause agents C_j, provided it has passed through one of the agents corresponding to the literals of C_j. The length of paths from literal agents (x_i^+ or x_i^-) to clause agents C_j is $q = p - 3$ which is slightly less than p to allow for the fact that a literal agent, say x_i^+, may have to send s_S along at most four paths: first towards D_i, then towards the (at most) three clauses in which x_i occurs.

It is important to note that S is necessarily occupied during the first p steps, as described above, so if S were to try to send its secret both to x_i^+ and x_i^- the secret could not arrive via the second of these paths at a clause agent C_j in less than $2p - n + q = 3p - n - 3$ steps which is greater than the upper bound of $2p$ steps (since $p = n + 4$). By our construction, the goal $K_{C_j} s_S$ is established only if the assignment to the variables x_i determined by the solution plan satisfies the clause C_j. Hence, parallel solution plans of length $2p$ steps correspond precisely to solutions of I_{3SAT}. We have therefore demonstrated a polynomial reduction from 3SAT to the parallel version of the epistemic gossip problem with a bound on the number of steps.

The following corollary follows from the fact that we can place an upper bound L on the number of steps in a plan by simply imposing via I_p an interval of possible instants $[1, L]$ for all calls.

Corollary 1. *TEGP is NP-complete.*

6 NP-completeness of gossiping with negative goals

We show in this section that even without temporal constraints or a bound on plan length, when we allow negative goals the problem of deciding the existence of a solution plan is NP-complete.

Proposition 5. *The epistemic gossip problem with possibly negative goals is NP-complete even in the absence of any temporal constraints or bound on plan length.*

Proof. The same argument as in the proof of Proposition 1 shows that the problem belongs to NP since it is a subproblem of TEGP.

To complete the proof, it suffices to give a polynomial reduction from SAT. Let I_{SAT} be an instance of SAT. We will construct a call graph G and a set of goals such that the corresponding instance I_{Gossip} of the epistemic gossip problem is equivalent to I_{SAT}. Recall that the nodes of the call graph G are the agents and the edges of G the communication links between agents.

For each propositional variable x in I_{SAT}, we add four nodes x^+, x^-, b_x, d_x to G joined by the edges shown in Figure 4(b). There is a source node S in G and edges (S, x^+), (S, x^-) for each variable x in I_{SAT}. For each clause C_j in I_{SAT}, we add a node C_j joined to the nodes corresponding to the literals of C_j. This is illustrated in Figure 4(a) for the clause $C_j = \neg x \vee y \vee z$. The solution plan to I_{Gossip} will make the secret s_S transit through x^+ (on its way from S to some clause node C_j) if and only if $x = true$ in the corresponding solution to I_{SAT}.

For each clause C_j in I_{SAT}, G contains a clause gadget as illustrated in Figure 4(a) for the clause $\neg x \vee y \vee z$. We also add $K_{C_j} s_S$ to the set of goals. Clearly, the secret s_S must transit through one of the nodes corresponding to the literals of C_j (x^-, y^+ or z^+ in the example of Figure 4) to achieve the goal $K_{C_j} s_S$.

To complete the reduction, it only remains to impose the constraint that s_a transits through at most one of the nodes x^+, x^-, for each variable x of I_{SAT}. This is achieved by the negation gadget shown in Figure 4(b) for each variable x. We add the goals $K_{d_x} s_{b_x}$, $\neg(K_{d_x} s_S)$ for each variable x, and the goal $\neg(K_{C_j} s_{b_x})$ for each variable x and each clause C_j (containing the literal x or $\neg x$). The goal $K_{d_x} s_{b_x}$ ensures that the secret s_{b_x} transits through x or $\neg x$. Now, recall that we assume that during a call, agents communicate all their knowledge. Suppose that s_{b_x} transits through x^+: then s_S cannot transit through x^+ before s_{b_x} (because of the negative goal $\neg(K_{d_x} s_S)$) and cannot transit through x^+ after s_{b_x} (because of the negative goal $\neg(K_{C_j} s_{b_x})$). By a similar argument, if s_{b_x} transits through x^-, then s_S cannot transit through x^+. Thus, this gadget imposes that s_S transits through exactly one of the nodes x^+, x^-.

We have shown that I_{SAT} has a solution if and only if I_{Gossip} has a solution. Since the reduction is clearly polynomial, this completes the proof.

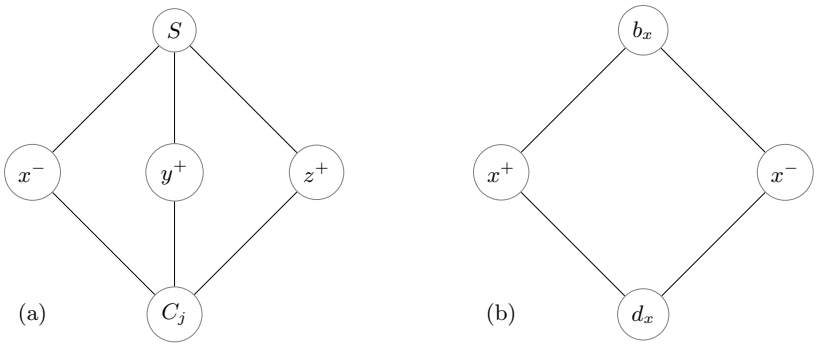


Fig. 4. (a) gadget imposing the clause $C_j = \neg x \vee y \vee z$; (b) gadget imposing the choice between x and $\neg x$.

The NP-completeness shown in the proof of Proposition 5 for two-way communication, would not be affected by a restriction to one-way communication. Similarly NP-completeness holds for both the sequential and parallel versions of the gossip problem.

7 Discussion and conclusion

We have defined temporal epistemic gossip problems and have investigated their complexity. Our results are in line with previous results concerning epistemic planning: it is possible to add an epistemic dimension to planning, thus increasing expressibility, without increasing complexity [10].

We have assumed a centralized approach in which a centralized planner decides the actions of all agents. Several other researchers have recently studied

distributed versions of the classical gossip problem where the agents have to decide themselves whom to call, based on the knowledge (and ignorance) they have [1, 15, 14, 4, 3, 5, 2]. An interesting avenue of future research would be to consider the epistemic gossip problem in this framework.

Several other variants of our centralized model could also be investigated, including the precondition that i has to know the telephone number of j in order to call j and telephone numbers are communicated in the same way as secrets. In another variant, the secrets can be passwords which are no longer constants since each agent i can change their own password [12].

References

1. Apt, K.R., Grossi, D., van der Hoek, W.: Epistemic protocols for distributed gossiping. In: Ramanujam, R. (ed.) Proceedings Fifteenth Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2015. EPTCS, vol. 215, pp. 51–66 (2015), https://doi.org/10.4204/EPTCS.215.5
2. Apt, K.R., Grossi, D., van der Hoek, W.: When are two gossips the same? types of communication in epistemic gossip protocols. CoRR **abs/1807.05283** (2018), http://arxiv.org/abs/1807.05283
3. Apt, K.R., Kopczynski, E., Wojtczak, D.: On the computational complexity of gossip protocols. In: Sierra, C. (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017. pp. 765–771. ijcai.org (2017), https://doi.org/10.24963/ijcai.2017/106
4. Apt, K.R., Wojtczak, D.: Common knowledge in a logic of gossips. In: Lang, J. (ed.) Proceedings Sixteenth Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2017. EPTCS, vol. 251, pp. 10–27 (2017), https://doi.org/10.4204/EPTCS.251.2
5. Apt, K.R., Wojtczak, D.: Decidability of fair termination of gossip protocols. In: Eiter, T., Sands, D., Sutcliffe, G., Voronkov, A. (eds.) IWIL@LPAR 2017 Workshop and LPAR-21 Short Presentations. Kalpa Publications in Computing, vol. 1. EasyChair (2017), http://www.easychair.org/publications/paper/342983
6. Attamah, M., van Ditmarsch, H., Grossi, D., van der Hoek, W.: A framework for epistemic gossip protocols. In: Bulling, N. (ed.) Multi-Agent Systems - 12th European Conference, EUMAS 2014. Lecture Notes in Computer Science, vol. 8953, pp. 193–209. Springer (2014), http://dx.doi.org/10.1007/978-3-319-17130-2_13
7. Aucher, G., Bolander, T.: Undecidability in epistemic planning. In: Rossi, F. (ed.) Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI). pp. 27–33. AAAI Press (2013)
8. Bolander, T., Andersen, M.B.: Epistemic planning for single and multi-agent systems. Journal of Applied Non-Classical Logics **21**(1), 9–34 (2011). https://doi.org/10.3166/JANCL.21.9
9. Bylander, T.: The computational complexity of propositional STRIPS planning. Artif. Intell. **69**(1-2), 165–204 (1994), https://doi.org/10.1016/0004-3702(94)90081-7
10. Cooper, M.C., Herzig, A., Maffre, F., Maris, F., Régnier, P.: A simple account of multi-agent epistemic planning. In: Kaminka et al. [19], pp. 193–201, http://dx.doi.org/10.3233/978-1-61499-672-9-193

11. Cooper, M.C., Herzig, A., Maffre, F., Maris, F., Régnier, P.: Simple epistemic planning: Generalised gossiping. In: Kaminka et al. [19], pp. 1563–1564, http://dx.doi.org/10.3233/978-1-61499-672-9-1563
12. Cooper, M.C., Herzig, A., Maffre, F., Maris, F., Régnier, P.: Simple epistemic planning: generalised gossiping. CoRR **abs/1606.03244** (2016), http://arxiv.org/abs/1606.03244
13. Cooper, M.C., Maris, F., Régnier, P.: Monotone temporal planning: Tractability, extensions and applications. J. Artif. Intell. Res. **50**, 447–485 (2014), https://doi.org/10.1613/jair.4358
14. Ditmarsch, H.v., Eijck, J.v., Pardo, P., Ramezanian, R., Schwarzentruber, F.: Epistemic protocols for dynamic gossip. J. Applied Logic **20**, 1–31 (2017), https://doi.org/10.1016/j.jal.2016.12.001
15. Ditmarsch, H.v., Grossi, D., Herzig, A., Hoek, W.v.d., Kuijer, L.B.: Parameters for epistemic gossip problems. In: Proc. LOFT 2016 (2016)
16. Hedetniemi, S.M., Hedetniemi, S.T., Liestman, A.L.: A survey of gossiping and broadcasting in communication networks. Networks **18**(4), 319–349 (1988), https://doi.org/10.1002/net.3230180406
17. Herzig, A., Lorini, E., Maffre, F.: A poor man's epistemic logic based on propositional assignment and higher-order observation. In: van der Hoek, W., Holliday, W.H., Wang, W. (eds.) Logic, Rationality, and Interaction - 5th International Workshop, LORI 2015. Lecture Notes in Computer Science, vol. 9394, pp. 156–168. Springer (2015), https://doi.org/10.1007/978-3-662-48561-3_13
18. Herzig, A., Maffre, F.: How to share knowledge by gossiping. AI Commun. **30**(1), 1–17 (2017), http://dx.doi.org/10.3233/AIC-170723
19. Kaminka, G.A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., van Harmelen, F. (eds.): ECAI 2016 - 22nd European Conference on Artificial Intelligence, Frontiers in Artificial Intelligence and Applications, vol. 285. IOS Press (2016)
20. Kominis, F., Geffner, H.: Beliefs in multiagent planning: from one agent to many. In: Brafman, R.I., Domshlak, C., Haslum, P., Zilberstein, S. (eds.) Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS). pp. 147–155. AAAI Press (2015)
21. Löwe, B., Pacuit, E., Witzel, A.: DEL planning and some tractable cases. In: Proceedings of the 3rd International International Workshop on Logic, Rationality and Interaction. pp. 179–192. Springer Berlin Heidelberg (2011)
22. Maffre, F.: Ignorance is bliss: observability-based dynamic epistemic logics and their applications. phdthesis, Paul Sabatier University, Toulouse, France (2016), https://tel.archives-ouvertes.fr/tel-01488408
23. Muise, C., Belle, V., Felli, P., McIlraith, S.A., Miller, T., Pearce, A.R., Sonenberg, L.: Planning over multi-agent epistemic states: A classical planning approach. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015). pp. 3327–3334. AAAI Press (2015)

Résolution des problèmes de planification de trajectoires en milieu urbain en fonction de la disponibilité a priori des capteurs et de la propagation des erreurs d'exécution

Article soumis et accepté à AIAA Scitech 2019

Jean-Alexis Delamer*[1], Yoko Watanabe†[1], and Caroline P. Carvalho Chanel‡[2]

[1]ONERA - The French Aerospace Laboratory, Toulouse, France
[2]ISAE-SUPAERO, Université de Toulouse, France

Abstract

This paper addresses safe path planning problem in urban environments under onboard sensor availability uncertainty. In this context, an approach based on Mixed-Observability Markov Decision Process (MOMDP) is presented. Such a model enables the planner to deal with a priori probabilistic sensor availability and path execution error propagation, the which depends on the navigation solution. Due to modelling particularities of this safe path planning problem, such as bounded hidden and fully observable state variables, discrete actions and particular transition function form, the belief state update function becomes a complex step that cannot be ignored during planning. Recent advances in Partially Observable Markov Decision Process (POMDP) solving have proposed a planning algorithm called POMCP, which is based on Monte-Carlo Tree Search method. It allows the planner to work on the history of the action-observation pairs without the need to compute belief state updates. Thereby, this paper proposes to apply a POMCP-like algorithm to solve the addressed MOMDP safe path planning problem. The obtained results show the feasibility of the approach and the impact of considering different a priori probabilistic sensor availability on the result policy.

1 Introduction

Navigating through an urban environment with autonomous vehicles is a challenging problem, as safety and efficiency should be ensured [17, 8]. Navigation capability of such vehicles highly depends on their onboard sensor performances – both availability and precision – which can vary with the environment. For example, the widely-used GPS (Global Positionning System) has its precision depending on its satellite constellation visibility which depends on geo-localization and time. Availability and precision of vision sensor measurements are influenced by image textures, visibility of object-of-interest, lighting conditions, etc. However, fortunately, some of such sensor performances can be predicted from a priori knowledge on vehicle operation surroundings. As the GPS satellite orbit is known, it is possible to have some knowledge on GPS precision, represented as Dilution of Precision (DOP), given a 3D environment model, geo-localization and time window[10]. Such information could be very useful in safe path planning task, as it enable the planner to predict localization and path execution error propagation[20, 17].

In this context, this paper tackles such safe path planning problem for autonomous vehicles, especially focusing on flying ones (drones, UAVs), in an urban environment by exploiting probabilistic onboard sensor availability maps and path execution error propagation. Similar problems have already been addressed by [20], [17], [4] and [1], by considering vehicle localization uncertainty propagated along a planned path in function of the environment. For instance, [20] applies the A* algorithm and makes use of a concept of uncertainty corridor to evaluate a path plan for choosing the most efficient and safe path. [4] and [1] propagate the position uncertainty during path search using the Rapidly-exploring Random Belief Trees (RRBT) algorithm. However, any of these approaches considers a complete closed-loop vehicle motion model with GNC (Guidance, Navigation, and Control) functions into the decisional process.

The safe path planning problem addressed in this paper is modeled as a Mixed-Observability Markov Decision Process (MOMDP) [16]. MOMDP is an extension of the classical Partially Observable Markov Decision Process (POMDP) [9]. MOMDP allows the

*PhD Student, Information Processing and Systems Departement (DTIS), jean-alexis.delamer@onera.fr
†Researcher, Information Processing and Systems Department (DTIS), Yoko.Watanabe@onera.fr
‡Researcher, Design and Control of Aerospace Vehicles Department (DCAS), caroline.chanel@isae-supaero.fr

factorization of the state space into fully and partially observable state variables. It holds in a smaller belief state space dimension, and hence decreases the time of policy computation. In this work, the state transition and observation functions of the MOMDP are built on the vehicle GNC model, and on the a priori knowledge of the environment given as probability grid maps of obstacles and onboard sensor availabilities.

The MOMDP model built for our safe path planning problem has some particularities – bounded hidden and fully observable state variables, discrete actions and transition function form. These particularities cause a specific belief state transition function which makes the resulting belief state not easy to be handled during planning. One approach to deal with this difficulty is to learn and approximate the belief state (or state distribution) by a mixture of Gaussian functions [6]. Nevertheless, the computation of path cost, which was defined based on the execution error (i.e. on the belief state transition) as in [20], makes time-expensive value and policy optimization.

This paper proposes to solve the MOMDP-modeled safe path planning problem in a different way, by making use of a POMCP-like algorithm, and by proposing a simpler cost function. POMCP [18] extends UCT [11], an online Monte-Carlo tree search algorithm, to partially observable environments. POMCP, as UCT, applies the UCB1 (Upper Confidence Bounds) action selection strategy during value and policy optimization, what allows to deal with the explore-exploit trade-off while minimizing the regret of choosing a wrong action. Moreover, POMCP approximates the value (which defines the most promising action) of a belief state, by the average of costs evaluated during simulations departing from an initial state distribution. Each simulation sequentially samples a state, performs a selected action and samples an observation following the MOMDP model. This sequential mechanism allows us to generate a policy tree. In this tree, each belief node is represented by a history of action-observation pairs from the initial belief state (state distribution). Such tree representation and value computation avoid the need of an explicit belief state representation during planning.

To evaluate the proposed POMCP-like planning algorithm, policies were computed and simulated for: (i) different probabilistic sensor availability maps, which have an impact on the execution error propagation; (ii) different penalty costs for collisions, which have a direct impact on behavior of the computed policy and, consequently, on the mission success rate. The obtained results are promising in terms of policy time computation, simulated paths success rate and averaged path cost.

This paper is organized as follows: firstly the MOMDP model for this application case is presented. Then, a POMCP-like algorithm is proposed; the simulation test results show the impact of different probablilistic sensor availability maps and penalty costs on the policy. Finally, conclusion and future works are discussed.

2 UAV Safe Path Planning Problem

This paper addresses a safe path planning problem of autonomous vehicles by considering it as a problem of finding a navigation and guidance strategy for making vehicles reach a given destination safely and efficiently in a cluttered environment. This challenging problem considers a priori probabilistic availability of the vehicle onboard sensors and execution error propagation which depends on the navigation solution being used.

Let us suppose a vehicle equipped with N different onboard navigation sensors, such as inertial sensors, GPS and vision sensors, which are used by the GNC system to execute a path. The navigation filter estimates the vehicle state x and its error covariance matrix P by using measurements from a set of selected and available sensors, which defines a navigation mode. The guidance and control module executes a selected path segment (or action) by using the navigation solution. A priori knowledge on the environment is assumed to be given by a set of probability grid maps of obstacles and availability of each of the N onboard navigation sensors. These maps are used during planning task to propagate the path execution uncertainty given the probabilistic sensors' availability, and then to evaluate obstacle collision risk.

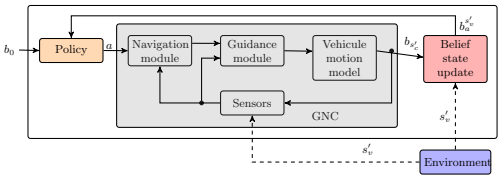


Figure 1: System architecture diagram. The GNC closed-loop vehicle model is incorporated into the MOMDP transition function. A priori information forms a set of probability grid maps of obstacles and sensor availability.

2.1 GNC transition model

The vehicle GNC model is described in this section (see [7] for more details). The transitional state of a vehicle $x = \begin{bmatrix}\mathcal{X}^T & \mathcal{V}^T & b_a^T\end{bmatrix}^T$ is defined respectively by its position, velocity and the accelerometer bias. The state transition is defined such as :

$$x_{k+1} = \Phi x_k + B a_k + v_{k+1} \tag{1}$$

where a_k is the acceleration, $v_{k+1} \sim N(0, Q)$ is the discretized process noise and

$$\Phi = \begin{bmatrix} I & \Delta t I & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}, B = \begin{bmatrix} \frac{\Delta t^2}{2} I \\ \Delta t I \\ 0 \end{bmatrix}$$

The state estimator is based on an EKF (Extended Kalman Filter, [19]) which proceeds in two steps; Prediction by the IMU acceleration measurements, and Correction by the other navigation sensor measurements $S_n, n \in N$, if available.

INS Prediction: The IMU acceleration measurement is given as:

$$a_{\mathrm{IMU}_k} = R_{BI_k}(a_k - g) + b_{a_k} + \xi_{\mathrm{IMU}_k} \quad (2)$$

where, R_{BI_k} is a rotation matrix from the inertial to the vehicle body frames provided by INS, g is the gravity vector and $\xi_{\mathrm{IMU}} \sim N(0, R_{\mathrm{IMU}})$ is the IMU acceleration measurement noise. According to the process model (1), the estimated state $\hat{x}_k$ is propagated to :

$$\hat{x}^-_{k+1} = \Phi \hat{x}_k + B\left(R^T_{BI_k}\left(a_{\mathrm{IMU}_k} - \hat{b}_{a_k}\right) + g\right). \quad (3)$$

Then, the predicted state estimation error can be written as:

$$\tilde{x}^-_{k+1} = x_{k+1} - \hat{x}^-_{k+1} = (\Phi - \Delta\Phi^a_k)\,\tilde{x}_k + v_{k+1} - BR^T_{BI_k}\xi_{\mathrm{IMU}_k} \quad (4)$$

where, $\Delta\Phi^a_k = BR^T_{BI_k}\begin{bmatrix}0 & 0 & I\end{bmatrix}$. And, the associated error covariance is then given by :

$$P^-_{k+1} = (\Phi - \Delta\Phi^a_k)\,P_k\,(\Phi - \Delta\Phi^a_k)^T + Q + \tilde{R}_{\mathrm{IMU}_k} \quad (5)$$

where, $\tilde{R}_{\mathrm{IMU_k}} = BR^T_{BI_k}R_{\mathrm{IMU}}R_{BI_k}B^T$. For simplicity, we consider the case of $R_{\mathrm{IMU}} = \sigma^2_{IMU}I$ and hence $\tilde{R}_{\mathrm{IMU}} = BR_{\mathrm{IMU}}B^T$ remains constant for all k.

Sensor correction: When the n-th onboard sensor measurement $z_{\mathrm{S_n}k+1}$ is available at t_{k+1}, the predicted state (1) can be corrected by using it:

$$z_{\mathrm{S_n}k+1} = H_{\mathrm{S_n}}x_{k+1} + \xi_{\mathrm{S_n}k+1}$$

where, $\xi_{\mathrm{S_n}} \sim N(0, R_{\mathrm{S_n}})$ is a measurement noise of the n-th sensor. Then, the estimated state is corrected such as :

$$\hat{x}_{k+1} = \hat{x}^-_{k+1} + K_{\mathrm{S_n}k+1}H_{\mathrm{S_n}}\left(z_{\mathrm{S_n}k+1} - H_{\mathrm{S_n}}\hat{x}^-_{k+1}\right) \quad (6)$$

where, $K_{\mathrm{S_n}k+1} = P^-_{k+1}H^T_{\mathrm{S_n}}\left(H_{\mathrm{S_n}}P^-_{k+1}H^T_{\mathrm{S_n}} + R_{\mathrm{S_n}}\right)^{-1}$ is the Kalman gain. Then, the estimation error and its covariance are updated as :

$$\begin{aligned} \tilde{x}_{k+1} &= \left(I - K_{\mathrm{S_n}k+1}H_{\mathrm{S_n}}\right)\tilde{x}_{k+1} - K_{\mathrm{S_n}k+1}\xi_{\mathrm{S_n}k+1} \\ P_{k+1} &= \left(I - K_{\mathrm{S_n}k+1}H_{\mathrm{S_n}}\right)P^-_{k+1} \end{aligned} \quad (7)$$

If there is no onboar sensor available or selected, the estimation error an its covariance remain as those from the prediction step.

Guidance law: Given a desired velocity $\mathcal{V}_{\mathrm{ref}}$, the following linear guidance law is applied:

$$a_k = \hat{K}_p\mathcal{V}_{\mathrm{ref}} - K_d(\hat{\mathcal{V}}_k - \mathcal{V}_{\mathrm{ref}}) = K_p\mathcal{V}_{\mathrm{ref}} - K_d\hat{\mathcal{V}}_k \quad (8)$$

where, K_p,$K_d > O$ are control gains and $\hat{\mathcal{V}}_k$ is the estimated vehicle velocity at instant t_k, i.e., $\hat{\mathcal{V}}_k = \begin{bmatrix}0 & I & 0\end{bmatrix}\hat{x}_k$. Then, x_{k+1} can be obtained by substituting this guidance law (8) into the discrete process model (1) :

$$x_{k+1} = (\Phi - \Delta\Phi^{\mathcal{V}})x_k + BK_p\mathcal{V}_{\mathrm{ref}} + \Delta\Phi^{\mathcal{V}}\tilde{x}_k + v_{k+1} \quad (9)$$

where, $\Delta\Phi^{\mathcal{V}} = BK_d\begin{bmatrix}0 & I & 0\end{bmatrix}$. Hence, given the current state x_k, the state x_{k+1} follows the Gaussian distribution as described below.

$$\begin{aligned} x_{k+1} &\sim N((\Phi - \Delta\Phi^{\mathcal{V}})x_k + BK_p\mathcal{V}_{\mathrm{ref}}, \Delta\Phi^{\mathcal{V}}P_k{\Delta\Phi^{\mathcal{V}}}^T + Q) \\ &= N(\bar{x}_{k+1|k}, \tilde{Q}^a_{k+1}) \end{aligned} \quad (10)$$

where, the covariance $\tilde{Q}^a_{k+1}$ becomes a function of the estimation error covariance P_k given by the navigation system (Eq. 7 or 5). Note that, this normal distribution defines the execution error of the path segment (or the action effect) being considered in the path planning problem.

3 MOMDP Model for efficient and safe path planning problem

In this paper, the safe and efficient path planning problem is modelled as a Mixed-Observability Markov Decision Process (MOMDP) ([2] and [16]), which is an extension of the POMDP (Partially Observable Markov Decision Process) [9]. In MOMDPs the state space is factorized into partially observable state variables and fully observable state variables. In this way, the belief state space (distribution probability over states) has smaller dimension compared to the classical POMDP framework. It decreases processing time for value and policy computation.

Applied to the path planning problem here addressed, one can assume that a vehicle always knows the current sensors' availability, i.e. at a given decision time step the embedded system knows if a given sensor can be used or not. Then, sensors' availability is considered as fully observable state variable of the model. On the other hand, the vehicle state vector x is considered as a hidden and non observable state from the planning model point of view. Given the GNC transition model described in Sec. 2.1, the only output considered is the execution error distribution (bounded by the covariance matrix $\tilde{Q}$, see. Sec. 2.1), and so, neither partial nor direct symbolic observation is possible for it. Figure 1 illustrates the system architecture with different modules.

Therefore, the MOMDP here addressed is defined as a tuple $\{\mathcal{S}_v, \mathcal{S}_h, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{O}, \mathcal{C}, b_0\}$, such as:

- $\mathcal{S}_v$ is the bounded set of fully observable states;
- $\mathcal{S}_h$ is the bounded set of hidden continuous states;
- $\mathcal{A}$ is the bounded set of actions;
- Ω is the bounded set of observations;
- $\mathcal{T}$ is the state transition function;
- $\mathcal{O}$ is the observation function such as : $\mathcal{O}(o, a, s'_h, s'_v) = p(o|s'_h, s'_v, a)$;
- $\mathcal{C} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the cost function;
- $b_0 = (s^0_v, b^0_{\mathcal{S}_h})$

where, $b^0_{\mathcal{S}_h} \in \mathcal{B}_h$ is the initial probability distribution over the initial hidden continuous state, conditioned to $s^0_v \in S_v$, the initial fully observable discrete state.
This model differs from the classical MOMDP approach presented in [2] or in [16]. It is important to note that, given the specificity of the model chosen by factorizing the state space into fully observable discrete state variables and hidden (and non observable) continuous state variables, the observation function in our MOMDP path planning model is defined as: $\mathcal{O}(o, a, s'_h, s'_v) = p(o|s'_h, s'_v, a) = 1$ if $o = s'_v$, or 0 otherwise; and so, $\Omega = \mathcal{S}_v$. However, it can be shown that all developments of [2] and [16] still remain valid. It is because $\Omega = \mathcal{S}_v$ climbs into a particular case of factorization already presented in [2], where $\Omega : \Omega_v \times \Omega_h$ is the complete set of observations composed by the observation set for the fully observable state Ω_v and the observation set for the partially state Ω_h. Thereby, in our model $\Omega_v = \mathcal{S}_v$ and $\Omega_h = \emptyset$ (see Eq. (3) of [2] for more details).

3.1 State space

The visible state $s_v \in \mathcal{S}_v$ is defined as a tuple containing: the fully observable boolean state variables for sensors' availability $[0; 1]$, a boolean variable for a collision flag, and the P the localization error covariance matrix propagated by the navigation module in function of a selected navigation mode in a given decision step.
Thus, s_v is define such as $s_v = \{F_{\mathrm{S}_1}, \ldots, F_{\mathrm{S}_N}, F_{\mathrm{Col}}, P\}$. It is assumed that the collision flag F_{Col} is also fully observable by measuring or estimating a force of contact.
The hidden and non observable continuous state $s_h \in S_h$ is defined such as $s_h = x$, recalling that x is the continuous vehicle state vector (see Sec. 2.1).

3.2 Action space

An action $a \in A$ is defined as a tuple $\{d, m_n\}$ where $d \in D$ is the desired direction of motion which specifies $\mathcal{V}_{\mathrm{ref}}$ (see Eq. (8)). D defined here is a finite set of discretized directions. $m_n \in \{S_1 \ldots, S_N\}$ is the navigation mode to be considered depending on the sensors' availability – or navigation solution to be considered during planning depending on sensor selection.

3.3 Transition function

The transition function $T(s_v, s'_v, a, s'_h, s_h)$ is composed of two functions:

- a transition function $T_{\mathcal{S}_h}$ such as:

  $$T_{\mathcal{S}_h}(s_h, s_v, a, s'_h) = f_{s'_h}(s'_h|s_h, s_v, a) \sim N(\bar{s'_h}, \tilde{Q}'(s_v)),$$

  which is based on the GNC closed-loop vehicle motion model, given that the probability distribution of a predicted state s'_h follows a normal distribution $N(\bar{s'_h}, \tilde{Q}'(s_v))$ (see Eq. 10), which in turn, is a function of the previous hidden state s_h, the previous visible state s_v and the action a.

- a transition function $T_{\mathcal{S}_v}$ such as:

  $$T_{\mathcal{S}_v}(s'_h, s'_v) = p(s'_v|s'_h),$$

  which represents the transition function for s'_v and depends on the probabilistic sensors availability maps, and therefore, depends only on the next state s'_h. Concretely,

  $$T_{\mathcal{S}_v}(s'_v|s'_h) = \prod_{i=1}^{N+1} p(s'_v(i)|s'_h) \tag{11}$$

  where, N is the number of sensors, thus $N + 1$ is the number of flags (booleans) in s_v, and $s'_v(i)$ the i-th flag.

Thus, the complete transition function becomes:

$$\begin{aligned} T(s_v, s'_v, a, s'_h, s_h) &= T_{\mathcal{S}_h}(s_h, s_v, a, s'_h) \times T_{\mathcal{S}_v}(s'_h, s'_v) \\ &= p(s'_v|s'_h) f_{s'_h}(s'_h|s_h, s_v, a) \end{aligned} \tag{12}$$

3.4 Cost function

The cost function to be minimized is defined as the vehicle travel (or flight) time plus a cost of collision. It is expected that by minimizing the cost, the algorithm will minimize the flight time (for efficiency) and the probability of collision (for safety) at the same time. More precisely the cost function is defined as :

$$\begin{cases} C(s_t, a_t) = f_t \text{ if } s_t \text{ not in collision} \\ \\ C(s_t, a_t) = K - \sum_{k=0}^{t-1} C(s_k, a_k),\ \forall a_t \in \mathcal{A} \text{ otherwise} \end{cases} \tag{13}$$

where, f_t is the flight time for a given action a at decision step t, and K a fixed cost in case of collision. When a collision occurs, the cost of the any action is a fixed penalty minus the total flight time since the initial state until the collision state. This trick avoids to penalize more if the collision occur after a longer time flight or near the goal. In other words, if the cost of an entire path is defined by the sum of action costs, this cost function equally penalizes all the paths ended up with collision.

3.5 Belief State update

The belief state b represents the probability distribution over states. In this MOMDP model, the belief state can be factorized into a probability distribution over the hidden state space $\mathcal{S}_h$ conditioned on the fully observable state s_v, such as $b^{s_v}_{\mathcal{S}_c} = (b_{\mathcal{S}_c}, s_v)$.
The current belief state is updated after each action a and each perceived visible state $o' = s'_v$ the using the Bayes rule [2]. It allows to update the belief state distribution over the hidden state space. And so, choosing actions during planning based on a complete history information state [5].
In this MOMDP model, the belief state update is decomposed into two functions. The first function corresponds to the GNC closed-loop transition function (Eq. 11 defining a belief state transition related with the action execution error propagation:

$$b_{s'_h}(s'_h) = \int_{\mathcal{S}_h} f_{s'_h}(s'_h|s_h, s_v, a) b_{s_h}(s_h) ds_h \tag{14}$$

The second function is related to the probability of observing s'_v (given by the probability grid maps). This observation probability is computed based on $b_{s'_h}$:

$$p(s'_v|b, a) = \sum_{i=1}^{|G|} p(s'_v|s'_h \in c_i) p(s'_h \in c_i|b, a) \tag{15}$$

where, c_i corresponding to the i^{th} cell of the probability map and $|G|$ is the number of cells in the map.
Finally, the complete belief state update function can be write as :

$$b^{'s'_v}_{s'_h,a}(s'_h) = \frac{p(s'_v|s'_h) \int_{\mathcal{S}_h} f_{s'_h}(s'_h|s_h, s_v, a) b_{s_h}(s_h) ds_h}{\sum_{i=1}^{|G|} p(s'_v|s'_h \in c_i) p(s'_h \in c_i|b, a)} \tag{16}$$

3.6 Value function

The aim of solving a MOMDP consists in finding *a policy* $\pi : \mathcal{B} \to \mathcal{A}$, where $\mathcal{B}$ defines the belief state space, which minimizes a given criterion usually determined by a value function.
The value function $V^\pi(b_0)$ is defined as the expected total cost (weighed by time using γ) the agent will receive from b_0 when following a policy π [9].

$$V^\pi(b) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathbb{E}\left[\mathcal{C}(b_t, \pi(b_t))\right] | b_0 = b \right] \tag{17}$$

where, $C(b, \pi(b) = a) = \sum_{s \in \mathcal{S}} C(s, a) b(s)$ is the expected cost of an action a in the belief state b for the discrete state space case.
The optimal policy π^* is defined by the optimal value function $V^{\pi *}$, such as :

$$V^{\pi^*}(b) = \min_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{E}\left[\mathcal{C}(b_t, \pi(b_t))\right] | b_0 = b \right] \tag{18}$$

Opening the sum in (Eq. 18), it holds to a Bellman's equation, which allows the application of dynamic programming to find the optimal policy. For example:

$$\begin{aligned} V(b) &= \min_{a \in A} \mathbb{E}\left[\mathcal{C}(b, a) + \gamma V(b^{s_v}_a)\right] \\ &= \min_{a \in A} \left[\mathcal{C}(b, a) + \gamma \sum_{s_v \in \mathcal{S}_v} p(s_v|b, a) V(b^{s_v}_a) \right] \end{aligned} \tag{19}$$

When the value (Eq. 19) converges for all belief states, it is possible to extract the optimal policy [9]. Such value and policy computation problem is known to be a hard decision problem (undecidable [14]), in particular given that the belief state space is a continuous space. Recent algorithms, such as SARSOP [13], RTDP-bel [3], or POMCP [18], approach the solution by searching a (partial-)policy only based on reachable belief states, following the MOMDP model dynamics. These algorithms are able to compute sub-optimal solutions in reasonable time.

3.7 Cost function and feasible policies

The MOMDP model presented in this work aims to compute a policy that will minimize the expected flight time and the collision risk, that is, maximize operation efficiency and safety respectively. This policy should be obtained by minimizing the value function defined in the model. Hereafter the value of the initial belief state is redefined in terms of expected flight time and collision risk in the case where $\gamma = 1$. For simplicity, it is assumed that all the action duration coincides with the planning epoch which is constant. This gives a constant f_t for any action a_t, any state s_t at any depth t. It simplifies the collision cost in Eq. 13 to $C(s_t, a_t) = K - t \times f_t = K_t$. Also, this assumption removes the dependency of our cost function Eq. 13 on the action: $C(s_t, a_t) = C(s_t)$.

Redefining the value of the initial belief state in terms of collision risk and flight time. This subsection shows that, with the cost function defined in Eq. 13 and with $\gamma = 1$, the value of the initial belief state $V(b_0)$ can be written as a sum of the expected flight time to reach a goal and the constant collision penalty K weighed by the collision probability. From the definition Eq. 19, $V(b_0)$ can be expanded as follows, knowing the value of any belief state, knowing the value of the collision K, recalling the value of goal state is 0, knowing that the values fully observable

state variable for collision are $F_{Col} = 1$ or $F_{Col} = 0$, and finally assuming $F_{Col_0} = 0$.

$$
\begin{aligned}
V(b_0) &= \min_{a \in A} \left[C(b_0) + \sum_{s_{v_1} \in \mathcal{S}_v} p(s_{v_1}|b_0, a) V(b_a^{s_{v_1}}) \right] \\
&= f_t + \sum_{s_{v_1} \in \mathcal{S}_v} p(s_{v_1}|b_0, a_0^*) V(b_{a_0^*}^{s_{v_1}}) \\
&= \sum_{s_{v_1} \in \mathcal{S}_v} p(s_{v_1}|b_0, a_0^*) \left(f_t + C(b_{a_0^*}^{s_{v_1}}) + \sum_{s_{v_2} \in \mathcal{S}_v} p(s_{v_2}|b_{a_0^*}^{s_{v_1}}, a_1^*) V(b_{a_1^*}^{s_{v_2}}) \right) \\
&= p(F_{Col_k} = 1|b_0, a_0^*) K + p(s_1 \in GOAL|b_0, a_0^*) f_t + \\
&\quad p(F_{Col_k} = 0 \cap s_1 \notin GOAL|b_0, a_0^*) \\
&\quad \left(2 f_t + \sum_{s_{v_2} \in (\mathcal{S}_v \cap F_{Col} = 0)} p(s_{v_2}|b_{a_0^*}^{s_{v_1}}, a_1^*, s_1 \in GOAL) V(b_{a_1^*}^{s_{v_2}}) \right) \\
&= p_{c_1} K + p_{g_1} f_t + \left(p_{c_2} K + p_{g_2} 2 f_t + \left(p_{c_3} K + p_{g_3} 3 f_t + \cdots \right) \right) \\
&= \cdots \\
&= \sum_{t=1}^{\infty} p_{c_t} K + \sum_{t=1}^{\infty} p_{g_t} t f_t = pcK + (1 - pc)T
\end{aligned}
$$

where p_{c_t} is the probability of being the collision state at the depth t when starting from the initial belief b_0 and following the optimal policy. Similarly, p_{g_t} is the probability in reaching at the goal state at the depth t. $pc = \sum_{t=1}^{\infty} p_{c_t}$ is the collision probability at the initial belief b_0 when following the optimal policy. $T =$ is the conditional expected total flight time to reach a goal starting from b_0 knowing that $F_{Col_k} = 0$ for $\forall k \geq 0$. Thanks to our cost function definition (see Eq. 13), the value of the initial belief $V(b_0)$ becomes a linear function of the constant collision penalty cost K with a slope of the collision probability pc. In the following section, this linear dependency will be used to determine the value of the collision penalty K from a given maximum allowable risk of collision.

Maximum allowable collision risk. Let us consider the following three extreme navigation policies;

- The most efficient (heuristic) policy (π_h): which minimizes the expected total flight time to the goal with considering neither the initial state uncertainty nor the collision risk due to the localization and path execution uncertainty. As it comes from the minimization of the total flight time only, the expected flight time T_h resulted from this heuristic policy is the shortest possible flight time of a given mission, and so $T_h \leq T$ is guaranteed. The collision probability p_h resulted from this policy could be high up to 1.

- The safest policy (π_s): which minimizes the expected flight time while not allowing any collision risk under the uncertainties. This policy does not necessarily exist for a given mission, but here we assume it does. Then the collision probability resulted from this policy should be 0. The resulting total flight time gives an upper-bound of T, because the optimal cost $V^{\pi^*}(b_0) = pcK + (1 - pc)T \leq V^{\pi}(b_0)$ for $\forall \pi$ and this safest policy gives $V^{\pi_s}(b_0) = T_{max}$.

- The collision policy (π_c): is a policy which always brings a vehicle in a collision state, i.e., the resulting collision probability is 1.

As derived in the previous section, the value of the initial belief state is linearly dependant on the collision penalty K and its slope is given by the collision probability. Figure 2 plots the values of these three extreme policies ($V^{\pi_h}(b_0)$, $V^{\pi_s}(b_0)$, $V^{\pi_c}(b_0)$ respectively) versus the collision penalty cost K. Since any policy line $V^{\pi}(b_0) = pcK + (1 - pc)T$ intersects with the collision policy line (Fig. 2 —■—) at $K = T$, the collision penalty should be chosen as $K > T_{max}$ so that the collision policy never becomes optimal.
Now, let us consider a maximum allowable collision probability p_{thd} given as a mission criteria. That is, the optimal navigation policy is required to have $pc \leq p_{thd}$ for being acceptable. From the condition $K > T_{max} \geq T$, for any feasible policy with $pc \leq p_{thd}$, the following is satisfied.

$$V^{\pi}(b_0) = pcK + (1 - pc)T \leq p_{thd}K + (1 - p_{thd})T$$

On the other hand, as the heuristic policy (π_h) gives the lower-bound of the expected total flight time, the right-hand side of the above inequality is lower-bounded by $p_{thd}K + (1 - p_{thd})T_h$. Hence, it can be said that if $V^{\pi}(b_0) \leq p_{thd}K + (1 - p_{thd})T_h$, the policy π is guaranteed to satisfy the maximum allowable collision risk. That is, the line corresponding to $p_{thd}K + (1 - p_{thd})T_h$ (Fig. 2 —○—) gives the upper threshold of the optimal policy line to be feasible.
At the same time, the safest policy gives the upper-bound of the value of the initial belief of any possible optimal policy (regardless of its feasibility), such as $V^{\pi^*}(b_0) \leq V^{\pi_s}(b_0) = T_{max}$. Therefore, if we choose the value of the collision penalty K as at which the safest policy line (Fig. 2 —●—) and the line of the upper threshold of the feasible policy (Fig. 2 —○—) intersect, it is guaranteed that any optimal policy satisfies

$$V^{\pi^*}(b_0) \leq p_{thd}K + (1 - p_{thd})T_h = T_{max} = V^{\pi_s}(b_0)$$

and hence $pc \leq p_{thd}$. This value of K is derived as follows.

$$K(T_{max}, T_h, p_{thd}) = T_h + \underbrace{\overbrace{(T_{max} - T_h)}}_{p_{thd}} \qquad (20)$$

4 POMCP-like algorithm

Keeping and updating the belief states can be a challenging and computationally expensive step during problem resolution due to the potential complexity of the belief state update function (Eq. 16). In particular in the problem addressed in this paper, as the hidden state space is continuous and the fully observable state space is discrete, the computation of the probability distribution over $\mathcal{S}_c$, and correction by s_v would

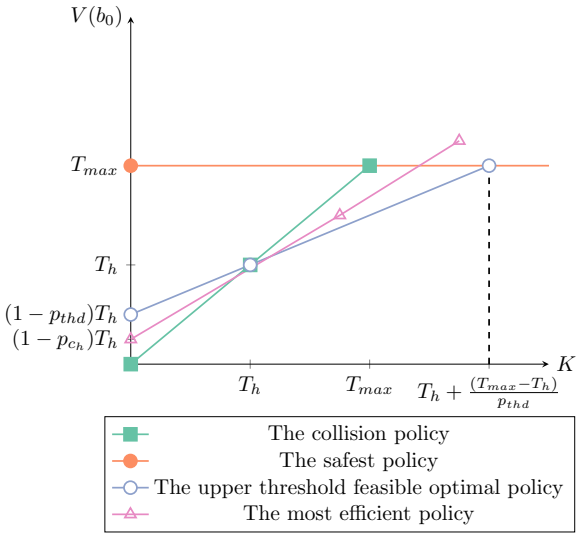


Figure 2: Representation of $V(b_0)$ in terms of flight time and maximum allowable collision risk

require expensive computational effort that could be bypassed using approximations [6, 7].

An interesting solution is to use algorithms that do not need to maintain and update the belief state in each decision step. In this sense, resolution approaches like POMCP algorithm [18] are promising. POMCP is a Monte-Carlo Tree Search algorithm for partially observable environments. POMCP works by sampling a state s from the initial belief state b_0, and simulating sequences of action-observation (by a trial procedure) to construct a tree of belief nodes. Each tree node h represents an history of action-observation pairs since the initial belief state (being a belief node h). POMCP calculates for a given node h of the tree the average cost observed for all trials that have started from this node. POMCP does not update the belief state after each action-observation pair but update the average cost for each node of the tree called, and keeps in memory the number of times a node was explored $N(h)$, and the number of times a given action was chosen $N(ha)$ in this node.

This procedure (trials) allows to POMCP to approach the value function and policy for a given node (a belief state represented by an history). During value and policy computation, trials are performed in a greedy way. The most promising action, following a given heuristic is chosen in each tree node. More precisely, in each node the greedy action selection strategy is based on the same heuristic choice as UCT [11], the UCB1 (Upper Confidence Bound 1) action selection strategy.

This action selection strategy is based on a combination of two characteristics, the action Q-value and a measure of how well explored an action is (how well estimated is its value).

The action's Q-value can be defined as:

$$Q(b,a) = \left[\mathcal{C}(b,a) + \gamma \sum_{s_v \in S_v} p(s_v|s,a) V(b_a^{s_v})\right], \quad (21)$$

being the value of performing an action a in the belief state $b = (s_v, b_{\mathcal{S}_h})$ – one-step lookahead computation of the value –, supposing that the optimal policy will be performed after. Note V is the true expected cost to reach the goal [12]. Because POMCP works by simulating sequences of action-observation each action Q-value is regularly updated, which allows to estimate the value of $V(b) \leftarrow \min_{a \in A} Q(b,a)$.

The second characteristic of this greedy action selection strategy used during policy computation is a measure (given by $c\sqrt{\frac{\log N(h)}{N(ha)}}$) of how well-explored the action a is, given the history h (or belief node). This measure is used to balance the action choice in the algorithm, which will select the action that minimizes $Q(h,a) - c\sqrt{\frac{\log N(h)}{N(ha)}}$. In other words, the algorithm will select the action that minimizes the regret of choosing a wrong action [11].

The exploration coefficient c is used to force the algorithm to try actions that seem a priori less interesting for avoiding to falling into a local optimum policy. If c is high the algorithm will try more often the actions that seem less interesting, on the contrary, if c is low the algorithm will rarely explore different actions.

However, due to the particularities of the problem addressed in this work, the POMCP algorithm was modified, and is presented on Alg. 1. As in the classical POMCP algorithm, it is starting with an initial belief state b_0 and an empty history h (line 5) that will be initialized to b_0, then the algorithm will expand the tree for a given number of trials. The principal differences between the classical POMCP and the algorithm here presented are hereafter discussed.

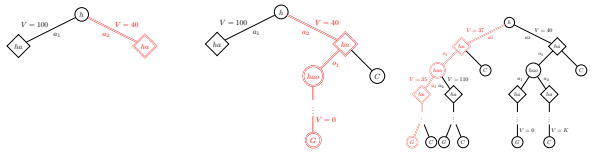
The classical POMCP algorithm estimates the value of a tree node by selecting an action using the UCB1 greedy action selection until a new node is reached. When a new node is created a *rollout* method is used to estimate the value of the new node, that simulates and evaluates sequences of random action-observation pairs starting from this new node. And, then POMCP backtracks this value until the initial belief node and starts a new trial.

The algorithm here presented estimates the value of a tree node by selecting an action using the UCB1 greedy action selection until *a final state is reached* (goal G or collision C). If a new node is discovered (this history is not yet in the tree), its Q-value and value are estimated using an initial heuristic value hereafter presented, and not a *rollout* policy.

The belief state value initialization considered for this work explores the A* shortest path solution consid-

Algorithme 1 : POMPC-like

Function *POMPC(h, b_0, c, γ)*
  $h \leftarrow b_0$
  while $nbTrial < Nb_{max}$ **do**
    $s_h \leftarrow$ sampling s_h from b_0
    Trial(h, s_h, c, γ)
  $a^* \leftarrow \underset{a \in A}{\operatorname{argmin}} V(b_0)$
Function *Trial(h, s_h, c, γ)*
  if $s_h \in Goal$ **then**
    return 0
  if $h \notin T$ **then**
    for $a \in A$ **do**
      for $s_v \in S_v$ **do**
        $hao \leftarrow h + a + s_v$
        $T(hao) \leftarrow (N_{\text{init}}(hao), V_{\text{init}}(hao), \emptyset)$
  $\bar{a} \leftarrow \underset{a \in A}{\operatorname{argmin}}\, Q(s_h, a) - c\sqrt{\frac{\log N(h)}{N(ha)}}$
  $ha \leftarrow h + \bar{a}$
  $s'_h, s_v \sim \mathcal{G}(s_h, \bar{a})$
  $hao \leftarrow ha + s_v$
  $Trial(hao, s'_h, c, \gamma)$
  $N(h) \leftarrow N(h) + 1$
  $N(ha) \leftarrow N(ha) + 1$
  $Q(h, \bar{a})' \leftarrow C(s_h, a) + \gamma \sum_{s_v \in S_v} p(s_v|s_h, \bar{a}) V(hao)$
  $Q(h, \bar{a}) \leftarrow Q(h, \bar{a}) + \frac{Q(h,\bar{a})' - Q(h,\bar{a})}{N(ha)}$
  $V(h) \leftarrow \min_{a \in A} Q(h, a)$

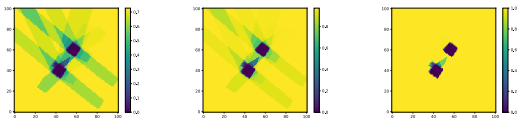


(a) History tree at the beginning of the POMCP. (b) History tree after the first trial. (c) History tree after N trials.

Figure 3: Evolution of the history tree during the value optimization.

ering only the obstacles grid map and computes the estimated flight time. This can be pre-calculated and hence save time during value and policy optimization process. Note that this value initialization gives a more informative value approximation in this goal-oriented path planning problem (for a given state in a given grid cell) compared to a *rollout* policy being, in this sense, preferable.

Figure 3 shows the different steps of the algorithm. Figure 3a is the initializing of the algorithm with an empty history and the estimated value of the two possible actions. Figure 3b shows an example after the first trial where the action a_2 has been selected at the beginning, and stopping only when the goal G has been reached. The policy after the first trial is represented in red. And, figure 3c shows what could happen after a number Nb of trials with the new policy that starts with the action a_2 that was not interesting at the beginning of the example.



(a) 1 meter precision. (b) 2 meters precision. (c) 5 meters precision.

Figure 4: Availability maps of the GPS in function of the minimum precision required.

5 Simulation results

5.1 Configuration

POMDP has been tested on a benchmarking framework for UAV obstacle field navigation[1] proposed in [15], which provides environments with different obstacle configurations. The selected benchmark "Cube baffle" contains two cube obstacles with a grid size of $100 \times 100 \times 20$ cells, where each grid cell has the size of $2\text{m} \times 2\text{m} \times 2\text{m}$.

In the following tests, two onboard sensors are considered: INS and GPS. While INS is known to be available anywhere, GPS is not. However, a priori knowledge on the GPS availability is supposed. Probability grid maps for GPS availability were created based on a DOP map calculated with a GPS simulator for a given geolocation and time, for different GPS precision thresholds (1, 2 and 5 meters, Fig. 4). These thresholds give different probabilistic availability maps. For instance, if the 1-meter GPS availability map indicates 60% probability in a given cell, it means that there is 60% of chance that GPS will be available with 1-meter of precision in this cell. Note that, these precision thresholds were only used to generate different GPS availability maps, but not used in the Kalman filter. These maps were created with the aim of comparing the effect of considering the GPS availability probability during planning on the resulting navigation policy for different environments.

In the following experiments, the planning objective was to find a policy allowing at most 10% of collision ($p_{thd} = 0.1$). For each of the three GPS probabilistic availability maps, the first set of optimization has been run to compute the safest policy with 0% collision risk (considering a prohibitive collision cost $\bar{K}$). This computation allows to approximate the lowest (i.e. the tightest) expected flight time T without collision. Thus, this expected value is used as T_{max} to define the collision penalty by Eq. 20. Then, a new set of optimization has been launched to compute another policy using this new collision penalty so that the resulting policy will respect the given admissible collision risk threshold.

Moreover, to compare the performance of policies obtained in these stochastic problems being solved with

[1] benckmark framework from: `www.aem.umn.edu/people/mettler/projects/AFDD/AFFDwebpage.htm`

POMCP-like algorithm, five distinct policy optimization processes, with 100000 trials each (see Alg. 1), have been run for each probabilistic GPS availability map. In addition to that, for each run, the policy being optimized was evaluated for 1000 simulations after each 5000 trials. For these experiments γ was set to 1. Note that this γ value does not prevent value and policy convergence, given that the stochastic shortest path problem here addressed respects all convergence assumptions (see [12] and [11] for more details).

5.2 Results

To highlight the contribution of using a path planning model like a MOMDP in an environment with uncertainty on the availability of the onboard sensors, the well-known A^* algorithm has been used to compute the shortest path ignoring the uncertainties in environment (sensor availability) as well as in the vehicle state transition. This *heuristic* policy chooses the best action to follow only based on the A^* shortest path from the current position. Then 1000 simulations have been run on each GPS availability map. Figure 8 shows among other things, the results of these simulations. It shows that even for the easiest environment with highest probability of GPS being available(i.e. the 5-meter GPS availability map), the simulations have only a success rate of 66%. This is because the uncertainties in the environment and in the vehicle state transition are ignored in the A^* planning task. In the following, we'll show that this success rate can be improved by the proposed path planner and hence the vehicle operation safety can be enhanced.

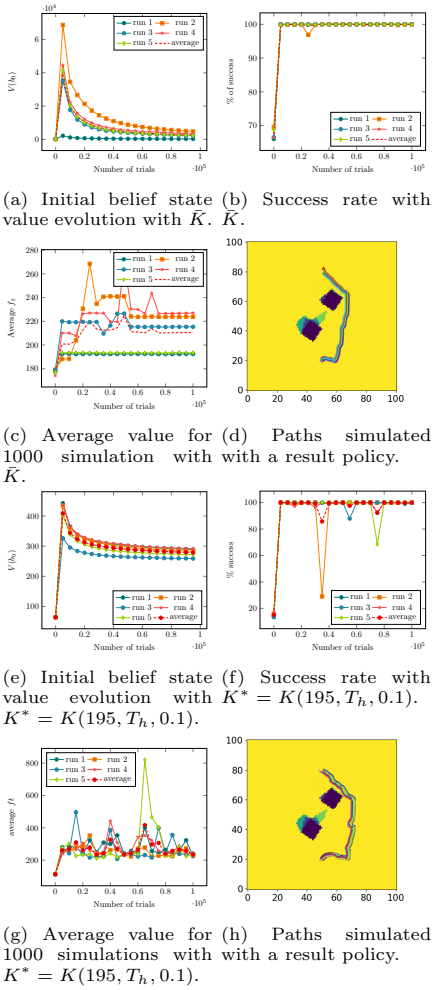
Figures 5, 6 and 7 present the averaged results for five runs of the POMCP algorithm (offline runs) for the three different probabilistic GPS availability maps respectively. The first row figures are the results of the safest policy with $\bar{K} = 10^6$ allowing to defining the tightest T_{max}. The second row are those of the policy calculated for $K^* = K(T_{max}, T_h, p_{thd})$, where T_h is the heuristic flight time given by the A^* algorithm and $p_{thd} = 0.1$.

The figure 5 shows the results with the 5-meter precision GPS availability map (Fig. 4c). This is the map where GPS is the most likely available: when less GPS precision is required more likely GPS availability is.

It is reminded that T_{max} value is obtained by the result from the firstly calculated safest policy. Figures 5a and 5e show the evolution of the initial belief state value. In this case, the results with $\bar{K}$ and $K^* = K(T_{max} = 195, T_h, p_{thd} = 0.1)$) show that after 100000 trials $V(b_0)$ has almost converged - the final initial belief state value is afterward the same for the 5 runs in both cases. Figures 5b and 5f show success rate for each computed policy, reaching 100%, and respecting the collision risk threshold of 10% required.

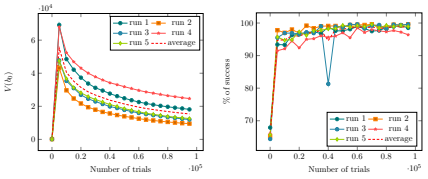
However, with the 2-meters probabilistic GPS availability map (Fig. 4b), the results are different (see Fig. 6). Figure 6a shows that the value of the initial belief state for $\bar{K}$ has not converged to a same value on these five runs, even after 100000 trials. Thereby, the result of the run 2, who has the highest success rate is the success, was used to define $T_{max} = 245$ for computing K^*. For this $K^*(T_{max} = 245, T_h, p_{thd} = 0.1)$, the value of b_0, shown in Figure 6e, decreases with 100000 trials reaching almost the same value among runs (except for run 4). The corresponding evolution of the success rate is shown in Figure 6f. It reaches 90%, respecting the maximum allowable collision risk fixed to 10% (except for run 4). The exception of the run 4 can be explained by the fact that even 100000 trials of POMCP-like search could be not enough to converge to a minimum value, and so, not respecting the expected success rate.

The results with the 1-meter precision probabilistic GPS availability map (Fig. 4a) reflect a more difficult optimization problem due to the increase in the uncertainty on sensors availability (Fig. 7). Figure



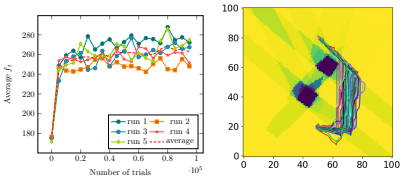
(a) Initial belief state value evolution with $\bar{K}$.
(b) Success rate with $\bar{K}$.
(c) Average value for 1000 simulation with $\bar{K}$.
(d) Paths simulated with a result policy.
(e) Initial belief state value evolution with $K^* = K(195, T_h, 0.1)$.
(f) Success rate with $K^* = K(195, T_h, 0.1)$.
(g) Average value for 1000 simulations with $K^* = K(195, T_h, 0.1)$.
(h) Paths simulated with a result policy.

Figure 5: Obtained results in function of the number of trials for 5-meters GPS precision probabilistic availability map.



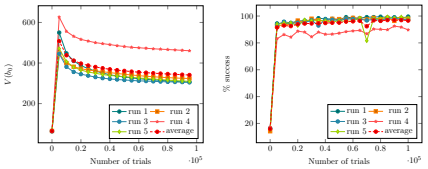
(a) Initial belief state value evolution with $\bar{K}$.

(b) Success rate with $\bar{K}$.



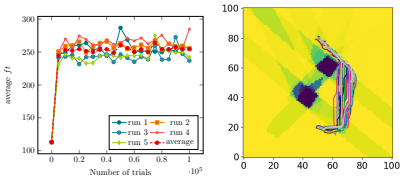
(c) Average value of the 1000 simulations with $\bar{K}$.

(d) Paths simulated with with a result policy.



(e) Initial belief state value evolution with $K^* = K(245, T_h, 0.1)$.

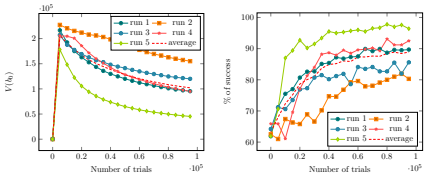
(f) Success rate with $K^* = K(245, T_h, 0.1)$.



(g) Average value of 1000 simulation with $K^* = K(245, T_h, 0.1)$.

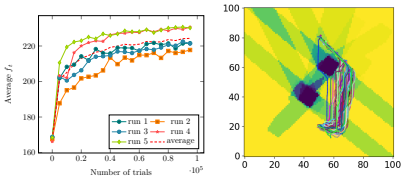
(h) Paths simulated with with a result policy.

Figure 6: Obtained results in function of the number of trials for 2-meters GPS precision probabilistic availability map.



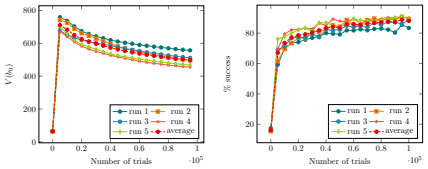
(a) Initial belief state value evolution with $\bar{K}$.

(b) Success rate with $\bar{K}$.



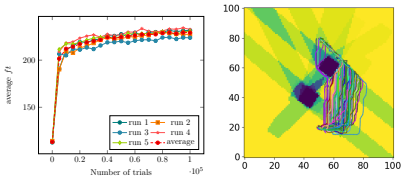
(c) Average value for 1000 simulations with $\bar{K}$.

(d) Paths simulated with with a result policy.



(e) Initial belief state value evolution with $K^* = K(230, T_h, 0.1)$.

(f) Success rate with $K^* = K(230, T_h, 0.1)$.



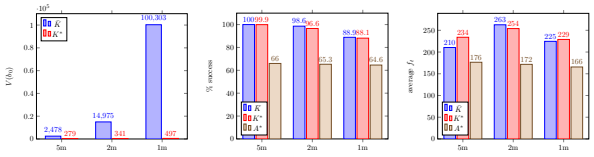
(g) Average value for 1000 simulations with $K^* = K(230, T_h, 0.1)$.

(h) Paths simulated with with one policy.

Figure 7: Obtained results in function of the number of trials for 1-meter GPS precision probabilistic availability map.

7a shows that, except for run 5, it needs more than 100000 trials to decrease (minimize) the initial belief state value when using $\bar{K}$. The success rates of these runs do not reach 100%. Indeed, the result of run 5 reaches the highest success rate, and so $T_{max} = 230$ from run 5 was used for the collision penalty computation, as a rough estimation of the expected flight time of the safest policy. Figure 7e shows that the majority of the runs are going to converge to the same expect initial belief state value. Also, the smooth increase in the success rates is shown in Fig. 7f. Note that, even if the value, and the related policy, have not converged, the maximum collision risk of 10% is near to be respected. In this sense, if more trials are performed during policy optimization, one could expect to reach the minimum value and to respect the collision risk defined.

Figure 8 summarizes the results presented with three bar plots. The first one (Fig. 8a) shows the average initial belief state value $V(b_0)$ after 100000 trials. It shows that less likely the GPS is available, higher is $V(b_0)$. The second bar plot (Fig. 8b) presents the average success rate. With 5-meters probabilistic GPS availability map, the initial belief state value has almost converged for both cases, thus both policies respect the success rates imposed (e.g. 100% for $\bar{K}$ and, for K^*, 99.9% being superior to 90%). For K^* in the 2-meter probabilistic GPS availability map, the success rate is above 96% which also respects the given collision risk threshold. However, for K^* in 5 and 2-meters cases, the average flight time is still nearly T_{max} (Fig. 8c). It is expected to be further improved (at a price of taking more collision risk), if we perform more trials in the optimization process. Similarly, for the 1-meter probabilistic GPS availability map, $V(b_0)$ did not converge and consequently, the success rate is below the expected. Moreover, the average flight time is lower than the 2-meters GPS availability map, because the algorithm did not have enough trials to optimize the success rate, taking the more uncertain (risky) but shorter path.



(a) Comparison of the initial belief state value in function of collision cost.

(b) Comparison of the success in function of the collision cost and the GPS availability maps.

(c) Comparison of the average flight time.

Figure 8: Summary of the initial belief state value and success rates in function of the collision cost.

Some conclusions can be extracted from these results. Firstly, the algorithm needs more time to converge when the uncertainty on the GPS availability grows. Indeed, when the GPS availability is less likely the UAV need to rely more on the INS measurement which leads to more risk of collision. Thus, to optimize the policy the algorithm needs more trials (more exploration) to find a policy with a minimum collision risk. Secondly, even when the GPS availability is less likely, the algorithm can compute a policy with a good success rate (above $88,1\%$) for this benchmark environment.

6 Conclusion and future work

This paper presents a MOMDP framework to model and solve the safe path planning problem in urban environments. Comparative results show the impact of the probabilistic GPS availability and collision costs on the safety of the path. Moreover, this paper proposes a simple method to impose a maximum allowable flight time and a collision risk threshold in order to compute a feasible policy. Indeed the number of trials necessary to value and policy convergence in the more uncertainty probabilistic sensor maps is important. However, the results show that it is possible to take into account the probabilistic availability of the onboard sensors in the planning process. Moreover, the results also show that in the considered benchmark map with important uncertainty on the sensors availability, significant solutions can be found even if the policy has not converged. Thereby, more evaluations are necessary, in particular to evaluate the proposed approach in real urban environments [15]. Further work will study an online optimization process given that the sensors probabilistic availability map evolves with time.

References

[1] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart. Motion- and uncertainty-aware path planning for micro aerial vehicles. *Journal of Field Robotics*, 2014.

[2] M. Araya-López, V. Thomas, O. Buffet, and F. Charpillet. A closer look at momdps. In *22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2010.

[3] B. Bonet and H. Geffner. Solving pomdps: Rtdp-bel vs. point-based algorithms. In *IJCAI*, 2009.

[4] A. Bry and N. Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[5] O. Buffet and O. Sigaud. Processus décisionnels de markov en intelligence artificielle, 2008.

[6] J.-A. Delamer, Y. Watanabe, and C. P. C. Chanel. Momdp solving algorithms comparison for safe path planning problems in urban environments. In *9th Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, 2017.

[7] J.-A. Delamer, Y. Watanabe, and C. Ponzoni Carvalho Chanel. Towards a momdp model for uav safe path planning in urban environment. In *International Micro-Air Vehicle Competition (IMAV)*, 2017.

[8] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 2010.

[9] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 1998.

[10] F. Kleijer, D. Odijk, and E. Verbree. Prediction of gnss availability and accuracy in urban environments case study schiphol airport. In *Location Based Services and TeleCartography II*. Springer, 2009.

[11] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *ECML*, 2006.

[12] A. Kolobov. *Planning with Markov decision processes: An AI perspective*, volume 6. Morgan & Claypool Publishers, 2012.

[13] H. Kurniawati, D. Hsu, and W. S. Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

[14] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. In *AAAI/IAAI*, pages 541–548, 1999.

[15] B. Mettler, Z. Kong, C. Goerzen, and M. Whalley. Benchmarking of obstacle field navigation algorithms for autonomous helicopters. In *66th Forum of the American Helicopter Society:" Rising to New Heights in Vertical Lift Technology", AHS Forum 66*, 2010.

[16] S. C. Ong, S. W. Png, D. Hsu, and W. S. Lee. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 2010.

[17] S. Prentice and N. Roy. The belief roadmap: Efficient planning in linear pomdps by factoring the covariance. In *Robotics Research*, pages 293–305. Springer, 2010.

[18] D. Silver and J. Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, 2010.

[19] H. W. Sorenson. *Kalman filtering: theory and application*. IEEE, 1985.

[20] Y. Watanabe, S. Dessus, and S. Fabiani. Safe path planning with localization uncertainty for urban operation of vtol uav. In *AHS Annual Forum*, 2014.

Apprentissage d'un Modèle d'Interaction Homme-Agent par Production Participative

Anonyme

[1] Laboratoire

nicolas.drougard@isae-supaero.fr

Résumé

Les missions impliquant des êtres humains interagissant avec des systèmes automatisés deviennent de plus en plus courantes. En raison du comportement non déterministe de l'être humain et du risque élevé de défaillance dû à des facteurs humains, un tel système intégré devrait réagir intelligemment en adaptant son comportement si nécessaire. Une avenue prometteuse pour concevoir un système efficace pour améliorer l'interaction est le paradigme de l'initiative mixte. Dans ce contexte, cet article propose une méthode pour apprendre le modèle d'une mission humain-robot à initiative mixte. La première étape pour mettre en place un modèle fiable est d'acquérir suffisamment de données. Dans ce but, une campagne de production participative a été menée et des algorithmes d'apprentissage ont été entrainés sur les données collectées afin de modéliser la mission human-robot et d'optimiser une politique de supervision du processus décisionnel de Markov (MDP) associé. Ce modèle prend en compte les actions de l'opérateur humain pendant l'interaction ainsi que l'état du robot et de la mission. Une fois qu'un tel modèle a été appris, la stratégie de supervision peut être optimisée en fonction d'un critère représentant l'objectif de la mission. Dans cet article, la stratégie de supervision concerne le mode de fonctionnement du robot. Des simulations basées sur le modèle MDP montrent que la planification dans l'incertain peut être utilisée pour adapter le mode du robot en fonction de l'état du système robotique humain. L'optimisation du mode de fonctionnement du robot semble pouvoir améliorer la performance de l'équipe. L'ensemble des données provenant de la production participative est donc un matériel qui peut être utile pour la recherche en interaction homme-machine, c'est pourquoi il a été rendu disponible sur notre site web.

Mots Clef

Interaction homme-robot, Mission à Initiative Mixte, Production Participative, Apprentissage de Chaîne de Markov, Processus Décisionnel de Markov, Classification

Abstract

Missions involving humans interacting with automated systems become increasingly common. Due to the non-deterministic behavior of the human and possibly high risk of failing due to human factors, such an integrated system should react smartly by adapting its behavior when necessary. A promise avenue to design an efficient interaction-driven system is the mixed-initiative paradigm. In this context, this paper proposes a method to learn the model of a mixed-initiative human-robot mission. The first step to set up a reliable model is to acquire enough data. For this aim a crowdsourcing campaign was conducted and learning algorithms were trained on the collected data in order to model the human-robot mission and to optimize a supervision policy with a Markov Decision Process (MDP). This model takes into account the actions of the human operator during the interaction as well as the state of the robot and the mission. Once such a model has been learned, the supervision strategy can be optimized according to a criterion representing the goal of the mission. In this paper, the supervision strategy concerns the robot's operating mode. Simulations based on the MDP model show that planning under uncertainty solvers can be used to adapt robot's mode according to the state of the human-robot system. The optimization of the robot's operation mode seems to be able to improve the team's performance. The dataset that comes from crowdsourcing is therefore a material that can be useful for research in human-machine interaction, that is why it has been made available on our web site.

Keywords

Human-Robot Interaction, Mixed-Initiative Mission, Crowdsourcing, Markov Chain Learning, Markov Decision Process, Classification

1 Introduction

Human and artificial agent interaction is an actual research track that covers various disciplines. Artificial intelligence, human factors and sociology are few examples of involved topics. Due to the increase of the decisional autonomy of artificial agents, *e.g.* robots, autonomous cars and unmanned aerial vehicles (UAVs), the role of the human operator is reduced regarding direct control, and concentrated on higher level decisions, that are not automated for practical, ethical or legal reasons. The use of automated plan-

ning for artificial agents actions has been amplified by the recent technical advances in artificial intelligence and machine learning. As an example, convolutional networks led to artificial vision [31] and popularized deep learning techniques, which played an important role in the latest successes of decision making algorithms based on Reinforcement Learning [33] and Planning under Uncertainty [18].

However, human operators are still vital in numerous scenarios. In particular they are able to produce tactical, moral, social and ethical decisions [22]. Such decisions are not (yet) assigned to machines. For instance, legal regimes need people for responsibility assessment issues, encouraging human supervision of automated systems.

On the other hand, this drastic change of the human operator role in favor of system's autonomy results in a new paradigm also known as mixed-initiative [15]. Mixed-initiative human-robot interaction considers human operators and artificial agents as a team [22], in which each agent can seize the initiative from the other. From the human operator's point of view it is not always bearable or acceptable that such an artificial system could seize the initiative, except if human cognitive capabilities or performance are degraded. A study reports that human factors are involved in 80% of autonomous aerial vehicles accidents [36]. This fact is due to several constraints experienced by humans during their missions. Stress, high workload, fatigue or boredom, which can be induced respectively by pressure (e.g. cause by a danger), complexity, hardness or duration of their tasks, are some of the main problems encountered by humans. As a result, an intelligent supervision system could lend a strong hand in order to help the human operator and the human-robot team to perform better.

Hence, an appropriate supervision strategy has to manage the information given to the human operator, the task allocation between the human and the machine, as well as the machine policy during its own tasks. In other words, the supervision strategy goal is to drive human-machine teams firstly by allocating the tasks that can be carried out by both the human and the machine, secondly, by providing, or not, appropriate alarms to the human operator, and finally by adapting actions of the machine according to the human behavior.

Considering that the human behavior is not deterministic, as well as environment dynamics, events occurring in a mixed-initiative mission can be considered as uncertain. A classical automated planning framework for probabilistic domains can be used to handle such a mixed-initiative human-robot interaction problem : the Markov Decision Processes (MDPs) [25]. MPDs allow to define the goal of a given mission in terms of rewards valuating states of the system. The optimization of MDPs consists in computing a strategy maximizing the expected sum of rewards over time [2].

The drawback of using MDPs is the need for a precise transition model, the which must faithfully represent the dynamics of the system. On one hand, Reinforcement Learning (RL) [35] can be explored in cases only a generative model is available to learn the optimal actions during repetitive mission realizations. On another hand, if a sufficient number of missions have been carried out before hand, it is possible to learn the MDP's parameters, and so, MDP optimization algorithms can be applied to obtain the optimal strategy.

This work is built based on this second approach. For this purpose, a mixed-initiative mission, called *Firefighter Robot* game [8], has been designed to reveal some general problems that occur in human-machine interaction. This mission, which simulates a remotely operated robot, is available on an opened website [1]. Advertising was done in the authors' (professional and social) networks to encourage Internet users to carry out the mission in order to collect as much anonymous data as possible. This crowdsourcing platform has collected more than a thousand mission realizations, allowing the application of machine learning techniques, namely classification and Markov Chain learning, to define the parameters of the MDP that models the mission.

This paper is organized as follows : firstly the designed mixed-initiative mission available on the crowdsourcing platform is presented as well as the produced dataset. Then the methodology used to learn a human-robot interaction model is given, followed by a complete mission model definition in the form of an MDP. Finally simulations are performed, evaluation results are presented and future work is discussed.

2 Crowdsourcing for massive data collection

As explained earlier, a crowdsourcing platform has been set up to allow users to perform the mission called Firefighter Robot game. This human-robot mission immerses the user in a scenario where he plays a fireman who must cooperate with a robot that is present in a small area with few trees. These trees have a weird tendency to self-ignite for some unknown reason. Through the graphical user interface (GUI) shown in Figure 1, the human operator gets the position of the robot in a map (bottom center), as well as the video streaming from its camera (top right).

The battery charge level of the robot decreases with time. However, when the robot is in the charging station, represented by a red square on the ground, the battery recharges. If the battery is empty and the robot is not on the red square, the mission fails and is finished. All the information related to the robot is summarized at the bottom right of the GUI.

The volume of water contained by the robot is not unlimited : to recharge in water, the robot has to be in the water station represented by a blue square on the ground and the associated tank has to contain enough water. For that, the human operator has to fill this ground tank using the buttons on the left-side of the interface : a tap, which can

1. `http://robot-isae.isae.fr`

move horizontally by actuating a wheel (top buttons), fills the tank when it is in the middle (which is an unstable equilibrium). To actually fill the tank, the button bellow (black tap) turns on the tap for few seconds. Leaks may appear on the tank during the mission causing it to lose water : the button below (black wrench) can be used to fix them.

