AfIA

Association française
pour l'Intelligence Artificielle

JFPDA

Journées Francophones sur
la Planification, la Décision et l'Apprentissage
pour la conduite de systèmes

PFIA 2022

Table des matières

Éditorial

Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes

Les Journées Francophones Planification, Décision et Apprentissage (JFPDA) ont pour but de rassembler la communauté des chercheurs francophones travaillant sur : les problèmes d'apprentissage par renforcement, de la théorie du contrôle, de programmation dynamique et plus généralement dans les domaines liés à la prise de décision séquentielle sous incertitude et à la planification. La conférence JFPDA est soutenue par le Collège Représentation et Raisonnement de l'AFIA.

En plus des articles présents dans cet ouvrage, nous avons eu le plaisir d'accueillir Hector Geffner (ICREA and Universitat Pompeu Fabra, Barcelona, Spain, Linköping University, Linköping, Sweden) en tant qu'orateur invité pour un exposé intitulé *Language-based representation learning for acting and planning*. Nous avons également eu le plaisir d'accueillir Tristan Cazenave (Université Paris-Dauphine, LAMSADE, CNRS) pour un exposé intitulé *Monte Carlo Search*.

Je tiens à remercier chaleureusement tous les membres du comité de programme, qui ont effectué un travail de relecture. Ils et elles ont fourni des rapports de relecture détaillées et pertinents. Enfin, je tiens à remercier chaleureusement toute l'équipe du comité d'organisation de PFIA, qui nous fait un accueil exemplaire au sein de la PFIA.

François SCHWARZENTRUBER

Comité de programme

Président

— François Schwarzentruber (ENS Rennes, IRISA).

Membres

— Pegah Alizadeh (University of Caen Normandy) ;
— Aurélie Beynier (LIP6, Sorbonne Université) ;
— Olivier Buffet (INRIA / LORIA) ;
— Martin Cooper (Université Toulouse 3, IRIT) ;
— Alain Dutech (Loria - Inria) ;
— Humbert Fiorino (LIG) ;
— Jérôme Lang (CNRS, LAMSADE, Université Paris-Dauphine) ;
— Frédéric Maris (Université Toulouse 3, IRIT) ;
— Laetitia Matignon (LIRIS CNRS) ;
— Damien Pellier (Laboratoire d'Informatique de Grenoble) ;
— Sophie Pinchinat (Université de Rennes 1, IRISA, Rennes) ;
— Philippe Preux (Université de Lille) ;
— Emmanuel Rachelson (ISAE-SUPAERO) ;
— Régis Sabbadin (INRAE) ;
— Francois Schwarzentruber (École normale supérieure de Rennes, IRISA, Rennes) ;
— Olivier Sigaud (ISIR, UPMC) ;
— Vincent Thomas (LORIA) ;
— Bruno Zanuttini (GREYC, Normandie Univ. ; UNICAEN, CNRS, ENSICAEN) ;
— Paul Weng (UM-SJTU Joint Institute).

Session 1

Planification robuste pour la collaboration homme-robot

Yang You[1], Vincent Thomas[1], Francis Colas[1], Rachid Alami[2], Olivier Buffet[1] *

[1] Université de Lorraine, INRIA, CNRS, LORIA, F-54000 Nancy, France
[2] LAAS-CNRS, Université de Toulouse, CNRS, F-31000 Toulouse, France

firstname.lastname@(loria|laas).fr

Résumé

Du point de vue du robot, une difficulté majeure de la collaboration homme-robot est d'être robuste face à des objectifs incertains de l'humain, et aux comportements incertains étant donné un objectif connu. Une question préliminaire clef est alors : Comment dériver des comportements humains réalistes étant donné un objectif connu ? En effet, pour rendre la collaboration possible, de tels comportements devraient aussi tenir compte du comportement du robot, alors que celui-ci n'est justement pas connu. Dans cet article, nous nous appuyons sur des modèles de décision markoviens, représentant l'incertitude sur l'objectif de l'humain par une distribution de probabilité sur un ensemble fini de fonctions de récompense (ce qui induira une distribution sur des comportements humains). Sur cette base, nous proposons deux contributions : 1. un algorithme de planification pour le robot qui est robuste au comportement incertain de l'humain et repose sur la résolution d'un POMDP obtenu en raisonnant sur la distribution sur les comportements humains ; et 2. une approche pour générer automatiquement un comportement humain incertain (une politique) pour chaque fonction de récompense fournie tout en tenant compte du comportement possible du robot. Un scénario de travail collaboratif permet de mener des expérimentations et de présenter des résultats qualitatifs et quantitatifs pour évaluer notre approche.

Mots-clés

Collaboration humain-robot, POMDP, planification robuste.

Abstract

From the robot's point of view, a major issue in human-robot collaboration is how to be robust against uncertain human objectives, and uncertain human behaviors given a known objective. A key preliminary question is then : How to derive realistic human behaviors given a known objective ? Indeed, to allow for collaboration, such behaviors should also account for the robot behavior, while it is not known in the first place. In this paper, we rely on Markov decision models, representing the uncertainty over the human objective as a probability distribution over a finite set of reward functions (what will induce a distribution over human behaviors). Based on this, we propose two contributions : 1. a robot planning algorithm that is robust to the uncertain human behavior and relies on solving a POMDP obtained by reasoning on the distribution over human behaviors ; and 2. an approach to automatically generate an uncertain human behavior (a policy) for each provided reward function while accounting for the possible robot behavior. A co-working scenario allows conducting experiments and presenting qualitative and quantitative results to evaluate our approach.

Keywords

Human-Robot Collaboration, POMDP, Robust Planning.

1 INTRODUCTION

Concevoir des robots intelligents pour assister un partenaire humain est un sujet d'actualité avec des applications dans l'industrie manufacturière [1]-[4], la santé [5], etc. Dans ces applications, le robot a souvent besoin de s'adapter à un objectif de l'humain fixé. Mais pour rendre ce robot assistant robuste, nous devons considérer comment ce robot pourrait s'adapter si l'objectif de l'humain et son comportement induit étaient incertains.

Pour contourner cette incertitude sur les objectifs de l'humain, HADFIELD-MENELL et al. [6] ont proposé le cadre CIRL (*Cooperative Inverse Reinforcement Learning*). Dans CIRL, l'humain et le robot doivent tous deux maximiser la récompense de l'humain. Mais la fonction de récompense de l'humain est paramétrée par une variable cachée au robot, ce qui conduit à un problème multi-agent sous observabilité partielle pour lequel une politique jointe optimale est calculée. CIRL a toutefois deux inconvénients : 1. CIRL calcule une politique jointe (pour l'humain et le robot) et fait l'hypothèse que l'humain doit suivre cette politique calculée. Cette hypothèse est souvent non-réaliste parce que la politique calculée pour l'humain peut être trop complexe pour être exécutée ou apprise par l'humain. 2. CIRL repose sur un environnement MDP avec une seule variable cachée θ correspondant à l'objectif de l'humain (paramètre de la fonction de récompense). Ce cadre assure que la croyance du robot sur θ est une statistique suffisante, faisant de CIRL un POMDP. Cependant, si l'état est partiellement observable, CIRL ne peut être réduit à un POMDP mais à un Dec-POMDP, problème qui est NEXP complet [7].

*This work was supported by the French National Research Agency (ANR) through the "Flying Coworker" Project under Grant 18-CE33-0001.

Dans cet article, nous considérons un humain indépendant, et devons donc tenir compte de l'incertitude sur son comportement, laquelle peut être due i. à l'incertitude sur son objectif (la fonction de récompense qu'il considère), et ii. aux différents comportements (politiques) qu'il pourrait adopter pour un objectif donné. Notre première contribution est ainsi un algorithme de planification pour le robot qui repose sur la résolution d'un POMDP où une des variables d'état cachées correspond au comportement humain actuel, chaque comportement étant représenté par un contrôleur à états fini (FSC). Ensuite, pour dériver des comportements humains réalistes pour un objectif donné, nous souhaiterions que l'humain tienne compte de la possibilité que le robot collabore, ce qui induit un paradoxe similaire au paradoxe de l'œuf et de la poule puisque nous ne disposons pas du comportement du robot à ce stade. Aussi, au lieu de dériver plusieurs comportements humains (FSC) pour un objectif donné, nous allons dériver un unique FSC représentant plusieurs comportements. Pour ce faire, notre seconde contribution consiste en 1. la résolution du problème en faisant l'hypothèse que le robot et l'humain partagent leurs observations (de sorte que le problème est un (M)POMDP, pas un Dec-POMDP), et 2. ensuite l'extraction d'un contrôleur à états fini stochastique (FSC) où une action a d'autant plus de chances d'être choisie qu'elle est prometteuse. En outre, comparé à CIRL, ni le robot ni l'humain n'ont accès à l'état réel de l'environnement.

La section 2 discute des travaux connexes en collaboration homme-robot. La section 3 définit formellement les POMDP, Dec-POMDP et FSC. La section 4 décrit comment concevoir une politique robuste pour le robot afin qu'il s'adapte à plusieurs politiques humaines possibles. La section 5 explique comment générer un FSC stochastique pour l'humain de manière automatique pour une fonction de récompense donnée. Enfin, la section 6 présente des résultats empiriques obtenus sur une tâche de haut-niveau dans un environnement simulé, et les analyse avant de conclure.

2 Travaux connexes

On peut distinguer différentes approches pour la collaboration homme-robot selon le "modèle mental de l'humain" (dont dispose le robot), lequel, d'après TABREZ et al. [8], peut appartenir à l'une des trois catégories suivantes : modèle mental du premier ordre *(first-order mental model - 1oMM)* : le robot considère que l'humain se comporte de manière indépendante et ne tient pas compte des actions possibles du robot ; modèle mental du second ordre *(second-order mental model - 2oMM)* : le robot considère que l'humain tient compte du robot, ce qui induit une forme de modélisation récursive entre humain et robot jusqu'à une certaine profondeur ; modèle mental partagé *(shared-mental model - SMM)* : un SMM suppose que tous les agents dans le groupe ont des attentes communes, donc raisonnent de la même manière, ce qui assure une coordination optimale. Évidemment cette catégorisation s'applique symétriquement à la modélisation du robot par l'humain.

Nous nous concentrons ici sur des problèmes avec des dynamiques stochastiques et une observabilité partielle, ce qui nous conduit à considérer des modèles de décision markoviens. Dans ce contexte, un 1oMM correspond typiquement à un POMDP dont l'objectif est d'optimiser la politique d'un agent en particulier, alors que les politiques (a priori connues) de l'autre agent font partie de la dynamique du système. Par exemple, un POMDP pour le robot supposant un comportement humain connu est résolu dans [9]-[12]. Les SMM peuvent être formalisés comme des *POMDP décentralisés* (Dec-POMDPs), typiquement utilisés pour optimiser la politique jointe d'une équipe d'agents (partageant une même fonction de récompense). CIRL peut être vu comme un cas particulier où l'humain a une observabilité complète et la seule variable cachée du robot correspond à l'objectif réel de l'humain (sa fonction de récompense), ce qui autorise des techniques de résolution dédiées proches de la résolution d'un POMDP. Pour leur part, les 2oMM peuvent être formalisés comme des *POMDP interactifs* (I-POMDPs [13]), dans lesquels les agents se modélisent mutuellement de manière imbriquée. Ainsi, les I-POMDP soulèvent aussi le paradoxe de l'œuf et de la poule qu'il faut résoudre.

Les 1oMM échoueront dans de nombreuses tâches nécessitant un comportement collaboratif explicite de l'humain, et l'hypothèse principale des SMM est typiquement trop forte quand on collabore avec un humain. Nous allons donc doter le robot d'un modèle mental du 2nd ordre de l'humain. Notre approche de la planification robuste pour le robot pourrait être formalisée comme un I-POMDP, ce que nous évitons principalement pour simplifier les notations. [1] Toutefois, les 2oMM soulèvent toujours le paradoxe de l'œuf et de la poule puisque le comportement humain que nous recherchons nécessite de raisonner sur le comportement du robot que nous cherchons justement à dériver au départ. C'est par exemple vrai dans le cas 1. de la spécification manuelle d'une politique humaine [9], [10], où le concepteur humain doit raisonner sur le comportement du robot ; 2. de l'apprentissage d'une politique humaine [14], [15] ou d'une récompense humaine (par apprentissage par renforcement inverse) [16], [17], ce qui requiert un robot collaboratif pour faire la démonstration d'un comportement collaboratif ; ou 3. de la planification d'une politique humaine : le modèle doit inclure le comportement du robot.

Dans notre travail, nous employons une approche par planification pour générer des comportements humains plausibles, et traitons le paradoxe de l'œuf et de la poule en répondant à la question : "Et si l'humain pouvait aussi contrôler le robot ?" : 1. Nous supposons que l'humain peut contrôler le robot et a directement accès à ses observations passées, ce qui revient à faire adopter à l'humain un SMM (modèle mental partagé), et dérivons un problème mono-agent ; 2. en utilisant un solveur en-ligne, nous extrayons une politique humaine sous la forme d'un FSC stochastique.

1. Aussi, nous faisons finalement face à des POMDP, pour lesquels il est préférable d'employer un solveur de l'état de l'art.

3 État de l'art

3.1 Dec-POMDP

Par commodité, nous utilisons les Dec-POMDP pour formaliser le problème de collaboration, mais ne résolvons pas de Dec-POMDP pour obtenir une politique jointe pour l'humain et le robot.

Definition 3.1 *Un* Dec-POMDP *avec $|\mathcal{I}|$ agents est défini par un tuple $M \stackrel{\text{def}}{=} \langle \mathcal{I}, \mathcal{S}, \mathcal{A}, \Omega, T, O, R, b_0, \gamma \rangle$, où : $\mathcal{I} = \{1, \ldots, |\mathcal{I}|\}$ est un ensemble d'*agents*; $\mathcal{S}$ est un ensemble d'*états*; $\mathcal{A} = \bigtimes_i \mathcal{A}^i$ est l'ensemble des actions jointes, avec $\mathcal{A}^i$ l'ensemble des* actions *de l'agent i; $\Omega = \bigtimes_i \Omega^i$ est l'ensemble des observations jointes, avec Ω^i l'ensemble des* observations *de i; $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ est la* fonction de transition*, avec $T(s, a, s')$ la probabilité de transiter de s à s' si a est effectuée; $O : \mathcal{A} \times \mathcal{S} \times \Omega \to \mathbb{R}$ est la* fonction d'observation*, avec $O(a, s', o)$ la probabilité d'observer o si a est effectuée et le prochain état est s'; $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ est la* fonction de récompense*, avec $R(s, a)$ la récompense immédiate pour l'exécution de a dans s; b_0 est la* distribution de probabilité initiale *sur les états; et $\gamma \in [0, 1)$ est le* facteur d'atténuation *appliqué aux récompenses futures.*

La *politique* d'action π^i d'un agent i associe à chacun de ses historiques d'action-observation une action à effectuer. L'objectif est alors de trouver une politique jointe $\pi = \langle \pi^1, \ldots, \pi^{|\mathcal{I}|} \rangle$ qui maximise l'espérance du retour atténué à partir de b_0 :

$$V^\pi(b_0) \stackrel{\text{def}}{=} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \mid S_0 \sim b_0, \pi\right].$$

Dans un POMDP, *c.-à-d.* quand $\mathcal{I} = \{1\}$, de nombreux solveurs reposent sur l'estimation de la fonction de valeur optimale $V^*(b)$, ou la fonction d'action-valeur $Q^*(b, a) \stackrel{\text{def}}{=} R(b, a) + \gamma \sum_o Pr(o|b, a) \cdot V^*(b_a^o)$, où b_a^o est le prochain état de croyance quand on effectue a et que l'on observe o.

Note : Dans notre contexte, nous allons considérer un problème à 2 agents (un humain et un robot) un peu différent puisqu'on remplace l'unique fonction de récompense r par une distribution sur des fonctions de récompense possibles, seul l'humain sachant quelle est la bonne fonction de récompense, et que l'on cherche à concevoir le comportement du robot, mais en raisonnant sur les comportements possibles de l'humain.

3.2 Contrôleurs à états finis

Dans notre travail, les politiques humaines sont représentées sous la forme de *contrôleurs à états finis* (FSC) (aussi appelés *graphes politiques* [18]), *c.-à-d.* des automates qui contiennent dans chaque état interne une distribution de probabilité à partir de laquelle une action est échantillonnée, et dont les transitions d'un état interne au suivant dépendent de l'action exécutée et de l'observation reçue.

Definition 3.2 *Pour des ensembles $\mathcal{A}$ et Ω d'un POMDP, un* FSC *stochastique est défini par un tuple $fsc \stackrel{\text{def}}{=} \langle N, \beta, \eta, \psi \rangle$, où :*

- *N est un ensemble fini de nœuds (internes);*
- *β est une distribution de probabilités à partir de laquelle échantillonner un nœud initial;*
- *$\eta : N \times \mathcal{A} \times \Omega \times N \to \mathbb{R}$, la fonction de transition, donne la probabilité $\eta(n, \langle a, o \rangle, n')$ de transiter du nœud n à n' si a est effectuée et o est observée; une transition déterministe peut être notée $n' = \eta(n, \langle a, o \rangle)$;*
- *$\psi : N \times \mathcal{A} \to \mathbb{R}$, la fonction de sélection d'action, donne la probabilité $\psi(n, a)$ de choisir $a \in \mathcal{A}$ depuis n.*

4 Planification de tâches robuste pour le robot

Nous voulons ici que le robot soit robuste aux différents objectifs (cachés) possibles de l'humain, chacun attaché à des comportements différents, et à même de suivre l'objectif réel de l'humain. Notre problème est formalisé comme un Dec-POMDP, si ce n'est que 1. la fonction de récompense exacte (parmi ρ candidates) n'est connue que de l'humain; 2. pour chaque fonction de récompense possible r_i, il y a des comportements humains associés (sous la forme d'un FSC fsc_i); et 3. le robot connaît la distribution de probabilité sur les paires possibles récompense-FSC : $P(\{(r_1, fsc_1), \ldots, (r_\rho, fsc_\rho)\})$. Comme détaillé ci-dessous, ce comportement de robot robuste est obtenu d'abord en transformant cette distribution sur des FSC en un unique FSC, puis en utilisant ce FSC pour dériver un POMDP, et finalement en résolvant ce POMDP. Une solution candidate pour obtenir un FSC pour chaque fonction de récompense est présentée en section 5.

Pour convertir cette distribution de probabilité sur des FSC humains en un unique FSC, nous prenons "l'union" de ces FSC, la nouvelle distribution sur les nœuds initiaux revenant à 1. échantillonner un FSC fsc_i de la distribution $P(\{fsc_1, \ldots, fsc_\rho\})$, et 2. échantillonner un nœud de la distribution β_i. Plus formellement, en notant un FSC de base $fsc_i \stackrel{\text{def}}{=} \langle N_i, \beta_i, \eta_i, \psi_i \rangle$, le *FSC union* est défini comme :

$$N \stackrel{\text{def}}{=} \bigcup_{i=1}^{\rho} N_i,$$

$$\beta(n_i) \stackrel{\text{def}}{=} \beta_i(n_i) \cdot P(fsc_i),$$

$$\eta(n, \langle a, o \rangle, n') \stackrel{\text{def}}{=} \begin{cases} \eta_{i(n)}(n, \langle a, o \rangle, n') & \text{si } n' \in N_{i(n)}, \\ 0 & \text{sinon,} \end{cases}$$

(où $i(n) \stackrel{\text{def}}{=} i$ t.q. $n \in N_i$, *c.-à-d.*, $i(n)$ est l'identifiant du FSC auquel appartient n au départ), et

$$\psi(n, a) \stackrel{\text{def}}{=} \psi_{i(n)}(n, a).$$

Étant donné ce FSC, nous pouvons maintenant formaliser le problème de décision du robot comme un POMDP dans lequel chaque *état étendu* $e^t \in \mathcal{E}$ contient un état courant du monde s^t, une observation courante du robot o_R^t et le nœud courant n_H^t du FSC union de l'humain : $e^t = \langle s^t, n_H^t, o_R^t \rangle$.

En partant du Dec-POMDP définissant la tâche et du FSC union associé, les fonctions de transition, d'observation et de récompense du problème de décision du robot peuvent s'écrire comme suit :

$$T_e(e^{t+1}, e^t, a_R^t) = Pr(e^{t+1}|e^t, a_R^t)$$
$$= \sum_{a_H^t} \sum_{o_H^{t+1}} T(s^t, \langle a_H^t, a_R^t\rangle, s^{t+1}) \cdot \eta(n_H^t, \langle a_H^t, o_H^{t+1}\rangle, n_H^{t+1}) \cdot O(s^{t+1}, \langle a_H^t, a_R^t\rangle, \langle o_H^{t+1}, o_R^{t+1}\rangle) \cdot \psi(n_H^t, a_H^t),$$
$$O_e(e^{t+1}, a_R^t, o_R^{t+1}) = Pr(o_R^{t+1}|e^{t+1}, a_R^t) = \mathbf{1}_{o_R^{t+1}=\tilde{o}_R^{t+1}}$$

(où $\tilde{o}_R^{t+1}$ est l'observation dans e^{t+1}), et

$$r_e(e^t, a_R^t) = \sum_{a_H^t} r_{i(n_H^t)}(s^t, \langle a_H^t, a_R^t\rangle) \cdot \psi(n_H^t, a_H^t).$$

Résoudre ce POMDP associé au robot fournit une politique du robot robuste qui constitue une meilleure réponse à la distribution de probabilité initiale sur les politiques humaines données.

5 Générer des politiques humaines à partir d'objectifs

Nous décrivons maintenant comment générer un FSC stochastique pour l'humain en partant d'une fonction de récompense. Ce processus peut être réglé pour créer des comportements plus rationnels ou plus erratiques.

En effet, nous voulons traiter l'incertitude non seulement sur l'objectif réel de l'humain (sa fonction de récompense), comme dans CIRL, mais aussi concernant le comportement humain exhibé pour atteindre un tel objectif. Pour chaque objectif possible de l'humain, nous créons un Dec-POMDP pour le problème de collaboration. Ces Dec-POMDP ne diffèrent qu'en leurs fonctions de récompense, chacune modélisant un objectif spécifique (de l'humain). Pour chaque tâche associée à un tel Dec-POMDP, nous souhaiterions dériver automatiquement une politique stochastique humaine qui représente des comportements humains possibles guidés par ses propres objectifs, et qui tienne compte des interactions possibles avec le robot. Comme mentionné plus haut, une difficulté est que ces politiques humaines dépendent des politiques du robot dont nous ne disposons justement pas. Pour surmonter ce paradoxe de l'œuf et de la poule, nous supposons d'abord que l'humain peut contrôler les actions du robot et a accès aux observations du robot. Chaque Dec-POMDP est ainsi relâché en un MPOMDP (POMDP multi-agent) [19], *c.-à-d.* un problème qui devient mono-agent. [2] Aussi, pour tenir compte de l'incertitude sur les comportements de l'humain, nous laisserons l'humain faire un choix au hasard parmi plusieurs actions optimales ou quasi-optimales en employant une fonction softmax sur les valeurs d'action : $f(a_H|T, b) = e^{\frac{Q^*(b,a_H)}{T}} / \sum_{a'} e^{\frac{Q^*(b,a')}{T}}$, où $T > 0$ est un paramètre de température qui rend l'humain plus *rationnel* si la température T est basse, seules des actions optimales étant sélectionnées, et plus *erratique* si la température T est élevée, avec des distributions presque uniformes.

Suivant ces idées, nous extrayons des politiques humaines (représentées par des FSC) en utilisant l'algorithme 1, lequel est en partie inspiré par l'extraction de solutions POMDP de GRZEŚ et al. [20]. D'abord, nous calculons une distribution initiale sur les actions de l'humain P_{A_H} avec l'état de croyance initial b_0 (ligne 2). Un nœud de départ unique est créé (l. 3) avec P_{A_H}, b_0 et un poids initial $w = 1$. Ensuite n_0 est ajouté à la fois au nouveau FSC (N) et à une liste ouverte (G) (l. 4). En s'inspirant de LAO* [21], nous souhaitons étendre à chaque itération le nœud qui a la plus grande contribution à la valeur de b_0. Ainsi, tant que G n'est pas vide, nous sélectionnons le nœud $n \in G$ qui a la plus haute valeur estimée $V^*(n.b)$, pondérée par la probabilité d'atteindre ce nœud (estimée à travers $n.w$) (l. 6), puis nous traitons ce nœud avec les paires observation-action possibles sous l'état de croyance courant b (l. 7–9), les paires observation-action impossibles induisant des boucles (l. 22). Les lignes 10–12 calculent l'état de croyance mis à jour b', sa distribution sur les actions P_{A_H}, et son poids w'. Avant de créer un nouveau nœud n' avec ces composants nouvellement calculés, nous devons aussi vérifier : 1. si le nouvel état de croyance b' (ou un état de croyance proche) n'existe pas déjà dans le FSC (N) en considérant un seuil de distance ϵ en norme-1 ; 2. si le FSC ne contient pas déjà le nombre de nœuds maximum $N_{\max}$. Si les deux conditions sont satisfaites (l. 13), nous créons un nouveau nœud n' et l'ajoutons à N. Sinon, nous cherchons et traitons le nœud n' de N dont l'état de croyance est le plus proche du nouvel état de croyance b'. Avec cette approche, nous pouvons extraire des FSC stochastiques de taille bornée encodant plusieurs comportements humains.

Dans nos expérimentations, toutes les valeurs $Q^*(b, a)$ et $V^*(b)$ sont estimées en utilisant l'algorithme POMCP [22], de manière à obtenir de bonnes estimations rapidement, même pour des états de croyance non visités par une politique optimale.

6 Expérimentations

6.1 Cadre expérimental

Nos expérimentations ont été conduites sur un ordinateur doté d'un microprocesseur i9 à 2,3 GHz. Le code source sera rendu disponible sous licence MIT.

Pour tester notre approche, nous avons conçu un scénario présenté figure 1. Dans ce scénario, un robot et un humain doivent réparer et maintenir plusieurs appareils situés dans un monde grille. Parmi les trois appareils présents sur la grille, un appareil doit être maintenu par le robot, qui doit se trouver sur la cellule de cet appareil pour effectuer l'opération ; deux autres appareils sont en panne, chacun requérant que l'humain et le robot accomplissent tous deux simultanément des actions de réparation à l'endroit où se trouve l'appareil. De plus, pour réparer un appareil, l'humain doit détenir un composent qu'il ne peut prendre que dans une

2. Ici, en effet, une seule entité reçoit toutes les observations et contrôle toutes les actions.

Algorithme 1 : Extraction d'une politique humaine sous la forme d'un FSC stochastique

[Input :] T : température softmax | $N_{\max}$: # max de nœuds | ϵ : distance max entre croyances
$P_{A_H} \leftarrow f(\cdot|T, b_0)$ // `softmax dist. sur les actions`
$n_0 \leftarrow node(\langle P_{A_H}, b_0, w = 1\rangle)$ // `nœud de départ`
$N \leftarrow \{n_0\}$; $G.pushback(n_0)$
while $|G| > 0$ **do**
  $G.sort()$ // `tri des nœuds selon` $w \cdot V^*(b)$
  $n = \langle b, P_{A_H}, w\rangle \leftarrow G.popfront()$
  forall $\langle o_h, a_h\rangle \in \Omega_h \times A_h$ **do**
    if $Pr(o_h, a_h|b, P_{A_H}) > 0$ **then**
      $b' \leftarrow BeliefUpdate(b, a_h, o_h)$
      $P'_{A_H} \leftarrow f(\cdot|T, b')$
      $w' \leftarrow w \cdot Pr(o_h, a_h|b, P_{A_H})$
      if $(b' \notin N(\epsilon)) \wedge (N.size() < N_{\max})$ **then**
        $n' \leftarrow node(P'_{A_H}, b', w')$
        $N \leftarrow N \cup \{n'\}$; $G.pushback(n')$
        $\eta(n, \langle a, o\rangle, n') \leftarrow 1$
      else
        $n' \leftarrow N.find(b')$
        $n'.w \leftarrow n'.w + w'$
        $\eta(n, \langle a, o\rangle, n') \leftarrow 1$
    else
      $\eta(n, \langle a, o\rangle, n) \leftarrow 1$
return $\langle N, \eta, \psi\rangle$

boîte à outil à une position spécifique. Notons aussi que l'humain comme le robot n'ont qu'une perception limitée de l'environnement.

Cette tâche est spécifiée à travers le Dec-POMDP suivant :

- États (S) : L'état $s \in S$ du problème est composé de : 1. la position de l'humain, 2. la position du robot, 3. le statut de chaque appareil, et 4. la détention ou non d'un composant par l'humain. Les positions de l'humain et du robot sont représentées par des coordonnées entières (x, y). Ils peuvent se trouver sur la même cellule. Le statut de chaque composant est soit "bon" (réparé), "en panne", ou "nécessitant maintenance".
- Observations de l'humain (Ω_H) : L'humain observe 1. sa position, 2. si le robot est sur la même cellule que lui ou pas, 3. le statut de l'appareil dans sa cellule (s'il y en a un).
- Observations du robot (Ω_R) : Le robot observe 1. sa position, 2. la position de l'humain, 3. le statut de l'appareil dans sa cellule (s'il y en a un).
- Actions et dynamique : L'humain et le robot disposent tous deux des actions suivantes : *Up, Down, Left, Right* (haut, bas, gauche, droite) qui sont les 4 actions de déplacement des agents ; *Attendre* : l'agent reste dans sa position courante ; *Réparer* : répare un appareil en panne si : 1. l'humain détient un nouveau composant ; 2. l'humain et le robot sont sur la même cellule que l'appareil en panne ; 3. l'humain et le robot effectuent tous deux l'action de réparation. En cas de succès, l'appareil en panne devient réparé et le composant de l'humain est consommé.
  L'humain peut *Prendre un composant* s'il se trouve dans la cellule de la boîte à outil et s'il n'en détient pas déjà un. Le robot peut *Maintenir* un appareil tout seul si l'appareil en a besoin et s'ils sont dans la même cellule. En cas de succès, le statut de l'appareil devient *bon*.
- Récompenses R : Une récompense de $+100$ est donnée quand tous les appareils ont été réparés et maintenus. Une récompense (pénalité) de -2 est donnée pour chaque action sauf pour l'action *Attendre* de l'humain, et une pénalité de -20 est donnée en cas d'action invalide. Une pénalité de -1 est donnée si l'humain attend alors que tous les appareils en panne n'ont pas encore été réparés. S'il n'y a plus d'appareils en panne, aucune pénalité n'est donnée en cas d'action *Attendre* de l'humain. La pénalité associée à l'action *Attendre* encourage le robot à repousser les actions de maintenance si cela permet à l'humain de fini les réparations tôt.

Ce grand Dec-POMDP a 2 304 états, 49 actions jointes (7 actions par agent), et 5 400 observations jointes (30 pour l'humain et 180 pour le robot). Note : Cet espace d'état croît de manière quadratique avec le nombre de cellules.

Pour représenter l'incertitude sur les objectifs de l'humain, nous remplaçons la fonction de récompense décrite précédemment par deux variantes. Dans la première, l'humain préfère réparer l'appareil de gauche en premier, recevant une récompense de $+10$ dans ce cas. Dans la seconde, symétriquement, l'humain préfère réparer l'appareil de droite en premier. Ces objectifs génèrent des politiques humaines différentes en utilisant l'algorithme 1. Initialement, le robot ne dispose que d'une distribution a priori sur les récompenses de l'humain et la politique associée (chacune avec probabilité $0,5$). Du fait de la coordination requise pour réparer les appareils, ces politiques humaines doivent tenir compte de l'aptitude du robot à aider l'humain quand c'est nécessaire.

6.2 Résultats qualitatifs

Nous utilisons la procédure décrite en section 5 pour calculer les FSC stochastiques de l'humain associés à chaque objectif de l'humain, et celle décrite en section 4 pour calculer la politique robuste du robot avec le solveur de POMDP SARSOP [23]. Pour économiser des ressources, un *seuil d'action* (ici toujours fixé à $0,1$) est utilisé pour élaguer les actions humaines de basse probabilité lors du calcul des FSC stochastiques de l'humain. Nous avons conduit des expérimentations avec 2 valeurs pour le paramètre de température T ($0,3$ et $0,5$) et 5 valeurs pour la taille maximale des FSC stochastiques $N_{\max}$ (voir table 1).

Nous avons d'abord observé que le FSC stochastique extrait pour l'humain peut effectivement encoder des trajectoires possibles de l'humain permettant de résoudre la

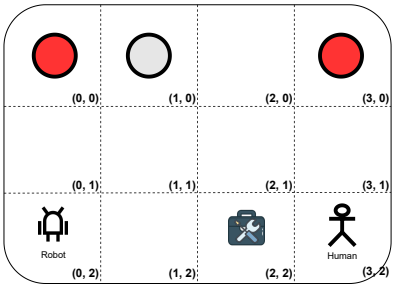


FIGURE 1 – Une tâche de collaboration : Un robot et un humain évoluent dans un monde grille de 4×3. Les cellules en haut à gauche $(0, 0)$ et en haut à droite $(3, 0)$ contiennent toutes deux des appareils en panne. Un appareil à maintenir est aussi situé en cellule $(1, 0)$. Une boîte à outil est située en cellule $(2, 2)$ où l'humain peut prendre des composants.

tâche si $N_{\max}$ est assez grand pour la température T courante. Par exemple, quand $T = 0,3$ et $N_{\max} = 100$, les FSC stochastiques de l'humain atteignent des profondeurs [3] de 22 et 18 (pour chaque objectif), lesquelles sont suffisantes pour que l'humain accomplisse la tâche si le robot collabore. Par contre, quand $N_{\max} = 50$, la profondeur atteinte par le FSC de l'objectif *répare-l'appareil-de-gauche-d'abord* est seulement de 13 (15 pour le 2nd objectif), ce qui ne permet pas à l'humain d'accomplir la tâche (une profondeur de 15 est requise). Nous avons aussi observé que, plus la température T est élevée, plus la valeur de $N_{\max}$ nécessaire pour qu'un FSC stochastique permettant à l'humain d'accomplir la tâche est élevée à son tour. Quand T croît, plus d'actions sont considérées dans chaque nœud, générant plus de branches et empêchant le processus d'expansion d'atteindre la fin de la tâche. Notons aussi que, même si nous fixons T à une valeur très basse (proche de 0), la distribution softmax peut contenir plusieurs actions. Cela permet d'encoder plusieurs politiques humaines optimales dans un unique FSC.

Quand $N_{\max}$ est assez grand pour la température T courante, nous observons que les politiques robustes obtenues pour le robot peuvent résoudre la tâche avec succès, tenant compte des incertitudes sur les objectifs et les comportements de l'humain. Il aide l'humain à réparer tous les appareils et effectue les opérations de maintenance requises. Par exemple, quand $T = 0,3$ et $N_{\max} = 100$, le robot, suivant la politique robuste obtenue, va d'abord à la cellule $(0, 0)$ et attend que l'humain prenne un composant à la cellule $(2, 2)$. Après, si le robot observe que l'humain s'approche, il continue de l'attendre et l'aide ensuite à réparer l'appareil de gauche quand l'humain atteint la cellule $(0, 0)$. Mais s'il observe que l'humain va vers le coin en haut à droite, alors le robot décide d'aller aussi vers la droite pour aider l'humain à réparer l'appareil de droite en premier. Le robot se déplace ensuite jusqu'à la cellule $(1, 0)$ pour effectuer l'opération de maintenance pendant que l'humain va prendre un second composant. Enfin, le robot aide l'humain à réparer l'autre appareil en panne pour finir la tâche.

Même si cela semble être un motif simple pour le robot, il requiert néanmoins que le robot raisonne sur de nombreuses trajectoires possibles de l'humain. S'il y a par exemple 2 actions optimales dans chaque état, même pour 5 pas de temps, il y a 2^5 trajectoires optimales (et ce nombre peut augmenter de manière spectaculaire si l'on considère des actions sous-optimales). De plus, dans ce scénario de collaboration complexe, les agents prennent des décisions en se reposant uniquement sur des observations partielles, et, comme illustré ci-dessus, le robot infère l'objectif de l'humain malgré son comportement incertain.

6.3 Analyse de la robustesse

Nous voulions aussi analyser quantitativement la robustesse de notre approche en comparant la méthode décrite en section 5 avec une méthode calculant la politique d'un robot reposant seulement sur des politiques humaines déterministes. Dans ce but, nous extrayons des paires de politiques humaines déterministes (une par objectif) ; puis, pour chaque paire, nous calculons la politique meilleure réponse du robot à une distribution uniforme sur ces deux politiques humaines. Pour extraire des politiques humaines déterministes, au lieu de garder la distribution softmax complète dans l'algorithme 1, nous n'échantillonnons qu'une action de cette distribution.

Nous avons collecté deux ensembles de paires de politiques humaines déterministes avec des températures de softmax T_{sample} différentes ($0,3$ et $0,5$), chaque ensemble contenant $Nb_{sample} = 50$ paires ($\Pi_H = \{\langle \pi^1_{H,l}, \pi^1_{H,r} \rangle, \ldots, \langle \pi^{50}_{H,l}, \pi^{50}_{H,r} \rangle\}$). Pour chaque paire de politiques humaines $\langle \pi^i_{H,l}, \pi^i_{H,r} \rangle$, nous notons π^i_R la meilleure réponse correspondante du robot à leur mixture (notée $\pi^i_{H,r+l}$). Nous estimons alors :

- la valeur moyenne (sur tous les $i \in [1, 50]$) de la politique du robot π^i_R faisant face à sa paire de politiques humaines correspondante, $V(\langle \pi^i_{H,l}, \pi^i_{H,r} \rangle, \pi^i_R)$;
- la valeur moyenne (sur tous les $i, j \in [1, 50]$) d'une politique de robot faisant face à une paire de politiques humaines, $V(\langle \pi^i_{H,l}, \pi^i_{H,r} \rangle, \pi^j_R)$, laquelle peut ne pas être celle utilisée pour générer la politique du robot (à moins qu'une seule politique optimale n'existe).

Ces valeurs sont ensuite comparées avec les valeurs obtenues, dans les mêmes situations, par des politiques robustes du robot π^*_R générées à l'aide du FSC stochastique de l'humain (comme présenté en section 5).

Les résultats obtenus sont présentés en table 1. Dans la première partie de la table ($T_{sample} = 0,3$, $Nb_{sample} = 50$), nous pouvons observer que les politiques meilleures réponses π^i_R ont les meilleures valeurs moyennes en cas de confrontation avec les paires de politiques humaines déterministes à partir desquelles elles ont été calculées ($19,20$ et $15,90$ pour chaque objectif ; $17,55$ pour l'objectif incertain). C'est attendu puisque les politiques du robot sont faites pour être optimales dans ce cas. Toutefois, quand

3. La profondeur d'un FSC est la distance maximum entre le nœud initial (ici unique) et un nœud quelconque.

ces politiques π^i_R font face à d'autres paires de politiques humaines, le robot échoue la plupart du temps à aider l'humain (et les scores chutent respectivement à $-23,96$, $-22,57$ et $-23,72$), n'ayant pas de comportement approprié quand le comportement de l'humain dévie des attentes du robot. D'un autre côté, même si la politique π^*_R du robot a une valeur plus basse que dans le cas d'une meilleure réponse (π^i_R contre $\langle \pi^i_{H,l}, \pi^i_{H,r} \rangle$), elle est plus robuste aux incertitudes sur les politiques et objectifs de l'humain. Cette robustesse s'améliore même quand $N_{\max}$ croît.

Dans la seconde partie de la table 1 ($T_{sample} = 0,5$, $Nb_{sample} = 50$), les politiques π^i_R manquent encore de robustesse face aux paires de politiques humaines $j \neq i$. Aussi, quand T croît, le comportement de l'humain est plus erratique, et il devient plus difficile pour la politique robuste du robot π^*_R de collaborer avec l'humain, en particulier pour l'objectif 1. Même pour $N_{max} = 600$, la valeur moyenne contre $\pi^j_{H,l}$ est toujours bien plus basse que celle avec l'autre objectif de l'humain ($0,74$ contre $13,05$), alors que π^*_R était meilleure pour le même objectif dans la première partie de la table ($T = 0,3$). Nous supposons que les raisons sont les suivantes : D'abord, quand l'humain est plus erratique (T plus élevée) et suit l'objectif 1 (*réparer-l'appareil-de-gauche-d'abord*), il a plus de chances d'accumuler des pénalités, puisque la trajectoire pour prendre le second composant après la première réparation est plus longue et sujette à plus d'incertitudes. Ensuite, quand l'incertitude sur les comportements humains est élevée et que les deux objectifs sont également probables, le robot peut préférer se concentrer sur l'hypothèse la plus profitable, et décider d'accomplir en priorité l'objectif 2, pour lequel l'humain a moins de chances d'effectuer des actions erratiques.

La table 2 présente les temps de calcul enregistrés à chaque étape du processus. La première étape consiste à convertir chaque modèle de tâche Dec-POMDP en un MPOMDP (un par objectif de l'humain) ; la 2e étape à générer des politiques humaines stochastiques comme présenté en section 5 (processus le plus chronophage) ; la 3e étape en la construction du POMDP pour le robot en utilisant les modèles de tâche (Dec-POMDP) et les politiques stochastiques de l'humain (FSC) ; et l'étape finale en la résolution du POMDP pour le robot à l'aide de SARSOP pour obtenir une politique robuste.

TABLE 1 – Valeurs de diverses politiques du robot face à divers comportements de l'humain. $\pi^j_{H,x}$ représente un comportement humain échantillonné (parmi les paires précalculées) avec priorité à droite ($x = r$) ou à gauche ($x = l$), ou une mixture de ces comportements si $x = r + l$. j est remplacé par i quand le comportement du robot a été conçu précisément pour la même paire échantillonnée.