With the help of this robot, the goal of the mission is to fight as many fires as possible in a limited amount of time (ten minutes). The temperature of the robot increases when it is too close to flames and the mission terminates when it is too hot. The presence of fires is supposed to be felt as a danger by the operator. The robot, when its mode is "manual", is controlled by the arrows (navigation) and the space bar (shoot water) of the keyboard. In "autonomous" mode, the robot drives itself with a hard-coded strategy, including shooting water and the recharge of water or battery when necessary.

For model learning purposes, the robot's operating mode can change randomly every ten seconds : it can be autonomous or it can need for manual control. This uniform sampling technique is used in order to get a balanced dataset (as much data under both supervision actions) needed to learn model probabilities for each action. Like the robot mode, the display of an alarm is also considered as a supervision action. When an alarm can be displayed, *i.e.* when the information it provides is true, the action of displaying this alarm is also randomly selected, in order to get data from both conditions (with and without alarms). The complete list of alarms is given in Table 1.

Temperature, battery and external tank management, as well as the score (number of extinguished fires) and the remaining time should imply stress and pressure. Pretests revealed that both tasks (robot control and water management inspired from MATB [6]) are complex enough to generate cognitive workload in the human operator, notably highlighted by variations in her engagement [9]. Such a (degraded) mental state can impair the human operator's cognitive abilities, and thus increase the risk of mission failure.

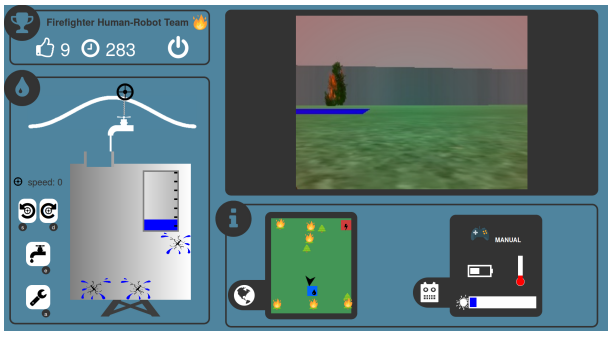


FIGURE 1 – Graphical user interface (GUI) of the Firefighter Robot mission. The score and remaining time are displayed at the top left and the video from the robot is available at the top right, above the robot position and status information. The water reserve management task is displayed at the bottom left of the interface.

Integer	Associated alarm
-1	No alarm displayed
0	"Low battery"
1	"Too-high temperature"
2	"Less than one minute before the end of the mission"
3	"The robot's tank will soon be empty (2 shoots left)"
4	"The robot is in autonomous mode"
5	"The robot is in manual mode"
6	"The ground tank's water level is low"

TABLE 1 – The different alarms that can be displayed during the mission as well as the associated integers, *i.e.* the numbers representing each of the alarms in the dataset.

2.1 A Human-Robot Mission Dataset

The total time of recorded missions reaches more than 85 hours. In addition, in at least 55% percent of the whole samples, the human operator interacts with the interface. For instance she clicks on the interface, or uses the keyboard. Note that, each sample represents the amount of data collected during one second.

All anonymous data collected is available online [2]. The history of each mission is saved in a comma-separated values (CSV) file *i.e.* in the form of a table. The system status, that is the mission context and human-system interaction data, is recorded at every second. The first line of the file is always the same since the mission always starts in the same initial state. Thus, the information actually generated by the human-machine system goes from line 2 to line $H+1$ (line 601 if no premature game over occurs) where $H \in \{1, \ldots, 600\}$ is the process horizon.

The remaining mission time r_t in seconds is the first column and takes values from 600 to $600-H$. If $H = 600$ the mission is completed until the end and without premature game over. Since the missions are not always successful, not all missions have the same duration.

The next two columns correspond to the potential actions of the human-robot team's supervision system. Indeed the second column contains the different operating modes taken by the robot during the process : $a_t^m = 1$ if the robot is autonomous during the second t, and $a_t^m = 0$ if it is in manual mode. The third column contains the alarms that can be triggered during the mission and is denoted by a_t^a. Please refers to Table 1 for alarm encoding details.

The next three columns describe the robot pose : $\forall t \in \{0, \ldots, H\}, (x_t, y_t, \theta_t) \in ([-20, 20]^2 \times]-\pi, \pi])$. For instance, on the map such as displayed on the GUI, or in Figure 2, the x_t (resp. y_t) increases when the robot moves to the right (resp. goes up the map). The angle θ_t is zero when

2. https://personnel.isae-supaero.fr/isae_ressources/caroline-chanel/horizon/

the robot is oriented to the right and grows in the trigonometric direction.
Then comes the column describing the condition of the trees over time, *i.e.* which trees are on fire and which are not. A number is assigned to each of the nine trees and their coordinates are given in Table 2. By noting $f_t^i \in \{0, 1\}$ the state of the tree i (1 for *on fire* and 0 otherwise) at time $t \in \{0, \ldots, H\}$, the value given in the CSV file is the forest state $f_t = \sum_{i=1}^{9} f_t^i \cdot 10^{9-i}$. Thus, the resulting binary number at i^{th} digit, beginning from the left, denotes the state of tree f_t^i.
The next columns are dedicated to battery level $b_t \in [0, 100]$ and temperature $T_t \in [20, 240]$. The mission fails and the interface displays a game over when the battery is empty, $b_t = 0$, or when the temperature is too high, $T_t \geqslant 240$. Then, the successive water levels of the tank embedded on the robot $w_t^r \in \{0, 10, \ldots, 90, 100\}$ and of the one on the ground $w_t^g \in [0, 100]$ are given in the following two columns. This last tank allows the robot to be filled during the mission if containing enough water.
Leaks can appear at nine different points on the ground tank. These points follow a 3×3 grid pattern. By ordering these points from left to right then from top to bottom, the state of the point $i \in \{1, \ldots, 9\}$ at the second $t \in \{0, \ldots, H\}$ is denoted by $l_t^i \in \{0, 1\}$, with 1 for *leak at this point* and 0 otherwise. As like the states of the trees (f_t), the values in this new column are $l_t = \sum_{i=1}^{9} l_t^i \cdot 10^{9-i}$.
Finally, the last four columns concern the operator's actions on the interface. Note that, in one second, several keyboard keys could be pressed or several clicks on buttons (say $n \in \mathbb{N}$) could be performed. Thus, these three columns contain word sequences separated by dashes. The first contains the keyboard key sequences used to control the robot,

$$k_t \in \{\text{“front”}, \text{“back”}, \text{“left”}, \text{“right”}, \text{“space”}\}^n,$$

with $n \in \mathbb{N}$. These keys correspond respectively to the top, down, left and right arrow keys, and to the space key. For example consider the user's actions when the mission duration is between $t-1$ and t seconds. If the "front" key is pressed, then the "right" key to turn the robot and finally the "space" key to throw water, then the $(t+1)^{th}$ line of this column will contain for instance $k_t =$ "front-front-front-right-space". The second column concerning human behavior contains the sequence of clicks on the interface buttons,

$$c_t \in \{\text{“left”}, \text{“right”}, \text{“push”}, \text{“wrench”}, (\text{“leak\_i”})_{i=1}^{9}, \text{“rm\_alarm”}\}^n$$

with $n \in \mathbb{N}$. Buttons "left" and "right" control the tap of the water management task, while "push" turns it on, and "wrench" turns the mouse pointer into a wrench. When the user then clicks on the leak $i \in \{1, \ldots, 9\}$, it disappears and adds "leak_i" to the clicks sequence, *e.g.* $c_t =$ "wrench-leak_2-wrench-leak_6". When a visual alarm occurs, and the operator clicks on the button "I got it" to remove it from the interface, it leads to the addition of the word "rm_alarm" in this word sequence. The third column related to human behavior records clicks and keys pressed by mistake, as :

$$e_t \in \{\text{“down”}, \text{“up”}, \text{“click”}\}^n, \text{ with } n \in \mathbb{N}.$$

If $e_t =$ "down-up-click" it means that, during second t, a key with no effect was pressed, then such a key was released, and finally an useless click was made. Finally, the last column contains the number of keyboard shortcuts used in the second $t \in \mathbb{N}$. Indeed, the use of some buttons (namely "left", "right", "push" and "wrench") of the GUI (*cf.* Figure 1) can be replaced by keyboard shortcuts (respectively, "s", "d", "e" and "a" keys). Thus, this column contains the number of times a shortcut has been used during the associated second : $\sigma_t \in \mathbb{N}$.
Note that the notation -1 for "No alarm displayed" (see Table 1) is also used in these last four columns. If, in a time step t, no useful key is pressed (resp. no click on buttons is performed, no useless action has been taken, no keyboard shortcut is used), *i.e.* if $n = 0$, then $k_t = -1$ (resp. $c_t = -1$, $e_t = -1$, $\sigma_t = -1$). In the same way, if the value in the dataset is -2, it means that the data is missing.
In summary, the number of columns of this Human-Robot Interaction dataset is equal to 16, and any line of such a CSV file describing a Firefighter Robot mission is organized as follows :

$$m_t = (r_t, a_t^m, a_t^a, x_t, y_t, \theta_t, f_t, b_t, T_t, w_t^r, w_t^g, l_t, k_t, c_t, e_t, \sigma_t)$$

where, m_t is the line number $t+1$ describing events occurring during the time segment $]t-1, t]$ in seconds (or a constant initial state if $t = 0$). As mentioned above, each mission (CSV) file is a $(H+1) \times 16$ table. Each line is a 16-dimensional vector m_t, thus the dataset can be seen as the realization of a random variable sequence of size $H+1$ representing the mission process $(m_t)_{t=0}^{H}$. Note that only k_t, c_t and e_t are not numbers.

tree	**x**	**y**
1	4.55076	14.66826
2	−0.7353	14.75052
3	−15.10146	15.76476
4	−2.6019	6.7425
5	−1.33158	10.02042
6	16.58292	−12.5847
7	16.87086	−16.01952
8	0.6078	−16.23906
9	−16.65378	−16.23906

red square (battery)

x	**y**
0.0	−10.0

blue square (water)

x	**y**
16.0	16.29

TABLE 2 – Location of the trees and the blue and red squares in the map (see Figure 2). When considering the map displayed on the interface, the x-axis points to the right, the y-axis to the top.

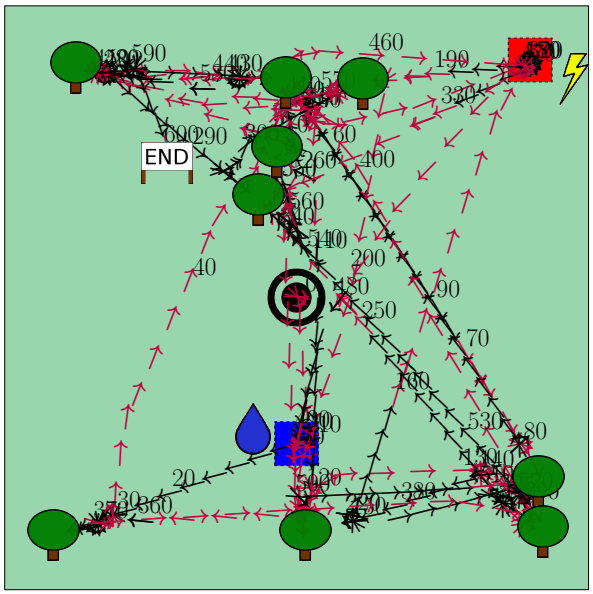


FIGURE 2 – Map of the Firefighter Robot mission on which appear data coming from a mission carried out on the website and acquired by crowdsourcing. The target symbol in the center is the initial location of the robot and the black (resp. red) arrows are the successive robot's locations when the robot is autonomous (resp. in manual mode). The current time steps of the mission are the numbers displayed at the current location of the robot. Trees and locations of squares are also displayed, as well as the terminal robot's location with an "End" flag.

In a few words, the following work assume that this sequence is a Markov chain in order to benefit from the associated learning and planning techniques. Next section describes the formal model used in this paper, the methodology applied to learn a human-agent interaction model, as well as the overall mission model.

3 Formal Model

In this work, and as often in the context of human-robot interaction, the evolution of the system cannot be considered as deterministic. Indeed, human behavior and environmental dynamics are uncertain and therefore require adequate modeling. The Markov Decision Process (MDP) framework [25] is a convenient choice for planning under uncertainty. This famous stochastic control process is an elegant way to model and solve probabilistic planning problems. Once the possible actions and system states have been identified, the goal of the problem is defined using a reward function that evaluates the utility of a state-action pair. This makes possible to define the utility of an action sequence as the expected sum of the rewards obtained over time given an initial state. The optimal sequence of actions is the one that maximizes such an expected sum of rewards. Formally, the (finite horizon) MDP model is defined as a tuple $(\mathcal{S}, \mathcal{A}, T, R, H)$, where :

- $\mathcal{S}$ is the finite set of states ;
- $\mathcal{A}$ is the finite set of actions ;
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function, which defines the probability $p(s' \mid s, a) = T(s, a, s')$ of reaching the state $s' \in \mathcal{S}$ given that the action $a \in \mathcal{A}$ is performed in state $s \in \mathcal{S}$;
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function that values any state-action pair ;
- $H \in \mathbb{N}$ is the horizon, that is the duration of the process in terms of discrete time steps [3].

A policy $\pi \in \Pi$ is a function that associates an action $a \in \mathcal{A}$ to each possible context, that is, in the case of finite horizon MDPs ($H < +\infty$), to each state $s \in \mathcal{S}$ and time step $t \in \{0, \dots, H\} : \pi : \mathcal{S} \times \{0, \dots, H-1\} \to \mathcal{A}$. Solving an MDP is finding a policy that maximizes the expected amount of rewards up to time step H :

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}}\, \mathbb{E}\left[\sum_{t=0}^{H-1} R\big(s, \pi(s,t)\big) \,\middle|\, s_0 = s \right]. \quad (1)$$

Such a policy defines the optimal action $a \in \mathcal{A}$ to perform in a given state $s \in \mathcal{S}$ and time step $t \in \{0, \dots, H-1\}$, and can be computed by *Dynamic Programming* [2] making use of the Bellman operator applied to the optimal value function V^* defined by induction as shown in Equation 2 : the last optimal decision rule is $\pi^*(s, H-1) = \operatorname{argmax}_{a \in \mathcal{A}} R(s, a)$, $V_1^* = \max_{a \in \mathcal{A}} R(s, a)$, and $\forall h \in \{2, \dots, H\}$,

$$\begin{aligned} &\pi^*(s, H-h) = \\ &\quad \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left\{ R(s,a) + \sum_{s' \in \mathcal{S}} T(s,a,s') \cdot V_{h-1}^*(s') \right\} \\ &\quad V_h^*(s) \\ &\quad = \max_{a \in \mathcal{A}} \left\{ R(s,a) + \sum_{s' \in \mathcal{S}} T(s,a,s') \cdot V_{h-1}^*(s') \right\}. \end{aligned} \quad (2)$$

Recent MDP algorithms [16, 4] explore dynamic properties of the model (mainly using Monte-Carlo methods) to optimize actions only for reachable states. Such methods help to decrease the time and memory needed to solve MDPs with large state space.

In the case of the Firefighter Robot mission optimization, the actions to choose over time are the supervision actions, *i.e.* robot mode $a^m \in \{0, 1\}$ and alarm display $a^a \in \{i\}_{i=-1}^{6}$ (see Table 1). As a result, the action space is $\mathcal{A} = \{0, 1\} \times \{i\}_{i=-1}^{6}$. The remaining time is linked to the time step and the horizon by the equality $r_t = H - t$ with $H = 600$. The remaining values in m_t, *i.e.* after removing a_t^m, a_t^a and r_t, constitute the current state $s_t \in \mathcal{S}$.

Note now that the state space defined in this way is not finite as imposed in the definition. For instance the robot pose (x_t, y_t, θ_t), the temperature T_t and the water level of

3. Note that in the most general definition, T and R could also depend on time t.

the ground tank w_t^g are three continuous variables. The battery level b_t, the state of the forest f_t and the leaks on the ground tank l_t are not continuous data, but they have a large number of possible values (100, 2^9 and 2^9 respectively). Since the operator can interact at any speed via keyboard and mouse, if the connection is good, the variables encoding the keystroke sequence k_t, the click sequence c_t and the error sequence e_t can be very long word sequences, and so the number of possible word combinations is also very large. The number of keyboard shortcuts σ_t can also be very large for the same reason. Only the water level w_t^r in the robot tank has a limited number of possible values which is equal to $\#w_t^r = 11$.

It is therefore necessary to discretize these state variables in order to be able to treat the problem with the desired tools. In addition, the discretization must be coarse enough to keep the state space small so that the learning and planning algorithms can solve the problem without too much memory or computation time.

In this study, the learning process consists in the estimation of the transition probability values, *i.e.* the transition function T, that are still missing to fully define the MDP. Crowdsourcing data will be used to estimate this function, but here again, a rough discretization allows for the learning algorithms to give more reliable estimates.

3.1 Variable Selection and Trade-off in Granularity

In this work we make the hypothesis that a rough discretization can allow a better estimation of the transition function T. Indeed such a processing increases the number of occurrence - in the database resulting from crowdsourcing - of the states thus defined. In this way the learning is based on a larger number of samples which should improve the accuracy of T.

Moreover the curse of dimensionality [3] prevents us from starting our supervision study of human-robot team with too many variables otherwise the learning and planning problems will not be practically solvable. Hence, this subsection is dedicated to the description of the MDP actually learned and solved for this first study on the Firefighter Robot dataset.

The selected state variables are limited to discretized versions of x_t, y_t, θ_t, b_t, w_t^r, f_t, and k_t. These variables seem to be the most relevant to take into account in order to optimize supervision to drive the human-robot system. Indeed, the state of the forest is the state of the system on which the goal of the mission depends. The robot's pose and water level are system's states that are quite directly related to fire extinction, so they were also chosen. The level of battery can lead to the end of a game, and therefore a sub-optimal mission given the chosen reward function. Finally, the sequence of keyboard keys used is a source of information about the operator that must be taken into account to allow the supervisory system to adapt to human behavior.

The robot positions are discretized according to a grid 3×3 whose cells have the same size : $pos \in \{NE, N, NW, E, C, W, SE, S, SW\}$. Thresholds are defined for the battery and water level to discretization into two values : $(bat, wat) \in \{\text{“nominal”}, \text{“low”}\}^2$. The total description of the condition of the trees requires a variable with 2^9 possible values. We therefore keep only the number of fires in progress $fire = \sum_{i=1}^{9} f_t^i \in \{0, \ldots, 9\}$. A variable $space$ is also introduced to encode if the user presses the space key during the current second : $space = \mathbb{1}_{\{\text{“space”} \in k_t\}} \in \{0, 1\}$. The variable $end = \mathbb{1}_{\{b_t = 0\}} \in \{0, 1\}$ symbolizing game overs is also introduced. The state variables necessary to define the reward function we had in mind have already been introduced. Since the goal of the human machine team is to keep the trees in good condition, the reward function can be defined as : $R(s, a) = R(s) = R(end, fire) = (1 - end) \cdot (9 - fire)$. Note that, the reward function depends only on end (no more reward if the mission fails) and on $fire$, but it does not depend on the action $a \in \mathcal{A}$.

The following section presents a technique for discretizing the remaining variables, namely the sequence of pushed keys k_t and the orientation θ_t. This leads to a variable called int (for “human's intention”) and taking nine possible values. We thus have seven state variables $s_t = (pos, wat, bat, fire, space, end, int) \in \mathcal{S}$ forming a state space of size $\#\mathcal{S} = 9 \times 2 \times 2 \times 10 \times 2 \times 2 \times 9 = 12960$. In this way, the size of the state space will not prevent the problem from beeing solved, *i.e.* optimized strategy computation using a state-of-the-art MDP solver will be possible.

3.2 Interaction Model Learning

Although the discretization of the rotation θ_t would be easy to implement, the discretization of the keyboard key sequences k_t is less direct. There is no intuitive and simple partition to make. While there are many clustering algorithms for continuous and numerical data [19, 32], the literature on clustering of less structured data is quite poor. Thanks to the technique presented in this section, these two variables will in fact be used to estimate the movement that the operator wants (intends) the robot to perform.

Knowledge of the human operator's intentions can only be beneficial to a supervision system of a human-machine team. For example, suppose that the robot is in autonomous mode in an area where it generally has more difficulty moving than when the human controls it. If the operator presses the directional keys and seems to want to move the robot in a satisfactory direction, a good supervision system would change the robot mode from autonomous to manual. Remember that the robot's hard-coded strategy when in autonomous mode is not optimal, so the human operator could also observe that a better strategy is possible and thus manipulate the keyboard keys to regain control over the robot. On the other hand, if the supervision system detects that the operator's intention is to move in a direction that is suboptimal for the mission, for example to move away from the charging area when the battery is low, it would be

optimal by switching to autonomous mode.

Thus, the variables θ_t and k_t will be discretized into the operator's intentions about the robot's movements. To do this, we will use a sub-dataset from our crowdsourcing database : the robot poses (x_t, y_t, θ_t) and keyboard key sequences k_t during the time steps when the robot is in manual mode. In this mode, the keyboard keys have an effect on the robot's movement. Apart from possible transmission delays, packet losses, obstacles on the way and the shape of the robot's velocity, it seems possible to reliably predict the robot's movement from its orientation and the sequence of directional keys used. Since, in this mode, most of the effects on the robot are those desired by the operator, such predictions provide us with a function of θ_t and k_t, having as a value a probable intention of the user. It is then sufficient to discretize the movements of the robot to be predicted to obtain a discretization of the couple (θ_t,k_t). Indeed the latter will be replaced by its prediction. For instance, even when it has no effect on the robot (*e.g.* obstacle or autonomous mode), a sequence of "front" keys will be replaced by the operator's intention to move forward.

In this work we consider the changes in the robot's position : $(x_{t+1} - x_t, y_{t+1} - y_t) \in \mathbb{R}^2$. These motions are simply discretized into the following values :

$$mot \in \mathcal{L} = \{\text{“NoMot”}, \text{“N”}, \text{“S”}, \\ \text{“O”}, \text{“E”}, \text{“NE”}, \text{“SE”}, \text{“SO”}, \text{“NO”}\},$$

i.e. the fact the robot does not move much, the cardinal and the inter-cardinal directions. These values will be used as labels to learn the prediction function with a supervised classification algorithm. Data concerning the keyboard keys k_t are provided to the classifier as follows. For each $key \in \{\text{“front”}, \text{“back”}, \text{“left”}, \text{“right”}\}$ and for each line of the sub-dataset (corresponding of a time step t of a mission), a truncated number of occurrences of the key in the sequence is computed : $key_t^{oc} = \min(\sum_{key \in k_t} 1, 10)$. The input data of the classifier are then $(front_t^{oc}, back_t^{oc}, left_t^{oc}, right_t^{oc}, \theta_t) \in \{0, \dots, 10\}^4 \times [-\pi, \pi]$: these values will be used to predict the user's intent.

Using Gradient Boosting algorithm [5, 11] available in scikit-learn library [24] the resulting classifier $c : \{0, \dots, 10\}^4 \times [-\pi, \pi] \to \mathcal{L}$ reached an accuracy of 87.5% to predict the motion associated to the robot's orientation and the keystrokes sequence. Using the prediction $int = c(k_t, \theta_t) \in \mathcal{L}$ of this classifier instead of the couple (k_t, θ_t) allows to consider directly the intention of the human operator at the level of the MDP state s_t, while benefiting from a coarse discretization of this couple of variables. The partition defining the discretization of (k_t, θ_t) is $\{c^{-1}(\{l\})\}_{l \in \mathcal{L}}$, that is $\#\mathcal{L} = 9$ subsets.

4 Learning Dynamics and Computing an Optimal Strategy

Now that the choice of variables and their discretization has been implemented, it remains to define the transition function from the crowdsourcing data *i.e.* for each current state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, and for each next state $s' \in \mathcal{S}$, the probability value of such a transition : $T(s, a, s') \in [0, 1]$ with $\sum_{s' \in \mathcal{S}} T(s, a, s') = 1$. Indeed, once this function T is estimated, the desired MDP is fully defined and a planning algorithm can be used to optimize the supervision strategy. The next subsection deals with learning, while the next one deals with planning.

4.1 Independence Assumptions and Markov Transition Learning

We used the *Pomegranate*[4] library [30] to learn the parameters of the MDP based on the seven state variables described above. In fact this library is meant to learn the transition matrix of Markov Chains (MCs) by using maximum likelihood estimates. The trick here is that when subjected to a constant strategy, an MDP becomes a Markov Chain. In this paper, we propose to build a strategy to decide only on the robot mode $a_t^m \in \{0, 1\}$. Future work will deal with alarms $a_t^a \in \{i\}_{i=-1}^{6}$. Two classes of sample transitions should then be differentiated. Those starting from manual mode ($a_t^m = 0$), and those starting from autonomous mode ($a_t^m = 1$). They could be separately given as input to the MC parameters learning algorithm which should return two transition matrices : one is $[T(s, 0, s')]_{(s,s') \in \mathcal{S}^2}$ and the other $[T(s, 1, s')]_{(s,s') \in \mathcal{S}^2}$, both of size 12960^2. Unfortunately, this number of system states prevents the learning algorithm of the Pomegranate library from being used directly, and it is necessary to trick once again to compute this transition function. The trick here consists in computing transition matrices on a sub-group of variables. This method is made possible by first assuming that the variables at time step $t + 1$ are independent of each other conditionally to the variables in step t (*i.e.* conditionally to the past since the Markov property is already assumed), regardless of the action chosen before the transition. An MDP whose variables have this independence property is called a *factored MDP* [14, 13]. Note that this formalism is used in the *International Probabilistic Planning Competition*[5] (IPPC), using the *Relational Dynamic Influence Diagram language* (RDDL [29]) that we'll use in the next subsection for strategy optimization. Complying with this framework is not a very strong assumption. Indeed, if the initial MDP has n variables, it is always possible to subdivide the time steps into n sub-steps during which only one variable makes its transition while the others remain constant : thus, by considering the MDP equipped with these new intermediate time steps, the variables are indeed independent conditional on the past. In our case we simply

4. https://pomegranate.readthedocs.io
5. https://ipc2018-probabilistic.bitbucket.io/

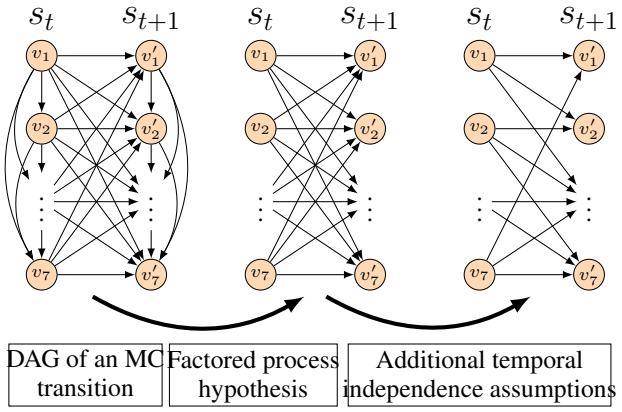


FIGURE 3 – Without any assumption of independence, the Directed Acyclic Graph (DAG) defining the Bayesian network representing the variables during a transition is complete (left). Assuming that the variables of the same time step are independent of each other, conditional on the past, the DAG becomes a complete bipartite graph (center). Finally, other independence hypotheses also make it possible to delete some arrows (right).

assume that the variables are conditionally independent, without subdividing the time steps.

More formally, let us denote the seven considered variables by $(v_t^i)_{i=1}^7$, *i.e.* $s_t = (v_t^1, \ldots, v_t^7)$. The independence properties assumed in the factored framework implies that it exists seven functions $(T_i)_{i=1}^7$, one for each variable, such that $T(s_t, a_t, s_{t+1}) = \prod_{i=1}^7 T_i(s_t, a_t, v_{t+1}^i)$. The function T_i is the transition function of variable v_i'. Thus, by representing uncertainty dynamics with Directed Acyclic Graph (DAG) [23], the resulting graph is bipartite and complete as shown in Figure 3, such that the two sets of nodes constituting the bipartition of the bipartite graph are the set of variables at time t and the set of variables at time $t+1$. In addition to enabling a more efficient optimization of the MDP strategy in different uncertainty models [14, 10, 7] (by decreasing the time spent to compute the optimal policy), this characteristic will enable the learning algorithm to compute an estimation of the transition function T. For this purpose, however, it is necessary to identify additional variable independence, this time in temporal terms.

The transition probability of each variable does not depend on all previous variables. For example, the battery level depends on the previous battery level, but not on the previous water level (and vice versa). In other words, conditional on the other variables of the same time step t, the battery level at time t is independent of the water level at time $t+1$. By introducing this kind of expert knowledge on the independences of variables, some arrows can be removed from the bipartite graph representing the transition uncertainty. Figure 3 illustrates also this last pruning operation. More concretely, for each transition function T_i (associated with variable v_i), there is a subset of variables

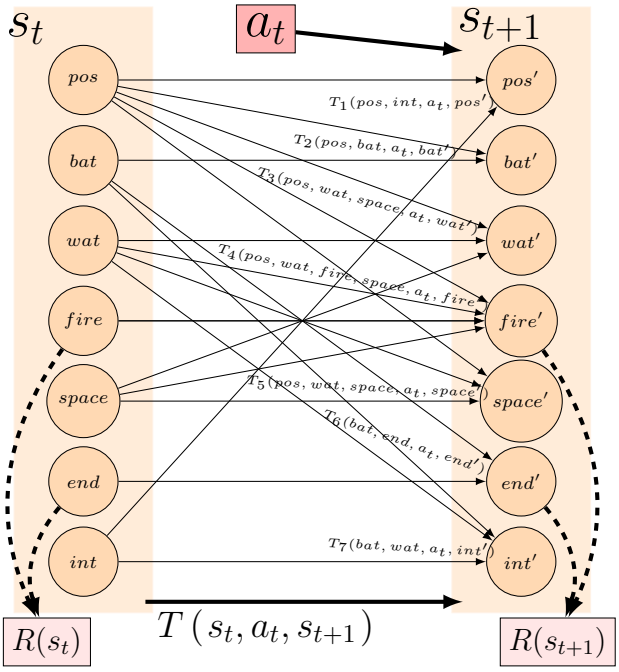


FIGURE 4 – The DBN of the learned MDP for the Firefighter Robot game. Primed (resp. unprimed) variables represent next (resp. current) variables.

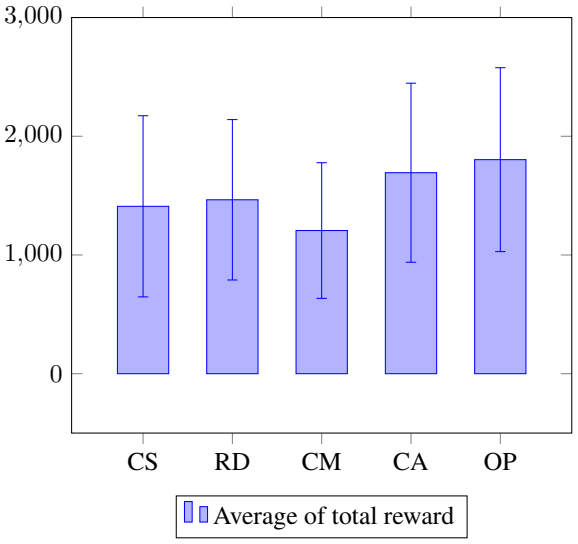


FIGURE 5 – Estimation of expected total reward $\mathbb{E}\left[\sum_{t=0}^{H} R(s_t)\right]$. On the left (CS), the average on the crowdsourcing data. Other values are the average of rewards computed based on 300 simulations of the MDP with a random policy (RD), a constant manual policy (CM), a constant autonomous policy (CA) and a policy optimized with an MDP solver (OP).

$W_i \subset \{v_1, v_2, \ldots, v_7\}$ such that $T_i(v_1, v_2, \ldots, v_7, v_i') = T_i(W_i, v_i')$, where primed variables represent next variable while unprimed ones stand for current variables. In the resulting DAG, this subset W_i is the set of parents of v_i'. The idea used here is that the calculation of the transition function T_i only requires transition samples concerning only the variables in W_i, so the learning algorithm is run several times ($\forall i \in \{1, \ldots, 7\}$), but on fewer variables (because $\#W_i < 7$) and therefore with dimensions that can be handled. Note that the learning algorithm returns the values of $T(W_i, a, W_i') = p(W_i' \mid W_i, a), \forall a \in \mathcal{A}$, so the transition function of variable v_i' can be computed by summing over variables $Z_i = W_i \setminus \{v_i\}$:

$$T_i(W_i, a, v_i') = \sum_{Z_i'} T_i(W_i, a, W_i') = \sum_{Z_i'} p(v_i', Z_i' \mid W_i, a).$$

4.2 Strategy Optimization and Simulation Results

The MDP that is finally obtained is illustrated in the form of a Dynamic Bayesian Network (DBN) [21] in Figure 4. This is a minimal modeling choice while remaining understandable for human being and tractable considering learning and policy optimization. In order to evaluate our model, we simulated the process thus defined in order to estimate the expected total reward, *i.e.* the criterion that we want to maximize (see Equation 1). To do this, the problem was written in RDDL format using the transition functions $(T_i)_{i=1}^7$ learned previously.

These simulations were carried out under four different conditions. In each condition, 300 simulations were performed : with a random strategy (at each time step t, $p(a_t = 0) = p(a_t = 1) = 0.5$), with a constant manual strategy ($\forall t$, $a_t = 0$), with a constant autonomous strategy ($\forall t$, $a_t = 1$) and with an online strategy optimization provided by *PROST*[6] [16]. This algorithm implements *Trial-based Heuristic Tree Search* (THTS) [17] with the settings of IPPC 2014. The action selection of the solver UCB1 is used, described by [1] for *Multi-armed Bandit problems* and used in the solver UCT [18]. It also uses the *Partial Bellman* backup function [17] and samples unsolved outcomes using its probability. Finally, the results obtained during these simulations are compared with those obtained using data collected via the crowdsourcing platform, in which the policy is random. More precisely, the average of total reward was computed with 678 missions that players have not left voluntarily (by leaving the interface) but performed entirely or failed because of a game over.

The results (expectation of the total reward) are illustrated in Figure 5, with the standard errors of the data coming from the 300 simulations. The estimation of the expected total reward is 1409.08 when crowdsourcing data is used. This value is close to that obtained by simulating the mission with a random policy (1465.52), which is reassuring as to the adequacy of reality with the MDP learned. The strategy of setting the robot mode to manual mode at all times leads to the lowest performance in terms of average total reward on simulations (1206.29). Simulations with a constantly autonomous robot give a much higher average total reward (1693.73), which is not contradictory to reality : the mission being multi-task, the operator cannot control the robot all the time. Thus, an autonomous robot is more beneficial for the mission than a manual robot. Finally, the optimized online strategy provides slightly better results (1803.56). Note that this strategy is not the optimal strategy, and that offline (and time consuming) resolution can provide a much higher total reward.

These initial results confirm us in the idea that a supervision strategy, based on the states of the human-robot system and calculated using a decision model under uncertainty, can be used and improved in order to obtain better team performance.

6. https://bitbucket.org/tkeller/prost/wiki/Home

5 Conclusion and Future Work

Instead of defining a probabilistic model with expert or arbitrary parameters, this paper proposes a methodology to learn an human-agent interaction model from crowdsourcing data collection. Our probabilistic interaction model takes into account human actions on the keyboard and random environment changes using the transition function T of an MDP defined thanks to machine learning techniques. A reliable T function should enable MDP solvers to optimize the supervision policy in accordance with the real human-robot interaction and mission considered.

Offline optimization of the strategy will be performed in future work with state-of-the-art algorithms [16, 4] to achieve near optimal solutions for such an MDP model. The policy computed will be deployed in laboratory conditions and on the project's website for an evaluation of the overall system's performance. Moreover, if we consider that the difficulty of the mission leads the human operator into degraded mental states (mental fatigue, working memory load [27], attentional tunneling [26], etc.) the estimation of the operator's mental state could enrich the representation of the system described by the MDP. Clearly, a human mental state is not a fully observable state variable. In this case, the use of Partially Observable Markov Decision Process [34] could be a promise avenue to model such a human-agent interaction problem.

Finally, one could explore frameworks able to provide a generative model [12, 28], that is a simulator, based on collected data described in this paper. The development of such a simulator could be explored in order to apply Reinforcement Learning techniques [35, 20] to learn the supervisory policy based on simulated missions. Even more, automated discretization, as clustering algorithms, could be used to infer a possible smarter discretization while keeping a sufficiently low number of clusters to ensure reliable transition function estimates.

Références

[1] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3) :235–256, 2002.

[2] Richard Bellman. The theory of dynamic programming. *Bull. Amer. Math. Soc.*, 60(6) :503–515, 11 1954.

[3] Richard Ernest Bellman. Rand corporation (1957). *Dynamic programming*.

[4] Blai Bonet and Hector Geffner. Action selection for mdps : Anytime ao* versus uct. In *AAAI*, 2012.

[5] Leo Breiman. Arcing the edge. Technical report, Technical Report 486, Statistics Department, University of California at Berkeley, 1997.

[6] J Raymond Comstock Jr and Ruth J Arnegard. The multi-attribute task battery for human operator workload and strategic behavior research. 1992.

[7] Karina Valdivia Delgado, Scott Sanner, Leliane Nunes De Barros, Fábio Gagliardi Cozman, et al. Efficient solutions to factored mdps with imprecise transition probabilities. *Artificial Intelligence*, 175(9-10) :1498–1527, 2011.

[8] Nicolas Drougard, Caroline Ponzoni Carvalho Chanel, Raphaëlle N Roy, and Frédéric Dehais. Mixed-initiative mission planning considering human operator state estimation based on physiological sensors. In *IROS Workshop on Human-Robot Interaction in Collaborative Manufacturing Environments (HRI-CME)*, 2017.

[9] Nicolas Drougard, Raphaëlle N Roy, Sébastien Scannella, Frédéric Dehais, and Caroline Ponzoni Carvalho Chanel. Physiological assessment of engagement during hri : Impact of manual vs automatic mode. In *2nd International Neuroergonomics Conference*, 2018.

[10] Nicolas Drougard, Florent Teichteil-Königsbuch, Jean-Loup Farges, and Didier Dubois. Structured possibilistic planning using decision diagrams. In *AAAI*, pages 2257–2263, 2014.

[11] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4) :367–378, 2002.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[13] Carlos Guestrin, Milos Hauskrecht, and Branislav Kveton. Solving factored mdps with continuous and discrete variables. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 235–242. AUAI Press, 2004.

[14] Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier. Spudd : Stochastic planning using decision diagrams. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 279–288. Morgan Kaufmann Publishers Inc., 1999.

[15] Shu Jiang and Ronald C Arkin. Mixed-initiative human-robot interaction : Definition, taxonomy, and survey. In *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, pages 954–961. IEEE, 2015.

[16] Thomas Keller and Patrick Eyerich. Prost : Probabilistic planning based on uct. In *ICAPS*, 2012.

[17] Thomas Keller and Malte Helmert. Trial-based heuristic tree search for finite horizon mdps. In *ICAPS*, 2013.

[18] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *ECML*, volume 6, pages 282–293. Springer, 2006.

[19] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2) :129–137, 1982.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540) :529, 2015.

[21] Kevin Patrick Murphy and Stuart Russell. Dynamic bayesian networks : representation, inference and learning. 2002.

[22] William D Nothwang, Michael J McCourt, Ryan M Robinson, Samuel A Burden, and J Willard Curtis. The human should be part of the control loop ? In *Resilience Week (RWS), 2016*, pages 214–220. IEEE, 2016.

[23] Judea Pearl. *Probabilistic reasoning in intelligent systems : networks of plausible inference*. Elsevier, 2014.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830, 2011.

[25] Martin L Puterman. *Markov decision processes : discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[26] Nicolas Régis, Frédéric Dehais, Emmanuel Rachelson, Charles Thooris, Sergio Pizziol, Mickaël Causse, and Catherine Tessier. Formal detection of attentional tunneling in human operator–automation interactions. *IEEE Transactions on Human-Machine Systems*, 44(3) :326–336, 2014.

[27] Raphaëlle N Roy, Stephane Bonnet, Sylvie Charbonnier, and Aurélie Campagne. Mental fatigue and working memory load estimation : interaction and implications for eeg-based passive bci. In *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, pages 6607–6610. IEEE, 2013.

[28] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[29] Scott Sanner. Relational dynamic influence diagram language (rddl) : Language description. *Unpublished ms. Australian National University*, page 32, 2010.

[30] Jacob Schreiber. Pomegranate : fast and flexible probabilistic modeling in python. *Journal of Machine Learning Research*, 18(164) :1–6, 2018.

[31] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat : Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv :1312.6229*, 2013.

[32] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8) :888–905, 2000.

[33] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587) :484–489, 2016.

[34] Richard D. Smallwood and Edward J. Sondik. *The Optimal Control of Partially Observable Markov Processes Over a Finite Horizon*, volume 21. INFORMS, 1973.

[35] Richard S Sutton and Andrew G Barto. *Reinforcement learning : An introduction*, volume 1. MIT press Cambridge, 1998.

[36] Kevin W. Williams. A summary of unmanned aircraft accident/incident data : Human factors implications. *U.S. Department of Transportation, Federal Aviation Administration, Civil Aerospace Medical Institute*.

Schedule Earth Observation satellites with Deep Reinforcement Learning

Adrien Hadj-Salah[1,2], Rémi Verdier[1], Clément Caron[1,2], Mathieu Picard[1,2], Mikaël Capelle[1]
[1]IRT Saint-Exupéry [2]Airbus Defence & Space
{adrien.hadj-salah, remi.verdier, clement.caron, mathieu.picard, mikael.capelle}@irt-saintexupery.com

Abstract

Optical Earth observation satellites acquire images world-wide, covering up to several million square kilometers every day. The complexity of scheduling acquisitions for such systems increases exponentially when considering the interoperability of several satellite constellations together with the uncertainties from weather forecasts. In order to deliver valid images to customers as fast as possible, it is crucial to acquire cloud-free images. Depending on weather forecasts, up to 50% of images acquired by operational satellites can be trashed due to excessive cloud covers, showing there is room for improvement. We propose an acquisition scheduling approach based on Deep Reinforcement Learning and experiment on a simplified environment. We find that it challenges classical methods relying on human-expert heuristic.

1 Introduction

Earth Observation (EO) systems acquire cloud-free images and deliver them to customers worldwide on a daily basis. Requests come in a variety of size and constraints, from the urgent monitoring of small areas to large area coverage. In this work we are particularly interested in the latter case, with requests covering whole countries or even continents. Depending on weather conditions, such requests may take several months to complete, even with multiple satellites.

In order to shorten the time required to fulfill requests, the mission orchestrator shall schedule acquisitions with both a short and a long-term strategy. Determining a strategy robust to an uncertain environment is a complex task, this is why current solutions mainly consist of heuristics configured by human-experts. This paper demonstrates that Reinforcement Learning (RL) might be well-suited for such a challenge. RL has proven to be of great value since these algorithms have mastered several games such as Pong on Atari 2600 (Mnih et al. 2013), Go with AlphaGo (Silver et al. 2017) and more recently Starcraft (Arulkumaran, Cully, and Togelius 2019).

2 Scheduling acquisitions for Earth observation systems

2.1 Single satellite acquisition scheduling

EO satellites carry optical instruments which are able to take acquisitions with a specific width, called swath, and a maximum length depending on the satellite models. The capacity of the satellites to take multiple images along their orbit track is related to their agility (Lemaître et al. 2002).

Due to limited swath and acquisition length, a large area must be split into tiles called meshes. For instance, considering the Pleiades satellites, covering France requires thousands of meshes. A satellite overflying an area is able to acquire a sub-part of those meshes due to its limited agility. With sun-synchronous orbit, revisit of a ground point takes days which explains the importance of mesh selection (Gleyzes, Perret, and Kubik 2012).

The satellite schedule is computed on ground by the Mission Planning Facility (MPF), where an optimization algorithm selects the top-ranked acquisitions and ensures the kinematic feasibility of the attitude maneuvers.

2.2 Interoperable EO systems scheduling for large-area coverage

The trend of EO systems is toward large constellations of heterogeneous satellites. For instance, Airbus Intelligence, operating the well-known SPOT and Pleaides satellites, will soon manage a new system of 4 satellites (Pleaides NEO). Dealing with multiple EO systems needs both human expertise and algorithms to dispatch requests over the satellites and to deliver end customers on time.

We approach the constellation scheduling by having an orchestrator responsible for request ranking towards each MPF. The orchestrator analyzes a large-scope of data (e.g., forecasts, access opportunities) to optimize the global schedule, while each MPF has a narrowed and short term vision of their single (or dual) satellite scheduling. Additionally, we focus in this paper on requests consisting in a large area (countries, continents). Such requests usually contain several hundreds

of meshes to acquire over long periods (up to several months).

The two main contributors to the overall uncertainty on the time to completion are: firstly the weather conditions at the time of acquisition, which can only be forecasted, and secondly the presence of other requests within the systems, arriving at an unknown rate.

This explains our focus on RL algorithms which have the capacity to learn new strategies, robust to uncertainties, while challenging traditional approaches.

3 Reinforcement Learning approach

In Reinforcement Learning, an agent learns how to behave through trial-and-error interactions with a dynamic environment. The actions the agent takes are decided by a *policy*, which can be seen as a function mapping information from the environment to actions. The goal of reinforcement learning algorithms is to find an optimal policy, i.e., a policy that maximizes the reward of the agent over the long-term.

Recently, deep neural networks have proven to be efficient for finding policies. Several deep-RL algorithms are actively studied to solve complex sequential decision-making problems. Among the best-known methods, one can cite value-based algorithms such as DQN, Rainbow (Hessel et al. 2018), policy-based algorithms such as REINFORCE (Sutton et al. 2000) or actor-critic methods such as A2C (Mnih et al. 2016) or PPO (Schulman et al. 2017).

3.1 Problem simplification

In order to evaluate the benefits of Reinforcement Learning, we propose a simplified environment.

We consider that all satellites have the same swath, thus the tessellation (i.e., the meshes) of the area is the same for all satellites. We also assume that each satellite can acquire at most one mesh per pass over the considered area. A satellite pass occurs when it overflies the large-area request on a given orbit. The planned mesh is validated or rejected depending on actual cloud cover observations at the time of acquisition. We do not consider uncertainties related to the load of our system, i.e., satellites are always fully available.

The area of interest (AOI) is enclosed in a rectangular box – considering a Mercator projection – containing $N_{lat} \times N_{lon}$ meshes. Since some meshes of this grid mesh may not belong to the AOI, we define $\mathcal{M} = \{m_k \ : \ 1 \leq k \leq K\}$, the set of meshes to acquire.

For each pass $t \in \mathbb{N}$, we denote by $\mathcal{M}_t \subseteq \mathcal{M}$ the subset of meshes in the AOI that can be acquired by the corresponding satellite knowing its orbit and agility.

We denote by $c_t^a(m)$ and $c_t^f(m)$ the actual and forecast cloud cover above mesh m during pass t.

3.2 Problem formulation

The given problem can be formalized as a **Markov Decision Process** (MDP) which is an intuitive and fundamental formulation for RL (Bensana et al. 1999).

An agent interacts with the environment by taking actions from a legal set of actions. The agent purpose is to acquire $\mathcal{M}$ as quickly as possible. For each step t, only one mesh can be selected. The chosen mesh is then validated or rejected depending on weather conditions.

The state space $\mathcal{S}$, the discrete action space $\mathcal{A} \subset \mathbb{N}$, the stochastic discrete-time transition function P and the reward function R define the underlying MDP: $M =< \mathcal{S}, \mathcal{A}, P, R >$.

The horizon is considered finite. Therefore, there is a finite number of discrete time steps t during an episode. Each episode comprises a maximum of $T \in \mathbb{N}^*$ steps. The state space $\mathcal{S}$ is defined as:

$$\mathcal{S} = \mathcal{S}_{status} \times \mathcal{S}_{time} \times \mathcal{S}_{passes}$$

where $\mathcal{S}_{status} = \{0,\ 1\}^{N_{lat} \times N_{lon}}$ characterizes the status of each mesh (i.e., already validated or to acquire), $\mathcal{S}_{time} \subset \mathbb{R}$ encodes the date of the current satellite pass t and $\mathcal{S}_{passes} \subset \mathbb{R}^d$ describes all pass dates, accessible meshes $\mathcal{M}_t$ and related weather forecasts.

The goal is to find a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the expected discounted reward over the finite horizon:

$$\sum_{t=0}^{T} \gamma^t R(s_t, \pi(s_t), s_{t+1}) \tag{1}$$

where $0 \leq \gamma < 1$ is the discount factor and $s \in \mathcal{S}$.

3.3 Action space

At each discrete step t, the learning agent takes an action. A step corresponds to a satellite pass over the AOI and the action is to pick up a single mesh to acquire during this pass.

$$\mathcal{A} = \{0,\ 1, \ldots,\ K\}$$

We denote a_k the action selecting the mesh m_k. Note that $|\mathcal{A}| = K+1$ because there is one more "do nothing" action available for the agent.

3.4 Observation space

At a given step, the agent perceives only useful and available information about the environment. The problem is generalized to a Partially Observable Markov Decision Problem (POMDP).

The observation space O provides information about the mesh status and their validation probability for the following N_{pass} passes, including the current pass for which the agent shall select a mesh. The validation probability of a mesh depends on weather forecast accuracy, as detailed in Section 3.6. Thus, an observation is a tensor with a shape $(N_{lat},\ N_{lon},\ N_{pass} + 1)$.

The observation can be seen as a stack of $N_{lat} \times N_{lon}$ matrices. Each frame (i.e., 2D matrix) encodes information for all tiles of the grid mesh. This representation preserves spatial information and enables the use of Convolutional Neural Networks.

The validation frame encodes the status of each mesh: validated (0) or to be validated (1). We denote the validation frame space $O_{status} = \mathcal{S}_{status}$.

The validation probability frames belong to the space $O_p = [0,\ 1]^{N_{lat} \times N_{lon} \times N_{pass}}$. They encode the probability p_t to acquire and validate each mesh for each pass in chronological order from time step t. For a given mesh m and a given pass $n \in \{1, \ldots, N_{pass}\}$ at the step t:

$$p_t(m,n) = \begin{cases} 0 & \text{if } m \notin \mathcal{M}_t \\ \mathbb{P}(c^a_{t_n}(m) \leq c_{max} \mid c^f_{t_n}(m)) & \text{otherwise} \end{cases}$$

with c_{max} the total cloud cover validation threshold. $t_n = t + n - 1$ is the time related to the pass n knowing that the current time step is t.

We can now define $O = O_p \times O_{status}$

3.5 Reward

A reward is given to the agent at each step. The value of the reward depends on the status of the chosen mesh before and after this step. $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ gives rewards for particular transitions between states.

$$R(s_t, a_k, s_{t+1}) = \begin{cases} 1 & \text{if } m_k \text{ is newly validated} \\ 0 & \text{otherwise} \end{cases}$$

This dense reward encourages the agent to reduce the completion time with a discount factor $\gamma < 1$ (1).

3.6 Transition function

$P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,\ 1]$ denotes the transition function.

$$P(s_t, a, s_{t+1}) = \mathbb{P}(s_{t+1} | s_t, a)$$

For each transition, the current state is updated. $s^{time} \in \mathcal{S}_{time}$ takes the value of the next pass date in the chronological order. $s^{passes} \in \mathcal{S}_{passes}$ remains the same during the whole episode. $s^{status} \in \mathcal{S}_{status}$ is updated if the selected mesh is validated:

$$\mathbb{P}(s^{status}_{t+1}(m_k) = 0 \mid s^{status}_t(m_k) = 1, a_k) = p_t(m_k, n)$$

where $s^{status}_t(m_k)$ is the status of m_k at t.
This probability is computed considering the following weather model:

$$c^a_t(m) = c^f_t(m) + \chi(m)$$

$$\begin{array}{ll} \text{with} & \chi(m) \sim \mathcal{N}(c^f_t(m), \sigma(c^f_t(m))^2) \\ \text{and} & \sigma(x) = u \times x + v \end{array}$$

σ is a linear function computing a representative deviation between forecast and observed data.

4 Experiments

Based on the hypotheses from Section 3.1, we implement a simulator using the OpenAI Gym framework.

4.1 Scenario

To evaluate the agents, we choose mainland France as our area of interest. It is an interesting case to study because one mesh selection can have an important impact on the mission length due to the territory climate variability.

We consider 4 satellites with a common 60 km swath, implying $K = 122$ meshes for our tesselation. Each scenario begins at a random date.

We use the ERA-Interim dataset (Dee et al. 2011) which provides total cloud cover observations on a $0.5° \times 0.5°$ grid to compute the observation space. In the weather model, u is fixed to 0.1 and v to 0.2. c_{max} is set to 20% for all scenarios.

4.2 Reference algorithms

In order to benchmark the performances of our agent, we define a random agent and a heuristic that selects one mesh among $\mathcal{M}_t$ for each time step t:

- **Random** that selects the mesh randomly among accessible meshes at each pass.
- **Heuristic** that selects the mesh with the highest trade-off score between short-term and long-term probabilities p_t:

  $$p_t(m,1) + \alpha \left(1 - \frac{1}{N_{pass} - 1} \sum_{n=2}^{N_{pass}} \beta^n p_t(m,n)\right)$$

  where α is the weight on future passes and β the discount factor that favors near future passes. The best performances are reached with ($\alpha = 1$, $\beta = 0.99$) for $N_{pass} = 20$.

4.3 Train and test methodology

To avoid overfitting, we use a train and test split methodology on the weather data. Training is done using data from the years 2013 and 2014, while testing is done with data from 2015.

We concentrate our experiments on the A2C algorithm which gives the best results. We train A2C agents using two observation spaces: one with a short-term vision ($N_{pass} = 1$) and one with a long-term vision ($N_{pass} = 20$). Those A2C agents are respectively named A2C-1 and A2C-20. We use the A2C implementation from the OpenAI baselines framework (Dhariwal et al. 2017) and train agents during 3×10^7 steps using 16 parallel environments ($\sim$ 30 hours using a K80 GPU and 8 vCPUs). Other hyper-parameters are set to default values.

We use a neural network architecture made of a convolution block followed by a dense block with two heads: one to estimate the state value and one to estimate the policy distribution. The convolution block contains three convolutional layers with decreasing kernel sizes (7×7, 3×3, 1×1), 128 filters per layer and ReLU activation functions. The value and policy heads are only made of a dense layer with respectively one unit and $K + 1$ units.

Figure 1 shows the mean length of the last 100 episodes as a function of the number of network weight updates for A2C-1 and A2C-20. In our environment, the length of an episode directly relates to the completion time of the area. We set a maximum number of $10 \times K$ time steps before resetting the environment

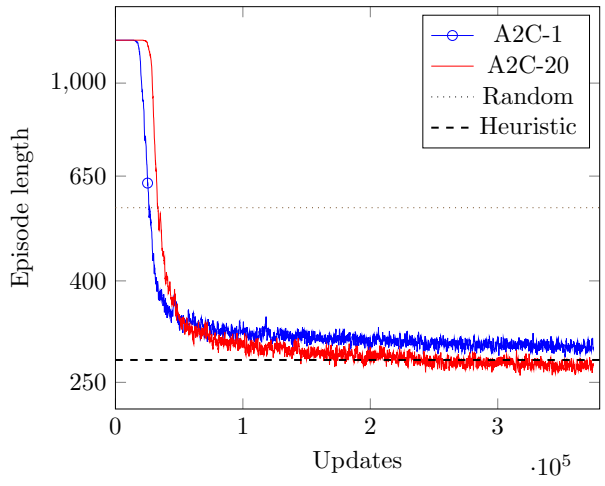


Figure 1: Episode mean length for the last 100 episodes of each training phase.

Agent	Mean	Median	Std
Random	568.8	572	110.5
Heuristic	292.7	298	56.0
A2C-1	299.3	304	58.2
A2C-20	278.5	281	55.8

Table 1: Mean, median, standard deviation of the episode lengths for the different agents.

to avoid too long episodes when the policy does not perform well. The performances of the trained agents converge close to the heuristic one. Best results are achieved with A2C-20.

During testing phase, we select days from 2015 as starting dates. For each date we assess the performances of the models and the reference algorithms. We repeat the operation using 3 different weather seeds (3×365 runs in total). Table 1 presents statistics on the episode length for the different agents. We find that for both agents the transfer on the new weather data went well.

A2C-20 still provides the best results winning the heuristic in almost 80% of the cases. It confirms the intuition that a long term strategy is necessary to optimize time-to-completion.

5 Conclusion

This paper demonstrates how Reinforcement Learning can be used in Earth Observation satellites scheduling in order to reduce the time-to-completion of large-area requests. The computed network has been trained to rank the requests and dispatch them to the satellites. In a series of simulation-based experiments, the proposed method challenges the state-of-the-art heuristics.

In future research, we aim to improve the simulation representativeness in order to pave the way for a potential industrial transfer.

References

Arulkumaran, K.; Cully, A.; and Togelius, J. 2019. Alphastar: An evolutionary computation perspective. *arXiv preprint arXiv:1902.01724*.

Bensana, E.; Verfaillie, G.; Michelon-Edery, C.; and Bataille, N. 1999. Dealing with uncertainty when managing an earth observation satellite. In *Proc. 5th International Symposium on Artificial Intelligence, Robotic and Automation in Space (ESA SP-440)*, 205–207.

Dee, D. P.; Uppala, S. M.; Simmons, A. J.; et al. 2011. The ERA-Interim reanalysis: Configuration and performance of the data assimilation system. *Quarterly Journal of the Royal Meteorological Society* 137(656):553–597.

Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; Wu, Y.; and Zhokhov, P. 2017. Openai baselines. `https://github.com/openai/baselines`.

Gleyzes, A.; Perret, L.; and Kubik, P. 2012. Pleiades system architecture and main performances. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science* XXXIX-B1.

Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Proc. of the 32nd AAAI Conference on Artificial Intelligence*.

Lemaître, M.; Verfaillie, G.; Jouhaud, F.; Lachiver, J.-M.; and Bataille, N. 2002. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology* 6(5):367–381.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T. P.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *Proc. of the 33rd International Conference on Machine Learning (JMLR: W&CP vol. 48)*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676):354.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Système multi-agent coopératif pour l'acquisition par satellites de grandes couvertures grâce à un maillage dynamique

Timothée Jammot[1] Elsy Kaddoum[2] Serge Rainjonneau[1] Mathieu Picard[1] Marie-Pierre Gleizes[2]

[1] Institut de Recherche Technologique - IRT Saint-Exupéry *(prenom.nom@irt-saintexupery.com)*
[2] Institut de Recherche en Informatique de Toulouse - IRIT *(prenom.nom@irit.fr)*

Résumé

Dans le cadre de missions d'observation de la Terre, des satellites sont utilisés pour l'acquisition de zones à la surface terrestre. Les grandes couvertures sont des chantiers d'acquisition de zones à grande échelle. Elles nécessitent l'utilisation d'un ensemble de satellites qui réalisent plusieurs passages pendant une longue période d'acquisition. Découper une zone d'intérêt en sous-zones appelées mailles qui peuvent être acquises par les satellites durant leurs passages est un problème d'optimisation difficile.

Dans ce papier, nous présentons le système Glimpse basé sur les systèmes multi-agents adaptatifs pour le découpage et positionnement dynamiques de mailles sur une grille de sous-zones élémentaires. Les expérimentations menées ainsi que la comparaison à une approche faisant partie de l'état de l'art opérationnel montrent l'adéquation d'une approche décentralisée et dynamique pour la résolution de ce problème.

Mots-clés

Planification satellite, système multi-agent adaptatif, maillage dynamique, optimisation, émergence

Abstract

As part of Earth observation missions, satellites are used to acquire areas on the Earth's surface. Large area coverage missions are large-scale acquisition projects. They require the use of a set of satellites that make several passes over a long acquisition period. Splitting an area of interest into sub-areas called meshes that can be acquired by satellites during their passes is a difficult optimization problem.

In this paper, we present the Glimpse system based on adaptive multi-agent systems for the dynamic positioning and sizing of meshes on a grid of elementary sub-areas. The experiments carried out and the comparison with an approach that is part of the operational state of the art show that a decentralised and dynamic approach to solving this problem is adequate.

Keywords

Dynamic meshing, adaptive multi-agent system, cooperation, optimization, emergence

1 Introduction

Les satellites d'observation en orbite basse ont pour objectif de réaliser des prises de vues de zones d'intérêt à la surface terrestre. Les missions dites grandes couvertures concernent des requêtes d'acquisition de grandes zones d'intérêt, à l'échelle de pays ou de continents. Les grandes couvertures permettent d'obtenir des images utilisées dans des domaines tels que la cartographie et la surveillance[14]. Ce sont des missions coûteuses en ressources satellites et qui nécessitent des images temporellement proches pour garantir la cohérence des raccordements entre elles. Pour ces raisons, la réalisation d'une grande couverture doit être effectuée en temps et ressources satellites minimaux. La durée nécessaire à un unique satellite pour réaliser une mission de grande couverture est trop important par rapport aux contraintes temporelles. Un ensemble de satellites doit être utilisé pendant une longue période de couverture. Ces satellites peuvent être hétérogènes, dotés de caractéristiques et d'instruments différents, et être opérés par différents centres de planification indépendants. Un enjeu des missions actuelles des satellites d'observation est la coordination de centres de planification pour la bonne résolution de missions de grande couverture par des satellites hétérogènes. Cette coordination a pour objectif d'améliorer l'utilisation des satellites d'observation pendant leur durée de vie limitée et de faciliter l'accès aux images satellitaires.

Le partage de la zone d'intérêt d'une grande couverture entre un ensemble de satellites est un problème à la fois géographique et temporel. Pour chaque passage d'un satellite un ensemble de sous-zones appelées mailles pouvant faire l'objet d'une acquisition doit être défini. Nous

définirons le problème des grandes couvertures comme le découpage et la répartition de la zone d'intérêt en mailles attribuées à des satellites pour l'optimisation des temps et ressources satellites de la mission. La technique de maillage couramment réalisée pour la résolution de ce problème sera appelée **maillage statique**. Dans cet article nous proposons une technique innovante pour le découpage et placement des mailles surnommée **maillage dynamique**. Cette approche relâche les contraintes géographiques liées aux mailles et permet un placement plus précis et relatif aux besoins d'acquisition au cours de la grande couverture. Cependant, l'augmentation de l'espace de recherche qui en résulte nécessite l'usage d'algorithmes par méthode approchée pour l'optimisation du problème des grandes couvertures.

La suite de l'article est organisée comme suit. La section 2 présente la formalisation du problème des grandes couvertures ainsi qu'un état de l'art des problèmes similaires et des méthodes d'optimisation utilisées pour les résoudre. La partie 3 présente le fonctionnement du système multi-agent coopératif Glimpse conçu pour résoudre ce problème. Enfin, la partie 4 propose des résultats et une discussion sur la comparaison de l'algorithme Glimpse avec un algorithme état de l'art de maillage statique pour des scénarios simulés de grandes couvertures.

2 Le problème des grandes couvertures

2.1 Description du problème

Le problème des grandes couvertures concerne l'optimisation de l'acquisition d'une **zone d'intérêt** définie par une **requête client** par un ensemble de satellites.

Un **satellite** est un outil d'acquisition opéré par un **centre de planification** suivant une **orbite** polaire. Selon son orbite et la rotation de la terre, le satellite survole des zones terrestres selon des intervalles temporels appelés **passages**. Lors d'un passage donné, une zone géographique appelée **corridor** est accessible par le satellite. La largeur d'un corridor dépend de l'agilité des satellites, soit leurs capacités d'orientation, et de la résolution minimale autorisée des images obtenues par acquisition. Les entités géographiques qui interviennent lors d'acquisition de **mailles** sont illustrées dans la figure 1.

Une première phase en amont du problème de grande couverture consiste à établir l'ensemble de satellites et leurs passages qui peuvent être utilisés pour acquérir la zone d'intérêt.

Pour chaque passage d'un satellite sur une zone d'intérêt, le processus d'acquisition suit le cycle illustré par la figure 2 et constitué des phases suivantes :

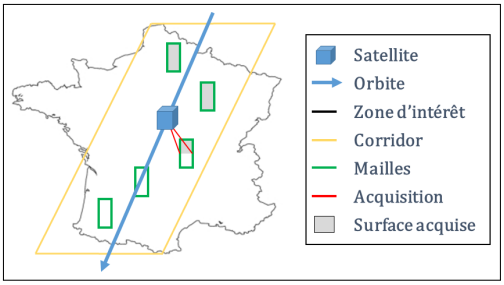


Figure 1: Schéma d'acquisition d'un ensemble de mailles pendant un passage de satellite

1. **Maillage.** Découpage en **mailles** du corridor du passage.
2. **Requête d'acquisition.** Sélection d'un sous ensemble des mailles et envoi d'une requête d'acquisition de ces mailles au centre de planification qui opère le satellite.
3. **Acquisition.** Suite au calcul du plan d'acquisition effectif du satellite par le centre de planification, une réponse contenant les mailles de la requête qui ont été retenues pour acquisition est formulée.
4. **Validation.** Réponse du centre de planification contenant l'état de validation ou de rejet des mailles et les images correspondantes.

La programmation d'une grande couverture est assurée par un opérateur indépendant des centres de planification. Cet opérateur s'occupe du maillage et l'émission des **requêtes d'acquisition** vers les centres de planification qui opèrent les satellites. Notons que des requêtes d'acquisition plus urgentes que celles émises dans le cadre de la grande couverture peuvent être traitées en priorité par les centres de planification. En conséquence, les requêtes d'acquisitions de mailles émises vers les centres de planification ne sont pas garanties de mener à l'acquisition de toutes ces mailles. De plus, l'acquisition réussie d'une maille par un satellite dépend des conditions météorologiques au moment de l'acquisition. L'image obtenue par une acquisition sera jugée nette ou floutée suivant son taux de nébulosité constaté. L'acquisition d'une maille par un centre de planification peut donc échouer en raison de différences entre les prévisions météorologiques effectuées à l'envoi du plan d'acquisition au satellite et les conditions réelles au moment du passage. Pour chaque maille, trois états sont à distinguer :

- **Active** : En attente de planification sur un passage.
- **Acquise** : Incluse dans le plan d'acquisition d'un satellite et une image correspondante sera acquise.
- **Validée** : Acquisition réussie, une image nette de la maille est disponible.

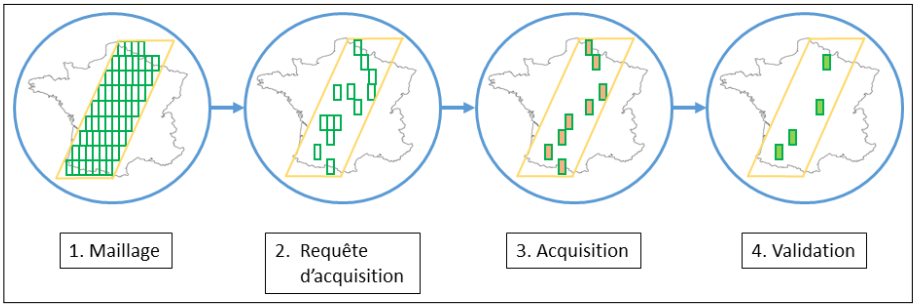


Figure 2: Processus d'acquisition et de validation pour un passage de satellite

Le problème des grandes couverture consiste à trouver un ensemble de mailles à acquérir pour chaque passage des satellites qui minimise à la fois **le temps** et **les ressources satellites** nécessaires pour obtenir des images nettes de la totalité de la zone d'intérêt. Le temps est mesuré par le **nombre de passages** des satellites de la constellation depuis le début de la grande couverture ou l'équivalent en nombre de jours. Les ressources satellites utilisées jusqu'à la complétion de la grande couverture sont mesurées par une métrique appelée **gâchis** définie comme le surplus d'acquisition de surfaces déjà acquises au cours de la grande couverture. Par exemple le gâchis minimal de 0% est atteint si toutes les acquisitions validées de la grande couverture ne concernent que des surfaces qui ont été acquises une seule fois.

2.2 Etat de l'art

Le problème de grande couverture tel que présenté dans la section précédente n'est pas traité dans son ensemble dans le domaine applicatif. Actuellement, le pré-traitement suivant est effectué :

1. découpage de la zone d'intérêt en plusieurs sous-zones,
2. répartition exclusive des sous-zones entre les centres de planification,
3. acquisition par les centres de planification de leur sous-zone respective.

Ce pré-traitement évite de submerger les centres de planification de requêtes d'acquisition sur la totalité de la zone d'intérêt. La coordination entre les centres de planification n'est pas nécessaire car chaque sous-zone est traitée de façon distincte. La répartition entre les centres de planification permet ainsi d'éviter la complexité du problème de grande couverture sur la zone d'intérêt entière. Cette approche implique cependant que certains passages de satellites seront ignorés pour des sous-zones car leur centre de planification s'est vu attribué une autre sous-zone. De nombreuses opportunités d'acquisition sont ignorées pour simplifier le problème et le temps de complétion de la mission augmente fortement en conséquence. Le problème des grandes couvertures considéré en sa totalité, augmente fortement la combinatoire mais permet la coordination des centres de planification pour une résolution plus rapide et moins coûteuse en ressources satellites.

La planification de plans d'acquisitions pour un ensemble de satellites est un problème connu. [2, 5, 8] proposent des techniques différentes pour la planification. [7] compare 13 techniques d'optimisation pour ce problème et obtient de meilleurs résultats avec un algorithme d'optimisation par recuit simulé. Sur la base de ces travaux, [3] propose un système multi-agent adapté au traitement de nouvelles requêtes d'acquisition pour l'optimisation d'un ensemble de plans d'acquisitions. Ces exemples considèrent cependant les acquisitions des satellites comme des tâches indépendantes à réaliser sur un horizon de planification court. Le problème considéré est celui d'un centre de planification qui a connaissance des plans d'acquisitions des satellites et planifie des acquisitions en conséquence.

Il existe peu de références dans la littérature sur le problème des grandes couvertures dans sa globalité tel que nous souhaitons le considérer dans ce papier. [4] cherche à améliorer le traitement opérationnel, il propose un algorithme glouton pour une répartition dynamique entre les centres de planification de sous-zones obtenues par découpage de la zone d'intérêt. [16] décrit des phases de discrétisation de la zone d'intérêt, découpage en bandes et planification d'acquisitions par algorithme génétique, avec de bons résultats obtenus par comparaison à un algorithme glouton. [10] et [13] considèrent un problème de coordination de satellites hétérogènes pour l'observation de catastrophes qui couvrent de grandes zones. Les auteurs proposent respectivement un algorithme par recuit simulé et un algorithme génétique pour l'optimisation du découpage et de l'acquisition de la zone d'intérêt. Le problème géométrique traité consiste à définir un ensemble de bandes qui couvrent la longueur de la zone d'intérêt et qui sont attribuées à des satellites hétérogènes. Ce problème est différent de celui des grandes couvertures car les requêtes d'acquisition sont considérées urgentes et toujours acceptée par les centres de planification, et les échecs d'acquisition dûs aux conditions météorologiques sont ignorés.

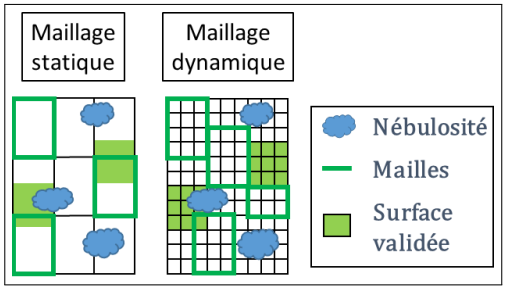


Figure 3: Exemple d'organisation de mailles dans la même sous-partie d'un corridor d'un satellite selon le type de maillage statique *vs.* dynamique

Dans le cadre opérationnel, un maillage est effectué par les centres de planification une fois qu'une sous-zone de la zone d'intérêt leur a été attribuée. Cette technique, que nous nommerons **maillage statique**, consiste à construire pour chaque passage de satellite, un ensemble de mailles qui peuvent être acquises. Une grille de mailles mitoyennes qui décrivent un pavage régulier sur le corridor du passage est calculée à la réception de la grande couverture. Les mailles obtenues sont fixes et de taille constante dans un même passage. Cette taille est définie suivant le maximum des capacités d'acquisition du satellite pour une seule maille. Le maillage statique permet de couvrir de mailles une grande partie du corridor de chaque passage. Le pavage régulier de mailles garantie l'absence de chevauchement entre elles et donc une bonne utilisation des ressources satellites. Cependant, le positionnement et les dimensions fixes des mailles impliquent qu'elles ne peuvent pas être adaptées aux conditions d'acquisition de la grande couverture durant la mission. Ces contraintes de placement peuvent entraîner des acquisitions surnuméraires de sous-zones validées. De plus, des sous-parties de corridor peuvent être inaccessibles pour acquisition en raison des contraintes de construction de la grille de mailles par rapport aux bords du corridor.

Nous proposons une nouvelle technique de maillage que nous nommerons **maillage dynamique**. Dans le cadre de ce maillage une grille de sous-zones élémentaires appelées **cellules** est construite sur la zone d'intérêt. Cette discrétisation permet un positionnement plus précis des mailles dans les corridor des passages, et notamment sur leurs bords. La position des mailles et leurs dimensions peuvent également être adaptés sur la grille de cellules pour une meilleure utilisation des ressources satellites. Des sous-parties de corridor validées ou nuageuses peuvent être ignorées en faveur de sous-parties non validées et bénéficiant de meilleur probabilité de validation. Les types de maillage, statique et dynamique, sont illustrés en figure 3.

Le problème de répartition de mailles sur une grille de cellules pour chaque passage des satellites est complexe. Un grand nombre d'entités est considéré et la taille de l'espace de recherche est liée aux placements et dimensions possibles des mailles dans l'ensemble des passages. Nous avons choisi d'implémenter un système multi-agent pour l'optimisation du problème des grandes couvertures par maillage dynamique. Le système Glimpse présenté dans cet article est basé sur la théorie AMAS[6]. Les systèmes multi-agents coopératifs ont montré leur efficacité dans le traitement de problèmes d'optimisation à forte combinatoire qui comportent un grand nombre d'entités grâce à l'émergence de fonctions globales de bonne qualité par coopération d'agents distribués aux comportements locaux simples [15]. Le fonctionnement du système Glimpse est présenté section 3.

2.3 Formalisation du problème

Dans cette partie nous proposons une formalisation du problème des grandes couvertures avec discrétisation de la zone d'intérêt en cellules. Étant donné :

Une zone Z sur laquelle est placée une grille de largeur X et de longueur Y de cellules $c_{11},\ldots,c_{ij},\ldots,c_{XY}$, avec i et j les coordonnées géographiques du coin inférieur gauche la cellule c_{ij}.

L'ensemble des cellules $C_{int}^{Z} = \{(i,j)|c_{ij} \cap Z \neq 0\}$ à l'intérieur ou partiellement à l'intérieur de la zone. Les cellules sont caractérisées par :

- une largeur l et une longueur L en kilomètre.
- Un statut d'acquisition $st(c_{ij},t) \in \{Active, Acquired, Validated\}$ indiquant respectivement si la cellule c_{ij} est en attente de planification, incluse dans une maille acquise ou incluse dans une maille validée au temps t.

L'ensemble de satellites $S = s_1,\ldots,s_k,\ldots,s_K$ de largeur d'acquisition minimum $f_k^{min} * l$ et de largeur d'acquisition maximum $f_k^{max} * l$ avec $f_k^{min}, f_k^{max} \in N$; et de longueur d'acquisition minimum $g_k^{min} * L$ et de longueur d'acquisition maximum $g_k^{max} * L$ avec $g_k^{min}, g_k^{max} \in N$.

Des passages P_k du satellite s_k connus jusqu'à un horizon de planification H en nombre de passages qui limite les erreurs de prévision météorologiques, et caractérisés par :

- une date d'occurrence t_{P_k}
- un ensemble de cellules accessibles $C_{P_k} = \{(i,j)|c_{ij} \in Corridor_{P_k}\}$
- un seuil maximal de nombre mailles $|M_{P_k}| < \frac{|C_{P_k}|}{f_k^{max} * g_k^{max}}$ afin d'éviter la surcharge de mailles dans la requête d'acquisition. Ce seuil est défini comme le nombre de mailles de taille maximale suffisant pour couvrir toutes les cellules du corridor du passage.

But. Le but de la résolution est de déterminer par passage un ensemble de mailles $M_{P_k} = (i_0, j_0, f, g)$, avec i_0, j_0 les coordonnées du coin inférieur gauche de la maille et $f_k^{min} \leq f \leq f_k^{max}$ et $g_k^{min} \leq g \leq g_k^{max}$, à inclure dans une requête d'acquisition.

Objectifs. L'objectif principal est la minimisation du temps nécessaire pour réaliser des acquisitions validées qui recouvrent l'intégralité de la zone d'intérêt. Cet objectif se traduit par la validation de toutes les cellules de la zone d'intérêt. La complétion du chantier est atteinte à un moment noté t_{comp}.
La mesure du progrès dans la complétion du chantier est notée $Progress(Z,t) = \left(\frac{\sum_{i,j|c_{ij} \in C_{int}^Z \wedge st(cij,t)=Validated} c_{ij}}{\sum_{i,j|c_{ij} \in C_{int}^Z} c_{ij}} \right)$
avec $Progress(Z, t_{comp}) = 1$.

L'objectif secondaire de l'optimisation est la minimisation du gâchis, soit du surplus d'acquisition de surfaces déjà acquises au cours de la grande couverture. Avec la discrétisation de la zone d'intérêt en cellules une mission sans gâchis peut être représentée comme l'acquisition de toutes les cellules une seule fois. On note le gâchis

$Waste(Z,t) = \frac{Nb_{acq}}{Nb_{C_{acq}}} - 1$
avec Nb_{acq} le nombre total d'acquisitions de cellules de la zone Z depuis le début du chantier jusqu'à l'instant t, et $Nb_{C_{acq}}$ le nombre de cellules de Z acquises au moins une fois depuis le début du chantier jusqu'à l'instant t. Par exemple, si toutes les cellules de Z ont été acquises une seule fois, $Nb_{acq} = Nb_{C_{acq}}$ et $Waste(Z,t) = 0$.

3 Le système Glimpse

Glimpse compte trois types d'agents : les agents Cellule, Maille et Passage. Ces agents suivent un cycle de vie constitué de phases synchronisées :

1. **Perception**, réception des messages et mise à jour des données internes,
2. **Décision**, traitement des données et choix d'actions à effectuer,
3. **Action**, envoi de messages et modification de l'environnement de l'agent.

Dans cette section, nous décrivons les agents du système par leurs objectifs, leurs interactions et leurs comportements représentés par des algorithmes de décision.

Le maillage statique couramment utilisé et décrit en section 2.2 est une heuristique de placement des mailles au sein des passages qui présente l'avantage de ne pas autoriser de chevauchements entre mailles. Le maillage dynamique proposé dans Glimpse apporte un degré de liberté supplémentaire aux mailles qui peuvent être placées sur n'importe quelle coordonnée (i, j) appartenant au corridor de leur passage. Le premier objectif des comportements des agents dans Glimpse est de trouver pour chaque agent Maille un placement coopératif prenant en compte la criticité des agents Cellule qui seront acquis avec l'agent Maille.

3.1 L'agent Cellule

Dans Glimpse, la **criticité des agents Cellule** représente l'importance d'acquisition de la zone couverte par la cellule pour un passage donné. Les criticités de plusieurs cellules peuvent être comparées pour déterminer leur ordre d'importance d'acquisition sur un passage. La comparaison suit un ordre leximin : les critères sont normalisés entre eux et comparés dans un ordre fixe jusqu'à trouver un critère différenciateur qui indique l'agent Cellule le plus critique. La comparaison de critères issus des données du problème implique la définition d'une importance relative de ces données. Dans le cadre de ce travail, l'importance des critères est issue d'analyses d'experts du domaine. La criticité des agents Cellule dans un passage donné est basée sur les critères suivants, dans leur ordre leximin :

1. **Reste à acquérir.** La criticité d'une cellule déjà validée est nulle.
2. **Probabilité de validation.** Probabilité de validation de la cellule si acquise dans ce passage. Pour une cellule acquise dans des passages antérieurs, elle est pondérée par les probabilités de validation dans ces passages.
3. **Probabilité de validation future.** Probabilité de validation de la cellule dans les passages restants sur l'horizon de planification pour lesquels le corridor couvre cette cellule.
4. **Proximité de cellules validées.** Ratio de cellules voisines déjà validées sur des passages antérieurs.
5. **Opportunités d'acquisition.** Ratio de passages restants sur l'horizon de planification pour lesquels le corridor couvre cette cellule.

Les agents Cellule ont pour objectif d'être acquis puis validés sur n'importe quel passage sur l'horizon de planification qui contient la cellule dans son corridor. Le comportement d'un agent Cellule se base sur ses connaissances locales, notamment les agents Maille visibles par l'agent Cellule dans chaque passage pour maximiser la probabilité d'acquisition de la cellule. La visibilité des agents Cellule est limitée aux mailles qui les recouvrent ou sont directement adjacentes sur la grille de cellules. Le comportement des agents Cellule suit l'algorithme 1.

La résolution de l'insatisfaction des agents Cellule se traduit par une notion d'adhésion à des mailles sur les passages connus. L'adhésion d'une cellule à une maille décrit

algorithm 1 Comportement agent Cellule

```
1: for each P_k connu do
2:    tri des mailles M_{P_k} connues par ordre de criticité
3:    for each m_{P_k} ∈ M_{P_k} triées do
4:       if cellule non adhérée et aucune demande d'adhésion then
5:          requête d'adhésion à la maille m_{P_k}
6:       end if
7:    end for
8: end for
```

la volonté de faire partie des cellules couvertes par cette maille dans le passage correspondant. Une cellule ne peut adhérer qu'à une seule maille par passage, soit la maille qu'elle préfère et qui a accepté sa requête d'adhésion parmi les mailles qu'elle connaît dans ce passage. L'objectif de l'adhésion est de promouvoir la répartition géographique des mailles dans le corridor des passages. Cette répartition est nécessaire en maillage dynamique pour éviter le regroupement et chevauchement des mailles d'un même passage sur les cellules les plus critiques dans ce passage. Les agents Cellule choisissent la maille à laquelle ils préfèrent adhérer suivant un tri par criticité des agents Maille visibles dans leur voisinage. La **criticité des agents Maille** dépend des criticités des agents Cellule qui adhèrent à elle, triées par ordre décroissant.

3.2 L'agent Maille

Les agents Maille ont pour objectif d'être acquises puis validées dans leur passage. Pour ce faire, elles utilisent les requêtes d'adhésion des agents Cellule dans leur voisinage pour altérer leur géométrie et leur placement dans le corridor de façon à maximiser leur probabilité d'acquisition et de validation. Une requête d'adhésion traitée par un agent Maille peut faire l'objet de trois types d'altération de sa géométrie sur la grille de cellules :

- **Déplacement.** Les coordonnées d'origine (i_0, j_0) de la maille sont modifiées.
- **Extension.** Le facteur f de largeur ou le facteur g de longueur de la maille est étendu dans un sens donné.
- **Réduction.** Le facteur f de largeur ou le facteur g de longueur de la maille est réduit dans un sens donné.

Le comportement d'un agent Maille est décrit dans l'algorithme 2. L'inclusion d'une cellule fait l'objet d'une extension de la maille tant qu'elle n'a pas encore atteint sa largeur maximale f_k^{max} ou longueur maximale g_k^{max} en nombre de cellules, suivant la direction relative de la cellule qui émet une requête d'adhésion. Lorsque la largeur ou longueur maximale est atteinte un déplacement peut être effectué. Ce déplacement causerait l'inclusion à la maille d'un ensemble de cellules et l'exclusion de cellules sur le bord opposé au sens du déplacement. Une comparaison entre les criticités des cellules à inclure suite au déplacement potentiel et les criticités des cellules à exclure est simulée au sein de l'agent Maille, en ligne 2 de l'algorithme 2. Le déplacement est effectué si la criticité des cellules qui adhéreront à la maille suite au déplacement est supérieure car cette altération augmente la criticité de la maille elle-même. La géométrie libre des mailles permet aussi la réduction des bords de la maille qui ne comprennent pas de cellules en adhésion. Ces réductions et l'adhésion des cellules aux mailles les plus critiques dans leur voisinage favorisent la formation de mailles juxtaposées. Ces juxtapositions diminuent le gâchis dû aux chevauchements de mailles et aux acquisitions surnuméraires de mêmes cellules au sein d'un passage.

algorithm 2 Comportement agent Maille

```
1: for each requête d'adhésion de c_ij do
2:    if criticité de c_ij supérieure à celle de la cellule exclue la plus critique et qui adhère à la maille then
3:       inclusion de la cellule c_ij
4:       acceptation de la requête d'adhésion de c_ij
5:    else
6:       rejet de la requête d'adhésion de c_ij
7:    end if
8:    réduction des bords sans adhésion
9: end for
```

3.3 L'agent Passage

Les comportements des agents Cellule et Maille ont pour objectif de placer les mailles dans leur environnement local pour maximiser leur probabilité d'être acquises et validées. Les altérations et déplacements des mailles sont basés sur leur voisinage de cellules qui émettent des requêtes d'adhésion. Ces comportements ne suffisent pas pour garantir la couverture des cellules les plus critiques à l'échelle du corridor d'un passage. Des sous-parties du corridor peuvent ainsi piéger les mailles dans un optimum de criticité local. La création de mailles par les agents Passage dans leur corridor correspond à un saut dans l'espace de recherche similaire à une augmentation de température d'une optimisation par recuit simulé[11], ou une mutation d'un algorithme génétique[12].

Les agents Passage ont pour objectif de maximiser la couverture des cellules les plus critiques au sein du corridor d'un passage. Leur comportement est lié au placement des mailles placées sur le corridor et doit permettre aux cellules les plus critiques d'être couvertes de façon dynamique, par exemple suite à une mise à jour des données météorologiques qui vient modifier la criticité des cellules. Le comportement de l'agent Passage est décrit dans l'algorithme 3.

Afin de laisser le placement local des mailles s'effectuer par les interactions entre agents Cellule et Maille, le

algorithm 3 Comportement agent Passage

```
1: if stabilisation des mailles M_{P_k} then
2:   for each cellule c_{ij} ∈ C_{P_k} do
3:     if c_{ij} non couverte et de forte criticité relative aux
       cellules couvertes then
4:       création d'un maille m = (i, j, 1, 1)
5:       if seuil maximal de nombre de mailles dépassé
         then
6:         suppression de la maille la moins critique
7:       end if
8:     end if
9:   end for
10: end if
```

comportement des agents Passage n'est effectué qu'après la stabilisation des mailles du passage. Les mailles d'un passage sont stabilisées quand toutes les cellules couvertes par des mailles adhèrent à l'une d'elles, et suite à un délai suffisant pour garantir qu'aucun message, notamment des requêtes d'adhésion, n'est en transit ou en cours de traitement. Suite à la stabilisation des mailles, les cellules appartenant au corridor du passage sont parcourues jusqu'à trouver une cellule non couverte ayant une criticité supérieure au point de référence de criticité de la première maille candidate au remplacement, soit la cellule la plus critique de la maille la moins critique. Le remplacement d'une maille est nécessaire pour garantir un nombre de mailles dans le passage inférieur au seuil maximal défini comme le nombre de mailles suffisant pour couvrir entièrement le corridor du passage. Ce seuil maximal a pour objectif d'éviter la surcharge de requêtes d'acquisition de mailles au centre de planification.

4 Résultats et discussion

Cette section présente l'évaluation et les performances du système Glimpse pour l'optimisation de problèmes de grandes couvertures simulés. Les résultats d'expérimentations sont suivis par une discussion et nos perspectives d'amélioration du système.

4.1 Méthodologie d'expérimentation

Scénarios de mission. Les scénarios de mission considérés comportent les entrées nécessaires au traitement du problème par le système Glimpse, soit les éléments suivants :

- La requête de programmation client contenant :
  - La zone d'intérêt à acquérir,
  - Les centres de planification,
  - Les satellites et leurs caractéristiques,
  - Les passages des satellites sur la zone.
- Les données météorologiques pour les cellules de chaque passage,
- La taille des cellules de la grille,
- La taille de l'horizon de planification.

Modules de simulation. Le processus d'acquisition décrit en section 2 (figure 2) suppose l'action d'éléments extérieurs à la résolution du problème, tels que les centres de planification. La planification des mailles à acquérir par un centre de planification par exemple est dépendante des autres tâches du satellite qui sont inconnues au système de programmation de la grande couverture. Pour les besoins d'expérimentation nous avons simulé les comportements des centres de planification par des modules de décision relatifs aux phases d'acquisition et de validation.

La phase d'acquisition comporte une sélection de mailles qui ont fait l'objet de requêtes d'acquisition à inclure au plan d'acquisition du satellite. Cette planification est effectuée par un algorithme de type glouton hiérarchique[1, 9] qui représente l'état de l'art opérationnel dans le domaine de la planification d'acquisitions par satellites à court-terme. Le comportement de cet algorithme consiste à trier les mailles du passage par ordre d'importance, soit ici par criticité, et de sélectionner des mailles jusqu'à atteindre une limite fixe de capacité d'acquisition du satellite. Pour la phase de validation, des mailles sont sélectionnées en quantité relative à la probabilité de validation moyenne des mailles du passages et dans l'ordre décroissant de criticité.

Éléments de comparaison. Comme décrit en section 2.2, l'état de l'art pour optimisation du problème des grandes couvertures consiste à sous-diviser le problème et le répartir entre centres de planifications distincts. Pour évaluer notre approche sur la résolution globale du problème nous avons choisi de comparer une approche basée sur un maillage statique; et les mailles obtenues par le système Glimpse, placées par maillage dynamique sur une grille de cellules qui couvre la zone d'intérêt. L'algorithme de maillage statique utilisé crée des mailles de taille maximale autorisée juxtaposées suivant un pavage régulier de sorte à couvrir la zone d'intérêt. De plus nous considérons des scénarios de mission dont les satellites sont caractérisés par les dimensions des mailles pouvant être acquises. Des scénarios seront décrits comme comportant des satellites **homogènes** si les mailles qu'ils acquièrent sont de mêmes dimensions, et **hétérogènes** sinon.

4.2 Expérimentations

Pour chaque expérimentation les métriques observées sont, par ordre d'importance :

1. Le temps de complétion à 95% de la grande couverture, en nombre de passage.
2. Le gâchis, surplus d'acquisition de surfaces déjà acquises au cours de la grande couverture, en pourcentage.

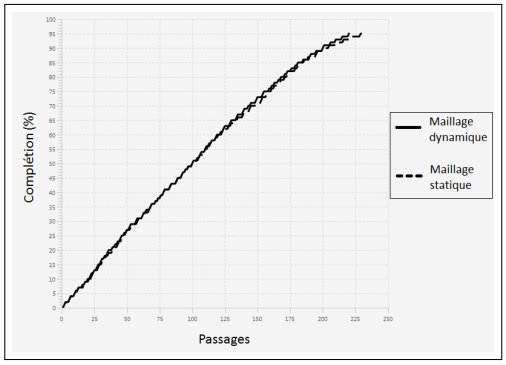


Figure 4: Évolution du taux de complétion par type de maillage pour un scénario aux satellites homogènes

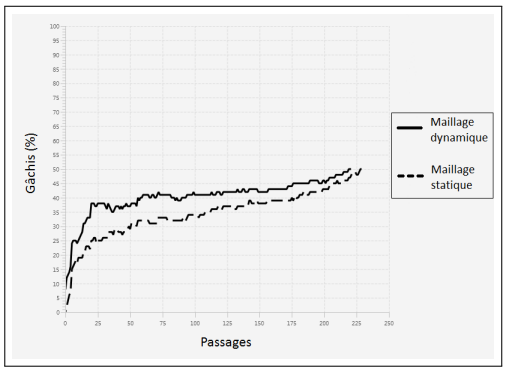


Figure 5: Évolution du gâchis par type de maillage pour un scénario aux satellites homogènes

La progression de la résolution au cours du temps est représentée par la validation successive de passage. La courbe de progression des métriques ainsi que leur mesure finale est présentée. Les scénarios étudiés présentent 9642 agents Cellule, 300 agents Maille concurrents en moyenne et 10 agents Passage concurrents.

La première expérimentation propose un scénario de mission avec les caractéristiques suivantes :

- L'Australie en zone d'intérêt,
- Des cellules de bord 30km,
- 4 satellites **homogènes** aux mailles de largeur 60km et longueur 120km,
- Un horizon de planification de 10 passages.

Les courbes d'évolution du taux de complétion et du gâchis au cours de la résolution de la grande couverture sont présentées respectivement en figures 4 et 5. Le tableau de la figure 6 contient les mesures finales pour ces métriques à 95% de complétion.

La deuxième expérimentation propose de réutiliser ce scénario en modifiant les dimensions de mailles acquérables par les satellites afin de les rendre **hétérogènes**. Les 4 satellites hétérogènes de ce nouveau scénario peuvent respectivement acquérir des mailles qui ont les dimensions suivantes : 60km/90km, 60km/120km, 90km/150km et 90km/180km.

	Passages	Gâchis(%)
Maillage dynamique	222	50
Maillage statique	227	50

Figure 6: Nombre de passages planifiés et gâchis à 95% de complétion de la grande couverture

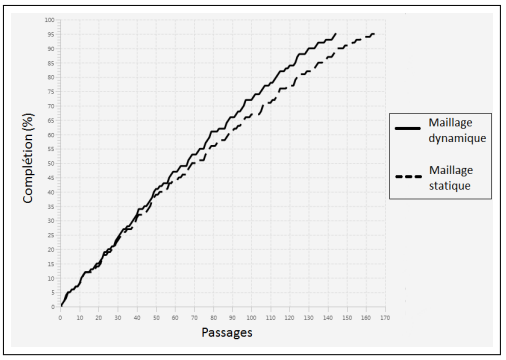


Figure 7: Évolution du taux de complétion par type de maillage pour un scénario aux satellites hétérogènes

Les courbes d'évolution, du gâchis et leurs mesures finales sont respectivement présentées en figures 7, 8 et 9.

4.3 Discussion

Les résultats obtenus montrent les avantages du maillage dynamique obtenu par Glimpse par rapport à un maillage statique traditionnel. Pour des scénarios aux satellites homogènes tels que présentés dans la première expérimentation, nous constatons une diminution du temps en nombre de passage nécessaire pour atteindre 95%

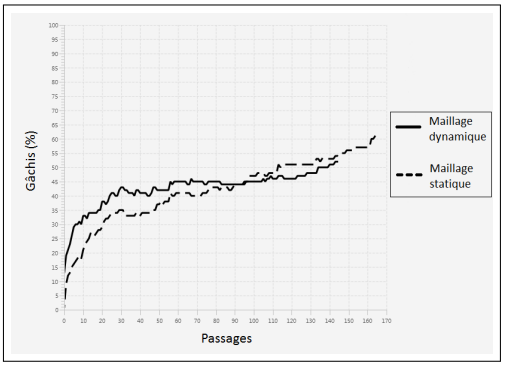


Figure 8: Évolution du gâchis par type de maillage pour un scénario aux satellites hétérogènes

	Passages	Gâchis(%)
Maillage dynamique	144	52
Maillage statique	163	61

Figure 9: Nombre de passages planifiés et gâchis à 95% de complétion de la grande couverture

de complétion. Cette réduction peut s'expliquer par la précision du maillage dynamique qui permet à Glimpse de placer des mailles qui n'appartenaient pas à la grille pré-construite du maillage statique, notamment sur les bords des corridors où les contraintes de placement sont plus fortes. L'évolution du gâchis montre que l'autorisation de chevauchements entre mailles du maillage dynamique entraîne une augmentation du gâchis en début de mission par rapport au maillage statique. Ces chevauchements n'existent pas en maillage statique dans le cas de satellites homogènes. En effet, dans ce cas, la grille pré-construite par le maillage statique est en superposition sur tous les passages du scénario. Le seul gâchis observé pour le maillage statique pour satellites homogènes provient donc de la ré-acquisition de mailles identiques sur des passages suites à des rejets sur des passages antérieurs. Le taux de rejet des acquisitions en maillage statique est plus important qu'en maillage dynamique car les mailles sont moins bien placées dans les corridors des passages. Ce taux de rejet du maillage statique élevé compense les chevauchements du maillage dynamique, le gâchis observé à la fin de la mission est identique pour les deux approches dans le cas de satellites homogènes.

La répartition de mailles pour des satellites homogènes est le meilleur cas de figure du maillage statique grâce à la superposition de toutes les mailles entre passages. Dans le cadre opérationnel des grandes couvertures, les centres de planification opèrent des satellites différents. La répartition de mailles pour des satellites hétérogènes est donc plus proche des conditions d'expérimentations opérationnelles. Les résultats obtenus dans la deuxième expérimentation qui comporte des satellites hétérogènes montrent une augmentation du gâchis obtenu dans le cas du maillage statique. Les grilles pré-construites par le maillage statique présentent en effet des chevauchements entre mailles dus aux dimensions variables entre satellites. Des cellules validées dans des passages antérieures sont donc acquises à nouveau car les mailles ne peuvent pas être re-dimensionnées. De plus, l'acquisition de cellules validées diminue le nombre de cellules non validées acquises dans un passage par limite de capacité d'acquisition du satellite, ce qui retarde le temps de complétion total. Dans le cas du maillage dynamique effectué par le système Glimpse le gâchis reste constant par rapport à la première expérimentation car les mailles peuvent altérer leur géométrie pour éviter de recouvrir des cellules déjà validées. En évitant ces cellules, les mailles maximisent le nombre de cellules pouvant être acquises dans la limite de capacité d'acquisition du satellite. Le temps de complétion et le gâchis obtenus par maillage dynamique avec Glimpse sont équivalents ou meilleurs à ceux obtenus par maillage statique pour des scénarios aux satellites homogènes, et meilleurs pour des scénarios aux satellites hétérogènes. Ces résultats montrent l'adéquation de l'approche dynamique proposée pour l'optimisation du problème des grandes couverture.

4.4 Perspectives

Le traitement actuel du problème par Glimpse est fortement dépendant de la valeur de criticité des agents Cellule. Cette criticité est calculée en utilisant des critères identifiés par des experts du domaines. Nos prochaines études vont venir remplacer le tri par leximin présent dans le calcul de criticité des agents de Glimpse par des méthodes d'évaluation plus performantes. Nous souhaitons améliorer l'estimation de la criticité en intégrant des techniques d'apprentissage par renforcement aux agents. Une évaluation plus précise de la probabilité d'acquisition des cellules aidera la répartition des mailles entre passages pour maximiser les probabilités d'acquisitions réussies sur l'horizon de planification. Le seuil du nombre de mailles envoyées aux centres de planification peut également faire l'objet d'une optimisation. En observant le nombre de mailles intégrées au plan du satellite pour des requêtes d'acquisition, un nombre de mailles mieux adapté aux contraintes du centre de planification sera estimé pour les passages suivants.

De plus, les capacités du maillage dynamique permettent de traiter le problème des grandes couvertures avec une validation dite partielle. Ce type de validation ne concerne plus les mailles mais les cellules. En validation partielle, la validation d'une cellule est basée sur la netteté de la sous-zone lui correspondant sur l'image obtenue. En conséquence, les images partiellement floues obtenues ne sont plus entièrement rejetées mais en partie validées de même que les cellules relatives aux sous-parties nettes. Des cellules isolées peuvent exister suite à leur rejet individuel, mais le maillage dynamique permet de créer des mailles de tailles variables adaptées à l'acquisition d'un petit nombre de cellules en utilisant des ressources satellites minimales. La validation partielle permet un traitement plus ciblé des images qui diminue le besoin de réacquérir des sous-zone de la zone d'intérêt et améliore le temps de complétion.

Enfin, le problème sera étendu pour améliorer son réalisme et ajouter des contraintes opérationnelles. Les différences entre conditions météorologiques prévues et réelles ainsi que les profils d'occupation des plans des satellites seront modélisés. Les temps de manoeuvres de rotation du satellite pour l'orienter vers les mailles à acquérir seront estimés

et pris en compte dans le placement des mailles en maillage dynamique. La montée à l'échelle sera également évaluée. Le système Glimpse peut actuellement traiter un problème qui nécessite des interactions locales entre des milliers d'agents tels que ceux présentés en section 4.2. Des scénarios à l'échelle de pays plus grand et à taille de cellule plus petite seront construits. La diminution de la taille des cellules permettra de placer les mailles dynamiques de façon plus précise sur les cellules de forte criticité dans les passages.

5 Conclusion

Cet article présente notre méthode d'optimisation du problème des grandes couvertures basée sur le maillage dynamique par système multi-agent coopératif. La méthode de résolution dans l'industrie se base sur l'expertise humaine pour sous-diviser le problème afin de ne pas avoir à traiter son aspect combinatoire. Nous avons implémenté un système multi-agent coopératif nommé Glimpse suivant la théorie AMAS. Afin d'évaluer Glimpse nous avons comparé le maillage dynamique employé au maillage statique couramment utilisé en contexte opérationnel. Les résultats que nous avons obtenu sur des scénarios de grandes couvertures simulés dans le cas général de mailles à dimensions hétérogènes montrent que le maillage dynamique permet d'atteindre 95% de complétion de la requête sur une période plus courte tout en diminuant le gâchis des ressources satellites par rapport au maillage statique.

Nos futurs travaux vont venir ajouter des données inconnues au problème telles que la probabilité de réalisation de valeurs de météo prédites et la probabilité de réponse positive à une requête d'acquisition de maille par un centre de planification. Ces données sont difficiles à modéliser et nous envisageons d'intégrer des techniques d'apprentissage par renforcement aux agents du système Glimpse afin de les estimer. Cette fusion aura pour objectif de mieux calculer l'importance d'acquisition de cellules sur des passages spécifiques dans un contexte incertain.

Remerciements

Les auteurs souhaitent remercier l'IRT Saint-Exupéry pour le financement de ces travaux.

References

[1] J.-C. Agn, N. Bataille, D. Blumstein, E. Bensana, and G. Verfaillie. Exact and approximate methods for the daily management of an earth observation satellite. *RAIRO–Oper. Res.*, 41(4):381–398, 2007.

[2] E. Bensana, M. Lemaitre, and G. Verfaillie. Earth observation satellite management. *Constraints*, 4(3):293–299, 1999.

[3] J Bonnet. *Multi-Criteria and Multi-Objective Dynamic Planning by Self-Adaptive Multi-Agent System, Application to Earth Observation Satellite Constellations*. PhD thesis, Université Paul Sabatier-Toulouse III, 2017.

[4] W. Fei and L. Zhi. Research on the optimization method of dynamic partitioning area target for earth observation satellites. *Procedia Engineering*, 15:3159–3163, 2011.

[5] J. Frank, A. Jonsson, R. Morris, D.-E. Smith, and P. Norvig. Planning and scheduling for fleets of earth observing satellites. 2001.

[6] M.-P. Gleizes. Self-adaptive complex systems. In *European Workshop on Multi-Agent Systems*, pages 114–128. Springer, 2011.

[7] A. Globus, J. Crawford, J. Lohn, and A. Pryor. A comparison of techniques for scheduling earth observing satellites. In *AAAI*, pages 836–843, 2004.

[8] R. Grasset-Bourdel. *Planification dynamique et réactive pour des satellites agiles d'observation de la Terre*. PhD thesis, Ph. D. thesis, Onera, 2011.

[9] N. G. Hall and M. J. Magazine. Maximizing the value of a space mission. *European journal of operational research*, 78(2):224–241, 1994.

[10] N. Holvoet, W. Vongsantivanich, S. Chaimatanan, and D. Delahaye. Mission planning for a non-homogeneous earth observation satellite constellation for disaster response. In *SpaceOps 2018*, 2018.

[11] S. Kirkpatrick, C.-D. Gelatt, and M.-P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[12] J. McCall. Genetic algorithms for modelling and optimisation. *Journal of Computational and Applied Mathematics*, 184(1):205–222, 2005.

[13] XN Niu, H Tang, and LX Wu. Multi-satellite observation scheduling for large area disaster emergency response. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42(3), 2018.

[14] D. W. Rhind and DR F. Taylor. *Cartography Past, Present and Future: A Festschrift for FJ Ormeling*. Elsevier, 2013.

[15] N. Verstaevel. *Self-organization of robotic devices through demonstrations*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2016.

[16] Y. Xu, X. Liu, R. He, Y. Chen, and Y. Chen. Multi-objective satellite scheduling approach for very large areal observation. In *IOP Conference Series: Materials Science and Engineering*, volume 435, page 012037. IOP Publishing, 2018.

Cooperative navigation of mobile robots fleet

Daravuth Koung[1,2,3] Isabelle Fantoni[2] Olivier Kermorgant[1,2] Lamia Belouaer[3]

[1] École Centrale de Nantes
[2] Laboratoire des Sciences du Numérique de Nantes, UMR CNRS 6004
[3] E-COBOT

[1,2] firstname.lastname@ls2n.fr
[3] l.belouaer@e-cobot.com

Résumé

Dans ce papier, nous considérons la commande coopérative pour une flotte de robots mobiles. Nous présentons un état de l'art des travaux les plus répondus dans ce domaine. La confrontation de l'état de l'art avec les exigences industirelles nous a permis de comprendre les limites de ces approaches.

Mots Clefs

Commande coopérative, commande en formation, robots mobiles.

Abstract

In this paper, we focus on the cooperative control for the fleet of mobile robots. We present the state of the art of the most significant works in this field. The correlation between the state of the art and the industry requirements has allowed us to understand the limits of these approaches.

Keywords

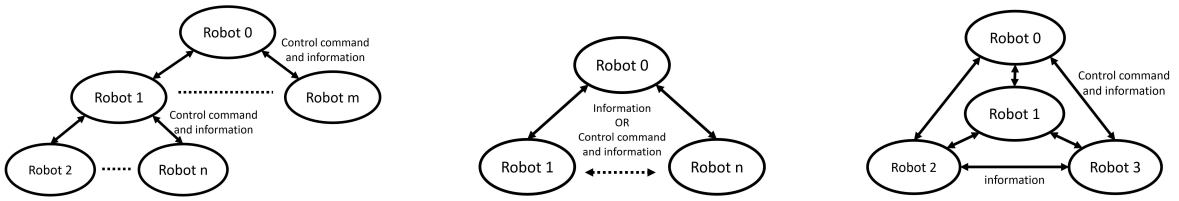
Cooperative control, formation control, mobile robots.

1 Introduction

The integration of robots in real-world applications is becoming more and more feasible. In fact, a multi-robot system (MRS) could perform complex tasks with more efficiency and robustness than one single robot [1]. MRS can be found in a wide range of applications in both the civilian and military sectors including tasks of patrolling [2], exploration and mapping of unknown environments [3], the transportation of huge and/or heavy objects [4].
The aim of this research is to develop an industrial-applicable approach for the MRS such that the fleet of robots can be modular with an efficient task allocating strategy using appropriate control architecture and strategy. The fleet, also, has to be able to do any deformation-free formation in order to carry huge loads.
The main objective of this paper is to review the current state of the art approaches in order to see their limitations facing the industrial needs.
The rest of the paper is organized as follows. The recent related works are discussed in Section 2, whereas Section 3 and Section 4 respectively present the details about the future contribution of this research and the conclusion.

2 Related works

For the past decades, many control architectures, which affects the system's robustness and scalability, and control strategies, which oversees the decisions and behaviors of the robots, have been developed. The control architecture, see figure 1, can be categorized into 3 main types [5] [6] : centralized/hierarchical, decentralized/distributed and hybrid. Hierarchical architecture is a control structure that has one top-leveled robot to control a group of other robots, while each of those others may control lower-leveled groups. Centralized is a special case of hierarchical architecture where the system has only one control level. This kind of architecture provides a faster convergence rate, but less reliability in case of failures of high-leveled control nodes. Decentralized or distributed structure allows each robot to make its own decisions. In case of distributed, information and control commands of each robot may be shared with its neighbors, while only information is shared for the decentralized case. This architecture is proved to be less prone to failures, while requiring higher communications between robots. Hybird architecture is a mixture of hierarchical/centralized and decentralized/distributed in order to obtain advantages from the two architectures.
In term of control strategy, numerous approaches can be seen [7] such as behavior-based, virtual structure, leader-follower. Behavior-based strategy is an example of decentralized control, where each robot has a set of desired behaviors corresponding to different situations, and the final control is obtained by weighting the importance between them. Virtual structure approach treats group of robots as a single entity. Each robot's motion is derived from a trajectory defined for the entire virtual entity. In leader-follower strategy, one or more robot are assigned to be leaders, while the remaining robots are set to be followers.
Even though there exists numerous studies on the MRS, many of which focus only on simulations. In this paper, we consider only works that present actual experiments. In [4], the authors developed a cooperative load-carrying system using two mobile robots. The system relies on a host PC to generate sub-goals for each robot from the set target position and orientation. Thus, it may suffer the issue of scalability if the number of robots in the group increases.



(a) Hierarchical/centralized (b) Decentralized/distributed (c) Hybrid

FIGURE 1 – Different control architectures

Moreover, obstacle avoidance is not included in the study. In [8], the authors presented a practical leader-follower control approach with obstacle avoidance using two robots : a follower and a leader. The obstacle avoidance algorithm was proposed by dividing the robot's surrounding into four different regions in order to set the motion accordingly. The problem of leader-loss was also addressed by just simply drive the follower to the last known position and orientation of the leader. However, the study was based on only two robots where inter-robots collision is not an issue, which is not the case for real application. In [9], the authors showed that it is possible to perform the circle formation using behavioural structure with a finite state machine. Each robot searches for another robot in circular motion, and closely follows that robot if it could find one. Despite the promising experimental result, the study didn't investigate into the collision avoidance as well as the safety distance between robots due to the fact that the following robots move closely to the others. Other than those well-known strategies, there exists also the consensus control approach [10], which is highly in the interest of decentralized formation control for this research. In summary, the current state of the art methodology has not completely answered to the needs for industrial implementation. Especially, considering that most of existing experiments were done in controlled environments, it requires more investigations and studies in order to ensure a fail-safe system for real-world applications.

3 Future contribution

This research will propose a robust consensus control approach taking into account practical industrial implementation requirements such as dynamic obstacles, limited sensors and communication capabilities. One of this work's main objectives is to provide a definitive solution to the challenge of transporting heavy load using multiple robots in manufacturing industries. The research will also focus on intelligent task allocation system where several load-carrying tasks have to be done in the same environment. It is best interest of this study to combine high level coordination, such as hierarchical task network approach [11], with low level decentralized control, such as consensus strategy.

4 Conclusion

This paper presents a review of related works of the MRS, and the objective of this research. We believe that this research will provide solutions to the current issues and limitations, which could potentially extend the scope of implementation from laboratories to the real world industries, where efficiency, repeatability as well as safety are needed.

Références

[1] A. Gautam and S. Mohan, A Review of Research in Multi-Robot Systems, *IEEE Int. Conf. on Industrial and Information System*, Chennai, India, Aug., 2012

[2] M. Popescu, H. Rivano, and O. Simonin, Multi-cycle Coverage for Multi-robot Patrolling - application to data collection in WSNs -, *Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes*, Caen, France, Jul., 2017

[3] A. Birk and S. Carpin, Merging Occupancy Grid Maps From Multiple Robots, *in Proc. of the IEEE*, Vol. 94, pp. 1384-1397, Jul., 2006

[4] M. Morishita, S. Maeyama, Y. Nogami, and K. Watanabe, Development of an Omnidirectional Cooperative Transportation System Using Two Mobile Robots with Two Independently Driven Wheels, *IEEE Inter. Conf. on Systems, Man, and Cybernetics (SMC)*, Oct., 2018

[5] I. Jawhar, N. Mohamed, J. Wu, J. Al-Jaroodi, Networking of Multi-Robot Systems : Architectures and Requirements, *Journal of Sensor and Actuator Networks*, Vol. 7, pp. 52, Nov, 2018

[6] C. Hu, C. Hu, D. He, and Q. Gu, A new ROS-based hybrid architecture for heterogeneous multi-robot systems, *Chinese Control and Decision Conf.*, May, 2015

[7] G. jinrong and Q. Dianwei, A Nonlinear Feedforward-feedback Controller Design for Formation Control of Multi-robot Systems, *SICE Annual Conf.*, Jul., 2015

[8] Y. Wang, D. Wang, S. Yang, and M. Shan, A Practical Leader-Follower Tracking Control Scheme for Multiple Nonholonomic Mobile Robots in Unknown Obstacle Environments, *IEEE Trans. on Control Systems Technology*, pp. 1-9, Apr., 2018

[9] D. St-Onge, C. Pinciroli, and G. Beltrame, Circle Formation with Computation-Free Robots shows Emergent Behavioural Structure, *IROS*, Oct., 2018

[10] Z. Wang, L. Wang, H. Zhang, and Q. Chen, A Graph Based Formation Control of Nonholonomic Wheeled Robots using a Novel Edge-Weight Function, *SMC*, Banff, Canada, Oct., 2017

[11] S. Zeng, Y. Zhu, and C. Qi, HTN-based multi-robot path planning, *Chinese Control and Decision Conf.*, Yinchuan, China, May, 2018

Processus décisionnels de Markov non-stationnaires une approche pire-cas utilisant l'apprentissage par renforcement basé modèle

Erwan Lecarpentier[1,2] Emmanuel Rachelson[2]

[1] Université de Toulouse, ONERA, The French Aerospace Lab, France
[2] Université de Toulouse, ISAE-SUPAERO, France

erwan.lecarpentier@isae-supaero.fr

Abstract

This work tackles the problem of robust zero-shot planning in non-stationary stochastic environments. We study Markov Decision Processes (MDPs) evolving over time and consider Model-Based Reinforcement Learning algorithms in this setting. We make two hypotheses : 1) the environment evolves continuously and its evolution rate is bounded, 2) a current model is known at each decision epoch but not its evolution. Our contribution can be presented in four points. First, we define this specific class of MDPs that we call Non-Stationary MDPs (NSMDPs). We introduce the notion of regular evolution by making an hypothesis of Lipschitz-Continuity on the transition and reward functions w.r.t. time. Secondly, we consider a planning agent using the current model of the environment but unaware of its future evolution. This leads us to consider a worst-case method where the environment is seen as an adversarial agent. Thirdly, following this approach, we propose the Risk-Averse Tree-Search (RATS) algorithm. This is a zero-shot Model-Based method similar to Minimax search. Finally, we illustrate the benefits brought by RATS empirically and compare its performance with reference Model-Based algorithms.

Keywords

Reinforcement Learning, Model-Based Reinforcement Learning, Markov Decision Processes, Non-Stationary Markov Decision Processes

1 Introduction

One of the hot topics of modern Artificial Intelligence (AI) is the ability for an agent to adapt its behaviour to changing tasks. In the literature, this problem is often linked to the setting of Lifelong Machine Learning (LML) [Silver et al., 2013, Abel et al., 2018a,b] and the one of learning in non-stationary environments [Dit-Yan et al., 1999, Jaulmes et al., 2005, Hadoux, 2015]. In LML, the tasks presented to the agent change sequentially at discrete transition epochs [Silver et al., 2013]. Similarly, the non-stationary environments considered in the literature often evolve abruptly at discrete transition epochs [Dit-Yan et al., 1999, Hadoux, 2015, Hadoux et al., 2014, Doya et al., 2002, Da Silva et al., 2006, Choi et al., 2000, 2001, Campo et al., 1991, Wiering, 2001]. In this paper, we investigate environments continuously changing over time that we call Non-Stationary Markov Decision Processes (NSMDPs). In this setting, it is realistic to bound the evolution rate of the environment using a Lipschitz Continuity (LC) assumption. Allowing for arbitrarily large changes would indeed lead to a chaotic evolution that is intractable.

Model-based Reinforcement Learning approaches [Sutton et al., 1998] benefit from the knowledge of a model allowing them to reach impressive performances, such as the famous example of Monte Carlo Tree Search (MCTS) in the game of Go [Silver et al., 2016]. In this matter, the necessity to have access to a model is a great concern of AI [Asadi et al., 2018, Osadchy et al., 2007, Jaulmes et al., 2005, Doya et al., 2002, Da Silva et al., 2006]. In the context of NSMDPs, we assume that an agent is provided with a *snapshot* model when its action is computed. By this, we mean that it only has access to the current model of the environment but not its future evolution, as if it took a photograph but would be unable to predict how it is going to evolve in the future. This hypothesis is realistic, because many environments have a tractable state while their future evolution is hard to predict [Da Silva et al., 2006, Wiering, 2001]. In order to solve LC-NSMDPs, we propose a method that considers the worst-case possible evolution of the model and performs planning w.r.t. this model. We call it the *worst-case* approach. This is equivalent to considering Nature as an adversarial agent.

The paper is organized as follows : first we describe the NSMDP setting and the regularity assumption (Section 2) ; then we outline related works to this setting (Section 3) ; follows the explanation of the worst-case approach proposed in this paper (Section 4) ; then we describe an algorithm reflecting this approach (Section 5) ; finally we illustrate its behaviour empirically (Section 6).

2 Non-Stationary Markov Decision Processes

NSMDP. To define a Non-Stationary Markov Decision Process (NSMDP), we go back to the initial MDP model

introduced by Puterman [2014] (chapter 2), where the transition and reward functions depend on time. An NSMDP is formally defined as follows.

Definition 1. ***NSMDP.*** *An NSMDP is an MDP whose transition and reward functions depend on the decision epoch. It is defined by a 5-tuple* $\{\mathcal{S}, \mathcal{T}, \mathcal{A}, p, r\}$ *where* $\mathcal{S}$ *is a state space;* $\mathcal{T} \equiv \{1, 2, \cdots, N\}$ *is the set of decision epochs with* $N \leq +\infty$*;* $\mathcal{A}$ *is an action space;* $p_t(s' \mid s, a)$ *is the probability to reach state* s' *while undertaking action* a *at decision epoch* t *in state* s*;* $r_t(s, a, s')$ *is the scalar reward associated to the transition from* s *to* s' *with action* a *at decision epoch* t*.*

This definition can be viewed as that of a stationary MDP whose state space has been enhanced with time. While this addition can be trivial in episodic tasks where an agent is given the opportunity to interact several times with the same MDP, it is different when the experience is unique. In non-episodic tasks, everything is meant to change in a way never encountered before. Conversely to the case where an agent can freely explore the state space in order to learn, no exploration is allowed along the temporal axis, since, in non-episodic tasks, one cannot go back in time. The value of N in this definition distinguishes the finite horizon case when N is finite from the infinite case when $N = +\infty$. Within a stationary MDP, it is proven that there exist a Markovian deterministic stationary policy [Puterman, 2014]. It is not the case within NSMDPs where the optimal policy is non-stationary in the most general case. Additionally, we define the expected reward received when taking action a at state s and decision epoch t with $R_t(s, a) = \mathbb{E}_{s' \sim p_t(\cdot|s,a)} \{r_t(s, a, s')\}$. Without loss of generality, we will assume the reward function to be bounded between -1 and 1, i.e. $r_t(s, a, s') \in [-1, 1], \forall s, a, s', t \in \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{T}$. Definition 1 echoes back to the one of Semi Markov Decision Processes (SMDPs) [Howard, 1963, Sutton et al., 1999] where a transition's duration is a continuous random variable. In this paper, we consider discrete time decision processes with constant deterministic duration $\Delta t = 1$ for each transition, which allows us to consider deterministic decision epochs in Definition 1 (this straightforwardly generalizes to other fixed durations than one). This assumption is mild since many discrete time sequential decision problems follow that assumption. Furthermore, all the results in this paper can be extended to the non-constant deterministic action duration case.
A non-stationary policy π is a sequence of *decision rules* π_t which map states to actions (or distributions over actions). For a stochastic non-stationary policy $\pi_t(a \mid s)$, the value of a state s at decision epoch t within an infinite horizon NSMDP is defined by :

$$V_t^\pi(s) = \mathbb{E}\left[\sum_{i=t}^{\infty} \gamma^{i-t} R_i(s_i, a_i) \,\middle|\, \begin{array}{l} s_t = s \\ s_{i+1} \sim p_i(\cdot \mid s_i, a_i) \\ a_i \sim \pi_i(\cdot \mid s_i) \end{array}\right]$$

Where $\gamma \in [0, 1)$ is a discount factor. The definition of the state-action value function Q_t^π for policy π at decision epoch t is straightforward and we have the following equation :

$$Q_t^\pi(s, a) = R_t(s, a) + \gamma \mathop{\mathbb{E}}_{s' \sim p_t(\cdot|s,a)} \left[V_{t+1}^\pi(s')\right] \quad (1)$$

Overall, we defined an NSMDP as an MDP where we stress out the distinction between state, time, and decision epoch due to the inability for an agent to explore the temporal axis at will. This distinction is particularly relevant for non-episodic tasks, i.e. when there is no possibility to re-experience the same MDP starting from a prior date. Otherwise, this definition amounts to the one of an MDP with time as component of the state.

The regularity hypothesis. Many real-world problems can be modelled as an NSMDP, such as, for instance, the problem of path planning for a glider immersed in a non-stationary atmosphere Âă[Chung et al., 2015, Lecarpentier et al., 2017], or that of vehicle routing in dynamic traffic congestion. Realistically, we consider that the expected reward and transition functions do not evolve arbitrarily fast over time. Conversely, if such an assumption was not made, a chaotic evolution of the NSMDP would be allowed which is both unrealistic and intractable. As stated in Dit-Yan et al. [1999], RL in non-stationary environments is impossible if no assumption is made on the way the environment changes. Hence, we assume that changes occur slowly over time. Mathematically, we formalize this hypothesis by bounding the evolution rate of the transition and expected reward functions, using the notion of Lipschitz Continuity (LC).

Definition 2. ***Lipschitz Continuity.*** *Let* (X, d_X) *and* (Y, d_Y) *be two metric spaces and* $f : X \to Y$*,* f *is L-Lipschitz Continuous (L-LC) with* $L \in \mathbb{R}^+$ *iff*

$$d_Y(f(x), f(\hat{x})) \leq L\, d_X(x, \hat{x}), \forall (x, \hat{x}) \in X^2.$$

L is called a Lipschitz constant of the function f.

We then apply this hypothesis to the transition and reward functions of an NSMDP so that those functions are LC w.r.t. time. For the transition function, this leads to the consideration of a metric between two probability density functions. For that purpose, we will use the 1-Wasserstein distance [Villani, 2008].

Definition 3. ***1-Wasserstein distance.*** *Let (X, d_X) be a Polish metric space and μ, ν any two probability measures on X. The 1-Wasserstein distance between μ and ν is*

$$W_1(\mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{X \times X} d_X(x, y) d\pi(x, y) \quad (2)$$

where $\Pi(\mu, \nu)$ is the set of all joint distributions on $X \times X$ with marginals μ and ν.

The choice of the Wasserstein distance is motivated by the fact that it quantifies the distance between two distributions in a physical and interpretable manner, respectful of the topology of the measured space [Dabney et al., 2017, Asadi et al., 2018]. We illustrate this with two facts. First, it is sensitive to the difference between the supports of the distributions. As a comparison, the Kullback-Leibler divergence does not respect this property since the divergence between distributions with disjoint supports is always infinite. Secondly, it is sensitive to the underlying metric. Indeed, if we consider two regions of the support where two distributions differ, the Wasserstein distance is sensitive to the distance between the elements of those two regions. As a comparison, the total-variation metric does not respect this property since it is the same regardless of this distance. We now introduce the notion of LC-NSMDP.

Definition 4. ***(L_p, L_r)-LC-NSMDP.*** *An (L_p, L_r)-LC-NSMDP is an NSMDP whose transition and reward functions are respectively L_p-LC and L_r-LC w.r.t. time i.e. $\forall (t, \hat{t}) \in \mathcal{T}^2, \forall (s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$,*

$$W_1(p_t(\cdot \mid s, a), p_{\hat{t}}(\cdot \mid s, a)) \leq L_p |t - \hat{t}|, \quad (3)$$

$$|r_t(s, a, s') - r_{\hat{t}}(s, a, s')| \leq L_r |t - \hat{t}|. \quad (4)$$

One should remark that the LC property should be defined with respect to actual decision times and not decision epoch indexes for the sake of realism. In the present case, both have the same value, and we choose to keep this convention for clarity. Our results however extend easily to the case where indexes and times do not coincide. From now on, we will be consider (L_p, L_r)-LC-NSMDPs, making Lipschitz Continuity our regularity property. Notice that R is defined as a convex combination of r by the probability measure p. As a result, the notion of Lipschitz Continuity of R is strongly related to the one of r and p. The following result holds (proof in the Appendix).

Property 1. *Given an (L_p, L_r)-LC-NSMDP, the expected reward function $R_t : s, a \mapsto \mathbb{E}_{s' \sim p_t(\cdot|s,a)} \{r_t(s, a, s')\}$ is L_R-LC with $L_R = L_r + L_p$.*

This result shows the conditioning of the evolution rate of R by the evolution rates of r and p. It allows us to work either with the reward function r or its expectation R, benefiting from the same LC property under the same hypothesis.

3 Related work

Non-stationary MDPs have been extensively studied. A very common framework is probably the one of HM-MDPs (Hidden Mode MDPs) introduced in [Dit-Yan et al., 1999]. This is a special class of POMDP (Partially Observable MDP [Kaelbling et al., 1998]) where a hidden mode indexes a latent stationary MDP within which the agent evolves. This way, similarly to the context of LML, the agent experiences a series of different MDPs over time. In this setting, [Dit-Yan et al., 1999, Choi et al., 2000] proposed methods to learn the different models of the latent stationary MDPs.[Doya et al., 2002] built a modular architecture switching between models and policies when a change is detected. Similarly, [Wiering, 2001, Da Silva et al., 2006, Hadoux et al., 2014] proposed a method tracking the switching occurrence and re-planning if needed. Overall, as in LML, the HM-MDP setting is different from ours in the sense that evolutions of the transition and reward functions are abrupt and not continuous.

Other settings have been considered, as in [Jaulmes et al., 2005] where they do not make particular hypothesis on the evolution of the NSMDP. Their algorithm is designed to the more general POMDP case where the uncertainty in the model is tackled by sampling models from a distribution and acting accordingly. The time dependency is taken into account by weighting recently experienced transitions more than older ones. Another approach is taken by [Lane et al., 2007] where they make use of relational representations in order to generalize through time which works well in cases with a strong topology but otherwise less for other cases.

To plan efficiently within an NSMDP, our approach consists in taking advantage of the *slow* evolution of the environment in order to plan according to the worst-case. This results practically in considering LC-NSMDPs. Generally speaking, taking advantage of Lipschitz continuity to infer bounds on the value of a function within a certain neighbourhood is a widely used tool in the RL, bandit and optimization communities [Kleinberg et al., 2008, Rachelson and Lagoudakis, 2010, Pirotta et al., 2015, Pazis and Parr, 2013, Munos et al., 2014]. We implement this approach with a Minimax algorithm [Fudenberg and Tirole, 1991] where the environment is seen as an adversarial agent. This is to be linked with the MCTS framework [Browne et al., 2012] that can be applied to single agent environments as well as adversarial games. MCTS-Minimax hybrids algorithms are able to perform shallow MCTS search for quick convergence to the optimal actions and take advantage from Minimax search to increase the robustness of the search [Lanctot et al., 2014, Baier and Winands, 2015, 2018]. To the best of our knowledge, this class of algorithms has never been applied to cases where the environment is seen as an adversarial agent in order to provide worst-case guarantees.

4 worst-case approach

Snapshot MDP hypothesis. We consider finding an optimal policy within an NSMDP under the non-episodic task hypothesis. The latter prevents us from learning from

previous experienced data since they become outdated with time and no information samples have been collected yet for future time steps. An alternative is to use model-based RL algorithms such as MCTS. For a current state s_0, such algorithms focus on finding the optimal action a_0^* by using a generative model. This action is then undertaken and the operation repeated at the next state. However, using the true NSMDP model for this purpose is an unrealistic hypothesis, since this model is generally unknown. This would amount to knowing the exact future evolution of the environment. We make the hypothesis that an agent does not have access to the true NSMDP model ; instead, we introduce the notion of *snapshot model* of an NSMDP. Intuitively, the snapshot associated to time t_0 is a temporal slice of the NSMDP at t_0. This means that we consider its transition and reward functions, though we *freeze* them so that they are evaluated at t_0. One can see this as taking a photograph and performing planning within the resulting stationary MDP. A snapshot MDP is formally defined as follows.

Definition 5. ***Snapshot of an NSMDP.*** *The snapshot of an NSMDP $\{\mathcal{S}, \mathcal{T}, \mathcal{A}, p, r\}$ at decision epoch t_0 denoted by MDP$_{t_0}$ is the stationary MDP defined by the 4-tuple $\{\mathcal{S}, \mathcal{A}, p_{t_0}, r_{t_0}\}$ where $\mathcal{S}, \mathcal{A}$ are shared with the NSMDP ; $p_{t_0}(s' \mid s, a)$ and $r_{t_0}(s, a, s')$ are the transition and reward functions of the NSMDP at decision epoch t_0.*

Similarly to the NSMDP, this definition induces the existence of the snapshot expected reward R_{t_0} defined by $R_{t_0} : s, a \mapsto \mathbb{E}_{s' \sim p_{t_0}(\cdot|s,a)} \{r_{t_0}(s, a, s')\}$. For simplicity, we will consider the same spaces $\mathcal{S}, \mathcal{A}$, and γ in our study. Notice that the snapshot MDP$_{t_0}$ is stationary and coincides with the NSMDP *only* at t_0. Particularly, one can generate a trajectory $\{s_0, r_0, \cdots, s_k\}$ within an NSMDP using the sequence of snapshots $\{\text{MDP}_{t_0}, \cdots, \text{MDP}_{t_0+k-1}\}$ as a model. Overall, the hypothesis of using snapshot models amounts to considering a planning agent only able to get the current stationary model of the environment. In real-world problems, predictions often are uncertain or hard to perform e.g. in the thermal soaring problem of a glider.

Planning with the snapshot. We consider a generic *planning* agent at s_0, t_0 using MDP$_{t_0}$ as a model of the NSMDP. By planning, we mean conducting a look-ahead search within the possible trajectories starting from the current state and time given a model of the environment. The search allows in turn to identify an optimal action w.r.t. the model. This action is then undertaken and the agent jumps to the next state where the operation is repeated. Such a method received much attention in the past decade due to its impressive results [Silver et al., 2016]. Particularly, the MCTS algorithm, both within single agent environments and in two players games [Kocsis and Szepesvári, 2006, Browne et al., 2012].

Consider an agent planning its next action in s_0, at t_0. This agent only has access to the MDP$_{t_0}$ snapshot which, in the general case, does not match the true NSMDP model. The state transitions and rewards from any state s are computed with p_{t_0} and R_{t_0}, even though in reality s might be reached at decision epoch $t > t_0$. The estimated value of the computed plan in s at t is thus the value of the optimal policy of MDP$_{t_0}$, which we write $V^*_{\text{MDP}_{t_0}}(s)$. The true optimal value of s at t within the NSMDP does not match this estimate because the values computed with MDP$_{t_0}$ cannot take the non-stationarity into account. Applying the optimal policy of MDP$_{t_0}$ within the NSMDP has, thus, no reason to be optimal. Indeed, unforeseen scenarios may lead to poor performances, and even worse, to catastrophic terminal states. Can we do better without further hypothesis ? In the next Section, we investigate a way to robustly insure a good performance and avoid catastrophic terminal states.

Worst-case approach. The intuition developed in this paper is that, given the *slow* evolution rate of the environment, for a state s seen at a future decision epoch during the search, we can predict a scope into which the transition and reward functions at s lie. Formally, the Lipschitz Continuity property allows us to define such a domain and we have the following property for the admissible models at s.

Property 2. ***Set of admissible snapshot models.*** *Consider an (L_p, L_r)-LC-NSMDP and a snapshot MDP$_{t_0}$ with $t_0 \in \mathcal{T}$. For any triplet $s, t, a \in \mathcal{S} \times \mathcal{T} \times \mathcal{A}$, the transition and expected reward functions (p_t, R_t) of the snapshot MDP$_t$ belong to the set $\Delta^t_{t_0}$ with the following definition :*

$$\Delta^t_{t_0} \stackrel{\text{def}}{=} \mathcal{B}_{W_1}\left(p_{t_0}(\cdot \mid s, a), L_p|t - t_0|\right) \times \mathcal{B}_{|\cdot|}\left(R_{t_0}(s, a), L_R|t - t_0|\right)$$

where $L_R = L_p + L_r$ and $\mathcal{B}_d(c, r)$ denotes the ball of center c, defined with metric d and radius r.

Démonstration. The proof is straightforward using the Lipschitz property of definition 4 and the result of property 1. □

For a future prediction at s, t, we consider the question of using a better model than p_{t_0}, R_{t_0}. The underlying evolution of the NSMDP being unknown, a desirable feature would be to use a model leading to a policy that is *robust* to every possible evolution. To that end, we propose to use the snapshots corresponding to the worst possible evolution scenario under the constraints of Property 2. We claim that such a practice is an efficient way to 1) insure robust performance to all possible evolutions of the NSMDP and 2) avoid catastrophic terminal states. Practically, this boils down to using a different value estimate for s at t than $V^*_{\text{MDP}_{t_0}}(s)$ that, as seen before, gives no performance guarantee within the true NSMDP.

Given a policy $\pi = (\pi_t)_{t \in \mathcal{T}}$ and a decision epoch t_0, a worst-case NSMDP corresponds to a sequence of transition and reward models that minimize the expected value of applying π in any pair (s, t), while remaining within the bounds of Property 2. We write $\overline{V}^{\pi}_{t_0,t}(s)$ this value for state s at decision epoch t.

$$\overline{V}^{\pi}_{t_0,t}(s) \stackrel{\text{def}}{=} \min_{\substack{(p_i,R_i)\in\Delta^i_{t_0}\\ \forall i\in\mathcal{T}}} \mathbb{E}\left[\sum_{i=t}^{N}\gamma^{i-t}R_i(s_i,a_i)\right] \quad (5)$$

$$\text{with } s_t = s,\ s_{i+1}\sim p_i(\cdot\mid s_i,a_i),\ a_i\sim\pi_i(\cdot\mid s_i)$$

Intuitively, the worst-case NSMDP is a model of a non-stationary environment leading to the poorest possible performance for π, while being an admissible evolution of MDP_{t_0}. It reflects a worst-case scenario and magnifies the possible pitfalls of the environment e.g. π leading to catastrophic terminal states. Let us define $\overline{Q}^{\pi}_{t_0,t}(s,a)$ as the worst-case Q value for the pair (s,a) at decision epoch t, given the snapshot at t_0 :

$$\overline{Q}^{\pi}_{t_0,t}(s,a) \stackrel{\text{def}}{=} \min_{(p,R)\in\Delta^t_{t_0}} \mathop{\mathbb{E}}_{s'\sim p}\left[R(s,a)+\gamma\overline{V}^{\pi}_{t_0,t+1}(s')\right] \quad (6)$$

A pair $(\overline{p}_t,\overline{R}_t)$ that participates in the minimization of Equation 6 defines a worst-case snapshot at decision epoch t. Such a snapshot is recursively defined as, for all (s,a) :

$$(\overline{p}_t,\overline{R}_t) \stackrel{\text{def}}{\in} \mathop{\arg\min}_{(p,R)\in\Delta^t_{t_0}} \mathop{\mathbb{E}}_{s'\sim p}\left[R(s,a)+\gamma\overline{V}^{\pi}_{t_0,t+1}(s')\right] \quad (7)$$

Identifying the worst-case snapshots would allow the planning agent to derive a cautious, minimax behaviour that provides a worst-case performance guarantee given only MDP_{t_0}. The best worst-case performance over all possible policies is defined as $\overline{V}^{*}_{t_0,t}(s)$ (Equation 8).

$$\overline{V}^{*}_{t_0,t}(s) = \max_{\pi}\min_{\substack{(p_i,R_i)\in\Delta^i_{t_0}\\ \forall i\in\mathcal{T}}} \mathbb{E}\left[\sum_{i=t}^{N}\gamma^{i-t}R_i(s_i,a_i)\right] \quad (8)$$

We define $\overline{Q}^{*}_{t_0,t}(s,a)$ in Equation 9.

$$\overline{Q}^{*}_{t_0,t}(s,a) = \min_{(p,R)\in\Delta^t_{t_0}} \mathop{\mathbb{E}}_{s'\sim p}\left[R(s,a)+\gamma\overline{V}^{*}_{t_0,t+1}(s')\right] \quad (9)$$

Recursively, this yields Equation 10 :

$$\overline{V}^{*}_{t_0,t}(s) = \max_{a\in\mathcal{A}}\overline{Q}^{*}_{t_0,t}(s,a) \quad (10)$$

In the next Section, we design a planning algorithm inspired by minimax search that computes at the same time an approximation of a worst-case NSMDP and a worst-case optimal first action.

5 Risk-Averse Tree-Search algorithm

The algorithm. Tree search algorithms within MDPs have been well studied and cover two classes of search trees, namely closed loop [Keller and Helmert, 2013, Kocsis and Szepesvári, 2006, Browne et al., 2012] and open loop [Bubeck and Munos, 2010, Lecarpentier et al., 2018]. Following [Keller and Helmert, 2013, Kocsis and Szepesvári, 2006], we consider closed loop search trees, composed of *decision nodes* alternating with *chance nodes*. We adapt their formulation to take time into account, resulting in the following definitions. A decision node at depth t, denoted by $\nu^{s,t}$, is labeled by a unique state / decision epoch pair (s,t). The edges leading to its children chance nodes correspond to the available actions at (s,t). A chance node, denoted by $\nu^{s,t,a}$, is labeled by a state / decision epoch / action triplet (s,t,a). The edges leading to its children decision nodes correspond to the reachable state / decision epoch pairs (s',t') after performing a in (s,t) as illustrated by Figure 1. We consider the problem of estimating the optimal action a^*_0 at s_0,t_0 within a worst-case NSMDP. Estimating this action boils down to solving Equation 10. This problem is twofold. It requires 1) to estimate the worst-case NSMDP given MDP_{t_0} and 2) to run explore the latter in order to identify a^*_0. We propose to tackle both problems with an algorithm inspired by the minimax algorithm [Fudenberg and Tirole, 1991] where the *max* operator corresponds to the policy of the agent, seeking to maximize the return ; and the *min* operator corresponds to the worst-case model, seeking to minimize the return.

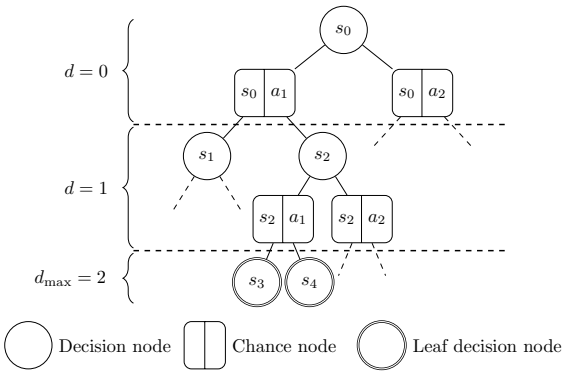


FIGURE 1 – Tree structure illustration for a maximum depth of $d_{\max}=2$ with two actions $\mathcal{A}=\{a_1,a_2\}$. Each depth, shown on the left side, corresponds to the decision epoch.

Max nodes. A decision node $\nu^{s,t}$ corresponds to a max node due to the greediness of the agent w.r.t. the subsequent values of the children. RATS aims at maximizing the return while having a risk-averse behaviour. As a result, the value of $\nu^{s,t}$ follows Equation 10 and is defined as :

$$V(\nu^{s,t}) = \max_{a\in\mathcal{A}} V(\nu^{s,t,a}) \quad (11)$$

Min nodes. A chance node $\nu^{s,t,a}$ corresponds to a min node due to the use of a worst-case NSMDP as a model which minimizes the value of $\nu^{s,t,a}$ w.r.t. the reward and the subsequent values of its children. Writing the value of $\nu^{s,t,a}$ as the value of s,t,a within the worst-case snapshot of Equation 7, using the children's values as values for the next reachable states, leads to Equation 12.

$$V\left(\nu^{s,t,a}\right) = \min_{(p,R)\in\Delta^t_{t_0}} R(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim p} V\left(\nu^{s',t+1}\right) \quad (12)$$

Algorithm 1 RATS algorithm

RATS. Input : s_0, t_0, maxDepth
ν_0 = rootNode(s_0, t_0)
Minimax(ν_0)
$\nu^* = \arg\max_{\nu' \textbf{ in } \nu.\text{children}} \nu'.\text{value}$
return ν^*.action

Minimax. Input : ν, maxDepth
if ν is DecisionNode **then**
  if ν.state is terminal **or** ν.depth = maxDepth **then**
    ν.value = heuristicValue(ν.state)
    return ν.value
  else
    ν.value = $\max_{\nu' \in \nu.\text{children}}$Minimax($\nu'$, maxDepth)
    return ν.value
  end if
else
  $\nu.\text{value} = \min_{(p,R) \in \Delta^t_{t_0}} R(\nu) +$
  $\gamma \sum_{\nu' \in \nu.\text{children}} p(\nu' \mid \nu)$Minimax($\nu'$, maxDepth)
  return ν.value
end if

Given that the values of the subsequent children estimate the optimal value $\overline{V}^*_{t_0,t+1}(s')$, this equations reflects the optimal Q value of Equation 9.

This approach considers the environment as an adversarial agent, as in a two-players game, in order to search for a robust plan. Note that this is an *asymmetric* two-players game, since both agents have different action sets : the actions the environment can take are modifications to the model itself. The resulting algorithm, called RATS for Risk-Averse Tree-Search, is described in Algorithm 1. Given an initial state / decision epoch pair, a minimax tree is built using the snapshot MDP_{t_0} and the operators corresponding to equations 11 and 12 in order to estimate the worst-case snapshots at each depth. Once the tree is built, the action leading to the best possible value from the root node is selected and a transition in the real-world is performed. The next state is then reached, the new snapshot model MDP_{t_0+1} is acquired and the process re-starts from the beginning. Notice the use of $R(\nu)$ and $p(\nu' \mid \nu)$ in the pseudo-code : they are light notations respectively standing for the expected reward of the (s, t, a) triplet corresponding to a chance node $\nu \equiv \nu^{s,t,a}$ and the probability to jump to the $(s', t+1)$ pair of a decision node $\nu' \equiv \nu^{s',t+1}$ given the (s, t, a) triplet of a chance node $\nu \equiv \nu^{s,t,a}$. The tree built by the algorithm is entirely developed until the maximum depth $d_{\max}$. Within the Minimax algorithm, a heuristic function is used to evaluate the leaf nodes of the tree. We now describe this heuristic and the way to compute the min operator.

Analysis of RATS. In this paper, we are only interested by the characterization of the RATS algorithm itself and therefore will focus on the exact case, i.e. without using value function approximation. Consequently and for the sake of clarity we only consider finite state-action spaces $\mathcal{S} \times \mathcal{A}$.

In this Section, we detail the implementation of RATS in this case, particularly the definition of the min operator and the heuristic function.

Min operator. The min operator of Algorithm 1 is described by Equation 12. In the exact case, the following property holds (proof in the appendix) :

Property 3. ***Closed-form expression of the worst case snapshot of a chance node.*** *Following Algorithm 1, a solution to Equation 12 is given by :*

$$\bar{R}(s,a) = R(s,a) - L_R|t - t_0|$$
$$\bar{p}(\cdot \mid s,a) = (1-\lambda)p_{t_0}(\cdot \mid s,a) + \lambda p_{sat}(\cdot \mid s,a)$$

with $p_{sat}(\cdot \mid s,a) = (0, \cdots, 0, 1, 0, \cdots, 0)$ *with* 1 *at position* $\arg\min_{s'} V(\nu^{s',t+1})$, $\lambda = 1$ *if* $W_1(p_{sat}, p_0) \leq C$ *and* $\lambda = C\, W_1(p_{sat}, p_0)$ *otherwise.*

Heuristic function. As in vanilla minimax algorithms [Fudenberg and Tirole, 1991], Algorithm 1 bootstraps the values of the leaf nodes with a heuristic function if these leaves do not correspond to terminal states. Given such a leaf node $\nu^{s,t}$, this heuristic aims at estimating the value of the optimal policy at (s, t) within the worst-case NSMDP i.e. $\overline{V}^*_{t_0,t}(s)$. Let $H : s, t \mapsto H(s,t)$ be such a heuristic function, we call *heuristic error* in (s, t) the difference between $H(s,t)$ and $\overline{V}^*_{t_0,t}(s)$. Assuming that the heuristic error is uniformly bounded, the following property provides an upper bound on the propagated error due to the choice of H (proof in the appendix).

Property 4. ***Upper bound on the propagated heuristic error within RATS.*** *Consider an agent executing Algorithm 1 at* s_0, t_0 *with a heuristic function* H*. We note* $\mathcal{L}$ *the set of all leaf nodes. Suppose that the heuristic error is uniformly bounded i.e.* $\exists \delta > 0, \forall \nu^{s,t} \in \mathcal{L}, |H(s) - \overline{V}^*_{t_0,t}(s)| \leq \delta$*. Then we have for every decision and chance nodes* $\nu^{s,t}$ *and* $\nu^{s,t,a}$ *at any depth* $d \in [0, d_{\max}]$ *:*

$$|V(\nu^{s,t}) - \overline{V}^*_{t_0,t}(s)| \leq \gamma^{(d_{\max}-d)}\delta$$
$$|V(\nu^{s,t,a}) - \overline{Q}^*_{t_0,t}(s,a)| \leq \gamma^{(d_{\max}-d)}\delta$$

In particular, for the root node's children we have $|V(\nu^{s_0,t_0,a}) - \overline{Q}^*_{t_0,t}(s_0,a)| \leq \gamma^{d_{\max}}\delta, \forall a \in \mathcal{A}$*.*

This last result implies that with *any* heuristic function H inducing a uniform heuristic error, the propagated error at the root of the tree is guaranteed to be upper bounded by $\gamma^{d_{\max}}\delta$. In particular, since the reward function is bounded by hypothesis, we have $\overline{V}^*_{t_0,t}(s) \leq 1/(1-\gamma)$. Thus, selecting for instance the zero function ensures a root node heuristic error of at most $\gamma^{d_{\max}}/(1-\gamma)$. We note $H_1(s)$ this first choice :

$$H_1(s) = 0$$

In order to improve the precision of the algorithm, we propose to guide the heuristic by using a function reflecting better the value of state s at leaf node $\nu^{s,t}$. The ideal function

would of course be $H(s) = \overline{V}^*_{t_0,t}(s)$, reducing the heuristic error to zero, but this is intractable. Instead, we suggest to use the value of s within the snapshot MDP_t using an evaluation policy π, i.e. $H(s) = V^\pi_{\text{MDP}_t}(s)$. This snapshot is also not available, but Property 5 provides a range wherein this value lies, given the value of $V^\pi_{\text{MDP}_{t_0}}(s)$.

Property 5. ***Bounds on the snapshots values.*** *Consider $s \in \mathcal{S}$, MDP_{t_0} and MDP_t two snapshot MDPs with $t, t_0 \in \mathcal{T}^2$. Let π be a policy. We note $V^\pi_{MDP_{t_0}}(s)$ the value of s within MDP_{t_0}, following π. Similarly we note $V^\pi_{MDP_t}(s)$ its value within MDP_t. The following bound holds :*

$$|V^\pi_{MDP_{t_0}}(s) - V^\pi_{MDP_t}(s)| \leq \frac{L_R}{1-\gamma}|t - t_0|$$

Since MDP_{t_0} is available, $V^\pi_{\text{MDP}_{t_0}}(s)$ can be estimated, e.g. via Monte-Carlo roll-outs. Let $\widehat{V}^\pi_{\text{MDP}_{t_0}}(s)$ denote such an estimate of $V^\pi_{\text{MDP}_{t_0}}(s)$. Then, following Property 5 :

$$V^\pi_{\text{MDP}_{t_0}}(s) - \frac{L_R}{1-\gamma}|t - t_0| \leq V^\pi_{\text{MDP}_t}(s)$$

Hence, a worst-case heuristic on the value of $V^\pi_{\text{MDP}_t}(s)$ is :

$$H_2(s) = \widehat{V}^\pi_{\text{MDP}_{t_0}}(s) - \frac{L_R}{1-\gamma}|t - t_0|$$

The bounds provided by Property 4 decrease quickly with $d_{\max}$, and given that $d_{\max}$ is *large enough*, RATS provides the optimal risk-averse behaviour, i.e.the behaviour that maximizes the worst-case value for any evolution of the NSMDP. However, RATS is computationally intensive since its complexity scales in $\mathcal{O}((|\mathcal{S}||\mathcal{A}|)^{d_{\max}})$. This exponential growth is a major drawback and scaling this approach up to large problem sizes is a challenge for future work in this area.

6 Experiments

We compare the RATS algorithm with two policies. The first one, denoted by DP-snapshot, uses Dynamic Programming (DP) [Bellman, 1957] to compute the optimal actions w.r.t. the provided snapshot models at each decision epoch. The second one, denoted by DP-NSMDP, uses the real NSMDP as a model to provide its optimal action. The latter behaves as an omniscient agent and should be seen as an upper bound on the performance.

We choose a particular grid-world domain called Non-Stationary bridge environment illustrated in Figure 2 [1]. An agent starts at the state labeled S in the center and the goal is to reach one of the two terminal goal positions labeled G where a reward of +1 is received. The grey cells represent holes that are terminal states where a reward of -1 is received. Reaching the goal on the right leads to the highest payoff since it is closest to the starting point S

1. We will provide a link to the code after the reviewing process in order to preserve the double-blind policy.

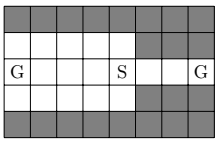


FIGURE 2 – The Non-Stationary bridge environment

and a discount factor $\gamma = 0.9$ is applied. The actions are $\mathcal{A} = \{\text{Up, Right, Down, Left}\}$. The transition function is stochastic and non-stationary. At decision epoch $t = 0$, any action deterministically yields the expected outcome. With time, the probability to reach the positions usually stemming from Up and Down increases symmetrically until reaching 0.45. We set the Lipschitz constant $L_p = 1$. Aside, we introduce a parameter $\epsilon \in [0, 1]$ defining the behaviour of the environment. If $\epsilon = 0$, only the left-hand side bridge becomes slippery with time. It reflects a close to worst-case evolution for a policy aiming to the left-hand side goal. If $\epsilon = 1$, only the right-hand side bridge becomes slippery with time. It reflects a close to worst-case evolution for a policy aiming to the right-hand side goal. In between, the misstep probability is proportionally balanced between left and right. One should note that changing ϵ from 0 to 1 does not allow to cover all the possible evolutions from MDP_{t_0} but will provide a concrete, graphical illustration of RATS's behaviour for various values of ϵ (various possible evolutions of the NSMDP).

We tested RATS with $d_{\max} = 6$ so that leaf nodes in the search tree are actually terminal states. Hence, the optimal risk-averse policy is applied and no heuristic approximation is made. Our goal is to demonstrate that planning in this worst-case NSMDP allows to minimize the loss given any possible evolution of the environment. To illustrate this, we report results reflecting different evolutions of the same NSMDP using the ϵ factor. It should be noted that at $t = 0$, RATS always moves to the left, even if the goal is further, since going to the right may be risky if the probabilities to go Up and Down increase. This corresponds to the careful, risk-averse, behaviour. Conversely, DP-snapshot always moves to the right since MDP_0 does not capture this risk. As a result, the case $\epsilon = 0$ reflects a favourable evolution for DP-snapshot and a bad one for RATS. The opposite is achieved with $\epsilon = 1$ where the cautious behaviour dominates over the risky one, and the in-between cases mitigate this effect.