	$\pi^i_{H,l}$	$\pi^i_{H,r}$	$\pi^i_{H,l+r}$	$\pi^j_{H,l}$	$\pi^j_{H,r}$	$\pi^j_{H,l+r}$
($T_{sample} = 0.3$, $Nb_{sample} = 50$)						
π^i_R	19.20	15.90	17.55	-23.96	-22.57	-23.72
$\pi^*_R(N_{\max}, T)$						
$\pi^*_R(50, 0.3)$				-25.20	5.76	-9.77
$\pi^*_R(100, 0.3)$				14.07	11.60	12.09
$\pi^*_R(150, 0.3)$				17.17	11.60	13.81
($T_{sample} = 0.5$, $Nb_{sample} = 50$)						
π^i_R	12.51	16.47	14.49	-25.48	-25.23	-25.95
$\pi^*_R(N_{\max}, T)$						
$\pi^*_R(150, 0.5)$				-25.77	14.57	-8.34
$\pi^*_R(200, 0.5)$				-23.32	12.97	-11.30
$\pi^*_R(600, 0.5)$				0.74	13.05	6.29

TABLE 2 – Temps CPU (en secondes) pour différentes expérimentations

	$\pi^*_R(N_{\max}, T)$					
Step	$(50, 0.3)$	$(100, 0.3)$	$(150, 0.3)$	$(150, 0.5)$	$(200, 0.5)$	$(600, 0.5)$
Dec-POMDP→MPOMDP	13	13	13	13	13	12
Get Human Stoc. FSCs	130	246	435	487	581	2389
Build Robot POMDP	5	11	17	17	23	203
Solve Robot POMDP	3	6	14	11	16	102
Total time	151	276	479	528	633	2706

7 CONCLUSION

Dans cet article, nous traitons le problème de l'incertitude sur les objectifs de l'humain dans la collaboration homme-robot. La contribution est double : 1. Premièrement, nous proposons un algorithme de planification robuste pour le robot qui repose sur un POMDP avec des incertitudes sur l'objectif réel de l'humain et sur son comportement. Nous détaillons formellement comment concevoir ce POMDP du robot en partant de modèles de tâche (Dec-POMDP) et d'une distribution sur les politiques humaines stochastiques (FSC), une par objectif possible. 2. Deuxièmement, nous discutons du paradoxe de l'œuf et de la poule dans les modèles mentaux du second ordre, et proposons une méthode pour surmonter cet obstacle et générer automatiquement un FSC humain qui contient des comportements humains possibles pour chaque objectif. Des paramètres peuvent être réglés pour ajuster la diversité des comportements humains générés. Notons que l'algorithme de planification robuste pour le robot (1) ne dépend pas de la façon de dériver les politiques de l'humain (2). À travers des expérimentations, nous montrons que notre approche peut s'adapter à des comportements humains incertains avec différents objectifs. Nous pensons que ce travail est important pour les situations de collaboration dans lesquelles le robot et l'humain ont besoin de raisonner sur les actions possibles de l'autre, et où considérer des politiques de l'humain myopes ou déterministes n'est pas suffisant pour générer des politiques de robot robustes. En outre, notre approche ne requiert qu'un Dec-POMDP décrivant la tâche de collaboration humain-robot, et pas de comportement humain donné a priori. Cela rend notre méthode générique pour aborder divers problèmes de collaboration homme-robot, la principale difficulté étant le passage à l'échelle face à de grands problèmes. Dans le futur, nous prévoyons d'étudier comment cette politique robuste du robot se comporte face à de vrais humains dans des environnements simulés, et comment tirer parti de ces simulations pour améliorer le comportement du robot, maintenant que le paradoxe de l'œuf et de la poule a été surmonté. Nous prévoyons aussi d'implémenter notre approche sur une situation réelle où un drone

doit aider un humain à réparer des appareils (dans le cadre du projet Flying Co-Worker).

Références

[1] A. M. ZANCHETTIN, N. M. CERIANI, P. ROCCO, H. DING et B. MATTHIAS, "Safety in human-robot collaborative manufacturing environments : Metrics and control," *IEEE Trans. on Automation Science and Engineering*, t. 13, n° 2, 2016.

[2] K. R. GUERIN, C. LEA, C. PAXTON et G. D. HAGER, "A framework for end-user instruction of a robot assistant for manufacturing," in *ICRA*, 2015.

[3] X. V. WANG, Z. KEMÉNY, J. VÁNCZA et L. WANG, "Human–robot collaborative assembly in cyber-physical production : Classification framework and implementation," *CIRP Annals*, t. 66, n° 1, p. 5-8, 2017.

[4] A. M. ZANCHETTIN et P. ROCCO, "Path-consistent safety in mixed human-robot collaborative manufacturing environments," in *IROS*, 2013.

[5] M. G. JACOB, Y.-T. LI, G. A. AKINGBA et J. P. WACHS, "Collaboration with a Robotic Scrub Nurse," *Com. ACM*, t. 56, n° 5, p. 68-75, mai 2013.

[6] D. HADFIELD-MENELL, S. J. RUSSELL, P. ABBEEL et A. DRAGAN, "Cooperative Inverse Reinforcement Learning," in *NIPS*, 2016.

[7] D. S. BERNSTEIN, S. ZILBERSTEIN et N. IMMERMAN, "The complexity of decentralized control of Markov decision processes," *Mathematics of Operations Research*, 2002.

[8] A. TABREZ, M. LUEBBERS et B. HAYES, "A Survey of Mental Modeling Techniques in Human–Robot Teaming," *Current Robotics Reports*, t. 1, déc. 2020.

[9] V. V. UNHELKAR, S. LI et J. A. SHAH, "Decision-Making for Bidirectional Communication in Sequential Human-Robot Collaborative Tasks," in *HRI*, 2020.

[10] S. NIKOLAIDIS, Y. X. ZHU, D. HSU et S. SRINIVASA, "Human-Robot Mutual Adaptation in Shared Autonomy," in *HRI*, 2017.

[11] M. CHEN, S. NIKOLAIDIS, H. SOH, D. HSU et S. SRINIVASA, "Planning with Trust for Human-Robot Collaboration," in *HRI*, 2018.

[12] ——, "Trust-Aware Decision Making for Human-Robot Collaboration : Model Learning and Planning," *J. Hum.-Robot Interact.*, t. 9, n° 2, jan. 2020. DOI : 10.1145/3359616.

[13] P. DOSHI et P. GMYTRASIEWICZ, "A Framework for Sequential Planning in Multi-Agent Settings," *JAIR*, t. 24, juil. 2005.

[14] W. ZHENG, B. WU et H. LIN, "POMDP Model Learning for Human Robot Collaboration," in *CDC*, 2018.

[15] V. V. UNHELKAR et J. A. SHAH, "Learning Models of Sequential Decision-Making with Partial Specification of Agent Behavior," in *AAAI*, 2019.

[16] S. RUSSELL, "Learning Agents for Uncertain Environments (Extended Abstract)," in *COLT*, 1998.

[17] A. Y. NG et S. RUSSELL, "Algorithms for Inverse Reinforcement Learning," in *ICML*, 2000.

[18] N. MEULEAU, K.-E. KIM, L. KAELBLING et A. CASSANDRA, "Solving POMDPs by searching the space of finite policies," in *UAI*, 1999.

[19] D. V. PYNADATH et M. TAMBE, "The Communicative Multiagent Team Decision Problem : Analyzing Teamwork Theories and Models," *JAIR*, t. 16, juin 2002.

[20] M. GRZEŚ, P. POUPART, X. YANG et J. HOEY, "Energy Efficient Execution of POMDP Policies," *IEEE Trans. on Cybernetics*, t. 45, 2015. DOI : 10.1109/TCYB.2014.2375817.

[21] E. A. HANSEN et S. ZILBERSTEIN, "LAO* : A heuristic search algorithm that finds solutions with loops," *Artificial Intelligence*, t. 129, n° 1, p. 35-62, 2001.

[22] D. SILVER et J. VENESS, "Monte-Carlo Planning in Large POMDPs," in *NIPS*, 2010.

[23] H. KURNIAWATI, D. HSU et W. S. LEE, "SARSOP : Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *RSS*, 2008.

An Implemented System for Cognitive Planning

Jorge Luis Fernandez Davila[1] , **Dominique Longin**[2] , **Emiliano Lorini**[2], and **Frédéric Maris**[1]
[1]IRIT, Toulouse University, France
[2]IRIT, CNRS, Toulouse University, France
{Jorge.Fernandez, Dominique.Longin, Emiliano.Lorini, Frederic.Maris}@irit.fr

Abstract

We present a system that implements a framework for cognitive planning[1]. The system allows us to represent and reason about the beliefs, desires and intentions of other agents using an NP-fragment of a multiagent epistemic logic. The system has three components: the belief revision, the planner and the translator modules. They work in an integrated way to firstly capture new information about the world, secondly to perform the planning process extracting operators from a set of actions that will lead to achieving a goal and, finally, to verify satisfiability of the formulas generated at each step of the process. We illustrate how our system can be used to implement an persuasive artificial agent interacting with a human user.

1 Introduction

Automated planning is at the center of AI research with a variety of applications ranging from control traffic and robotics to logistics and services. Epistemic planning extends automated planning incorporating notions of knowledge and beliefs [Bolander and Andersen, 2011; Löwe *et al.*, 2011; Kominis and Geffner, 2015; Muise *et al.*, 2015; Cooper *et al.*, 2020]. Cognitive planning is a generalization of epistemic planning, where the goal to be achieved is not only a belief state but a cognitive state of a target including not only beliefs but also intentions. Moreover, we are particularly interested in persuasive goals of the planning agent, aimed at influencing another agent's beliefs and intentions.

The increasing number of applications in social robotics, social networks, virtual assistants together with sentiment analysis techniques allow us to collect data related to humans' beliefs and intentions. In [Akimoto, 2019] a framework for modeling mental attitudes of an agent, based on her narratives, is proposed. In addition, cognitive models can be used to predict agents' decision-making by taking psychological factors like motivation and emotions into account [Prezenski *et al.*, 2017]. Nonetheless, few approaches exist which leverage this information about humans' cognitive states for changing their attitudes and behaviors through persuasion.

[1]https://www.irit.fr/CoPains/software/

Our work aims to fill this gap by introducing a system based on a simple framework detailed in [Fernandez *et al.*, 2021] in which we can represent an agent's cognitive state in a compact way, reason about it and planning a sequence of speech acts aimed at changing it. Our approach is based on an epistemic logic introduced in [Lorini, 2018; Lorini, 2020], which allows us to represent an agent's explicit beliefs, as the information in the agent's belief base, and the agent's implicit beliefs, as the information which is deducible from the agent's belief base. Given that the satisfiability problem for the full logic is PSPACE-hard, we focus on an NP-fragment that makes the logic suitable for implementing real-world applications.

The core components of the system are the belief revision, the planner and the translator modules. The formulas representing the rules and constraints of a specific problem are loaded into the system. We encode these rules using the NP-fragment presented in [Fernandez *et al.*, 2021]. The system takes this information as the initial state and some actions — which are of type speech act — to build a plan that leads to the goal. An important feature is that actions have preconditions that impose constraints on their execution order. We illustrate the implementation of our system in a human-machine interaction (HMI) scenario in which an artificial agent has to persuade a human agent to practice a sport based on her preferences.

2 A Language for Explicit and Implicit Belief

This section describes the basics of the Logic of Doxastic Attitudes (LDA) introduced in [Lorini, 2018; Lorini, 2020]. It is a multiagent epistemic logic which supports reasoning about explicit and implicit beliefs. Assume a countably infinite set of atomic propositions Atm and a finite set of agents $Agt = \{1, \ldots, n\}$. We define the language in two steps.

We first define the language $\mathcal{L}_0(Atm, Agt)$ by the following grammar in Backus-Naur Form (BNF):

$$\alpha \quad ::= \quad p \mid \neg\alpha \mid \alpha_1 \wedge \alpha_2 \mid \alpha_1 \vee \alpha_2 \mid \triangle_i\alpha,$$

where p ranges over Atm and i ranges over Agt. $\mathcal{L}_0(Atm, Agt)$ is the language for representing agents' explicit beliefs. The formula $\triangle_i\alpha$ is read "i explicitly believes that α". The language $\mathcal{L}(Atm, Agt)$ extends the language $\mathcal{L}_0(Atm, Agt)$ by modal operators of implicit belief and is defined by the following grammar:

$$\mathcal{L}^{+}_{\mathsf{Frag}} \overset{red}{\dashrightarrow} \mathcal{L}_{\mathsf{Frag}} \overset{nnf}{\dashrightarrow} \mathcal{L}^{\mathsf{NNF}}_{\mathsf{Frag}} \overset{tr_1}{\dashrightarrow} \mathcal{L}_{\mathsf{Mod}} \overset{tr_2}{\dashrightarrow} \mathcal{L}_{\mathsf{Prop}}$$

Figure 1: Summary of reduction process

$$\varphi \quad ::= \quad \alpha \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \Box_i\varphi \mid \Diamond_i\varphi,$$

where α ranges over $\mathcal{L}_0(Atm, Agt)$ and i ranges over Agt. For notational convenience we write $\mathcal{L}_0$ instead of $\mathcal{L}_0(Atm, Agt)$ and $\mathcal{L}$ instead of $\mathcal{L}(Atm, Agt)$, when the context is unambiguous. The formula $\Box_i\varphi$ is read "i implicitly believes that φ" and $\Diamond_i\varphi$ is read "φ is compatible (or consistent) with i's explicit beliefs". The other Boolean constructions $\top$, $\bot$, $\rightarrow$ and $\leftrightarrow$ are defined in the standard way.

The language is interpreted with respect to a formal semantics using belief bases whose details are given in [Lorini, 2018; Lorini, 2020]. Checking satisfiability of $\mathcal{L}$ formulas relative to this semantics is a PSPACE-hard problem. For that reason, in [Fernandez *et al.*, 2021], we looked for an interesting NP-fragment of $\mathcal{L}$ that we called $\mathcal{L}_{\mathsf{Frag}}$:

$$\varphi \quad ::= \quad \alpha \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \Box_{\mathfrak{m}}\alpha \mid \Diamond_{\mathfrak{m}}\alpha,$$

where α ranges over $\mathcal{L}_0$ and $\mathfrak{m}$ is a special agent in Agt called the 'machine'. In $\mathcal{L}_{\mathsf{Frag}}$, all agents have explicit beliefs but only agent $\mathfrak{m}$ has implicit beliefs, and moreover the latter are restricted to $\mathcal{L}_0$ formulas of type α. Agent $\mathfrak{m}$ is the artificial planning agent. In order to represent agents' belief dynamics, language $\mathcal{L}_{\mathsf{Frag}}$ is extended by belief expansion operators. Such an extension will allow us to represent the actions of the planning agent in the cognitive planning problem. Specifically, we introduce the following language $\mathcal{L}^{+}_{\mathsf{Frag}}$:

$$\varphi ::= \alpha \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \Box_{\mathfrak{m}}\alpha \mid \Diamond_{\mathfrak{m}}\alpha \mid [+_i\alpha]\varphi,$$

where α ranges over $\mathcal{L}_0$ and i ranges over Agt. The formula $[+_i\alpha]\varphi$ is read "φ holds after agent i has privately expanded her belief base with α". Event of type $+_i\alpha$ are generically called informative actions.

3 Cognitive Planning

The planning problem in the context of the logic $\mathcal{L}^{+}_{\mathsf{Frag}}$ is to find a sequence of informative actions for agent $\mathfrak{m}$ of type $+_{\mathfrak{m}}\alpha$ which guarantees that agent $\mathfrak{m}$ will knowingly achieve its goal. Let $Act_{\mathfrak{m}} = \{+_{\mathfrak{m}}\alpha : \alpha \in \mathcal{L}_0\}$ be agent $\mathfrak{m}$'s set of informative actions and let the elements of $Act_{\mathfrak{m}}$ be noted $\epsilon, \epsilon', \ldots$ Agent $\mathfrak{m}$'s informative actions have executability preconditions that are specified by the following function: $\mathcal{P} : Act_{\mathfrak{m}} \longrightarrow \mathcal{L}_{\mathsf{Frag}}$. So, we can define the following operator of successful occurrence of an informative action:

$$\langle\langle\epsilon\rangle\rangle\varphi \overset{\text{def}}{=} \mathcal{P}(\epsilon) \wedge [\epsilon]\varphi$$

with $\epsilon \in Act_{\mathfrak{m}}$. The formula $\langle\langle\epsilon\rangle\rangle\varphi$ has to be read "agent $\mathfrak{m}$'s informative action ϵ can take place and φ holds after its occurrence".

Informative actions of type 'speech act' are of interest here. In particular, we consider speech acts of type 'to inform', where $\mathfrak{m}$ is assumed to be the speaker and $j \in Agt$ such that $j \neq \mathfrak{m}$ is assumed to be the hearer. We define the speech act "agent $\mathfrak{m}$ informs agent j that α" as follows:

$$inform(\mathfrak{m},j,\alpha) \overset{\text{def}}{=} +_{\mathfrak{m}} \triangle_j\alpha.$$

In [Fernandez *et al.*, 2021] the planning problem is defined as a tuple $\langle\Sigma, Op, \alpha_G\rangle$ where:

- $\Sigma \subset \mathcal{L}_0$ is a finite set of agent $\mathfrak{m}$'s available information,
- $Op \subset Act_{\mathfrak{m}}$ is a finite set of agent $\mathfrak{m}$'s operators,
- $\alpha_G \in \mathcal{L}_0$ is agent $\mathfrak{m}$'s goal.

The planning problem has a solution if the formula $\neg\big((\bigwedge_{\alpha\in\Sigma}\Box_{\mathfrak{m}}\alpha) \rightarrow \langle\langle\epsilon_1\rangle\rangle \ldots \langle\langle\epsilon_k\rangle\rangle\Box_{\mathfrak{m}}\alpha_G\big)$ is unsatisfiable. The translator module converts the $\mathcal{L}^{+}_{\mathsf{Frag}}$-formula into its equivalent formula in propositional logic following the sequence of reductions detailed in Figure 1. In a second step, the translator interfaces with the SAT encoding tool TouIST [Fernandez *et al.*, 2020] to verify the validity of the propositional formula. TouIST will encode the formula in its DIMACS CNF format and send it to the MiniSAT solver for checking satisfiability.

4 Example and Implementation: Artificial Assistant

In order to probe the potential of our implemented system for cognitive planning we test it in a HMI scenario detailed in [Fernandez *et al.*, 2021]. In this scenario agent $\mathfrak{m}$ is the artificial assistant of the human agent $\mathfrak{h}$. Agent $\mathfrak{h}$ has to choose a sport to practice since her doctor recommended her to do a regular physical activity to be in good health. Agent $\mathfrak{m}$'s aim is to help agent $\mathfrak{h}$ to make the right choice, given her actual beliefs and desires. Agent $\mathfrak{m}$ has to provide information about the possible options (Opt) the user can choose and their properties (Var). For each pair (Opt, Var) we assing a valuation Val. In this example, we suppose that Opt is composed of the following eight elements: swimming (sw), running (ru), horse riding (hr), tennis (te), soccer (so), yoga (yo), diving (di) and squash (sq). Moreover, there are exactly six variables in Var which are used to classify the available options: environment (**env**), location (**loc**), sociality (**soc**), cost (**cost**), dangerousness (**dan**) and intensity (**intens**). The variable assignments are showed in Table 1:

Opt	Var					
	env	**loc**	**soc**	**cost**	**dan**	**intens**
sw	*water*	*mixed*	*single*	*med*	*low*	*high*
ru	*land*	*outdoor*	*single*	*low*	*med*	*high*
hr	*land*	*outdoor*	*single*	*high*	*high*	*low*
te	*land*	*mixed*	*mixed*	*high*	*med*	*med*
so	*land*	*mixed*	*team*	*med*	*med*	*med*
yo	*land*	*mixed*	*single*	*med*	*low*	*low*
di	*water*	*mixed*	*single*	*high*	*high*	*low*
sq	*land*	*indoor*	*mixed*	*high*	*med*	*med*

Table 1: Variable assignments. For every option $o \in Opt$ and variable $x \in Var$, we denote by $v_{o,x}$ the corresponding entry in the table. For instance, we have $v_{\mathrm{sw},\mathbf{env}} = water$.

Formulas representing the rules and constraints are loaded as part of agent $\mathfrak{m}$'s belief base. For example, the implementation of the formula representing the fact that agent $\mathfrak{h}$ explicitly believes that a sport cannot have two different values for

a given property is formalized as follows:

$$\bigwedge_{\substack{o\in Opt\\ x\in Var\\ v_1,v_2\in Val_x: v_1\neq v_2}} \Big(\triangle_{\mathfrak{h}}\mathsf{val}(o,x\mapsto v_1)\rightarrow \triangle_{\mathfrak{h}}\neg\mathsf{val}(o,x\mapsto v_2)\Big)$$

```
bigand
  $o,$x,$v1,$v2 in $Opt,$Var,$Val($x),$Val($x)
  when $v1 != $v2:
  {h}val($o,ass($x,$v1))=>{h}not val($o,ass($x,$v2))
end
```

The set of actions are generated from table 1:

```
inform(m,h,val_so_ass_env_land)
```

The previous informative action is interpreted as the speech act used by agent $\mathfrak{m}$ to inform agent $\mathfrak{h}$ that the valuation of the `property`:environment for the `option`:soccer is land. In order to help agent $\mathfrak{h}$ to select an activity, agent $\mathfrak{m}$ also needs information about $\mathfrak{h}$'s set of actual desires. Information about agent $\mathfrak{h}$'s desires is gathered by agent $\mathfrak{m}$ during its interaction with $\mathfrak{h}$. The interaction interface between agents $\mathfrak{h}$ and $\mathfrak{m}$ is showed in figure 2. The belief revision module is called after each agent $\mathfrak{h}$'s feedback and it restores consistency of the agent $\mathfrak{m}$'s belief base, in case the incoming information is inconsistent with agent $\mathfrak{m}$'s pre-existent beliefs.

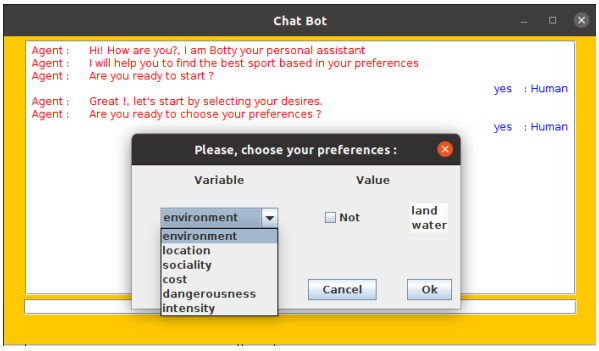


Figure 2: Collecting agent $\mathfrak{h}$'s preferences

In the example, agent $\mathfrak{h}$ would like to practice a land activity, with medium intensity, which is not exclusively indoor, and which can be practiced both in single and team mode, if its cost is high. The next rule for precondition states that agent $\mathfrak{h}$ must be informed by agent $\mathfrak{m}$ about the dangerousness level of a sport, before presenting other properties for an option. For $a \notin Assign_{\mathbf{dan}}$:

$$\mathcal{P}\Big(inform\big(\mathfrak{m},\mathfrak{h},\mathsf{val}(o,a)\big)\Big) = \Box_{\mathfrak{m}}\Big(\mathsf{val}(o,a)\wedge \bigwedge_{v\in Val_{\mathbf{dan}}} \big(\mathsf{val}(o,\mathbf{dan}\mapsto v)\rightarrow \triangle_{\mathfrak{h}}\mathsf{val}(o,\mathbf{dan}\mapsto v)\big)\Big)$$

In the next lines we illustrate how the precondition is assigned by the planner module together with its $+_{\mathfrak{m}}\alpha$ operator in order to specify the successful occurrence of an informative action:

```
[m]((val_te_ass_intens_med) and
    (val_te_ass_danger_med =>
                    {h}val_te_ass_danger_med)) and
    plus({h}val_te_ass_intens_med, ...
```

The planner reads the Initial State, the Actions and the Goal files. The planning module generates plans with the elements contained in the Action file. It starts with plans of length 1, and enters in a loop. At each interaction the planner asks the SAT solver to verify whether the plan allows to achieve the Goal. If no plan of length k is found, the program will increase the counter in one and look for a plan of length $k+1$.

```
plus({h}(val_te_ass_danger_med)
plus({h}(val_te_ass_intens_med)
plus({h}(val_te_ass_soc_mixed)
plus({h}(val_te_ass_loc_mixed)
plus({h}(val_te_ass_env_land)
plus({h}(ideal_h_te)
```

The order of speech acts is determined by the preconditions. Specifically, the planner informs firstly about the dangerousness level of the sport. Secondly, it provides explanation of why the user's desires are satisfied. Finally, it indicates the ideal sport for the user, in this case tennis.