In Figure 3, we display the achieved expected return for each algorithm as a function of ϵ, i.e. as the function of the possible evolutions of the environment. As expected, the performance of DP-snapshot really depends on this evolution. It achieves high return for $\epsilon = 0$ and low return for $\epsilon = 1$. Conversely, the performance of RATS varies less across the different values of ϵ. The effect illustrated here is that RATS maximizes the minimal possible return given any evolution of the NSMDP. Hence it provides guarantees on the worst-case performance : the guarantee to achieve the best worst-case return. This behaviour is highly desirable when one requires

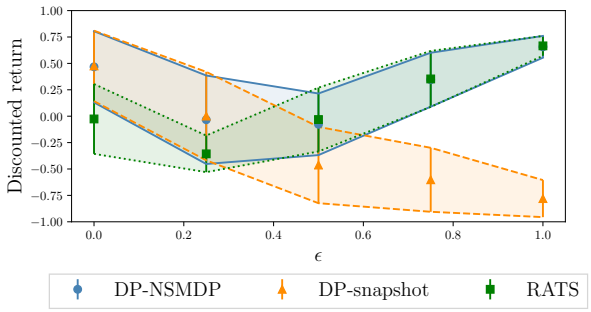


FIGURE 3 – Discounted return represented with 50% of the standard deviation for each algorithm against ϵ.

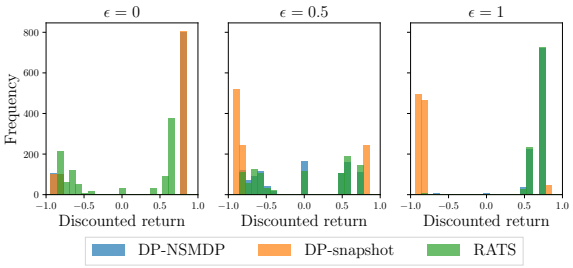


FIGURE 4 – Discounted return distributions of the three algorithms represented for $\epsilon \in \{0, 0.5, 1\}$.

worst-case performance guarantees of robustness, as, for instance in critical certification processes.
Figure 4 displays the return distributions of the three algorithms for $\epsilon \in \{0, 0.5, 1\}$. The effect seen here is the tendency for RATS to diminish the left tail of the distribution corresponding to low returns for each evolution. It corresponds to the optimized criteria i.e. robustly maximizing the worst-case value. A common risk measure is the Conditional Value at Risk (CVaR) defined as the expected return in the worst q% cases. We illustrate the CVaR at 5% achieved by each algorithm in Table 1. Notice that RATS always maximizes the CVaR compared to both DP-snapshot and DP-NSMDP. Indeed, even if the latter uses the true model, the optimized criteria in DP is the expected return.

ϵ		RATS	DP-snapshot	DP-NSMDP
0	$\mathbb{E}$[R]	-0.026	**0.48**	**0.47**
	CVaR	**-0.81**	-0.90	-0.9
0.5	$\mathbb{E}$[R]	**-0.032**	-0.46	**-0.077**
	CVaR	**-0.81**	-0.90	**-0.81**
1	$\mathbb{E}$[R]	**0.67**	-0.78	**0.66**
	CVaR	**0.095**	-0.90	**-0.033**

TABLE 1 – Expected return denoted by $\mathbb{E}$[R] and CVaR at 5%.

7 Conclusion

We have proposed an approach for robust zero-shot planning in non-stationary stochastic environments. We introduced the framework of Lipchitz-Continuous non-stationary MDPs (NSMDPs) evolving with time and have derived the Risk-Averse Tree-Search (RATS) algorithm able to predict the worst-case evolution and to plan optimally w.r.t. this worst-case NSMDP. We analysed RATS theoretically and showed that it approximates the true worst-case NSMDP with a control parameter that is the depth of the search tree. We showed empirically the benefit of the approach that searches for the highest lower bound on the worst achievable score. RATS provides a policy robust to every possible evolution of the environment, i.e. maximizing the expected worst-case outcome on the whole set of possible NSMDPs. Our method was here applied to the uncertainty on the evolution of a model. More generally, it could be extended to any uncertainty on the model used for planning, given bounds on the set of the feasible true model. The purpose of this contribution is to lay the basis of worst-case analysis for robust solutions to NSMDPs. As is, RATS cannot scale with the size of the state-action space and adapting it to larger (possibly continuous) decision problems is an exciting future challenge.

Références

David Abel, Dilip Arumugam, Lucas Lehnert, and Michael Littman. State Abstractions for Lifelong Reinforcement Learning. In *International Conference on Machine Learning*, pages 10–19, 2018a.

David Abel, Yuu Jinnai, Sophie Yue Guo, George Konidaris, and Michael Littman. Policy and Value Transfer in Lifelong Reinforcement Learning. In *International Conference on Machine Learning*, pages 20–29, 2018b.

Kavosh Asadi, Dipendra Misra, and Michael L Littman. Lipschitz continuity in model-based reinforcement learning. *arXiv preprint arXiv :1804.07193*, 2018.

Hendrik Baier and Mark H. M. Winands. MCTS-Minimax Hybrids. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(2) :167–179, 2015.

Hendrik Baier and Mark H. M. Winands. MCTS-Minimax Hybrids with State Evaluations. *Journal of Artificial Intelligence Research*, 62 :193–231, 2018.

Richard Bellman. *Dynamic programming*. Princeton, USA : Princeton University Press, 1957.

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1) :1–43, 2012.

Sébastien Bubeck and Rémi Munos. Open loop optimistic planning. In *10th Conference on Learning Theory*, 2010.

L. Campo, P. Mookerjee, and Y. Bar-Shalom. State estimation for systems with sojourn-time-dependent Markov model switching. *IEEE Transactions on Automatic Control*, 36(2) :238–243, 1991.

Samuel P.M. Choi, Dit-Yan Yeung, and Nevin L. Zhang. Hidden-mode Markov decision processes for nonstationary sequential decision making. In *Sequence Learning*, pages 264–287. Springer, 2000.

Samuel P.M. Choi, Nevin Lianwen Zhang, and Dit-Yan Yeung. Solving hidden-mode Markov decision problems. In *Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics, Key West, Florida, USA*, 2001.

Jen Jen Chung, Nicholas R.J. Lawrance, and Salah Sukkarieh. Learning to soar : Resource-constrained exploration in reinforcement learning. *The International Journal of Robotics Research*, 34(2) :158–172, 2015.

Bruno C. Da Silva, Eduardo W. Basso, Ana L.C. Bazzan, and Paulo M. Engel. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 217–224. ACM, 2006.

Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. *arXiv preprint arXiv :1710.10044*, 2017.

Samuel Choi Dit-Yan, Samuel P.M. Choi, Dit-yan Yeung, and Nevin L. Zhang. Hidden-mode Markov decision processes. In *In IJCAI Workshop on Neural, Symbolic, and Reinforcement Methods for Sequence Learning*. Citeseer, 1999.

Kenji Doya, Kazuyuki Samejima, Ken-ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. *Neural computation*, 14(6) :1347–1369, 2002.

Drew Fudenberg and Jean Tirole. Game theory, 1991. *Cambridge, Massachusetts*, 393(12) :80, 1991.

Emmanuel Hadoux. *Markovian sequential decision-making in non-stationary environments : application to argumentative debates*. PhD thesis, UPMC, Sorbonne Universites CNRS, 2015.

Emmanuel Hadoux, Aurélie Beynier, and Paul Weng. Sequential decision-making under non-stationary environments via sequential change-point detection. In *Learning over Multiple Contexts (LMCE)*, 2014.

R. A. Howard. Semi-Markovian decision processes. In *Proceedings of international statistical institute, Ottawa, Canada*, 1963.

Robin Jaulmes, Joelle Pineau, and Doina Precup. Learning in non-stationary partially observable Markov decision processes. In *ECML Workshop on Reinforcement Learning in non-stationary environments*, volume 25, pages 26–32, 2005.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2) :99–134, 1998.

Thomas Keller and Malte Helmert. Trial-based heuristic tree search for finite horizon MDPs. In *ICAPS*, 2013.

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 681–690. ACM, 2008.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

Marc Lanctot, Mark H.M. Winands, Tom Pepels, and Nathan R. Sturtevant. Monte Carlo tree search with heuristic evaluations using implicit minimax backups. *arXiv preprint arXiv :1406.0486*, 2014.

Terran Lane, Martin Ridens, and Scott Stevens. Reinforcement learning in nonstationary environment navigation tasks. In *Advances in Artificial Intelligence*, pages 429–440. Springer, 2007.

Erwan Lecarpentier, Sebastian Rapp, Marc Melo, and Emmanuel Rachelson. Empirical evaluation of a Q-Learning Algorithm for Model-free Autonomous Soaring. *arXiv preprint arXiv :1707.05668*, 2017.

Erwan Lecarpentier, Guillaume Infantes, Charles Lesire, and Emmanuel Rachelson. Open loop execution of tree-search algorithms. *IJCAI*, 2018.

Rémi Munos et al. From bandits to monte-carlo tree search : The optimistic principle applied to optimization and planning. *Foundations and Trends® in Machine Learning*, 7(1) :1–129, 2014.

Margarita Osadchy, Yann Le Cun, and Matthew L. Miller. Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research*, 8 (May) :1197–1215, 2007.

Jason Pazis and Ronald Parr. PAC Optimal Exploration in Continuous Space Markov Decision Processes. In *AAAI*, 2013.

Matteo Pirotta, Marcello Restelli, and Luca Bascetta. Policy gradient in lipschitz Markov Decision Processes. *Machine Learning*, 100(2-3) :255–283, 2015.

Martin L. Puterman. *Markov decision processes : discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Emmanuel Rachelson and Michail G. Lagoudakis. On the locality of action domination in sequential decision making. 2010.

Daniel L. Silver, Qiang Yang, and Lianghao Li. Lifelong Machine Learning Systems : Beyond Learning Algorithms. In *AAAI Spring Symposium : Lifelong Machine Learning*, volume 13, page 05, 2013.

David Silver, Aja Huang, Chris .J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587) :484, 2016.

Richard S. Sutton, Andrew G. Barto, et al. *Reinforcement learning : An introduction*. MIT press, 1998.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs : A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2) :181–211, 1999.

Cédric Villani. *Optimal transport : old and new*, volume 338. Springer Science & Business Media, 2008.

Marco A. Wiering. Reinforcement learning in dynamic environments using instantiated information. In *Machine Learning : Proceedings of the Eighteenth International Conference (ICML2001)*, pages 585–592, 2001.

Comparaison entre optimisation et Q-learning : application pour l'aide à la décision de la pêche en Corse

Martelloni Paul-Henri[1]
Bisgambiglia Paul-Antoine[3]

Poiron-Guidoni Nicolas[2]
Bisgambiglia Paul-Antoine[4]

Université de Corse

[1] martelloni_ph@univ-corse.fr
[2] nicolaspg2b@gmail.com
[3] bisgambiglia_pa@univ-corse.fr
[4] bisgambiglia_p@univ-corse.fr

Résumé

Dans cet article, nous présentons deux approches d'aide à la décision appliquées à un modèle de pêcherie. La première consiste à se servir de méthodes d'optimisation via simulation pour déterminer les quotas optimaux. La seconde à entrainer un algorithme de q-learning pour déterminer une politique de pêche en temps réel. L'objectif est la simulation de modèle biologique à partir d'ensemble de données réelles, afin de mettre un place un processus de décision pour les pêcheurs. Ainsi, nous comparons les résultats de nos algorithmes à plus de 50 ans de données de captures afin de démontrer leur intérêt dans ce contexte. Enfin, nous comparons les résultats des deux approches, identifions les avantages et inconvénients de chacune et le contexte dans lequel les utiliser.

Mots Clés

Optimisation, Apprentissage, Modèle, Aide à la Décision.

Abstract

In this article, we present two decision support approaches applied to a fishery model. The first is to use simulation-based optimization methods to determine optimal quotas. The second to train a q-learning algorithm to determine a real-time fishing policy. The objective is the simulation of biological models from real data sets, in order to set up a decision-making process for fishermen. Thus, we compare the results of our algorithms to more than 50 years of captures data in order to demonstrate their interest in this context. Finally, we compare the results of the two approaches, identify the advantages and disadvantages of each and the context in which to use them.

Keywords

Optimization, Learning, Model, Decision Making.

1 Introduction

L'Institut Français de Recherche pour l'Exploitation de la Mer (IFREMER) vient de publier un rapport [5] montrant que la surpêche touche un quart des stocks de poissons pêchés en France : "48% des volumes pêchés en France sont issus de stocks de poissons exploités durablement, et 27% de stocks surpêchés.". Les océans étant la principale source de protéines de la planète la gestion durable des stocks est un enjeu fondamental. C'est l'objectif du projet FEDER MoonFish à l'échelle des côtes Corse. En Corse, les métiers de la pêche côtière sont artisanaux. Quatre prud'homies (Bastia, Balagne, Bonifacio, Ajaccio) se partagent plus de 1 000 kilomètres de côtes, avec 42 ports ou points de débarquement, pour environ 200 navires (49 Bastia, 23 Balagne, 44 Bonifacio, 76 Ajaccio) et 300 marins pêcheurs. L'outil informatique, au travers des méthodes de planification, de décision et d'apprentissage, nous permet détudier l'évolution des stocks et de dégager des préconisations voire d'aider à la prise de décision. L'objectif de ce travail est de tester deux types de méthodes : optimisation et apprentissage par renforcement sur des simulations de modéle de pêcherie. Nous pourrons ainsi visualiser l'état des ressources halieutiques et les données d'exploitation. Les résultats obtenus sont comparés à des données réelles sur plus de 50 ans d'exploitation pour mettre en évidence que l'exploitation des ressources halieutiques n'est pas optimale, et qu'une exploitation durable et optimisée peut permettre d'obtenir de meilleurs résultats sur les plans écologique comme économique.
Les apports de ce travail sont :

- Le calage d'un modèle biologique de croissance de population à partir de données de pêches.
- La comparaison de deux méthodes d'aide à la décision dans le but de proposer des stratégies de pêche.

Dans une première section, nous commencerons par présenter les notions de base : en commençant par les modèles de simulation de pêcheries puis les méthodes que nous allons utiliser pour améliorer l'exploitation. Ensuite, nous présenterons la façon dont nous sommes passés de données de captures réelles aux dynamiques de population de certaines espèces d'intérêts. Finalement nous montrerons les résultats de chacune des méthodes et discuterons de leurs avantages et inconvénients.

2 Background

Dans cette section, nous présentons les notions à la base de notre travail.

2.1 Modèles de pêcherie

Les modèles de pêcheries se basent à l'origine sur des modèles biologiques de croissance de population, prenant en compte une estimation des migrations, des naissances et de la mortalité. Nous retrouvons d'une part les modèles globaux, qui considèrent une population dans sa globalité, et un individu comme un élément d'une population homogène ; et, d'autre part les modèles structuraux qui distinguent des classes d'âge, de poids ou de taille dans lesquelles les individus vont êtres classés. Le modèle de Leslie étant le plus connu [8].

Ils prennent en compte des détails permettant de tester des stratégies plus précises, telle que la modification de la taille des mailles de filet. Néanmoins, pour être utilisés, ils nécessitent une grande quantité de données aussi bien sur la population que sur l'exploitation. C'est pourquoi ils semblent plus adaptés à la gestion de pêcherie industrielle mono-spécifique.

Les modèles globaux ont l'avantage d'être utilisable et paramétrable avec peu de données. En ayant une connaissance des prises et des efforts de pêches appliqués par l'exploitation, les quelques paramètres nécessaires peuvent être déduits. Une notion particulière de ces modèles est de considérer que sans exploitation, la population tend vers un niveau d'équilibre. Par contre, la perte de précision empêche toute régulation en matière de poids ou de taille des individus.

De nombreuses études ont déjà été faites dans le monde pour aider les professionnels et les décideurs du domaine dans le but d'avoir la meilleure gestion possible des exploitations. Nous pouvons en particulier parler de MEFISTO [9] qui cherche à proposer des solutions permettant une reconstitution des stocks tout en préservant des revenus corrects aux exploitants [4] [20]. Nous pouvons également parler du modèle de Graham-Schaefer [17] [18] utilisé et associé à une méthode de maximisation de la robustesse dans [2].

La plupart des travaux se fixent sur une espèce particulière avec une exploitation de type industrielle. Ce type d'étude n'est pas possible dans le cas qui nous intéresse. En effet, la pêche sur le littoral Corse est de type artisanale avec des petites embarcations exploitant de nombreuses espèces. Ce type d'exploitation rend difficile l'obtention de données précises sur une espèce en particulier, C'est pourquoi nous avons trouvé préférable de nous orienter vers un modèle de type global.

Nous avons choisi le modèle de Graham-Schaefer. Il est basé sur le modèle généralisé de production de stock introduit par [14], et suit l'équation suivante:

$$\frac{dB(t)}{dt} = r(1 - \frac{B(t)}{k})B(t) - C(t) \quad (1)$$

avec:

- k : la biomasse à l'équilibre (en masse)
- r : le taux de croissance par unité de temps
- q : la capturabilité par unité d'effort
- B_0 : la biomasse à l'instant initial (en masse)

Dans un contexte de pêcherie, nous introduisons un cinquième paramètre $E(t)$ représentant l'effort appliqué sur l'espèce d'un temps t à un temps $t + 1$. Ainsi, les prises sont données par la formule suivante

$$C(t) = qE(t)B(t) \quad (2)$$

2.2 Optimisation

De façon général, l'optimisation est un processus cherchant à déterminer le meilleur ensemble de paramètres possible sur un problème et des critères donnés. Le but est de maximiser ou minimiser une ou plusieurs fonction(s) objectif(s). Les méthodes d'optimisation peuvent être déterministes ou stochastiques. Lorsque la complexité du problème devient trop importante, tel que l'optimisation via simulation de modèle biologique, on privilégiera des méthodes stochastiques, approchées telles que les métaheuristiques [19]. Celles-ci permettent d'obtenir des solutions acceptables en temps limité à la différence des méthodes déterministes qui demandent souvent un grand temps de calcul. Elles ne peuvent cependant pas assurer l'optimalité de la solution proposée.

En gestion des pêcheries, les métaheuristiques ont déjà été utilisés avec succès pour résoudre des problèmes tels que l'optimisation des routes de surveillance des zones de pêche [10] ou pour l'optimisation de stratégies de gestion d'une pêcherie mono-espèce par la détermination de taux de pêche optimum pour chaque sous-zone et année [23].

2.3 Apprentissage par renforcement

Les prémices du Reinforcement Learning remontent aux années 1960, où l'on retrouve les premiers travaux dans lesquels un algorithme apprend une tache par un système "d'essai-erreur". Ainsi nous trouvons

l'exemple d'un jeu de morpion [11]. D'apres [13] l'apprentissage par renforcement tel que nous le connaissons aujourd'hui a été développé depuis la fin des années 1980 avec l'ajout de la notion de processus de décision markovien (MDP) et de la fonction d'optimalité de Bellman [1].
D'après [6] l'apprentissage consiste, pour chaque agent, à interagir avec son environnement pour maximiser une récompense (Reward) résultante de ces actions. Un problème de reinforcement learning est formulé comme un MDP [15].
Les MDPs sont une extension des chaînes de Markov. Cette extension comporte l'addition des actions choisies par l'agent et des récompenses gagnées. Il est composé de cinq éléments principaux, l'espace d'état S, l'espace d'action A, l'environnement ϵ, la fonction de récompense r et un agent de renforcement qui interagit avec l'environnement via une politque π. Celle-ci permet à l'agent de choisir une action en fonction de l'état courant du système. Ainsi, le processus d'apprentissage se déroule de la façon suivante:

- l'agent reçoit l'état de l'environnement s_t et sélectionne une action a_t
- l'environnement répond à l'action en produisant un nouvel état s_{t+1}
- une récompense r est déterminée en fonction de s_t, a_t et s_{t+1}
- la politique π est mise à jour en fonction de la récompense générée. Le but étant de maximiser l'ensemble des récompenses sur le long terme.

3 Expérimentations

Dans le cas d'une pêcherie, il est presque impossible d'obtenir des données de biomasse précises. Les remontées des professionnels sont souvent sous estimées et les embarquements scientifiques trop fragmentaire pour donner une bonne représentativité de l'activité. En revanche, des données de captures peuvent être relevées ou estimées. À ces captures, il est possible d'associer un effort de pêche, quantifiant à quel point la pêche a été intensive ou non.
Dans [7], les auteurs présentent un ensemble d'estimations de captures pour 6 espèces d'intérêts sur 58 ans. Les données qu'ils présentent sont discutables, mais elles conviennent parfaitement à une approche théorique telle que nous le proposons ici. Dans un premier temps, nous allons réaliser une optimisation via simulation dont le but est de caler les paramètres du modèle de Graham-Schaeffer pour chacune de ces espèces.

3.1 Calage du modèle

Dans un premier temps, la population initiale est créée selon le tableau 1 montrant les bornes inférieurs et supérieurs de chaque paramètre. On y retrouve chacun des 4 paramètres du modèle, ainsi qu'un ensemble de 58 valeurs, représentant l'effort de pêche à appliquer pour chacune des années de données que nous avons.

Paramètre	Borne inférieure	Borne supérieure
q	0.05	0.5
r	0.05	0.5
Biomasse initiale	10	5000
k	10	5000
Effort[58]	0	5

TAB. 1 – *Bornes des paramètres de l'optimisation*

À chaque itération et pour chaque solution potentielle, une simulation de la pêcherie va être réalisée. Cela va déterminer des captures associées à la dynamique de population proposée. La solution est alors évaluée en comparant captures "réelles", C, et captures simulées, Cs, suivant la fonction suivante:

$$f(x) = \sum_{i=0}^{nbAnnee} [C_i - Cs_i]^2$$

Les solutions proposées seront alors modifiées de différentes façon en fonction de l'algorithme utilisé afin de caler le plus possible aux captures réelles au cours des itérations. Après avoir tester plusieurs méthodes, l'essaim particulaire standard de 2007 (SPSO 2007, [3]) semble être particulièrement efficace sur ce problème.
La figure 1 montre les courbes d'évolution des captures réelles de denti, en pointillées bleus, et celles simulées avec la dynamique de population que nous déterminons, en trait plein rouge. Les deux courbes étant presque parfaitement superposées, le calage est donc efficace.

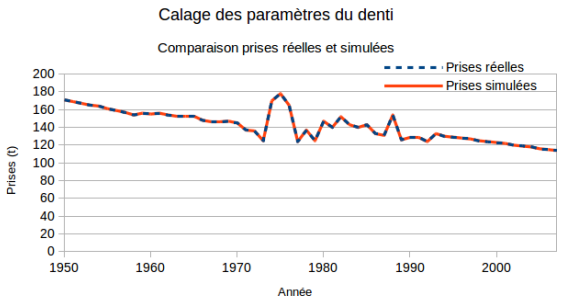


FIG. 1 – *Comparaison de l'évolution des prises réelles et simulées de denti*

Cette méthode est parfaitement efficace pour toutes les espèces sauf la sardine où il semble impossible d'obtenir les même valeurs de captures en ayant une solution cohérente biologiquement. En effet, pour parfaitment coller aux données, il est nécessaire d'étendre

les bornes des paramètres jusqu'à atteindre un taux de reproduction ou une biomasse bien trop important. Ainsi, on peut raisonnablement penser que ces données sont incohérentes, nous allons donc exclure cette espèce de notre étude.

Dans la suite de ce document, nous utiliserons les dynamiques de populations de chaque espèce, déterminées à partir de notre méthode, afin de tester des stratégies de pêche que nous comparerons aux résultats des données de [7].

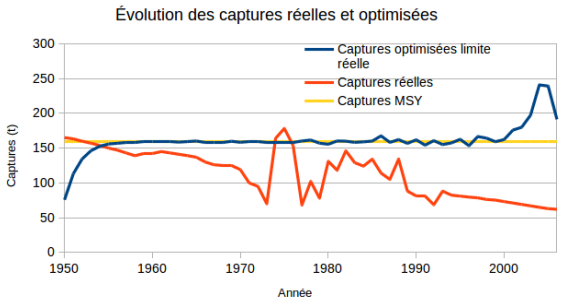


FIG. 2 – *Comparaison des captures réelles et optimisées de rascasses*

3.2 Résultats par optimisation

Pour cette application, nous repartons des résultats présentées précédemment en cherchant cette fois à maximiser le nombre de captures pour chaque espèce. On a donc, q, r, Biomasse initiale et k (tableau 1) connus et cherchons l'ensemble des efforts à appliquer par an. Nous avons également choisit de définir une contrainte: avoir plus de biomasse finale que dans la réalité.

Cette contrainte étant assez simplement satisfaisable, il n'est pas nécessaire d'utiliser des méthodes spécialisées dans la gestion des contraintes telle que le stochastic ranking [16], une simple gestion via la méthode de la peine de mort est suffisante [19]. La fonction d'évaluation étant particulièrement rapide, nous pouvons nous permettre d'utiliser des méthodes de type recherche à voisinage variable générale qui semblent particulièrement adapté à ce type de problème [12].

La figure 2 montre l'évolution des captures réelles et optimisées au cours du temps pour une espèce. Les captures optimisées (en bleu) sont, presque tout au long de la simulation, supérieurs aux captures réelles. On remarque de plus qu'elles se calent sur une valeur remarquable: le Most Sustainable Yield (MSY), sans avoir donnée d'indication préalable quant à cette valeur. Elle représente le maximum de prises que l'on peut prendre par unité de temps sans impacter le stock et est un indicateur biologique bien connu. En revanche, en fin de simulation, le niveau de prise augmente. On peut facilement en déduire qu'il s'agit d'un biais d'optimisation. En effet, la contrainte étant qu'en fin de simulation le stock soit supérieur au stock réel, celle-ci se permet de prendre plus car il n'y aura pas le temps de subir les conséquences.

Ainsi, si on trace les courbes d'évolution des biomasses (figure 3), on voit bien que le niveau de stock augmente jusqu'à atteindre la valeur de stock du MSY, s'y stabilise (ou oscille autour) jusqu'en fin de simulation où il va finir par rejoindre le stock réel. Ce phénomène permet de mettre en lumière le fait que l'optimisation est un problème complexe dont il faut comprendre les mécanismes et analyser les résultats humainement plutôt que de toujours considérer la fonction d'évaluation comme une vérité absolue. Dans ce cas précis, il peut alors être intéressant de modifier la contrainte afin de limiter le niveau de stock au niveau du MSY. Cette nouvelle stratégie est représentée par la courbe verte.

Le tableau 2 montre ainsi la différence de résultat entre les stratégies pour cette espèce. La "stratégie MSY" permet ainsi d'obtenir un nombre de captures bien supérieur à la réalité tout en finissant à un niveau de stock optimal permettant ainsi une meilleure durabilité de l'exploitation.

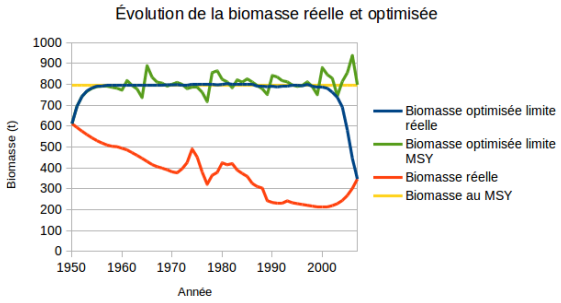


FIG. 3 – *Comparaison de l'évolution des biomasses, réelle, optimisée et optimisée de rascasses sous contrainte du msy*

	Réalité	Strat. 1	Strat. MSY
Captures totales	6475	9147,51	8791,61
Biomasse Finale	347,02	347,03	798,22

TAB. 2 – *Captures totales et biomasse finale de rascasses pour les différentes stratégies*

Finalement, la figure 4 montre les prises totales de chacune des 5 espèces d'intérêt en fonction de la stratégie utilisée. Les résultats peuvent être très variables mais toujours positifs. Cette différence s'explique cependant facilement par les différences dans la qualité de l'exploitation.

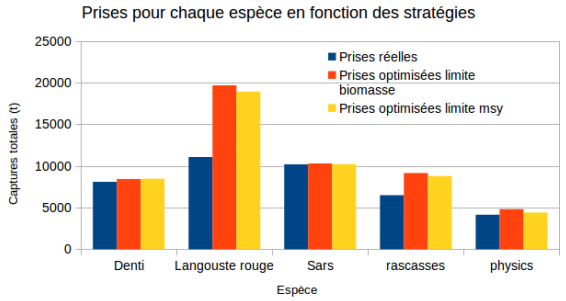


FIG. 4 – *Comparaison du nombre de prise pour chaque espèce en fonction de la stratégie proposée*

Nous savons cependant qu'en réalité il est impossible d'obtenir des résultats aussi précis que ceux donnés par une optimisation de ce type. De plus, l'exploitation réelle est également plus complexe à modéliser. Il peut donc être intéressant de se pencher vers d'autres types de méthodes tel que l'apprentissage par renforcement.

3.3 Résultats par apprentissage par renforcement

Dans cette section, nous repartons de la même problématique en l'adaptant à l'apprentissage par renforcement et plus spécifiquement à l'algorithme de QLearning. Les informations nécessaires sont donc: l'état s_n , l'action a_n, la recompense r_n et l'état s_{n+1}. Cette simplicité a fait la réputation de l'algorithme. Très connu et probablement le plus utilisé en raison des preuves formelles de convergence qui ont accompagné sa publication [21].
A partir de là, nous cherchons à déterminer la meilleure action à effectuer, soit ici l'effort de pêche à appliquer. Nous pouvons donc définir le processus d'apprentissage ainsi:

- L'agent regarde la biomasse et détermine l'effort de pêche à appliquer;
- L'environnement évolue suivant les prises effectuées;
- Une récompense est générée en fonction de la biomasse résultante et des prises effectuées (tableau 3);
- La politique est mise à jour pour maximiser la récompense totale.

Les récompenses de base correspondent la quantité de biomasse capturée. Néanmoins l'objectif étant dans l'idéal d'avoir des prises maximum nous avons choisi d'augmenter légèrement la récompense si les prises sont supérieurs aux prises au MSY. De la même façon nous souhaitons obtenir une population supérieurs à la population réelle donc nous avons là aussi diminué légèrement la récompense si la population est inférieure à la population réelle. Enfin si le stock est détruit nous appliquons un fort malus.

	Récompense
$C > C_{MSY}$	prises (t) + 10
$C \leq C_{MSY}$	prises (t)
B < B réelle finale	-10
B $\approx$ 0	-10000

TAB. 3 – *Récompenses obtenue chaque année*

La figure 5 montre l'évolution de la biomasse réelle et simulée pour une espèce. La biomasse simulée (en bleu) est quasiment constante et est toujours supérieure à la biomasse réelle (orange).

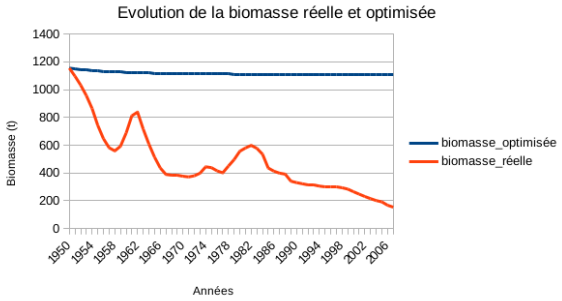


FIG. 5 – *Comparaison de la biomasse de langoustes simulée et réelle*

En parallèle nous voyons sur la figure 6 l'évolution des captures réelles et simulées de la même population. Nous pouvons remarquer que les captures simulées (en bleu) sont là encore quasiment constantes ce qui correspond bien à l'évolution de la population de la figure 5.
Sur les premières années, les prises simulées sont inférieures aux prises réelles. Cela permet cependant de garder une population stable au cours du temps. Ainsi, sur la durée, nous arrivons à assurer des prises et une biomasse supérieure à la réalité.
Le tableau 4 permet de comparer la population finale et la population réelle ainsi que la quantité totale de captures sur toute la durée de la simulation. Nous voyons donc bien que malgré des prises plus faibles en

début de simulation, la stabilité de celles-ci sur l'ensemble de la simulation aboutit au fait qu'au final les prises simulées sont supérieures aux prises réelles.

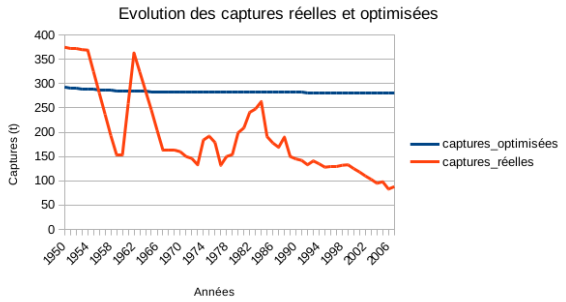


FIG. 6 – *Comparaison de l'évolution des captures de langoustes, réelle et optimisée par apprentissage par renforcement*

	Réalité	Simulation
Captures totales	11069	16443,51
Biomasse finale	153,39	1109,94

TAB. 4 – *Captures totales et biomasse finale de langouste rouge par RL*

L'apprentissage par renforcement permet de s'orienter vers un comportement plus réaliste. Nous avons un agent qui tente d'améliorer ses gains en recevant une récompense pour ses captures, ainsi qu'un possible malus si la biomasse est trop faible. Bien que la réalité du problème soit bien plus complexe que cela à modéliser, nous pouvons penser que ce type de méthode peut être intéressante à exploiter. Nous retrouvons néanmoins l'une des limites de ce type de méthodes quand le nombre d'états deviens grand [22].

4 Comparaison et discussion

Comme nous pouvons le constater dans le tableau 5, les deux méthodes présentent des résultats comparable pour 3 espèces sur 5. Pour les langoustes rouges et les rascasses, les résultats par optimisation sont cependant nettement meilleurs.

Pour le denti, on remarque que la valeur de biomasse au MSY est inférieur à la biomasse finale réelle. Cela peut être signe d'une sous-exploitation.

Ces résultats sont cependant à pondérer car le calage (section 3.1) peut présenter des problèmes d'équifinalité et une dynamique de population différente peut alors conduire à des conclusions différentes, ce travail reste théorique.

L'optimisation, bien que présentant de meilleurs résultats, est difficile à utiliser à l'échelle de la personne. En effet, cette approche suppose un contrôle parfait de la pêcherie. Elle semble donc particulièrement utile pour déterminer des quotas et des comportements optimaux pour la pêcherie dans sa globalité. À l'inverse, l'apprentissage par renforcement semble pouvoir s'appliquer comme outil d'aide à la décision individuel par exemple via des recommandations de comportement à préconiser pour chaque exploitant plutôt que pour les décideurs.

Enfin, il semble que l'optimisation soit plus intuitive à mettre en place. En effet, le couple fonction d'évaluation/contrainte semble plus simple à utiliser de façon efficace que récompense/malus de l'apprentissage par renforcement du fait de la difficulté à déterminer un malus optimal.

5 Conclusion et perspectives

Dans ce travail, nous nous sommes basés sur des données estimées afin de proposer un modèle de la pêche en Corse. Celui-ci nous a alors permis d'essayer deux méthodes d'aide à la décision et de les comparer aux résultats réels avec succès. Ainsi, nous avons pu montrer l'utilité de ces approches dans un contexte de pêcherie. Elles semblent cependant chacune adapté à un objectif différent. L'optimisation pour déterminer des quotas et comportement généraux, s'adressant plutôt aux décideurs. L'apprentissage par renforcement s'adressant plus directement aux exploitants via des recommandation en temps réel.

La complexité de la réalité est cependant bien supérieure à notre modèle. Ainsi, la complexification future du modèle de pêcherie et les limites des méthodes de RL nous forceront, à l'avenir, à nous orienter vers des méthodes telles que le Deep Reinforcement Learning (DRL).

Remerciement

Ce travail est réalisé dans le cadre du projet européen MoonFish CO0011363, PO FEDER-FSE 2014-2020

Références

[1] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[2] Hans-Georg Beyer and Bernhard Sendhoff. Robust optimization–a comprehensive survey. *Computer methods in applied mechanics and engineering*, 196(33):3190–3218, 2007.

[3] Maurice Clerc. Beyond standard particle swarm optimisation. In *Innovations and Developments of Swarm Intelligence Applications*, pages 1–19. IGI Global, 2012.

[4] Jordi Guillen, Francesc Maynou, Christos Floros, David Sampson, Alexis Conides, and Kostas Kapiris. A bio-economic evaluation of the potential for establishing a commercial fishery on two newly developed stocks: The ionian red shrimp fishery. *Scientia Marina*, 76(3):597–605, 2012.

	C réelles	C opti msy	C RL	B finale réelle	B finale opti msy	B finale RL
Denti	8095	8467,70	8253,86	614,71	530,49	528,82
Langouste rouge	11069	18934,45	16443,51	153,39	1421,15	1109.94
Sars	10194	10417,70	10210,76	611,77	727,24	952.20
Rascasses	6475	8791,61	6511,32	347,02	798,22	491.76
Physics	4146	4412,78	4325,21	164,98	759,90	947.34

Tab. 5 – *Comparaison des prises et biomasse finale de chaque espèce en fonction de l'algorithme utilisé*

[5] IFREMER. Les ressources halieutiques franaises : bilan 2018. `https\protect\relax\unskip\penalty\@M\protect\relax\kern.16667em:\@beginparpenalty=\@M\relax//wwz.ifremer.fr/content/download/124503/file/DP_halieutique_ifremer.pdf`, 2019.

[6] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[7] Marie Catherine Santonif Lejeunec, Jean Michel Culiolif, and Daniel Paulyg. Frédéric le manacha, delphine durab, anthony perecd, jean jacques riutorte, pierre.

[8] Patrick H Leslie. On the use of matrices in certain population mathematics. *Biometrika*, 33(3):183–212, 1945.

[9] Jordi Lleonart, Francesc Maynou, Laura Recasens, and Ramón Franquesa. A bioeconomic model for mediterranean fisheries, the hake off catalonia (western mediterranean) as a case study. *Scientia Marina*, 67(S1):337–351, 2003.

[10] Marta Mesquita, Alberto G Murta, Ana Paias, and Laura Wise. A metaheuristic approach to fisheries survey route planning. *International Transactions in Operational Research*, 2016.

[11] Donald Michie. Trial and error. *Science Survey, Part*, 2:129–145, 1961.

[12] Nenad Mladenović, Milan Dražić, Vera Kovačevic-Vujčić, and Mirjana Čangalović. General variable neighborhood search for the continuous optimization. *European Journal of Operational Research*, 191(3):753–770, 2008.

[13] Groupe PDMIA. Processus décisionnels de markov en intelligence artificielle. *Edité par Olivier Buffet et Olivier Sigaud*, 1, 2008.

[14] Jerome J Pella and Patrick K Tomlinson. A generalized stock production model. *Inter-American Tropical Tuna Commission Bulletin*, 13(3):416–497, 1969.

[15] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[16] Thomas P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on evolutionary computation*, 4(3):284–294, 2000.

[17] Milner B Schaefer. Some aspects of the dynamics of populations important to the management of the commercial marine fisheries. *Inter-American Tropical Tuna Commission Bulletin*, 1(2):23–56, 1954.

[18] Milner B Schaefer. Some considerations of population dynamics and economics in relation to the management of the commercial marine fisheries. *Journal of the Fisheries Board of Canada*, 14(5):669–681, 1957.

[19] Patrick Siarry. *Métaheuristiques*. Editions Eyrolles, 2014.

[20] Silvia Silvestri and Francesc Maynou. Application of a bioeconomic model for supporting the management process of the small pelagic fishery in the veneto region, northern adriatic sea, italy. *Scientia Marina*, 73(3):563–572, 2009.

[21] CJCH Watkins and P Dayan. gtechnical note: Q-learning, h machine learning, vol. 8. 1992.

[22] Florentin Woergoetter and Bernd Porr. Reinforcement learning. *Scholarpedia*, 3(3):1448, 2008.

[23] E Yücesan, CH Chen, JL Snowdon, and JM Charnes. Simulation based optimization in fishery management.

A Deliberative and Reactive Architecture for a Multi-Robot System

Antoine Milot[1] Christophe Grand[1] Charles Lesire[1]

[1] ONERA/DTIS, Université de Toulouse, F-31055 Toulouse, France
firstname.lastname@onera.fr

Abstract

Autonomous multi-robot system must face unforeseen events while executing a mission. To be reliable and overcome these events the system has to be able to quickly make its own decisions and guarantee performance on the mission. Therefore, we propose an architecture based on specialized reactors for a multi-robot system in order to bring both deliberation and reactivity to the system.

Keywords

Decision-making, MRS architecture, Autonomous robots

1 Introduction

When deploying a robot to execute a mission, the operator who supervises the system must be prepared to face unforeseen events such as communication interruptions, breakdowns, or new goals. In the case of a multi-robot system, such events are even more likely to appear. Therefore, the system has to be resilient by embedding a decision-making process to handle those events and be able to guarantee a certain performance on the mission.

However, planning is an expensive process, in terms of CPU time, and the system needs reactivity to be able to overcome quickly unforeseen events. This necessity for a reliable long-term plan and strong reactivity to repair it, combined with the needs to use specialized algorithms to plan specific tasks, leads us to work on a distributed architecture.

Such an architecture allows reactivity and deliberation between its components at the same time to propose a dedicated way to implement these specialized algorithms.

For instance, given a team of robots which accomplish survey tasks, such as to put sentries on fixed positions or to patrol on zones, we propose an architecture divided in specialized modules. Each module will plan specific goals or sub-goals for a Sentry or Patrol task.

2 The T-ReX Architecture

Based on the sense-deliberate-act paradigm, T-ReX [5] interleave sensing, planning and execution around a goal-oriented system. This architecture is divided into reactors, each reactor plans and supervises its own plan. Together, they form a global consistent plan. For instance, a robot capable of moving and observing will have one reactor dedicated to navigation goals, such as *reach a position* or *follow a path*, and another reactor for observation. These reactors send their sub-goals to the robot's functional layer, where no planning or deliberation is needed, to interact with sensors and actuators thanks to elementary orders or basic methods. The reactor takes a goal as input, plans it, and gets sub-goals. Then the supervisor follows sub-goals execution and, reactively, tries to repair the reactor's local plan by replanning failed sub-goals or integrating new goals to the plan. In order to organize goal-planning and deliberation, T-ReX reactors rely on timelines synchronized by an internal clock. Between each tick reactors may deliberate to plan desired future timeline values.

A previous work proposing cooperation between AUVs with T-ReX was done in [3]. Each robot exchanged data with the aim of building a shared local map and using it to take motion decision. However, the cooperation was restrained to this data exchange and no deliberation between robots in order to accomplish a goal was present.

3 A deliberative and reactive architecture for MRS

Our scheme is to bring a deeper cooperation inside a multi-robot system. To this end, we propose an architecture that relies on the concept of specialized reactors, as proposed in T-ReX [5], and sequential planning, therefore we do not implement timelines. The choice not to use timelines is motivated by the will to plan undated goals in order to handle a new task without knowing in advance its start or end time. This kind of application is limited by timelines.

Moreover, each T-ReX reactors usually use a common planning algorithm (EUROPA [2], in [5]; APSI [1], in [4]), we propose to dedicate each reactor to specific tasks by specializing its planning algorithm and supervisor.

Besides, to go further than shared data ([3]) and allow deliberation between robots, we propose to see the multi-robot system as a unique set of reactors, each robot is composed of one or more reactors and a reactor can belong to only one robot. This principle allows us to extend the deliberation/reactivity between reactors to the robots.

3.1 Integration on a multi-robot system

Given a team of heterogeneous robots, composed of rovers and drones with different capacities (velocity, range of

communication, etc.) and sensors (IR camera, etc.), the operator wants to fulfill a survey mission composed of Sentry and Patrol tasks. For instance, the operator wants to put two sentries on certain positions. The parameters of the Sentry goal are the positions to adopt, the sensors to use, the durations, the starting times, and any specific rules (like keeping battery autonomy above 25%).
An example of a possible architecture to accomplish this multi-robot mission is shown in figure 1.

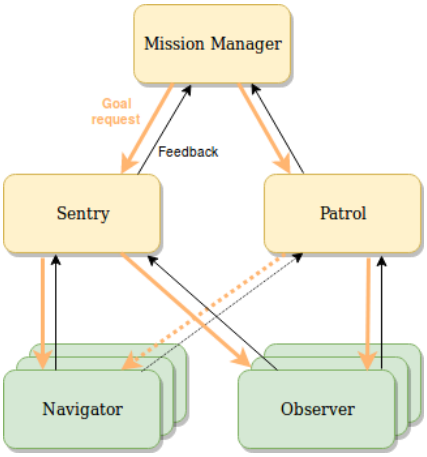


Figure 1: Example of architecture for a MRS

When the operator sends the mission goal, this goal will be received by the Mission Manager, which will issue a Sentry goal to the Sentry Reactor. Next, the latter will use its own algorithm to optimize the preferred criteria (duration or start time) to choose robots and plan the sentry task. The planning will result in navigation and observation sub-goals that will be sent to the concerned robots Navigator and Observer reactors. Finally, these sub-goals will be planned by those reactors, resulting in sub-sub-goals for others low-levels reactors or the robot functional layer (not showed in the figure). This cascade process goes on until a goal is received by a robot functional layer.
We differentiate reactors that organize the mission and plan team tasks, called team reactors (in yellow), from reactors that are proper to the robot operation, called basic reactors (in green). In this example, we propose to place team reactors in a leader that will thus be able to plan multi-robot tasks.

3.2 Goal lifecycle formalism

The multiple exchange of goals between reactors bring the need to describe the evolution of the goal inside the reactors. In T-ReX [5], a reactor can follow goals evolution thanks to external timelines, timelines that belong to others reactors, and act in consequence of a delay. However, we want to accomplish more general goals that are not always based on time, such as a tracking task. To this end we do not rely on timelines and we need to use another approach to observe goals evolution.

Our approach is based on the goal lifecycle formalism proposed in [6] to describe the goal states. In figure 2, arcs (select, expand, etc.) are orders passed by the client reactor, which emitted the sub-goal, to the server reactor that has to plan the sub-goal. These orders will change the goal from one state to another.

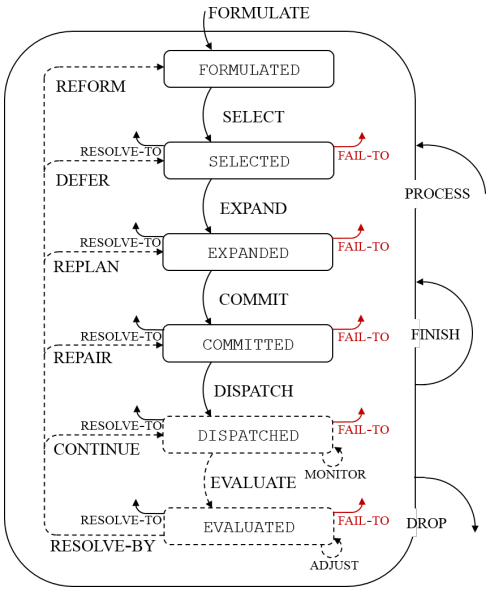


Figure 2: The goal lifecycle from [6]

This is important to highlight that the cascade process described before is supported by the concept of orders. For instance, when the Sentry goal, asked by the Mission Manager, is *Committed* (a plan is set for this goal) and the Mission Manager sends a dispatch order to start its execution, the order is reverberated by the Sentry Reactor to its sub-goals, there, navigation and observation sub-goals. Therefore, the dispatch order is propagated to the concerned robots Navigators and Observers. The process goes on until it reaches the last sub-goal, addressed to the functional layer, and then the result goes up the cascade to the Mission Manager.

3.3 Reactivity and reperation

To highlight the architecture dynamism we propose to study an example based on the architecture proposed in figure 1. In our precedent example, we found a plan to put two sentries on position, took on by two robots, roverA and roverB. If roverB has a breakdown and fail to join its position, its Navigator Reactor will send a Failed lifecycle for the corresponding goal to the Sentry Reactor. Then the Sentry Reactor will cancel the rest of roverB sub-goals. Next, the Sentry Reactor will try to repair the plan, for example asking roverC to do the sentry task. If it is possible to use roverC, the plan will be repaired at this level and continue. Else, the Sentry Reactor will send a Failed lifecycle for the Sentry task to the Mission Manager.

This dynamic allows reparation at each reactor concerned, from bottom to top. Combined with the goal lifecycle formalism we can also specialize reparation methods for each level of failure (expand level, commit level, etc..). Therefore, it is possible to specialize each supervisor reactor in accordance with the mission to handle multiple events and gain in resilience.

4 Conclusion and future work

The implementation of the architecture of the MRS is yet work in progress but rise already some questions, like what to do if the leader breaks down. A possible answer is to implement the same architecture on all robots but to limit reactor access. Thereby, a slave has the Sentry Reactor has a ghost reactor, unable to use it, but if its leader breaks down, it can be promoted and the reactor access unlocked to handle the mission. Another development perspective is how to use undated goals with dated goals and temporal constraints.

References

[1] G. Cortellessa A. Cesta, S. Fratini, A. Oddi, and G. Bernardi. Deploying Interactive Mission Planning Tools - Experiences and Lessons Learned. *Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII)*, 15:1149–1158, 2011.

[2] J. Barreiro, M. Boyce, M. Do, J. Frank, M. Iatauro, T. Kichkaylo, P. Morris, J. Ong, E. Remolina, T. Smith, and D. Smith. EUROPA : A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization. In *International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS)*, Sao Paulo, Brazil, 2012.

[3] A. Belbachir, F.Ingrand, and S. Lacroix. A cooperative architecture for target localization using multiple auvs. *Intelligent Service Robotics*, 5:119–132, 2012.

[4] A. Ceballos, S. Bensalem, A. Cesta, L. De Silva, S. Fratini, F. Ingrand, J. Ocon, A. Orlandini, F. Py, K. Rajan, R. Rasconi, and M. Van Winnendael. A Goal-Oriented Autonomous Controller for space exploration. In *Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 2011)*, Noordwijk, Netherlands, 2011.

[5] C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, and R. McEwen. A deliberative architecture for AUV control. In *ICRA*, St. Paul, Minnesota, USA, 2012.

[6] M. Roberts, L. Hiatt, A. Coman, D. Choi, B. Johnson, and D. Aha. ActorSim, A Toolkit for Studying Cross-Disciplinary Challenges in Autonomy. In *AAAI Fall Symposium*, Arlington, Virginia, USA, 2016.

Learning-based modelling of physical interaction for assistive robots

Andrei Mitriakov*, Panagiotis Papadakis, Mai Nguyen, Serge Garlatti

IMT Atlantique, Lab-STICC, UMR 6285
F-29238 Brest, France

*andrei.mitriakov@imt-atlantique.fr

Abstract

Deploying companion robots for assisting humans requires safe and robust interaction with the environment, both in terms of mobility and object manipulation. To extend a robot's workspace, we are here concerned with multifloor operation and staircase traversal, as a building block for the development of object fetching services. In this article, we advocate a life-long learning treatment of this problem within a reinforcement learning (RL) framework. In view of sparse earlier work for the scenario of interest, we hereby identify relevant methodological aspects and report our preliminary developments.

Keywords

Assistive robotics, Control learning, Autonomous flipper control, Policy search methods

1 Introduction

Among various application areas, service robots can be particularly useful in assisting daily human activities, for example via companion robots [6]. As far as our application of interest is concerned, such robots are assumed to operate within an indoor environment. In such scenarios, robot operation is most often presumed as 2D although, in practice, the environment can be 3D and highly variable. To compensate motor impairments of elderly or frail people with limited autonomy, it becomes reasonable to develop robots with improved autonomy in navigation and object manipulation. Relatedly, for the task of stair traversal, this can be addressed by hard-coded behaviours that make use of the exact robot kinematics and the staircase characteristics [12, 13]. In general terms, this approach is not efficient as it is not easily transferable to different robots. Likewise, manipulation with objects or movement on flat or uneven surfaces separately have been receiving considerable attention during recent years [22, 32, 4]. On the contrary, transporting potentially sensitive objects while performing staircase traversal has not been studied, especially in the context of a personal assistance scenario. Conventionally, stair traversal is performed by methods presented in the work [1] that usually make some artificial hypotheses used in control that makes the robot less adaptable to new situations, i.e. new robot configurations and stair shapes. Finally, reinforcement learning could constitute a more suitable approach for avoiding ad hoc solutions that are customized to particular platforms but it has not widely been studied.

A number of questions are posed in order to accommodate such scenarios. How does the task of safe stair traversal is formulated as a reinforcement learning problem while taking into consideration the presence of an arm carrying a sensitive object? How is it possible to generate actions which respect robot safety? How can novel (unseen) environment states or robot configurations be accommodated? How can we account both for robot safety and other potentially conflicting criteria of performance such as traction? In view of these questions, our research goal is to develop robust algorithms that would allow safe operation for a service robot capable of (i) indoor, multifloor navigation via obstacle negotiation such as stairs or steps and (ii) combination with object-centric assistive services. In this context, the use of approaches that are not based on learning is overly restrictive because of their inferior generalization capability. Therefore, the key challenge of the doctoral research concerns the elaboration of methods for 3D mobility and/or manipulation where the role of learning is eminent, allowing them to be less dependent on a particular platform and to require less expert supervision.

The rest of the paper is structured as follows. Section 2 presents related works of the problem of interest. In section 3 we formulate the problem of stair traversal using notions of reinforcement learning. Section 4 provides preliminary results related to machine vision algorithms for stair detection in color and depth images. Finally, section 5 is focused on future works in navigation and reinforcement learning.

2 Related work

2.1 Learning-free approaches

The task of stair traversal has been predominantly studied in learning-free frameworks in search and rescue applications where robots do not generally face the same constraints as in service robotics, namely, object and environment safety is usually less important relatively to assistive robot applications where it is inadmissible to dam-

age human property. Related robot navigation problems were studied outside the context of machine learning. For example, the authors of [23] propose a framework for autonomous 3D path and motion planning including flipper control for tracked robots where the main idea is to keep flippers tangentially in contact with the surface. Authors show that their solution can face any kind of structure within the limits of the robot's obstacle negotiation abilities, however they only demonstrate it with two trials. In [30] the authors present an approach where a tracked robot with passive sub-crawlers faces obstacles that exceed obstacle negotiation capabilities of [23], a warning system based on normalized energy stability margin (NESM) and a traversal algorithm that is able to apply force against an obstacle in order to climb it. The NESM-based approaches for stability assessment became widespread in robotics while at the same time being easy to understand because the stability criterion simply supposes the calculation of vertical deviation from the lowest stable position of the main robotic chassis [14].

2.2 Learning-based approaches

In this section we begin by presenting works that are more relevant to learning-based staircase traversal and we conclude by focusing on the most significant algorithms. The authors of [25] were among the first to test the deep deterministic policy gradient (DDPG) algorithm for the stair traversal task following an end-to-end fashion. The idea is to get actions from input data received from the Inertial measurement unit (IMU) sensor, front and back cameras. Q-values, which are the quality measure of a state-action combination, and policy parameters were approximated using sophisticated multilayer convolutional neural networks (CNN), which inevitably induce significant processing costs. Another limitation of this work is due to the sole use of time-demanding simulations as the result of the chosen function approximations that require a large number of trials. In turn, this results in lower generalization as a result of training on a single stair type.

Another more recent and elaborate approach is proposed by authors of [26]. The authors' contribution amounts to the development of RL algorithms for the scenario of stair traversal. They implement constraints in the contextual relative entropy policy search (Contextual REPS) algorithm [20] based in turn on [28] and [10]. That extension showed that a small number of iterations is sufficient to learn stair traversal of a previously unknown obstacle where the safety of a rollout is computed by a physics-based simulator. The latter represents software simulating the main interactions and influence of the real-world system and determining as the safety of the rollout as 1 if it is safe or 0 otherwise. However, the authors did not investigate the impact of context represented by variables like the height of an obstacle that do not change during the task execution but might change from task to task. We consider this paper [26] as reference work with respect to the approach that we envision to advance.

3 Reinforcement learning problem

To exploit the advantages of learning-based staircase traversal, we intend a reinforcement learning approach applied to flipper control. Two families of reinforcement learning methods exist in robotics, the first one is the policy search (PS) which learns the policy, and the second one is value-based where one wishes to estimate the quality of the value function in conformity with the expected cumulative reward. PS [31] provides more advantages over the value-based approach in robotics because the former allows expert knowledge integration, domain appropriate prestructuring of the policy, and reduced complexity of policy approximation relative to value function approximation. Finally, small changes in policy do not consequently lead to a large change in a value function and again in the policy [17] like in value-based methods.

3.1 Problem description

We briefly recall associated RL notions following the description given in [19]. We denote the state of the robot as $\boldsymbol{x}$. The vector of control $\boldsymbol{u} \in \mathbb{R}^d$ is generated by the lower-level policy $\pi_l(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\omega})$ that is usually parameterized with feedback controllers, movement primitives or torque profiles [24, 19, 15] by $\boldsymbol{\omega} \in \Omega^d$ where Ω is the space of the lower-level policy parameters and d denotes the total number of degrees of freedom. We decouple policy into the lower and upper-level policies in accordance with [26, 20, 19]. The upper-level policy $\pi_u(\boldsymbol{\omega}|\boldsymbol{s})$ provides the parametrization $\boldsymbol{\omega}$ of the lower-level policy and can be generalized to different shapes of stairs. Let $\boldsymbol{s}$ the context containing information about the environment such as number, height and depth of steps has to be used in the upper-level policy distribution in order enable the algorithm to choose from different parametrization parameters $\boldsymbol{\omega}$ according to each context.

We consider an episode-based policy search framework with length T. The trajectory $\boldsymbol{\tau} = \{\boldsymbol{x_1}, \boldsymbol{u_1}, .., \boldsymbol{x_T}, \boldsymbol{u_T}\}$ defines the set of state-action pairs. We assume that the context vector $\boldsymbol{s}$ is drawn from an unknown distribution $\mu(\boldsymbol{s})$ and our goal is to learn the optimal upper-level policy $\pi_u^*(\boldsymbol{\omega}|\boldsymbol{s})$ which yields the parametrization of the lower-level policy given the context $\boldsymbol{s}$. The trajectory $\boldsymbol{\tau}$ has the probability $p(\boldsymbol{\tau}|\boldsymbol{s}, \boldsymbol{\omega})$ given the context $\boldsymbol{s}$ and the reward function $R(\boldsymbol{\tau}, \boldsymbol{s})$ depends on the context. In the case of staircase traversal it should further incorporate safety constraints, because the robot has to generate trajectories that are as safe as possible, and potentially additional criteria such as total travelled distance and platform slipping that we estimate using the model proposed in [8]. During learning N rollouts will be generated. In the beginning of every episode the low level parameters $\boldsymbol{\omega}$ are drawn from the upper-level policy $\pi_u(\boldsymbol{\omega}|\boldsymbol{s})$ given the observed context, then the control vector is retrieved from the lower-level policy $\pi_l(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{\omega})$. Lately, the policy parameters are evaluated

on the simulated and real robot, the reward is collected and, lastly, the policy is updated.
The problem formalization may directly serve for a stair climbing learning. The lower-level policy is parametrized by movement primitives whose parameters correspond to the parameter vector $\boldsymbol{\omega}$ of the upper-level policy. This policy defines the command vector $\boldsymbol{u}$, which contains 6 elements which are 2 torques applied to tracks and 4 flipper angles. The robot state $\boldsymbol{x}$ presents actual applied torques to tracks, flipper angles, robot orientation and the position on the stair. The context $\boldsymbol{s}$ comprises information about the environment, e.g. size, number of steps and stair inclination. Every rollout generates a trajectory $\boldsymbol{\tau}$ which has the maximum length T.
Various PS methods can be applied to the described RL problem. We consider that the relative entropy policy search (REPS) algorithm family is the state-of-the-art [26] which we seek to improve. In the next section, we briefly describe algorithms that we have interested in and consider to use as a baseline.

3.2 Policy Search algorithms

REPS. An interesting PS algorithm was developed in [27] based on a conventional RL setting [31]. The REPS keystone is simply the optimization problem where we attempt to find optimal policies that maximize the expected reward while satisfying constraints (cf. eq. 5 - 8 [27]). It binds the loss of information measured between observed data distribution and the data distribution generated by the new policy in order to prevent aggressive policy update steps.

Contextual REPS. The authors of the Contextual REPS [16] were inspired by the recent work [27], they add a task-dependent context s, from which changing features of the environment can be integrated. This also requires to divide the policy into two levels. The lower-level policy is used to generate control commands and typically parametrized with a small number of parameters $\boldsymbol{\omega}$, controllers like linear feedback controllers, movement primitives [15], torque profiles [24] or, even, neural network controllers, that are harder to learn in comparison with others, are usually used [19]. The high-level policy $\pi(\boldsymbol{\omega}|\boldsymbol{s})$ searches for parameters of the lower-level policy $\boldsymbol{\omega}$ maximizing the expected reward (cf. [16]). After sampling the parameter vector $\boldsymbol{\omega}$ from the high level-policy given the context $\boldsymbol{s}$, the lower-level policy $\pi(\boldsymbol{u}|\boldsymbol{x},\boldsymbol{\omega})$ defines the control vector $\boldsymbol{u}$ during the episode based the state $\boldsymbol{x}$ of the robot. Contextual REPS searches over the joint distribution $p(\boldsymbol{s},\boldsymbol{\omega})$ of contexts and parameters to maximize the expected reward J bounding the relative entropy to the previously observed distribution $q(\boldsymbol{s},\boldsymbol{\omega})$. The Contextual REPS optimization problem aims to maximize the expected reward J while satisfying bounding constraints (cf. eq. 6 [19]).

Gaussian Process REPS. The Contextual REPS is further developed by the authors of [19]. Inspired by the recent works [28, 20] they developped a model-based contextual policy search algorithm named gaussian process relative entropy policy search (GPREPS) that learns a representation of the robot dynamics and the reward function giving equivalent learning results 100 times faster than the original REPS. Its main motivation is to improve the data-efficiency of the model-free REPS using artificial rollouts. This algorithm simply adds a Gaussian Process (GP) in the loop, which learns the representation of the system dynamics. At each iteration of the GPREPS algorithm, N trajectories are generated by observing the context and executing the policy on the real system, then the learned models are updated and based on them M artificial samples are created. In the following, the algorithm samples L trajectories and averages over the trajectory rewards getting the expected reward. Finally, these artificial samples are used for optimization of the dual function by updating the upper-level policy.

Constrained REPS. Another improvement is made in [26] that extends the Contextual REPS algorithm with constraints and replaces the GP by a cautious physics-based simulator, which evaluates generated policy in the safety simulator, constrains the upper-level policy distribution reducing the required number of iterations and generates safe trajectories. It was made by extending the original Contextual REPS constraints with an additional one. The upper-level distribution $p(\boldsymbol{s},\boldsymbol{\omega})$ is forced to have the expected safety higher then the vector of bounds $\boldsymbol{\delta}$:

$$\sum_{\boldsymbol{s},\boldsymbol{\omega}} p(\boldsymbol{s},\boldsymbol{\omega})(\mathbf{1} - \boldsymbol{C}_{\boldsymbol{s}\boldsymbol{\omega}}) \leq \boldsymbol{\delta}, \tag{1}$$

where $\boldsymbol{C}_{\boldsymbol{s}\boldsymbol{\omega}}$ is a collection of evaluated quantities which are safety and mechanical constraints, and $\mathbf{1}$ is a vector with all-ones of a corresponding dimension. The optimization problem of the Contextual REPS (cf. eq. 6 [19]) is enriched only with one condition (cf. 1). As was shown in [26] Constrained REPS takes about 40 iterations to learn how to traverse previously unknown obstacles represented by one step. Nonetheless, this is still overly high if we intend to adress learning of unknown stair traversal during the robot operation. Two possible extensions of this algorithm concern learning of the expected return models along with stronger prior knowledge on the policy structure than a prior knowledge of physical limits.

4 Preliminary work

We consider the Jaguar V4 (Fig. 1, `jaguar.drrobot.com`) as the experimental platform, designed for traversing 3D terrain via its active components (flippers). As we consider an indoor deployment of the robot, conventional RGB-D sensors will be used for its situation awareness. Low-level robot control is performed within Robot Operating System (ROS) [29], while the robot is simulated in the Gazebo environment (`gazebosim.org`). Preliminary results concern perception as the achieved performance of this stage will have an influence on all subsequent stages related to learning and execution.

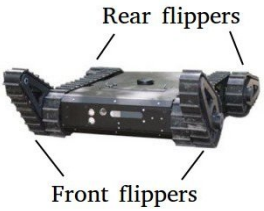
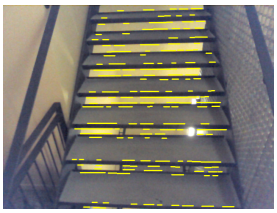
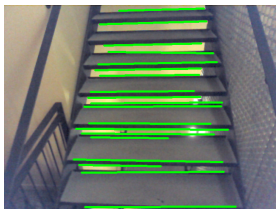


Figure 1: Jaguar V4



(a) Dominant horizontal lines obtained after applying Hough transform



(b) Detected stair edges

Figure 2: Line and step detection

4.1 Perception

The main perception skill required by the robot is to detect and estimate the state of a given staircase. Treating the problem as a sequential estimation problem wherein the robot observes the staircase from multiple viewpoints and two sensor modalities (color (RGB) and depth), a Kalman filter-based fusion scheme is applicable. At this point, we assume that a preliminary algorithm enables the robot to discriminate a staircase from other similar looking objects.

Staircase detection in the RGB image. We hereby assume a right-handed image coordinate system. A staircase is initially detected in the RGB image following ideas borrowed from [5] which consists in estimating a 2D line for each step edge. In detail, the image is converted to grayscale and undergoes erosion and dilation operations to filter out small or noisy line segments. Subsequently, the Canny edge detector together with Hough transform are used to estimate the residual lines that do not intersect with the 2D ground plane (see Fig. 2a) and extract the corresponding line coefficients in 2D image plane. In order to isolate lines corresponding to stairs, the algorithm operates in Hough space where every line is represented by a point (cf. Fig. 3) which coordinates correspond to slope and intercept values of a line.

In example, we can distinguish a multitude of lines that have been detected whose majority clusters around a single line slope, that is assumed to correspond to the line slope of the step edges of the staircase. To extract the lines belonging to the steps of the staircase, the Hough space is uniformly discritized along the slope axis into a fixed number of sections, each one corresponding to a histogram bar. Counting in each histogram bin the total number of point entries, we can thereby determine the dominant line.

Not surprisingly, Hough transform detects multiple line segments with varying length along a step edge, so these lines should be regrouped and filtered by length (cf Fig. 2a).

Regrouping could be done with respect to the fact that segment lines of different step edges have the same slope but different intercept values whereas lines from one step edge have approximately the same intercept value. In order to associate every small segment to corresponding edges, points are grouped in the following way. If the distance between two consecutive points in Hough space is less than the average distance between all consecutive points, these two points belong to one line, if not, this is the next step edge. Grouped line segments are finally filtered by length, so that short segments are discarded (Fig. 2b). Eventually, the coordinates of the staircase are obtained through a bounding box in the RGB image that contains all the grouped lines. These coordinates are then jointly considered with the estimated staircase coordinates obtained by processing the Depth image.

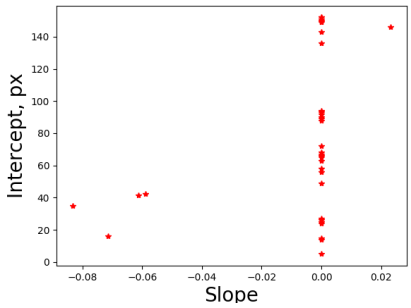
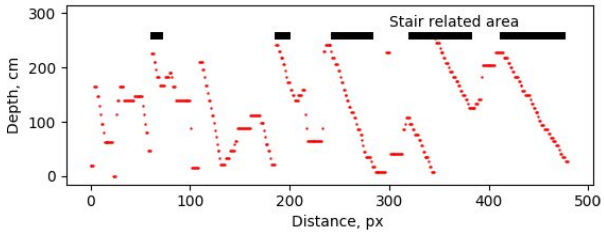


Figure 3: Lines in Hough space

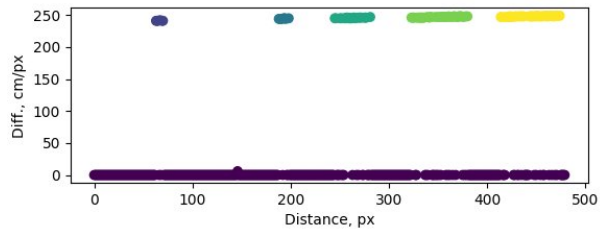
Staircase detection in the depth image. We begin by extracting N uniformly spaced vertical depth profiles of one pixel from this image starting from 0 X-axis position (see Fig. 4a). Let $p_{i,j}$ be image pixels where $i \in [1, .., w]$, $j \in [1, .., h]$, w and h are the image width and height. Depth profile at horizontal position k is defined as $dp_{k,j} = p_{i=k,j}$. Each depth profile is differentiated along itself and inverted as it is presented in (2) where Δ_j means distance between $j-1$ and j depth points (Fig. 4a), H_{max} is the maximum depth in cm.

$$dp^{diff}_{k,j} = \frac{H_{max}}{\Delta_j} - \frac{dp_{k,j} - dp_{k,j-1}}{\Delta_j} \tag{2}$$

Afterwards, we filter outliers in the differentiated profile by eliminating points which do not have necessary number of neighbors in its vicinity. Finally, we apply the DBSCAN clustering algorithm [7] to the pre-filtered differentiated profile and obtain clusters that correspond to different steps. Figure 4b illustrates clustering results where



(a)



(b)

Figure 4: (a) Depth profile at 160 px. (b) Differentiated, filtered and clustered depth profile at 220 px (0 corresponds to the top and 400 to the bottom of the image)

different clusters are associated with different colors. Accordingly, every cluster is analyzed and individual steps are distinguished from one another. Finally, the set of marked profiles fully represents the stair region and the lower step position (Fig. 5), we note that figure 5 visualizes the staircase in an uncommon way in order to increase contrast and comprehensibility of the depth image. In this way, we can obtain the position of the lowermost step which we choose to assign as the reference pose of the entire staircase. Moreover, the normal vector to the front lowermost step surface representing staircase orientation is calculated. Stair position on the 2D depth image is retrieved in the form of bounding box and is fused with the results of the color image stair detection algorithm. Finally, the staircase state parameters that are retained are the total number of steps, the minimum, maximum and average step height and depth.

To retain these parameters, we obtain the point cloud in the camera frame by transforming color and depth images with calibration parameters. Knowing camera orientation relatively to the main robot chassis and assuming that the robot is in the vertical position along gravity, we transform the point cloud coordinates from the camera frame to the main robot coordinate frame. The step depth is simply the mean distance between corresponding step edge point depths. The height is the mean distance between corresponding step edge points along gravity.

After obtaining two hypotheses on the presence of a staircase, one from RGB and the other from depth sensory data, the final decision is based on the surface area overlap of bounding boxes. The staircase estimation process continues by acquiring uniformly distributed observations in 2D space, determined by the robot's location.

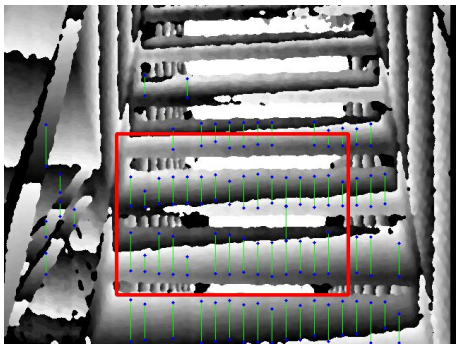


Figure 5: Detected stair in depth image; the red box is the stair related area

5 Future work

5.1 Navigation

We distinguish two mobility modes, namely, floor-based and staircase-based modes. The former is used only in the case of path execution on 2D surfaces. For this case, we will employ ideas borrowed from [9] where commands are continuously computed for track motors using the robot's kinematic model and flippers are raised up in order to decrease friction. The latter mode is applied when the robot faces the staircase. Switching between modes happens when the robot decides to traverse a staircase and approaches it at a certain distance (Fig. 6, transition state 1). When the mode changes to staircase traversal, motor commands start being computed by the trained algorithm (Fig. 6, transition states 2-4) and the robot returns to the flat mode once the traversal is accomplished (Fig. 6, transition state 5).

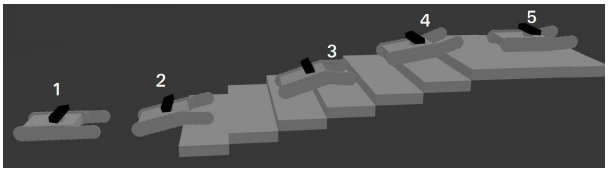


Figure 6: Robot stair climbing

We consider that the robot may traverse stairs up and down. When the robot finishes climbing, the robot saves its position on the next floor as the staircase position. Staircase descending is more challenging. Knowing staircase position after climbing up, the robot will keep in mind its location. Once the wished staircase position to descend is approached, the robot turns to the stair-traversal mode, but with another behaviour that should be learned as well as . If the robot wishes to find an unknown stair to descend, we will detect it as in [13] where authors use optical flow to detect descending stairs and extract the leading stair edge. Having a predominantly flat surface, we can rely on state-of-the-art 2D Simultaneous Localization and Mapping (SLAM) such as the one proposed in [11]. Stairs will

be localized by the perception module with their position associated to a certain location on the map as stated earlier. There are two navigable types of surfaces, namely, floor and stairs. We consider 2D SLAM, for the case when the robot resides on a flat surface. Staircases will serve as bridges shifting from one floor to another where the final staircase position is hence continuously estimated by using a Kalman filter. Eventually, this implies that we expect multiple 2D floor maps that are interconnected by staircases. The question arises about how the robot localizes itself while traversing a staircase. To address this question we consider using triangulation with odometry whereby regular patterns such as corner points could be used as landmarks whose position is calculated from 3D input depth data. To compensate for landmark disappearance along traversal, the localization of the robot will further rely on odometric data obtained either from the IMU or the track encoders that should be sufficiently robust because of small size of stairs and their regular shape. Once traversed, the robot triggers a new 2D SLAM operation corresponding to the level wherein the robot is situated.

To perform 2D path planning while navigating in floor-mode, D* Lite algorithm presented in [18] was designed to create optimal paths in dynamic scenarios by focusing replanning on the affected area. It is an incremental heuristic search algorithm based on the Lifelong Planning A*. It allows to plan paths on the occuppancy grid maps in reasonbale time and to avoid dynamic obstacles recalculating only influenced parts of the map graph. Like A* and LPA*, D* Lite uses a heuristic, that limits the cost of the path from a given node to the start. We refer the reader to the original paper [18] for more details and its pseudocode. Two flat maps connect each other by a node representing stair, hence, path planning problem is conveniently addressed in a lower-dimensional state space. The robot will have a module that tracks stair position and, if it decides to traverse stairs, the robot switches between flat surface movement mode and staircase movement mode.

5.2 Reinforcement learning-based control

Algorithms from section 3.2 may be used in application to different robotic problems. They were mostly tested in activities such as hockey [20], stair traversal [26] or table tennis [28, 19] where they showed results for the safety constraints and learning time. Thus, this renders this algorithm family particularly attractive in the case of stair traversal in order to accommodate the need for safe actions and learning in a handful of trials.

The future work comprises elaboration of Constrained REPS algorithm proposed in [26]. First, it demands a reasonably small amount of needed iterations to learn how to traverse a previously unknown obstacle. Second, its cautious physics-based simulator is important for applications in assistive robotics where we wish to avoid any damage of the environment and the robot. Lastly, being based on the Contextual REPS, the Constrained REPS enables the robot to use the context which corresponds to different, even not seen before, situations that would be new stair geometries. In our case, the context relating to staircase configurations will be employed and safety constraints will be implemented using a safety simulator, finally, results will be compared.