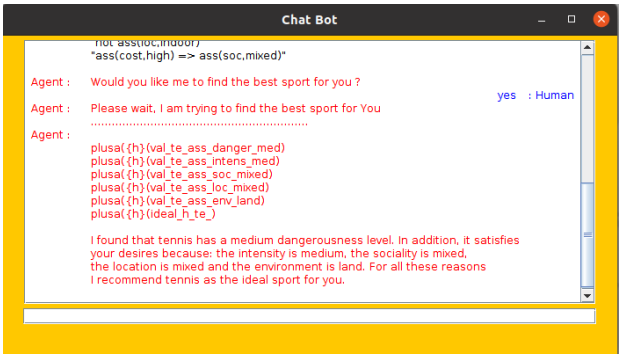


Figure 3: Plan showed by the chatbot to the human

The chatbot writes both the formal actions sequence and its translation in natural language.

5 Conclusion

The implementation demonstrates that the NP-complete epistemic logic presented in [Fernandez *et al.*, 2021] and the cognitive planning problem formulated in this logic are suitable for real-world applications in the domain of human-machine interaction. In future work, we plan to extend the implemented system by speech acts of type question to capture both sides of interaction, from agent $\mathfrak{m}$ to agent $\mathfrak{h}$ (handled by the actual implementation) and from agent $\mathfrak{h}$ to agent $\mathfrak{m}$. We also plan to combine our implementation of cognitive planning with machine learning and data mining techniques, as presented in [Krzywicki *et al.*, 2016], which are aimed at extracting information about the human user from real data.

References

[Akimoto, 2019] Taisuke Akimoto. Narrative structure in the mind: Translating genette's narrative discourse theory into a cognitive system. *Cognitive Systems Research*, 58:342–350, 2019.

[Bolander and Andersen, 2011] T. Bolander and M. B. Andersen. Epistemic planning for single- and multi-agent systems. *Journal of Applied Non-Classical Logics*, 21(1):9–34, 2011.

[Cooper *et al.*, 2020] Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris, Elise Perrotin, and Pierre Régnier. A lightweight epistemic logic and its application to planning. *Artificial Intelligence*, page 103437, 2020.

[Fernandez *et al.*, 2020] Jorge Fernandez, Olivier Gasquet, Andreas Herzig, Dominique Longin, Emiliano Lorini, Frédéric Maris, and Pierre Régnier. Touist: a friendly language for propositional logic and more. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 5240–5242. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Demos.

[Fernandez *et al.*, 2021] Jorge Fernandez, Dominique Longin, Emiliano Lorini, and Frédéric Maris. A simple framework for cognitive planning. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*. AAAI Press, 2021. To appear.

[Kominis and Geffner, 2015] F. Kominis and H. Geffner. Beliefs in multiagent planning: from one agent to many. In Ronen I. Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, editors, *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS 2015)*, pages 147–155. AAAI Press, 2015.

[Krzywicki *et al.*, 2016] Alfred Krzywicki, Wayne Wobcke, Michael Bain, John Calvo Martinez, and Paul Compton. Data mining for building knowledge bases: Techniques, architectures and applications. *The Knowledge Engineering Review*, -1:1–27, 03 2016.

[Lorini, 2018] E. Lorini. In praise of belief bases: Doing epistemic logic without possible worlds. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 1915–1922. AAAI Press, 2018.

[Lorini, 2020] E. Lorini. Rethinking epistemic logic with belief bases. *Artificial Intelligence*, 282, 2020.

[Löwe *et al.*, 2011] B. Löwe, E. Pacuit, and A. Witzel. DEL planning and some tractable cases. In *Proceedings of the 3rd International International Workshop on Logic, Rationality and Interaction (LORI 2011)*, pages 179–192. Springer Berlin Heidelberg, 2011.

[Muise *et al.*, 2015] C. Muise, V. Belle, P. Felli, S. A. McIlraith, T. Miller, A. R. Pearce, and L. Sonenberg. Planning over multi-agent epistemic states: A classical planning approach. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015)*, pages 3327–3334. AAAI Press, 2015.

[Prezenski *et al.*, 2017] Sabine Prezenski, André Brechmann, Susann Wolff, and Nele Russwinkel. A cognitive modeling approach to strategy formation in dynamic decision making. *Frontiers in Psychology*, 8:1335, 2017.

Planification automatique de scénarios d'entraînement robustes

R. Goutiere[1], D. Lourdeaux[1], S. Lagrue[1]

[1] Alliance Sorbonne Université,
UTC Heudiasyc, CNRS, UMR 7253
Compiègne, France

{romain.goutiere;domitile.lourdeaux;sylvain.lagrue}@hds.utc.fr

Résumé

Dans cet article nous nous intéressons à la génération de scénario d'entraînement pour la formation à la gestion de crise en environnement virtuel. Celle-ci utilise un système de planification mêlant planification temporelle et réseaux de tâches hiérarchiques reposant sur le langage de planification ANML et permettant une meilleure représentation des objectifs à la fois narratifs et pédagogiques. Nous proposons également plusieurs algorithmes permettant de générer automatiquement des alternatives au scénario initialement planifié, répondant aux mêmes objectifs, afin de le rendre plus robuste.

Mots-clés

ANML, Planification automatique de scénario, Narration interactive, Génération d'alternatives, HTN

Abstract

In this paper, we focus on the generation of scenarios for crisis management training in virtual environment. It uses a planning system mixing temporal planning and hierarchical task networks based on the ANML planning language and allowing a better representation of both narrative and pedagogical objectives. We also propose several algorithms to automatically generate alternatives to the initially planned scenario, meeting the same objectives, to make it more robust.

Keywords

ANML, Automated scenario planning, Interactive narrative, Generation of alternatives, HTN

1 Introduction

L'objectif de la recherche en narration interactive est de réaliser des systèmes multimédias au sein desquels les utilisateurs peuvent interagir et influencer, en temps interactif, l'évolution de la narration. Plusieurs dénominations existent dans la littérature pour parler de narration interactive, les plus répandues sont *Interactive Narrative* (IN) et *Interactive Storytelling (IS)*. Riedl et Bulitko [15] définissent la narration interactive comme « une forme d'expérience interactive numérique dans laquelle les utilisateurs créent ou influencent une intrigue dramatique par des actions, soit en assumant le rôle d'un personnage dans un monde virtuel fictif, en émettant des commandes à des personnages contrôlés par ordinateur, soit en manipulant directement l'état du monde fictif ».

Dans ce contexte la scénarisation occupe une place de premier plan. Barot [3] définit cette scénarisation comme « un processus comprenant à la fois la spécification du ou des déroulements possibles ou souhaitables de la simulation, et le contrôle (exécution et/ou suivi et correction) du déroulement des événements en temps interactif ». Le coeur de la scénarisation est donc le scénario, définit formellement dans ce cadre comme « un ensemble d'événements particuliers, partiellement ordonnés et instanciables dans un environnement virtuel » [3].

L'un des domaines de recherche les plus actifs autour de la narration interactive est la génération automatique de ces scénarios. L'objectif des approches génératives est de diminuer la charge de travail de l'auteur (réduction de l'*authoring bottleneck*) lors de la conception du système de narration interactive. Le scénario est ainsi généré automatiquement en amont et/ou dynamiquement pendant la simulation ouvrant ainsi la voie à des adaptations du scénario en cours de simulation et à une plus grande variété dans le contenu proposé.

La planification est aujourd'hui une méthode très répandue pour la génération automatique de scénarios. Son utilisation dans le cadre de la narration interactive nécessite cependant quelques adaptations par rapport aux cadres d'applications plus classiques et plusieurs challenges subsistent. Porteous [10] cite notamment : (1) la capacité du planificateur à raisonner sur le contenu narratif, (2) le contrôle en temps interactif sur la génération du plan, (3) l'interactivité de l'utilisateur qui nécessite que le système soit résilient et (4) la planification sur des critères qualitatifs autres que l'optimisation pure.

Les systèmes de narration interactive sont également utiles dans la cadre de l'entraînement et la formation. Gupta [7] donne plusieurs avantages comme de pouvoir mettre plus facilement les apprenants face à des situations variées, et à les faire recommencer au besoin. Les systèmes proposant de la formation doivent cependant répondre à certains critères pour garantir l'efficacité de l'expérience. Il est par exemple important que les situations ne mettent pas en danger les utilisateurs, que la difficulté soit bien calculée, que les scénarios soient cohérents et explicables pour le forma-

teur, etc.
Dans cet article nous proposons l'utilisation d'une méthode de planification hybride permettant une meilleure prise en charge du contenu narratif et pédagogique par le planificateur dans le but de générer des scénarios d'entraînement offrant plus d'expressivité, permettant ainsi une meilleure compréhension de ses décisions par les formateurs. Pour cela nous nous appuyons sur l'utilisation du langage de planification ANML, offrant une représentation plus naturelle des contenus narratifs et pédagogiques. Nous proposons également des algorithmes de modification automatique du domaine de planification afin de permettre une génération *offline* d'alternatives pour le scénario planifié. Ces alternatives, dont les objectifs sont les mêmes que ceux du scénario initial, offrent une plus grande robustesse au scénario et permettent de se prémunir des potentielles dérives issues des actions des utilisateurs.

2 État de l'art

2.1 Planification de scénario

La génération de scénario s'appuyant sur la planification est aujourd'hui largement utilisée [10]. La planification est une branche de l'intelligence artificielle qui vise à produire des plans, un plan étant une séquence d'actions qui a pour but d'atteindre un objectif donné [6].
La planification présente plusieurs avantages à savoir, la causalité, la puissance générative et la structure narrative [10]. Dans le contexte de la narration interactive, la causalité se réfère à la relation entre des événements temporels et ordonnés qui permet de prédire l'apparition d'événements futurs [17]. La puissance générative représente la capacité d'un système à générer un contenu varié. Enfin, la structure narrative permet de définir les étapes clés du déroulement d'un récit.
La planification classique est très utilisée en narration interactive. Un plan est alors une succession d'actions et d'événements afin d'atteindre un but souhaité. Il existe alors de nombreuses variations ayant pour but de gérer des cas plus complexes comme l'ajout de contraintes temporelles [11] permettant d'ajouter des durées ou de paralléliser certaines actions. Cependant, une autre méthode est également utilisée en narration interactive [2] : les réseaux de tâches hiérarchiques (HTN). Le plan d'un problème HTN n'est pas représenté sous la forme d'une séquence d'événements et d'actions, mais comme une décomposition successive d'actions complexes, en actions simples. La structure d'un plan se présente alors sous la forme d'un arbre. De nombreux systèmes utilisent les HTN car ils offrent une représentation plus simple et expressive de certains concepts, notamment en ce qui concerne les comportements humains [9].

2.2 Langages de planification

Pour qu'un problème de narration interactive puisse être résolu par une méthode de planification il est nécessaire de pouvoir l'exprimer dans un langage compréhensible par un planificateur. Dans le but de pouvoir comparer les performances des différents planificateurs, des projets de standardisation de ces langages sont apparus. Le *Stanford Research Institute problem solver* (STRIPS) [5] est un planificateur qui à permis de grandes avancées sur le sujet car son langage d'entrée a posé les fondations d'une standardisation et a directement inspiré le *Planning Domain Description Language* (PDDL) [1] actuellement utilisé lors des compétitions internationales de planification (IPC). PDDL a évolué au fil des années pour intégrer diverses évolutions telles que la prise en charge des aspects temporels [8]. Les évolutions ajoutent des possibilités au langage et permettent ainsi de fonctionner avec des planificateurs plus complexes.
Les différentes versions de PDDL permettent aujourd'hui de représenter assez simplement la plupart des problèmes de planification classiques et la quasi-totalité des planificateurs est conçue pour pouvoir résoudre des problèmes définis en PDDL. Cependant, les domaines/problèmes étudiés en narration interactive sont souvent plus complexes que ceux étudiés en planification classique car ils décrivent des concepts de nature parfois très variables ce qui se traduit généralement par un plus grand nombre d'actions, des difficultés à représenter l'état du monde sans perdre trop d'informations ou encore l'ajout de beaucoup de contraintes. Cette différence impose certains choix lors de la formalisation du domaine/problème qui peuvent conduire à une baisse globale d'intelligibilité chez l'utilisateur humain et une diminution de l'expressivité du scénario.
Le principal intérêt de PDDL est de proposer un standard compréhensible par le plus grand nombre possible de planificateurs. Toutefois, PDDL présente des limitations dans le cadre de la narration interactive comme l'impossibilité de représenter simplement des problèmes de planification hiérarchique ou des difficultés à représenter des états complexes du monde contenant des concepts de natures très différentes. D'autant plus que les critères de qualité des plans ne sont pas les mêmes en narration interactive et en planification classique, ce qui implique une conception des versions de PDDL tournée vers les performances évaluées, tels que la longueur des plans, lors des compétitions de planification plutôt que vers d'autres critères, plus liés à la narration et souvent propres aux domaines étudiés.
L'*Action Notation Modeling Language* (ANML) [16] est une option intéressante pour apporter une meilleure modélisation aux problèmes de narration interactive. C'est un langage de planification offrant des fonctionnalités de plus haut niveau que PDDL. Il permet notamment l'utilisation de contraintes temporelles assez complexes et permet également de modéliser et résoudre des problèmes de planification représentés à la fois sous forme classique et sous forme hiérarchique. Il est ainsi possible de développer des domaines de planification hybrides où il n'est pas nécessaire d'adapter sa modélisation à l'un ou l'autre des systèmes mais où chaque élément est représenté selon l'approche qui lui correspond le mieux. Ce qui apporte une grande expressivité dans un cadre comme celui de la narration interactive, où des concepts assez différents comme la pédagogie, les émotions ou les actions physiques des acteurs doivent cohabiter. Enfin, il est à noter que l'ANML est dans la plupart des cas convertible en PDDL au prix d'une

perte d'intelligibilité [16].

2.3 Robustesse du scénario

Dans un système de narration interactive, l'exécution du scénario peut se révéler problématique. En effet, en plus des différents événements directement déclenchés par le système, un scénario doit également compter sur certaines actions des différents acteurs (utilisateurs humains, personnages virtuels). Ainsi, un scénario peut potentiellement se retrouver bloqué si une action devant être réalisée par un utilisateur n'arrive jamais.

De nombreuses stratégies ont été développées pour résoudre cette problématique. Elles se réunissent aujourd'hui autour du concept d'*Experience Management* [15] qui consiste à suivre le déroulement du scénario et mettre en place des stratégies d'intervention pour lui permettre de se poursuivre de la meilleure façon possible.

L'une des principales stratégies, la médiation narrative [13], propose, tant que cela est possible, de laisser les acteurs effectuer librement leurs actions. Si celles-ci rentrent en conflit avec le scénario prévu, le scénario est replanifié à partir du point de divergence pour prendre en compte les actions problématiques dans le plan, afin de résoudre les mêmes objectifs que le scénario initial.

Cette solution n'est cependant pas parfaite, car pour pouvoir être utilisable elle doit être appliquée en temps interactif au cours de la simulation. Cela implique souvent l'utilisation d'algorithmes rapides ne garantissant pas forcément la qualité du nouveau plan [12].

Enfin, une autre approche vise à prédire les potentielles erreurs pouvant conduire à des problèmes de scénario et à générer *offline*, des alternatives directement applicables. *Automated Story Director* [14] en est un bon exemple. Dans ce système, un scénario est constitué d'événements de faible importance et d'*Island*, des événements importants pour le scénario. Le principe est, pour chaque action, de prévoir des situations susceptibles de mettre en péril la prochaine *Island* et de générer une alternative permettant de la rejoindre. La génération d'alternatives *offline* présente l'intérêt de ne pas être soumis à la contrainte de temps interactif. En cas de problème ou de situation non prévue, il peut être nécessaire de faire appel à une méthode de réparation classique *online*, le scénario pouvant se retrouver bloqué.

3 Propositions

Notre première proposition vise à améliorer la représentation de la structure narrative de nos scénarios par l'utilisation du langage ANML. Celui-ci permet, au travers de sa représentation hybride et de haut niveau des domaines de planification, de modéliser au mieux des éléments narratifs dont la nature est parfois très différente. Dans notre contexte, qui est celui de l'entraînement, nous nous appuyons sur cette souplesse pour représenter à la fois des éléments narratifs, instanciables dans un environnement virtuel, et du contenu pédagogique, plus abstrait.

Notre seconde proposition a, quant à elle, pour but l'amélioration de la robustesse de nos scénarios, par la génération *offline* de plans alternatifs permettant d'atteindre les mêmes objectifs que notre plan initial. Cette génération ne s'appuie pas directement sur l'utilisation d'un planificateur, mais sur la modification dynamique du domaine/problème de planification. Nous présentons donc deux algorithmes de modification de domaine, dont le but est de forcer le planificateur à choisir de nouveaux plans.

3.1 Planification hybride

La planification HTN permet une représentation plus simple de certains concepts complexes. En effet, la décomposition est une approche qui peut s'avérer utile pour décrire des problèmes dont les actions simples, prises indépendamment, ne permettent pas directement de se représenter le problème global. Un exemple concret est l'action complexe « construire une maison ». On peut facilement décomposer cette tâche en trois autres actions complexes telles que « couler la dalle, construire les murs et enfin poser le toit », tâches que l'on peut à nouveau décomposer jusqu'à obtenir des actions simples (directement réalisables). De ce fait il est possible, pour chaque action simple réalisée, de se rattacher directement au problème global. Cependant, cette représentation n'est pas la plus efficace pour tous les concepts. Ainsi, certains concepts abstraits liés à la progression de l'apprenant ou encore de ces émotions sont difficilement représentables en HTN.

Dans notre contexte de formation, nous avons besoin de représenter des éléments liés à la progression des compétences de l'utilisateur, de sa maîtrise ou non des différentes habiletés sur lesquelles on souhaite le faire progresser. Ce genre de contenu narratif, que l'on peut qualifier de contenu pédagogique est difficilement représentable sous forme de HTN. Cependant, ce type de contenu s'appréhende facilement en planification classique. Il est en effet assez logique de structurer une séance d'entraînement comme une succession d'exercices travaillant chacun un ensemble de compétences différentes. Nous pouvons même y ajouter une dimension temporelle afin d'avoir une séance d'entraînement représentée comme une succession d'opérateurs de planification pédagogiques (que nous appelons ici *unités d'apprentissage*) ayant un horaire de début et une durée. Le but d'une telle séance étant de faire progresser l'utilisateur dans un ensemble prédéfini de compétences.

Nous proposons donc une approche de planification hybride. La figure 1 illustre cette approche. Le contenu pédagogique est représenté comme un problème de planification classique avec contraintes temporelles. Ce niveau de planification ne contient pas d'actions ou d'événements instanciables dans un environnement virtuel mais constitue un suivi au niveau pédagogique de l'entraînement de l'utilisateur. Chaque unité d'apprentissage, modélisée par une action, représente une période pendant laquelle l'utilisateur va être mis en face de situations lui permettant de travailler une ou plusieurs compétences.

Ainsi, chaque ensemble de compétences travaillé peut être vu comme un problème de planification hiérarchique dont les buts sont à la fois narratifs et pédagogiques. En effet, c'est à ce niveau de planification que se situent les actions véritablement réalisables dans l'environnement virtuel.

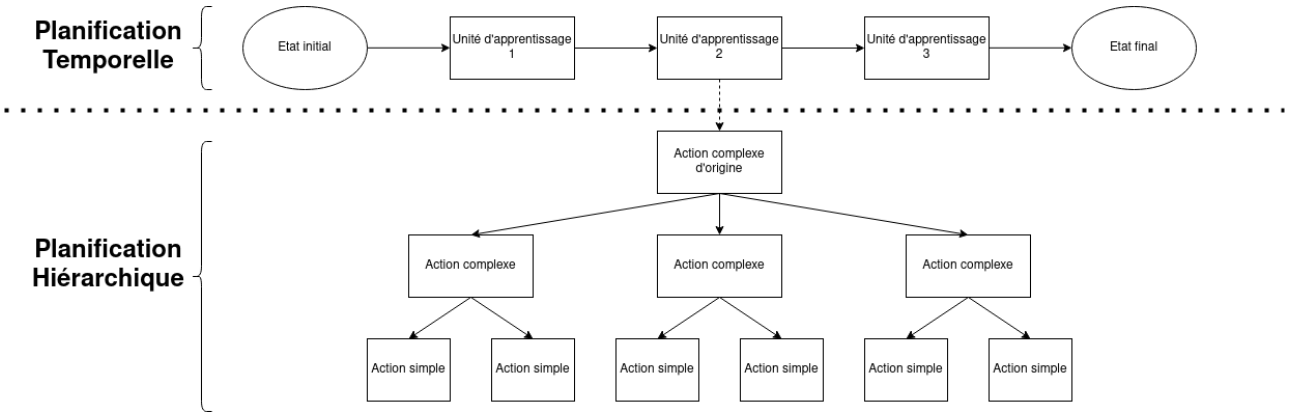


FIGURE 1 – Structure d'un scénario

3.2 Domaine de planification

Dans le cadre d'un projet visant à proposer des scénarios d'entraînement à des contrôleurs aériens militaires (projet ORCHESTRAA), nous avons réalisé un domaine de planification simplifié représentant des situations d'entraînement. Le domaine réel pour ce cadre d'application est beaucoup plus complexe, mais cette version simplifiée a pour but de présenter la structure de nos scénarios et les différents choix que nous avons faits pour modéliser le contenu narratif. Nous nous appuierons également sur cette structure pour expliquer le fonctionnement de nos algorithmes dans la section suivante.

3.2.1 Contenu pédagogique

Le contenu pédagogique est modélisé comme un domaine de planification classique avec contraintes temporelles. L'ensemble de la planification à ce niveau se fait sur des aspects pédagogiques. Ainsi, un plan à ce niveau de planification est une simple succession d'unités d'apprentissage (UA) caractérisées au minimum par une ou plusieurs compétences à travailler et une durée. Ce qui donne en ANML :

```
action UA1Fuel(validation v1) {
  duration := 10;

  [start] v1 == fuel;
  [end] isValidated(v1) := true;
};
```

Dans cet exemple l'UA1 permet de travailler une compétence de gestion de problème de *fuel*. Cette compétence doit être validée à la fin de l'UA. La durée de cette UA est de 10 unités de temps.

Le but de la planification à ce niveau est donc de valider toutes les compétences instanciées. Les UA peuvent être complexifiées avec du contenu pédagogique optionnel tel qu'un niveau de difficulté ou une représentation plus complexe de l'acquisition des compétences, mais ce n'est pas le but de ce travail pour lequel la version simplifiée convient parfaitement.

3.2.2 Contenu narratif

Le contenu narratif de notre scénario est modélisé comme un ensemble de problèmes de planification hiérarchique. Ainsi, chaque UA peut être représentée comme une action complexe, décomposable, dont le but est de valider l'ensemble de compétences liées à l'UA.

L'ensemble des actions simples, issues de la décomposition des actions complexes du plan, correspondent à des événements ou des actions directement instanciables dans l'environnement virtuel. Voici un exemple d'une action complexe décomposable en ANML :

```
action CAS() {
  motivated;

  :decomposition {
        triggerTIC(),
        informLearner(),
        learnerReaction(),
        endCAS()
  };
};
```

Cette action complexe décrit le décomposition d'un exercice de demande de support aérien dans un cadre militaire. Nous voyons donc que l'action de demande de support aérien (*CAS*) est décomposée en plusieurs étapes : le déclenchement d'une attaque militaire (*triggerTIC*), la notification à l'apprenant de ce déclenchement (*informLearner*), la réaction de celui-ci (*learnerReaction*) et enfin la clôture de cette demande d'assistance (*endCAS*).

Ainsi une action complexe peut-être décomposée en une liste d'actions (complexes ou non) qui sont ainsi directement liées à un certain contexte narratif. Ce domaine de planification est plus difficile à modéliser que pour le contenu pédagogique. Ceci vient du fait que ce contenu modélise des événements qui se déroulent dans l'environnement virtuel.

3.3 Génération automatique d'alternatives

Nous proposons maintenant de traiter le besoin de robustesse de nos scénarios grâce à un système de génération

d'alternatives *offline*. La principale caractéristique de notre système est qu'il utilise la planification de manière indirecte pour générer des alternatives. En effet, le fonctionnement normal d'un planificateur est, pour un domaine et un problème donné, de ne renvoyer qu'un seul plan correspondant aux critères de qualité de la technique de planification utilisée. Le rôle de nos algorithmes est ainsi d'effectuer des modifications dans le domaine de planification afin de forcer le planificateur à réaliser des choix différents à chaque planification.
Le principal avantage de cette approche est qu'elle permet d'utiliser n'importe quel planificateur acceptant de l'ANML en entrée. La génération des alternatives n'est ainsi pas dépendante des algorithmes de planification utilisés, mais est réalisée par rapport aux différents ajouts/suppressions effectués dans le domaine du problème de planification.
Ce système repose sur deux algorithmes. L'un pour la génération d'alternatives au niveau de la planification avec contraintes temporelles, l'autre au niveau de la planification hiérarchique.

Data : dom, the planning domain
Result : $plans$, a list of generated plans
$plans$.add(planner(dom))
$actionsSave \leftarrow plans[0]$
$domSave \leftarrow dom$
$combinations \leftarrow$ getCombinations($actionsSave$)
foreach $combi \in combinations$ **do**
  $actions \leftarrow actionsSave$
  $dom \leftarrow domSave$
  foreach $action \in combi$ **do**
    $actions$.add(failActions($action$))
    $dom \leftarrow$ addFailActions($action$)
  end
  $permutations \leftarrow$ getPermutations($actions$)
  foreach $perm \in permutations$ **do**
    $dom \leftarrow$ addConstraints($perm$)
    $plans$.add(planner(dom))
  end
end
return $plans$

Algorithme 1 : Génération de contraintes et d'échecs

La planification avec contraintes temporelles représente les aspects pédagogiques du scénario. Le but de planification à ce niveau est la validation de chacune des UA. L'un des problèmes qui peut donc arriver est l'échec de cette unité d'apprentissage. Cet échec se représente formellement par le non-respect de la post-condition de l'UA :

```
[end] isValidated(v1) := true;
```

L'algorithme 1 propose donc d'ajouter au domaine de planification des actions correspondant à des échecs sur les Unités d'apprentissage. Ces échecs sont des copies des UA originales mais avec un effet différent. L'Unité d'Apprentissage d'origine est également modifiée pour ajouter la validation de l'échec comme pré-condition :

```
action UA4e(validation v1)
{
  duration := 10;
  [start] v1 == UA4e;
  [end] isValidated(v1) := true;
};
action UA4(validation v1, validation v2)
{
  duration := 10;
  [start] v1 == UA4;
  [start] v2 == UA4e;
  [start] isValidated(v2) == true;
  [end] isValidated(v1) := true;
};
```

L'algorithme ajoute également des pré-conditions sur les autres UA pour leur donner un ordre que le planificateur est obligé de suivre. Ainsi, le rôle de la planification est ici de déterminer les combinaisons de contraintes permettant de trouver un plan valide.
Pour remplir ces objectifs, l'algorithme demande dans un premier temps une planification pour générer le plan initial. Si un plan est trouvé, celui-ci contient une organisation des différentes compétences à travailler lors de la séance (il n'existe donc pas d'autres actions potentiellement utiles, mais non utilisées dans le plan). L'algorithme calcule ensuite les différentes combinaisons possibles de ces actions (en comptant la combinaison nulle). Ces différentes combinaisons sont utilisées pour déterminer les actions échouées à générer. Par exemple :

$$\{UA1, UA3\} \rightarrow UA1e, UA3e$$

Ensuite, pour chacun de ces ensembles d'actions (initiales et échecs), nous calculons les différentes permutations afin d'ajouter des pré-conditions de validation entre chacune. Nous limitons cependant ces permutations à celles dont la première action est soit, la première action du plan initial, soit sa version échouée. Pour chacune de ces permutations, le planificateur est appelé pour déterminer sil existe un plan ou non. Chacun des plans ainsi générés constitue une alternative au scénario initial.
L'algorithme 2 prend en charge la partie hiérarchique du scénario. L'objectif est ici d'explorer les différentes décompositions possibles des actions complexes. En effet, pour une action complexe donnée il peut y avoir plusieurs décompositions proposées. Le planificateur, lorsqu'il génère son plan, choisit les décompositions en fonction de sa stratégie de planification. Pour forcer le planificateur à nous proposer des alternatives, notre algorithme commence par générer un plan initial. Celui-ci est ensuite parcouru en chaînage arrière. Lorsque l'action observée est une action complexe, l'algorithme vérifie le nombre de décompositions dont elle dispose. Si plusieurs décompositions cohabitent, celle ayant déjà été utilisée est supprimée du domaine de planification et le planificateur est rappelé pour

Data : dom, the planning domain
Result : $plans$, a list of generated plans
$plans$.add(planner(dom))
$domSave \leftarrow dom$
$actions \leftarrow plans[0]$
$count \leftarrow 0$
while $count < actions.size()$ **do**
  $reverseActions \leftarrow$ getReversedActions($actions$)
  foreach $action \in reverseActions$ **do**
    if *hasDecompositions(action)* **then**
      $dom \leftarrow$ delCurrentDec($action$)
      $plans$.add(planner(dom))
      $actions \leftarrow$ lastPlan($plans$)
      $count \leftarrow 0$
    end
    else
      $count \leftarrow count + 1$
    end
  end
end
$dom \leftarrow domSave$
return $plans$

Algorithme 2 : Exploration HTN

générer un nouveau plan. Le planificateur est ainsi obligé de choisir une nouvelle décomposition. Si un nouveau plan est trouvé, l'algorithme recommence le chaînage arrière avec le nouveau plan selon le même principe. Il se termine lorsqu'il n'existe plus d'alternatives dans les décompositions des actions complexes.
Pour que cette approche fonctionne, il est cependant nécessaire de poser quelques limites fortes. En premier lieu, nous posons comme condition que les décompositions soient ordonnées. En effet, en planification hiérarchique, deux types de décompositions sont possibles : ordonnées et non-ordonnées. Dans les deux cas, l'ensemble des actions de la décomposition doit être résolu, le changement vient de l'ordre de résolution. Dans le premier cas l'ordre est fixé par le domaine, dans le second cas c'est le planificateur qui fait ce choix. Avec notre approche, une décomposition non ordonnée impliquerait autant de possibilités que de permutations des actions pour chaque décomposition. Nous avons cependant écarté cette possibilité car, dans notre cadre d'application, les actions complexes correspondent à des processus qui suivent un ordre précis, il n'est donc pas nécessaire de gérer les décompositions non-ordonnées.
Notre seconde condition est de ne pas permettre la réutilisation d'une action complexe présentant plusieurs décompositions dans différentes décompositions d'autres actions complexes. Cette règle permet d'éviter que la suppression d'une décomposition à un instant donné ne modifie le domaine plus haut dans le plan. Encore une fois cette règle ne pose pas de problème dans notre cadre d'application, car nos actions complexes représentent des événements uniques. Cette règle peut en outre être contournée en créant des versions différentes des actions potentiellement concernées.
Les plans générés au niveau pédagogique et narratifs constituent ainsi un ensemble précalculés de plans validés par un planificateur.

4 Expérimentations préliminaires

Nous avons testé la puissance générative de notre système avec notre domaine ANML simplifié. Le planificateur utilisé pour ce test est le planificateur FAPE [4]. Au niveau pédagogique nous avons effectué plusieurs tests en augmentant le nombre d'UA. Au niveau narratif, notre domaine de planification était composé de 16 actions, dont 10 actions complexes. Nous avons fait varier le nombre de décompositions de ces différentes actions complexes. Nos résultats sont résumés dans les tableaux suivants :

Nombre d'Unités d'Apprentissage	2	3	4
Nombre de plans générés	7	74	1134

TABLE 1 – Plans générés en fonction du nombre d'Unités d'Apprentissage

Nombre de décompositions	4	10	15
Nombre de plans générés	4	9	12

TABLE 2 – Plans générés en fonction du nombre de décompositions

Nous observons dans le tableau 1 que le nombre de plans générés augmente très rapidement dès que l'on ajoute une UA. Nous ne sommes pas montés au-dessus de quatre UA car le temps de calcul nécessaire devient trop long. Cette augmentation très rapide est due au calcul des permutations qui ne se limitent pas aux seules UA, mais également aux versions échouées. Ainsi, pour un domaine composé de quatre UA, la liste d'éléments monte jusqu'à huit éléments. Le tableau 2 montre une progression différente du nombre de plans générés. En effet, ce nombre reste toujours inférieur au nombre de décompositions présentes dans le domaine. Ces résultats semblent logiques étant donné que les décompositions sont supprimées au fur et à mesure. La différence entre le nombre de décompositions et le nombre de plans vient du fait que certaines décompositions ne peuvent pas être choisies par le planificateur car leur parcours est soit impossible du fait des pré-conditions, soit ne permettant pas d'atteindre le but de la planification.

5 Conclusion

Le système de planification hybride que nous proposons permet de séparer les aspects pédagogiques et narratifs au sein d'un même domaine de planification. Au-delà du gain d'expressivité apporté par cette modélisation, ce système

ouvre la voie à certaines améliorations importantes. En effet, le contenu pédagogique peut sembler très simple et ne nécessite pas un grand travail d'écriture. Mais cette simplicité devrait permettre, dans le cadre de travaux futurs, à la fois une génération automatique du scénario, mais également de la partie narrative du domaine de planification. Cette automatisation permettrait ainsi une configuration très simple des aspects pédagogiques d'une séance d'entraînement.

Les résultats obtenus sur la génération d'alternatives sont également intéressants. Ils doivent être complétés par d'autres tests avec des domaines de planification et des configurations différents, mais nous pouvons d'ores et déjà voir que cette approche permet de générer un nombre conséquent d'alternatives. Parmi les alternatives générées il n'y a cependant pas d'indication sur la qualité des plans, l'une de nos perspectives sur ce travail est donc de réussir à trier ces différentes propositions afin de supprimer les plans les moins utiles et classer les plans restant en fonction de métriques qu'il reste à déterminer. Il faut également réfléchir à l'impact des contraintes internes des différents domaines de planification sur le nombre de plans générés. Ainsi, plus le domaine contient de restrictions et moins il y a de possibilités de générer des alternatives. Avoir un certain nombre de restrictions est cependant intéressant pour ne pas générer trop d'alternatives dont une grande partie risque de se révéler inutile. Une recherche du niveau de restriction souhaitable pourrait donc apporter des informations utiles pour la création de domaines de planification efficaces.

Enfin, le langage ANML utilisé pour écrire nos domaines de planification, de par ses fonctionnalités de haut niveau par rapport à PDDL, nous semble très adapté au contexte de la narration interactive. Ce langage reste cependant très peu utilisé par rapport à PDDL et il existe peu de planificateurs fonctionnels. Cette situation fait que certaines fonctionnalités de l'ANML qui pourraient être utiles restent à ce jour inutilisables, car les planificateurs existants ne les implémentent pas.

Références

[1] Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins SRI, Anthony Barrett, Dave Christianson, et al. Pddl| the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

[2] Ruth Aylett, Joao Dias, and Ana Paiva. An affectively driven planner for synthetic characters. In *Icaps*, pages 2–10, 2006.

[3] Camille Barot. *Scénarisation d'environnements virtuels. Vers un équilibre entre contrôle, cohérence et adaptabilité.* PhD thesis, Université de Technologie de Compiègne, 2014.

[4] Arthur Bit-Monnot, Malik Ghallab, Félix Ingrand, and David E Smith. Fape : a constraint-based planner for generative and hierarchical temporal planning. *arXiv preprint arXiv :2010.13121*, 2020.

[5] Richard E Fikes and Nils J Nilsson. Strips : A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4) :189–208, 1971.

[6] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated planning and acting*. Cambridge University Press, 2016.

[7] Satyandra K. Gupta, Davinder K. Anand, John E. Brough, Maxim Schwartz, and Robert Kavetsky. *Training in Virtual Environments : A Safe, Cost-Effective, and Engaging Approach to Training*. University of Maryland, 2008.

[8] Daniel L Kovacs. Bnf definition of pddl 3.1. *Unpublished manuscript from the IPC-2011 website*, 15, 2011.

[9] Kalpesh Padia, Kaveen Herath Bandara, and Christopher G Healey. A system for generating storyline visualizations using hierarchical task network planning. *Computers & Graphics*, 78 :64–75, 2019.

[10] Julie Porteous. Planning technologies for interactive storytelling. In *Handbook of Digital Games and Entertainment Technologies*. Springer, 2016.

[11] Julie Porteous, Jonathan Teutenberg, Fred Charles, and Marc Cavazza. Controlling narrative time in interactive storytelling. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 449–456, 2011.

[12] Alejandro Ramirez and Vadim Bulitko. Automated planning and player modeling for interactive storytelling. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(4) :375–386, 2014.

[13] Mark Riedl, Cesare J Saretto, and R Michael Young. Managing interaction between users and agents in a multi-agent storytelling environment. In *Proceedings of the 2nd international joint conference on Autonomous agents and multiagent systems*, pages 741–748, 2003.

[14] Mark O Riedl, Andrew Stern, Don Dini, and Jason Alderman. Dynamic experience management in virtual worlds for entertainment, education, and training. *International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning*, 4(2) :23–42, 2008.

[15] Mark Owen Riedl and Vadim Bulitko. Interactive narrative : An intelligent systems approach. *Ai Magazine*, 34(1) :67–67, 2013.

[16] David E Smith, Jeremy Frank, and William Cushing. The anml language. In *The ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, volume 31, 2008.

[17] Tom Trabasso and Paul Van Den Broek. Causal thinking and the representation of narrative events. *Journal of memory and language*, 24(5) :612–630, 1985.

Session 2

Learning Path Constraints for UAV Autonomous Navigation under Uncertain GNSS Availability

Marion Zaninotti[1,2] Charles Lesire[2] Yoko Watanabe[2] Caroline P. C. Chanel[1]

[1] ISAE-SUPAERO, University of Toulouse, France
[2] ONERA, University of Toulouse, France

marion.zaninotti@isae-supaero.fr, charles.lesire@onera.fr
yoko.watanabe@onera.fr, caroline.chanel@isae-supaero.fr

Résumé

Cet article adresse un problème de planification de chemin sûr pour la navigation de drone en environnement urbain, sous une disponibilité du GNSS incertaine. Le problème peut être modélisé comme un POMDP et résolu en utilisant des algorithmes à base d'échantillonnage. Cependant, un domaine aussi complexe souffre d'un coût de calcul important et atteint des résultats médiocres sous une contrainte temps-réel. Des recherches récentes visent à intégrer un apprentissage hors ligne pour ensuite guider de manière efficace la planification en ligne. Inspiré par la formalisation de l'état de l'art CAMP (Context-specific Abstract Markov decision Process), cet article propose de réaliser une étape d'apprentissage hors ligne de contrainte sur le chemin et d'appliquer le résultat pour réduire l'espace de recherche de politique pendant la résolution en ligne du POMDP. Plus précisément, le sélecteur de la meilleure contrainte est appris de manière hors ligne et il est utilisé pour sélectionner la contrainte à imposer, pendant la planification en ligne, en fonction des caractéristiques de la tâche, i.e. la probabilité de disponibilité du GNSS dans notre cas d'application. Les conclusions des expériences menées pour différents environnements montrent qu'utiliser l'approche proposée permet d'améliorer la qualité d'une solution atteinte par un planificateur en ligne, en particulier lorsque la probabilité de disponibilité du GNSS est faible.

Mots Clef

Planification de chemin en ligne, Apprentissage pour la planification, POMDP.

Abstract

This paper addresses a safe path planning problem for UAV urban navigation, under uncertain GNSS availability. The problem can be modeled as a POMDP and solved with sampling-based algorithms. However, such a complex domain suffers from high computational cost and achieves poor results under real-time constraint. Recent research seeks to integrate offline learning in order to efficiently guide online planning. Inspired by the state-of-the-art CAMP (Context-specific Abstract Markov decision Process) formalization, this paper proposes to perform an offline path constraint learning process and to apply its result to reduce the policy search space during online POMDP solving. More precisely, the best constraint selector is learnt offline and it is used to select the constraint to impose, during online planning, according to features of the task, i.e. GNSS availability probability in our application case. Conclusions of experiments carried out for different environments show that using the proposed approach allows to improve the quality of a solution reached by an online planner, particularly when GNSS availability probability is low.

Keywords

Online path planning, Learning for planning, POMDP.

1 Introduction

Solving autonomous navigation problems consists in finding a path from an initial position to a goal with a maximum efficiency, while avoiding the obstacles. These problems become challenging when the state of the vehicle is uncertain. Particularly, most of Unmanned Aerial Vehicles (UAVs) are equipped with a Global Navigation Satellite System (GNSS) receiver as navigation system. In an urban environment, the visibility of the GNSS satellite constellation can be reduced by the buildings surrounding the UAV, the precision or even the availability of the GNSS position estimate can then be significantly altered, what can lead to a fatal collision.

[Delamer *et al.*2021] formalize the UAV urban navigation problem under uncertain GNSS availability as a Partially Observable Markov Decision Process (POMDP) [Kaelbling *et al.*1998]. The latter is a principled approach to solve planning problems under uncertainty. However, POMDP planning faces two notorious problems. The first one is the "*curse of dimensionality*" : the size of the belief state space grows up exponentially with that of the state space. The second problem is the "*curse of history*" : the number of action/observation sequences to evaluate during

research grows up exponentially with the planning horizon [Pineau *et al.*2006]. The use of a Partially Observable Monte-Carlo Planning (POMCP) [Silver and Veness2010] algorithm makes it possible to overcome these difficulties. Nevertheless, the performance of these algorithms remains dependent on the search depth reached within the planning horizon, which is itself dependent on the *branching factor* of the search tree [Hostetler *et al.*2014]. The branching factor includes both the *action factor*, *i.e.* the number of actions available in each belief state, and the *stochastic factor*, *i.e.* the number of possible observations for each action. The stake is then to reduce the branching factor in order to scale up planning. This is all the more important for online planning : the planner has to make a decision quickly whereas the long-planning horizon of such a real-world task incurs prohibitive computational cost. For that, incorporating domain abstraction is a promising approach. [Chitnis *et al.*2021] introduce Context-specific Abstract Markov Decision Process (CAMP), an abstraction of the original MDP model, obtained by learning and imposing the *best constraint* on the states and actions considered by the agent. This best constraint is chosen by an offline learnt *context selector* according to the *features* of a task.