The GPREPS learns the robot dynamics finally decreasing the total amount of required real robot trials, however, this dynamics corresponds only to one robot configuration. The Constrained REPS does not use the Gaussian Process because, as authors say, it is difficult to provide guarantees of the safety of the generated trajectories based on data-driven models without any prior knowledge about underlying physics. The cautious physics-based simulator should be used to overcome this limitation. Nevertheless, it was shown that model-based approaches are more promising for learning in a handful of trials [2]. The GPREPS and the Constrained REPS have not yet been compared and the latter was not applied to different contexts which motivates our interest for further investigation. It should be possible to fuse these two algorithms to exploit the advantages of each of them and seek an implementation with more complex constraints.

Wishing to decrease the need of human expertise, we also desire to transfer the learned robot dynamics to different robot configurations of the spatial shape. It appears that knowledge about the robot's configuration could be added into the context vector or, possibly, it should learn its geometry using sensorimotor invariants as proposed in [21].

The next step concerns robotic arm integration. It will influence robot dynamics by translation of the mass center, therefore it has to be taken into account in the context vector and reward function. At the same time, the robot could move its arm improving capability of staircase traversal with respect to safety constraints, such movements should be also received like output of the lower-level policy. The last intention is to enable the robot to execute learning during operation in the real world in a handful of following the example of [3].

6 Conclusion

This paper presents the problem addressed in the context of a doctoral thesis, associated preliminary developments and workplan along with whose goal is to provide a tracked robot capable of safe navigation in 3D indoor environments. Contemporary robotic solutions of tracked robot stair climbing were presented and proposals for advancing the state-of-the-art were considered. Two main perspectives can be distinguished for addressing this problem. Relatedly, most contemporary approaches rely on over-customized solutions with poor generalization to different contexts. As an alternative, we advocate a reinforcement-learning, policy search based approach to reduce the amount of expert supervision and allow more flexibility to varying conditions. Subsequent work concerns development and evaluation of these modules in real-

istic conditions, system integration and scaling up learning complexity as the result of inclusion of the robotic arm.

7 Acknowledgments

The present work is performed in the context of the project M@D (chaire Maintien@Domicile) and the project VITAAL (Vaincre l'Isolement par les TIC pour l'Ambient Assisted Living) and is financed by Brest Metropole, the region of Brittany (France) and the European Regional Fund (FEDER).

References

[1] M. Brunner, T. Fiolka, D. Schulz, and C. M. Schlick. Design and comparative evaluation of an iterative contact point estimation method for static stability estimation of mobile actively reconfigurable robots. *Robotics and Autonomous Systems*, 63:89 – 107, 2015.

[2] K. I. Chatzilygeroudis, V. Vassiliades, F. Stulp, S. Calinon, and J.-B. Mouret. A survey on policy search algorithms for learning robot controllers in a handful of trials. *CoRR*, abs/1807.02303, 2018.

[3] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *CoRR*, abs/1805.12114, 2018.

[4] H. Chung, C. Hou, Y. Chen, and C. Chao. An intelligent service robot for transporting object. In *2013 IEEE Int. Symp. on Industrial Electronics*, pages 1–6.

[5] Y. Cong, X. Li, J. Liu, and Y. Tang. A stairway detection algorithm based on vision for ugv stair climbing. In *2008 IEEE Int. Conf. on Networking, Sensing and Control*.

[6] K. Doelling, J. Shin, and D. O. Popa. Service robotics for the home: A state of the art review. In *Int. Conf. on Pervasive Technologies Related to Assistive Environments*, New York, USA, 2014. ACM.

[7] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[8] G. Genki, K. Nagatani, T. Hashimot, and K. Fujino. Slip-compensated odometry for tracked vehicle on loose and weak slope. *ROBOMECH Journal*, 4(1):27, Nov 2017.

[9] M. Gianni, F. Ferri, M. Menna, and F. Pirri. Adaptive robust three-dimensional trajectory tracking for actively articulated tracked vehicles. *Journal of Field Robotics*, 33(7):901–930, 2016.

[10] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[11] G. Grisetti, G. D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi. Fast and accurate slam with rao-blackwellized particle filters. *Robotics and Autonomous Systems*, 55:30–38, 2007.

[12] D. M. Helmick, S. I. Roumeliotis, M. C. McHenry, and L. Matthies. Multi-sensor, high speed autonomous stair climbing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002.

[13] J. A. Hesch, G. L. Mariottini, and S. I. Roumeliotis. Descending-stair detection, approach, and traversal with an autonomous tracked vehicle. In *2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

[14] S. Hirose, H. Tsukagoshi, and K. Yoneda. Normalized energy stability margin and its contour of walking vehicles on rough terrain. In *ICRA. IEEE Int. Conf. on Robotics and Automation*, 2001.

[15] J. A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, pages 1547–1554, Cambridge, MA, USA, 2002. MIT Press.

[16] J.Kober and J. Peters. Policy search for motor primitives in robotics. *Mach. Learn.*, 84(1-2):171–203, July 2011.

[17] J. Kober and J. Peters. *Reinforcement Learning in Robotics: A Survey*, pages 9–67. Springer International Publishing, Cham, 2014.

[18] S. Koenig and M. Likhachev. D*lite. In *Eighteenth National Conference on Artificial Intelligence*, pages 476–483, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[19] A. Kupcsik, M. P. Deisenroth, J. Peters, A. P. Loh, P. Vadakkepat, and G. Neumann. Model-based contextual policy search for data-efficient generalization of robot skills. *Artificial Intelligence*, 247:415 – 439, 2017. Special Issue on AI and Robotics.

[20] A. G. Kupcsik, M. P. Deisenroth, J. Peters, and G. Neumann. Data-efficient generalization of robot skills with contextual policy search. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[21] A. Laflaquière, J. K. O'Regan, S. Argentieri, B. Gas, and A. V. Terekhov. Learning agent's spatial configuration from sensorimotor invariants. *CoRR*, abs/1810.01872, 2018.

[22] M. Menna, M. Gianni, F. Ferri, and F. Pirri. Real-time autonomous 3d navigation for tracked vehicles in rescue environments. In *2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

[23] K. Nagatani, A. Yamasaki, K. Yoshida, T. Yoshida, and . Koyanagi. Semi-autonomous traversal on uneven terrain for a tracked vehicle using autonomous control of active flippers. In *2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

[24] G. Neumann, W. Maass, and J. Peters. Learning complex motions by sequencing simpler motion templates. volume 382, page 95, 01 2009.

[25] G. Paolo, L. Tai, and M. Liu. Towards continuous control of flippers for a multi-terrain robot using deep reinforcement learning. *CoRR*, abs/1709.08430, 2017.

[26] M. Pecka, S. Valansky, K. Zimmermann, and T. Svoboda. Autonomous flipper control with safety constraints. In *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

[27] J. Peters, K. Muelling, and Y. Altun. Relative entropy policy search. In *AAAI*, 2010.

[28] J. Peters, K. Mülling, , and Y. Altun. Reinforcement learning by relative entropy policy search. *30th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering (MaxEnt 2010)*, 30:69, 2010.

[29] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[30] S. Soichiro, H. Satoshi, and O. Masayuki. Remote control system of disaster response robot with passive sub-crawlers considering falling down avoidance. *ROBOMECH Journal*, 1(1):20, Nov 2014.

[31] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[32] L. Zhang, K. Thurow, H. Liu, J. Huang, N. Stoll, and S. Junginger. Multi-floor laboratory transportation technologies based on intelligent mobile robots. *Transportation Safety and Environment*, 2019.

Bandits Manchots Survivants

Filipo Studzinski Perotto[1] Mathieu Bourgais[1] Bruno Castro Silva[2] Laurent Vercouter[1]

[1] Normandie Université, UNIROUEN, UNIHAVRE, INSA, 76000 Rouen, France
[2] Universidade Federal do Rio Grande do Sul, Instituto de Informática, Porto Alegre, Brésil

filipo.perotto@litislab.fr, {mathieu.bourgais, laurent.vercouter}@insa-rouen.fr, bsilva@inf.ufrgs.br

Résumé

Dans cet article, nous formalisons une classe particulière de problèmes appelée bandits manchots survivants (S-MAB), qui constitue une version modifiée des bandits manchots avec budget (B-MAB) dans laquelle un véritable risque de ruine doit être pris en compte, ce qui le rapproche des bandits manchots averses au risque (RA-MAB) et du problème classique de la ruine du joueur (GR). Dans un S-MAB, une action peut entraîner des récompenses positives ou négatives. L'agent dispose d'un budget initial qui évolue dans le temps en fonction des récompenses reçues. L'objectif est de trouver un bon compromis entre exploration-exploitation-sécurité, maximisant les récompenses tout en minimisant le risque de ruine (c'est-à-dire, la probabilité de se retrouver avec un budget négatif). Une telle modification dans l'énoncé du problème change la façon de l'aborder et demande des algorithmes adaptés. Deux nouveaux algorithmes conçus pour résoudre les S-MAB sont proposés. L'intérêt potentiel de ces méthodes est appuyé par des résultats expérimentaux.

Mots Clef

Bandits Manchots avec Budget, Théorie de la Ruine, Prise de Décision avec Risque.

Abstract

In this paper we formalize a particular class of problems called survival multiarmed bandits (S-MAB), which constitutes a modified version of budgeted multiarmed bandits (B-MAB) where a true risk of ruin must be considered, bringing it closer to risk-averse multiarmed bandits (RA-MAB) and to the gambler's ruin game (GR). In an S-MAB, pulling an arm can result in both positive and negative rewards. The agent has an initial budget that evolves in time with the received rewards. The goal is finding a good exploration-exploitation-safety trade-off, maximizing rewards while minimizing the possibility of getting ruined (i.e. the probability of hitting a negative budget). Such modification in the problem statement changes the way to approach it and asks for adapted algorithms. Two new algorithms designed to solve S-MABs are proposed. The potential interest of such methods is supported by experimental results.

Keywords

Budgeted Multiarmed Bandits, Ruin Theory, Risk-Averse Decision Making.

1 Introduction

Le défi de trouver un bon équilibre entre *exploration* (i.e. agir pour apprendre de nouvelles choses) et *exploitation* (i.e. agir de manière optimale en fonction de ce qui est déjà connu) constitue un problème largement étudié en *apprentissage par renforcement* [28, 31]. Les *bandits manchots* (*multiarmed bandits* - MAB) fournissent un cadre simple et adapté pour modéliser la *prise de décision séquentielle* face à ce dilemme entre exploration et exploitation [33]. Il s'agit d'une métaphore des machines à sous dans les casinos.
Un MAB est typiquement représenté par un agent qui interagit avec un processus à chaque pas de temps t par le choix d'une action i à effectuer parmi k actions possibles, puis en recevant une récompense r correspondante. Étant donné que l'information complète sur les fonctions de récompense n'est pas disponible, l'agent doit estimer leurs paramètres en les échantillonnant, c'est-à-dire en tirant les bras des différentes machines et en observant les récompenses reçues. Résoudre un MAB signifie maximiser les récompenses que l'agent espère accumuler au fil du temps, en trouvant le meilleur compromis entre *exploration* (tester les différentes actions) et *exploitation* (choisir la meilleure action estimée). Différentes méthodes et garanties ont été proposées dans la littérature en fonction des informations disponibles et des hypothèses assumées concernant les fonctions de récompense [7, 19]. Cependant, la plupart des recherches existantes sur les bandits ignorent toute notion de coût et la possible finitude des ressources pour tirer les bras [13]. Pourtant, en pratique, prendre des actions est souvent coûteux. Autrement dit, dans des situations réelles, il est normal d'espérer qu'un agent dépense des ressources pour pouvoir agir.
Dans la littérature, une variante du problème appelée *bandit manchot avec budget* (B-MAB) [32] associe une fonction supplémentaire dite de *coût* à chaque bras, indépendante de la fonction de récompense. Après avoir réalisé une action (i.e. tiré un bras), l'agent reçoit une récompense aléatoire, mais paie aussi un coût aléatoire. Les coûts demandés pour

réaliser les actions sont retirés d'un *budget* initial b_0. Le total des dépenses possibles est alors limité par ce budget. Lorsque le budget est fini, le jeu est terminé. Ainsi, dans ce contexte, étant donné qu'il existe toujours un coût associé aux actions, le budget limite l'horizon temporel du processus.

Dans cet article, nous nous intéressons à une classe de problèmes que nous appelons *bandits manchots survivants* (S-MAB), qui constitue une version modifiée de B-MAB où un véritable *risque de ruine* doit être pris en compte. Dans un S-MAB, le budget initial b_0 évolue dans le temps en fonction des récompenses reçues, de sorte que $b_{t+1} = b_t + r_t$. Au contraire des B-MAB, il n'y a pas de fonction de coût séparée. Les gains et les coûts liés à chaque bras sont donnés dans une même « monnaie commune », conjuguée dans une fonction de récompense unique. Ainsi, la récompense reçue peut tout aussi bien être positive que négative, et affecte la même ressource : le budget. Le budget correspond, de cette manière, au « capital » ou à la « fortune » de l'agent, et le coût pour tirer un bras est payé avec les récompenses cumulées. Par conséquent, résoudre un S-MAB signifie estimer la meilleure action afin de maximiser la fortune au fil du temps, mais tout en gérant l'évolution du budget, qui peut diminuer avec le coût des actions exploratoires, afin d'éviter la ruine.

Pour gérer ce risque de ruine, l'agent doit prendre en compte l'incertitude des récompenses attendues. L'origine de cette incertitude peut être propre aux distributions dont la variance est trop élevée, mais aussi conséquence d'un manque de confiance dans les estimations. Des préoccupations similaires existent dans la littérature dans le cadre appelé MAB *averse au risque* (RA-MAB) [27, 17], où l'agent doit identifier (et éviter) les actions dont le retour est moins prévisible, sans se soucier pourtant de la ruine puisqu'aucune notion de budget n'est considérée.

Si le budget est pris comme une *variable essentielle* (e.g. comme le niveau d'énergie de l'agent), un S-MAB devient vraiment un problème de survie, et peut être associé à la notion cybernétique de *homeostasis* [3, 8, 12] : la condition d'un système lorsqu'il est capable de maintenir ses variables essentielles dans les limites nécessaires pour viabiliser son existence malgré les perturbations externes. Ainsi, dans un S-MAB, l'agent veut apprendre l'action optimale, tout en restant en vie (i.e. préservant un budget positif pendant tout le temps de son existence). Le dilemme exploration-exploitation devient plus complexe lorsque ce risque de ruine est pris en compte.

Le problème MAB défini comme un S-MAB se rapproche plus de certains problèmes généraux du monde réel : dans ce jeu d'équilibre entre exploration et exploitation, en essayant de maximiser les récompenses reçues, les joueurs dans un casino ont un budget et doivent éviter la ruine, les investisseurs en bourse ont des fonds et doivent éviter la faillite, les organismes naturels ont des variables essentielles, inhérentes à leur structure, et doivent éviter la mort.

La contribution de cet article est la définition d'un scénario nouveau et original appelé *survival MAB* (S-MAB), en le situant entre d'autres versions du problème tels que B-MAB et RA-MAB, mais également le jeu classique de la *ruine du joueur* (*gambler's ruin* - GR). Une heuristique, appelée *seuil de sécurité*, qui peut être couplée à tout algorithme MAB classique est ensuite proposée pour les adapter aux S-MAB. Une version modifiée de l'algorithme UCB standard, appelée *joueur positif*, est également introduite. Les algorithmes sont comparés expérimentalement et les méthodes proposées présentent de bons résultats. Des mesures de performance spécifiques aux S-MAB sont aussi proposées.

Le reste de l'article est organisé comme suit : la section 2 révise la littérature concernant MAB, en se concentrant sur les méthodes standard pour équilibrer exploration et exploitation ; la section 3 présente des travaux connexes concernant B-MAB, RA-MAB et GR, indiquant pourquoi S-MAB ne peut être réduit à aucun de ces modèles ; la section 4 énonce formellement le problème de recherche et introduit deux méthodes spécialement adaptées : *seuil de sécurité* et *joueur positif* ; la section 5 présente les résultats expérimentaux ; la section 6 discute des conclusions et des perspectives.

2 Bandits Manchots

Dans la définition standard du MAB, il n'y a pas de budget à considérer, ni de risques à prendre en compte. Un tel MAB peut être formellement défini comme :

$$\mathcal{M} = \begin{cases} I = \{1, ..., k\} & \text{l'ensemble d'actions possibles} \\ F = \{f_1, ..., f_k\} & \text{les fonctions de récompense} \end{cases} \tag{1}$$

Le processus évolue discrètement dans le temps. Soit $a_t = i$ l'action choisie et r_t la récompense immédiate reçue à l'instant t, tiré de la distribution f_i. Pour chaque bras, les récompenses sont indépendantes mais identiquement distribuées. Une fonction de distribution stationnaire $f_i \in F$ définit la probabilité d'avoir la récompense $r_t \in \mathbb{R}$ après avoir choisi le bras i au moment t :

$$r_t \sim f_i \mid a_t = i \tag{2}$$

Étant donné que $\mu_i = \mathbb{E}[f_i]$ est la moyenne de la distribution f_i, l'action optimale i^* est celle qui présente la récompense moyenne la plus élevée μ^* parmi tous les bras :

$$\mu^* = \max_{i \in I}[\mu_i] \qquad i^* = \arg\max_{i \in I}[\mu_i] \tag{3}$$

Soit $A_t = \{a_1, ..., a_t\}$ la séquence des bras tirés et $R_t = \{r_1, ..., r_t\}$ la séquence des récompenses reçues jusqu'au temps t. Un algorithme MAB doit proposer la prochaine action à choisir a_{t+1} en fonction de l'historique d'observations $\{A_t, R_t\}$. La meilleure stratégie possible (si toutes les fonctions f_i auraient pus être connues à l'avance) consiste à toujours tirer i^*. Dans sa version standard (stochastique et non-paramétrique) [5], résoudre un MAB signifie trouver

une stratégie qui puisse minimiser le regret (i.e. la différence cumulée entre les récompenses attendues en suivant la stratégie choisie et les récompenses qui auraient pu être obtenues en tirant toujours le meilleur bras). Réduire le regret permet de maximiser la somme espérée de récompenses. Une solution optimale doit garantir que $\lambda_h \to 0$ quand $h \to \infty$. Considérant un horizon temporel donné h (qui peut être infini), le *regret cumulatif* λ et le *regret cumulatif théorique* $\hat{\lambda}$ sont définis comme suit :

$$\lambda_h = \sum_{t=1}^{h} [\mu^* - r_t] \qquad \hat{\lambda}_h = \sum_{t=1}^{h} [\mu^* - \mu_{a_t}] \tag{4}$$

La *vraie moyenne* μ_i de la distribution f_i peut être estimée en temps h par la *moyenne empirique* $\hat{\mu}_{i,h}$, comme suit :

$$\hat{\mu}_{i,h} = \frac{1}{n_{i,h}} \sum_{t=1}^{h} r_{i,t} \text{ où } r_{i,t} = \begin{cases} r_t & \text{if } a_t = i \\ 0 & \text{if } a_t \neq i \end{cases} \tag{5}$$

et $n_{i,h}$ est le nombre de fois où le bras i a été tiré dans l'horizon temporel h donné. Une limite inférieure asymptotique (i.e. le plus petit regret possible) correspondant à $\Omega(\ln t)$ a été prouvée pour le MAB stochastique [23, 1]. On dit qu'un algorithme résout efficacement le problème si sa limite supérieure (i.e. dans le pire cas) se rapproche du même ordre de regret, $O(\ln t)$.

2.1 Exploration Non-Dirigée

Les méthodes dites d'*exploration non-dirigée* équilibrent exploration et exploitation par l'ajout du non-déterminisme (de l'aléatoire) dans leur choix d'actions à entreprendre [28]. ε-greedy appartient à cette classe de méthodes. La stratégie consiste à choisir, à chaque instant, soit une action aléatoire avec probabilité ε, soit l'action estimée optimale $\hat{\imath}_t^*$ avec probabilité $1-\varepsilon$, où $\hat{\imath}_t^*$ est, en temps t, le bras ayant la plus grande récompense moyenne estimée $\hat{\mu}_t^* = \max_{i \in I}[\hat{\mu}_{i,t}]$.
De cette manière, à chaque pas de temps, l'action choisie par ε-greedy est :

$$a_{t+1} = \begin{cases} \hat{\imath}_t^* & \text{avec probabilité } (1-\varepsilon) + \varepsilon/k \\ j \in I \mid j \neq \hat{\imath}_t^* & \text{avec probabilité } \varepsilon/k \end{cases} \tag{6}$$

Une telle stratégie naïve est cependant sous-optimale. Le taux d'exploration restant constant pendant tout le processus provoque un regret cumulatif linéaire. Une variation de ε-greedy appelée GreedyMix réduit progressivement ε en utilisant un facteur décroissant $\ln(t)/t$, ce qui permet un regret logarithmique. C'est de toute façon moins efficace que les méthodes d'exploration dirigée [10]. D'autres méthodes d'exploration non-dirigée sont présentées dans [28].

2.2 Exploration Dirigée

L'approche alternative pour résoudre le dilemme entre exploration et exploitation est d'être *optimiste face à l'incertitude*. Cela consiste à favoriser le choix des actions insuffisamment expérimentées [26]. Selon la *loi des grands nombres*, plus une action est tentée, plus la moyenne estimée de la récompense se rapproche de sa valeur réelle. En conséquence, ces algorithmes ont tendance à promouvoir l'exploration dans les premières étapes d'exécution et à passer progressivement à l'exploitation.
Une exploration intelligente peut être réalisée en contrôlant statistiquement la confiance sur les moyennes estimées. Basé sur ce principe de la *limite supérieure de la confiance* (*upper confidence bound* - UCB), différentes méthodes conçues pour le MAB stochastique et paramétrique (quand la famille à laquelle appartient la distribution sous-jacente est donnée) se sont avérées efficaces pour atteindre un regret logarithmique de manière asymptotique [23, 1]. Pour la version non-paramétrique (standard), il a été prouvé que plusieurs variantes de la méthode UCB permettent d'obtenir un regret logarithmique avec la seule hypothèse que les récompenses aient un support connu fini $\mathcal{D} = [r_{min}, r_{max}]$. La méthode de référence UCB1 [5], conçue pour $\mathcal{D} = [0, 1]$, choisit simplement, à chaque instant t, l'action qui porte la plus grande valeur d'un indice v calculé, comme suit :

$$a_{t+1} = \arg\max_{i \in I}[v_{i,t}]$$

où

$$v_{i,t} = \begin{cases} \hat{\mu}_{i,t} + \sqrt{\frac{2 \ln t}{n_{i,t}}} & \text{if } n_{i,t} > 0 \\ +\infty & \text{if } n_{i,t} = 0 \end{cases} \tag{7}$$

Ainsi, UCB1 exécute chaque action une fois (de $t = 1$ à k), et ensuite ($t > k$) passe à choisir l'action maximisant la somme de sa moyenne estimée avec l'erreur d'estimation maximale, donnée par une limite supérieure de confiance qui augmente progressivement avec le temps.
Parmi les méthodes d'exploration dirigée qui ont suivi UCB1 se trouvent UCB2 et UCB-tuned [5], UCB-V et MOSS [4], KL-UCL [25, 18], Bayes-UCB [21], et KL-UCB- Switch [19]. Ces méthodes s'appuient sur des astuces statistiques plus fines qui améliorent la précision de la limite supérieure de confiance.

3 Budget, Risque et Ruine

Deux versions alternatives du problème MAB sont pertinentes pour la construction du *MAB survivant* : *MAB avec budget* (B-MAB) et *MAB averse au risque* (RA-MAB). Le jeu mathématique classique de la *ruine du joueur* (GR) est un autre problème connexe important. Nous présentons dans cette section l'état de l'art concernant ces problèmes, ainsi que les raisons pour lesquelles S-MAB ne peut être réduit à aucun d'eux.

3.1 MAB avec Budget

Dans un MAB typique (non-budgétisé), l'agent souhaite maximiser les récompenses reçues au fil du temps tout en échantillonnant les actions possibles. Différemment, dans un B-MAB, le joueur reçoit une récompense mais doit payer

un coût après avoir tiré un bras [32]. Deux fonctions indépendantes sont associées à chaque action : *coût* et *récompense*. Les coûts pour tirer les bras sont pris d'un *budget* initial donné, ce qui limite l'horizon temporel du processus. Plus aucun bras ne peut être tiré après l'épuisement du budget, i.e. lorsque le budget est terminé, le jeu est terminé.

Certains travaux, tels que [29], étudient le cas où le coût des actions est fixe et devient connu après que le bras a été tiré une fois. D'autres, comme [13, 32], considèrent le coût comme une variable aléatoire plutôt qu'une constante. Dans cette version où le coût est variable, l'agent doit explorer les actions pour estimer non seulement la fonction de récompense d'un bras, mais également sa fonction de coût [33]. L'algorithme `UCB-BV` [13] échantillonne les coûts et les récompenses issus de distributions inconnues et montre une limite supérieure du regret de l'ordre $O(\ln b_0)$. Dans une version similaire appelée *bandits à sac à dos* (*bandits with knapsacks*), chaque bras est associé à d différentes fonctions de coût qui consomment de manière stochastique d différents budgets.

La stratégie pour évaluer les actions tenant compte de ces deux fonctions différentes (récompenses f_i et coûts c_i) est basée sur un coefficient appelé *densité* qui divise la récompense attendue $\hat{\mu}_{r_i}$ d'un bras i donné par son coût attendu $\hat{\mu}_{c_i}$. Le bras choisi est celui qui présente la meilleure densité parmi les bras dont le coût espéré rentre dans le budget restant :

$$v_{i,t} = \begin{cases} \frac{\hat{\mu}_{r_i}}{\hat{\mu}_{c_i}} & \text{if } \hat{\mu}_{c_i} \leq b_t \\ 0 & \text{if } \hat{\mu}_{c_i} > b_t \end{cases} \tag{8}$$

Il n'y a pas une action optimale absolue, puisque le choix dépend du budget restant [6]. Il correspond au bras « payable » avec le meilleur ratio récompense/coût.

D'autres travaux récents abordent des variations du problème budgétisé en considérant des scenarios spécifiques [33, 2, 11, 22]. Une autre version du problème budgétisé, trouvée dans [4, 20], considère que tirer un bras a un coût, mais que le budget n'est imposé qu'à une phase préliminaire d'exploration. La phase ultérieure d'exploitation n'est pas associée à des coûts ni contraintes par un budget. La question est de savoir comment dépenser le budget pour bien identifier le meilleur bras, ce qui fait de lui un « problème d'exploration pure ».

En fait, tous ces algorithmes B-MAB calculent l'utilité des bras en fonction du rapport récompense/coût (eq. 8), qui n'existe pas dans S-MAB simplement parce qu'il n'y a pas une fonction de coût séparée de la fonction de récompense. Par conséquent, aucun de ces algorithmes B-MAB ne peut être appliqué aux problèmes S-MAB. En outre, un B-MAB est un processus toujours limité dans le temps, car le budget diminue après chaque action et sera tôt ou tard épuisé. En revanche, un S-MAB constitue un processus potentiellement infini dans le temps, dans la mesure où les récompenses rechargent le budget.

3.2 MAB Averse au Risque

Alternativement, dans un RA-MAB, l'agent doit tenir compte de la variabilité espérée des récompenses attendues afin d'identifier (et d'éviter) les actions instables (alors considérées comme risquées), mais sans se soucier de la ruine puisqu'aucune notion de budget n'est considérée.

Dans certains environnements, il peut être souhaitable de suivre une politique qui réalise un compromis entre exploration, exploitation et stabilité ou prévisibilité [17]. Dans la littérature concernant RA-MAB, la notion de « bras à risque » est généralement associée aux actions qui ont présenté une forte variabilité sur les récompenses observées. En ce sens, le compromis risque-récompense peut être abordé en utilisant une métrique du type moyenne/variance $\hat{w}$ [27, 30].

$$\hat{w}_{i,t} = \varrho\hat{\mu}_i - (1-\varrho)\hat{\sigma}_i^2 \tag{9}$$

où le paramètre $\{\varrho \in \mathbb{R} \mid 0 \leq \varrho \leq 1\}$ représente la tolérance au risque, et $\hat{\mu}_i$ et $\hat{\sigma}_i^2$ sont, respectivement, l'estimation de la récompense moyenne et de la variance du bras i. Lorsque $\varrho \to 0$, le meilleur bras devient celui avec la plus petite dispersion. Lorsque $\varrho \to 1$, le meilleur bras devient celui avec la plus haute récompense moyenne. L'algorithme `MV-LCB` [27] est une extension de l'algorithme `UCB` qui utilise cette métrique moyenne/variance.

Un autre critère pour évaluer le risque associé à un bras est sa *valeur conditionnelle à risque*, une mesure basée sur les quartiles (ou queues inférieures) des distributions de récompense. Soit f_i la distribution du bras i, et Φ_i sa fonction de distribution cumulative. Dans cette proposition, l'utilité d'un bras n'est pas basée sur sa récompense moyenne estimée, mais sur $\Phi_i(\alpha)$, où α est le paramètre indiquant le risque tolérable. C'est l'intuition derrière `MaRaB`, un autre algorithme de type UCB proposé dans [17]. Lorsque $\alpha \to 0$, cette méthode coïncide avec l'algorithme `MIN`, qui utilise la récompense minimale observée comme valeur d'utilité du bras. `MIN` sélectionne à chaque pas de temps l'action avec la valeur minimale empirique maximale. [24, 9] présentent des mesures alternatives, aussi basées sur les quartiles, utilisées dans des méthodes similaires.

S-MAB ne peut pas être réduit à RA-MAB parce que, dans RA-MAB, il n'y a aucune notion de ruine. Par conséquent, un bras récompensé négativement mais stable peut être préféré à un bras positif. Une telle situation est clairement démontrée dans les résultats expérimentaux (sec. 5). En effet, choisir un bras parce que sa variance est inférieure à celle du bras dont la moyenne est la plus élevée entraîne généralement un regret linéaire de récompense.

3.3 Ruine du Joueur

Dans le jeu mathématique classique appelé *ruine du joueur* [16], à chaque pas de temps t, l'agent peut augmenter sa fortune b de 1 avec probabilité p, ou perdre 1 avec probabilité $q = 1-p$. La fortune totale dans le jeu équivaut à g. L'agent commence avec un budget initial $\{b_0 \in \mathbb{N} \mid 0 < b_0 < g\}$. Le jeu s'arrête lorsque $b_t = 0$ ou $b_t = g$ est atteint pour

la première fois (soit le joueur est ruiné, soit il a tout gagné). Il n'y a ni décision à prendre (puisqu'il n'y a qu'une seule action possible), ni paramètre à estimer (puisque p est donné).
La *ruine du joueur* peut être joué contre un adversaire infiniment riche [15], ce qui rapproche le jeu des pratiques standard du MAB : aucune valeur de fortune n'est particulièrement à atteindre, et l'objectif est de maximiser les récompenses en considérant un horizon temporel potentiellement infini. Dans ce cas, la probabilité d'être ruiné $P^\dagger$ et le temps de survie estimé $\mathbb{E}(\tau)$ (la durée du jeu espérée avant de faire faillite) sont donnés par :

$$P^\dagger = \begin{cases} (q/p)^b & \text{if } p > q \\ 1 & \text{if } p \leq q \end{cases} \tag{10}$$

$$\mathbb{E}(\tau) = \begin{cases} \infty & \text{if } p \geq q \\ \frac{b}{q-p} & \text{if } p < q \end{cases} \tag{11}$$

4 MAB Survivant

Un S-MAB est une extension du problème MAB standard, où, outre l'ensemble des bras et ses fonctions de récompense respectives, l'agent dispose d'un budget initial $\{b_0 \in \mathbb{R} \mid b_0 \geq 0\}$. Il peut être formellement décrit par $\mathcal{M} = \{I, F, b_0\}$, en élargissant la définition proposée dans l'eq. 1. Dans un S-MAB, les coûts des actions sont pris sur le même budget vers lequel les récompenses reçues sont accumulées. Il n'y a pas de fonction de coût distincte. C'est la fonction de récompense qui affecte le budget, en l'augmentant ou en le diminuant de la manière suivante :

$$b_h = b_0 + \sum_{t=1}^{h} r_t \tag{12}$$

où b_h est le budget au temps h, et r_t est la récompense reçue au temps t.
Dans un S-MAB, la fonction de récompense est supportée dans l'intervalle $\mathcal{D} = [r_{min}, r_{max}]$, avec $r_{min} < 0 < r_{max}$. On suppose qu'au moins un bras présente une récompense moyenne négative et au moins un, une récompense moyenne positive :

$$\exists i, j \in I \ : \ \mu_i > 0 > \mu_j \tag{13}$$

Dans un tel scénario, en fonction du budget initial, des distributions associées aux bras, et de la stratégie choisie, et en supposant un horizon temporel infini ($h \to \infty$), l'agent peut augmenter la probabilité d'exécuter le processus indéfiniment, devenant infiniment riche (i.e. $b_\tau \to \infty, \tau \to \infty$), ou inversement, peut augmenter la probabilité de ruine, jusqu'à vraiment faire faillite (i.e. $\tau < \infty, b_\tau < 0$), avec τ le moment où l'agent est ruiné.
Dans un S-MAB, la seule minimisation du regret de récompense n'est pas suffisante pour bien évaluer la qualité d'un algorithme ; des stratégies qui auraient un très petit regret asymptotique peuvent souvent conduire à la ruine. En fait, réduire la probabilité de ruine peut impliquer un regret croissant. Ainsi, un critère supplémentaire doit également être pris en compte pour optimiser la survie.
D'un point de vue théorique, la résolution d'un S-MAB peut être définie comme le problème de trouver une séquence optimale d'actions $A_h^* = \{a_1, ..., a_h\}$ qui minimiserait le regret de récompense attendu tout en minimisant la probabilité d'être ruiné, dans la forme d'une optimisation multiobjectif :

$$A_h^* = \arg\min_{A_h} \begin{cases} \hat{\lambda}_h \\ P_h^\dagger \end{cases} \tag{14}$$

où $\hat{\lambda}_h$ est le regret cumulatif théorique attendu sur un horizon temporel (potentiellement infini) h et $P_h^\dagger$ est la probabilité d'être ruiné avant h lors de l'exécution de la séquence d'actions A_h. Ce genre d'optimisation produit une frontière de Pareto, où une solution optimale unique capable de minimiser les deux objectifs simultanément n'existe parfois pas. Une des manières de résoudre ce type d'optimisation est de minimiser la somme pondérée de ces deux fonctions [14].
Dans cet article, nous adoptons une approche expérimentale. Chaque expérience est répétée n fois (ou épisodes), avec un horizon temporel fini limité à h. Les différents algorithmes évalués sont comparés à travers les métriques suivantes : le regret de récompense cumulé (λ), tel que défini dans l'éq. 4, le taux de survie (χ), et le temps de survie (ξ). λ est la métrique standard du MAB. χ et ξ sont des métriques intuitives pour analyser la capacité à survivre, supposant que le processus doit s'arrêter si le budget est épuisé avant h. Pour un épisode, ces métriques sont définies comme suit :

$$\chi = \begin{cases} 1 & \text{si } (\forall t \leq h)\ b_t \geq 0 \\ 0 & \text{sinon} \end{cases} \tag{15}$$

$$\xi = \begin{cases} \tau & \text{si } (\exists \tau\ \forall t \leq \tau)\ b_\tau < 0 \leq b_t \\ h & \text{sinon} \end{cases} \tag{16}$$

4.1 Seuil de Sécurité Paramétré

Puisque nous souhaitons maximiser les récompenses futures attendues tout en évitant la ruine, le compromis entre sécurité et risque doit être prioritaire au compromis entre exploration et exploitation. L'utilisation d'un *seuil de sécurité*, noté ω, constitue une heuristique pour aborder ce problème. Ce seuil de sécurité est un paramètre réglé manuellement qui indique le niveau en dessous duquel le budget doit être considéré comme étant trop bas. Le principe est le suivant : lorsque le budget est inférieur à cette valeur ($b_t < \omega$), mais que la récompense moyenne estimée du meilleur bras est positive ($\hat{\mu}_t^* > 0$), alors l'algorithme devient gourmand dans l'espoir de remonter le budget à un niveau confortable. Cela fonctionne comme un voyant d'avertissement, qui incite l'algorithme à éviter l'exploration lorsque les ressources sont trop basses. Cet avertissement est ignoré si la récompense espérée du meilleur bras est négative. Dans ce cas, l'algorithme considère qu'il est préférable de poursuivre l'exploration, en quête d'une action positive.

En fait, tout algorithme MAB standard peut être encapsulé dans l'algorithme du seuil de sécurité paramétré. L'intuition est de laisser l'algorithme de base $\mathcal{A}$ choisi fonctionner normalement lorsque le budget est suffisamment élevé (c'est-à-dire supérieur au seuil de sécurité) ou lorsque les récompenses moyennes estimées des bras observés sont toutes négatives, et être gourmand dans le cas contraire.

Algorithm 1 Seuil de Sécurité Paramétré

1: **if** $b_t < \omega$ **and** $\max_i[\hat{\mu}_{i,t}] > 0$ **then**
2: $\quad a_t \leftarrow \arg\max_i[\hat{\mu}_{i,t}]$ *{gourmand}*
3: **else**
4: $\quad a_t \leftarrow$ **call** $\mathcal{A}$ *{méthode base}*
5: **end if**

Le premier inconvénient de cette méthode est la nécessité d'ajuster le paramètre ω à la main. Si $\omega \geq -r_{min}$, alors il est possible d'assurer que la prochaine action ne peut pas conduire l'agent à la ruine immédiate, indépendamment du bras choisi. Cependant, la valeur optimale de ω est difficile à déterminer car elle dépend fortement des distributions associées aux bras, et du budget initial. Dans certains scénarios, ajuster ω à une valeur trop basse peut être insuffisant pour éviter la ruine. En revanche, si ω est trop élevé, les performances de l'algorithme peuvent être détériorées, puisqu'il pourra devenir gourmand avant d'avoir réalisé une exploration minimal, ce qui peut également conduire à la ruine. Deuxièmement, lorsque le budget passe sous le seuil de sécurité, l'agent tente de se récupérer en jouant le bras avec la meilleure moyenne estimée. Mais un tel bras n'est pas nécessairement le moins risqué. Parfois le bras qui présente la meilleure moyenne estimée peut également avoir une variance très élevée, ou une confiance dans l'estimation trop basse.
Lorsque le budget devient dangereusement faible ($b_t < \omega$), deux situations sont possibles : soit le meilleur bras estimé $\hat{\imath}^*$ est positif ($\hat{\mu}^* > 0$), alors poursuivre une stratégie d'exploitation fera probablement augmenter le budget et permettra d'explorer en toute sécurité par la suite ; soit il est négatif ($\hat{\mu}^* \leq 0$), et l'agent fait face à un risque sérieux de ruine. Dans ce cas, même si le risque n'est pas éliminé, la meilleure option est de maintenir l'exploration, comptant sur la possibilité de découvrir un bras positif, puis, grâce à lui, d'assurer sa survie.

4.2 Joueur Positif

Le principe UCB consiste à choisir les actions sur la base d'une limite de confiance supérieure de leurs récompenses moyennes estimées. Soit α le taux d'erreur maximale accepté pour les moyennes estimées. La stratégie pour résoudre le dilemme *exploration-exploitation* dans cette famille de méthodes consiste à diminuer progressivement α, qui converge vers zéro légèrement plus vite que $1/t$. Ainsi, à chaque pas de temps, la limite de confiance supérieure se rapproche de la moyenne estimée lorsque le bras est tiré, et inversement, s'en éloigne légèrement si le bras n'est pas tiré.
Dans S-MAB, le budget et le risque de ruine doivent être pris en compte. La tendance exploratoire de l'agent doit changer non seulement en fonction du temps et des estimations, mais également en fonction du budget. Cela peut être fait en remplaçant le terme $\ln(t)$ par $\ln(b_t)$ dans l'eq. 7. Ainsi, le temps est remplacée par le budget dans l'algorithme classique `UCB1` lors du calcul de la limite de confiance supérieure. De cette manière, le taux d'erreur maximale pour la limite de confiance supérieure diminue lorsque le budget augmente, et non en fonction du temps, et converge vers zéro lorsque le budget converge vers l'infini.
Un autre problème est celui de considérer la confiance sur les estimations. Les actions dont les récompenses sont plus incertaines peuvent être considérées comme plus risquées. Une solution possible à ce problème consiste à remplacer la moyenne estimée par la probabilité estimée d'avoir une moyenne positive, en tant que mesure d'utilité d'une action, i.e. la probabilité que l'action i ait une vraie récompense moyenne positive ($\mu > 0$) étant donné que sa moyenne estimée est $\hat{\mu}$ après n observations. L'inégalité de Hoeffding (qui constitue la garantie statistique fondamentale de `UCB1`) peut être utilisée pour calculer cette probabilité, comme suit :

$$P^+ = P(\mu \geq 0 \mid \hat{\mu}, n) = \begin{cases} \theta/2 & \text{if } \hat{\mu} \leq 0 \\ 1 - \theta/2 & \text{if } \hat{\mu} > 0 \end{cases} \quad (17)$$

où

$$\theta = \exp\left(\frac{2n\hat{\mu}^2}{\Delta_r^2}\right) \quad (18)$$

et $\Delta_r = r_{max} - r_{min}$.
Une action pour laquelle l'estimation de la récompense moyenne est négative peut avoir une bonne probabilité d'être en réalité positive si elle a été peu essayé. Inversement, des actions peu expérimentées et qui présentent une récompense moyenne estimée positive peuvent avoir une probabilité importante d'être en réalité négatives. La manière de permettre l'exploration des bras à risque ne passerait pas par l'élévation graduelle du niveau de confiance (comme dans le modèle UCB standard), mais par l'exploitation des bras sûrs afin de recevoir des récompenses positives et, ainsi avoir un budget suffisamment élevé de façon à permettre le choix de ces bras à risque dans une condition de sécurité.
L'équation qui donne l'utilité d'un bras i en temps t pour la méthode du joueur positif (*positive gambler* - `PG-UCB`) est la suivante :

$$v_{i,t} = P^+_{i,t} + \sqrt{\frac{2\ln b_t}{n_{i,t}}} \quad (19)$$

Comparée à la définition classique d'UCB (eq. 7), l'utilité de base (premier terme de la somme) d'une action est sa probabilité d'avoir une moyenne réelle positive (P^+), au lieu d'être sa moyenne estimée, et la limite de confiance supérieure (deuxième terme de la somme) dépend du budget restant (b), et non pas simplement du temps.

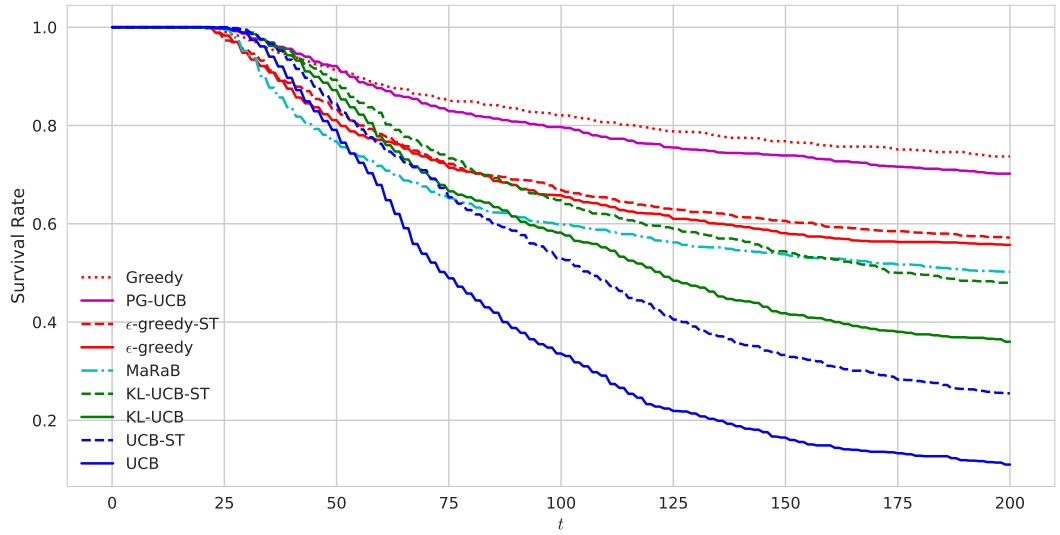


FIGURE 1 – Taux de survie : la proportion d'épisodes au cours desquels l'agent est toujours vivant jusqu'à l'horizon temporel $h = 200$ (court terme), en $n = 1000$ épisodes.

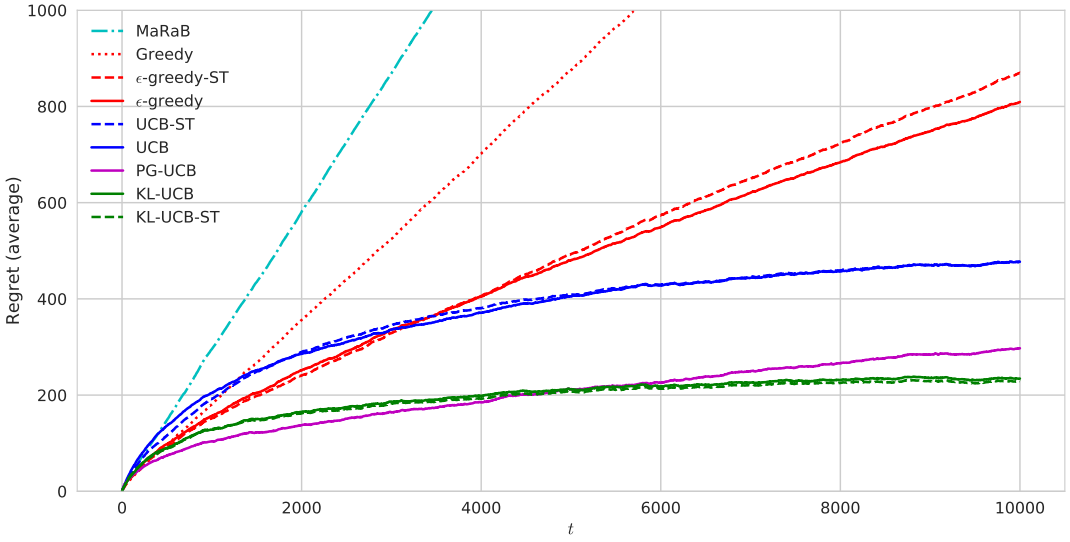


FIGURE 2 – Regret en fonction du temps, moyenne sur $n = 100$ épisodes, avec horizon temporel $h = 10000$ (long terme).

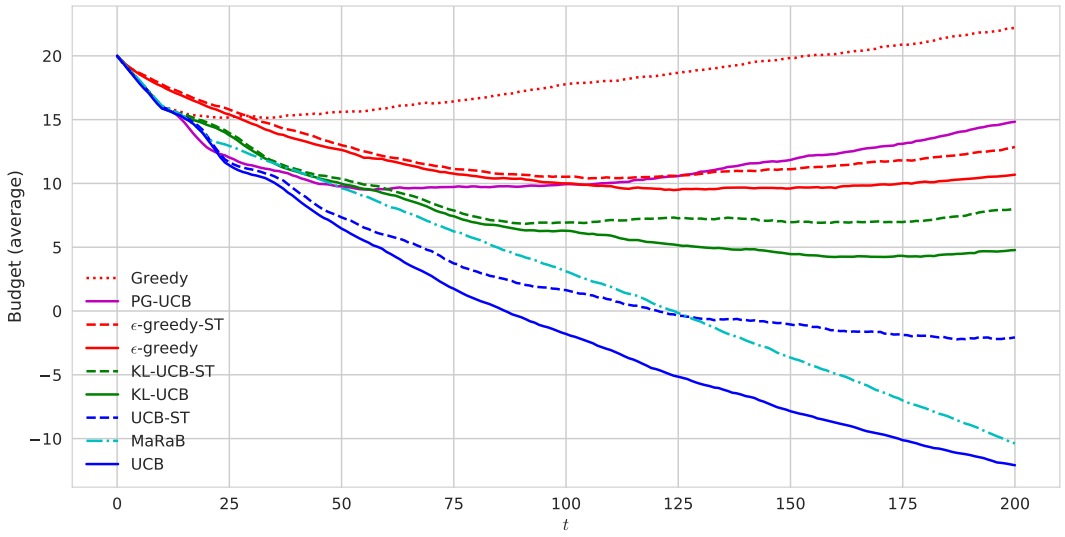


FIGURE 3 – Budget en fonction du temps, moyenne sur $n = 1000$ épisodes, avec horizon temporel $h = 200$ (court terme).

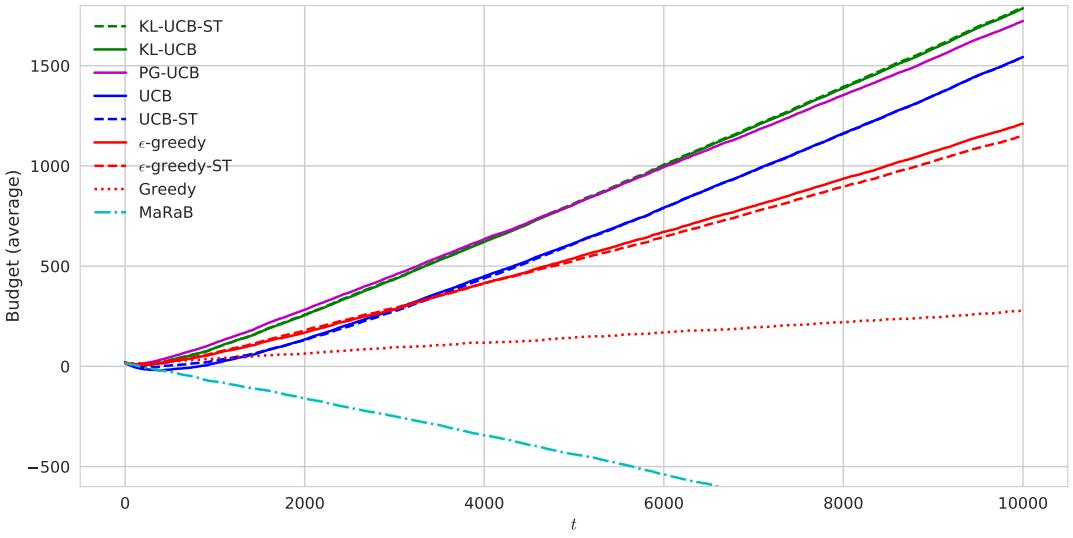


FIGURE 4 – Budget en fonction du temps, moyenne sur $n = 100$ épisodes, avec horizon temporel $h = 10000$ (long terme).

5 Résultats Expérimentaux

Les versions standard de `UCB` [5], `KL-UCB` [18], et ε`-greedy` [28] ont été comparés à leurs versions avec *seuil de sécurité* (alg. 1), appelées respectivement `UCB-ST`, `KL-UCB-ST`, et ε`-greedy-ST`, et aussi comparées à `PG-UCB` (eq. 19), à l'algorithme standard d'aversion pour le risque `MaRaB` [17], et à l'algorithme naïve *moyenne empirique maximale*, appelée simplement `Greedy`, qui n'est pas optimal. La distribution de Bernoulli est utilisée dans les expériences pour définir les fonctions de récompense, de la façon suivante : pour chaque bras i, un paramètre p_i définit la probabilité de succès, qui entraîne une récompense positive dont la valeur est $+1$. Le paramètre complémentaire $q_i = 1 - p_i$ définit la probabilité d'un échec, qui est suivi d'une récompense négative dont la valeur est -1 (en contraste avec l'usage classique de cette distribution, où l'échec est associé à la valeur 0).

Dans le scénario expérimenté, 10 de ces bras sont déployés avec différentes moyennes répartis linéairement entre $-1,0$ et $+0,1$ (i.e. p entre $0,0$ et $0,55$ inclus). Dans cette configuration, un seul parmi ces dix bras présente une moyenne positive. Le budget initial est fixé à $b_0 = 20,0$. Pour les versions sécurisées (alg. 1), le seuil de sécurité est défini à $\omega = 3,0$, ce qui garantit qu'au moins 2 actions supplémentaires peuvent être exécutées en toute sécurité après être averti du niveau faible du budget et avant la ruine.

En plus d'être mesurées par rapport au regret ou à la récompense cumulée, les méthodes ont été comparées en fonction de la progression de leur taux de survie (i.e. nombre d'épisodes qui se finissent sans que le budget touche des valeurs négatives). Ces résultats peuvent être observés dans les figures 1 à 4. L'analyse de court terme (résultats obtenus pour des simulations qui s'arrêtent sur un horizon temporel $h = 200$), correspondant au début du processus où le danger d'épuiser le petit budget initial est plus grand, révèle la capacité de chaque algorithme à éviter la ruine. L'analyse de long terme (pour des simulations qui s'étendent sur un horizon temporel $h = 10000$) montre leur efficacité à minimiser le regret.

Les résultats indiquent que `UCB1-ST` et `KL-UCB-ST`, qui utilisent le seuil de sécurité, présentent clairement une meilleure performance si comparés à la performance des versions standard correspondantes, `UCB1` et `KL-UCB`, dans l'aspect survie, c'est-à-dire en ce qui concerne χ et ξ. Il est possible d'observer aussi que le regret supplémentaire payé pour utiliser le seuil de sécurité n'est pas significatif à long terme (fig. 2). Les méthodes naïves `Greedy`, ϵ`-Greedy-ST`, et ϵ`-Greedy` semblent très performantes si analysées dans le court terme, mais à long terme, on observe un regret linéaire.

Le dernier algorithme présenté, le *joueur positif* (PG-UCB) montre un compromis intéressant entre performance et survie, cependant, la dernière figure laisse entendre un regret linéaire. Cela peut être expliqué par l'utilisation de la probabilité des actions d'être positives comme premier terme de l'équation 19. Dû aux différentes variances, l'action qui présente la plus haute probabilité d'être positive à partir des observations n'est pas nécessairement celle qui a la meilleur moyenne.

6 Conclusion et Perspectives

Cet article définit une nouvelle version du problème classique des bandits manchots appelée MAB survivant. Cette nouvelle version ne peut pas être résolue par les algorithmes développés pour le MAB budgétisé ou pour les MAB averses au risque. Dans cette définition originale, en plus de résoudre le dilemme classique entre exploration et exploitation, l'agent doit trouver un compromis entre sécurité et risque, afin d'éviter sa propre ruine, tout en essayant de minimiser le regret.

Deux algorithmes ont été présentés pour aborder cette nouveau problème : seuil de sécurité et joueur positif. Les deux algorithmes surpassent les méthodes classiques du MAB en termes de maximisation du taux de survie. Ces deux algorithmes constituent une première proposition de méthodes pour répondre à de tels problèmes. Les travaux futurs incluent l'analyse théorique sur les limites de regret de tels algorithmes, ainsi que l'extension du modèle de survie aux processus de décision markoviens.

Références

[1] R. Agrawal. Sample mean based index policies with o(log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, 27(4) :1054–1078, 1995.

[2] S. Agrawal, N.R. Devanur, and L. Li. An efficient algorithm for contextual bandits with knapsacks, and an extension to concave objectives. In *29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, pages 4–18, 2016.

[3] W.R. Ashby. *Introduction to Cybernetics*. Chapman & Hall, 1956.

[4] J.-Y. Audibert, S. Bubeck, and R. Munos. Best arm identification in multi-armed bandits. In *23rd Conference on Learning Theory (COLT), Haifa, Israel, June 27-29, 2010*, pages 41–53, 2010.

[5] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3) :235–256, 2002.

[6] A. Badanidiyuru, R. Kleinberg, and A. Slivkins. Bandits with knapsacks. *J. ACM*, 65(3) :13 :1–13 :55, 2018.

[7] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. *X*-armed bandits. *Journal of Machine Learning Research*, 12 :1655–1695, 2011.

[8] W.B. Cannon. *The wisdom of the body*. W. W. Norton, New York, 1939.

[9] A. Cassel, S. Mannor, and A. Zeevi. A general approach to multi-armed bandits under risk criteria. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, pages 1295–1306, 2018.

[10] N. Cesa-Bianchi and P. Fischer. Finite-time regret bounds for the multiarmed bandit problem. In *5th Int. Conf. on Mach. Learn. (ICML)*, pages 100–108. Morgan Kaufmann, 1998.

[11] O. Dekel, J. Ding, T. Koren, and Y. Peres. Bandits with switching costs : $T^{2/3}$ regret. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 459–467, 2014.

[12] R. Dekkers. *Applied Systems Theory*. Springer, Switzerland, 2015.

[13] W. Ding, T. Qin, X.-D. Zhang, and T.-Y. Liu. Multi-armed bandit with budget constraint and variable costs. In *27th AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*, 2013.

[14] M. Ehrgott. Multiobjective optimization. *AI Magazine*, 29 :47–57, 2008.

[15] S.N. Ethier. Gambler's ruin. In *The Doctrine of Chances*, Probability and its Applications, pages 241–274. Springer, 2010.

[16] W. Feller. *An Introduction to Probability Theory and Its Applications*. Wiley, New York, NY, 1966.

[17] N. Galichet, M. Sebag, and O. Teytaud. Exploration vs exploitation vs safety : Risk-aware multi-armed bandits. In *Asian Conference on Machine Learning (ACML), Canberra, ACT, Australia, November 13-15, 2013*, pages 245–260. PMLR, 2013.

[18] A. Garivier and O. Cappé. The KL-UCB algorithm for bounded stochastic bandits and beyond. In *24th Annual Conference on Learning Theory (COLT), June 9-11, 2011, Budapest, Hungary*, pages 359–376, 2011.

[19] A. Garivier, H. Hadiji, P. Ménard, and G. Stoltz. Kl-ucb-switch : optimal regret bounds for stochastic bandits from both a distribution-dependent and a distribution-free viewpoints. *CoRR*, abs/1805.05071, 2018.

[20] S. Guha and K. Munagala. Approximation algorithms for budgeted learning problems. In *39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 104–113, 2007.

[21] E. Kaufmann, O. Cappé, and A. Garivier. On bayesian upper confidence bounds for bandit problems. In *15th International Conference on Artificial Intelligence and Statistics (AISTATS), La Palma, Canary Islands, Spain, April 21-23, 2012*, pages 592–600, 2012.

[22] T. Koren, R. Livni, and Y. Mansour. Multi-armed bandits with metric movement costs. In *30th Neural Information Processing Systems, 4-9 December 2017, Long Beach, CA, USA*, pages 4122–4131, 2017.

[23] T.L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.*, 6(1) :4–22, 1985.

[24] O.-A. Maillard. Robust risk-averse stochastic multi-armed bandits. In *24th Int. Conf. on Algorithmic Learning Theory (ALT) Singapore, October 6-9, 2013*, pages 218–233, 2013.

[25] O.-A. Maillard, R. Munos, and G. Stoltz. A finite-time analysis of multi-armed bandits problems with kullback-leibler divergences. In *24th Annual Conference on Learning Theory (COLT), June 9-11, 2011, Budapest, Hungary*, pages 497–514, 2011.

[26] N. Meuleau and P. Bourgine. Exploration of multi-state environments : Local measures and back-propagation of uncertainty. *Mach. Learn.*, 35(2) :117–154, 1999.

[27] A. Sani, A. Lazaric, and R. Munos. Risk-aversion in multi-armed bandits. In *26th Conference on Neural Information Processing Systems, December 3-6, 2012, Lake Tahoe, Nevada, USA*, pages 3284–3292, 2012.

[28] R.S. Sutton and A.G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

[29] L. Tran-Thanh, A.C. Chapman, A. Rogers, and N.R. Jennings. Knapsack based optimal policies for budget-limited multi-armed bandits. In *26th AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Canada*, 2012.

[30] S. Vakili and Q. Zhao. Risk-averse multi-armed bandit problems under mean-variance measure. *J. Sel. Topics Signal Processing*, 10(6) :1093–1111, 2016.

[31] M. Wiering and M. Otterlo. Reinforcement learning and markov decision processes. In *Reinforcement Learning : State-of-the-Art*, pages 3–42. Springer, 2012.

[32] Y. Xia, T. Qin, W. Ding, H. Li, X.-D. Zhang, N. Yu, and T.-Y. Liu. Finite budget analysis of multi-armed bandit problems. *Neurocomputing*, 258 :13–29, 2017.

[33] D.P. Zhou and C.J. Tomlin. Budget-constrained multi-armed bandits with multiple plays. In *32nd AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.

Contingent planning using counter-examples on deterministic plans

S. Piedade[1] A.Grastien[2] C.Lesire[1] G.Infantes[3]

[1] ONERA, Toulouse, France
[2] Data61/CSIRO, Canberra, Australia
[3] JOLIBRAIN, Toulouse, France

sebastien.piedade@onera.fr

Résumé

Dans cet article, nous détaillons une méthode de planification en environnement incertain. L'approche consiste à étendre le comportement du planificateur conformant CPCES au problème de planification contingente. CPCES est utilisé pour calculer un premier plan en réduisant un problème conformant en un problème de planification classique. Une regression est ensuite utilisée pour déterminer si le plan est valide pour tous les états initiaux possibles. Si ce n'est pas le cas CPCES extrait un contre-exemple parmi ces états initiaux. Ce contre-exemple est ensuite inclus dans l'ensemble des états initiaux possibles et le processus de planification recommence à partir de cette nouvelle configuration. La contribution intervient lorsque CPCES détermine qu'il n'y a pas de plan valide pour la configuration des états initiaux possibles. Dans ce cas, l'idée est d'ajouter une observation de l'état du système et de calculer une branche alternative afin de transformer le plan en plan conditionnel permettant de gérer la configuration des états initiaux.

Mots Clef

Planification contingente, incertitude, décision.

Abstract

In this paper, we detail a method to handle planning in uncertain worlds. The approach consists in extending the behaviour of CPCES developed for Conformant Planning to the problem of Contingent Planning. The conformant planner CPCES is used to compute a first plan by reducing a conformant problem in a classical planning problem. It then uses regression to determine whether the plan is valid or not for any initial state and extracts a counter-example if necessary. This counter-example is then included in the set of possible initial states and the planner restarts the planning process in the new configuration. The contribution lies in the case where CPCES determines that there is no conformant plan for the configuration of possible initial states. In this case, the idea is to add an observation of the system and to compute an alternative branch to transform the plan in a conditional plan able to handle the initial state configuration.

Keywords

Contingent planning, uncertainty, decision.

1 Introduction

Nowadays, autonomous robots missions are becoming more and more complicated. The need to explore increasingly dynamic and uncertain environments requires an improvement of systems autonomy. This improvement requires better management of environmental uncertainty that affects the robot's decision-making process during the mission. To deal with this uncertainty, it is necessary to improve the robustness of the planning process used to compute the various tasks the robot needs to perform to carry out its mission. One way to improve the robustness of a plan is Conformant Planning [12], which consists in finding a plan with no observability and working for several possible initial states. Another way to deal with uncertainty is Contingent Planning [7], which consists in computing a conditional plan containing branches allowing an online decision making influenced by the results of some system observations.

In this paper we propose to extend the behaviour of CPCES (Conformant Planner via Counter Example and Sampling) [4] developed for Conformant Planning to the problem of Contingent Planning. We chose to use CPCES because of its ability to handle large problems and to prove that some instances are unsatisfiable. This last ability is useful to determine if it is possible to compute a plan without any observation of the system. If it is not possible, the idea is to add an observation allowing to compute a conditional branch to the plan, which becomes a contingent plan able to handle the initial states for which there is no possible conformant plan.

We implement our approach for the application case of a robot that must collect some rocks in an environment containing obstacles that could harm the robot. The uncertainty here is represented by the obstacles positions : the robot has a knowledge about the possible locations of obstacles, without knowing their exact position.

In the following section, we will expose some commonly used methods able to handle uncertainties, then we will detail the formalization of the planning problem and we will present and discuss about the approach we develop to solve this problem.

2 Related works

Several approaches and methods are already used to solve problems containing uncertainties. When uncertainties result in non-deterministic action effects, a replanning approach is generally used. This approach consists in finding a plan without considering the uncertainty with a classical planner by relaxing the planning problem, and then replan when an action results in an unplanned outcome. A way to relax the planning problem in a replanning approach is to separate a non-deterministic action in various deterministic actions dividing each possible outcome like in [9]. When actions outcomes are stochastic, the relaxation of the planning problem consists most of the time in substituting each stochastic outcome by the most probable outcome like in RFF [13]. RFF is a Markov Decision Process (MDP) planning algorithm that generates offline robust policies in probabilistic domains. RFF relaxes an MDP planning problem by translating it into a deterministic planning problem. This method then uses the FAST-FORWARD (FF) classical planner [6] to compute a policy which has a low probability of causing any replanning during execution. In order to compute the probability that the policy will cause a failure, RFF uses sequential Monte-Carlo simulation.

When the uncertainty is about the initial state of the system, some planners have been developed based on an heuristic forward search and relying on the FF planner. Conformant-FF [8] is one of these planners considering no observation of the system. Conformant-FF extends FF by reasoning about the effects of action sequences instead of the sets of possible worlds thanks to conjunctive normal form implication tests on propositional formulas.

The Contingent-FF planner [7] has been developed, extending the Conformant-FF planner so that it could handle system observations via a modification of its heuristic techniques and its sparse belief space representation.

When uncertainty can be modeled as probabilities, MDPs [10] and partially observable MDPs (POMDPs) are generally used to model the planning domain as a stochastic system assigning probabilities to state transitions and modeling the actions effects by a probability distribution. In MDPs and POMDPs there is generally no set of desired goal states but a utility function expressing preferences on the execution path of a plan.

Another method recently developed is the robot system Dora [5]. To deal with the incompleteness in knowledge, Dora integrates a planner which has assumptive actions making assumptions about instance knowledge that is, knowledge about environment entities. Dora deals efficiently with incomplete knowledge thanks to a task-driven selection of these assumptions.

Contrary to a replanning method which usually computes a new plan online to react to environments changes, our approach computes a conditional plan offline allowing the robot to make online decisions without replanning. Unlike MDPs and POMDPs in which we observe the system in each state, our approach tries to minimize the number of observations of the system by performing the observations only when it is needed. Instead of considering the belief states as action sequences like in Contingent-FF, we consider the belief states as a set of interpretations of the possible initial states and we advantageously use the intelligent belief state sampling of CPCES to compute a contingent plan.

3 Problem Formalization

The problem formalization is inspired by [3],[2] and [11].

3.1 Planning Domain definition

Definition 1 (State)**.** *We define the set of states S. We write s a state in S.*

Following [11], we define :

Definition 2 (Actions)**.** *We write A the set of action operators $a = (c, e)$ in which :*
- *c is a formula on S describing the preconditions of a*
- *e is a formula on S describing the effects of a.*

Definition 3 (Observations)**.** *We write Ω the set of observation operators $o = (c, e)$ in which :*
- *c is a formula on S describing the preconditions of o*
- *e is a formula on S describing the effects of o.*

The following definition is inspired by [11] and [2]. We define the planning domain as follows :

Definition 4 (Planning domain)**.** *A planning domain $D = (S, I, G, A, \Omega)$ is a state-transition system, where S is the set of states of the system, $I \subset S$ is the set of possible initial states and $G \subset S$ is the goal. The state space is built by applying an action $a \in A$ or an observation $o \in \Omega$ in every state $s \in S$.*

3.2 Conditional plan representation

We use the definition from [3] to describe conditional plans.

Definition 5 (Conditional plans from [3])**.** *The set of conditional plans Π for a domain D and χ the evaluation of an observation function is the minimal set such that :*
- *$\lambda \in \Pi$;*
- *if $a \in A$, then $a \in \Pi$;*
- *if $\pi_1, \pi_2 \in \Pi$, then $\pi_1; \pi_2 \in \Pi$;*
- *if $o \in \Omega$, and $\pi_1, \pi_2 \in \Pi$ then **if** χ_o **then** π_1 **else** $\pi_2 \in \Pi$.*

*With λ the empty plan. The plan $\pi_1; \pi_2$ is the sequential composition of plans π_1 and π_2. The plan **if** χ_o **then** π_1 **else** $\pi_2 \in \Pi$ is a conditional plan that branches on the value of observation o.*

In our approach, χ_o corresponds to the effects e of observation o.

4 Contingent Planning using CPCES

4.1 Architecture

The architecture described in Fig. 1 is an extension of the CPCES architecture [4]. The original architecture of CPCES is represented in orange. Our contribution is illustrated by the blue parts of the architecture.

CPCES considers a first set of interpretations of the possible initial states. These interpretations are implemented thanks to a multi interpretation domain (1) and a multi interpretation instance (2). The problem is reduced to classical planning for this set of interpretations, and the FF classical planner (3) is used to compute a plan (4). The regression (5) of the plan is computed by generating a SAT-problem. Using the Z3 SAT-Solver (6) CPCES determines whether the computed plan is correct for any initial state. If the plan is not correct then the initial state for which the plan is not valid is considered as a counter-example. This counter-example is extracted and added to the set of considered interpretations (7). Once the counter-example has been added, the planning process is restarted with the new multi interpretation instance and the process is repeated until no counter-example is found.

Our contribution steps in when the FF planner determines that there is no plan for the set of interpretations. The idea is to find an observation $o = (c, e) \in \Omega$ which allows to discriminate the counter-example and to add this observation in the plan. A new branch of the plan is then computed to handle this counter-example. For that, the previously computed plan (8) is transmitted to a simulator (9) that uses the multi interpretation instance and the multi interpretation domain to simulate the plan until it reaches the position where the discriminating observation should be performed. If no observation can be performed to discriminate the counter-example, the current plan cannot be fixed (10). If an observation exists, it is then placed in this position in the plan and transmitted (11) to the conditional module (12). The current set of interpretations is modified including the result of the observation discriminating some of the initial states. This new set of interpretation is then transmitted by the conditional module to the FF planner as a multi interpretation instance (13) in order to use the CPCES planning process to compute a new conformant branch of the plan from this position. This conformant branch is then transmitted (14) to the conditional module that adds it to the plan which becomes a conditional plan. The conditional module verifies if there is no counter-example of the contingent plan by successively transmitting each branch of the plan (15) to the SAT-Solver. If there is no counter-example the contingent plan is returned (16), otherwise the planning process restarts for the branch in which a counter-example has been found.

4.2 Planning process illustration

To illustrate the architecture and the approach, we study the application case of a robot that has to collect a rock. There are two obstacles and the robot has an uncertain knowledge about the location of these obstacles, considering three possible locations for the first obstacle and two possible locations for the second obstacle. The robot's initial knowledge is divided in a set of interpretations that corresponds to the possible initial states.

Fig. 2 represents the uncertain knowledge the robot has about the environment. In Fig. 2 the robot is represented by an octogone, the rock is diamond shaped, the first obstacle is a black square and the second obstacle is a grey square. In our example there are six interpretations $I = \{s_0^1, s_0^2, s_0^3, s_0^4, s_0^5, s_0^6\}$. We consider that the set of states S is represented by a set of literals like in the PDDL language [3]. Each interpretation is a state represented by the following predicates : x, y describing the position of the robot, b_x, b_y describing the position of the black obstacle, gr_x, gr_y describing the position of the grey obstacle, r_x, r_y describing the position of the rock, $harmed$ describing the life status of the robot and $collected$ describing the fact that the rock has been collected.

The set of actions $A = \{up, down, right, left, collect\}$ with up (abbreviated as u), $down$ (d), $right$ (r), $left$ (l) the actions moving the robot in an adjacent position (we write $adjacent(x, y, x', y')$ the predicate indicating that positions (x,y) and (x',y') are adjacent) and $collect$ (col) the action allowing to collect a rock in the robot's position.

There is only one type of observation $o \in \Omega$ that can be performed by the robot. This observation is a partial observation of the local state of the robot, that provides information about a position adjacent to the robot. We have $o(x', y') = (c, e)$ with $c = adjacent(x, y, x', y')$ and $e = b\_x(x') \wedge b\_y(y') \vee gr\_x(x') \wedge gr\_y(y')$.

The set of goal states G is defined by formula $(\neg harmed \wedge collected)$.

4.3 CPCES planning process

In this section we shortly illustrate the planning process of CPCES [4].

Multi interpretation Domain and Multi interpretation instance (Module 1 and 2 in Fig. 1). The multi interpretation domain and multi interpretation instance are implemented in the PDDL language [3]. CPCES authors extend each predicate of the planning domain with a parameter that represents the currently considered interpretations. Thanks to this parameter, action preconditions and effects are applied over all the possible interpretations. This ensures that an action can be applied only if each of the possible interpretations satisfies its preconditions. The use of a multi interpretation domain and instance allows to handle efficiently the implementation of the set of considered interpretations of the robot's knowledge.

In the rock collecting problem, we choose the interpretation s_0^1 in the robot's knowledge as a first considered initial

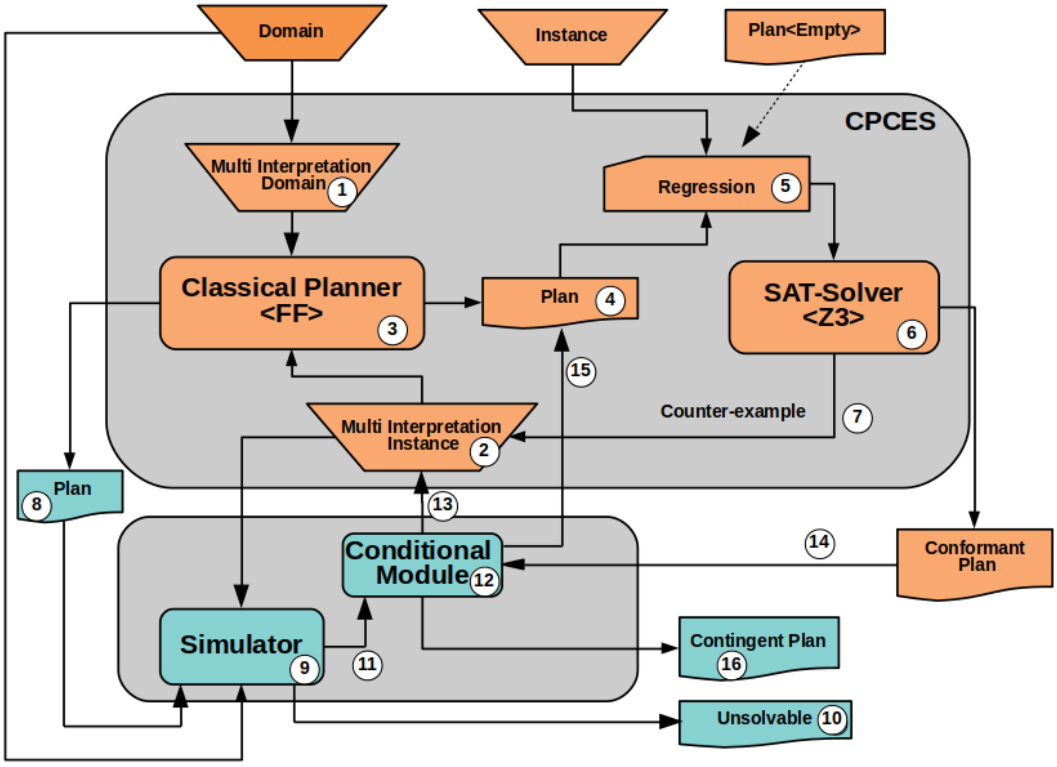


FIGURE 1 – Approach architecture

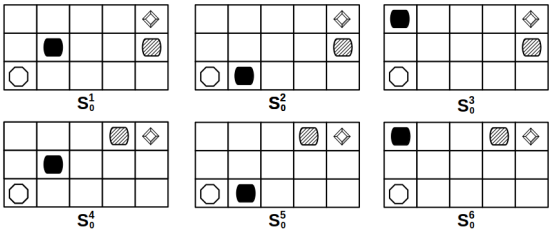


FIGURE 2 – Illustration of the robot's knowledge

state. At this stage of the process, the multi interpretation instance is only representing this first instance (see Fig. 3).