Inspired by this way of domain abstraction of CAMP, this paper proposes to integrate an offline learning step in order to reduce the policy search space during online POMDP solving. In our application case, the context selector returns the constraint which reduces the UAV position state space while preserving the solution optimality, in function of the GNSS availability probability map of a task. Unlike the original CAMP, we address a partially observable domain. In our case, applying action space abstraction is not straightforward as states are not fully observable. Nevertheless, a state space abstraction can be achieved through modification of the cost function for penalizing the constraint violation, which will modify the action outcomes. Additionally, as our objective is to perform online planning for the UAV safe navigation problem, whereas an offline POMCP variant is used in the offline process, an online version is applied for planning. As result, we investigate the use of different algorithms for learning and planning, which has not been done in the original work of CAMP. Thus, regarding the UAV navigation problem with uncertain GNSS availability, our contribution is twofold : (i) we investigate if domain abstraction, by adapting the CAMP framework for a partially observable domain, gives better results when compared to a full POMDP model, and (ii) we evaluate if such a CAMP-inspired approach is robust if we use a different algorithm for learning and planning.

After providing the theoretical background and the related work in the next section, we present the CAMP method adapted to our problem in Section 3. Experimental results are reported in Section 4, demonstrating the planning performance improvement. Section 5 includes concluding remarks and future works.

2 Background and Related Work

2.1 POMDP Preliminaries

A POMDP [Kaelbling *et al.*1998] is defined as a tuple $(\mathcal{S}, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{O}, \mathcal{C}, b_0, \gamma)$, where $\mathcal{S}$, $\mathcal{A}$, and Ω denote respectively spaces of states, actions, and observations. The *transition function* $\mathcal{T}(s, a, s') = p(s'|s, a)$ represents the dynamics of the agent as the probability of transiting from s to s' by taking action a. The *observation function* $\mathcal{O}(a, s', o) = p(o|s', a)$ specifies the probability of observing o after taking action a to reach s'. The *cost function* $\mathcal{C}(s, a)$ defines the cost of taking action a in s. b_0 denotes the initial belief state. $\gamma \in [0, 1]$ is a discount factor expressing a preference for minimizing immediate over future cost.
POMDPs capture partial observability of the system using the *belief state* b, *i.e.* a probability distribution over $\mathcal{S}$, that is updated after each action a and observation o using the Bayes' rule. A POMDP policy $\pi : \mathcal{B} \rightarrow \mathcal{A}$ prescribes an action for each belief state in the *belief space* $\mathcal{B}$. Solving a POMDP requires finding the *optimal policy* π^* minimizing the expected future cost, called the *value*, for all $b \in \mathcal{B}$. The value of the policy π^* in belief b is defined as :

$$V^{\pi^*}(b) = \min_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{C}(b_t, \pi(b_t)) \big| \, b_0 = b\right] \quad (1)$$

Additionally, the Q-value of an action a in a belief state b can be defined as :

$$Q^{\pi}(b, a) = \mathbb{E}\left[\mathcal{C}(b, a) + \sum_{t=1}^{\infty} \gamma^t \mathcal{C}(b_t, \pi(b_t))\right]. \quad (2)$$

2.2 UAV Urban Navigation POMDP-based Problem

The original planning model proposed by [Delamer *et al.*2021] is formalized as a Mixed-Observability Markov Decision Process (MOMDP) [Ong *et al.*2010], a special class of the POMDP framework. The state space is factorized into fully, s_v, and partially, s_h, observable state variables, what reduces the belief state space dimension, and in turn, reduces policy computation time. The state tuple $s = (s_h, s_v) \in \mathcal{S}$ is defined with $s_h = (\mathcal{X}, \mathcal{V}, \beta_a)$ where $\mathcal{X}$ and $\mathcal{V}$ are the vehicle position and velocity, and β_a is the IMU acceleration measurement bias, and $s_v = (F_{col}, F_{GNSS}, P, t_{flight})$ with F_{col} and F_{GNSS} the collision and GNSS availability Boolean flags, P the estimation error covariance matrix over s_h, and t_{flight} the flight time elapsed. An action $a \in \mathcal{A}$ corresponds to the desired velocity direction. The action space $\mathcal{A}$ is a finite set of 10 actions, following 8 radial directions in the 2D horizontal plane, plus up and down. An observation $o \in \Omega$ is defined as the sub-tuple $o = s_v$ of the state tuple, given a full observability of (F_{col}, F_{GNSS}) and a deterministic transition of (P, t_{flight}). This partial state observability limits the branching factor of the search tree. The transition function follows a GNC (Guidance,

Navigation and Control) model, composed of the vehicle motion model, a guidance law, a state estimator, and the IMU and GNSS sensor models. The GNSS availability F_{GNSS} affects how the state estimator propagates the error covariance P, which affects the belief state b' after transition. In brief, P grows when GNSS is unavailable, resulting in more collision risk. Finally, the cost function is defined as :

$$\mathcal{C}(s,a) = \begin{cases} 0 & \text{if goal reached} \\ K_{col} - t_{flight} & \text{if collision} \\ \Delta T_a & \text{otherwise} \end{cases} \quad (3)$$

with $\Delta T_a > 0$ the action execution time, and $K_{col} > 0$ the cost penalty in case of collision. When a collision occurs, the cost of any action is this penalty subtracted with the flight time elapsed. It avoids penalizing more if the collision occurs after a long flight time or near the goal.

2.3 MinPOMCP-GO Algorithm

[Delamer *et al.*2021] propose the POMCP - Goal-Oriented (POMCP-GO) algorithm, an offline goal-oriented variant of POMCP [Silver and Veness2010]. It samples a state s from the initial belief state b_0 corresponding to the *root node*, and simulates action/observation sequences, through *trials*, in order to evaluate actions while building a tree of nodes. To perform a *trial*, POMCP-GO follows a given action selection strategy and a heuristic node value initialization. For the action selection, it relies on the Upper Confidence Bounds (UCB1) strategy [Kocsis and Szepesvári2006] to deal with the exploration–exploitation dilemma. A trial is stopped when a terminal state is reached (a goal or collision state), and this procedure is repeated during a fixed timeframe.
Each tree node h represents a history of action/observation sequences from the initial belief state. The Q-value (Eq. 2) of a belief state is approximated by $Q(h,a)$, which is the mean cost returned from all trials started from h when action a was selected. This approximation incurs a well-known bias, which decreases as the number of trials increases [Keller and Helmert2013]. To accelerate the policy value convergence by reducing the Q-value bias, [Carmo *et al.*2020] propose the MinPOMCP-GO algorithm which uses a *Min-Monte-Carlo backup* [Keller and Helmert2013].
The present paper approach is based on this MinPOMCP-GO algorithm. During tree building, MinPOMCP-GO initializes the Q-value of a newly created node to a precomputed heuristic value, corresponding to the flight time left to the goal estimated by the Dijkstra algorithm [Dijkstra1959]. Even if this heuristic function is more informative than the traditional rollout used in POMCP, it does not consider GNSS availability probability, hence sampling trajectories that may lead to collisions due to the uncertain UAV position estimate. The impact of GNSS unavailability is only considered indirectly, by back-propagating the cost penalty when a collision occurs.

2.4 Domain Abstraction

Sampling-based algorithms, such as POMCP and variants, suffer from exponential complexity with respect to the branching factor of the search tree. In our UAV navigation problem under uncertain GNSS availability, the solver cannot explore enough, within a short decision-making timeframe, to prevent collisions. In difficult environments, with obstacles reducing GNSS probability availability, navigation mission safety may be compromised. So, we focus on incorporating domain abstraction to reduce the branching factor and to improve online planning solutions.

State Aggregation. One well-known technique of domain abstraction is *state aggregation* : the state space is reduced by clustering equivalent states, *i.e.* states that share some fully-identical properties - *exact aggregation* - or nearly-identical properties - *approximate aggregation* - and treating each of these resulting state clusters as one. In [Li *et al.*2006], the authors list the existing methods of exact state aggregation and unify them to deduce five generic functions. However, since two states rarely share some fully-identical properties, exact abstraction is often useless, while approximate abstraction can achieve greater degrees of compression. In [Abel *et al.*2016], the authors present four types of approximate aggregation and demonstrate that they lead to a bounded loss of optimality of behavior. In [Hostetler *et al.*2014], the authors generalize the formulation of two of these four types of aggregation and apply them to Monte-Carlo Tree Search (MCTS). AS-UCT [Jiang *et al.*2014], ASAP-UCT [Anand *et al.*2015], and OGA-UCT [Anand *et al.*2016] are other implementations of state or state-action pair aggregation to UCT, a MCTS algorithm variant. All of these methods have not been applied in the partially observable framework.

Hierarchical Planning. Another approach to domain abstraction is *hierarchical planning*. It consists in decomposing the planning problem into a network of independent subgoals. *Hierarchical Dynamic Programming* (HDP) [Bakker *et al.*2005] is an example of hierarchical planning for navigation problems. A hierarchy of MDPs is constructed and solved using a hierarchical variation of value iteration. *Abstract Markov Decision Process* (AMDP) [Gopalan *et al.*2017] is a more general method, which allows any MDP planner to be used. Both HDP and AMDP are *top-down* approaches : they select the subgoal before performing planning to reach it. Contrary to *bottom-up* approaches, they present the advantage that planning is necessary only for subgoals used for task completion. Nevertheless, the way to define appropriate subgoals remains an open question.

Integrating Learning for Planning. A third method is to integrate an offline learning phase as a first step, to guide the search during online planning. The CAMP approach [Chitnis *et al.*2021], that has inspired this paper, is part of it. It searches the best reduced state and action spaces by imposing a constraint learnt according to the features of a

task. Another example is Macro-Action Generator-Critic (MAGIC) [Lee *et al.*2021], which learns the more efficient set of candidate macro-actions to cut down the effective planning horizon, being a kind of *temporal abstraction*.

As previously discussed in Section 2.3, one of the weaknesses of MinPOMCP-GO is that the heuristic function does not consider the GNSS availability probability, what can misguide the search. The planning efficiency can hence be improved if we could use this information to further focus the search on more relevant areas, *i.e.* where GNSS is more likely available. For this purpose, the CAMP method seems a good candidate to leverage. Implemented for our problem, it allows to reduce the state space in function of a probability map of GNSS availability, considering the latter as a task feature.

3 Learning Path Constraints based on GNSS Availability

3.1 Approach Overview

The objective of the CAMP method [Chitnis *et al.*2021] is to learn a context selector $f : \Theta \rightarrow \mathcal{C}$. Each *training task* corresponds to a *feature vector* $\theta \in \Theta$. For each feature vector, the best constraint $C^* \in \mathcal{C}$ is identified. The pairs (θ_i, C_i^*) are given to a neural network to learn f. Once the context selector f is learnt, the best constraint C^* returned from the feature vector θ is then imposed to guide online planning.

In our navigation problem under uncertain GNSS availability, we assume a given environment, *i.e.* known obstacles on a map, and a given navigation mission, *i.e.* fixed initial position and goal. Figure 1 describes our application of the context selector learning process to our problem. The probability maps of GNSS availability are used as feature vectors. For each training probability map of GNSS availability, the best constraint is identified. We define a constraint as a corridor of the environment in which the UAV must stay, and we evaluate it by performing planning within a *training timeout*. Then, these probability maps of GNSS availability and the associated best constraints are used to learn the context selector. Finally, the *test tasks* are solved online, imposing the best constraints returned by the context selector from the test probability maps of GNSS availability. Each step of this process is detailed in the following sub-sections.

3.2 Feature Vectors

As previously discussed, the GNSS availability is crucial to determine safe paths for our UAV. As the GNSS satellites are orbiting around the Earth, the GNSS availability probability varies with the time-of-the-day even for a fixed obstacle environment. We then propose to leverage the CAMP learning approach to reduce the state space by computing navigation constraints based on probability maps of GNSS availability.

A quality of the GNSS position estimate is given as a metric called *Position Dilution Of Precision* (PDOP) [Kleijer *et al.*2009]. Given satellite geometry and user location, a PDOP map is generated by using a GNSS simulator. We consider PDOP value as a standard deviation of the GNSS positioning error, assumed to follow a zero-mean Gaussian distribution [Delamer *et al.*2021]. Then, the PDOP map is transformed to a probability map of GNSS availability by using $\mathbf{erf}$, the Gauss error function, and by setting a maximum position error threshold ϵ :

$$\Pr(F_{GNSS} = 1) = \mathbf{erf}(\frac{\epsilon}{\sqrt{2}\mathrm{PDOP}}). \quad (4)$$

First, we generated test task features by setting different ϵ values to cover the easy and difficult cases where GNSS is most-like available/unavailable. Then, the training task features were generated by linear combination of these test tasks with randomly selected coefficients, for more feature variety.

3.3 Constraint Definition and Evaluation

Constraint Definition. We divide the environment map into $n_L \times n_l \times n_h$ areas in an uniform way. n_L denotes the number of areas over the length, n_l over the width, and n_h over the height. For each of these areas, we define a corridor of areas leading from the initial position to the goal one, passing through this area, called *passage area*. For that, we concatenate the paths resulting from the A* algorithm [Hart *et al.*1968] from the area including the initial position to this passage area, and from the latter to the area including the goal position. We use the number of areas constituting the path as cost function in the A* algorithm. We obtain thus at most $n_L \times n_l \times n_h$ different corridors of areas, corresponding to *candidate constraints*, in which the UAV is allowed to navigate. The sub-figure in the middle of Figure 2 shows a candidate constraint defined by dividing the environment map into $(n_L = 5) \times (n_l = 5) \times (n_h = 1)$ areas, and using the top left area as passage area, highlighted in blue.

Planning with Constraint. For each training probability map of GNSS availability, all the candidate constraints are evaluated. For that, planning imposing the candidate constraint is executed. We use the *MinPOMCP-GO* planning algorithm [Carmo *et al.*2020], adapting its heuristic function, which estimates the flight time left to the goal, so that the navigation constraint is respected. Figure 2 illustrates an example of the heuristic map obtained from a given environment, navigation mission, and candidate constraint. On the environment map on the left, as on the following maps, the initial position and the goal are respectively represented by a point and a star, and the obstacles are depicted in yellow. On the heuristic map on the right, the estimated flight time left to the goal is represented inside the constraint. To impose a constraint, the cost function of the planning model (Eq. 3) is also adapted so that it considers a violation of the constraint as a terminal

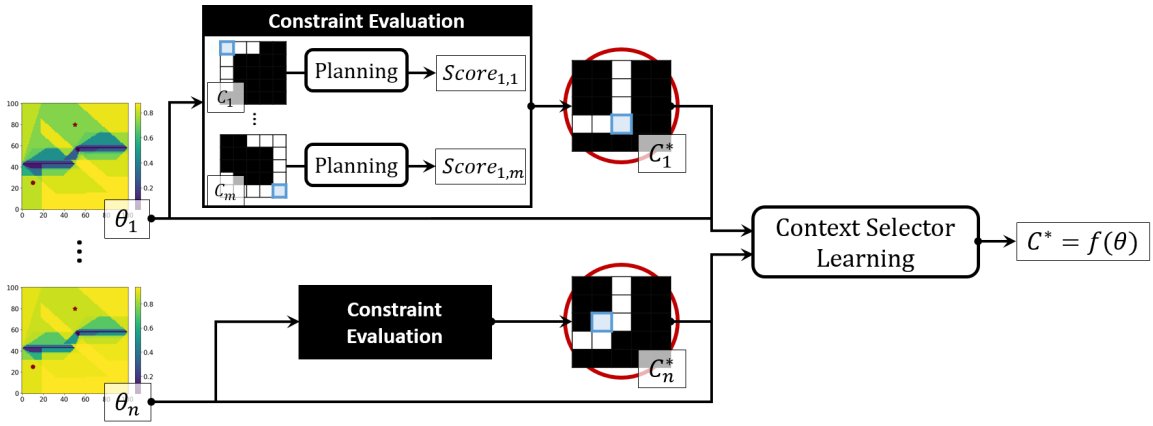


FIGURE 1 – Learning the context selector f by identifying the best constraint C_i^*, for each training task i associated to a probability map of GNSS availability θ_i.

state which leads to a cost penalty K_{constr}. In addition, the collision cost is saturated by a minimal threshold $K_{col_{thr}}$, as some imposed constraints incur longer flight times. The cost function then becomes :

$$\mathcal{C}(s,a) = \begin{cases} 0 & \text{if goal reached} \\ \max(K_{col} - t_{flight}, K_{col_{thr}}) & \text{if collision} \\ K_{constr} & \text{if constraint violation} \\ \Delta T_a & \text{otherwise} \end{cases} \quad (5)$$

Best Constraint Identification. In [Chitnis *et al.*2021], a candidate constraint is evaluated using a score formulation, which expresses the trade-off between *how much planning is sped up* and *how much optimality is preserved* imposing this constraint. Reaching the convergence on the policy value is required to estimate the planning time and the policy value. However, it is difficult to judge and can take too long to achieve this convergence in our problem. Therefore, to evaluate a candidate constraint, we stop planning when a *training timeout* is reached and express its score as the inverse of the resulting initial belief state value $V^\pi(b_0)$. For a probability map of GNSS availability θ_i, the candidate constraint that achieves the highest score, *i.e.* the lowest initial belief state value, is chosen as the best constraint, and is noted C_i^*.

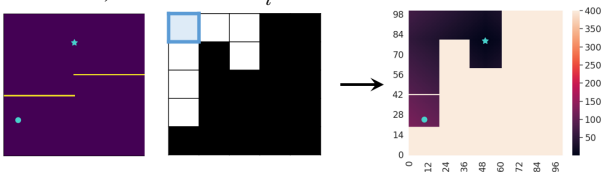


FIGURE 2 – Generation of the heuristic map (right) from an environment, a navigation mission (left), and a candidate constraint (middle).

3.4 Context Selector Learning and Online Planning

The feature vectors $\{\theta_i\}$ of the training tasks and the associated best constraints $\{C_i^*\}$ are used to train a neural network with cross-entropy loss, resulting in the context selector f (Fig. 1). The generic neural network available in the CAMP framework is applied, with the proposed Fully Connected Network architecture [Chitnis *et al.*2021].

At test time, the best constraint is returned by the context selector, given the feature vector of the test task : $C^* = f(\theta)$. This constraint is then integrated in the model to reduce the state space, by imposing to compute navigation paths that stay within the constraint, *i.e.* the corridor. We use two planning algorithms to compute these navigation paths. The first one is *MinPOMCP-GO*, also used for evaluating the constraints in the training phase. The second algorithm is *MinPOMCP-GO** : it is a variant of MinPOMCP-GO in which trials end whenever a previously unvisited leaf node is encountered instead of ending a trial only when a terminal state is reached. MinPOMCP-GO* is aimed to be used online, as it produces more trials with a shortest depth, hence favoring short-term performance that would help avoiding collisions, while taking into account actual observations in an online setting.

4 Experiments

We implement the previously described method to three navigation benchmark environments available in [Mettler *et al.*2010] : *Cube Baffle*, containing two cubes, *Wall Baffle*, containing two walls, and the real downtown of *San Diego*. They are illustrated in Figure 3.

(a) (b) (c)

FIGURE 3 – *Cube Baffle* (a), *Wall Baffle* (b), and *San Diego* (c).

To evaluate our approach, four test tasks are solved for each environment, numbered from 1 to 4, corresponding to the maps presenting from the lowest to the highest GNSS avai-

lability probabilities. We compare the results obtained imposing the best constraint returned by the context selector with those obtained without constraint. The performance metrics are the number of collisions and the mean costs obtained considering a fixed decision-making timeframe. The lower they are, the better performance is.

4.1 Material

To carry out the experiments, we use a supercomputer constituted of 24 cores. For each of these cores, the frequency is of 2.60 GHz, the Random Access Memory size is of 96 Go, and the cache size is of 19.25 Mo.

4.2 Settings

In the following, we describe the settings used in our experiments. The GNC model and the reference velocity settings are the same as those described in [Delamer *et al.*2021].

Initial Position and Goal. The mean initial position is set to X_0 = [10, 25, 5] for *Cube Baffle* and *Wall Baffle*, and X_0 = [110, 60, 5] for *San Diego*. The goal position is set to X_G = [85, 78, 5] for *Cube Baffle*, X_G = [50, 80, 5] for *Wall Baffle*, and X_G = [200, 125, 5] for *San Diego*.

Map Decomposition. The map size of *Cube Baffle* and *Wall Baffle* is [101, 101, 21]. For *San Diego*, it is [217, 167, 24]. The maps are uniformly divided into $(n_L = 5) \times (n_l = 5) \times (n_h = 1)$ areas.

Model and Solver. The action cost ΔT_a is set to 2.2. The collision penalty K_{col}, its threshold $K_{col_{thr}}$, and the constraint violation penalty K_{constr} (Eq. 5) are respectively set to 450, 350, and 450. The exploration coefficient c of UCB1 is set to 6.

Training Tasks. The training timeout is set to 2 minutes and the number of training tasks, *i.e.* the number of probability maps of GNSS availability used for training, is 30.

Neural Network and Test Tasks. The neural network loss threshold is set to 1.8. The decision-making timeframe is set to 2 seconds and the number of test tasks, *i.e.* the number of probability maps of GNSS availability used for testing, is 4. These maps are generated with the error thresholds : ϵ = 1, 2, 5, and 10 meters. For each test task, 50 episodes are launched.

4.3 Results

The performance metric values obtained for each environment are summarized in Table 1. The probability maps of GNSS availability at the initial and goal altitude are displayed as background of the following figures, the resulting paths are plotted in red and the collisions are represented by black dots.

For the *Cube Baffle* environment, the costs obtained without constraint and imposing the best constraint, using MinPOMCP-GO or MinPOMCP-GO*, are similar for all the test tasks. Indeed, the UAV does not fly close enough to the cubes and the GNSS availability probability is sufficiently high. Hence, very few collisions occur, even without constraint. Figure 4 shows the resulting paths without constraint and imposing the best constraint for the first test task, corresponding to the lowest GNSS availability. The imposed constraint makes the resulting paths deviate to avoid the zones of possible GNSS loss to reduce the collision risk.

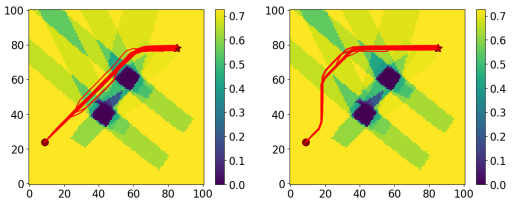


FIGURE 4 – Results obtained for *Cube Baffle*, for test task 1 : paths obtained using MinPOMCP-GO*, without constraint (a), and with the best constraint (b).

Figure 5 shows the resulting paths without constraint and imposing the best constraint for the *Wall Baffle* environment. With MinPOMCP-GO or MinPOMCP-GO*, the number of collisions and the cost obtained imposing the best constraint are considerably lower than those without constraint for the test tasks (1) and (2), corresponding to the two maps presenting the lowest GNSS availability probabilities. For test task (1), using MinPOMCP-GO, the number of collisions obtained imposing the best constraint is reduced of almost 72% ; and using MinPOMCP-GO*, it is reduced to 0. For these test tasks, the best constraint forces to fly over the wall, where GNSS availability probability is greater, instead of flying between the two walls as obtained when no constraint is imposed. Even if the flight time becomes a bit longer, the cost is much reduced because less collisions occur. That is, the mission safety is largely improved. For the third test task, with MinPOMCP-GO or MinPOMCP-GO*, the cost is slightly increased when imposing the best constraint, still favoring the safer paths flying over the wall. Finally, for the fourth test task presenting the highest GNSS availability probabilities, the best constraint only imposes to slightly move away from the first wall. It results in a slight decrease of the collision rate, with MinPOMCP-GO or MinPOMCP-GO*.

The *San Diego* environment includes multiple buildings, that incurs a lot of regions where GNSS availability probability is low. Without constraint, the mission leads to a collision in most episodes, for each test task. The best constraints returned correspond to pass to the left of the obstacles (Fig. 6). With MinPOMCP-GO or MinPOMCP-GO*, the number of collisions and the cost are decreased imposing the best constraint, particularly for the two maps presenting the highest GNSS availability probabilities, test tasks (3) and (4). For the third test task, the cost is decreased to almost 46% using MinPOMCP-GO*, and for the fourth task, it is reduced to almost 38% using MinPOMCP-GO.

		MinPOMCP-GO						*MinPOMCP-GO**					
		No constraint		**Constraint**		**Relative Gain (%)**		**No constraint**		**Constraint**		**Relative Gain (%)**	
		N_{col}	$Cost$	N_{col}	$Cost$	N_{col}	$Cost$	N_{col}	$Cost$	N_{col}	$Cost$	N_{col}	$Cost$
Cube Baffle	1	2	115.144	0	114.296	100.00	0.74	0	91.912	0	106.072	/	-15.41
	2	0	96.936	0	97.808	/	-0.90	0	93.592	0	95.144	/	-1.66
	3	1	104.360	0	105.688	100.00	-1.27	2	108.848	0	100.720	100.00	7.47
	4	1	102.648	0	98.384	100.00	4.19	0	94.824	0	99.368	/	-4.79
Wall Baffle	1	21	243.848	6	162.936	**71.43**	**33.18**	12	179.696	0	116.664	**100.00**	**35.08**
	2	14	191.600	3	141.616	**78.57**	**26.09**	9	152.024	0	116.720	**100.00**	**23.22**
	3	0	85.336	0	95.632	/	-12.07	0	84.816	0	95.768	/	-12.91
	4	3	110.976	2	102.616	33.33	7.53	1	95.216	0	87.280	100.00	8.33
San Diego	1	37	387.776	36	370.904	2.70	4.35	40	385.800	35	355.352	12.50	7.89
	2	39	381.368	31	347.800	**20.51**	8.80	27	305.408	23	278.600	14.81	8.78
	3	34	355.872	18	249.392	**47.06**	**29.92**	32	334.264	8	180.576	**75.00**	**45.98**
	4	27	305.496	9	189.824	**66.67**	**37.86**	28	311.352	11	199.136	**60.71**	**36.04**

TABLE 1 – Comparison of the performance metrics obtained by imposing the best constraint with the ones without constraint. The relative gains are computed as relative changes, taking the performance metric value obtained without constraint as reference. The considerably performance gains are presented in bold.

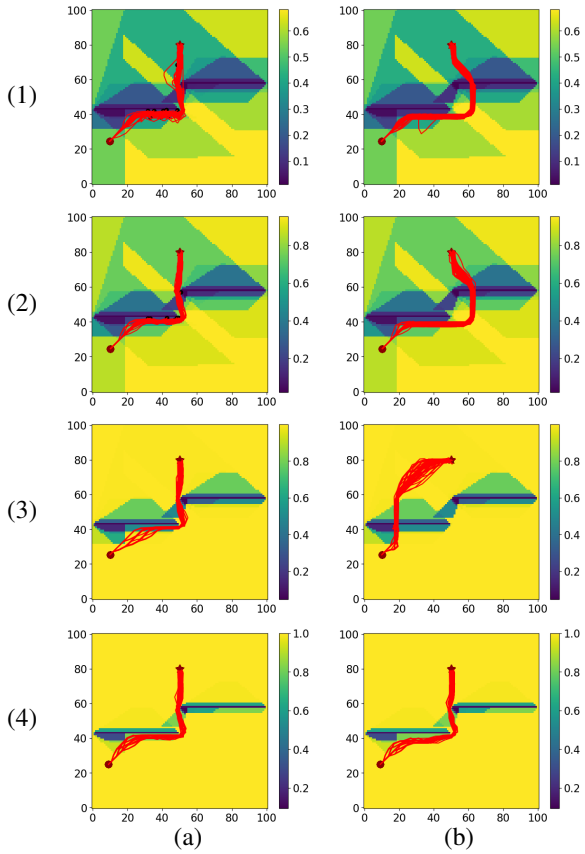


FIGURE 5 – Results obtained for *Wall Baffle* : paths obtained using MinPOMCP-GO*, without constraint (a), and with the best constraint (b)

In conclusion, for the three environments, imposing the best constraint always reduces the number of collisions, with any MinPOMCP-GO variant. This gain on the number of collisions and the one on the cost are much greater for difficult environments, comprising multiple obstacles and presenting low GNSS availability probabilities. Moreover, although the context selector is learnt from MinPOMCP-GO, imposing the best constraint that it returns improves clearly the performance of the algorithms even when using the online equivalent, MinPOMCP-GO*.

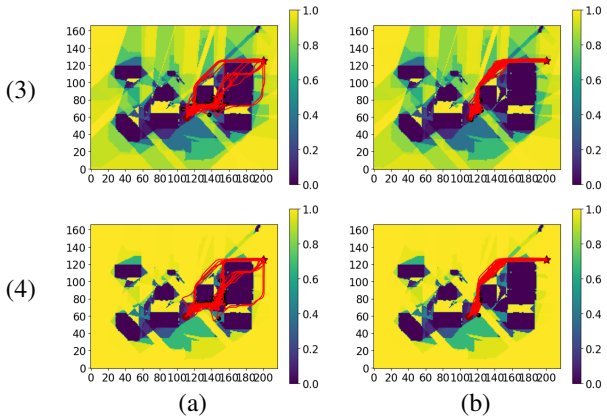


FIGURE 6 – Results obtained for *San Diego*, for test tasks (3) and (4) : paths obtained using MinPOMCP-GO*, without constraint (a), and with the best constraint (b)

5 Conclusion

In this paper, we have used a CAMP-inspired approach to address a partially observable problem of UAV autonomous navigation, where the GNSS availability may have a dramatic impact on the UAV path. We have then implemented a process to learn the best path constraint, *i.e.* the best corridor in which the UAV must navigate, from a set of GNSS availability probability maps. We have evaluated this approach on different environments, including a realistic urban one. The presented results have shown that first, learning these path constraints based on GNSS availability can indeed improve the quality of the computed paths, especially when uncertainty is high, and second, it has good performances on problems where only the state space is abstracted, and in situations where the constraint is learnt using one algorithm, and then used online with another algorithm.

Future works will generalize this approach by not only considering the GNSS availability map as feature, but also the initial and goal positions. To do so, we will avoid to evaluate all the possible constraints by only considering the most suitable candidate constraints, in order not to generate a huge number of training data. For example, in our navigation problem, only three constraints may be considered for each feature vector : the one corresponding to the shortest path, the one maximizing GNSS availability probability, and the one weighting the both of them.

Références

[Abel *et al.*2016] David Abel, D. Ellis Hershkowitz, and Michael L. Littman. Near optimal behavior via approximate state abstraction. In *International Conference on International Conference on Machine Learning (ICML)*, New York City, NY, USA, 2016.

[Anand *et al.*2015] Ankit Anand, Aditya Grover, Mausam Mausam, and Parag Singla. ASAP-UCT : Abstraction of State-Action Pairs in UCT. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Buenos Aires, Argentina, 2015.

[Anand *et al.*2016] Ankit Anand, Ritesh Noothigattu, Mausam, and Parag Singla. OGA-UCT : On-the-Go Abstractions in UCT. In *International Conference on Automated Planning and Scheduling (ICAPS)*, London, UK, 2016.

[Bakker *et al.*2005] Bram Bakker, Zoran Zivkovic, and Ben Kröse. Hierarchical dynamic programming for robot path planning. In *International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2005.

[Carmo *et al.*2020] Ana Raquel Carmo, Jean-Alexis Delamer, Yoko Watanabe, Rodrigo Ventura, and Caroline Ponzoni Carvalho Chanel. Entropy-based adaptive exploit-explore coefficient for Monte-Carlo path planning. In *International Conference on Prestigious Applications of Intelligent Systems (PAIS)*, (Digital ECAI), 2020.

[Chitnis *et al.*2021] Rohan Chitnis, Tom Silver, Beomjoon Kim, Leslie Kaelbling, and Tomas Lozano-Perez. CAMPs : Learning Context-Specific Abstractions for Efficient Planning in Factored MDPs. In *Conference on Robot Learning*, London, UK, 2021.

[Delamer *et al.*2021] Jean-Alexis Delamer, Yoko Watanabe, and Caroline Ponzoni Carvalho Chanel. Safe path planning for UAV urban operation under GNSS signal occlusion risk. *Robotics and Autonomous Systems*, 142 :103800, 2021.

[Dijkstra1959] Edsger W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1 :269–271, 1959.

[Gopalan *et al.*2017] Nakul Gopalan, Marie desJardins, Michael L. Littman, James MacGlashan, Shawn Squire, Stefanie Tellex, John Winder, and Lawson L. S. Wong. Planning with Abstract Markov Decision Processes. In *International Conference on Automated Planning and Scheduling (ICAPS)*, Pittsburgh, PA, USA, 2017.

[Hart *et al.*1968] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2) :100–107, 1968.

[Hostetler *et al.*2014] Jesse Hostetler, Alan Fern, and Tom Dietterich. State Aggregation in Monte Carlo Tree Search. In *AAAI Conference on Artificial Intelligence (AAAI)*, Québec City, QC, Canada, 2014.

[Jiang *et al.*2014] Nan Jiang, Satinder Singh, and Richard Lewis. Improving UCT Planning via Approximate Homomorphisms. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Paris, France, 2014.

[Kaelbling *et al.*1998] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101 :99–134, 1998.

[Keller and Helmert2013] Thomas Keller and Malte Helmert. Trial-Based Heuristic Tree Search for Finite Horizon MDPs. In *International Conference on Automated Planning and Scheduling (ICAPS)*, Rome, Italy, 2013.

[Kleijer *et al.*2009] Frank Kleijer, Dennis Odijk, and Edward Verbree. Prediction of GNSS Availability and Accuracy in Urban Environments Case Study Schiphol Airport. In *Location Based Services and TeleCartography II. Lecture Notes in Geoinformation and Cartography*. Springer, Berlin, Heidelberg, 2009.

[Kocsis and Szepesvári2006] Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-Carlo Planning. In *European Conference on Machine Learning (ECML)*, Berlin, Germany, 2006.

[Lee *et al.*2021] Yiyuan Lee, Panpan Cai, and David Hsu. MAGIC : Learning Macro-Actions for Online POMDP Planning. In *Robotics : Science & Systems (RSS)*, (Held Virtually), 2021.

[Li *et al.*2006] Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a Unified Theory of State Abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, Fort Lauderdale, FL, USA, 2006.

[Mettler *et al.*2010] Berenice Mettler, Zhaodan Kong, Chad Goerzen, and Matthew Whalley. Benchmarking of obstacle field navigation algorithms for autonomous helicopters. In *Forum of the American Helicopter Society (AHS)*, Phoenix, AZ, USA, 2010.

[Ong *et al.*2010] Sylvie C. W. Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under Uncertainty for Robotic Tasks with Mixed Observability. *The International Journal of Robotics Research*, 29(8) :1053–1068, 2010.

[Pineau *et al.*2006] Joelle Pineau, Geoffrey Gordon, and Sebastian Thrun. Anytime Point-Based Approximations for Large POMDPs. *Journal of Artificial Intelligence Research (JAIR)*, 27 :335–380, 2006.

[Silver and Veness2010] David Silver and Joel Veness. Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vancouver, BC, Canada, 2010.

Exploitation vs Caution: Bayesian Risk-sensitive Policies for Offline Learning

Giorgio Angelotti[1,2] Nicolas Drougard[1,2] Caroline P. C. Chanel[1,2]

[1] ISAE-SUPAERO, University of Toulouse, France
[2] ANITI, University of Toulouse, France

{name.surname}@isae-supaero.fr

Résumé

Lors de l'apprentissage hors-ligne d'un modèle tel que le Processus de Décision de Markov (MDP), la taille limitée dû jeu de données utilisé peut rendre l'estimation de la fonction de valeur du MDP sous-jacent une tâche difficile. Ceci est du à l'erreur inhérente des distributions apprises (ex. fonction de transition) avec cet ensemble limité de données, ce qui peut limiter la performance de la politique résultante dans le monde réel. Des travaux récents ont montré qu'une méthode de résolution hors-ligne d'un MDP appris qui utilise un facteur d'actualisation inférieur à celui utilisé pour l'évaluation produit des politiques plus performantes. Cependant, le calcul du facteur d'actualisation optimal est fait par validation croisée. Dans ce contexte, nous proposons la méthode Exploitation vs Caution (EvC): un algorithme qui sélectionne, hors-ligne, une politique sensible au risque pour un MDP Bayésien parmi un ensemble de politiques. Ces politiques sont obtenues en résolvant plusieurs MDP caractérisés par différents facteurs d'actualisation et dynamiques de transition. D'une part, le formalisme bayésien inclut élégamment l'incertitude du modèle et, d'autre part, l'introduction d'une fonction d'utilité sensible aux risques garantit la robustesse. Nous avons comparé EvC avec des approches de l'état-de-l'art dans différents environnements simples offrant une bonne variété de classes de MDP. Dans les scénarios testés, EvC est plus robuste que les algorithmes de l'état-de-l'art, suggérant que la résolution hors-ligne d'un MDP Bayésien et sensible aux risques, même que sous-optimale, pourrait définir un cadre de base pour la résolution de MDP sous incertitude de modèle.

Mots Clef

Planification hors ligne, Apprentissage de modèle hors ligne, Optimisation de politiques sensitive aux risques, MDP bayésien.

Abstract

In offline model learning for planning the limited data set hinders the estimate of the Value function of the relative Markov Decision Process (MDP), bounding the performance of the obtained policy in the real world. Recent works showed that planning with a discount factor lower than the one used for evaluation yields more performing policies. However, the optimal discount factor is finally chosen by cross-validation. Our aim is to show that looking for a sub-optimal solution of a Bayesian MDP might lead to better performances with respect to the current baselines. Hence, we propose Exploitation vs Caution (EvC), a proof-of-concept algorithm which automatically selects the policy that solves a Risk-sensitive Bayesian MDP (RBMDP) in the offline setting in a set of policies obtained by solving several MDPs characterized by different discount factors and transition dynamics. On one hand, the Bayesian formalism elegantly includes model uncertainty and, on another hand, the introduction of a risk-sensitive utility function guarantees robustness. We compared EvC with state-of-the-art approaches in different discrete simple environments offering a fair variety of MDP classes. In the tested scenarios EvC is more robust than the said approaches suggesting that sub-optimally solving a Risk-sensitive BMDP in the offline setting could define a sound baseline framework for planning under model uncertainty.

Keywords

Offline planning, Offline model learning, Risk-sensitive Policy Optimization, Bayesian Markov Decision Processes.

1 Introduction

The development of autonomous agents in an unknown, and possibly stochastic, environment is a delicate task that usually requires a continuous agent-environment interaction, which is not always affordable in real life situations. Indeed, in multiple applications

like the training of medical robots and automated vehicles Jonsson [2018], Mirchevska et al. [2018] the interaction with the environment can be both too risky and expensive since: 1) any mistake could lead to catastrophic aftermaths; or 2) the data collection phase requires a direct human involvement, which is not always available. Hence, it can be convenient to exploit previously collected data sets in order to limit additional (dangerous or superfluous) interaction.

Offline learning for planning is the branch of machine learning that leverages previously collected batches of experiences with the aim of establishing an optimal behavioural policy. In recent years, the RL community published a great number of papers on the subject Laroche et al. [2019], Fujimoto et al. [2019], Kumar et al. [2019], Wu et al. [2019], Lee et al. [2020], Chen and Jiang [2019], Yu et al. [2020], Kidambi et al. [2020], Levine et al. [2020], demonstrating the growing interest in the field. The proposed algorithms try to improve the performance of a policy obtained either with model-free or with model-based RL approaches. The intuition behind these methods is always the same: optimizing a trade-off between *exploitation and caution*. The policy optimization procedure is constrained in order to generate strategies that are not too distant from the one originally used to collect the batch. In this way, the agent will follow a strategy that will not drive him towards regions of the state-action space for which it possesses a high degree of uncertainty. More specifically, building upon the Markov Decision Process (MDP), such a constraint is implemented: (i) in Wu et al. [2019] as a penalty in the value function proportional to an estimate of the policies' distributional shift, (ii) in the MOPO algorithm Yu et al. [2020] as a penalization of the reward function proportional to an estimate of the distributional shift in the dynamical evolution of the system, also called *epistemic* (model) error, (iii) in the MOReL algorithm Kidambi et al. [2020] creating an additional and highly penalized absorbing state and makes the agent transit to it when the model uncertainty for a specific state-action pair is above a given threshold. The said baselines notably require the fine tuning of domain dependent hyperparameters. Such an empirical operation requires additional interaction with the environment and thus betrays the original purpose of offline learning. With the aim of obtaining a robust policy with a true hyperparameter agnostic approach the SPIBB methodology Laroche et al. [2019] and the BOPAH algorithm Lee et al. [2020] were developed. The former generates a PAC-style provably safe policy improvement over the data collecting policy, the latter uses a classic batch RL approach with a gradient-based optimization of the hyperparameters using held-out data.

Interestingly, Jiang et al. [2015] showed that planning using a discount factor γ^* lower than the one used in the final evaluation phase γ_{ev} yields more performing policies when a trivially learnt model is considered. A trivially learnt model is said to be the one which maximizes the likelihood of the transitions collected in the batch. Nevertheless, the mathematical expression that should be optimized in order to find γ^* is intractable. Therefore, the optimal discount factor is finally found by cross-validation which requires additional interaction with the true environment.

In parallel, with the aim of finding a policy that optimizes the trade-off between *exploitation and exploration* in an *online* setting, model uncertainty has been included in a Bayesian extension of the MDP framework called Bayesian (Adaptive) MDP (BAMDP) Strens [2000]. Fixing a prior for the distribution of transition models, a posterior distribution is computed from the likelihood of the sampled trajectories. Sharma et al. [2019] suggested that risk-sensitive utility functions can replace the common BAMDP value function. Doing so, the said work proposes an algorithm that trades off exploration, exploitation and robustness (caution).

Other works about solving MDPs under model uncertainty focused on the resolution of *robust MDPs* Nilim and El Ghaoui [2005], Iyengar [2005]. In this context the model dynamics lie in a constrained ambiguity set which is smaller than the whole space of distributions, and the problem is formulated as a dynamical game against a malevolent nature which at every time step chooses the worst model in the set according to the agent action. Subsequently, the offline solving of a Risk-sensitive Bayesian (non-adaptive) MDP (RBMDP) framework was partially introduced in Delage and Mannor [2010] under the name of *chance constrained MDP* and optimized for the *percentile criterion*: the Value-at-Risk (*VaR*) metric. Delage and Mannor [2010] proved that robust MDPs can generate overly conservative strategies depending on the size and shape of the ambiguity set. Petrik et al. [2016] proposed approximate solutions to generate safe policy improvements of the data collector policy. Laroche et al. [2019] built over the last approach and in the SPIBB algorithm reformulated the problem with PAC guarantees, obtaining better results. Recently, Petrik and Russel [2019], Behzadian et al. [2021] incorporated prior knowledge in a Bayesian fashion in order to obtain less conservative ambiguity sets that can yield tighter safe returns estimates. Lobo et al. [2021] introduced the *soft-robust criterion*, a weighted average between the classic value function and the Conditional *VaR* risk metric with epistemic model uncertainty, to solve a Robust MDP.