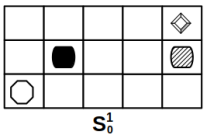


FIGURE 3 – First considered initial state

For a complete description of the implementation see [4].

Planning with FF (3,4). The FF planner is used to compute a plan able to handle the set of considered initial states. FF returns a plan as an action sequence or determines that there is no solution considering the multi interpretation instance. The resulting plan is transmitted to the SAT Solver in order to perform a regression and to find a counter-example in the set of possible initial states I.

In the rock collecting problem, from the initial state s_0^1 of Fig. 3, FF returns the plan depicted in Fig. 4, the circled state representing a goal state.

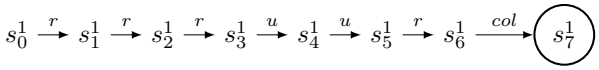


FIGURE 4 – Illustration of the computed plan

Counter-example finding (5,6,7) . The regression of the plan computed by FF is done by generating a SAT problem and using the Z3 SAT-Solver [1]. Regression consists in the computation of the weakest precondition on the initial set that guarantees that the plan is a solution. The regression process is described in [4]. If no counter-example is found in I, then the plan computed by FF is conformant and the problem is solved. Otherwise a counter-example has been found. The multi interpretation instance is then modified to include the new counter-example and the planning process restarts with the new set of interpretations.

In our example, CPCES find a counter-example and adds it to the set of considered initial states (see Fig. 5).

During the next iteration, FF cannot compute a conformant plan able to handle the set of initial states $\{s_0^1, s_0^4\}$.

FIGURE 5 – New set of initial states

4.4 Contingent planning process

In this section we illustrate the contingent planning process used to handle the case where FF cannot compute a plan handling the set of initial states.

Simulator (8,9,10,11). The plan computed by FF in the previous step and the multi interpretation instance are transmitted to the simulator.
Let S_0^i be the set of initial states in the multi interpretation instance with $1 \leq i \leq M$ and M the size of the set of considered interpretations, S_g^i the set of states in which the goal is reached. The plan computed by FF and transmitted to the simulator is an action sequence $< a_0, .., a_g >$. Since CPCES has not succeeded in finding a plan for the multi interpretation instance transmitted to the simulator, then there is a counter-example in S_0^i for which an action a_{k+1} in the action sequence $< a_0, .., a_g >$ cannot be performed. In order to find this counter-example the simulator executes the plan $< a_0, .., a_g >$ on the set of interpretations S_0^i until an action of the plan cannot be performed in a state $s_k^c \in S_k^i$. We write s_k^c the state that corresponds to the k^{th} step of execution of the plan on the counter-example interpretation. Since the action sequence $< a_0, .., a_k >$ is applied until the state s_k^c is reached, we can identify the initial state s_0^c as the counter-example interpretation of the previous plan.
An observation needs to be performed in order to discriminate the state s_k^c for which the previous plan is not valid. We need to find an observation o such that the preconditions of o are verified in S_k^i and the effects of the observation o are in s_k^c. If no observation can be added to the plan to discriminate the counter-example, we failed to find a plan for this set of interpretations. Otherwise, the simulator adds the observation to the simulated plan that becomes the incomplete contingent plan described in Fig. 6. χ_o is the result of the observation o applied in states S_k^i. $\{S_k^i\}_{i \neq c}$ corresponds to the set of states at the k^{th} step of the execution of the plan in the case where $\chi_o \neq s_k^c$. In this case we keep the previously computed branch $< a_{k+1}, .., ag >$ that is valid from the set of states $\{S_k^i\}_{i \neq c}$ that does not contain the counter-example. The result of an observation allows to filter the set of possible interpretations by deleting the interpretations that do not correspond to the observation result. In the figures we write $I = \{1, .., 6\}$ to illustrate the indexes of the possible interpretations in which we look for a counter-example. We have $S_k^c = \{s \in S_k^i \ / \ \chi_o \in s\}$ and $\{S_k^i\}_{i \neq c} = \{s \in S_k^i \ / \ \chi_o \notin s\}$ The contingent plan described in Fig. 6 is then transmitted to the conditional module. In the rock collecting problem, the plan computed in the

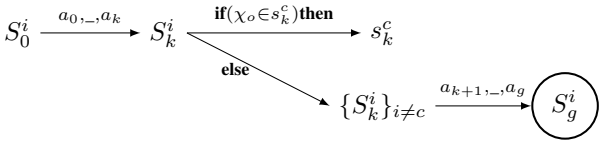


FIGURE 6 – Incomplete contingent plan transmitted to the conditional module

previous step by FF (see Fig. 4) and the multi interpretation instance for which FF cannot compute a plan (see Fig. 5) are transmitted to the simulator. We simulate the plan transmitted to the simulator until we reach the state s_4^4 (see Fig. 7) in which the robot cannot move up because of the obstacle. The state s_4^4 is reached after the execution of the action sequence $< r, r, r, u >$ from the s_0^4 interpretation (see Fig. 5). Once this position is reached, we need to perform observation $o(3, 2)$ in order to discriminate the interpretation s_4^4. The simulator adds the observation to the plan and transmits the incomplete contingent plan (see Fig. 8) to the conditional module.

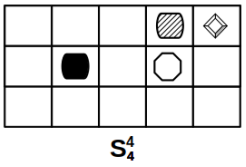


FIGURE 7 – State where the simulator can not apply the fifth action of the plan described in Fig. 4

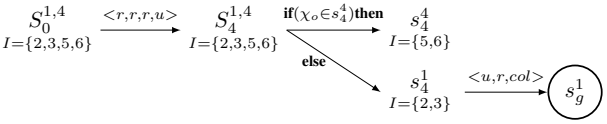


FIGURE 8 – Contingent plan transmitted to the conditional module

Conditional module (12,13,14,15,16). The first contingent plan received by the conditional module contains an observation giving two possible branches (see Fig. 6). The branch containing the discriminated counter-example state s_k^c is incomplete because it does not lead to a goal state. In order to compute the incomplete branch, the conditional module uses the planning process of CPCES [4] by transmitting the multi interpretation instance describing the state s_k^c where the observation has discriminated the counter-example to the FF planner. If CPCES does not succeed to compute a conformant branch from s_k^c, the simulation process is restarted from the new set of interpretation containing a counter-example of the computed branch. The simulator then transmits the conditional branch to the conditional module which restarts the process with the incomplete part of the new branch. Otherwise, if CPCES succeeds to find a conformant branch then the conditional module adds the conformant branch to the previous plan (illus-

trated by Fig. 6).
We write $\pi' =< a'_{k+1}, .., a'_g >$ the plan computed by CPCES from the state s^c_k. If π' is conformant then the conditional module adds it to the previous plan (see Fig 9).

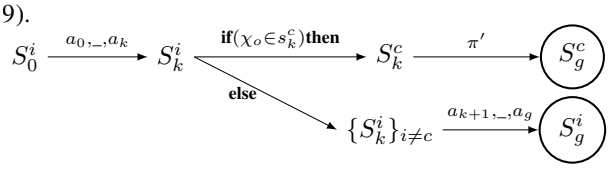


FIGURE 9 – Contingent plan with a conformant π'

Else the simulation process is restarted for π' and the planning process compute the contingent branch π'' until each branch of π'' leads to a goal. The conditional module adds the contingent branch π'' to the previous plan.
In the rock collecting example, CPCES computes the conformant branch $s^4_4 \xrightarrow{<r,u,col>} s^4_g$ to replace the incomplete branch of Fig. 8. The conditional module assembles this branch to the plan described by Fig. 8 (see Fig. 10).

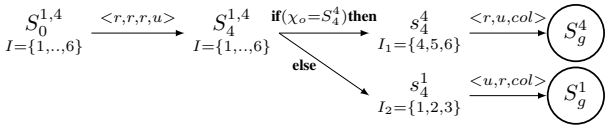


FIGURE 10 – Contingent plan assembled by the conditional module

Every branch of the contingent plan in Fig. 8 are transmitted to the SAT-Solver to determine if there is a counter-example in any branches of the plan.

In the rock collecting example, a counter-example is found for the branch $S^{1,4}_0 \xrightarrow{<r,r,r,u>} S^{1,4}_4$ of the plan. The counter-example s^5_0 is added to the set of considered initial states (see Fig. 11). The new set of initial states (see Fig. 11) and the branch $S^{1,4}_0 \xrightarrow{<r,r,r,u>} S^{1,4}_4$ are transmitted to the simulator which adds an observation in the branch that discriminates the counter-example (see Fig. 11).

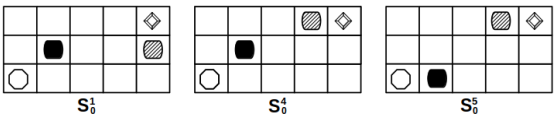


FIGURE 11 – Set of initial states

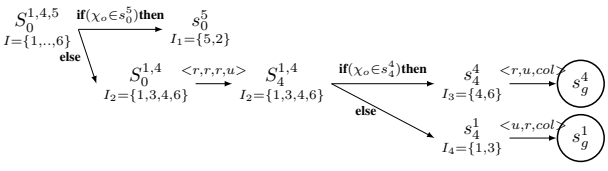


FIGURE 12 – Contingent plan assembled by the conditional module

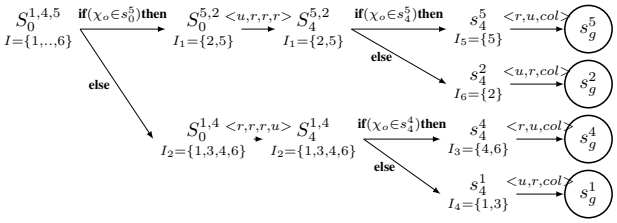


FIGURE 13 – Final Contingent plan

We then restart the CPCES planning process for the incomplete branch. Here we need to compute a new branch starting from s^5_0, The observation reduces the set of initial states from this position by eliminating the s^1_0 and s^4_0 interpretations.
The branch $s^5_0 \xrightarrow{<u,r,r,r,r,u,col>} s^5_g$ is computed but the interpretation s^2_0 is a counter-example for this branch. The simulation and planning process restarts until no counter-example is found in any branches of the plan. The conditional module then assembles the contingent branch with the plan described in Fig. 12 to create the contingent plan described by Fig. 13.
The contingent plan described by Fig. 13 handles every possible initial states in the robot's knowledge illustrated by Fig. 2.

5 Conclusion

In this paper we describe a new approach extending the behaviour of CPCES developed for Conformant Planning to the problem of Contingent Planning. The contribution steps in when there is no possible conformant plan for the set of interpretations. An observation is then added to the plan to discriminate the problematic counter-example. From the new set of considered interpretations reduced by the observation, a new branch is computed and added to the plan. The counter-example search is then restarted for each branch of the plan and the conditional planning process restarts until no counter-example has been found for any branches of the plan. The formalization described in this paper can be generalized to other planning problems and one of the next steps of this work is to implement the method on various application cases in order to compare our method with existing Contingent planning methods.

Références

[1] L. De Moura and N. Bjørner. Z3 : An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008.

[2] P. E. U. de Souza, C. P. C. Chanel, and F. Dehais. Momdp-based target search mission taking into account the human operator's cognitive state. In *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*. IEEE, 2015.

[3] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning : theory and practice*. Elsevier, 2004.

[4] A. Grastien, E. Scala, and F. B. Kessler. Intelligent belief state sampling for conformant planning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017.

[5] M. Hanheide, M. Göbelbecker, G. S. Horn, A. Pronobis, K. Sjöö, A. Aydemir, P. Jensfelt, C. Gretton, R. Dearden, M. Janicek, et al. Robot task planning and explanation in open and uncertain worlds. *Artificial Intelligence*, 247, 2017.

[6] J. Hoffmann. Ff : The fast-forward planning system. *AI magazine*, 22(3), 2001.

[7] J. Hoffmann and R. Brafman. Contingent planning via heuristic forward search with implicit belief states. In *Proc. ICAPS*, volume 2005, 2005.

[8] J. Hoffmann and R. I. Brafman. Conformant planning via heuristic forward search : A new approach. *Artificial Intelligence*, 170(6-7), 2006.

[9] U. Kuter, D. Nau, E. Reisner, and R. P. Goldman. Using classical planners to solve nondeterministic planning problems. In *Proceedings of the Eighteenth International Conference on International Conference on Automated Planning and Scheduling*. AAAI Press, 2008.

[10] M. L. Puterman. *Markov decision processes : discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[11] J. Rintanen. Complexity of planning with partial observability. In *ICAPS*, 2004.

[12] D. E. Smith and D. S. Weld. Conformant graphplan. In *AAAI/IAAI*, 1998.

[13] F. Teichteil-Koenigsbuch, G. Infantes, and U. Kuter. Rff : A robust, ff-based mdp planning algorithm for generating policies with low probability of failure. *Sixth International Planning Competition at ICAPS*, 8, 2008.

A study of local search approaches for plan repair with plan distance criteria

H. Soubaras
Thales Research & Technology
1 av. A. Fresnel 91767 Palaiseau Cedex - France
helene.vandenbroek@thalesgroup.com

Résumé

Ce papier présente une étude de benchmarking de deux algorithmes de réparation de plan : la recherche tabou dérivée de Gambardella & Mastrolilli [5] et l'algorithme du taquin à potentiel qui a été proposé récemment [11, 12, 13]. Dans un problème d'atelier flexible (FJSP, Flexible Job-Shop scheduling Problem) généralisé, il faut réparer un plan lorsqu'une machine tombe en panne.

L'algorithme de réparation de plan requiert en deuxième critère (en plus du makespan) une mesure de distance entre le plan réparé et le plan initial. Cette étude montre des résultats sur deux critères de distance proposés obtenus sur une série de problèmes de planification. Des idées sur l'explicabilité en réparation de plan sont aussi fournies.

Mots Clef

Réparation de plan, mesures de distance entre plans, recherche tabou, multicritère, explicabilité.

Abstract

This paper presents a benchmark study of two plan repair algorithms : a tabu search derived from Gambardella & Mastrolilli [5] and the 15-puzzle algorithm with potential that was proposed recently [11, 12, 13]. In a generalized Flexible Job-Shop scheduling Problem (FJSP), one must repair a plan when a machine breaks down.

The plan repair algorithm needs, as a second criterion (in addition to the makespan), a measure of distance between the initial and the repaired plan. This study shows some results on two proposed distance criteria and a series of planning problems. Some ideas about explanability are also provided.

Keywords

Plan repair, plan distance measures, tabu search, multicriteria, explanability.

1 Introduction

This paper presents some results to compare two plan repair algorithms that were proposed by the authors is previous works. Plan repair is a research topic of major practical interest but still lacking of general methods. When an initial plan is being executed, some discrepancies may appear between the observations given by the monitoring system and what was expected. This may occur because of an extra event causing a change in the planning problem. One must then perfrom plan repair in order to preserve some stability (as recommended by Fox *et al.* [4]). The proposed work focuses on a generalized FJSP model for the planning problem and ensures plan stability (i.e. avoiding too much chages) thanks to a plan distance criterion. It is used to modify a classical tabu search planning algorithm [5], and also in the 15-puzzle algorithm with potential that was proposed in [11, 12, 13]

The motivation of the presented work is to carry on studying these algorithms by comparing them, and comparing the possible distance criteria they involve. Thus one will provide a state-of-the art of existing plan distance measures and describes the two new measures that are studied. The concrete planning problems and the parameters for benchmark study are given before the statistical results. Some conclusions about explanability are also given.

2 Distance between plans

2.1 In the literature

In plan repair, the makespan is not the only criterion to optimize, since the differences between the repaired plan and the original one can provoke unstability, as explained by Fox *et al.* [4]. So one uses a distance measure between the initial and the repaired plan as a second criterion. Several distance measures for plans are proposed in the literature. Fox *et al.* [4] measure consists in the number of added actions plus the number of removed actions in the new plan. Following this, a very influential work on plan diversity measures is that of Srivastava *et al.*[15], refined in Nguyen *et al.*[9]. They improved the first measure by proposing three different distance measures for comparing plans, and for measuring the diversity of a set of plans : Action distance (AD), Causal link distance (CLD), and state distance (SD). Goldman [6] proposed a measure based on the Kolmogorov complexity. AD and CLD both project the plan, an ordered sequence of actions, down to an unordered set, and then compute a set-difference based distance between these sets. Let $A(\Pi)$ denote the set of actions of plan Π, and $C(\Pi)$ denote the set of causal links (i.e. triples

(a, p, a') where p is a proposition of a state s, a is an action leading to p and a' is a following action made possible by p). The action distance (AD) is defined by

$$AD(\Pi, \Pi') = 1 - \frac{A(\Pi) \cap A(\Pi')}{A(\Pi) \cup A(\Pi')}$$

and the causal link distance is defined by

$$CLD(\Pi, \Pi') = 1 - \frac{C(\Pi) \cap C(\Pi')}{C(\Pi) \cup C(\Pi')}$$

The state distance is defined from the state sequence of the plans (rather than sets)

$$SD(\Pi, \Pi') = \frac{1}{l(\Pi)} \left[\Sigma_{i=1}^{l(\Pi')} \Delta(s_i, s'_i) + l(\Pi) - l(\Pi') \right]$$

where $l(\Pi') \leq l(\Pi)$ denote the lengths of the plans and $\Delta(s, s')$ is a measure of the difference in the fluents holding the two states s and s'.

2.2 Proposed distance measures

The distance criteria described above (in Section 2.1) have some drawbacks for us since they do not take into account
– the actions that are modified by simply changing the machines which perform them,
– the actions that are modified by time delays,
– the independence between some jobs,
– the parallelism between the machine queues (subplans).
Moreover, as the jobs must be preserved (oherwise the plan repair fails). Thus the AD, SD and CLD distances remain null within them.

In other words, the suppression of an action prevents the plan from reaching its goal, however changing the machine which performs it is a minor issue. And in operational contexts, time delays may be important. This is why we preferred the following measures. A change in a plan may affect the parameters of some actions in different ways. The approach proposes a quadratic distance between times (or time Mean Square Error MSE) in order to fulfill the mathematical properties of a distance :

$$D_t(\Pi, \Pi') = \sqrt{\frac{1}{2N} \sum_i d_t(a_i, a'_i)}$$

where a_i are the actions of Π, a'_i are the corresponding actions of Π', N is the total number of actions of Π, and

$$d_t(a_i, a'_i) = (t_{start}(a_i) - t_{start}(a'_i))^2 + (t_{end}(a_i) - t_{end}(a'_i))^2$$

is the square distance between the initial time $t_{start}(a_i)$ and final time $t_{end}(a_i)$ of each pair of corresponding actions a_i and a'_i before and after a plan modification.

The second distance criterion that will be studied here is `assign_changes+`, the total number of actions whose assignment has changed while performing the plan repair.

3 Implementation in two plan repair approaches

3.1 Plan repair problems to solve

The suitable planning problems for the studied algorithms are generalized FJSPs. This Section provide their mathematical expression related to several practical use-cases.

Mathematical model of the planning problems. A JSP (Job-Shop scheduling Problem) is a problem where one has jobs (ordered sequences of actions to do) and machines to perform the actions. In a JSP, there is only one machine able to perform each action. A FJSP (Flexible Job-Shop Problem) is a JSP where one can choose which machine will perform each action ; this choice is called *assignment* or *resource allocation*. So the FJSP is modelled as follows :
– a list of task types,
– a given number N_j of jobs, each one described as a ordered list of tasks to do,
– a given number N_m of machines,
– a *machine abilities* matrix containing the time duraction needed by each machine for each task type it can perform.
The FJSP model is generalized to movings, i.e. the machines or the jobs locations can move. The duration of the transit actions is taken into account inside the algorithms. This extension of the FJSP corresponds to a model known in the literature as the SDST-FJSSP (Flexible Job Shop Scheduling with Sequence Dependent Setup Times) [10]. A moving time is a kind of setup time. And when the moving objects are the machines, the moving times are sequence-dependent.

As will be detailed further, the FJSP is also generalized to jobs precedence constraints (see Section 3.1 below).

Looking at one planning problem. This Section examines in particular the off-shore windmill farm maintenance scenario, which associate planning problem is referred as `SWARMS_UC4`. It was one of the use-cases of SWARMs, a ECSEL project for the study of swarms of underwater vehicles [14]. Indeed, windmills are spreading as a means to produce renewable energy. When they are gathered in the sea in windmill farms, one difficulty is to maintain their basis which remains always under the water. The objective is to perform this maintenance thanks to underwater vehicles.

As in all the considered underwater scenarios of the SWARMs project, some underwater drones called AUVs (Autonomous Underwater Vehicles) and ROVs (Remotely Operated Vehicles)will be brought in a vessel near to the windmill farm place. Then they must be dropped in the water at a given point. In this example, there are 3 AUVs named AUV1, AUV2 and AUV3, and 3 ROVs named ROV1, ROV2 and ROV3.

The first phase of the mission consists in making a general survey of the seabed in the region around the windmills to detect problems or dangers. This survey can be shared between several vehicles equipped with appropriate wide angle cameras.

In a second phase, the feet of the windmills will be surveyed more closely, and if something wrong appears (a hole or a crack or a stone to pick up) a vehicle will inspect the issue and fix it. In the studied example, one windmill needed such a pickup operation.

Thus there are 3 task types in this problem (survey, inspect and pick up). The machines of the FJSP model are the vehicles. Each machine can do perform some task types with a lapse time provided in the *maachine abilities* matrix (see Figure 2). Solving the planning problem consists in assigning and ordering all the tasks to the appropriate vehicles. A plan repair problem arises when a vehicle has a breakdown. Thus the tasks that where assigned to this vehicle must be reassigned to others.

Jobs precedence constraints. To generalize the FJSP model, precedence constraints between jobs are proposed. This allows the fact that some jobs can start only when others are completed. These constraints are entirely described in the job precedence constraints binary matrix. For instance, in the windmill farm maintenance scenario, the machines are the vehicles (AUVs and ROVs) and the jobs are the windmills to inspect and then to fix. Prior to these maintenance actions, there is a mission of overall region survey that must be completed. The region to survey can be divided in zones that can be surveyed in parallel by several vehicles. An example is shown in Figure 1 for 3 zones. Each zone to survey is a single job. There is a jobs precedence constraint between each of them and the inspection of windmills. The precedence constraints are modelled by a binary matrix and illustrated by diamonds in Figure 2 (i.e. a diamond corresponds to "1" in the jobs precedence matrix, and no diamond corresponds to "0").

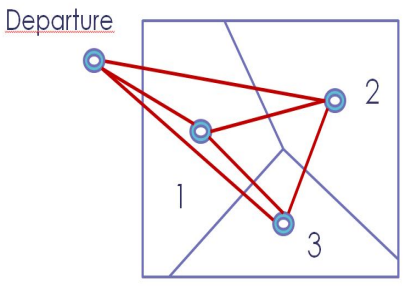


FIGURE 1 – The region to survey is divided into zones to survey in parallel.

JOBS

zone1	Survey		
zone2	Survey		
zone3	Survey		
windmill1_hole	Survey	Inspect	Pickup

Associate colors:
J1

MACHINES
Execution times

	Survey	Inspect	Pickup
AUV1	120	20	
AUV2	120	20	
AUV3	120	20	
ROV1	120	20	80
ROV2	120	20	80
ROV3	120	20	80

Precedence constraints between jobs

	zone1	zone2	zone3	windmill1_hole
zone1				◆
zone2				◆
zone3				◆
windmill1_hole				

FIGURE 2 – The planning problem SWARMS-UC4 of underwater windmill maintenance.

The tested plan repair problems. The tested planning problems are a list of use-cases given in Table 1. Some of them come from the SWARMS project (see Section 3.1). The plan repair problems that are tested are the breakdown of one machine, so the machine is suppressed from the planning problem. This is modelled by making the machine unavailable since time $t_0 = 0$. To generate benchmarks from one planning problem and its initial plan, the plan repair is tested successively for the suppression of each one of the problem machines. The breakdown of more than one machine at the same time is out of the scope of this study.

3.2 The two studied algorithms

The two plan repair algorithms that are studied here (the tabu search and the 15-puzzle algorithm) have been described in previous works [11, 12, 13]. Of course, there are other existing approaches in the literature for plan repair with distance criteria vor various planning problem models ([2, 3, 16, 1, 7, 8]).

The two plan repair approaches of this study can also be applied to simply compute a plan from a random initial plan. Gambardella & Mastrolilli's tabu search [5] was proposed to solve FJSPs (Flexible Job-Shop scheduling Problems). It is a local search consisting in moving iteratively actions's assignement one by one (like the pieces of a sliding puzzle). At each iteration, one keeps the solution with best makespan. Solutions that have already been explored are put in a tabu list to avoid staying in local optima. We proposed a multi-criteria version of this tabu search algorithm for plan repair, and its generalization to FJSP with movings and jobs precedence constraints.

We also proposed a new algorithm derived from this tabu search, the 15-puzzle with potential. It uses the same multi-criteria optimization, the same neighborhood ("moving" actions like sliding-puzzle pieces) and the same tabu list parameters as the tabu search. But this second algorithms

uses a potential calculus as a surrogate function to avoid computing all the makespans at each iteration. The potential is a rough estimation of the risk to increase the makespan. Its expression is detailed in [13]. At each iteration,
– the potential is computed for all the candidate neighbor solutions,
– a shortlist of the 10 best potentials is kept,
– the makespan and the distance criterion are computed for all the candidates of the shortlist,
– the multi-criteria choice is made on the shortlist.

3.3 The criteria

As explained, for plan repair one needs two (at least) criteria :
– the makespan (C_{max}, overall duration of the plan execution),
– a distance measure between the repaired plan and the initial plan.
For the distance measure, the two measures that were described in 2.2 were studied :
– the time MSE, which has to be minimized,
– the number of assignment changes, which has to be minimized.
Of course, the distance criterion can also be composed of various distance criteria (e.g. a weighted combination of time MSE and number of change assignments).

In the two studied algorithms, at each iteration, the best candidate is chosen following the two criteria in lexicographic ordering. This means the candidate solution with best makespan is taken ; but if there are several ex-aequo candidates (i.e. with the same makespan), one retains the one which has the best distance criterion. It is important to know if the presence of several ex-aequo happens often, because if there is almost all the time one single candidate with best makespan, the second criterion will have no influence.

3.4 Running benchmarks

The objective of this work is to run the two algorithms on the series of plan repair problems that were presented in 3.1 in order to adjust the algortihms parameters and to improve the knowledge about which algorithm and which distance criterion is the most performant, in which case.

Both algorithms are local searches starting from an initial solution. To evaluate their performance in direct plan computation for each planning problem, we generated a series of random plans and ran the algorithms on them. For plan repair, we studied the performance for each removed machine. When a machine is removed, a random solution is first generated before launching the algorithm. The results presented here where made with 20 random runs.

The objective was to compare the two algorithms, and also the two distance criteriria (time MSE and number of assignment changes).

4 Results

The presented results show the average, the standard deviation, the miniimum and the maximum values of the performance criteria. Balance-sheet statistics tables are also presented to compare two algorithms.

4.1 Principle of the balance-sheet statistical results

The balance-sheet tables show whether an algorithm is statistically better than another one. They contain the following probability (in percentage) : let's take any pair of results $x_{a,b}$ and $x_{a',b}$ obtained with each one of the two algorithms a and a' to compare and the same parameters $b = (b_1, b_2...)$; we extract the probability p that the result with the first algorithm is better than the the results with the second algorithm w.r.t. a given performance criterion c :

$$p\left(c(x_{a,b}) > c(x_{a',b}))\right)$$

We also extract the rate of solutions that have same performance

$$p\left(c(x_{a,b}) = c(x_{a',b}))\right)$$

and the rate of identical solutions

$$p\left(x_{a,b} = x_{a',b}\right)$$

The two algorithms are said *equal* if they give the same results,

$$p\left(x_{a,b} = x_{a',b}\right) = 1 \ \forall b$$

The two algorithms are said *equivalent* if they give results of identical performance

$$p\left(c(x_{a,b}) = c(x_{a',b})\right) = 1 \ \forall b$$

Otherwise the best algorithm is the one which has the best probability. When the probability for the 1st algorithm to be better than the 2nd algorithm is equal to the probability that the 2nd is better than the 1st one, one says that there is "no answer" (i.e. one cannot say which one is the best algorithm).

4.2 Using the algorithms for plan computation

In this Section we evaluate the algortihms on direct plan computation instead of plan repair. In other words, the algorithms are used to compute simply a plan from a planning problem instead of modifying a disrupted existing plan. So, there is only one criterion (the makespan C_{max}).

The statistical results obtained are shown in Table 2. The reference C_{max} if the makespan of the optimal solution (or the good solution that we used as the reference plan in problems where there was no available optimal solution (as indicated in Table 1). Note that in the `SWARMS_vehicle` and `Sahara` problems, all the solutions had the same C_{max} (75

and 25 respectively) however they were not always identical for both algorithms. However, these results show that the tabu search algorithm is the most performant at a C_{max} point of view.

4.3 Convergence study

In this Section, we wanted to know which maximal number of iterations one must take for the algorithms. An example for the computation of one plan repair problem with both algos is given in Figure 3. It shows that the makespan obtained at each iteration reaches rapidly its best value, and it increases again after that. Note that this increase is due to the tabu list principle : a given solution cannot be chosen more than once, thus the successive solutions cannot stay in a local minimum.

The trend of the curve 3 is confirmed by the average curve shown in Figure 4. We can conclude that the 15-puzzle rises again a little bit more than the tabu search, and in any case the best is reached before 10 iterations.

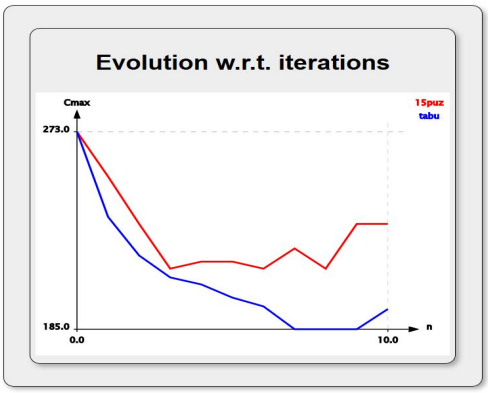


FIGURE 3 – Value of the makespan w.r.t. algo iterations for FixedFJSP problem repair with time MSE as second criterion.

4.4 Multi-criteria study

To evaluate the relevance of the lexicographic multi-critera aggregation, we extracted the number of ex-aequo that were found inside the algorithms. This number can reach 11 (with the `FixedFJSP` planning problem). This means that sometimes during the processing of the algorithm a choice was made on the distance criterion because there were 11 candidates with the same value of the makespan C_{max}. So we conclude that the makespan is not always the unique criterion used to choose between the candidates ; the second distance criterion plays a role.

Figures 5 and 6 show the histograms of the second criterion for both algorithms.

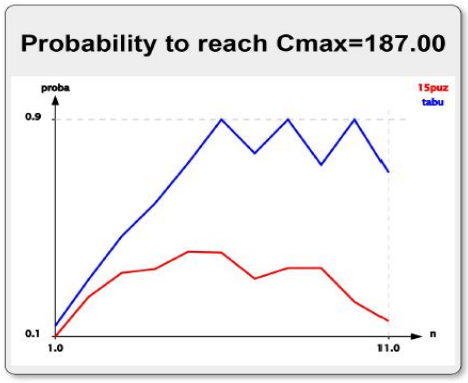


FIGURE 4 – Probability of convergence of the makespan for FixedFJSP problem repair with time MSE as second criterion.

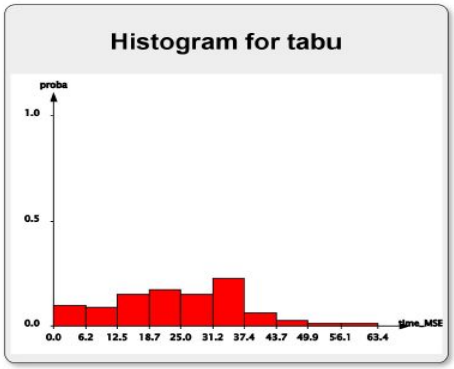


FIGURE 5 – Histogram of the time MSE criterion obtained in repair with the tabu search algorithm.

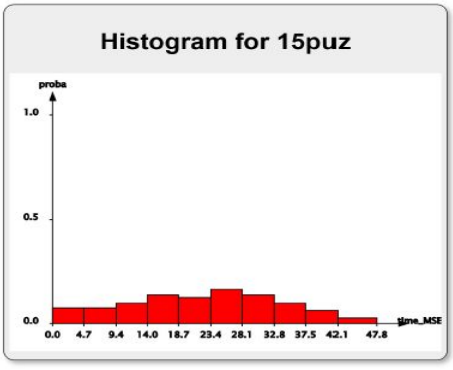


FIGURE 6 – Histogram of the time MSE criterion obtained in repair with the 15-puzzle algorithm.

4.5 Algorithm comparison

Table 3 shows the balance-sheet for the problem `FixedFJSP` with time MSE used as distance criterion. Table 4 shows the same thing when the distance criterion is the number of assignment changes. In these tables, the "Nb best" line shows for each algorithm the number of criteria for which it has been the most performant (0, 1 or 2). One can see that the tabu search has the best performance for the makespan, but not always for the distance criterion.

Tables 5 and 6 summarize which algorithm was the best for each criterion and each planning problem. Note that here again, all the solutions for the `Sahara` problem has the same $C_{max} = 25$.

One can see the trends that are indicated by thoses results : the tabu search is the best w.r.t. the makespan, and the 15-puzzle is the best for the time MSE and sometimes for the number of assignemnt changes.

The Pareto front, i.e. the second criterion w.r.t. the first one, can also be displayed to give a good idea of the solutions behavior (see Figure 7).

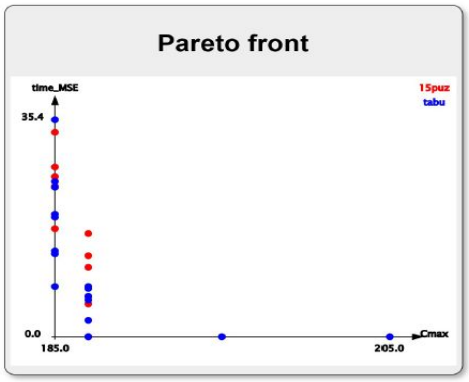


FIGURE 7 – Pareto front obtained with the two algorithms performing plan repair on the FixedFJSP problem with time MSE as plan distance criterion.

5 Explanability in plan repair

Explanability is a new active research topic belonging to both Artificial Intelligence and Human-Computer Interaction. It aims at providing the human user with the information he needs to trust the algorithmic computations. The major question to answer is "why did the algorithm make this decision ?" if it seems different from what the user was expecting intuitively. One difficulty is to guess the user's expectation. But before dealing with the "why ?", explanability begins with a good representation of the processed data to the human when they are complex. This is the case for plans.

5.1 Explaining the obtained solution

Planning solvers dealing with planning problems written in PDDL language represent the obtained plan as a list of actions with attributes. This is hard to interpret for a human. He needs a graphical representation as a Gantt chart, which is well suited to the generalized FJSP problems of the proposed work. Each machine has a row, and the abscissa is the time. Figure 8 shows an example for the `Sahara` planning problem. Here the machines are the drones that have survey tasks to do on targets that are located on a road in Sahara, and one of the tasks is the tracking of a moving vehicle.

Figure 9 shows the result of plan repair from this initial plan with the 15-puzzle algorithm when the drone `UAV09` is suppressed because of a breakdown.

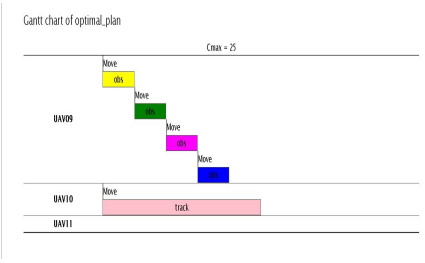


FIGURE 8 – Gannt chart of the initial plan for the Sahara planning problem.

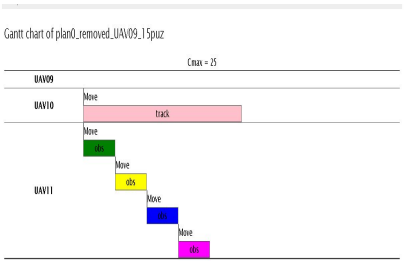


FIGURE 9 – Gantt chart of the repaired plan obtained with the 15-puzzle algorithm from the plan of Figure 8 when the drone AUV09 had a breakdown.

5.2 Towards the explanability of the algorithm decision

The above Section deals with the explanability for information representation. Explaining *why* the plan repair algorithm made a decision is another topic.

In the presented work, there are random initial plans used by the algorithms. Randomness is difficult to explain. In fact the plan repair could also be run on the actual inital

Removed	Algo	Cmax after repair algo 15puz	Time MSE	Mean delay	Assign. changes	Hamming distance	Nb iterations	Nb ex-aequo	Nb ex-aequo max
UAV09	15puz	25	1.73e+01	-6.00e+01	4.00e+00	8	2	1	6

Explanation: 2 actions were assigned to a different machine and 2 actions were reordered on their own machine queue

FIGURE 10 – Explanation provided with the plan repair of Figure 9.

plan even if it is not valid. This would avoid randomness.

Another perspective of work is to analyze and describe what has changed between the initial and the repaired plan (which action has changed assignment, or which has been inserted in a machines queue...)

One could also use the Shapley representation of the contribution of each criterion, which is already known in the multi-criteria decision making domain.

One could also compare a repaired plan with what would obtain the same algorithm without the distance criterion (so optimizing the makespan only).

More generally, one can notice that the plans obtained with solvers using the PDDL description langage are a list of words that are close to sentences of natural language (the subject is the machine, the verb is the action and the object is the job). So it is easy to represent plans in natural language. Note that natural language analysis is already used to extract plans.

6 Conclusion

The obtained benchmark results allow to see that the tabu search is the most performant in simple plan computation at a makespan point of view, but for plan repair the 15-puzzle obtains a better performance in time MSE. But to compare them more deeply, one should implement the multi-criteria function that was used in the algorithms themselves.

The studied convergence behavior of both algorithms shows that taking a maximal number of iterations equal to 10 is enough. So, this study allowed us to improve the parameters used in the algorithms.

Our preliminary results on plan repair explanation are promising. They show that representing the plans and what has changed during the repair is helpful for the human user. Many research possibilities to improve these descriptions and to go further in explaining why the algorithms did so.

The plan distance criteria can be improved. They can be mixed. One can also improve the performance evaluation by showing more why a too distant plan is not good. We can propose a human-related view, since in fact some distance criteria are delays related to the jobs, so they concern the customer satisfaction. And other distance criteria concern the changes in the machines workload, so they concern the stress of the employees.

Références

[1] Patrick Bechon, Magali Barbier, Charles Lesire, Guillaume Infantes, and Vincent Vidal. Using hybrid planning for plan reparation. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE, 2015.

[2] Ripon K Chakrabortty, Ruhul A Sarker, and Daryl L Essam. Multi-mode resource constrained project scheduling under resource disruptions. *Computers & Chemical Engineering*, 88 :13–29, 2016.

[3] Filip Deblaere, Erik Demeulemeester, and Willy Herroelen. Reactive scheduling in the multi-mode rcpsp. *Computers & Operations Research*, 38(1) :63–74, 2011.

[4] Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina. Plan stability : Replanning versus plan repair. In *In Proc. ICAPS*, pages 212–221. AAAI Press, 2006.

[5] LM Gambardella and M Mastrolilli. Effective neighborhood functions for the flexible job shop problem. *Journal of Scheduling*, 3(3), 1996.

[6] Robert P Goldman and Ugur Kuter. Measuring plan diversity : Pathologies in existing approaches and a new plan distance metric. In *AAAI*, pages 3275–3282, 2015.

[7] Li Minglei, Wang Hongwei, and Qi Chao. A novel htn planning approach for handling disruption during plan execution. *Applied Intelligence*, 46(4) :800–809, 2017.

[8] Swarup Kumar Mohalik, Mahesh Babu Jayaraman, Ramamurthy Badrinath, and Aneta Vulgarakis Feljan. Hipr : An architecture for iterative plan repair in hierarchical multi-agent systems. *Journal of Computers*, 13(3) :351–360, 2018.

[9] Tuan Anh Nguyen, Minh Do, Alfonso Emilio Gerevini, Ivan Serina, Biplav Srivastava, and Subbarao Kambhampati. Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence*, 190 :1–31, 2012.

[10] Andrea Rossi. Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships. *International Journal of Production Economics*, 153 :253–267, 2014.

[11] Hélène Soubaras. Reactive multiobjective local search schedule adaptation and repair in flexible job-shop problems. In *Proc. of MCO (Int. Conf. on Modelling, Computation and Optimization of Information Systems and Management Science)*, Metz, France, 11–13 May 2015.

[12] Hélène Soubaras. Méthodes locales pour la réparation en ordonnancement de FJSP avec transport. In *Proc. of ROADEF*, Compiègne, France, 10–12 Feb 2016.

[13] Hélène Soubaras. A new local search algorithm with shortlisting for plan repair in flexible job-shop problems : the 15 puzzle algorithm with potential. In *Proc. of JFPDA*, Grenoble, France, 7–8 Jul 2016.

[14] Hélène Soubaras. Plan repair applied to autonomous underwater vecicle swarms. In *Proc. of CoDIT (Int. Conf. on Control, Decision and information Technologies)*, Paris, France, 23–26 Apr 2019.

[15] Biplav Srivastava, Tuan Anh Nguyen, Alfonso Gerevini, Subbarao Kambhampati, Minh Binh Do, and Ivan Serina. Domain independent approaches for finding diverse plans. In *IJCAI*, pages 2016–2022, 2007.

[16] Guidong Zhu, Jonathan F Bard, and Gang Yu. Disruption management for resource-constrained project scheduling. *Journal of the Operational Research Society*, 56(4) :365–381, 2005.

TABLE 1 – Planning problems that were used in the study.

Problem	Type	Number of jobs	Number of machines	Initial plan is optimal
Small_FJSP	FJSP	3	3	Yes
FixedFJSP	FJSP	8	5	No
SWARMS_UC4	FJSP with moving machines and jobs precedence constraints	4	6	Yes
SWARMS_vehicle	FJSP	3	5	Yes
SWARMS_benchmark	FJSP with moving machines	14	4	No
Sahara	FJSP with moving machines	5	3	Yes

TABLE 2 – Algorithms that had statistically the best C_{max} in plan computation.

Planning problem	$< C_{max} >$ ref	$< C_{max} >$ tabu search	$< C_{max} >$ 15 puzzle	Algo that had the best C_{max} in balance	Identical solutions given by both algos
Small_FJSP	39	44.8	41.55	tabu	0 %
FixedFJSP	187	191.0	185.5	tabu	15 %
SWARMS_UC4	64	66.4	65.4	tabu	55 %
SWARMS_vehicle	75	75.0	75.0	equivalent	50 %
SWARMS_benchmark	1618	1953.0	1983.65	tabu	0 %
Sahara	25	25.0	25.0	equivalent	60 %

TABLE 3 – Balance-sheet of benchmarks comparing the two algorithms in repair in problem FiedFJSP with the time MSE as distance critrerion.

criterion	algo=15puzzle is better	algo=tabu is better	both are equal	best
C_{max}	24	64	13	tabu
time MSE	54	45	1	15puz
Nb best	1	1	0	

TABLE 4 – Balance-sheet of benchmarks comparing the two algorithms in repair in problem FiedFJSP with the number of assignment changes as distance critrerion.

criterion	algo=15puzzle is better	algo=tabu is better	both are equal	best
C_{max}	24	61	16	tabu
assign. changes	14	29	57	tabu
Nb best	0	2	0	

TABLE 5 – Algorithms that had statistically the best C_{max} and best time MSE in plan repair with the time MSE used as distance criterion.

Planning problem	$< C_{max} >$ tabu	$< C_{max} >$ 15 puzzle	Best C_{max}	Best time MSE	Identical solutions given by both algos
Small_FJSP	80.8				100 %
FixedFJSP	170.72	176.8	tabu	15 puzzle	28 %
SWARMS_UC4	79.83				100 %
SWARMS_vehicle	82.6				100 %
SWARMS_benchmark	1969.66	2018.96	tabu	15 puzzle	40 %
Sahara	25				100 %

TABLE 6 – Algorithms that had statistically the least number of assignment changes in plan repair when it is used as distance criterion.

Planning problem	Best number of assigment changes
Small_FJSP	(no answer)
Small_FJSP	tabu
SWARMS_UC4	equal
SWARMS_vehicle	equal
SWARMS_benchmark	15 puzzle
Sahara	(no answer)

Planification Monte Carlo orientée information

Vincent Thomas[1] Gérémy Hutin[2] Olivier Buffet[1]

[1] Université de Lorraine, INRIA, CNRS, LORIA, Nancy, FRANCE
[2] École Normale Supérieure, Lyon, FRANCE

{prénom.nom@(inria.fr|ens-lyon.fr)}

Résumé

Dans cet article, nous nous intéressons à la résolution de problèmes de collecte active d'information exprimés sous la forme de ρ-POMDP, une extension des Processus Décisionnels de Markov Partiellement Observables (POMDP) dont la récompense ρ dépend de l'état de croyance. Des approches utilisées pour résoudre les POMDP ont déjà été étendues au cadre ρ-POMDP lorsque la récompense ρ est convexe ou lipschtizienne, mais ces approches ne permettent pas de résoudre toutes les instances de ρ-POMDP. Afin de proposer un algorithme en-ligne efficace qui s'affranchit des contraintes sur ρ, cet article se concentre sur les méthodes à base de recherche arborescente Monte Carlo et cherche à adapter POMCP à la résolution de ρ-POMDP. Comme les récompenses dépendent de l'état de croyance, il est nécessaire de modifier POMCP (i) pour échantillonner plusieurs états lors des trajectoires suivies et (ii) pour éviter les biais dans l'estimation des valeurs. Des expériences ont été conduites pour étudier les propriétés de l'approche proposée.

Mots Clef

Planification dans l'incertain, recherche active d'information, Approche MCTS, POMDP, ρ-POMDP.

Abstract

In this article, we address the issue of solving information-gathering problems expressed as ρ-POMDPs, an extension of Partially Observable Markov Decision Processes (POMDPs) whose reward ρ depends on the belief state. Approaches already used for solving POMDPs have been extended to solving ρ-POMDPs as belief MDPs when the reward ρ is convex in $\mathcal{B}$ or when it is Lipschitz-continuous (LC), using respectively PWLC or LC approximators. In the present paper, we build on the POMCP algorithm to propose a Monte Carlo Tree Search for ρ-POMDPs, aiming for an efficient online planner. Adaptations are required due to the belief-dependent rewards to (i) propagate more than one state at a time, and (ii) prevent biases in value estimates. Experiments are conducted to analyze the approach at hand.

Keywords

Planning under uncertainty, active information gathering, MCTS, POMDP, ρ-POMDP.

1 Introduction

De nombreux algorithmes de l'état de l'art pour la résolution de *processus de décision markoviens partiellement observables* (POMDP) consistent à transformer le problème en un problème «complètement observable»—plus précisément un *belief MDP* (ou MDP sur les états de croyance)—et à exploiter la linéarité par morceaux et la convexité de la fonction de valeur optimale dans l'espace d'état de ce nouveau problème (ici l'espace des croyances $\mathcal{B}$) en maintenant des approximateurs de fonction généralisant. Cette approche a été étendue à la résolution de ρ-POMDP, c.-à-d. des problèmes dont le critère de performance dépend de la croyance (par exemple en collecte active d'information), quand ρ elle-même est convexe en $\mathcal{B}$ [1] ou quand ρ est lipschitzienne [10]. Dans cet article, nous proposons un nouvel algorithme pour résoudre des ρ-POMDP qui ne nécessite ni la convexité ni la Lipschitz-continuité de ρ, mais repose sur de l'échantillonnage Monte Carlo et est inspiré par POMCP [17]. Cet algorithme utilise des filtres particulaires et échantillonne des particules pendant les trajectoires pour estimer les états de croyance visités B et les récompenses associées $\rho(B)$. Cet algorithme et ses variantes sont évaluées empiriquement sur divers problèmes de collecte d'information.

L'article est organisé comme suit. La section 2 discute de travaux connexes sur le contrôle orienté information. La section 3 présente un état de l'art (i) d'abord sur les processus de décision markoviens (MDP) et les recherches arborescentes Monte Carlo (MCTS), (ii) puis sur les POMDP et l'algorithme *Partially Observable Monte Carlo Planning* (POMCP), une approche MCTS pour la résolution de POMDP et, (iii) enfin sur les ρ-POMDP. La section 4 décrit notre contribution : deux algorithmes, ρ-beliefUCT et ρ-POMCP, pour la résolution de ρ-POMDP avec des approches MCTS. Enfin, la section 5 présente les expérimentations conduites et analyse les résultats avant de conclure et de donner quelques perspectives.

2 Travaux connexes

Les premiers travaux sur le contrôle orienté information (IOC) impliquaient des problèmes formalisés soit (i) comme des POMDP (comme EGOROV, KOCHENDERFER et UUDMAE [9] l'ont fait récemment, puisqu'une récompense dépendant de l'observation peut trivialement être reformulée comme une récompense dépendant de l'état), ou (ii) avec des récompenses dépendant de la croyance (et des méthodes de résolution essentiellement ad-hoc comme [11, 14]).

ARAYA-LÓPEZ et al. [1] proposent les ρ-POMDP pour formaliser facilement de nombreux—si ce n'est la plupart—problèmes IOC. Ils montrent qu'un ρ-POMDP dont la fonction de récompense dépendant de la croyance ρ est convexe peut être résolu avec des solveurs de POMDP à base de points modifiés qui exploitent toujours la propriété PWLC (avec des bornes d'erreurs qui dépendent de la qualité de l'approximation PWLC de ρ).

Le cadre POMDP-IR (*POMDP with Information Reward*) de SPAAN, VEIGA et LIMA [19] permet de décrire des problèmes IOC avec des récompenses linéaires en b qui sont démontrés comme équivalents aux ρ-POMDP «PWLC» (c.-à-d. quand ρ est PWLC) [16]. Dans les deux cas les techniques de résolution proposées sont des solveurs de POMDP modifiés. Pour sa part, le cadre général des ρ-POMDP permet de formaliser plus de problèmes, par exemple en spécifiant directement un critère reposant sur l'entropie de Shannon.

Plus récemment, FEHR et al. [10] ont appliqué la même approche que ARAYA-LÓPEZ et al. [1], mais pour des récompenses croyance-dépendantes lipschitziennes (LC) plutôt que convexes. Ils ont démontré que, quand ρ est lipschitzienne, alors la fonction de valeur optimale est aussi lipschitzienne à horizon fini. Elle peut alors être minorée (et majorée) par des bornes uniformément améliorables, et deux algorithmes reposant sur HSVI ont été proposés pour tirer parti de ces encadrements.

A contrario, le présent article ne repose pas sur des approximateurs de la fonction de valeur généralisant (PWLC ou LC). N'importe quelle récompense croyance-dépendante peut ainsi être considérée. Dans ce but, nous avons décidé de tirer parti des recherche arborescentes Monte Carlo (MCTS), en particulier de l'algorithme Partially Observable Monte Carlo Planning (POMCP) [17] qui ne repose pas sur la propriété PWLC. MCTS et POMCP présentent en outre plusieurs autres intérêts : (i) ils ne requièrent pas un modèle complet mais seulement un simulateur ; (ii) contrairement aux recherches heuristiques, ils ne requièrent pas d'initialisations optimistes ou pessimistes de la fonction de valeur ; et (iii) ils se sont montrés très efficaces pour résoudre des problèmes avec de très grands espaces d'action et d'observation.

3 État de l'art

3.1 MDP et algorithmes par échantillonage

Processus de décision markoviens Un processus de décision markovien (MDP) [15] est défini par un tuple $\langle S, A, P, r\rangle$ où S est l'ensemble fini des états du système ; A est l'ensemble fini des actions possibles ; la fonction de transition P donne la probabilité de transiter d'un état s à un autre s' quand une action a est accomplie : $P_a(s, s') = Pr(s'|s, a)$; et $r(s, a, s')$ est la récompense scalaire instantanée reçue pendant cette transition. Résoudre un MDP consiste à trouver une politique—une application $\pi : S \to A$—maximisant un critère de performance (dans notre cas, la somme γ-atténuée des récompenses).

Algorithmes en-ligne Dans cet article, nous nous concentrerons sur les algorithmes *en-ligne*. À chaque pas de temps, un tel algorithme estime la meilleure action à effectuer. Le système effectue alors cette action et reçoit de l'information de l'environnement pour mettre à jour sa connaissance de l'état courant. Cet algorithme en-ligne est ensuite appelé à nouveau pour décider de la prochaine action à accomplir. Les approches en-ligne ont l'avantage de restreindre les états considérés à ceux qui sont atteignables, et peuvent se concentrer seulement sur ceux qui apparaissent pertinents.
Un premier algorithme naïf consiste à appliquer la programmation dynamique à l'arbre complet des futurs possibles jusqu'à une certaine profondeur. Les valeurs sont alors remontées depuis les feuilles pour estimer la valeur de chaque action à la racine. Toutefois, cet algorithme souffre d'une explosion combinatoire du fait de la croissance exponentielle du nombre de nœuds avec la profondeur considérée.

Algorithmes reposant sur l'échantillonnage Une première amélioration a été faite avec *Sparse Sampling* (SS) [12], lequel n'échantillonne qu'un nombre limité de prochains états après chaque action, et estime les valeurs des actions par une moyenne. Cela réduit grandement le temps de calcul et requiert non pas un modèle complet du processus, mais seulement un simulateur. Toutefois SS échantillonne des trajectoires uniformément sans considérer les valeurs estimées, alors qu'il pourrait être amélioré en se concentrant sur les parties prometteuses de l'arbre, ce qui est le but des recherches arborescentes Monte Carlo (MCTS).

Approches MCTS Dans les approches MCTS [7, 13, 6], l'arbre représentant les futurs possibles à partir d'un état de départ est construit d'une manière incrémentale non-uniforme. Chaque itération consiste en 4 étapes :

- **sélection** : une trajectoire est échantillonnée dans l'arbre selon une stratégie d'exploration jusqu'à ce qu'un nœud n'appartenant pas à l'arbre soit atteint ;
- **expansion** : ce nouveau nœud est ajouté à l'abre ;
- **simulation** : la valeur de ce nouveau nœud est estimée en échantillonnant une trajectoire depuis ce nœud suivant

une politique *rollout* (de «déroulement») ;

rétro-propagation : cette estimation et les récompenses reçues durant l'étape de sélection sont alors rétro-propagées aux nœuds visités pour mettre à jour leurs statistiques (valeur et nombre de visites).

UCT MCTS a été appliqué avec succès pour résoudre des MDP [13]. Dans *Upper Confidence Bound applied to trees* (UCT), la stratégie d'exploration utilisée pour échantillonnée une trajectoire dans l'arbre repose sur *Upper Confidence Bounds* (UCB1). À chaque nœud état, cette stratégie sélectionne l'action qui maximise la valeur estimée augmentée d'un bonus d'exploration $c_{t,s} = C_p\sqrt{\frac{logN(s)}{N(s,a)}}$, où C_p est une constante d'exploration qui doit être choisie correctement, $N(s)$ est le nombre de visites passées du nœud s, et $N(s,a)$ est le nombre de fois où l'action a a été choisie dans le nœud s. Ce bonus d'exploration dépend du nombre de visites et garantit que toutes les actions seront sélectionnées dans le future, même si l'une d'entre elles a une valeur estimée plus grande que les autres. De plus, cette stratégie de sélection d'action fait converger asymptotiquement les valeurs d'actions estimées vers la valeur optimale.

3.2 POMDP

Un POMDP [3] est défini par un tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, P, r, \gamma, b_0 \rangle$, où $\mathcal{S}$, $\mathcal{A}$ et $\mathcal{Z}$ sont des ensembles finis d'états, d'actions et d'observations ; $P_{a,z}(s,s')$ donne la probabilité de transiter vers l'état s' en observant z quand l'action a est appliquée en l'état s ($P_{a,z}$ est une matrice $\mathcal{S} \times \mathcal{S}$) ; $r(s,a) \in \mathbb{R}$ est la récompense associée à l'accomplissement de l'action a dans l'état s ; $\gamma \in [0;1)$ est un facteur d'atténuation ; et b_0 est l'état de croyance initial—c.-à-d. la distribution de probabilité initiale sur les états possibles. L'objectif est alors de trouver une politique π qui prescrit des actions en fonction des actions et observations passées de façon à maximiser l'espérance de la somme des récompenses atténuées (ici avec un horizon temporel infini).

Pour cela, un POMDP est souvent transformé en un belief MDP $\langle \Delta, \mathcal{A}, P, r, \gamma, b_0 \rangle$, où Δ est le simplexe des états de croyance possibles, $\mathcal{A}$ est le même ensemble d'actions, et $P_a(b,b') = P(b'|b,a)$ et $r(b,a) = \sum_s b(s)r(s,a)$ sont les fonctions de transition et de récompense induites. Ce cadre permet de considérer des politiques $\pi \colon \Delta \to \mathcal{A}$, chacune étant associée à sa fonction de valeur $V^\pi(b) \stackrel{\text{def}}{=} E[\sum_{t=0}^{\infty} \gamma^t r(b_t, \pi(b_t))|b_0 = b]$. Les politiques optimales maximisent V^π dans tous les états de croyance accessibles depuis b_0. Leur fonction de valeur V^* est le point fixe de l'opérateur *d'optimalité* de Bellman ($\mathcal{H}$) [4] $\mathcal{H}V : b \mapsto \max_a[r(b,a) + \gamma \sum_z \|P_{a,z}b\|_1 V(b^{a,z})]$, et agir de manière gourmande par rapport à V^* fournit une telle politique.

3.3 MCTS pour POMDP

SILVER et VENESS [17] ont proposé l'algorithme *Partially Observable Monte Carlo Planning* (POMCP) pour

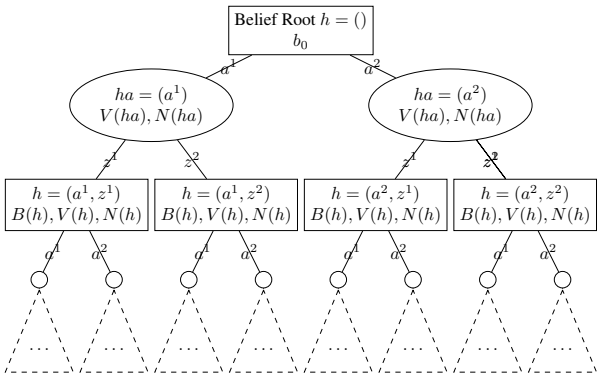


FIGURE 1 – Exemple d'un arbre de croyances avec 2 actions a^1 et a^2, et 2 observations z^1 et z^2. Chaque nœud croyance est représenté par un nœud carré et associé à un historique $h = (a_0, z_0, \ldots, z_t)$. Pendant son exécution, POMCP va calculer $B(h)$, une estimation de son état de croyance, et $V(h)$, une estimation de sa valeur optimale. Il comptera aussi le nombre des visites passées $N(h)$. Chaque nœud-action est représenté par une ellipse et associé à un historique ha. POMCP va calculer une valeur estimée $V(ha)$ et compter le nombre de visites $N(ha)$.

appliquer MCTS à la résolution de POMDP. Un POMDP est abordé à travers le belief MDP correspondant, un arbre d'états de croyance étant fait d'une alternance de nœuds action et de nœuds croyance comme présenté dans la figure 1. Le chemin jusqu'à un nœud croyance à profondeur t suit l'historique action-observation $h_t = (a_0, z_0, a_1, z_1, \ldots, z_t)$ allant de la croyance racine à cette croyance $b(h_t)$.

Appliquer directement UCT au belief MDP échantillonnerait des trajectoires $(b_0, a_0, b_1, a_1, \ldots, b_t)$ dans l'espace des croyances, requiérant ainsi le modèle POMDP complet et un coût computationnel élevé pour calculer des états de croyance exacts. Pour éviter ce coût, POMCP échantillonne des trajectoires dans l'espace d'états, ce qui ne requiert qu'un simulateur comme modèle génératif du POMDP. Étant donné un état s et une action a, ce simulateur $\mathcal{G}$ fournit un état successeur s', une observation z, et une récompense r supposée échantillonnés d'après le vrai modèle sous-jacent $(s', z, r) \sim \mathcal{G}(s,a)$.

Pendant la phase de sélection, POMCP utilise ce modèle génératif pour échantillonner une trajectoire : le premier état est échantillonné à partir de l'estimation de la croyance à la racine, puis on alterne entre (i) choisir une action suivant UCB1 (appliqué aux valeurs d'action estimées dans le nœud croyance courant), et (ii) échantillonner un prochain état, une observation et une récompense selon le modèle génératif $\mathcal{G}$. En accumulant des valeurs dans les nœuds croyance, faire une moyenne sur toutes les trajectoires simulées donne une estimation de la valeur $V(h)$ du nœud croyance h, même si seules des trajectoires sur les états ont été échantillonnées.

En outre, les états sont collectés dans chaque nœud croyance visité de manière à estimer la prochaine croyance à la racine quand une action est effectivement appliquée.

En évitant le calcul des croyances exactes $b(h)$ dans chaque nœud croyance, POMCP permet d'aborder des problèmes plus grands, tout en préservant les garanties de convergence asymptotiques comme UCT.
SILVER et VENESS [17] ont aussi proposé l'algorithme PO-UCT comme une première étape vers POMCP. PO-UCT diffère de POMCP dans la façon de calculer l'état de croyance de la nouvelle racine : au lieu de collecter des états échantillonnés, quand une action est effectivement appliquée et une observation reçue, alors l'état de croyance est calculé avec une mise à jour de Bayes exacte.

3.4 ρ-POMDP

Les ρ-POMDP [1] diffèrent des POMDP en ce que leur fonction de récompense ρ dépendent de l'état de croyance, permettant ainsi de définit non seulement des critères orientés contrôle, mais aussi des critères orientés information, généralisant ainsi les POMDP. La récompense immédiate ρ est naturellement définie comme $\rho(b, a, b')$, la récompense immédiate associée à la transition de la croyance b à la croyance b' après avoir effectué l'action a. Par simplicité et sans perte de généralité, cet article utilise la notation $\rho(b, a)$ (même si les problèmes considérés reposent sur $\rho(b, a, b')$). Il est aisé d'étendre les algorithmes proposés en raisonnant sur $\rho(b, a, b')$ au lieu de $\rho(b, a)$.
Comme présenté dans les travaux connexes, ARAYA-LÓPEZ et al. [1] et FEHR et al. [10] ont exploité les récompenses ρ PWLC et lipschitziennes pour résoudre des ρ-POMDP généraux. Toutefois, si de nombreux problèmes peuvent être modélisés avec des récompenses ρ convexes ou lipschitziennes, cela nous laisse avec de nombreux problèmes qui ne peuvent être résolus avec des approximations similaires.
Ici, nous proposons d'utiliser l'approche MCTS pour aborder le cas ρ-POMDP général sans contrainte sur la fonction ρ. À titre d'exemple, considérons un agent surveillant un musée et dont l'objectif est de localiser un visiteur avec une certitude donnée. Si X dénote la variable aléatoire pour la position du visiteur et b_X la croyance associée, alors la fonction de récompense suivante

$$\rho_X(b, a) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{si } \|b_X\|_\infty > \alpha \\ 0, & \text{sinon} \end{cases}$$

récompense les distributions dont la probabilité maximale est plus grande que $\alpha \in [0, 1]$. Il s'agit d'une fonction seuil, donc ni convexe, ni lipschitzienne du fait de sa discontinuité là où $\|b_X\|_\infty = \alpha$.

4 Algorithmes MCTS pour ρ-POMDP

POMCP et sa variante PO-UCT ne peuvent être appliqués directement à la résolution de ρ-POMDP. PO-UCT ne calcule pas les croyances exactes durant ses étapes de sélection et de rétro-propagation (mais seulement à la racine de l'arbre), et ne peut donc calculer les récompenses croyance-dépendantes le long d'une trajectoire. POMCP génère des trajectoires en ne suivant qu'une séquence d'états et échantillonne des récompenses liées à ces états. Il collecte les états visités dans les nœuds croyance, mais ceux-ci ne sont pas suffisants pour correctement estimer les états de croyance et calculer précisément les récompenses croyance-dépendantes comme requis par les ρ-POMDP.

4.1 Algorithme ρ-beliefUCT

Le premier algorithme proposé dans cet article est appelé *ρ-beliefUCT*. Il consiste à appliquer directement UCT au belief MDP dérivé du ρ-POMDP. Cela requiert d'avoir accès au modèle ρ-POMDP complet et de calculer des croyances exactes pour chaque nœud croyance de l'arbre visité en effectuant la mise à jour de Bayes :

$$b_{haz}(s') \propto \sum_{s \in S} P_{a,z}(s, s').b_h(s).$$

Pendant l'étape de simulation, une politique rollout dépendant de la croyance (par opposition à une politique aléatoire) a besoin de calculer non seulement la croyance à chaque choix d'action, mais aussi les récompenses immédiates $\rho(b, a)$ pour estimer la valeur du nœud ajouté. Cela induit un important coût computationnel.
ρ-beliefUCT peut toutefois tirer parti de plusieurs éléments. D'abord, puisque chaque historique correspond à un unique état de croyance, chaque état de croyance n'a besoin d'être calculé qu'une fois quand le nœud correspondant est ajouté. Deuxièmement, alors que, dans POMCP, $r(b, a)$ est estimé comme une moyenne non biaisée parce que $r(b, a)$ est linéaire en b, dans notre contexte, $\rho(b, a)$ est une fonction déterministe de la croyance : sa valeur exacte peut être mémorisée directement dans le nœud action. Enfin, on peut échantillonner une observation pendant l'étape de sélection sans calculer la probabilité d'observation $P(z|b(h))$. Cela peut être fait sans biais en échantillonnant d'abord un état s de la croyance courante $b(h)$, puis un état s' et une observation z du modèle génératif du POMDP (ou en employant la distribution de probabilité $P_{z,a}(s, s')$ si elle est disponible).
ρ-beliefUCT est un algorithme de référence intéressant. Toutefois, pour éviter (i) le coût associé au calcul exact des croyances, et (ii) le besoin d'un modèle complet du POMDP, nous proposons un autre algorithme, ρ-POMCP, qui ne calcule pas les croyances exactes, mais les estime en collectant des états échantillonnés dans les nœuds croyance.

4.2 Algorithme ρ-POMCP

L'algorithme ρ-POMCP est similaire à POMCP. Pendant l'étape de sélection, les trajectoires sont générées en échantillonnant des états et des observations à l'aide du modèle génératif du POMDP $\mathcal{G}$. Lors de leur visite, chaque nœud croyance h collecte l'état qui a conduit à ce nœud de manière à construire une estimation du vrai état de croyance $b(h)$ correspondant à cette historique.

Appliquer POMCP directement en n'ajoutant que l'état courant de la trajectoire au nœud croyance sans plus de considération conduira toutefois à de très mauvaises estimations des récompenses immédiates, lesquelles amèneraient l'algorithme (i) à dépenser inutilement du temps de calcul en se concentrant sur des branches avec des récompenses surestimées et (ii) à ralentir l'exploration de branches intéressantes dont les récompenses seraient sous-estimées.

Nous proposons ainsi d'ajouter non pas un, mais plusieurs états à chaque nœud visité pendant l'étape de sélection. Ainsi, à chaque transition, quand l'action a est sélectionnée à partir de l'état de la trajectoire s et que l'observation z est échantillonnée, l'algorithme génère un ensemble β de $|\beta|$ états, appelé un *petit sac de particules*, en utilisant le modèle génératif pour faire $|\beta|$ échantillonnages cohérents avec l'observation z. Ensuite, il ajoute toutes les particules de ce petit sac β à $B(haz)$, appelé dans ce contexte le *sac cumulé* de l'historique haz, et stocké dans le nœud croyance correspondant : $B(haz) \leftarrow B(haz) \cup \beta$.

Avec un modèle génératif $\mathcal{G}$, seules les approches par réjection peuvent être considérées pour produire des échantillonnages cohérents. Dans ce cas, un couple $(\tilde{s}', \tilde{z})$ est échantillonné du modèle génératif $(\tilde{s}', \tilde{z}) \sim \mathcal{G}(s, a)$ et $\tilde{s}'$ est gardé dans le sac généré si et seulement si l'observation échantillonnée $\tilde{z}$ correspond à l'observation z de la trajectoire suivie. Ce processus s'accomplit jusqu'à ce que le nombre requis d'échantillonnages cohérents soit atteint (et les états correspondants soient stockés dans β). Ceci peut toutefois être coûteux en temps de calcul, en particulier quand l'observation z a une faible probabilité étant donnés h et a.

Échantillonnage selon l'importance Pour éviter ce coût computationnel, nous utilisons l'échantillonnage selon l'importance [8] au lieu de l'échantillonnage par réjection, ce qui suppose que la fonction d'observation est disponible, et conduit à pondérer chaque particule. Après avoir effectué l'action a_t et reçu l'observation z_t pendant l'étape de sélection, un nouveau petit sac β_{t+1} est généré à partir de β_t en utilisant le modèle génératif et en suivant les étapes suivantes $|\beta|$ fois : (1) échantillonner un état $\tilde{s}$ de β_t ; (2) échantillonner un état $\tilde{s}'$ en appliquant le modèle génératif sur $\tilde{s}$, $\tilde{s}' \sim \mathcal{G}(\tilde{s}, a_t)$; (3) stocker cet état $\tilde{s}'$ dans β_{t+1} et lui associer le poids correspondant à la probabilité $P(z|\tilde{s}, a, \tilde{s}')$ d'avoir généré l'observation z. Les poids dans deux petits sacs associés au même nœud peuvent être comparés, et on peut donc accumuler de tels sacs dans le nœud croyance correspondant. Pour économiser de la mémoire, quand un petit sac β est ajouté à un *sac cumulé* $B(h)$, les particules correspondant au même état sont fusionnées et leurs poids additionnés. Enfin, pendant les diverses descentes d'une exécution de ρ-POMCP, le nombre de particules accumulées dans le *sac cumulé* $B(h)$ stocké dans le nœud croyance h croit, donnant une meilleure estimation de la vraie croyance $b(h)$. On n'a alors qu'à normaliser les poids pour obtenir une distribution de probabilité valide dès que nécessaire.

Notons que l'état s_t employé dans la génération de la trajectoire $h_t = (s_0, a_0, \ldots, a_{t-1}, s_t)$ au pas de temps t est aussi inclus dans le petit sac β_t pendant la génération de particules, de sorte que l'observation générée avec la trajectoire peut être expliquée au moins par une particule dans le nouveau sac. Alors que cela introduit un biais dans les petits sacs β de cette trajectoire, la distribution des états de la trajectoire dans les *sacs cumulés* n'est pas biaisée puisque ces états sont obtenus par échantillonnage depuis la croyance initiale.

Lors de l'étape de simulation, des estimations de croyance sont requises ne serait-ce que pour estimer les récompenses instantanées, et éventuellement pour prendre des décisions. En conséquence, la séquence de petits sacs β doit être perpétuée.

Enfin, pendant l'état de rétro-propagation, le *sac cumulé* de particules $B(h)$ présent dans un nœud croyance est utilisé comme estimation du vrai état de croyance $b(h)$ pour calculer la récompense ρ. Les poids ne sommant pas à 1 dans le *sac cumulé*, l'estimation de la croyance requiert une normalisation ; mais les poids restent intacts dans le sac de sorte que de nouvelles particules peuvent être ajoutées sans introduire de biais.

La description ci-dessus conduit à l'algorithme 1. Dans celui-ci, la notation $\{s\} \cup \beta \overset{1+n}{\sim} I$ implique que $1 + n$ états sont échantillonnés de I, dont un est stocké dans s et les n autres dans β. La fonction *PF* correspond au processus de filtrage particulaire décrit précédemment, et $w_{s'} = P(z|s, a, s')$.

On notera que, comme dans POMCP ou dans des filtres particulaires standards, quand le système évolue effectivement (une action étant accomplie réellement), l'état réel peut ne pas être dans le sac cumulé de la nouvelle racine, et il peut arriver que l'observation reçue ne puisse être expliquée par aucun état contenu dans ce sac. Nous ne discuterons pas plus avant de cette situation, mais il reste possible de ré-estimer la croyance courante en simulant le processus depuis la croyance initiale jusqu'au pas de temps en question.

4.3 Variantes de ρ-POMCP

La différence courante entre POMCP et ρ-POMCP se trouve dans la façon dont les particules sont collectées afin d'estimer la récompense obtenue à chaque transition, alors que les mises à jour des valeurs restent identiques. Celles-ci peuvent toutefois être améliorées, conduisant à plusieurs variantes de ρ-POMCP puisque, lors de son exécution, ρ-POMCP construit des estimations de la croyance $b(h)$ de qualité croissante dans les nœuds croyance visités. Pour ce faire, nous revenons d'abord sur ce qui est calculé dans les mises à jour effectuée par POMCP (et la version de base de ρ-POMCP), puis proposons plusieurs variantes.

Calculs effectués par POMCP Si on ignore l'initialisation du nœud dans POMCP, alors, pour un couple nœud-action ha, la valeur stockée dans $V(ha)$ fait la moyenne,

Algorithm 1: ρ-POMCP

/* Le texte en rouge souligne les différences avec POMCP. */

Fct SEARCH (h)
  repeat
    if $h = empty$ **then** $\{s\} \cup \beta \overset{1+n}{\sim} I$
    else $\{s\} \cup \beta \overset{1+n}{\sim} B(h)$
    SIMULATE $(s, \beta, h, 0)$
  until TIMEOUT()
  return $\arg\max_b V(hb)$

Fct ROLLOUT (s, β, h, δ)
  if $\gamma^\delta < \epsilon$ **then return** 0
  $a \sim \pi_{rollout}(h, \cdot)$
  $(s', z) \sim \mathcal{G}(s, a)$
  $\beta' \leftarrow$PF$(\beta, a, z) \cup \{(s', w_{s'})\}$
  return $\rho(\beta, a) + \gamma$.ROLLOUT $(s', \beta', haz, \delta + 1)$

Fct SIMULATE (s, β, h, δ)
  if $\gamma^\delta < \epsilon$ **then return** 0
  if $h \notin T$ **then**
    forall $a \in \mathcal{A}$ **do**
      $T(ha) \leftarrow (N_{init}(ha), V_{init}(ha), \emptyset)$
    return ROLLOUT (s, β, h, δ)
  $a \leftarrow \arg\max_b V(hb) + c\sqrt{\frac{\log N(h)}{N(hb)}}$
  $(s', z) \sim \mathcal{G}(s, a)$
  $\beta' \leftarrow$PF$(\beta, a, z) \cup \{(s', w_{s'})\}$
  $B(h) \leftarrow B(h) \cup \beta$
  $R \leftarrow \rho(B(h), a) + \gamma$.SIMULATE $(s', \beta', haz, \delta + 1)$
  $N(h) \leftarrow N(h) + 1$
  $N(ha) \leftarrow N(ha) + 1$
  $V(ha) \leftarrow V(ha) + \frac{R - V(ha)}{N(ha)}$
  return R

sur $N(ha)$ échantillons (/visites), de :

— $\sum_{s \in \beta_{ha}} r(s, a)$: la récompense totale sur les états s qui ont été échantillonnés quand l'action a a été choisie dans h (β_{ha} dénote cet ensemble d'états) ;

— $\sum_{z \in \mathcal{Z}_{ha}} \mathbf{1}_{N(h,a,z) \geq 1} Rollout(haz)$: le retour total des rollouts générés à partir de chaque observation z échantillonnée après avoir choisi a dans h (ensemble $\mathcal{Z}_{ha}$), où $N(h, a, z)$ est le nombre de fois où l'action a a été suivie de l'observation z dans le nœud h (alors que $N(haz)$ est le nombre de mises à jour du nœud haz) ;

— $\sum_{z \in \mathcal{Z}_{ha}} \sum_{a' \in \mathcal{A}} N(haza')V(haza')$: la somme des valeurs de tous les nœuds fils, pondérée par leurs nombres de visites (ce qui inclut aussi les rollouts effectués depuis ces nœuds).

En introduisant les estimations de valeurs des nœuds croyance $V(h)$, initialisées avec les valeurs des rollouts (pour $N(h) = 1$), cela conduit aux formules suivantes :

$$V(h) \leftarrow \frac{1}{N(h)} \left[Rollout(h) + \sum_{a' \in \mathcal{A}} N(ha')V(ha') \right] ;$$

$$V(ha) \leftarrow \frac{1}{N(ha)} \left[\sum_{s \in \beta_{ha}} r(s, a) + \gamma \sum_{z \in \mathcal{Z}_{ha}} N(haz)V(haz) \right] .$$

Last-reward-update ρ-POMCP Ainsi, dans POMCP, $V(ha)$ est une moyenne mobile qui *«contient»* une estimation de $r(b(h), a)$, cette estimation étant calculée comme $\frac{\sum_{s \in \beta_{ha}} r(s,a)}{N(ha)}$.
Dans ρ-POMCP de base, la valeur de r au pas de temps courant est remplacée par l'estimation de $\rho(b(h))$ comme $\rho(\hat{B}_{N(h)}(h))$, où $\hat{B}_{N(h)}(h)$ est le sac cumulé après les $N(h)$ premières visites. Dans ce cas, $V(ha)$ inclut ainsi (entre autres choses) une moyenne d'estimations successives (c.-à-d. qu'il calcule $\sum_{i=1}^{N(ha)} \rho(\hat{B}_{\phi(i)}(h), a)$, où $\phi(i)$ est la $i^{\text{ème}}$ visite de h où a a été sélectionnée). Il semblerait pourtant plus approprié d'employer à la place la récompense associée à la dernière estimation de la croyance, $\rho(\hat{B}_{\phi(N(ha))}(h), a)$, laquelle est une estimation moins biaisée.
Cette variante, appelée *last-reward-update ρ-POMCP* (ou *"lru-ρ-POMCP"*), corrige cela assez simplement en remplaçant la mise à jour de $V(ha)$ dans l'algorithme 1 par

$$V(ha) \leftarrow \frac{N(ha) - 1}{N(ha)} [V(ha) - \rho^{prev}(h, a)] + \rho(B(h), a) + \frac{1}{N(ha)} \gamma.\text{SIMULATE}(s', haz, \delta + 1),$$

où $\rho^{prev}(h, a)$ est la précédente valeur de la récompense quand ha a été rencontrée pour la dernière fois (valeur qui doit donc être mémorisée).

Last-value-update ρ-POMCP De la même manière, $V(ha)$ «contient» aussi des estimations des récompenses moyennes des futurs pas de temps, lesquelles souffrent du

même problème. Pour réparer cela, SIMULATE devrait retourner non pas un échantillon de retour, mais une estimation du retour moyen.
Les mises à jour dans l'étape de rétro-propagation consistent ainsi à ré-estimer toutes les valeurs des nœuds croyance visités en utilisant la récompense obtenue à chaque transition et le rollout initial qui doit avoir été mémorisé auparavant. Ceci est accompli dans la variante *last-value-update ρ-POMCP* (ou *"lvu-ρ-POMCP"*) avec la formule suivante :

$$V(h) \leftarrow \frac{1}{N(h)}\left[Rollout(h) + \sum_a N(ha)V(ha)\right],$$
$$V(ha) \leftarrow \rho(B(h),a) + \frac{\gamma}{N(ha)}\sum_z \left[N(haz).V(haz)\right].$$

Le processus résultant est montré dans l'algorithme 2.

Algorithm 2: Last-value-update ρ-POMCP

Fct SEARCH (h)
  /* Untouched/Identical */
Fct ROLLOUT (s, β, h, δ)
  /* Untouched/Identical */
Fct SIMULATE (s, β, h, δ)
  if $\gamma^\delta < \epsilon$ **then return** 0
  if $h \notin T$ **then**
    forall $a \in \mathcal{A}$ **do** $T(ha) \leftarrow (0, 0, \emptyset)$
    $N(h) \leftarrow 1$
    $V(h) \leftarrow rollout(h) \leftarrow$ ROLLOUT (s, β, h, δ)
    return $V(h)$
  $a \leftarrow \arg\max_b V(hb) + c\sqrt{\frac{\log N(h)}{N(hb)}}$
  $(s', z, r) \sim \mathcal{G}(s, a)$
  $\beta' \leftarrow$PF$(\{s\} \cup \beta, a, z)$
  $V(haz) \leftarrow$SIMULATE $(s', \beta', haz, \delta + 1)$
  $B(h) \leftarrow B(h) \cup \beta$
  $N(h) \leftarrow N(h) + 1$
  $N(ha) \leftarrow N(ha) + 1$
  $V(ha) \leftarrow \rho(B(h), a) + \frac{\gamma}{N(ha)}[\sum_z N(haz)V(haz)]$
  $V_h \leftarrow \frac{1}{N(h)}.[rollout(h) + \sum_a N(ha)V(ha)]$
  return V_h

5 Expérimentations

5.1 Bancs d'essai

Les POMDP étant une sous-classe des ρ-POMDP, les premiers bancs d'essai que nous considérons sont le problème du tigre (tel que décrit sur la page POMDP de Cassandra) et de petites instances du problème *Rock Sampling* [18] avec une grille de 4×4 cellules, 4 pierres à échantillonner et une récompense de 100 (respectivement -100) pour l'échantillonnage d'une bonne (resp. mauvaise) pierre.
Il est connu que, dans de nombreux problèmes de collecte d'information, une simple stratégie myope donne souvent de très bons résultats [5]. Pour évaluer les algorithmes proposés, nous considérons des problèmes où celles-ci rencontrent des difficultés.

Nous proposons ainsi le *museum problem* (problème du musée) inspiré par [16], dans lequel un agent doit déterminer la position d'un visiteur dans un environnement torique (4×4 dans nos expérimentations). L'état correspond à la position inconnue du visiteur. À chaque pas de temps, le visiteur reste immobile avec probabilité $0,6$, et bouge vers l'une de ses 4 cellules voisines avec probabilité $0,4$ (en choisissant uniformément au hasard). En guise d'action, l'agent peut activer une caméra dans l'une des cellules possibles. Il reçoit alors une observation déterministe qui peut être *"present"* quand le visiteur est à cette position, *"close"* quand il est sur une cellule voisine, et *"absent"* si le visiteur est plus loin. En outre : la récompense immédiate correspond à la néguentropie de la distribution de probabilité sur la position du visiteur $\rho(b, a, b') = -H(b') = \sum_s b'(s).log(b'(s))$; la croyance initiale sur la position du visiteur est une distribution uniforme sur toutes les cellules ; et γ est fixé à $0,95$. L'intérêt de ce problème repose sur le grand nombre d'actions (une par cellule). Nous avons aussi employé une variante, *Museum Threshold*, comme proposé en section 3.4, où la récompense repose sur une fonction seuil de l'état de croyance (avec $\alpha = 0,8$). La récompense est alors nulle dans la plupart de l'espace de croyance (c.-à-d., partout où $max_s b(s) \leq 0,8$).

Les prochains problèmes appartiennent à la classe des problèmes de (auto-)localisation. L'agent est placé dans une grille torique discrète où les cellules sont colorées en noir ou blanc. À chaque pas de temps, l'agent peut se déplacer vers une cellule voisine. Il reçoit alors une observation correspondant à la couleur de la cellule atteinte. En outre : observations et transitions sont déterministes ; la croyance initiale de l'agent est uniforme sur toutes les cellules blanches de la grille ; la récompense reçue correspond à la différence d'entropie entre b et b' : $\rho(b, a, b') = -H(b') + H(b)$; et $\gamma = 0,9$. Plusieurs configurations ont été étudiées, comme présenté sur la figure 2 : (i) *MazeCross*, où les cellules noires forment une croix, rendant facile l'autolocalisation ; (ii) *MazeLines*, où la grille est divisée en deux régions : la première région contient des rayures et l'autre ne contient qu'une cellule noire ; l'agent ne peut se localiser dans la régions rayée du fait des ambiguïtés et doit planifier plusieurs pas de temps à l'avance pour chercher le point dans la région vide ; (iii) *MazeHole*, où des cellules noires sont séparées par des cellules blanches et une cellule noire manque ; l'agent doit raisonner un pas à l'avance pour chercher la cellule noire manquante dans cette configuration régulière : un agent myope sur une cellule noire ne voit aucun intérêt à se déplacer dans une direction plutôt qu'une autre puisqu'un seul pas ne lui apportera pas d'information ; et (iv) *MazeDots*, qui est la même configuration que MazeHole mais où les cellules noires sont séparées par plusieurs cellules blanches.

Les problèmes de *localisation active* emploient les mêmes

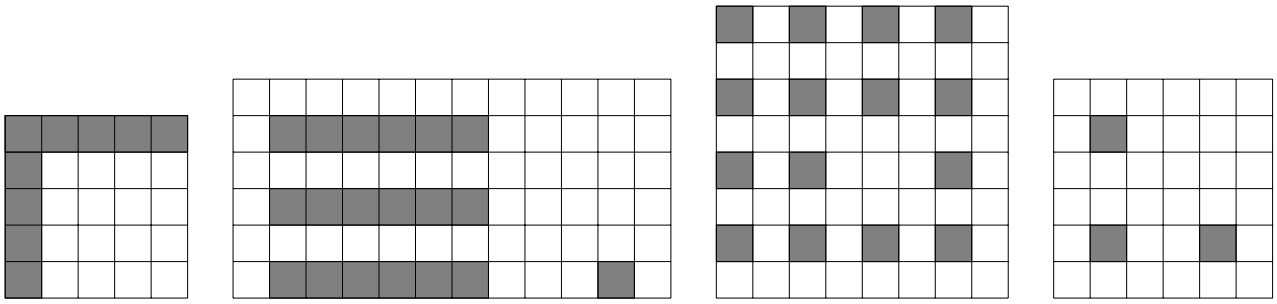


FIGURE 2 – Les diverses configurations de cellules employées pour les problèmes de localisation passive ou active ; de gauche à droite : *MazeCross*, *MazeLines*, *MazeHole* et *MazeDots*.

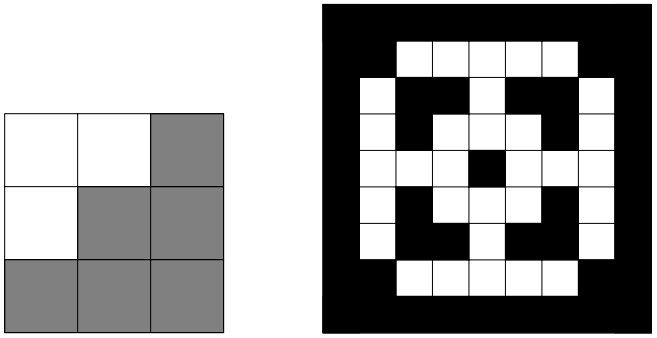


FIGURE 3 – (gauche) la configuration de cellules employée pour *GridX* et *GridNotX*, et (droite) la configuration d'obstacles pour *SeekAndSeek*.

labyrinthes, récompenses et γ que les problèmes de localisation, mais l'agent ne reçoit une information concernant la couleur de sa cellule courante non plus quand il se déplace, mais quand il effectue une action *«observer»* spécifique. La localisation active est ainsi difficile pour les stratégies myopes qui ne voient aucun bénéfice (immédiat) à se déplacer.

GridX et *GridNotX* [10] diffèrent des premiers problèmes de localisation en ce que : (i) la position initiale peut être n'importe quelle cellule (pas juste les blanches) ; (ii) les déplacements (n,s,e,w) réussissent avec une probabilité $0,8$, sinon l'agent reste immobile ; (iii) $\gamma = 0,95$; et (iv) en notant b_x (resp. b_y) la croyance sur l'abscisse x (resp. l'ordonnée y), la fonction de récompense est $\rho(b) = +\|b_x - \frac{1}{3}\mathbf{1}\|_1$ (resp. $-\|b_x - \frac{1}{3}\mathbf{1}\|_1$) pour *GridX* (resp. *GridNotX*). *GridNotX* est un problème particulièrement intéressant parce que son objectif est de maximiser l'incertitude, ce qui ne peut être modélisé avec des récompenses dépendant de l'état.

Dans le problème *SeekAndSeek*, un objet est perdu dans un labyrinthe avec des obstacles restreignant les déplacements possibles de l'agent. L'agent peut se déplacer dans le labyrinthe et reçoit des observations spécifiques selon que l'objet est à la même position (*present*), à côté de celle-ci (*close*), ou dans les autres cas (*absent*). La récompense immédiate reçue est la néguentropie sur les positions possibles de l'objet $\rho(b, a, b') = -H(b') + H(b)$. L'objet étant immobile, quand un agent myope est dans un cul-de-sac, il n'a aucune motivation pour explorer l'environnement puisque revenir en arrière sur ses pas ne lui apporte pas d'information.

Nous avons aussi abordé le problème de diagnostic *CameraClean* [2] qui a été conçu pour empêcher les stratégies myopes d'être efficaces. Dans ce problème une caméra robot peut être orientée pour prendre une photo dans l'une des quatre zones existantes et doit trouver un objet immobile dans l'une d'elles. L'état contient la position de l'objet, la zone cible actuelle, et un booléen spécifiant si la lentille est propre ou non. Les actions consistent pour la caméra à tourner (de manière déterministe) vers la prochaine zone, photographier la zone courante, ou nettoyer sa lentille. Quand la caméra prend une photo, sa qualité dépend du statut de la lentille. Si elle est propre, la probabilité de bien observer la présence de l'objet dans la zone cible est de $0,8$ (la probabilité de faux-positifs et faux-négatifs est de $0,2$), et cette probabilité est de $0,5$ quand la lentille est sale. La récompense est la différence de néguentropie entre les croyances de départ et d'arrivée $\rho(b, a, b') = -H(b') + H(b)$

Toutes les expérimentations ont été conduites sur des cœurs Intel Xeon Gold 6130 à 2.1GHz avec 512Mo.

5.2 Influence des rollouts

Des expérimentations préliminaires ont été conduites avec des rollouts aléatoires. Dans de nombreux problèmes, la politique de rollout aléatoire requiert un important temps

de calcul. Pour un même nombre de descentes, la durée observée pour ρ-POMCP avec des rollouts aléatoires (interrompus quand $\gamma^{depth} < 0.01$) est entre 5 et 10 fois plus élevée que sans aucun rollout (en fixant la valeur des nouveaux nœuds à 0). De plus, dans la majorité des problèmes, utiliser une politique de rollout complètement aléatoire n'a pas d'avantage significatif. Dans certains problèmes, les performances observées sont même dégradées (comme dans le problème du *Tigre*). Nous avons donc préféré nous concentrer d'abord sur ρ-POMCP et ρ-beliefUCT sans rollouts, et observerons l'influence de la politique de rollout en section 5.6.

5.3 Comparaison avec des stratégies myopes

La table 1 présente de premiers résultats comparant la stratégie *aléatoire pure*, les stratégies *look-ahead*, ρ-beliefUCT, et ρ-POMCP sur les problèmes proposés.

Random correspond à une stratégie dans laquelle les actions sont sélectionnées d'une manière complètement aléatoire. Les algorithmes *look-ahead-H* effectuent une programmation dynamique sur tous les futurs possibles pour un horizon temporel fini H en employant le modèle POMDP complet. Ainsi, la stratégie myope pure, dans laquelle l'agent accomplit une action pour maximiser la récompense immédiate, correspond à *Look-ahead-1*, alors que *Look-ahead-3* correspond à une stratégie qui tient compte des toutes les conséquences trois pas de temps à l'avance. La colonne ρ-POMCP correspond à l'algorithme ρ-POMCP avec un nombre fixe de descentes $nb_{descents} = 10\,000$, sans aucun rollout (en fixant la valeur des nouveaux nœuds à 0), avec échantillonnage selon l'importante et $|\beta| = 50$. Enfin, la colonne ρ-beliefUCT correspond à l'algorithme ρ-beliefUCT avec 10 000 descentes. ρ-POMCP et ρ-beliefUCT utilisent tous deux une constante UCB spécifique pour chaque problème comme spécifié dans la table 1 (usuellement correspondant à $(R_{max} - R_{min})/(1-\gamma)$).

(a) La première chose à souligner est que, à part pour *GridX* et *Museum*, les problèmes posés requièrent une stratégie plus élaborée qu'une stratégie myope simple puisque *Look-ahead*-3 obtient de meilleurs résultats que *Look-ahead*-1. La stratégie myope donne aussi de bons résultats pour *Museum Threshold* : même si les récompenses non-nulles sont rares, l'existence, pour chaque action, d'une observation qui désambiguise complètement l'état (quand le visiteur est observé par la caméra) est suffisante pour guider une stratégie myope telle que *Look-ahead-1*.

(b) Quand le problème requiert plus qu'une simple stratégie myope, ρ-POMCP et ρ-beliefUCT donnent de meilleurs résultats que *Look-ahead*-1 et des résultats similaires à *Look-ahead*-3. Dans la plupart des cas, comme le montre l'erreur standard, la différence entre *Look-ahead*-1 et *ρ-POMCP* est significative. Dans certains problèmes, la différence entre *Look-ahead*-3 et *ρ-POMCP* est aussi significative (*SeekAndSeek* et *RockSampling* de manière évidente, mais aussi *GridNotX*, *Mazeline* et *ActiveLocLines*).

(c) On notera aussi que ρ-POMCP et ρ-beliefUCT donnent les mêmes résultats pour ce nombre de descentes et prennent beaucoup plus de temps que les simples algorithmes *Look-ahead* (sauf quand le nombre d'actions devient important comme dans *RockSampling* et *Museum*). ρ-beliefUCT est généralement plus rapide que ρ-POMCP, mais cela dépend plus précisément du problème puisque les coûts de calcul de ces deux algorithmes viennent de différentes opérations. Dans ρ-beliefUCT, ce coût est dû au calcul des croyances exactes à chaque fois qu'un nouveau nœud croyance est ajouté à l'arbre. Dans ρ-POMCP, ce coût est dû à la génération de petits sacs β à chaque transition le long des trajectoires. C'est pourquoi, alors que le temps requis par ρ-POMCP est plus régulier (sauf pour le cas non élucidé de *GridX*), le temps requis par ρ-beliefUCT dépend fortement de la croyance considérée et du nombre d'états possibles (par exemple, le calcul de la croyance est rapide dans les problèmes *Tiger* et *CameraClean*, mais requiert plus de temps dans les problèmes de *localisation active* où les états de croyance b incluent de nombreux états s de probabilité non-nulle $b(s) \neq 0$). La section 6 discute du temps de calcul élevé de ρ-POMCP en proposant de possibles améliorations.

Ces premiers résultats suggèrent donc d'abord que les bancs d'essai choisis sont appropriés, et que ρ-POMCP et ρ-beliefUCT sont des approches intéressantes. La prochaine étape a été d'étudier l'influence de la taille des sacs de particules échantillonnés $|\beta|$ pour un budget temps fixe.

5.4 Influence de $|\beta|$

Des expériences ont été conduites pour évaluer l'influence de $|\beta|$, la taille des sacs de particules générés le long des trajectoires. Nous avions l'intuition que, quand $|\beta|$ est petit avec un nombre fixe de descentes, les états de croyance seraient mal approchés et ρ-POMCP donnerait de moins bons résultats qu'avec une grande valeur de $|\beta|$. Toutefois, avec un nombre fixe de 10 000 descentes, il s'avère que $|\beta|$ n'a pas d'influence notable. Avec tant de descentes, la petite valeur de $|\beta|$ est compensée par le grand nombre de trajectoires échantillonnées.

Avec un plus petit nombre de descentes, l'influence de $|\beta|$ apparaît, comme visible table 2, où le nombre de descentes a été fixé à $nb_{descents} = 1000$. Il est difficile d'interpréter précisément ces résultats, mais nous allons essayer de fournir des pistes d'explications générales.

(a) Dans certains problèmes (comme *GridX*, *Museum* et quelques labyrinthes : *MazeCross*, *MazeHole*, *MazeDots* et *ActiveLoc Cross*), on n'observe aucune influence de $|\beta|$ sur la performance. Pour ceux-ci, la performance avec un petit $|\beta|$ est déjà proche de la performance observée table 1 avec une valeur de $|\beta|$ de 50 et 10 000 descentes. Ce sont des problèmes faciles pour lesquels *Look-ahead-1* fournit déjà de bonnes performances ; ils ne requièrent donc pas de bonnes estimations de nœuds croyance profonds pour obtenir une stratégie efficace.

(b) Au contraire, $|\beta|$ a une influence significative dans des problèmes où *Look-ahead-1* a des difficultés, comme

TABLE 1 – Résultats obtenus sur des épisodes de 40 actions en exécutant l'algorithme en-ligne correspondant avant chaque action. V correspond à la performance γ-atténuée moyenne sur tous les épisodes ; Err est l'erreur standard ; nb_{xp} est le nombre d'expériences effectuées ; et $t(s)$ est la durée moyenne d'un épisode.

	Random			Look-ahead-1			Look-ahead-3			ρ-beliefUCT			ρ-POMCP		
Problem (UCB cst)	$V \pm Err$	nb_{xp}	$t(s)$	$V \pm Err$	nb_{xp}	$t(s)$	$V \pm Err$	nb_{xp}	$t(s)$	$V \pm Err$	nb_{xp}	$t(s)$	$V \pm Err$	nb_{xp}	$t(s)$
Tiger (360)	-122.13 ± 2.68	200	0.00	1.86 ± 0.13	200	0.01	2.06 ± 0.12	200	0.04	2.03 ± 0.12	200	9.90	2.05 ± 0.12	200	97.31
CameraClean (14)	0.24 ± 0.01	100	0.02	0.19 ± 0.01	100	0.26	0.55 ± 0.03	100	4.84	0.58 ± 0.03	100	12.44	0.56 ± 0.02	100	92.64
SeekAndSeek (69.3)	-41.04 ± 1.97	100	0.02	-44.02 ± 2.04	100	14.57	-39.86 ± 2.36	100	17.53	-24.66 ± 1.60	100	55.93	-25.29 ± 1.69	100	112.45
Museum entropy (1)	-26.39 ± 0.38	100	0.01	-16.98 ± 0.42	100	0.11	-16.44 ± 0.42	100	103.42	-17.26 ± 0.42	100	38.90	-15.70 ± 0.42	100	73.28
Museum threshold (1)	1.70 ± 0.08	200	0.01	6.36 ± 0.17	200	0.11	6.34 ± 0.16	200	105.55	6.55 ± 0.17	200	46.20	6.32 ± 0.18	200	119.81
GridX (26)	16.38 ± 0.23	100	0.01	21.68 ± 0.06	100	0.02	21.67 ± 0.06	100	0.28	21.69 ± 0.05	100	88.99	21.63 ± 0.05	100	527.64
GridNotX (26)	-16.77 ± 0.14	200	0.01	-4.06 ± 0.16	200	0.02	-3.61 ± 0.15	200	0.24	-3.19 ± 0.14	200	17.93	-3.16 ± 0.15	200	89.39
MazeCross (3.2)	1.63 ± 0.03	200	0.00	2.21 ± 0.01	200	0.02	2.22 ± 0.01	200	0.25	2.20 ± 0.01	200	28.73	2.20 ± 0.01	200	72.78
MazeLines (4.3)	1.52 ± 0.04	200	0.01	2.34 ± 0.03	200	0.03	2.63 ± 0.03	200	0.80	2.73 ± 0.03	200	84.02	2.74 ± 0.03	200	76.99
MazeHole (4.2)	1.27 ± 0.03	200	0.01	1.73 ± 0.05	200	0.03	2.02 ± 0.04	200	0.70	2.04 ± 0.04	200	98.06	1.99 ± 0.04	200	88.26
MazeDots (3.6)	1.35 ± 0.05	200	0.01	1.98 ± 0.04	200	0.03	2.09 ± 0.04	200	0.41	2.16 ± 0.04	200	55.79	2.06 ± 0.04	200	79.26
ActivlocCross (3.2)	0.87 ± 0.03	200	0.01	1.45 ± 0.01	200	0.02	2.07 ± 0.02	200	0.20	2.08 ± 0.02	200	37.61	2.10 ± 0.02	200	49.77
ActivlocLines (4.3)	0.63 ± 0.03	200	0.01	1.17 ± 0.04	200	0.04	2.01 ± 0.04	200	0.63	2.21 ± 0.04	200	97.85	2.25 ± 0.04	200	60.81
ActivlocHole (4.2)	0.40 ± 0.04	200	0.01	0.97 ± 0.07	200	0.02	1.42 ± 0.05	200	0.31	1.37 ± 0.05	200	56.65	1.44 ± 0.05	200	63.73
ActivlocDots (3.6)	0.65 ± 0.03	200	0.01	1.19 ± 0.05	200	0.03	1.79 ± 0.04	200	0.57	1.65 ± 0.03	200	84.42	1.75 ± 0.04	200	68.95
RockSampling 44 (100)	-8.58 ± 4.17	100	0.00	0.00 ± 0.00	100	0.04	0.00 ± 0.00	100	38.00	108.25 ± 7.08	100	20.97	109.96 ± 6.53	100	79.31

TABLE 2 – Influence de $|\beta|$ pour un nombre fixe de 1000 descentes. Chaque colonne correspond aux résultats obtenus sur 200 expériences en exécutant ρ-POMCP avec différentes valeurs de $|\beta|$ sur des épisodes de 40 actions. V est la valeur γ-atténuée cumulée moyenne, Err l'erreur standard, et $t(s)$ la durée moyenne d'une expérience.

	$\lvert\beta\rvert = 0$		$\lvert\beta\rvert = 1$		$\lvert\beta\rvert = 2$		$\lvert\beta\rvert = 5$		$\lvert\beta\rvert = 10$		$\lvert\beta\rvert = 50$		$\lvert\beta\rvert = 100$	
Problem (UCB cst)	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$
Tiger (360)	0.18 ± 0.23	0.55	1.76 ± 0.12	0.71	-0.11 ± 0.15	0.84	-3.95 ± 0.03	1.23	-4.00 ± 0.00	1.84	-4.00 ± 0.00	6.63	-4.00 ± 0.00	12.01
CameraClean (14)	0.17 ± 0.01	0.57	0.21 ± 0.01	0.66	0.32 ± 0.01	0.76	0.49 ± 0.01	1.05	0.55 ± 0.02	1.55	0.58 ± 0.02	6.81	0.56 ± 0.02	12.88
SeekAndSeek (69.3)	-33.79 ± 1.57	0.70	-33.45 ± 1.59	0.75	-35.33 ± 1.44	0.91	-33.96 ± 1.47	1.31	-32.51 ± 1.45	1.94	-31.17 ± 1.27	8.06	-29.01 ± 1.28	14.51
Museum entropy (1)	-17.06 ± 0.31	0.57	-17.18 ± 0.30	0.60	-16.57 ± 0.31	0.58	-17.21 ± 0.32	0.79	-17.15 ± 0.30	1.15	-16.87 ± 0.29	5.53	-17.16 ± 0.31	10.38
Museum threshold (1)	6.05 ± 0.16	0.61	5.95 ± 0.19	0.70	6.08 ± 0.19	0.78	6.29 ± 0.18	0.98	5.82 ± 0.16	1.48	6.01 ± 0.18	6.62	6.25 ± 0.18	12.28
GridX (26)	21.64 ± 0.04	0.47	21.61 ± 0.04	0.60	21.70 ± 0.04	0.71	21.68 ± 0.04	0.99	21.61 ± 0.04	1.49	21.69 ± 0.04	6.31	21.67 ± 0.04	11.41
GridNotX (26)	-3.78 ± 0.17	0.46	-3.83 ± 0.16	0.58	-3.72 ± 0.16	0.70	-3.76 ± 0.15	0.95	-3.58 ± 0.16	1.45	-3.56 ± 0.16	6.03	-3.35 ± 0.15	11.22
MazeCross (3.2)	2.20 ± 0.01	0.51	2.18 ± 0.01	0.60	2.20 ± 0.01	0.66	2.21 ± 0.01	0.99	2.18 ± 0.01	1.51	2.22 ± 0.01	5.66	2.20 ± 0.01	10.40
MazeLines (4.3)	2.46 ± 0.03	0.59	2.48 ± 0.03	0.70	2.53 ± 0.03	0.82	2.54 ± 0.03	1.14	2.64 ± 0.03	1.73	2.67 ± 0.03	6.03	2.64 ± 0.03	11.08
MazeHole (4.2)	1.91 ± 0.05	0.65	1.87 ± 0.05	0.76	1.93 ± 0.04	0.90	1.98 ± 0.04	1.23	2.02 ± 0.04	1.84	1.98 ± 0.04	6.66	2.01 ± 0.04	12.17
MazeDots (3.6)	2.01 ± 0.04	0.54	2.02 ± 0.04	0.67	2.01 ± 0.04	0.77	2.05 ± 0.04	1.06	2.05 ± 0.04	1.62	2.05 ± 0.04	6.12	2.04 ± 0.04	11.31
ActivlocCross (3.2)	2.09 ± 0.02	0.50	2.09 ± 0.02	0.55	2.09 ± 0.02	0.62	2.08 ± 0.02	0.83	2.12 ± 0.02	1.19	2.12 ± 0.02	3.85	2.10 ± 0.02	7.03
ActivlocLines (4.3)	1.83 ± 0.04	0.62	1.93 ± 0.04	0.71	1.95 ± 0.04	0.82	2.07 ± 0.04	1.10	2.08 ± 0.04	1.50	2.10 ± 0.04	5.29	2.08 ± 0.04	9.33
ActivlocHole (4.2)	1.34 ± 0.02	0.67	1.40 ± 0.03	0.79	1.41 ± 0.03	0.94	1.49 ± 0.03	1.36	1.57 ± 0.04	1.89	1.63 ± 0.03	6.12	1.67 ± 0.03	10.42
ActivlocDots (3.6)	1.22 ± 0.05	0.59	1.14 ± 0.05	0.72	1.24 ± 0.05	0.82	1.31 ± 0.05	1.09	1.36 ± 0.05	1.57	1.43 ± 0.05	5.10	1.52 ± 0.06	9.49
RockSampling 44 (100)	100.43 ± 4.76	0.53	90.86 ± 4.67	0.52	96.86 ± 4.50	0.61	94.65 ± 4.81	0.93	86.29 ± 4.70	1.50	72.59 ± 4.43	7.37	70.51 ± 4.46	13.09
RockSampling 44 (800)	56.29 ± 3.72	0.53	60.71 ± 3.89	0.52	50.18 ± 3.75	0.60	64.28 ± 4.15	0.92	65.59 ± 4.24	1.53	60.75 ± 4.00	7.41	50.26 ± 3.72	13.22

dans les problèmes de localisation active *ActivLoc Lines*, *ActivLoc Dots*, *GridNotX*, *SeekAndSeek* et *CameraClean*. Dans ce cas, le petit nombre de descentes ne permet pas à ρ-POMCP de construire de bonnes approximations des vrais états de croyance et des valeurs des nœuds croyance. En ce qui concerne plus spécifiquement *GridNotX*, ρ est concave et de petites valeurs de $|\beta|$ pourraient empêcher MCTS de se concentrer sur une action intéressante. Dans ce cas, les récompenses, et donc les valeurs, vont être initialement sous-évaluées et, dans certaines situations, cela empêchera l'algorithme d'essayer l'action optimale même si, sur le long terme, ce phénomène sera compensé par le bonus d'exploration.

(c) De façon surprenante, dans certains cas (comme *RockSampling 44* et *Tiger*), on observe que de petites valeurs de $|\beta|$ donnent de meilleurs résultats. Nous comptons étudier ce phénomène, mais une explication possible serait que de petites valeurs de $|\beta|$ donnent parfois des sur-estimations importantes des valeurs de nœuds croyance, ce qui favorise l'exploitation des branches associées et donc une recherche plus profonde.

La prochaine étape consiste en l'analyse des résultats avec un budget temps fixe pour voir si de petites valeurs de $|\beta|$ sont compensées par la possibilité d'échantillonner plus de trajectoires.

5.5 Résultats avec budget temps fixe

Les tables 3 et 4 (voir annexe) présentent des résultats concernant l'influence de $|\beta|$ avec un budget temps fixe (respectivement de 1s et 100ms).

Avec un budget temps fixe et un petit $|\beta|$, seules peu de particules sont ajoutées aux sacs cumulés $B(h)$ des nœuds croyance visités lors d'une trajectoire. Toutefois, dans ce cas, ρ-POMCP effectue de nombreuses descentes, ajoutant

de nombreuses particules sur le long terme, et peut donc obtenir de bonnes estimations des états de croyance. À l'inverse, quand $|\beta|$ est grand, chaque descente ajoute de nombreuses particules dans les nœuds croyances rencontrés, les états de croyance sont plus précisément approchés, et la sélection des trajectoires échantillonnées devrait être guidée par des valeurs estimées plus précises, mais l'algorithme effectue moins de descentes.

Dans les deux cas (budget temps de 100ms et 1s), quand $|\beta|$ croît, le nombre de descentes effectuées par ρ-POMCP décroît naturellement, mais cela ne semble pas avoir d'impact sur les valeurs cumulées γ-atténuées à l'exécution. Pour de très grandes valeurs de $|\beta|$ avec 100ms de budget temps, les expériences n'atteignent pas toutes la fin de l'épisode. Cela arrive quand une action réelle génère une observation qui n'a pas été échantillonnée dans l'arbre. Dans ce cas, l'algorithme n'a pas d'estimation de la prochaine croyance racine et s'arrête (voir discussion pour des propositions concernant ce point). Ce phénomène s'observe dans des problèmes avec de nombreuses actions (comme *Museum*) puisque, dans ce cas, le nombre de descentes est très faible par rapport au nombre d'actions à explorer.

Des comportements différents s'observent dans deux problèmes. Dans *CameraClean*, il y a une différence de performance significative entre $|\beta| = 0$ et $|\beta| = 5$ avec un budget temps de 100ms. Cela pourrait venir du processus d'observation hautement stochastique de *CameraClean*, lequel nécessiterait de meilleures estimations de l'état de croyance pour agir correctement. Dans *Rocksampling*, on observe, comme quand ρ-POMCP effectue un nombre fixe de descentes, que de petites valeurs de $|\beta|$ donnent de meilleurs résultats. Cela pourrait être dû à des sur-estimations favorisant l'exploitation et au fait que de petits valeurs de $|\beta|$ avec un budget de temps fixe favorisent un grand nombre de descentes avec une plus grande profondeur. Mais ce cas précis doit encore être étudié plus en détails.

5.6 Variantes de ρ-POMCP

Nous avons enfin regardé les résultats pour différentes variantes de ρ-POMCP avec budget temps fixe. Ces variantes correspondent à différents types de rollouts : les rollouts aléatoires (les actions étant choisies aléatoirement) et les rollouts myopes (les action étant choisies selon une politique *look-ahead-1*—sélectionnant une action maximisant la récompense immédiate). Elles correspondent aussi à *Last-Reward update* et *Last-Value update* comme présenté en section 4.3. Nous nous sommes concentrés sur un budget temps de 500ms afin d'avoir assez de temps pour exécuter ρ-POMCP avec suffisamment de descentes pour les rollouts coûteux (tels que les myopes).

La table 5 (voir annexe) présente les résultats obtenus avec différents rollouts. Comme attendu, le nombre de descentes décroît quand $|\beta|$ croît. Aussi, pour une même valeur de $|\beta|$, le nombre de descentes décroît de *constant-rollout* à *random-rollout* et de *random-rollout* à *myopic-rollout* à cause du temps nécessaire à l'exécution des rollouts complexes. Dans le problème *museum entropy*, du fait du petit nombre de descentes, plusieurs épisodes se terminent sur une observation non-anticipée.

La table 6 (voir annexe) présente des résultats obtenus avec ρ-POMCP de base contre les versions avec mise à jour *lru* et *lvu* pour différentes valeurs de $|\beta|$. Nous nous attendions à ce que les mises à jour *lru* et *lvu* donnent de meilleurs résultats que la mise à jour ρ-POMCP de base, en particulier quand $|\beta|$ est petit, mais, comme le montre cette table, ce phénomène n'est pas observé ici.

6 Discussion

Dans cet article, nous avons proposé deux algorithmes, ρ-POMCP et ρ-beliefUCT, afin d'aborder la résolution de ρ-POMDP sans contrainte sur la fonction de récompense ρ. Des expérimentations simples ont montré que ρ-POMCP et ρ-beliefUCT donnent des résultats intéressants dans ces problèmes, mais qu'on n'observe pas de différences significatives avec plusieurs variantes de ρ-POMCP (modifiant les rollouts ou les mises à jour). Ces résultats ouvrent cependant plusieurs directions de recherche.

$|\beta|$ dynamique pour ρ-POMCP Nous avons observé que ρ-POMCP est coûteux en temps de calcul du fait des nombreuses particules échantillonnées Pourtant, après un certain nombre d'itérations, les nœuds croyance proches de la racine pourraient contenir assez de particules pour estimer leur état de croyance précisément. Une direction prometteuse serait d'économiser du temps en faisant croître les sacs cumulés $B(h)$ sous-linéairement (plutôt que linéairement) avec le nombre de visites $N(h)$. Cela nécessitera de modifier la façon dont les particules sont générées, ce qui n'est pas une tâche aisée puisque les particules sont générées du nœud croyance racine et doivent être propagées jusqu'à la fin de chaque trajectoire de façon (i) à exécuter une politique de rollout non-aléatoire et (ii) à construire de meilleures estimations des états de croyance aux nœuds feuilles.

À propos des transitions réelles imprévues Un autre problème qui concerne aussi POMCP et le filtrage particulaire est de gérer les transitions imprévues. Si, après avoir effectivement accompli une action a, l'observation réelle z a été rarement (voire pas du tout) échantillonnée, la nouvelle racine pourrait n'être dotée que d'une mauvaise estimation de la croyance, voire ne pas exister du tout, de sorte que les calculs en ligne donneront de mauvais résultats s'ils peuvent seulement être accomplis. Dans ce cas, effectuer un filtrage particulaire depuis le père du nouveau nœud racine pourrait fournir une nouvelle croyance racine mais à un coût de calcul élevé. Une autre amélioration complémentaire serait d'échantillonner régulièrement des particules de la croyance initiale du POMDP en suivant la trajectoire réelle suivie par l'agent pour ajouter de nouvelles particules à la racine courante et se prémunir de l'appauvrissement le long des épisodes.

Travaux futurs Nous espérons que les approches MCTS telles que ρ-POMCP fourniront une manière d'affronter des problèmes avec des grands nombres d'états, d'actions ou d'observations. Les travaux futurs incluent le fait des considérer des ρ-POMDP continus (c.-à-d. avec des états, actions ou observations continues), par exemple en partant des travaux de SUNBERG et KOCHENDERFER, SUNBERG et KOCHENDERFER [20, 21], afin d'aborder des problèmes de contrôle orientés information dans un contexte robotique.

Remerciements Les expériences présentées dans cet article ont été menées sur Grid5000, qui bénéficie du support d'un groupe d'intérêt scientifique hébergé par Inria, et incluant le CNRS, RENATER et quelques universités et organisations (voir https://www.grid5000.fr). Enfin, nous remercions les relecteurs pour leurs remarques constructives intégrées dans cet article.

Références

[1] M. ARAYA-LÓPEZ, O. BUFFET, V. THOMAS et F. CHARPILLET. « A POMDP Extension with Belief-dependent Rewards ». In : *Advances in Neural Information Processing Systems 23 (NIPS-10)*. [see also companion techreport]. 2010.

[2] M. ARAYA-LÓPEZ. « Near-Optimal Algorithms for Sequential Information-Gathering Decision Problems ». Thèse de doct. Université de Lorraine, fév. 2013.

[3] K. ÅSTRÖM. « Optimal control of Markov processes with incomplete state information ». In : *Journal of Mathematical Analysis and Applications* 10.1 (1965). ISSN : 0022-247X.

[4] R. BELLMAN. « A Markovian Decision Process ». In : *Journal of Mathematics and Mechanics* 6.5 (1957).

[5] M. BONNEAU, N. PEYRARD et R. SABBADIN. « A Reinforcement-Learning Algorithm for Sampling Design in Markov Random Fields ». In : 2012.

[6] C. BROWNE et al. « A Survey of Monte Carlo Tree Search Methods ». In : *IEEE Transactions on Computational Intelligence and AI in Games* 4.1 (2012).

[7] R. COULOM. « Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search ». In : *Proceedings of the Fifth International Conference on Computer and Games (CG-2006)*. 2006.

[8] A. DOUCET, S. GODSILL et A. CHRISTOPHE. « On sequential Monte Carlo sampling methods for Bayesian filtering ». In : *Statistics and Computing* 10.3 (2000), p. 197–208.

[9] M. EGOROV, M. J. KOCHENDERFER et J. J. UUDMAE. « Target Surveillance in Adversarial Environments using POMDPs ». In : *AAAI-16*. 2016.

[10] M. FEHR, O. BUFFET, V. THOMAS et J. DIBANGOYE. « ρ-POMDPs have Lipschitz-Continuous ϵ-Optimal Value Functions ». In : *Advances in Neural Information Processing Systems 32 (NIPS-18)*. 2018.

[11] D. FOX, W. BURGARD et S. THRUN. « Active Markov Localization for Mobile Robots ». In : *Robotics and Autonomous Systems* 25.3–4 (1998). DOI : http://dx.doi.org/10.1016/S0921-8890(98)00049-9.

[12] M. KEARNS, Y. MANSOUR et A. NG. « A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes ». In : *Machine Learning* 49 (2002).

[13] L. KOCSIS et C. SZEPESVARI. « Bandit based Monte-Carlo Planning ». In : *ECML-06*. 2006.

[14] L. MIHAYLOVA, T. LEFEBVRE, H. BRUYNINCKX et J. D. SCHUTTER. In : *NATO Science Series on Data Fusion for Situation Monitoring, Incident Detection, Alert and Response Management*. T. 198. 2006. Chap. Active Robotic Sensing as Decision Making with Statistical Methods.

[15] M. L. PUTERMAN. *Markov Decision Processes – Discrete Stochastic Dynamic Programming*. 1994.

[16] Y. SATSANGI, S. WHITESON et M. T. J. SPAAN. *An Analysis of Piecewise-Linear and Convex Value Functions for Active Perception POMDPs*. Rapp. tech. IAS-UVA-15-01. IAS, Universiteit van Amsterdam, 2015.

[17] D. SILVER et J. VENESS. « Monte-Carlo Planning in Large POMDPs ». In : *NIPS-10*. 2010.

[18] T. SMITH et R. SIMMONS. « Heuristic Search Value Iteration for POMDPs ». In : *UAI-04*. 2004.

[19] M. T. SPAAN, T. S. VEIGA et P. U. LIMA. « Decision-theoretic planning under uncertainty with information rewards for active cooperative perception ». In : *JAAMAS* 29.6 (2015).

[20] Z. N. SUNBERG et M. J. KOCHENDERFER. « Online Algorithms for POMDPs with Continuous State, Action, and Observation Spaces ». In : *ICAPS-18*. 2018.

[21] Z. SUNBERG et M. J. KOCHENDERFER. « POMCPOW : An online algorithm for POMDPs with continuous state, action, and observation spaces ». In : *CoRR* abs/1709.06196 (2017). URL : http://arxiv.org/abs/1709.06196.

TABLE 3 – Influence de $|\beta|$ pour budget temps fixe de 1 seconde. Chaque colonne correspond aux résultats obtenus par ρ-POMCP pour différentes valeurs de $|\beta|$. Chaque valeur est le résultat de 200 expériences conduites sur des épisodes de 40 actions. V est la valeur cumulée γ-atténuée moyenne, Err l'erreur standard, nb_{xp} le nombre d'expériences accomplissant effectivement les 40 actions, et nb_d le nombre de descentes effectuées en 1 seconde.

	$\|\beta\|=0$			$\|\beta\|=1$			$\|\beta\|=5$			$\|\beta\|=10$			$\|\beta\|=20$			$\|\beta\|=50$			$\|\beta\|=100$			$\|\beta\|=500$		
Problem (UCB cst)	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d
Tiger (360)	-13.33 ± 0.07	200	17764	1.34 ± 0.18	200	16518	1.96 ± 0.12	200	13060	1.82 ± 0.12	200	9496	2.05 ± 0.12	200	6382	0.88 ± 0.13	200	3515	-1.41 ± 0.12	200	2159	-4.00 ± 0.00	200	621
CameraClean (14)	0.23 ± 0.02	200	20136	0.23 ± 0.02	200	17985	0.46 ± 0.02	200	13537	0.54 ± 0.02	200	9623	0.53 ± 0.02	200	6443	0.59 ± 0.02	200	3282	0.56 ± 0.02	200	2020	0.54 ± 0.02	200	545
SeekAndSeek (69.3)	-29.15 ± 1.29	200	11626	-27.51 ± 1.29	200	10971	-27.65 ± 1.27	200	8842	-28.88 ± 1.26	200	6919	-27.90 ± 1.25	200	4608	-30.32 ± 1.19	200	2383	-27.01 ± 1.25	200	1367	-29.63 ± 1.30	200	356
Museum entropy (1)	-16.67 ± 0.29	200	12100	-16.75 ± 0.27	200	10788	-16.58 ± 0.31	200	9007	-16.17 ± 0.28	200	7558	-16.83 ± 0.29	200	5714	-16.08 ± 0.28	200	3329	-16.75 ± 0.29	200	2030	-16.92 ± 0.29	199	599
Museum threshold (1)	6.16 ± 0.17	200	13831	6.00 ± 0.17	200	12580	5.95 ± 0.16	200	11065	6.12 ± 0.16	200	8675	6.13 ± 0.16	200	6300	6.26 ± 0.16	200	3513	6.45 ± 0.17	200	2141	6.29 ± 0.17	200	609
MazeCross (3.2)	2.21 ± 0.01	200	16140	2.20 ± 0.01	200	14946	2.21 ± 0.01	200	12594	2.23 ± 0.01	200	10028	2.19 ± 0.01	200	6806	2.19 ± 0.01	200	3492	2.20 ± 0.01	200	2058	2.21 ± 0.01	200	538
MazeLines (4.3)	2.67 ± 0.03	200	11413	2.68 ± 0.03	200	10161	2.73 ± 0.03	200	8427	2.70 ± 0.03	200	6826	2.69 ± 0.02	200	4981	2.73 ± 0.03	200	2582	2.65 ± 0.03	200	1421	2.69 ± 0.03	200	371
MazeHole (4.2)	2.03 ± 0.04	200	12215	2.05 ± 0.04	200	11596	2.02 ± 0.04	200	9485	2.10 ± 0.04	200	7693	1.92 ± 0.04	200	5384	2.04 ± 0.04	200	2722	1.99 ± 0.04	200	1537	2.03 ± 0.04	200	390
MazeDots (3.6)	2.07 ± 0.04	200	12167	2.09 ± 0.04	200	10553	2.07 ± 0.04	200	8648	2.08 ± 0.03	200	7065	2.04 ± 0.04	200	5018	2.11 ± 0.04	200	2633	2.10 ± 0.04	200	1518	2.07 ± 0.04	200	382
Activloc Cross (3.2)	2.10 ± 0.02	200	13072	2.10 ± 0.02	200	11930	2.13 ± 0.02	200	10613	2.13 ± 0.02	200	9124	2.13 ± 0.02	200	6848	2.09 ± 0.02	200	3835	2.10 ± 0.02	200	2297	2.09 ± 0.02	200	619
ActivLoc Lines (4.3)	2.15 ± 0.04	200	8462	2.09 ± 0.03	200	8086	2.21 ± 0.04	200	6810	2.15 ± 0.03	200	5921	2.11 ± 0.04	200	4469	2.14 ± 0.04	200	2375	2.16 ± 0.04	200	1333	2.11 ± 0.04	200	334
ActivLoc Hole (4.2)	1.66 ± 0.03	200	9261	1.71 ± 0.04	200	8503	1.75 ± 0.04	200	7503	1.65 ± 0.03	200	6379	1.73 ± 0.03	200	4593	1.73 ± 0.03	200	2521	1.69 ± 0.03	200	1403	1.71 ± 0.03	200	356
Activloc Dots (3.6)	1.37 ± 0.05	200	11384	1.45 ± 0.06	200	10485	1.37 ± 0.05	200	9077	1.43 ± 0.05	200	7613	1.42 ± 0.05	200	5468	1.38 ± 0.05	200	3059	1.39 ± 0.05	200	1652	1.47 ± 0.05	200	466
GridX (26)	21.64 ± 0.04	200	24773	21.68 ± 0.04	200	22485	21.65 ± 0.04	200	17109	21.67 ± 0.04	200	12633	21.67 ± 0.04	200	7653	21.66 ± 0.04	200	4220	21.66 ± 0.04	200	2431	21.69 ± 0.04	200	656
GridNotX (26)	-3.29 ± 0.15	200	21832	-3.31 ± 0.15	200	19934	-3.24 ± 0.15	200	15571	-3.34 ± 0.15	199	11177	-3.51 ± 0.15	200	7622	-3.29 ± 0.15	199	4127	-3.41 ± 0.15	200	2472	-3.56 ± 0.16	200	687
RockSampling44 (100)	115.92 ± 4.79	200	11414	107.73 ± 4.67	200	9898	104.46 ± 4.43	200	8591	106.05 ± 5.04	200	6249	108.23 ± 4.82	200	4424	95.38 ± 4.66	200	2483	89.22 ± 4.67	200	1444	59.21 ± 3.95	200	450
RockSampling44 (800)	77.87 ± 3.92	200	12533	66.09 ± 3.65	200	11583	62.69 ± 3.66	200	9689	51.50 ± 3.45	200	7493	57.01 ± 3.40	200	5111	57.20 ± 3.37	200	2800	53.97 ± 3.39	200	1695	61.91 ± 3.77	200	454

TABLE 4 – Influence de $|\beta|$ pour un budget temps fixe de 100 ms.

	$\|\beta\|=0$			$\|\beta\|=1$			$\|\beta\|=5$			$\|\beta\|=10$			$\|\beta\|=20$			$\|\beta\|=50$			$\|\beta\|=100$			$\|\beta\|=500$		
Problem (UCB cst)	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d
Tiger (360)	0.97 ± 0.18	200	955	1.73 ± 0.13	200	829	0.28 ± 0.12	200	671	-3.31 ± 0.06	200	425	-4.00 ± 0.00	200	272	-3.71 ± 0.07	200	158	-1.31 ± 0.13	200	91	1.37 ± 0.14	198	27
CameraClean (14)	0.23 ± 0.02	200	1509	0.22 ± 0.01	200	1224	0.46 ± 0.01	200	912	0.53 ± 0.02	200	661	0.52 ± 0.02	200	394	0.55 ± 0.02	200	226	0.53 ± 0.02	200	130	0.54 ± 0.02	199	42
SeekAndSeek (69.3)	-34.16 ± 1.44	195	1127	-34.65 ± 1.44	198	973	-31.15 ± 1.41	200	670	-33.57 ± 1.33	200	518	-34.16 ± 1.34	199	347	-31.51 ± 1.40	198	171	-33.02 ± 1.36	196	100	-35.29 ± 1.50	188	30
Museum entropy (1)	-16.95 ± 0.29	200	913	-17.12 ± 0.30	198	875	-17.17 ± 0.30	200	864	-16.83 ± 0.29	200	653	-17.02 ± 0.32	200	475	-16.78 ± 0.30	197	252	-17.75 ± 0.31	195	147	-18.36 ± 0.40	143	45
Museum threshold (1)	6.35 ± 0.16	200	756	5.99 ± 0.16	199	669	6.30 ± 0.17	200	638	6.20 ± 0.17	200	596	6.35 ± 0.16	199	366	5.96 ± 0.16	194	218	6.19 ± 0.18	189	127	5.41 ± 0.19	164	45
GridX (26)	21.64 ± 0.04	200	1480	21.67 ± 0.04	199	1295	21.61 ± 0.04	200	885	21.66 ± 0.04	200	677	21.75 ± 0.04	200	428	21.63 ± 0.04	199	226	21.62 ± 0.04	200	132	21.61 ± 0.05	189	40
GridNotX (26)	-3.70 ± 0.16	200	1535	-3.68 ± 0.15	200	1211	-3.36 ± 0.15	199	958	-3.62 ± 0.15	199	649	-3.60 ± 0.16	199	488	-3.76 ± 0.15	199	223	-3.52 ± 0.16	197	148	-3.42 ± 0.18	181	41
MazeCross (3.2)	2.20 ± 0.01	199	1091	2.19 ± 0.01	199	928	2.23 ± 0.01	200	709	2.19 ± 0.01	200	532	2.22 ± 0.01	200	408	2.21 ± 0.01	199	209	2.20 ± 0.01	200	121	2.21 ± 0.01	194	35
MazeLines (4.3)	2.61 ± 0.03	188	722	2.50 ± 0.03	181	635	2.57 ± 0.03	195	517	2.59 ± 0.03	196	383	2.58 ± 0.03	197	269	2.58 ± 0.03	198	142	2.61 ± 0.03	199	84	2.49 ± 0.03	184	24
MazeHole (4.2)	2.03 ± 0.05	189	879	2.03 ± 0.04	195	808	2.02 ± 0.04	197	592	1.99 ± 0.04	200	451	1.99 ± 0.04	199	316	1.96 ± 0.04	199	164	1.95 ± 0.04	198	100	1.96 ± 0.04	189	28
MazeDots (3.6)	2.01 ± 0.04	190	805	2.06 ± 0.04	194	747	2.07 ± 0.04	199	555	2.07 ± 0.04	200	381	1.96 ± 0.03	198	275	2.15 ± 0.04	199	151	2.03 ± 0.04	196	100	2.05 ± 0.04	189	29
ActivlocCross (3.2)	2.09 ± 0.02	200	1005	2.06 ± 0.02	200	873	2.11 ± 0.02	200	678	2.12 ± 0.02	200	508	2.09 ± 0.02	200	410	2.09 ± 0.02	200	245	2.08 ± 0.02	200	154	2.12 ± 0.02	199	47
ActivlocLines (4.3)	2.03 ± 0.04	200	755	1.95 ± 0.04	198	671	2.02 ± 0.04	200	580	2.07 ± 0.03	200	466	2.09 ± 0.04	200	314	2.07 ± 0.04	200	165	2.01 ± 0.04	200	95	2.05 ± 0.04	183	30
ActivlocHole (4.2)	1.50 ± 0.03	198	856	1.47 ± 0.03	195	810	1.56 ± 0.03	200	697	1.55 ± 0.03	200	535	1.59 ± 0.03	200	358	1.59 ± 0.03	199	177	1.60 ± 0.03	199	103	1.56 ± 0.03	181	34
ActivlocDots (3.6)	1.30 ± 0.05	200	1010	1.42 ± 0.05	199	858	1.45 ± 0.05	200	670	1.40 ± 0.05	200	492	1.39 ± 0.05	200	377	1.34 ± 0.05	200	220	1.41 ± 0.05	199	123	1.31 ± 0.05	188	42
RockSampling 44 (100)	104.56 ± 4.37	200	1055	109.47 ± 5.01	200	902	97.97 ± 4.29	200	730	90.61 ± 4.57	200	516	85.99 ± 4.70	200	315	68.58 ± 4.16	200	154	69.35 ± 4.04	199	82	31.71 ± 2.48	198	24
RockSampling 44 (800)	67.71 ± 4.23	200	1381	64.91 ± 4.28	200	1224	66.05 ± 3.90	200	787	72.71 ± 4.19	200	553	62.15 ± 4.13	200	354	53.58 ± 3.76	199	155	64.06 ± 3.93	198	88	39.45 ± 2.88	197	26

TABLE 5 – Résultats obtenus avec différents rollout sur ρ-POMCP avec un budget temps fixe de 500ms et plusieurs valeurs de $|\beta|$. La colonne *No Rollout* correspond à une valeur de rollout constante (0), les colonnes *Random* à des rollouts aléatoires (actions sélectionnées aléatoirement pendant l'étape de simulation) et les colonnes *Myopic* à des rollouts myopes (actions sélectionnées selon une stratégie *Look-ahead-1*. Chaque valeur correspond au résultat aggrégé de 200 expériences.

	No rollout $\|\beta\|=1$		No rollout $\|\beta\|=10$		No rollout $\|\beta\|=50$		Random $\|\beta\|=1$		Random $\|\beta\|=10$		Random $\|\beta\|=50$		Myopic $\|\beta\|=1$		Myopic $\|\beta\|=10$		myopic $\|\beta\|=50$	
Problem (UCB cst)	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d
Museum entropy (1)	-17.33 ± 0.41	5049	-16.88 ± 0.44	2834	-16.63 ± 0.46	1367	-17.27 ± 0.41	819	-17.86 ± 0.49	281	-20.99 ± 0.43	67	- ± -	-	- ± -	-	- ± -	-
SeekAndSeek (69.3)	-30.98 ± 1.87	5013	-25.53 ± 1.88	2911	-29.80 ± 1.73	1096	-36.71 ± 1.92	1076	-29.53 ± 1.97	420	-28.24 ± 1.85	104	-37.91 ± 2.18	182	-31.45 ± 2.05	119	-35.75 ± 1.83	54
CameraClean (14)	0.25 ± 0.03	9679	0.58 ± 0.03	4025	0.57 ± 0.02	1496	0.20 ± 0.02	1862	0.42 ± 0.02	810	0.55 ± 0.02	205	0.21 ± 0.02	424	0.45 ± 0.02	297	0.54 ± 0.02	109
MazeCross (3.2)	2.21 ± 0.01	5753	2.20 ± 0.01	3419	2.17 ± 0.01	1438	2.20 ± 0.01	1675	2.18 ± 0.01	811	2.23 ± 0.02	211	2.20 ± 0.01	425	2.18 ± 0.01	316	2.19 ± 0.01	126
MazeLines (4.3)	2.63 ± 0.04	5484	2.70 ± 0.04	3437	2.66 ± 0.04	1154	2.62 ± 0.04	1670	2.60 ± 0.04	731	2.60 ± 0.04	203	2.45 ± 0.05	427	2.64 ± 0.04	276	2.56 ± 0.04	118
MazeHole (4.2)	2.11 ± 0.06	4362	2.02 ± 0.05	2892	2.07 ± 0.06	1107	1.95 ± 0.06	1696	2.06 ± 0.06	789	2.00 ± 0.06	195	1.96 ± 0.07	411	1.91 ± 0.05	271	1.89 ± 0.06	111
MazeDots (3.6)	2.02 ± 0.05	4023	2.07 ± 0.05	2598	2.00 ± 0.05	1054	2.00 ± 0.05	1590	2.11 ± 0.05	784	2.11 ± 0.06	198	2.01 ± 0.05	419	2.06 ± 0.05	294	2.02 ± 0.06	118
ActivlocCross (3.2)	2.10 ± 0.03	4774	2.12 ± 0.03	3364	2.15 ± 0.03	1635	2.14 ± 0.03	2095	2.13 ± 0.03	1095	2.08 ± 0.03	357	2.10 ± 0.03	513	2.08 ± 0.03	430	2.09 ± 0.03	184
ActivlocLines (4.3)	2.13 ± 0.06	3356	2.10 ± 0.05	2309	2.14 ± 0.05	1057	1.96 ± 0.05	1723	2.07 ± 0.05	980	2.13 ± 0.05	296	1.99 ± 0.05	484	2.13 ± 0.05	385	2.04 ± 0.06	146
ActivlocHole (4.2)	1.63 ± 0.04	3856	1.67 ± 0.04	2656	1.66 ± 0.04	1120	1.58 ± 0.05	1864	1.64 ± 0.05	972	1.54 ± 0.04	308	1.42 ± 0.04	472	1.53 ± 0.04	384	1.53 ± 0.04	148
ActivlocDots (3.6)	1.59 ± 0.08	4514	1.41 ± 0.07	2977	1.40 ± 0.07	1332	1.35 ± 0.07	1973	1.32 ± 0.07	1062	1.31 ± 0.06	317	1.19 ± 0.08	474	1.44 ± 0.07	397	1.30 ± 0.07	161
GridNotX (26)	-3.50 ± 0.22	8651	-3.24 ± 0.22	4293	-3.29 ± 0.21	1806	-3.66 ± 0.18	1082	-3.61 ± 0.22	410	-4.01 ± 0.26	103	-3.60 ± 0.25	124	-3.78 ± 0.25	76	-3.50 ± 0.24	46
GridX (26)	21.67 ± 0.05	10270	21.75 ± 0.06	5089	21.77 ± 0.06	1750	21.64 ± 0.06	1083	21.66 ± 0.07	397	21.46 ± 0.08	104	21.45 ± 0.14	120	21.41 ± 0.07	77	21.43 ± 0.08	48
RockSampling44 (100)	113.53 ± 6.83	3797	115.85 ± 6.75	2253	96.08 ± 5.84	979	51.77 ± 5.04	924	31.92 ± 3.72	337	24.49 ± 3.33	81	16.67 ± 3.20	63	51.82 ± 5.38	43	97.45 ± 6.44	21
RockSampling44 (800)	65.76 ± 5.42	4673	50.28 ± 4.94	2842	55.05 ± 5.35	1205	20.78 ± 2.61	926	20.01 ± 2.74	334	31.24 ± 4.07	82	22.95 ± 3.79	59	47.96 ± 4.54	45	100.72 ± 6.72	22

TABLE 6 – Résultats obtenus avec différentes formules de mise à jour dans ρ-POMCP avec un temps budget fixe de 500ms et plusieurs valeurs de $|\beta|$. Les colonnes *Vanilla* correspondent à la mise à jour de POMCP, les colonnes *Last-reward* aux mises à jour où la dernière récompense reçue remplace son estimation par moyenne mobile, et les colonnes *Last-value* aux mises à jour où les valeurs des nœuds visités sont recalculées. Chaque valeur correspond au résultat aggrégé de 200 expériences.

	Vanilla $\|\beta\|=1$		Vanilla $\|\beta\|=10$		Vanilla $\|\beta\|=50$		Last-Reward $\|\beta\|=1$		Last-Reward $\|\beta\|=10$		Last-Reward $\|\beta\|=50$		Last-Value $\|\beta\|=1$		Last-Value $\|\beta\|=10$		Last-Value $\|\beta\|=50$	
Problem (UCB cst)	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d
Museum entropy (1)	-17.33 ± 0.41	5049	-16.88 ± 0.44	2834	-16.63 ± 0.46	1367	-16.16 ± 0.46	4280	-16.15 ± 0.37	2739	-16.87 ± 0.42	1356	-16.88 ± 0.40	5080	-16.26 ± 0.42	2771	-17.36 ± 0.44	1406
SeekAndSeek (69.3)	-30.98 ± 1.87	5013	-25.53 ± 1.88	2911	-29.80 ± 1.73	1096	-28.00 ± 1.91	5176	-33.26 ± 1.84	3024	-28.39 ± 1.80	1057	-30.59 ± 1.84	5263	-28.73 ± 1.77	2870	-26.45 ± 1.89	1074
CameraClean (14)	0.25 ± 0.03	9679	0.58 ± 0.03	4025	0.57 ± 0.02	1496	0.28 ± 0.02	8000	0.59 ± 0.02	3925	0.56 ± 0.02	1458	0.53 ± 0.02	8969	0.55 ± 0.02	4214	0.55 ± 0.02	1444
MazeCross (3.2)	2.21 ± 0.01	5753	2.20 ± 0.01	3419	2.17 ± 0.01	1438	2.16 ± 0.01	5528	2.21 ± 0.01	3472	2.21 ± 0.02	1488	2.20 ± 0.01	7922	2.22 ± 0.02	3805	2.22 ± 0.01	1608
MazeLines (4.3)	2.63 ± 0.04	5484	2.70 ± 0.04	3437	2.66 ± 0.04	1154	2.64 ± 0.04	4727	2.65 ± 0.04	3082	2.62 ± 0.04	1090	2.63 ± 0.04	5516	2.69 ± 0.04	2978	2.72 ± 0.04	1057
MazeHole (4.2)	2.11 ± 0.06	4362	2.02 ± 0.05	2892	2.07 ± 0.06	1107	2.01 ± 0.05	4155	2.09 ± 0.06	2803	2.09 ± 0.06	1112	2.12 ± 0.06	5104	2.00 ± 0.05	3370	1.94 ± 0.05	1138
MazeDots (3.6)	2.02 ± 0.05	4023	2.07 ± 0.05	2598	2.00 ± 0.05	1054	2.14 ± 0.05	4112	2.18 ± 0.05	2540	2.08 ± 0.05	1067	2.17 ± 0.06	5343	2.17 ± 0.06	2933	2.17 ± 0.06	1100
ActivlocCross (3.2)	2.10 ± 0.03	4774	2.12 ± 0.03	3364	2.15 ± 0.03	1635	2.09 ± 0.03	5056	2.06 ± 0.03	3449	2.10 ± 0.03	1586	2.09 ± 0.03	5715	2.04 ± 0.03	3964	2.09 ± 0.03	1570
ActivlocLines (4.3)	2.13 ± 0.06	3356	2.10 ± 0.05	2309	2.14 ± 0.05	1057	1.81 ± 0.05	3078	2.08 ± 0.05	2381	2.15 ± 0.06	1016	2.15 ± 0.05	3908	2.24 ± 0.05	2430	2.18 ± 0.05	1017
ActivlocHole (4.2)	1.63 ± 0.04	3856	1.67 ± 0.04	2656	1.66 ± 0.04	1120	1.58 ± 0.05	3582	1.58 ± 0.04	2681	1.77 ± 0.05	1103	1.71 ± 0.05	3827	1.62 ± 0.04	2524	1.66 ± 0.04	1086
ActivlocDots (3.6)	1.59 ± 0.08	4514	1.41 ± 0.07	2977	1.40 ± 0.07	1332	1.41 ± 0.07	4372	1.57 ± 0.08	2906	1.46 ± 0.08	1331	1.37 ± 0.08	5070	1.45 ± 0.07	3227	1.48 ± 0.07	1292
GridNotX (26)	-3.50 ± 0.22	8651	-3.24 ± 0.22	4293	-3.29 ± 0.21	1806	-3.68 ± 0.22	7961	-3.56 ± 0.17	4504	-3.27 ± 0.22	1811	-3.34 ± 0.23	8478	-3.10 ± 0.21	4173	-3.80 ± 0.21	1665
GridX (26)	21.67 ± 0.05	10270	21.75 ± 0.06	5089	21.77 ± 0.06	1750	21.71 ± 0.06	9303	21.69 ± 0.05	4722	21.64 ± 0.06	1831	21.63 ± 0.05	10273	21.66 ± 0.06	4150	21.70 ± 0.06	1737
RockSampling44 (100)	113.53 ± 6.83	3797	115.85 ± 6.75	2253	96.08 ± 5.84	979	114.13 ± 6.96	3811	114.61 ± 6.78	2108	91.66 ± 5.98	981	110.04 ± 6.90	4800	112.79 ± 6.37	2460	84.70 ± 6.23	907
RockSampling44 (800)	65.76 ± 5.42	4673	50.28 ± 4.94	2842	55.05 ± 5.35	1205	65.62 ± 5.85	4120	54.49 ± 5.54	2654	63.67 ± 5.69	1179	76.05 ± 5.49	5000	54.84 ± 4.84	2870	48.10 ± 4.93	1022

The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Action Schema Networks: Generalised Policies with Deep Learning

Sam Toyer,[1] Felipe Trevizan,[1,2] Sylvie Thiébaux,[1] Lexing Xie[1,3]
[1] Research School of Computer Science, Australian National University
[2] Data61, CSIRO [3] Data to Decisions CRC
first.last@anu.edu.au

Abstract

In this paper, we introduce the Action Schema Network (ASNet): a neural network architecture for learning generalised policies for probabilistic planning problems. By mimicking the relational structure of planning problems, ASNets are able to adopt a weight sharing scheme which allows the network to be applied to any problem from a given planning domain. This allows the cost of training the network to be amortised over all problems in that domain. Further, we propose a training method which balances exploration and supervised training on small problems to produce a policy which remains robust when evaluated on larger problems. In experiments, we show that ASNet's learning capability allows it to significantly outperform traditional non-learning planners in several challenging domains.

1 Introduction

Automated planning is the task of finding a sequence of actions which will achieve a goal within a user-supplied model of an environment. Over the past four decades, there has been a wealth of research into the use of machine learning for automated planning (Jiménez et al. 2012), motivated in part by the belief that these two essential ingredients of intelligence—planning and learning—ought to strengthen one other (Zimmerman and Kambhampati 2003). Nevertheless, the dominant paradigm among state-of-the-art classical and probabilistic planners is still based on heuristic state space search. The domain-independent heuristics used for this purpose are capable of exploiting common structures in planning problems, but do not learn from experience. Top planners in both the deterministic and learning tracks of the International Planning Competition often use machine learning to configure portfolios (Vallati et al. 2015), but only a small fraction of planners make meaningful use of learning to produce domain-specific heuristics or control knowledge (de la Rosa, Celorrio, and Borrajo 2008). Planners which transfer knowledge between problems in a domain have been similarly underrepresented in the probabilistic track of the competition.

In parallel with developments in planning, we've seen a resurgence of interest in neural nets, driven largely by their success at problems like image recognition (Krizhevsky, Sutskever, and Hinton 2012) and learning to play video games (Mnih et al. 2013). This paper brings some gains of deep learning to planning by proposing a new neural network architecture, the ASNet, which is specialised to the structure of planning problems much as Convolutional Neural Networks (CNNs) are specialised to the structure of images. The basic idea is illustrated in Figure 1: rather than operating on a virtual graph of pixels with edges defined by adjacency relationships, an ASNet operates on a graph of actions and propositions (i.e. Boolean variables), with edges defined by relations of the form "action a affects proposition p" or "proposition p influences the outcome of action a". This structure allows an ASNet to be trained on one problem from a given planning domain and applied to other, different problems without re-training.

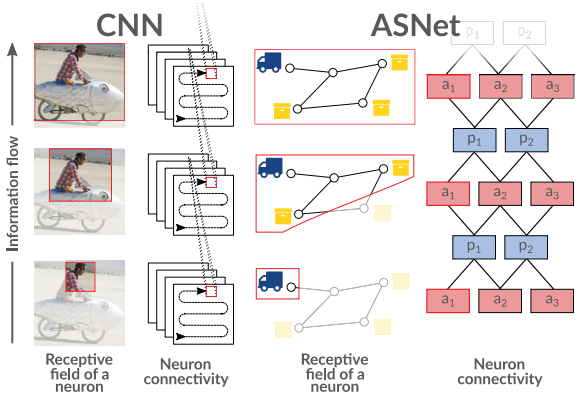


Figure 1: In a CNN, successive convolutions grow the receptive field of a neuron at higher layers; analogously for ASNet, a neuron for a particular action a or proposition p *sees* a larger portion of the current state at higher layers.

We make three new contributions. (1) A neural network architecture for probabilistic planning that automatically generalises to any problem from a given planning domain. (2) A representation that allows weight sharing among actions modules belonging to the same action schema, and among proposition modules associated with the same predicate. This representation is augmented by input features from domain-independent planning heuristics. (3) A train-

ing method that balances exploration and supervision from existing planners. In experiments, we show that this strategy is sufficient to learn effective generalised policies. Code and models for this work are available online. [1]

2 Background

This work considers probabilistic planning problems represented as Stochastic Shortest Path problems (SSPs) (Bertsekas and Tsitsiklis 1996). Formally, an SSP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, \mathcal{G}, s_0)$ where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{T}\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function, $\mathcal{C}\colon \mathcal{S} \times \mathcal{A} \to (0, \infty)$ is a cost function, $\mathcal{G} \subseteq \mathcal{S}$ is a set of goal states, and s_0 is an initial state. At each state s, an agent chooses an action a from a set of enabled actions $\mathcal{A}(s) \subseteq \mathcal{A}$, incurring a cost of $\mathcal{C}(s, a)$ and causing it to transition into another state $s' \in \mathcal{S}$ with probability $\mathcal{T}(s, a, s')$.

The solution of an SSP is a policy $\pi\colon \mathcal{A} \times \mathcal{S} \to [0, 1]$ such that $\pi(a \mid s)$ is the probability that action a will be applied in state s. An optimal policy π^* is any policy that minimises the total expected cost of reaching $\mathcal{G}$ from s_0. We do not assume that the goal is reachable with probability 1 from s_0 (i.e. we allow problems with unavoidable dead ends), and a fixed-cost penalty is incurred every time a dead end is reached (Mausam and Kolobov 2012).

A *factored SSP* is a compact representation of an SSP as a tuple $(\mathcal{P}, \mathcal{A}, s_0, s_\star, \mathcal{C})$. $\mathcal{P}$ is a finite set of binary *propositions* and the state space $\mathcal{S}$ is the set of all binary strings of size $|\mathcal{P}|$. Thus, a state s is a value assignment to all the propositions $p \in \mathcal{P}$. A *partial state* is a value assignment to a subset of propositions; a partial state s is consistent with a partial state s' if the value assignments of s' are contained in s ($s' \subseteq s$ for short). The goal is represented by a partial state $s_\star$, and $\mathcal{G} = \{s \in \mathcal{S} | s_\star \subseteq s\}$. Each action $a \in \mathcal{A}$ consists in a precondition pre_a represented by a partial state, a set of effects eff_a each represented by a partial state, and a probability distribution Pr_a over effects in eff_a.[2] The actions applicable in state s are $\mathcal{A}(s) = \{a \in \mathcal{A} \mid pre_a \subseteq s\}$. Moreover, $\mathcal{T}(s, a, s') = \sum_{e \in \mathit{eff}_a | s' = res(s,e)} Pr_a(e)$ where $res(s, e) \in \mathcal{S}$ is the result of changing the value of propositions of s to make it consistent with effect e.

A *lifted* SSP compactly represents a set of factored SSPs sharing the same structure. Formally, a lifted SSP is a tuple $(\mathcal{F}, \mathbb{A}, \mathcal{C})$ where $\mathcal{F}$ is a finite set of predicates, and $\mathbb{A}$ is a finite set of action schemas. Each predicate, when grounded, i.e., instantiated by a tuple of names representing objects, yields a factored SSP proposition. Similarly, each action schema, instantiated by a tuple of names, yields a factored SSP action. The Probabilistic Planning Domain Definition Language (PPDDL) is the standard language to describe lifted and factored SSPs (Younes and Littman 2004). PPDDL splits the description into a general *domain* and a specific *problem*. The domain gives the predicates $\mathcal{F}$, action schemas $\mathbb{A}$ and cost function $\mathcal{C}$ specifying a lifted SSP. The problem additionally gives the set of objects $\mathcal{O}$, initial state s_0 and goal $s_\star$, describing a specific SSP whose propositions and actions are obtained by grounding the domain predicates and action schemas using the objects in $\mathcal{O}$. For instance the domain description might specify a predicate $\mathrm{at}(?l)$ and an action schema $\mathrm{walk}(?\mathrm{from}, ?\mathrm{to})$, while the problem description might specify objects home and work. Grounding using these objects would produce propositions $\mathrm{at}(\mathrm{home})$ and $\mathrm{at}(\mathrm{work})$, as well as ground actions $\mathrm{walk}(\mathrm{work}, \mathrm{home})$ and $\mathrm{walk}(\mathrm{home}, \mathrm{work})$.

Observe that different factored SSPs can be obtained by changing only the problem part of the PPDDL description while reusing its domain. In the next section, we show how to take advantage of action schema reuse to learn policies that can then be applied to any factored SSP obtained by instantiating the same domain.

3 Action Schema Networks

Neural networks are expensive to train, so we would like to amortise that cost over many problems by learning a *generalised policy* which can be applied to any problem from a given domain. ASNet proposes a novel, domain-specialised structure that uses the same set of learnt weights θ regardless of the "shape" of the problem. The use of such a weight sharing scheme is key to ASNet's ability to generalise to different problems drawn from the same domain, even when those problems have different goals or different numbers of actions and propositions.

3.1 Network structure

At a high level, an ASNet is composed of alternating action layers and proposition layers, where action layers are composed of a single *action module* for each ground action, and proposition layers likewise are composed of a single *proposition module* for each ground proposition; this choice of structure was inspired by the alternating action and proposition layers of Graphplan (Blum and Furst 1997). In the same way that hidden units in one layer of a CNN connect only to nearby hidden units in the next layer, action modules in one layer of an ASNet connect only to directly related proposition modules in the next layer, and vice versa. The last layer of an ASNet is always an action layer with each module defining an action selection probability, thus allowing the ASNet to scale to problems with different numbers of actions. For simplicity, we also assume that the first (input) layer is always an action layer.

Action module details. Consider an action module for $a \in \mathcal{A}$ in the lth action layer. The module takes as input a feature vector u_a^l, and produces a new hidden representation

$$\phi_a^l = f(W_a^l \cdot u_a^l + b_a^l) \,,$$

where $W_a^l \in \mathbb{R}^{d_h \times d_a^l}$ is a learnt weight matrix for the module, $b_a^l \in \mathbb{R}^{d_h}$ is a learnt bias vector, $f(\cdot)$ is a nonlinearity (e.g. $\tanh$, sigmoid, or ReLU), d_h is a (fixed) intermediate representation size, and d_a^l is the size of the inputs to the action module. The feature vector u_a^l, which serves as input to the action module, is constructed by enumerating the propositions $p_1, p_2, \ldots, p_M$ which are *related* to the action

[1] https://github.com/qxcv/asnets

[2] Factored SSPs sometimes support conditional effects and negative or disjunctive preconditions and goals. We do not use these here to simplify notation. However, ASNet can easily be extended to support these constructs.

a, and then concatenating their hidden representations. Formally, we say that a proposition $p \in \mathcal{P}$ is related to an action $a \in \mathcal{A}$, denoted $R(a, p)$, if p appears in pre_a or in an effect e where $Pr_a(e) > 0$. Concatenation of representations for the related propositions produces a vector

$$u_a^l = \begin{bmatrix} {\psi_1^{l-1}}^T & \cdots & {\psi_M^{l-1}}^T \end{bmatrix}^T ,$$

where ψ_j^{l-1} is the hidden representation produced by the proposition module for proposition $p_j \in \mathcal{P}$ in the preceding proposition layer. Each of these constituent hidden representations has dimension d_h, so u_a^l has dimension $d_a^l = d_h \cdot M$.

Our notion of propositional relatedness ensures that, if ground actions a_1 and a_2 in a problem are instances of the same action schema in a PPDDL domain, then their inputs u_1^l and u_2^l will have the same "structure". To see why, note that we can determine which propositions are related to a given ground action a by retrieving the corresponding action schema, enumerating the predicates which appear in the precondition or the effects of the action schema, then instantiating those predicates with the same parameters used to instantiate a. If we apply this procedure to a_1 and a_2, we will obtain lists of related propositions $p_1, p_2, \ldots, p_M$ and $q_1, q_2, \ldots, q_M$, respectively, where p_j and q_j are propositions with the same predicate which appear in the same position in the definitions of a_1 and a_2 (i.e. the same location in the precondition, or the same position in an effect).

Such structural similarity is key to ASNet's generalisation abilities. At each layer l, and for each pair of ground actions c and d instantiated from the same action schema s, we use the same weight matrix W_s^l and bias vector b_s^l—that is, we have $W_c^l = W_d^l = W_s^l$ and $b_c^l = b_d^l = b_s^l$. Hence, modules for actions which appear in the same layer and correspond to the same action schema will use the same weights, but modules which appear in different layers or which correspond to different schemas will learn different weights. Although different problems instantiated from the same PPDDL domain may have different numbers of ground actions, those ground actions will still be derived from the same, fixed set of schemas in the domain, so we can apply the same set of action module weights to any problem from the domain.

The first and last layers of an ASNet consist of action modules, but their construction is subtly different:

1. The output of a module for action a in the final layer is a single number $\pi^\theta(a \mid s)$ representing the probability of selecting action a in the current state s under the learnt policy π^θ, rather than a vector-valued hidden representation. To guarantee that disabled actions are never selected, and ensure that action probabilities are normalised to 1, we pass these outputs through a *masked softmax* activation which ensures that $\pi^\theta(a \mid s) = 0$ if $a \notin \mathcal{A}(s)$. During training, we sample actions from $\pi^\theta(a \mid s)$. During evaluation, we select the action with the highest probability.
2. Action modules in the first layer of a ASNet are passed an input vector composed of features derived from the current state, rather than hidden representations for related propositions. Specifically, modules in the first layer are given a binary vector indicating the truth values of related propositions, and whether those propositions appear in the goal. In practice, it is helpful to concatenate these propositional features with heuristic features, as described in Section 3.2.

Proposition module details. Proposition modules only appear in the intermediate layers of an ASNet, but are otherwise similar to action modules. Specifically, a proposition module for proposition $p \in \mathcal{P}$ in the lth proposition layer of the network will compute a hidden representation

$$\psi_p^l = f(W_p^l \cdot v_p^l + b_p^l) ,$$

where v_p^l is a feature vector, f is the same nonlinearity used before, and $W_p^l \in \mathbb{R}^{d_h \times d_p^l}$ and $b_p^l \in \mathbb{R}^{d_h}$ are learnt weights and biases for the module.

To construct the input v_p^l, we first find the predicate $\mathrm{pred}(p) \in \mathcal{F}$ for proposition $p \in \mathcal{P}$, then enumerate all action schemas $A_1, \ldots, A_L \in \mathbb{A}$ which reference $\mathrm{pred}(p)$ in a precondition or effect. We can define a feature vector

$$v_p^l = \begin{bmatrix} \mathrm{pool}(\{{\phi_a^l}^T \mid \mathrm{op}(a) = A_1 \wedge R(a,p)\}) \\ \vdots \\ \mathrm{pool}(\{{\phi_a^l}^T \mid \mathrm{op}(a) = A_L \wedge R(a,p)\}) \end{bmatrix} ,$$

where $\mathrm{op}(a) \in \mathbb{A}$ denotes the action schema for ground action a, and pool is a pooling function that combines several d_h-dimensional feature vectors into a single d_h-dimensional one. Hence, when all pooled vectors are concatenated, the dimensionality d_p^l of v_p^l becomes $d_h \cdot L$. In this paper, we assume that pool performs max pooling (i.e. keeps only the largest input). If a proposition module had to pool over the outputs of many action modules, such pooling could potentially obscure useful information. While the issue could be overcome with a more sophisticated pooling mechanism (like neural attention), we did not find that max pooling posed a major problem in the experiments in Section 5, even on large Probabilistic Blocks World instances where some proposition modules must pool over thousands of inputs.

Pooling operations are essential to ensure that proposition modules corresponding to the same predicate have the same structure. Unlike action modules corresponding to the same action schema, proposition modules corresponding to the same predicate may have a different number of inputs depending on the initial state and number of objects in a problem, so it does not suffice to concatenate inputs. As an example, consider a single-vehicle logistics problem where the location of the vehicle is tracked with propositions of the form $\mathrm{at}(\iota)$, and the vehicle may be moved with actions of the form $\mathrm{move}(\iota_{\text{from}}, \iota_{\text{to}})$. A location ι_1 with one incoming road and no outgoing roads will have only one related move action, but a location ι_2 with two incoming roads and no outgoing roads will have two related move actions, one for each road. This problem is not unique to planning: a similar trick is employed in network architectures for graphs where vertices can have varying in-degree (Jain et al. 2016; Kearnes et al. 2016).

As with the action modules, we share weights between proposition modules for propositions corresponding to the

same predicate. Specifically, at proposition layer l, and for propositions q and r with $\mathrm{pred}(q) = \mathrm{pred}(r)$, we tie the corresponding weights $W_q^l = W_r^l$ and $b_q^l = b_r^l$. Together with the weight sharing scheme for action modules, this enables us to learn a single set of weights

$$\theta = \{W_a^l, b_a^l \mid 1 \leq l \leq n+1, a \in \mathbb{A}\} \cup \{W_p^l, b_p^l \mid 1 \leq l \leq n, p \in \mathcal{F}\}$$

for an n-layer model which can be applied to any problem in a given PPDDL domain.

3.2 Heuristic features for expressiveness

One limitation of the ASNet is the fixed receptive field of the network; in other words, the longest chain of related actions and propositions which it can reason about. For instance, suppose we have I locations $\iota_1, \ldots, \iota_I$ arranged in a line in our previous logistics example. The agent can move from ι_{k-1} to ι_k (for $k = 2, \ldots, I$) with the $\mathrm{move}(\iota_{k-1}, \iota_k)$ action, which makes $\mathrm{at}(\iota_{k-1})$ false and $\mathrm{at}(\iota_k)$ true. The propositions $\mathrm{at}(\iota_1)$ and $\mathrm{at}(\iota_I)$ will thus be related only by a chain of move actions of length $I-1$; hence, a proposition module in the lth proposition layer will only be affected by at propositions for locations at most $l+1$ moves away. Deeper networks can reason about longer chains of actions, but that an ASNet's (fixed) depth necessarily limits its reasoning power when chains of actions can be *arbitrarily* long.

We compensate for this receptive field limitation by supplying the network with features obtained using domain-independent planning heuristics. In this paper, we derive these features from disjunctive action landmarks produced by LM-cut (Helmert and Domshlak 2009), but features derived from different heuristics could be employed in the same way. A disjunctive action landmark is a set of actions in which at least one action must be applied along any optimal path to the goal in a deterministic, delete-relaxed version of the planning problem. These landmarks do not necessarily capture all useful actions, but in practice we find that providing information about these landmarks is often sufficient to compensate for network depth limitations.

In this paper, a module for action a in the first network layer is given a feature vector

$$u_a^1 = \begin{bmatrix} c^T & v^T & g^T \end{bmatrix}^T.$$

$c \in \{0,1\}^3$ indicates whether a_i is the sole action in at least one LM-cut landmark ($c_1 = 1$), an action in a landmark of two or more actions ($c_2 = 1$), or does not appear in a landmark ($c_3 = 1$). $v \in \{0,1\}^M$ represents the M related propositions: v_j is 1 iff p_j is currently true. $g \in \{0,1\}^M$ encodes related portions of the goal state, and g_j is 1 iff p_j is true in the partial state $s_\star$ defining the goal.

4 Training with exploration and supervision

We learn the ASNet weights θ by choosing a set of small training problems P_{train}, then alternating between guided exploration to build up a state memory $\mathcal{M}$, and supervised learning to ensure that the network chooses good actions for the states in $\mathcal{M}$. Algorithm 1 describes a single epoch of exploration and supervised learning. We repeatedly apply this procedure until performance on P_{train} ceases to improve, or until a fixed time limit is reached. Note that this strategy is only intended to learn the *weights* of an ASNet—module connectivity is not learnt, but rather obtained from a grounded representation using the notion of relatedness which we described earlier.

Algorithm 1 Updating ASNet weights θ using state memory $\mathcal{M}$ and training problem set P_{train}

procedure ASNET-TRAIN-EPOCH(θ, $\mathcal{M}$)
  for $i = 1, \ldots, T_{\text{explore}}$ **do** ▷ Exploration
    for all $\zeta \in P_{\text{train}}$ **do**
      $s_0, \ldots, s_N \leftarrow$ RUN-POL$(s_0(\zeta), \pi^\theta)$
      $\mathcal{M} \leftarrow \mathcal{M} \cup \{s_0, \ldots, s_N\}$
      for $j = 0, \ldots, N$ **do**
        $s_j^*, \ldots, s_M^* \leftarrow$ POL-ENVELOPE(s_j, π^*)
        $\mathcal{M} \leftarrow \mathcal{M} \cup \{s_j^*, \ldots, s_M^*\}$
  for $i = 1, \ldots, T_{\text{train}}$ **do** ▷ Learning
    $\mathcal{B} \leftarrow$ SAMPLE-MINIBATCH$(\mathcal{M})$
    Update θ using $\frac{d\mathcal{L}_\theta(\mathcal{B})}{d\theta}$ (Equation 1)

In the exploration phase of each training epoch, we repeatedly run the ASNet policy π^θ from the initial state of each problem $\zeta \in P_{\text{train}}$, collecting $N+1$ states $s_0, \ldots, s_N$ visited along each of the sampled trajectories. Each such trajectory terminates when it reaches a goal, exceeds a fixed limit L on length, or reaches a dead end. In addition, for each visited state s_j, we compute an optimal policy π^* rooted at s_j, then produce a set of states $s_j^*, \ldots, s_M^*$ which constitute π^*'s policy envelope—that is, the states which π^* visits with nonzero probability. Both the trajectories drawn from the ASNet policy π^θ and policy envelopes for the optimal policy π^* are added to the state memory $\mathcal{M}$. Saving states which can be visited under an optimal policy ensures that $\mathcal{M}$ always contains states along promising trajectories reachable from s_0. On the other hand, saving trajectories from the exploration policy ensures that ASNet will be able to improve on the states which it visits most often, even if they are not on an optimal goal trajectory.

In the training phase, small subsets of the states in $\mathcal{M}$ are repeatedly sampled at random to produce minibatches for training ASNet. The objective to be minimised for each minibatch $\mathcal{B}$ is the cross-entropy classification loss

$$\mathcal{L}_\theta(\mathcal{B}) = \sum_{s \in \mathcal{B}} \sum_{a \in A} \left[(1 - y_{s,a}) \cdot \log(1 - \pi^\theta(a \mid s)) + y_{s,a} \cdot \log \pi^\theta(a \mid s) \right]. \quad (1)$$

The label $y_{s,a}$ is 1 if the expected cost of choosing action a and then following an optimal policy thereafter is minimal among all enabled actions; otherwise, $y_{s,a} = 0$. This encourages the network to imitate an optimal policy. For each sampled batch $\mathcal{B}$, we compute the gradient $\frac{d\mathcal{L}_\theta(\mathcal{B})}{d\theta}$ and use it to update the weights θ in a direction which decreases $\mathcal{L}_\theta(\mathcal{B})$ with Adam (Kingma and Ba 2015).

The cost of computing an optimal policy during supervised learning is often non-trival. It is natural to ask whether

it is more efficient to train ASNets using unguided policy gradient reinforcement learning, as FPG does (Buffet and Aberdeen 2009). Unfortunately, we found that policy gradient RL was too noisy and inefficient to train deep networks on nontrivial problems; in practice, the cost of computing an optimal policy for small training problems more than pays for itself by enabling us to use sample-efficient supervised learning instead of reinforcement learning. In the experiments, we investigate the question of whether suboptimal policies are still sufficient for supervised training of ASNets.

Past work on generalised policy learning has employed learnt policies as control knowledge for search algorithms, in part because doing so can compensate for flaws in the policy. For example, Yoon, Fern, and Givan (2007) suggest employing policy rollout or limited discrepancy search to avoid the occasional bad action recommended by a policy. While we could use an ASNet similarly, we are more interested in its ability to learn a reliable policy on its own. Hence, during evaluation, we always choose the action which maximises $\pi^\theta(a \mid s)$. As noted above, this is different from the exploration process employed during training, where we instead sample from $\pi^\theta(a \mid s)$.

5 Experiments and discussion

In this section, we compare ASNet against state-of-the-art planners on three planning domains.

5.1 Experimental setup

We compare ASNet against three heuristic-search-based probabilistic planners: LRTDP (Bonet and Geffner 2003), ILAO* (Hansen and Zilberstein 2001) and SSiPP (Trevizan and Veloso 2014). Two domain-independent heuristics are considered for each of the three planners—LM-cut (admissible) and the additive heuristic h^{add} (inadmissible) (Teichteil-Königsbuch, Vidal, and Infantes 2011)—resulting in 6 baselines. During evaluation, we enforce a 9000s time cutoff for all the baselines and ASNets, as well as a 10Gb memory cutoff.

Since LRTDP and ILAO* are optimal planners, we execute them until convergence ($\epsilon = 10^{-4}$) for each problem using 30 different random seeds. Notice that, for h^{add}, LRTDP and ILAO* might converge to a suboptimal solution. If an execution of LRTDP or ILAO* does not converge before the given time/memory cutoff, we consider the planner as having failed to reach the goal. SSiPP is used as a replanner and, for each problem, it is *trained* until 60s before the time cutoff and then evaluated; this procedure is repeated 30 times for each problem using different random seeds. The training phase of SSiPP consists in simulating a trajectory from s_0 and, during this process, SSiPP improves its lower bound on the optimal solution. If 100 consecutive trajectories reach the goal during training, then SSiPP is evaluated regardless of the training time left. For the 6 baselines, we report the average running time per problem.

For each domain, we train a single ASNet, then evaluate it on each problem 30 times with different random seeds. The hyperparmeters for each ASNet were kept fixed across domains: three action layers and two proposition layers in each network, a hidden representation size of 16 for each internal action and proposition module, and an ELU (Clevert, Unterthiner, and Hochreiter 2016) as the nonlinearity f. The optimiser was configured with a learning rate of 0.0005 and a batch size of 128, and a hard limit of two hours (7200s) was placed on training. We also applied ℓ_2 regularisation with a coefficient of 0.001 on all weights, and dropout on the outputs of each layer except the last with $p = 0.25$. Each epoch of training alternated between 25 rounds of exploration shared equally among all training problems, and 300 batches of network optimisation (i.e. $T_{\text{explore}} = 25/|P_{\text{train}}|$ and $T_{\text{train}} = 300$). Sampled trajectory lengths are $L = 300$ for both training and evaluation. LRTDP with the LM-cut heuristic is used for computing the optimal policies during training, with a dead-end penalty of 500. We also repeated this procedure for LRTDP using h^{add} (inadmissible heuristic) to compare the effects of using optimal and suboptimal policies for training. Further, we report how well ASNet performs when it is guided by h^{add}, but *not* given the LM-cut-derived heuristic features described in Section 3.2. For the ASNets, we report the average *training time plus time to solve the problem* to highlight when it pays off to spend the one-off cost of training an ASNet for a domain.

All ASNets were trained and evaluated on a virtual machine equipped with 62GB of memory and an x86-64 processor clocked at 2.3GHz. For training and evaluation, each ASNet was restricted to use a single, dedicated processor core, but resources were otherwise shared. The baseline planners were run in a cluster of x86-64 processors clocked at 2.6GHz and each planner again used only a single core.

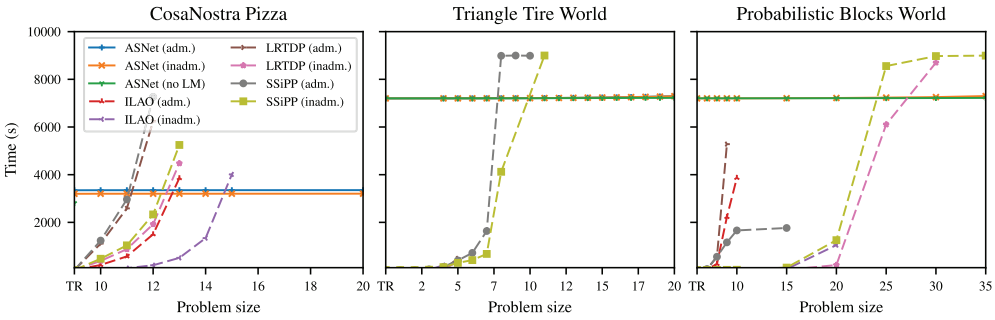
5.2 Domains

We evaluate ASNets and the baselines on the following probabilistic planning domains:

CosaNostra Pizza: as a Deliverator for CosaNostra Pizza, your job is to safely transport pizza from a shop to a waiting customer, then return to the shop. There is a series of toll booths between you and the customer: at each booth, you can either spend a time step paying the operator, or save a step by driving through without paying. However, if you don't pay, the (angry) operator will try to drop a boom on your car when you pass through their booth on the way back to the shop, crushing the car with 50% probability. The optimal policy is to pay operators when travelling to the customer to ensure a safe return, but not pay on the return trip as you will not revisit the booth. Problem size is the number of toll booths between the shop and the customer. ASNets are trained on sizes 1-5, and tested on sizes 6+.

Probabilistic Blocks World is an extension of the well-known deterministic blocks world domain in which a robotic arm has to move blocks on a table into a goal configuration. The actions to pick up a block or to put a block on top of another fail with probability 0.25; failure causes the target block to drop onto the table, meaning that it must be picked up and placed again. We randomly generate three different problems for each number of blocks considered during testing. ASNet is trained on five randomly generated problems of each size from 5–9, for 25 training problems total.

Triangle Tire World (Little and Thiébaux 2007): each



Problem			ASNet	Others
CN	14	Cover.	30/30	30/30
		Avg. C.	46.0	46.0
	15	Cover.	30/30	30/30
		Avg. C.	49.0	49.0
TTW	7	Cover.	30/30	30/30
		Avg. C.	40.8	41.4
	10	Cover.	30/30	1/30
		Avg. C.	59.0	60.0
PBW	25	Cover.	30/30	30/30
		Avg. C.	92.9	145.6
	30	Cover.	30/30	27/30
		Avg. C.	116.4	340.0
	35	Cover.	30/30	1/30
		Avg. C.	138.8	366.0

Figure 2: Comparison of planner running times on the evaluation domains. TR refers to the time used for training (zero for baselines). ASNet runs with (adm.) used optimal policies for training while (inadm.) used potentially suboptimal policies, and runs with (no LM) did not use heuristic input features. The table at right shows, for selected problems, the coverage and average solution cost for the best ASNet and baseline. We use TTW for Triangle Tire World, CN for CosaNostra Pizza, and PBW for Probabilistic Blocks World. In PBW, running times are averaged over the three problems of each size. In TTW and PBW, ASNet (no LM) occludes ASNet (inadm.). ASNet (adm.) is also occluded in TTW, but is absent entirely from PBW as the optimal planner used to generate training data could not solve all training problems in time.

problem consists of a set of locations arranged in a triangle, with connections between adjacent locations. The objective is to move a vehicle from one corner of the triangle to another. However, each move has a 50% chance of producing a flat tire, which must be replaced at the next visited location. The vehicle thus requires a sequence of moves between locations where replacement tires are available. Tires are arranged such that the most reliable policy is one which travels the *longest* path to the goal, along the outside edge of the triangle. This task can be made more challenging by scaling up the number of locations. Per Little and Thiébaux (2007), a problem of size n has $(n+1)(2n+1)$ locations. We use sizes 1-3 for training, and test with sizes from 4 onward.

5.3 Results

Figure 2 shows the time taken to train and evaluate ASNet using optimal (adm.) and suboptimal (inadm.) policies as training data. In addition, it shows coverage (proportion of runs which reached the goal) and average solution cost when the goal is reached for selected problems for the best ASNet and best baseline. The following is a summary of our results:

When is it worth using ASNet? All ASNets obtained 30 out of 30 coverage for all Triangle Tire World problems, and the ASNets with heuristic input features similarly obtained perfect coverage on CosaNostra. In contrast, the baselines failed to scale up to the larger problems. This shows that ASNet is well-suited to problems where local knowledge of the environment can help to avoid common traps, for instance: in CosaNostra, the agent must learn to pay toll booth operators when carrying a pizza and not pay otherwise; and in Triangle Tire World, the agent must learn to sense and follow the outer edge of the triangle. Not only could ASNets learn these tricks, but the average solution cost obtained by ASNets for CosaNostra and Triangle Tire World was close to that of the optimal baselines (when they converged), suggesting that the optimal solution was found.

Probabilistic Blocks World is more challenging as there is no single pattern that can solve all problems. Even for the deterministic version of Blocks World, a generalised policy requires the planner to learn a recursive property for whether each block is in a goal position (Slaney and Thiébaux 2001). The ASNet appears to have successfully learnt to do this when trained by a suboptimal teacher and given landmarks as input, and surpassed all baselines in coverage (reaching the goal on 30/30 runs on each instance). Moreover, the average solution cost of ASNet (inadm.) is similar to the optimal baselines (when they converge) and up to 3.7 times less than SSiPP (inadm.), the baseline with the best coverage. The ASNet (inadm.) policy typically obtained a mean solution cost somewhere between the US and GN1 strategies presented by Slaney and Thiébaux: it is suboptimal, but still better than unstacking and rebuilding all towers from scratch. Note that the ASNet could not obtain a policy within the allotted time when trained by an optimal teacher.

Are the heuristic features necessary? In some cases, ASNet's performance can be improved by omitting (expensive) LM-cut heuristic input features. For instance, in Triangle Tire World, ASNet (inadm.) took 2.4x as much time as ASNet (no LM) to solve problems of size 15, and 4.3x as much time to solve problems of size 20, despite executing policies of near-identical average cost. Notice that this difference cannot be seen in Figure 2 because the training time (TR) is much larger than the time to solve a test instance.

Interestingly, ASNet (no LM) was able to obtain 100% coverage on the Probabilistic Blocks World problems in Figure 2, despite not receiving landmark inputs. To gain stronger assurance that it had learnt a robust policy, we tested on 10 more instances with 10, 15, 20, 25, 30 and 35 blocks (60 more instances total). ASNet (no LM) could not solve all the additional test instances. In contrast, ASNet (inadm.)—which *was* given landmarks as input—reliably solved all test problems in the extended set, thus showing that heuristic inputs are necessary to express essential recursive properties like whether a block is in its goal position.

Heuristic inputs also appear to be necessary in CosaNostra, where ASNet (no LM) could not achieve full coverage on the test set. We suspect that this is because an ASNet without heuristic inputs cannot determine which direction leads to the pizza shop and which direction leads to the customer when it is in the middle of a long chain of toll booths.

How do suboptimal training policies affect ASNet? Our results suggest that use of a suboptimal policies is sufficient to train ASNet, as demonstrated in all three domains. Intuitively, the use of suboptimal policies for training ought to be beneficial because the time that would have been spent computing an optimal policy can instead be used for more epochs of exploration and supervised learning. This is somewhat evident in CosaNostra—where a suboptimal training policy allows for slightly faster convergence—but it is more clear in Probabilistic Blocks World, where the ASNet can only converge within our chosen time limit with the inadmissible policy. While training on fewer problems allowed the network to converge within the time limit, it did not yield as robust a policy, suggesting that the use of a suboptimal teacher is sometimes a necessity.

Is ASNet performing fixed-depth lookahead search? No. This can be seen by comparing SSiPP and ASNet. SSiPP solves fixed-depth sub-problems (a generalization of lookahead for SSPs) and is unable to scale up as well as ASNets when using an equivalent depth parametrisation. Triangle Tire World is particularly interesting because SSiPP can outperform other baselines by quickly finding dead ends and avoiding them. However, unlike an ASNet, SSiPP is unable to generalize the solution of one sub-problem to the next and needs to solve all of them from scratch.

6 Related work

Generalised policies are a topic of interest in planning (Zimmerman and Kambhampati 2003; Jiménez et al. 2012; Hu and De Giacomo 2011). The earliest work in this area expressed policies as decision lists (Khardon 1999), but these were insufficiently expressive to directly capture recursive properties, and thus required user-defined *support predicates*. Later planners partially lifted this restriction by expressing learnt rules with *concept language* or *taxonomic syntax*, which can capture such properties directly (Martin and Geffner 2000; Yoon, Fern, and Givan 2002; 2004). Other work employed features from domain-independent heuristics to capture recursive properties (de la Rosa et al. 2011; Yoon, Fern, and Givan 2006), just as we do with LM-cut landmarks. Srivastava et al. (2011) have also proposed a substantially different generalised planning strategy that provides strong guarantees on plan termination and goal attainment, albeit only for a restricted class of deterministic problems. Unlike the decision lists (Yoon, Fern, and Givan 2002; 2004) and relational decision trees (de la Rosa et al. 2011) employed in past work, our model's input features are fixed before training, so we do not fall prey to the *rule utility problem* (Zimmerman and Kambhampati 2003). Further, our model can be trained to minimise any differentiable loss, and could be modified to use policy gradient reinforcement learning without changing the model. While our approach cannot give the same theoretical guarantees as Srivastava et al., we are able to handle a more general class of problems with less domain-specific information.

Neural networks have been used to learn policies for probabilistic planning problems. The Factored Policy Gradient (FPG) planner trains a multi-layer perceptron with reinforcement learning to solve a factored MDP (Buffet and Aberdeen 2009), but it cannot generalise across problems and must thus be trained anew on each evaluation problem. Concurrent with this work, Groshev et al. (2017) propose generalising "reactive" policies and heuristics by applying a CNN to a 2D visual representation of the problem, and demonstrate an effective learnt heuristic for Sokoban. However, their approach requires the user to define an appropriate visual encoding of states, whereas ASNets are able to work directly from a PPDDL description.

The integration of planning and neural networks has also been investigated in the context of deep reinforcement learning. For instance, Value Iteration Networks (Tamar et al. 2016; Niu et al. 2017) (VINs) learn to formulate and solve a probabilistic planning problem within a larger deep neural network. A VIN's internal model can allow it to learn more robust policies than would be possible with ordinary feedforward neural networks. In contrast to VINs, ASNets are intended to learn reactive policies for known planning problems, and operate on factored problem representations instead of (exponentially larger) explicit representations like those used by VINs.

In a similar vein, Kansky et al. present a model-based RL technique known as schema networks (Kansky et al. 2017). A schema network can learn a transition model for an environment which has been decomposed into entities, but where those entities' interactions are initially unknown. The entity–relation structure of schema networks is reminiscent of the action–proposition structure of an ASNet; however, the relations between ASNet modules are obtained through grounding, whereas schema networks learn which entities are related from scratch. As with VINs, schema networks tend to yield agents which generalise well across a class of similar environments. However, unlike VINs and ASNets—which both learn policies directly—schema networks only learn a model of an environment, and planning on that model must be performed separately.

Extension of convolutional networks to other graph structures has received significant attention recently, as such networks often have helpful invariances (e.g. invariance to the order in which nodes and edges are given to the network) and fewer parameters to learn than fully connected networks. Applications include reasoning about spatio-temporal relationships between variable numbers of entities (Jain et al. 2016), molecular fingerprinting (Kearnes et al. 2016), visual question answering (Teney, Liu, and Hengel 2017), and reasoning about knowledge graphs (Kipf and Welling 2017). To the best of our knowledge, this paper is the first such technique that successfully solves factored representations of automated planning problems.

7 Conclusion

We have introduced the ASNet, a neural network architecture which is able to learn generalised policies for proba-

bilistic planning problems. In much the same way that CNNs can generalise to images of arbitrary size by performing only repeated local operations, an ASNet can generalise to different problems from the same domain by performing only convolution-like operations on representations of actions or propositions which are *related* to one another. In problems where some propositions are only related by long chains of actions, ASNet's modelling capacity is limited by its depth, but it is possible to avoid this limitation by supplying the network with heuristic input features, thereby allowing the network to solve a range of problems.

While we have only considered supervised learning of generalised policies, the ASNet architecture could in principle be used to learn heuristics or embeddings, or be trained with reinforcement learning. ASNet only requires a model of which actions affect which portion of a state, so it could also be used in other settings beyond SSPs, such as MDPs with Imprecise Probabilities (MDPIPs) (White III and Eldeib 1994) and MDPs with Set-Valued Transitions (MDP-STs) (Trevizan, Cozman, and Barros 2007). We hope that future work will be able to explore these alternatives and use ASNets to further enrich planning with the capabilities of deep learning.

References

Bertsekas, D., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming*. Athena Scientific.

Blum, A. L., and Furst, M. L. 1997. Fast planning through planning graph analysis. *AIJ*.

Bonet, B., and Geffner, H. 2003. Labeled RTDP: improving the convergence of real-time dynamic programming. In *AAAI*.

Buffet, O., and Aberdeen, D. 2009. The factored policy-gradient planner. *AIJ*.

Clevert, D.-A.; Unterthiner, T.; and Hochreiter, S. 2016. Fast and accurate deep network learning by exponential linear units (ELUs). *ICLR*.

de la Rosa, T.; Jiménez, S.; Fuentetaja, R.; and Borrajo, D. 2011. Scaling up heuristic planning with relational decision trees. *JAIR*.

de la Rosa, T.; Celorrio, S. J.; and Borrajo, D. 2008. Learning relational decision trees for guiding heuristic planning. In *ICAPS*.

Groshev, E.; Tamar, A.; Srivastava, S.; and Abbeel, P. 2017. Learning generalized reactive policies using deep neural networks. *arXiv:1708.07280*.

Hansen, E. A., and Zilberstein, S. 2001. LAO: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: what's the difference anyway? In *ICAPS*.

Hu, Y., and De Giacomo, G. 2011. Generalized planning: Synthesizing plans that work for multiple environments. In *IJCAI*.

Jain, A.; Zamir, A. R.; Savarese, S.; and Saxena, A. 2016. Structural-RNN: Deep learning on spatio-temporal graphs. In *CVPR*.

Jiménez, S.; de la Rosa, T.; Fernández, S.; Fernández, F.; and Borrajo, D. 2012. A review of machine learning for automated planning. *Knowl. Eng. Rev.*

Kansky, K.; Silver, T.; Mély, D. A.; Eldawy, M.; Lázaro-Gredilla, M.; Lou, X.; Dorfman, N.; Sidor, S.; Phoenix, S.; and George, D. 2017. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *ICML*.

Kearnes, S.; McCloskey, K.; Berndl, M.; Pande, V.; and Riley, P. 2016. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*.

Khardon, R. 1999. Learning action strategies for planning domains. *AIJ*.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Little, I., and Thiébaux, S. 2007. Probabilistic planning vs. replanning. In *ICAPS workshops*.

Martin, M., and Geffner, H. 2000. Learning generalized policies in planning using concept languages. In *KRR*.

Mausam, and Kolobov, A. 2012. *Planning with Markov Decision Processes*. Morgan & Claypool.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing Atari with deep reinforcement learning. In *NIPS workshops*.

Niu, S.; Chen, S.; Guo, H.; Targonski, C.; Smith, M. C.; and Kovačević, J. 2017. Generalized value iteration networks: Life beyond lattices. *arXiv:1706.02416*.

Slaney, J., and Thiébaux, S. 2001. Blocks world revisited. *AIJ*.

Srivastava, S.; Immerman, N.; Zilberstein, S.; and Zhang, T. 2011. Directed search for generalized plans using classical planners. In *ICAPS*.

Tamar, A.; Wu, Y.; Thomas, G.; Levine, S.; and Abbeel, P. 2016. Value iteration networks. In *NIPS*.

Teichteil-Königsbuch, F.; Vidal, V.; and Infantes, G. 2011. Extending Classical Planning Heuristics to Probabilistic Planning with Dead-Ends. In *AAAI*.

Teney, D.; Liu, L.; and Hengel, A. v. d. 2017. Graph-structured representations for visual question answering. In *CVPR*.

Trevizan, F., and Veloso, M. 2014. Depth-based Short-sighted Stochastic Shortest Path Problems. *Artificial Intelligence*.

Trevizan, F.; Cozman, F. G.; and Barros, L. N. 2007. Planning under risk and knightian uncertainty. In *IJCAI*.

Vallati, M.; Chrpa, L.; Grześ, M.; McCluskey, T. L.; Roberts, M.; Sanner, S.; et al. 2015. The 2014 International Planning Competition: Progress and trends. *AI Mag.*

White III, C. C., and Eldeib, H. K. 1994. Markov decision processes with imprecise transition probabilities. *Operations Research* 42(4):739–749.

Yoon, S.; Fern, A.; and Givan, R. 2002. Inductive policy selection for first-order MDPs. In *UAI*.

Yoon, S.; Fern, A.; and Givan, R. 2004. Learning reactive policies for probabilistic planning domains. In *IPC Probabilistic Track*.

Yoon, S. W.; Fern, A.; and Givan, R. 2006. Learning heuristic functions from relaxed plans. In *ICAPS*.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. Using learned policies in heuristic-search planning. In *IJCAI*.

Younes, H. L., and Littman, M. L. 2004. PPDDL1.0: an extension to PDDL for expressing planning domains with probabilistic effects.

Zimmerman, T., and Kambhampati, S. 2003. Learning-assisted automated planning: looking back, taking stock, going forward. *AI Mag.*

Augmenter les processus de décision Markoviens via des conseils

Loïs Vanhée Laurent Jeanpierre Abdel-Illah Mouaddib

GREYC, Université de Caen, France *

lois.vanhee@unicaen.fr

Résumé

Ce document présente Advice-MDPs, une extension des processus de décision Markoviens permettant de générer des politiques prenant en compte les conseils relatifs au caractère désirable, indésirable et interdit de certains états et actions. Les Advice-MDPs permettent de concevoir des systèmes semi-autonomes (systèmes nécessitant le support de l'opérateur pour au moins certaines situations), capables de gérer efficacement des environnements complexes et inattendus. Via le conseil, les opérateurs améliorent le modèle de planification afin de couvrir les irrégularités inattendues du monde réel. Ce conseil peut rapidement augmenter le degré d'autonomie du système, de sorte qu'il fonctionne sans besoin d'intervention humaine ultérieure.
Ce document détaille le formalisme Advice-MDP, un algorithme de résolution rapide pour les Advice-MDPs et une démonstration de l'applicabilité des Advice-MDPs pour résoudre des tâches du monde réel, via la conception d'un système de robot semi-autonome de classe professionnelle, prêt à être déployé dans une large gamme d'environnements et capable d ?intégrer efficacement les conseils des opérateurs.

Mots Cles

Décision sous incertitude, Semi-autonomie, Processus de décision Markoviens, Applications.

Abstract

This paper introduces Advice-MDPs, an expansion of Markov Decision Processes for generating policies that take into consideration advising on the desirability, undesirability, and prohibition of certain states and actions. Advice-MDPs enable designing semi-autonomous systems (systems that require operator support for at least handling certain situations) that can efficiently handle unexpected complex environments. Operators, through advising, can augment the planning model for covering unexpected real-world irregularities. This advising can swiftly augment the degree of autonomy of the system, so it can work without subsequent human intervention.
This paper details the Advice-MDP formalism, a fast Advice-MDP resolution algorithm, and its applicability for real-world tasks, via the design of a professional-class semi-autonomous robot system ready to be deployed in a wide range of unexpected environments and capable of efficiently integrating operator advising.

Keywords

Decision under uncertainty ; Semi-Autonomy ; Markov Decision Processes

1 Introduction

Markov Decision Processes (MDPs) are one of the most classic model for planning under uncertainty. However, MDPs are difficult to deploy for many real-world problems (e.g. robots in public space, military operations, disaster-recovery). These problems introduce irregularities that are difficult to foresee at design time (e.g. path-finding hardened by wind, water, glass, acid, armed threats). These irregularities cause discrepancies between the planning model and the real-world, thus entailing the generation of irrelevant plans and failures.
Mixed-Initiative Planning Systems (MIPS) aim to overcome these issues by providing means for human operators to improve plan generation through the validation of actions and plans, and by providing information that the system cannot access or infer otherwise (e.g. presence of threat) [16]. However, operator support should be requested with care, as it burdens the operators, causing Operator-Workload (OW), which is costly.
Available MIPS suffer from three major limitations : 1) *OW costs are linear to system use* (e.g. a check for each generated plan), 2) *no flexibility on OW management* (operator support must be performed last minute, causing time pressure, frustration, and stress [5]), and 3) the *system requires an expensive training phase*. Plan-validation based approaches suffer from limitations 1 and 2, while learning-based approaches suffer from all three limitations. This paper aims to answer the following research question : *how to decrease OW costs and enable flexible OW management, with limited training phase ?* The key novelty brought by this paper lies in covering environmental irregularities by providing means for the user to *augment the planning model* using (prescriptive) advising. This way, operator support is required only once for multiple plan generations

*Research funded by the Agence Nationale de la Recherche and the Direction Générale de l'Armement (ANR-14-ASTR-0018).

(thus lowering OW costs), can be performed both *a priori* and on-demand (before and when plans are to be generated), and no expensive training is required for the system to be operational.

This paper proposes a new formalism for overcoming these issues, which we call *Advice-MDPs* (Advice Markov Decision Processes). Advice-MDPs are Markov Decision Processes (MDPs [7]) expanded with advising, for defining situationally-forbidden actions and augmenting state definition with an external information on whether it is desired, undesired, or forbidden. The various types of advice are interpreted following an order, from the hardest (forbidden) to the softest (desired) : we aim to generate advice-compliant policies that first avoid performing forbidden situational actions, then optimize the compromise between reaching goals and avoiding forbidden states, and then optimize the compromise between reaching desirable states and avoiding undesirable states.

In addition to the formalism, this paper proposes a fast approximate algorithm for solving Advice-MDPs. This algorithm is based on a lexicographic multi-criteria value-iteration algorithm using Ordered Weighted Regret [11]

Finally, this paper demonstrates the relevance of Advice-MDPs for solving real-world problems, by deploying them for a professional-class application. Our industry partner seeks to deploy mobile-robots in unexpected and complex environments, for contexts such as supporting the evacuation of civilians in harmful situations, disaster recovery [8], and security missions (e.g. terrorism, patrol in sensitive sites).

The system, which we call the “Anywhere Deployment" Robot System (ADRS), should be extremely flexible i.e. capable of adapting to unexpected irregularities that can affect robot orientation (e.g. wind, water, sand, acidity, glass, unstable ground, threats), which go beyond the possibilities of classic fully-autonomous systems. Furthermore, operator availability is highly limited and unpredictable : while operators can generally watch and support the system for a few minutes after deployment, they have little time for actively training it and, suddenly, they may become overloaded due to arising threats. Illustrations of the implementation of the ADRS are presented in Figure 1 and Figure 2.

FIGURE 1 – Two NERVA robots we experimented with, in the realistic arena set by our industry partners.

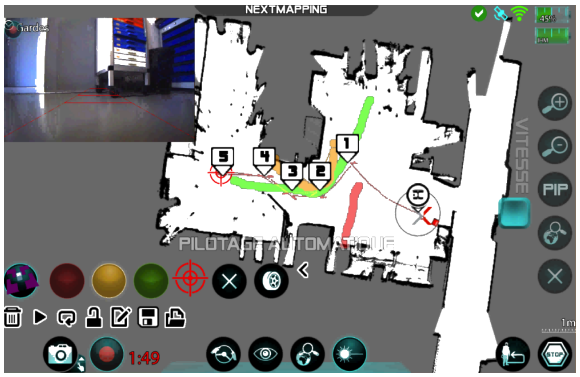


FIGURE 2 – Operator interface : map, video stream. Lines on the map represent operator advising : desired (green), undesired (orange), and forbidden (red) advising. The target shape is the goal of the robot. Numbers are navigation waypoints.

2 Related Research

Mixed-Initiative Planning Systems (MIPSs) : [16] define MIPSs as automated systems that collaboratively build plans with their operators. MIPS are acknowledged to be particularly useful when the planning model of the automated system is partial and when human judgment is needed to evaluate the plans generated based on the model [16].

Classic approaches for implementing MIPSs consists of an iterative loop where the system generates a plan and operators validate or refuse it, with a mechanism for justifying the rejection. If accepted, the plan is executed by the system. If refused, the system generates another plan, taking into consideration the reasons for the rejection of the first plan (e.g. the MAPGEN system, for planning Mars rover activities). The design of these systems is generally coupled with the design of interfaces for enabling operators to evaluate plans and argue on their rejection.

Advanced (MDP-based) MIPS approaches aim to reduce OW costs by reasoning about the operators’ workload and capabilities and cost for requesting support from them. The HHP-MDP, HOP-POMDP, and OPOMDP formalisms integrate operators as a manageable (costly) resource of the planning domain, which can be used for acquiring observations and setting subgoals [2, 3, 10]. However, these approaches require from the system to be capable of evaluating the relevance of asking operator support, something which can be difficult to achieve.

MIPSs suffer from the limitations 1) and 2) presented in the introduction : OW costs are proportional to system use, as plans must be validated each time they are issued. Second, little OW flexibility is possible, as plans must be validated after goals are issued and before plans are executed. Operator support cannot be done *a priori* and delayed support generally entails neglect costs or even system failures [4].

Advice-Sensitive Reinforcement Learning (ASRL) :

Advising has been applied for speeding up the learning process of Reinforcement Learning [1, 6, 14]. In ASRL, operators can advise learning agents by providing feedback on the desirability of (intended) agent actions, or by relating the task to similar ones.

ASRL faces the three limitations presented in the introduction (high OW costs, low OW management flexibility, expensive training phase). The training phase is particularly expensive in terms of OW costs, as operators are constantly involved during this period. Furthermore, reinforcement learning may be infeasible due to real-world constraints (e.g. the delay for being rewarded by a human is incompatible with a robot affected by world-dynamics such as sliding on ice). In addition, this form of advising only tweaks reward and transition functions, disregarding finer-grained advising that can introduce a richer semantic and behavioral indications for the system (e.g. a “bystanders may be encountered there" advice).

3 Background on Markov Decision Processes

Classic Markov Decision Processes : Markov Decision Processes (MDPs) are stochastic processes that are (partially) controlled by an agent [11]. Formally, a MDP is represented by a tuple (S, A, T, R), where S is a set of states ; A is a set of actions ; $T : S \times A \times S \to [0, 1]$ is a transition function, where $T(s, a, s')$ is the probability of reaching s' when playing a from s ; and $R : S \to \mathbb{R}$ is the reward function, where $R(s)$ is the expected reward for reaching s.

Optimizing MDPs within a bounded horizon $h \in \mathbb{N}$ consists in finding a policy that maximizes the expected reward in using h actions. A policy $\pi : S \to A$ defines an action to be played for each state and Π is the set of all policies. The value of a policy π within a horizon h is $V_h^\pi : S \to \mathbb{R}$, where : $V_k^\pi(s) = R(s) + \sum_{s' \in S} T(s, \pi(s), s').V_{k-1}^\pi(s')$ if $k > 0$, else $V_0^\pi(s) = R(s)$. The set of optimal policies for the horizon h, represented by $\Pi_h^* \subseteq \Pi$ is the set of policies $\pi^* \in \Pi_h^*$ such that for all $s \in S$, $V_h^{\pi^*}(s) = \max_{\pi \in \Pi} V_h^\pi(s)$.

Multi-Objective MDPs : In Multi-Objective MDPs (MOMDPs) [9], the agent controls a process that is related to not one, but a set of reward criteria (e.g. money, satisfaction, environment quality). The reward function of MOMDPs is represented by $R : S \to \mathbb{R}^k$. Optimizing MOMDPs is influenced by this multiplicity of criteria : the set of optimal policies is a Pareto-front of policies along $\mathbb{R}^k$ (e.g. two policies, one generating less money, but offering higher satisfaction, belong to the same Pareto-front). Often, MOMDPs are turned into MDPs using a formula for linearizing the MOMDP rewards (e.g. $R(s) = 0.5 \times R_{mon}(s) + 0.7 \times R_{sat}(s) + 2.3 \times R_{env}(s)$). However, doing so reduces the capability for expressing more elaborated reward profiles and constraints (e.g. the value of environment is minimal, as long as ten points are scored).

4 Formalizing Advice-MDPs

Advice-MDPs aim to enable operators to *augment the planning model with advice*, towards fitting the irregularities of the environment (like ASRL) with prescriptive *a priori* advising (like MIPSs). This way, Advice-MDPs cut down the OW costs of MIPS, by requiring operators to augment the planning model for including real-world irregularities once for all instead of once per use. Advice-MDPs offer higher OW management flexibility, by enabling advising *a priori* (once the environment is known, without needing goals to be set) instead of last-minute advising for MIPSs ; Advice-MDPs require no training phase and advising can be achieved without executing plans, unlike ASRL approaches. Advice-MDPs enable for a large variety of advice types, which can introduce a helpful semantic context for advanced decision-making, unlike ASRL approaches that only enable to tweak mono-dimensional rewards.

Advice-MDPs expand classic MDPs, replacing reward function with advising. This paper considers the following five *advice types* : $\{\mathcal{G}, S_f, S_d, S_u, C_{\hat{\pi}}\}$, where $\mathcal{G} \subseteq S$ are goal states, to be reached by the system, $S_f \subseteq S$ are forbidden states, $S_d \subseteq S$ are desired states, $S_u \subseteq S$ are undesired states, and $C_{\hat{\pi}} \in f : S \to 2^A$ are forbidden actions to be avoided in certain states. In our ADRS application, S_d are safe and efficient zones, S_u are slow or inconvenient zones (e.g. sand, rock), and S_f are dangerous zones that should be avoided, unless inevitable (e.g. very poor floor, very close to enemy location), $C_{\hat{\pi}}$ are contextually forbidden actions (e.g. activating a visible sensor). The *relative importance* given to the various advice types follows a lexicographic order : first, match $C_{\hat{\pi}}$; then, best satisfy the tradeoff $\langle \mathcal{G}, S_f \rangle$; finally, best satisfy the trade-off $\langle S_d, S_u \rangle$. Formally, policies matching $C_{\hat{\pi}}$ are represented by :

$$\Pi_{C_{\hat{\pi}}} = \Pi \setminus \{\pi | \exists s \in S, \pi(s) \in C_{\hat{\pi}}(s)\}$$

Policies that best satisfy the tradeoff $\mathcal{G}, S_f$ belong to the Pareto-front of policies evaluated by $V^{\mathcal{G},S_f,\pi} = (V^{\mathcal{G},\pi}, -V^{S_f,\pi})$, where : $V^{S',\pi}(s) = \mathcal{N}_{vis}(S'|s, \pi)$, for which $\mathcal{N}_{vis}(S'|s, \pi)$ is the expected number of times S' is expected to be visited from s following π. Policies that best satisfy the tradeoff S_d, S_u belong to the Pareto-front of policies evaluated by $V^{S_d,S_u,\pi} = (V^{S_d,\pi}, -V^{S_u,\pi})$. Note that this representation for advising is selected for its simplicity, genericity, and ease to connect to user interfaces. However, more advice types can be integrated in the model.

Different models for advising can be considered. A straightforward model consists in integrating advising within a classic MDP reward function. However, such approach reduces advising to a linearly addable and tradeable reward. This linearity causes undesirable side-effects : without tweaking the world-model, this form of reward cannot represent that losing three weels during run is ten times worse than losing two ; desirable states can be traded for undesirable states with a uniform cost. A more advanced approach consists in using non-Markovian rewards [13].

These rewards are acquired when some conditions defined by temporal logic formulas are met, rather than when a state is reached. Non-Markovian rewards only partly mitigates the issues of linear rewards : they do offer capabilities for modelling rewards, though rewarding still introduces a form of linearity (as rewards are acquired every time the condition is met) at the expenses of greater computational costs. To our experience, advising through non-Markovian rewards is often impractical, as manipulating temporal formulas is unhandy for (untrained) operators, cognitively expensive and the additional computational time reduces the fluidity of interaction. The Advice-MDP representation is simpler for operators, which lowers risks of errors. They introduce a semantic context for applying advising and for easing system management (e.g. a slider for setting the balance between “following desirable" and “avoiding undesirable", a slider for balancing the importance between achieving goals and the tradeoff desirable/undesirable). This operation is less intuitive with linear MDPs, as it implies to reason about fiddling reward functions. Last Advice-MDPs simplifie the introduction of a wider range of advice-types (e.g. a “can be pushed", “can annoy bystanders"), without having to re-design the reward function as a whole.

5 Efficiently Solving Advice-MDPs

Algorithme 1 : Fast Advice-MDP policy computer

Input : $S, A, T, \mathcal{G}, S_f, S_d, S_u, C_{\hat{\pi}}, H$
Output : MDP Value Function V'
$T \leftarrow T \setminus \{(s,a,s') | \forall s, s', a \in C_{\hat{\pi}}(s)\}$
$V^{\pi,0} \leftarrow 0$
$\forall s \in S, R'(s) \leftarrow \begin{cases} (1,0,0,0) & \text{if } s \in \mathcal{G} \\ (0,-1,0,0) & \text{if } s \in S_f \\ (0,0,1,0) & \text{if } s \in S_d \\ (0,0,0,-1) & \text{if } s \in S_u \end{cases}$
for $t = 0 \dots H-1$ **do**
  foreach $s \in S$ **do**
    foreach *criterion* $c \in \{\mathcal{G}, S_f, S_d, S_u\}$ **do**
      $V_c^{\pi_c^*,t+1}(s) \leftarrow R_c(s) +$
        $\max_{a \in A} \sum_{s' \in S} T(s,a,s').V_c^{\pi_c^*,t}(s')$
      $ideal_c \leftarrow$
      $\max_a R_c(s) + \sum_{s' \in S} T(s,a,s').V_c^{\pi,t}(s')$
      $antiIdeal_c \leftarrow$
      $\min_a R_c(s) + \sum_{s' \in S} T(s,a,s').V_c^{\pi,t}(s')$
      foreach $a \in A$ **do**
        $Rgt_c^t(s,a,\pi) \leftarrow$
        $\frac{||[R_c(s)+\sum_{s' \in S} T(s,a,s').V_c^{\pi,t}(s')] - V_c^{\pi_c^*,t+1}(s)||}{||ideal_c^{t,\pi}(s) - antiIdeal_c^{t,\pi}(s)||}$
    $optA \leftarrow argmin_a lexdiff(C_{\hat{\pi}}, Rgt_{\mathcal{G}}(s,a,\pi) +$
    $Rgt_{S_f}(s,a,\pi), Rgt_{S_d}(s,a,\pi) + Rgt_{S_u}(s,a,\pi))$
    $V^{\pi,t+1}(s) \leftarrow$
    $R(s) + \sum_{s' \in S} T(s, optA, s').V^{\pi,t}(s')$
  return V

We resolve Advice-MDPs by transforming them into MOMDPs. As a transformation, actions belonging to $C_{\hat{\pi}}$ are removed from the base T, and each advice-type is mapped to a criterion, which is rewarded when the system reaches an accordingly-advised state (e.g. $R_{S_d}(s)$ is 1 (else 0) if $s \in S_d$). Rewards are positive for desirable or goal states and negative otherwise. Since rewards are summed, the value of criterion c when following π is the number of expected visitations $\mathcal{N}^{\pi}_{vis,c}$ (e.g. “V(s)=(2,0,5,-7)" when following π for h steps means : “from s, following π is expected to lead to the reach of 2 goals, 0 forbidden, 5 desirable, 7 undesirable states). Optimal solutions for this MOMDP are Pareto-optimal tradeoffs that maximize the number of visited goals and desired states, and minimize the number of forbidden and undesired states.

Our fast approximate algorithm [1] for solving Advice-MDPs (Algorithm 1) is inspired by Algorithm 10.1 from [11, p.325]Exact MOMDP-resolution algorithms [15] were left out due to their exponential complexity and our need for responsiveness. At each iteration, our algorithm back-propagates for each state the set of most promising expected multidimensional value for each action (and ideal and anti-ideal values for computing the relative value of each state, see next paragraph). Then, a *comparator function* selects the action leading to the most promising reward tuple, given the immediate choice (e.g. “(2,0,5,-7)" is better than “(2,-1,5,0)").

The MOMDP comparator is based on the LexDiff operator [11] while maintaining the priority order of advice types. The LexDiff operator computes, for each criterion, the regret for playing each action with respect to estimated ideal and anti-ideal reference points for this criterion. The regret for criterion c when playing action a in state s given a policy π for a horizon h is :

$$Rgt_c^h(s,a,\pi) = \frac{||V_c^{\pi,h}(s) - V_c^{\pi_c,h}(s)||}{||ideal_c^{h,\pi}(s) - antiIdeal_c^{h,\pi}(s)||}$$

where $V_c^{\pi,t}$ is expected total reward for c from following π for t steps, π_c is the “pure" policy that single-mindedly maximizes c for one step (discarding other criteria) and then plays π, $ideal_c^{t,\pi(s)}$ (respectively $antiIdeal_c^{t,\pi}(s)$) is the highest (respectively lowest) expected reward for c that can be acquired from s for all actions and then following π. $||ideal_c^{t,\pi}(s) - antiIdeal_c^{t,\pi}(s)||$ makes the regret relative to the relative grasp that the system has on c. If $ideal_c^{t,\pi}(s) = antiIdeal_c^{t,\pi}(s)$, then $Rgt_c^h(s,a,\pi) = 0$ (no regret when no action can change the outcome, though this value has no impact on the outcome as, in this case, no action can change the regret). Moreover, we maintain a strict priority order between sets of criteria (in our case, $\{C_{\hat{\pi}}\} > \{\mathcal{G}, S_f\} > \{S_d, S_u\}$), i.e. lower-rank criteria are only considered when higher-rank criteria are equally satisfied. Note that this comparator function can be changed for altering the reaction of the system with regards to advising (e.g. allowing to trade the crossing of one forbidden state for avoiding many undesirable states).

1. Note that ideal and anti-ideal values are computed in parallel by the algorithm.

In terms of solution quality, in our experiments, our algorithm always finds optimal MDP policies when undesirable and forbidden states can be avoided. When being forced to cross undesirable and forbidden states, our algorithm generates very satisfactory trajectories (avoiding as much as possible forbidden states, then a relatively conservative balance between avoiding undesirable states and reaching desirable states).

6 Professional-Grade Application

6.1 Application Context

Our industry partners aim to deploy highly flexible mobile robots for supporting operators for contexts such as disaster-recovery [8], scouting in risky areas (e.g. wildlife monitoring, terrorism, illegal activities), and surveying in sensitive sites (e.g. chemical factory, nuclear plant).

Robots are given missions such as live observations (camera feed, establishing maps, monitoring threats) and situational actions (e.g. open a door, disarm traps). The robot we work with are NERVA robots illustrated in Figure 1, which are actually already being deployed by our partners, exclusively controlled through teleoperation. NERVAs are very robust (can be thrown over obstacles) and fast (up to 15km/h). They are equipped with four cameras and a wide array of specific sensors can be plugged in for capturing situational measurements (e.g. sound, gas, explosives, radioactivity).

As an application for intersecting both industrial and academic goals, we aim to make the robot capable of reaching as autonomously as possible set of locations that is required for completing sensing tasks. This goal revisits the very classical pathfinding problem, except that the many irregularities of varied, difficult, and unexpected real-world situations are added to the equation. These irregularities introduce two challenges making fully-autonomy currently unfeasible : first, available models fail to incorporate all these irregularities and the design of such an all-encompassing model would be very difficult and likely to be computationally too expensive ; second, robot sensory capabilities are sometimes insufficient for recognizing such irregularity (e.g. the robot, only equipped with cameras, faces a puddle in a chemical plant : can the puddle be recognized ? Can the puddle be crossed ? Is it deep ? Is it a chemical that will destroy the robot ? If so, how fast ?).

We called our system the Anywhere Deployment Robot System (ADRS). An overview of the system loop presented in Figure 3. The ADRS generates a map of the surroundings via a Simultaneous Localization and Mapping (SLAM) using a laser. This map is then displayed with a Graphical User Interface (GUI), presented in Figure 2, on a tablet that operators wear on their wrist. This GUI enables operators to advise on goals, desirable, undesirable and forbidden areas, by simply clicking on an “advice-type" button and drawing on the map (points, lines, shapes). Then, the system generates an advice-compliant waypoint-based trajectory that minimizes the total distance crossed for reaching all goals. This trajectory is displayed back to the operator, who decides whether the robot should apply it (a “rush mode" can be turned on for skipping this check). At any time, before and during plan execution, the operator can update the advising, which then updates the waypoint trajectory. Likewise, robot trajectories are updated if the SLAM discovers new obstacles that jeopardize the current plan. Towards achieving high responsiveness and smooth interactions with operators, our fast Advice-MDP resolution algorithm enables *completing a whole sense-reason-act loop within two seconds* [2], which is satisfactory for operators. Special cases can be improved through engineering (e.g. having dodge reactive behaviors during replanning, so the robots does not become an easy target).

As an overall idea of the system behavior, and thus of the advising semantics, desirable areas (in green) mark zones that are interesting to be crossed by robots (safe ground, concealed from opposing threats), while still generally moving towards the goal. Undesirable areas (in orange), mark zones that are best avoided if possible, but that can be crossed for the sake of reaching the goal, or many more desirable areas (e.g. populated non-hostile areas, unsteady ground that slows the robot, slow acid or irradiated areas that will destroy the robot after the mission). Red areas mark zones that may harm the mission, to be avoided unless they there is no other means for reaching the goal (e.g. risk of being spotted, slippery ground that may cause crashes, unidentified puddle). These categories remain flexible and can change due to situational context (e.g. slow acid areas may be set to forbidden if losing the robot after its mission becomes an issue for follow-up actions).

Technically, the generated map is a 4096×4096 pixel map. Each pixel captures a 4cm^2 square. For keeping high responsiveness, this map is abstracted into a 400×400-tiles hexagonal grid (any tile containing at least one obstacle pixel is considered as an obstacle tile). This grid is then transformed into a canonical transition function for moving on grids (e.g. playing “East" on state/tile “(4,12)" leads to state/tile “(5,12)"). The optimal policy, π^*, provides a pathway, i.e. a sequence of sets of optimal tiles per iteration. As a straight transformation of this pathway creates highly inefficient step-by-step “stop-and-turn" move, a smoothing phase fastens robot moves by greedily searching for the waypoint path with the least number of steps that is totally covered by the optimal pathway. This step is illustrated in Figure 4a. Note that, for descriptive simplicity, this implementation relies on a deterministic MDP. However, Advice-MDPs can be applied on non-deterministic MDPs (e.g. uncertainty on the tile to be reached when playing move actions).

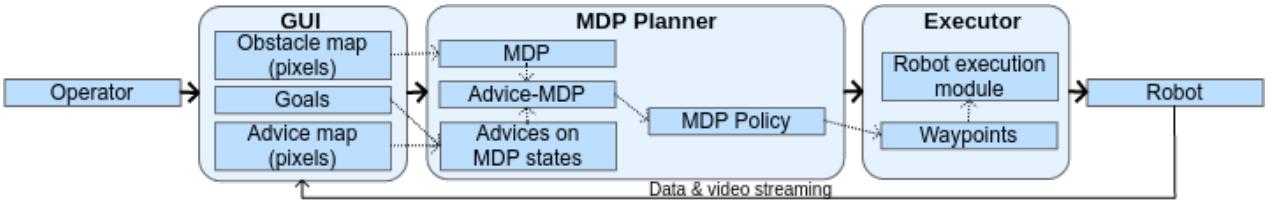
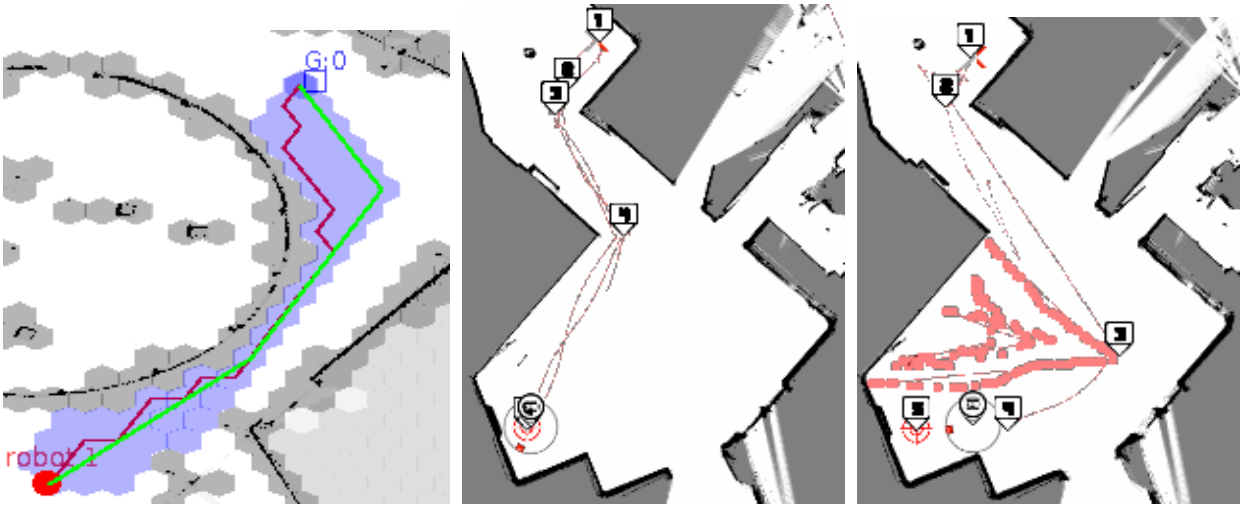


FIGURE 3 – System loop : operator and sensor input are merged and displayed by the GUI. When goals are set, this input is turned into Advice-MDPs, given to the planner for generating MDP policies, turned into waypoints, smoothed, and executed by the robot.



(a) Optimal pathway towards the goal (blue tiles), considering obstacles (grey tiles). Red lines are the base waypoint path generated from the base MDP policy (many stop and turn operations). Green lines represented the smoothed path (minimizing the amount of waypoint, exploiting diagonals).

(b) Advice-MDP waypoint trajectories for the corridor scenario. Lines are former robot paths.

(c) Advice-MDP trajectories for the hole scenario. Red marks are "forbidden" advice, drawn by the operator.

FIGURE 4 – Pathways on hexagonal and pixel maps

7 Empirical Evaluation

7.1 Varying The Degree of Autonomy

We compared Advice-MDPs against Fully-Autonomous Systems (FAS, based on classic advice-less MDPs) and Non-Autonomous systems (i.e. teleoperation), along the following criteria, partly drawn from [5, 12] : 1) *flexibility* (range of situations that can be handled by the system) ; 2) *efficiency* (time for completing the mission) ; 3) *Operator Workload (OW)* (time required from operators for having the task completed, i.e. for teleoperation and advising) ; 4) *effectiveness* (percentage of the mission that is autonomously completed) ; 5) *OW flexibility* (flexibility left for the operator to decide when to operate), 6) *responsiveness* (time from task-allocation to action).

2. Demonstration videos are available at `https://www.youtube.com/playlist?list=PLcULZgrd6hEgBYO6zJ74S7tH3DvX3Sf7g`

As a setup, we compared the performance of FAS, Advice-MDPs, and teleoperation in two scenarios of differing degrees of flexibility : in the corridor scenario, the robot is tasked to move to the other side of a building (Figure 4b) ; in the hole scenario, a (virtual) hole in the ground is to be avoided, else the robot risks breaking (Figure 4c). Each experiment was repeated 20 times. Each experimenter was trained to teleoperate and advise the robot for at least ten minutes. The weight values of OWR are all equal, while still preserving the precedence of goal/forbidden over desirable/undesirable.

Experimental results (Table 1) detail the compromises between efficiency, flexibility, and OW costs. In terms of flexibility, FASs only succeed in the corridor scenario, as they risk getting stuck in the hole scenario ; the Advice-MDPs systems handle both the corridor and hole missions ; teleoperation handles all situations. Teleoperation remains more flexible than Advice-MDPs, as humans can make situatio-

TABLE 1 – Average autonomy effectiveness, efficiency, and operator workload costs, over 20 repetitions for the corridor and hole scenarios, depending on the system type. Lines marked by an asterisk note the OW cost during a training phase.

Scenario	System	Autonomy Effectiveness	Efficiency (s)	OW (s)
Corridor	Advice-MDPs	100%	33.7	0
Corridor	Teleop	0%	22.3	22.3
Corridor	Base MDPs	100%	34.5	0
Corridor	Action Pruning (ASRL)	NA	269.9	348.8*
Corridor	Plan Revision (MIPS)	95.1%	35.0	1.7
Hole	Advice-MDPs	81.3%	43.4	8.1
Hole	Teleop	0%	27.2	27.2
Hole	Base MDPs	100%	**Falls**	0
Hole	Action Pruning (ASRL)	NA	316.7	472.7*
Hole	Plan Revision (MIPS)	54.6%	75.2	34.9

nal assumptions that fall out of the planning model (e.g. seeing empty crates as pushable objects rather than impassable obstacles).

In terms of OW costs, FASs are obviously the most effective, followed by Advice-MDPs, which only require one fast intervention for augmenting the planning model, and teleoperation last, which is much more expensive. Note that, once augmented, Advice-MDPs can operate with no subsequent OW costs, unless major changes in the environment (e.g. a zone catches fire).

In terms of efficiency, teleoperation is the most efficient ; Advice-MDPs and FASs are equivalent (teleoperation allows for higher speed and avoiding ineffective stop-and-turn actions at waypoints).

In terms of effectiveness, Advice-MDPs achieve a score of 81.3% when augmenting the planning model is needed, which is very reasonable. In terms of OW flexibility, Advice-MDPs offers the highest OW-flexibility as advising can be done a priori, while teleoperation offers no OW-flexibility (must be done last minute, any delay lowers the efficiency of the system). Last, in terms of responsiveness, the software responds within two seconds for FASs and Advice-MDPs.

7.2 Comparing Against Related Approaches

This section compares Advice-MDPs against the two related approaches : MIPS and ASRL approaches. The experimental results are detailed in Table 1.

For the MIPS approaches, the operator is in charge of validating or rejecting the trajectories generated by the system. When rejecting a trajectory, operators indicate at which point the trajectory is incorrect. Then, the system removes the indicated state, replans and proposes an alternative trajectory.

In terms of performance, Advice-MDPs achieve slightly better efficiency for lower OW costs for the corridor experiment, as operators are asked to perform an unnecessary check for the MIPS (although, the cost is relatively low). For the hole experiment, Advice-MDPs achieve much better performance than MIPS, as rejecting plans until one becomes acceptable incurs low efficiency and high OW costs. Furthermore, after one planning phase, the Advice-MDPs system is ready to be operated without subsequent OW costs, while the whole process is to be repeated for the MIPS approach.

For the ASRL approach, a module has been implemented using the most relevant technique from [1] for this context : action pruning. During the training phase, the system asks before playing an action whether this action is the most relevant. If not, the system associates it with a low reward and proceeds with the new best action. Results highlight the that the Advice-MDP approach overcomes the ASRL approach. First, Advice-MDP avoids a very slow and OW-cost expensive learning phase, totally in 98 operator checks for the corridor scenario and 123 operator checks for the hole scenario. Second, the Advice-MDP approach enables for an effective prescription of all undesired states, while the ASRL approach only learns what is related to this specific trajectory. While some knowledge is re-used, the system is to be trained again for moving back to the initial position. Furthermore, during this learning phase, the robot can only rely on brute MDP-based plans, which imply operating on the level of “stop and turn" low-level actions, which dramatically slow down robot operations.

8 Conclusions and Perspectives

This paper introduces Advice-MDPs, a new formalism for designing systems capable of integrating operator support for augmenting planning models towards efficiently adapting the behavior of the system to real-world unexpected irregularities. Advice-MDPs expand the Markov Decision Process (MDP) formalism with the possibility to advise on the desirability, undesirability, or prohibition of states and actions. Furthermore, this paper introduces a fast algorithm for generating near-optimal advice-compliant policies, making possible to create reactive interaction-loops with operators for smoothing the task of supporting the robot.

We empirically demonstrate the benefits of Advice-MDPs against fully autonomous planning systems (Advice-MDPs trade reasonable operator workload costs for greater system flexibility), fully teleoperated systems (Advice-MDPs can dramatically decrease the operator workload costs), classic mixed-initiative planning systems (Advice-MDPs

lower operators workload costs, the invested time from operator is better re-used over multiple planning operations, operators can better manage when to advise the system), and reinforcement-learning based planning systems (Advice-MDPs skip a long learning phase, thus decreasing operator workload costs and making the system much faster to set up). Conceptually, the core novelty and benefits offered by Advice-MDPs compared to classic approaches lies in enabling operators to *prescriptively augment the world-model*. This augmentation can then be used for multiple subsequent planning operations, be given early on (before planning is required), and does not require system training. Though, the benefits of advising depend on external factors, such as the advising quality (e.g. mistakes from operators) and unexpected environmental changes (e.g. new areas becoming forbidden or desirable), which can require additional efforts from operators (e.g. watching the environment and the robot, updating advising) and introduce risks of failure.

We demonstrate the applicability of these theoretical findings, by deploying Advice-MDPs in the design of a professional-grade robotic system, which we called the Anywhere Deployment Robot System (ADRS). The ADRS has been tested by its intended final users (our industry partners), who were highly satisfied by the flexibility features offered by Advice-MDPs and the relief the ADRS brings on the cognitive strain required for operating the system. The ADRS is being integrated within the robotic platform of our industry partners.

Regarding future work, we are expanding Advice-MDPs for integrating more advice types (e.g. a “passable" advice type for handling e.g. box-pushing behavior). Additionally, we are expanding our application to a multi-robot setting, considering new challenging issues in term of multi-agent planning and multi-robot coordination. Furthermore, we are investigating the combination between Advice-MDP and HOP-POMDP for combining the benefits of both approaches [10].

Acknowledgements The authors wish to thank Melania Borit for her support improving the text and the reviewers for their constructive feedback.

Références

[1] D. Abel, J. Salvatier, A. Stuhlmüller, and O. Evans. Agent-Agnostic Human-in-the-Loop Reinforcement Learning. *arXiv preprint arXiv :1701.04079*, 2017.

[2] N. Armstrong-Crews and M. Veloso. Oracular partially observable markov decision processes : A very special case. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2477–2482. IEEE, 2007.

[3] N. Côté, A. Canu, M. Bouzid, and A.-I. Mouaddib. Humans-Robots Sliding Collaboration Control in Complex Environments with Adjustable Autonomy. In *Proceedings of Intelligent Agent Technology*, 2012.

[4] J. W. Crandall, M. A. Goodrich, D. R. Olsen, and C. W. Nielsen. Validating human-robot interaction schemes in multitasking environments. *IEEE Transactions on Systems, Man, and Cybernetics-Part A : Systems and Humans*, 35(4) :438–449, 2005.

[5] S. G. Hart and L. E. Staveland. Development of NASA-TLX (Task Load Index) : Results of empirical and theoretical research. *Advances in psychology*, 52 :139–183, 1988.

[6] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations : Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

[7] M. L. Puterman. *Markov Decision Processes : Discrete Stochastic Dynamic Programming*, volume 10 of *Wiley Series in Probability and Mathematical Statistics : Applied Probability and Statistics*. John Wiley & Sons, Inc., 1994.

[8] S. D. Ramchurn, T. D. Huynh, Y. Ikuno, J. Flann, F. Wu, L. Moreau, N. R. Jennings, J. E. Fischer, W. Jiang, T. Rodden, E. Simpson, S. Reece, and S. J. Roberts. HAC-ER : A Disaster Response System Based on Human-Agent Collectives. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’15, pages 533–541, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.

[9] D. Roijers, P. Vamplew, and S. Whiteson. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48 :67–113, 2013.

[10] S. Rosenthal and M. Veloso. Modeling humans as observation providers using pomdps. pages 53–58. IEEE, 2011.

[11] O. Sigaud and O. Buffet. *Markov Decision Processes in Artificial Intelligence*. Wiley-IEEE Press, 2010.

[12] A. Steinfeld, T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, and M. Goodrich. Common metrics for human-robot interaction. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 33–40. ACM, 2006.

[13] S. Thiébaux, C. Gretton, J. Slaney, D. Price, and F. Kabanza. Decision-theoretic planning with non-Markovian rewards. *Journal of Artificial Intelligence Research*, 25 :17–74, 2006.

[14] L. Torrey, T. Walker, J. Shavlik, and R. Maclin. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *ECML*, pages 412–424. Springer, 2005.

[15] K. Wakuta. A multi-objective shortest path problem. *Mathematical Methods of Operations Research*, 54(3) :445–454, 2001.

[16] S. Zilberstein. Building Strong Semi-Autonomous Systems. In *Proceedings of the Twenty-Ninth Conference on Artificial Intelligence*, pages 4088–4092, Austin, Texas, 2015.