Note that when an MDP is learned from a fixed batch of precollected trajectories or when the posterior of a Bayesian MDP does not change with time, but it is

fixed and obtained by an already precollected batch of transitions, the solution to such a MDP/BMDP is said to be in the offline setting or offline.
This present work aims to show that solving an Risk-senstive BMDP in the offline setting considering different risk-metrics (VaR and also the Conditional *VaR*) could define the state-of-the-art standard of both optimization and evaluation for offline model learning for planning. Therefore, taking inspiration from Jiang et al. [2015] and Sharma et al. [2019], this study proposes Exploitation vs Caution (EvC): an algorithm which balances exploitation and robustness by finding a reasonable solution in the offline setting for a RBMDP while constraining the quest for the policy to a limited set. More specifically, the said strategies are the ones obtained by solving several MDPs with different discount factors and different transition functions which are sampled from the Bayesian posterior inferred from the fixed batch. In conclusion, EvC aims to be a proof-of-concept paradigm. While it does not computationally scale to solve large environments in its current version, it serves the purpose of supporting the claim that in the offline setting a Risk-sensitive BMDP is the right problem to be solved if the goal is to obtain a policy balancing exploitation and robustness. The contributions of this work are: (1) it proposes a proof-of-concept method to find risk-sensitive policies for BMDPs in the offline setting based on a simple yet efficient candidate policies selection strategy; (3) it gives theoretical guaranties on the accuracy of the computed risk-sensitive criteria, and (4) the empirical results demonstrates the validity of this approach in toy environments compared to state-of-the-art algorithms. The paper is organized as follows: The MDP and the Bayesian MDP (BMDP) formalisms are recalled. Then, risk-sensitive measures following the prescriptions of Majumdar and Pavone [2020] and the Risk-Sensitive BMDP are introduced. After, the EvC algorithm that obtains a risk-sensitive policy for a BMDP in the offline setting is proposed. Finally, EvC performance is compared against SPIBB, BOPAH and a modified version of MOPO and MOReL with the epistemic uncertainties given by an oracle. The last section concludes the paper discussing its limitations and pointing to future work perspectives.

2 Background

Definition 1. *A Markov Decision Process (MDP) is a 6-tuple* $M \stackrel{\text{def}}{=} \langle S, A, T, R, \gamma, \mu_0 \rangle$ *where* S *is the set of states,* A *the set of actions,* $T : A \times S \times S \to [0,1]$ *is the state transition function defining the probability that dictates the evolution from* $s \in S$ *to* $s' \in S$ *after taking the action* $a \in A$*,* $R : S \times A \to [R_{min}, R_{max}]$ *with* $R_{max}, R_{min} \in \mathbb{R}$ *is the expected reward function that indicates what the agent gains, on average, when the system state is* $s \in S$ *and action* $a \in A$ *is applied,* $\gamma \in [0,1)$ *is called the discount factor and* $\mu_0 : S \to [0,1]$ *is the distribution over the initial state.*

Definition 2. *A policy is a mapping from states to a probability distribution over actions, such as* $\pi : A \times S \to [0,1]$.

Definition 3. *Solving an MDP amounts to finding a policy* π^* *which,* $\forall s \in S$*, maximizes the value function:*

$$V_M^{\pi}(s) \stackrel{\text{def}}{=} \mathbb{E}_{\substack{a_t \sim \pi \\ s_t \sim T}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s \right]. \quad (1)$$

It has been proved that an MDP for which the value function is defined as Eq. (1) admits a deterministic optimal policy (a map from states to actions) Mausam and Kolobov [2012]:

$$\pi^*(s) = \text{argmax}_{\pi} V_M^{\pi}(s). \quad (2)$$

Definition 4. *The performance of a policy* π *in an MDP* M *with value function* V_M^{π} *is:*

$$U_M^{\pi} = \mathbb{E}_{s \sim \mu_0}\left[V_M^{\pi}(s)\right]. \quad (3)$$

Definition 5. *A BMDP is a 8-tuple* $\beta \stackrel{\text{def}}{=} \langle S, A, \mathcal{T}, \mathcal{R}, \tau, \rho, \gamma, \mathcal{B} \rangle$ *where* S *is the set of states,* A *the set of actions, and* $\gamma \in [0,1)$ *is a discount factor.* $\mathcal{T}$ *is a parametric family of transition functions of any MDP compatible with* S *and* A*,* $\mathcal{R}$ *is a parametric family of reward functions of any MDP compatible with* S *and* A*,* τ *is a non-informative prior distribution uniform over* $\mathcal{T}$*,* ρ *is a non-informative prior distribution uniform over* $\mathcal{R}$*,* $\mathcal{B} = \{(s_t, a_t, r_t, s_{t+1})\}$ *is a batch of transitions generated by acting in an unknown MDP with transition function* T*, reward* R *and initial state distribution* μ_0*.*

For instance, in a finite state and action spaces environment, $\mathcal{T}$ is the set of all $|S| \times |A|$ different categorical distributions and τ is made of $|S| \times |A|$ Dirichlet probability density functions – the conjugate prior of the said distribution.

Definition 6. τ_p *is a posterior distribution over* $\mathcal{T}$ *obtained by updating the prior* τ *with the information contained in* $\mathcal{B}$*.*

In particular, the $|S|$ probability values x describing $(s = s^*, a = a^*) \to s'$ can be distributed as:

$$\tau_p^{s^*,a^*}\left(x_1, \ldots, x_{|S|} \middle| n_1, \ldots, n_{|S|}\right) = \Gamma(\nu) \prod_{i=1}^{|S|} \frac{x_i^{n_i}}{\Gamma(n_i + 1)} \quad (4)$$

where, Γ is the Euler gamma function, n_i counts how many times the transition $(s^*, a^*) \to s_i$ appears in $\mathcal{B}$ and $\nu = \sum_{k=1}^{|S|}(n_k + 1)$.
Remark. Notice that the mode (the most likely configuration) of the posterior in Eq. (4) is given by $\hat{x}_i = \frac{n_i}{\sum_{k=1}^{|S|} n_k}$ while its expected value is $\mathbb{E}_{\tau_p}[X_i] = \frac{n_i+1}{\nu}$.

In discrete environments the most likely transition model with respect to τ_p is the one for which the transition probabilities are given by the transition frequencies in $\mathcal{B}$. We refer to these distributions also as $\hat{T}$ or as the trivial model. A similar reasoning can be done for $\mathcal{R}$ and ρ but for simplicity's sake we will assume to know the reward function R.

Remark. It would be possible to define a prior over the initial states and obtain a posterior taking into account the information contained in the batch $\mathcal{B}$. For simplicity we will also assume that μ_0 is known.

Definition 7. *A solution to a BMDP β is a policy which maximizes the following utility function:*

$$\mathcal{U}_\beta^\pi \stackrel{\text{def}}{=} \mathbb{E}_{M\sim\tau_p}\left[U_M^\pi\right] \tag{5}$$

where, $U_M^\pi \stackrel{\text{def}}{=} \mathbb{E}_{s\sim\mu_0}\left[V_M^\pi(s)\right]$ is the value function of an MDP , averaged on the initial state, with transition function sampled from τ_p.

Remark. Since the true MDP is unknown, leveraging the Bayesian framework is an elegant way to incorporate uncertainty. However, the additional expected value makes Eq. (5) hard to be computed with Bellman's recursive approaches or approximated with temporal differences methods.

The optimal performance with respect to Eq. (5) will be the one that, on average, works the best on the BMDP β when the model is distributed according to the Bayesian posterior:

$$U_\beta^* = \max_\pi \mathcal{U}_\beta^\pi \tag{6}$$

3 Risk-sensitive measures

Solving a BMDP deals with epistemic uncertainty more elegantly than solving an MDP for the most likely model $\hat{M} = (\hat{T}, \hat{R})$, but the utility function defined in Eq. (5) does not minimize the risk of obtaining a bad performing policy in the real environment. E.g. let $p\left(U_M^\pi = u|\mathcal{B}\right)$ be the probability density function (pdf) of the performance of the policy π when the model is distributed according to the Bayesian posteriors. Given two policies π_0 and π_1 we can have that $\mathbb{E}\left[U_M^{\pi_0}\right] > \mathbb{E}\left[U_M^{\pi_1}\right]$ (Figure 1). In this case, following the BMDP optimization criterion, π_0 is better than π_1. However, fixing a value u less than both expected values, it can happen that $p\left(U_M^{\pi_0} < u|\mathcal{B}\right) > p\left(U_M^{\pi_1} < u|\mathcal{B}\right)$. With this in mind, it can be useful to define risk-sensitive utility functions. Risk measures are widely studied in mathematical finance Artzner et al. [1999] since they are a way to rationally quantify risk. Their application to MDPs under model uncertainty was investigated in Majumdar and Pavone [2020]. We now introduce two risk measures, the Value at Risk (VaR) and the Conditional Value at Risk ($CVaR$).

Definition 8 (Value at Risk)**.** *Let X be a continuous random variable with values in $[A, B]$ distributed according to a pdf $p : [A, B] \to [0, +\infty[$ and let $q \in [0, 1]$, then VaR_q is the solution to the equation $\int_A^x dx' p(x') = q$.*

Definition 9 (Conditional Value at Risk)**.** *Let X be a continuous random variable with values in $[A, B]$ distributed according to a pdf $p : [A, B] \to [0, +\infty[$ and let $q \in [0, 1]$, then $CVaR_q = \mathbb{E}_p\left[X|X \leq VaR_q\right]$.*

Strictly speaking, the VaR_q is the q-quantile of the distribution, while the $CVaR_q$ is the expected value of the distribution up to the q-quantile.

3.1 Risk-sensitive Solutions to BMDPs

In the following, the BMDP utility of Eq. (5) taking also the risk into account is generalized.

Definition 10. *Let β be a BMDP and let $V_M^\pi(s)$ be the value function at state s while following a policy π in the MDP M with transitions distributed according to the posterior. Let also $p\left(U_M^\pi|\mathcal{B}\right)$ be the pdf over the possible values assumed by $U_M^\pi = \mathbb{E}_{s\sim\mu_0}\left[V_M^\pi(s)\right]$. Then a risk-sensitive utility function is defined as:*

$$\mathcal{U}_{\beta,\sigma}^\pi \stackrel{\text{def}}{=} \sigma_{M\sim\tau_p}\left[U_M^\pi\right] \tag{7}$$

where, σ is a risk measure.

Remark. As a consequence of Definition 9 if $\sigma = CVaR_1$, considering that $CVaR_1 = \mathbb{E}_p[X]$, then the BMDP utility of Eq. (5) is a particular case of (7).

4 Solving offline a risk-sensitive BMDP

The expectation over the distribution of models makes the resolution of a BMDP a computational task which is often intractable. Moreover, a Risk-sensitive BMDP also presents an additional difficulty: the risk measure requires an estimate of the quantiles of the unknown value distribution. An analytical maximization of the performance defined in (Eq. 7) is often either impossible or too computationally demanding. In order to tackle the maximization problem, a valuable choice is resorting to a Monte Carlo estimate of the performance. We will then look for a sub-optimal policy, rather than an optimal one, by constraining the search to a set of policies Π. What number $L_\pi \in \mathbb{N}$ of models would be necessary to sample in order to have an accurate estimate of the performance of a policy within a chosen confidence interval? Ideally, L_π should be as small as possible because Policy Evaluation has to be performed L_π times in order to obtain the Bayesian posterior distribution of values assumed by the Value Functions. Providentially, this problem has been addressed by Briggs and Ying [2018] where a procedure that allows to iteratively sample values from

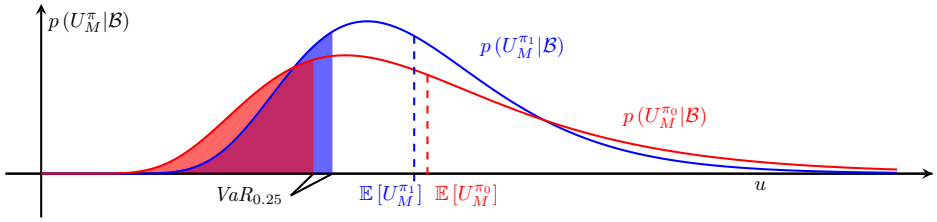


Figure 1: Pdf for the performance U_M^{π} given the batch $\mathcal{B}$ and two different policies π_0 and π_1. Dashed lines correspond to the expected value. Both curves are filled up to their 0.25-quantiles. The red curve has a higher expected value while the blue one corresponds to a safer policy.

a distribution whose quantile is required until the estimate of the quantile will fall within an interval with a required probabilistic significance. We exploit this idea to propose a Monte Carlo Confident Policy Selection (MC2PS) algorithm, which is presented in Algorithm 1, to identify a robust policy for a BMDP in the offline setting among a set of candidate policies. For a given set of policies Π and for every policy $\pi \in \Pi$, the algorithm incrementally samples k transition models from τ_p and performs Policy Evaluation in parallel for each one of them until the stopping criterion is reached (see the RiskEvaluation procedure call in Alg. 1). The stopping criterion guarantees that the estimate of the q-quantile is statistically well approximated with a significance level α within a dynamically sampled confidence interval whose width is proportional to ε_{rel} (lines 19-22) given the total L_π models sampled. Then it leverages the estimate of both the quantile and of the sampled values to obtain an estimate of the utility function $\mathcal{U}_{\beta,\sigma}^{\pi}$ for a specific risk measure σ: *VaR* or *CVaR* (lines 23-26). Finally, when the utility function has been estimated for every policy, it outputs the one that maximizes it (line 4).

Remark. Let Λ be the total number of models sampled to estimate the quantile of the performance distribution among policies: $\Lambda = \sum_{\pi\in\Pi} L_\pi$. MC2PS performs Policy Evaluation Λ times. The size of the space of all applicable policies of a finite state and action space MDP is $|\Pi| = |A|^{|S|}$. It goes without saying that looking over the whole policy space can be practically intractable even for not so big MDPs. Nevertheless, restricting the research to a subset of policies could be a viable solution also for big MDPs, also considering that the Policy Evaluations are carried out in parallel.

4.1 Exploitation vs Caution (EvC)

Reference Jiang et al. [2015] shows that the policy obtained by solving an MDP $\hat{M} = (S, A, \hat{T}, R, \gamma^*)$, trivially learnt from a batch of experiences collected from another MDP $M = (S, A, T, R, \gamma_{ev})$, with γ^* a discount factor such as $\gamma^* \leq \gamma_{ev}$, is more efficient in M than the policy obtained by solving $\hat{M}$ using γ_{ev}. The reason is that $\hat{T}$ is an approximation of T and may not be trusted for the longest planning horizon. The selection of the best γ^* optimizes a trade-off between the exploitation of the information contained in the batch and the necessity of being cautious since the model estimate is not perfect. Inspired by Jiang et al. [2015]'s conclusions, and also guided by the intuition

Algorithm 1 MC2PS

Input: set of policies Π, significance level $\alpha \in [0,1]$, sampling batch size $k \in \mathbb{N}$, relative error tolerance $\varepsilon_{rel} \in [0,1]$, posterior distribution τ_p, quantile order $q \in [0,1]$, risk measure σ, initial state distribution μ_0.
Output: best policy π^*.

1: **for** $\pi \in \Pi$ **do**
2: $\quad \mathcal{U}_{\beta,\sigma}^{\pi} \leftarrow$ RiskEvaluation $(\pi, \sigma, \tau_p, \mu_0, \varepsilon_{rel}, \alpha, q, k)$
3: **end for**
4: **return** $\pi^* = \arg\max_{\pi\in\Pi} \mathcal{U}_{\beta,\sigma}^{\pi}$
5:
6: **procedure** RiskEvaluation
7: $\quad$ **Input**: policy π, risk measure σ, posterior distribution τ_p, initial state distribution μ_0, relative error threshold $\varepsilon_{rel} \in [0,1]$, significance level $\alpha \in [0,1]$, quantile order $q \in [0,1]$, sampling batch size $k \in \mathbb{N}$.
8: $\quad$ Initialize $U^{\pi} = \varnothing$
9: $\quad$ *(loop estimates the quantile needed in Eq. 7)*
10: $\quad$ **repeat**
11: $\quad\quad$ **for** $j \in \{1, \ldots, k\}$ **in parallel do**
12: $\quad\quad\quad$ Sample $\mathcal{M}_j \sim \tau_p$
13: $\quad\quad\quad$ $V_{\mathcal{M}_j}^{\pi}(s) \leftarrow$ Policy Eval. on model $\mathcal{M}_j$
14: $\quad\quad\quad$ $U_{\mathcal{M}_j}^{\pi} \leftarrow \mathbb{E}_{s\sim\mu_0}[V_{\mathcal{M}_j}^{\pi}(s)]$ (Equation 3)
15: $\quad\quad\quad$ $U^{\pi} \leftarrow$ **append** $U_{\mathcal{M}_j}^{\pi}$
16: $\quad\quad$ **end for**
17: $\quad\quad$ $L_\pi \leftarrow |U^{\pi}|$
18: $\quad\quad$ Sort U^{π} in increasing order
19: $\quad\quad$ Find $(s, r) \in \mathbb{N}^2$ such that $|s - r|$ is min and:
20: $\quad\quad\quad$ $\sum_{i=r}^{s-1} \binom{L_\pi}{i} q^i (1-q)^{L_\pi - i} > 1 - \alpha$;
21: $\quad$ **until** :
22: $\quad\quad$ $U_s^{\pi} - U_r^{\pi} < \varepsilon_{rel} \cdot \left(U_{L_\pi}^{\pi} - U_1^{\pi}\right)$
23: $\quad$ **if** $\sigma = VaR_q$ **then**
24: $\quad\quad$ **return** $U_{\lfloor qL_\pi \rfloor}^{\pi}$
25: $\quad$ **else if** $\sigma = CVaR_q$ **then**
26: $\quad\quad$ **return** mean$\left(U_1^{\pi}, \ldots, U_{\lfloor qL_\pi \rfloor}^{\pi}\right)$
27: **end procedure**

that the true model M will be different from $\hat{M}$, but close to it, we hope that the policies obtained by solving an MDP $\tilde{M} = (S, A, \tilde{T}, R, \gamma)$ with $\tilde{T}$ *close* to $\hat{T}$ and $\gamma \leq \gamma_{ev}$ can be viable solutions for the RBMDP in the offline setting.

Henceforth, the Exploitation *versus* Caution (EvC) algorithm is presented and schematized in Algorithm 2. EvC will search for a promising risk-sensitive policy by focusing the search in a set of candidate policies Π computed with different MDPs $\tilde{M}$ and $\gamma \leq \gamma_{ev}$ values. As already stated, we do not aim to find the optimal solution to the Bayesian MDP, but rather, to find a policy that is more robust than the one that we would obtain by optimally solving the trivial MDP. In detail, EvC first generates candidate

Algorithm 2 EvC

Input: quantile order $q \in [0,1]$, significance level $\alpha \in [0,1]$, sampling batch size $k \in \mathbb{N}$, relative error tolerance $\varepsilon_{rel} \in [0,1]$, posterior distribution τ_p, risk measure σ, initial state distribution μ_0, set of discount factors G, number of models to solve $l \in \mathbb{N}$.

Output: best policy π^*.

1: $\Pi \leftarrow$ GeneratePolicies (τ_p, G, l)
2: **return** $\pi^* =$ MC2PS $(q, \alpha, k, \varepsilon_{rel}, \tau_p, \Pi, \sigma, \mu_0)$
3:
4: **procedure** GeneratePolicies
5: **Input**: posterior distribution τ_p, set of discount factors G, number of models to solve $l \in \mathbb{N}$.
6: Initialize $\mathbb{M} = \{l$ transition models $\sim \tau_p\} \cup \{\hat{T}\}$
7: Initialize $\Pi = \varnothing$ (an empty set)
8: **for** $(\gamma \in G, T \in \mathbb{M})$ **do**
9: $\pi_{(T,\gamma)} =$ solution to the MDP with T and γ
10: Append $\pi_{(T,\gamma)}$ to Π if $\pi_{(T,\gamma)} \notin \Pi$
11: **end for**
12: **return** Π
13: **end procedure**

policies that will constitute the set Π (line 1 calls GeneratePolicies procedure). For this, the trivial MDP $\hat{M}$ and l additional MDPs are sampled from the Bayesian posterior τ_p obtained from the batch (line 6), and then solved with different values of $\{\gamma \in G | \gamma \leq \gamma_{ev}\}$ (lines 8-11). Recalling that γ_{ev} is the discount factor of the RBMDP. Note that the obtained set Π has unrepeated solutions (line 10). As a last step, MC2PS is launched with the obtained set of candidate policies Π returning the best risk-sensitive solution $\pi^* \in \Pi$ (line 2).

Remark. EvC does not computationally scale to solve large environments in its current version. Note, if we test over 9 different discount factors, such as $G = \{0.1, 0.2, \ldots, \gamma_{ev} = 0.9\}$, and 5 different $(l = 5)$ MDPs $\tilde{M}$ (including $\hat{M}$), then we solve $|\Pi| \leq 9l = 45$ MDPs to obtain the set of candidate policies. Yet, EvC serves the purpose of suggesting a (sub-optimal) policy balancing exploitation and robustness in such RBMDP framework with a fixed batch.

4.2 Theoretical guarantees

Since EvC searches for the policy $\pi \in \Pi$ that maximizes the criterion of Eq. (7), the Algorithm 2 yields a sub-optimal solution to the RBMDP. Additionally, the lack of an exact analytical expression for Eq. (7), and the subsequent Monte Carlo estimate of the quantile makes the theoretical guarantee of the EvC paradigm loose. In any case, assuming that the Bayesian posterior τ_p efficiently encodes the model uncertainty, EvC outputs a policy whose performance in the real environment is guaranteed in probability to be greater than some value that changes with respect to the chosen risk-sensitive measure. We expect that for a sufficiently big batch $\mathcal{B}$ the optimal trivial policy should be more performing than the one obtained through EvC since the most-likely model should converge to real one.

Theorem 1. *Let π^* be the policy obtained by EvC and $U^{\pi^*}_{\lfloor qL_{\pi^*} \rfloor}$ be its q-quantile calculated through EvC. Let $U^{\pi^*}_M$ be the performance of π^* with M distributed according to the Bayesian posterior τ_p. The performance of π^* in this MDP M is greater than its q-quantile with probability $Pr(U^{\pi^*}_M \geq U^{\pi^*}_{\lfloor qL_{\pi^*} \rfloor} - \varepsilon^{\pi^*}) \geq (1-q)(1-\alpha)$, where $\varepsilon^{\pi^*} \overset{\text{def}}{=} \varepsilon_{rel} \cdot (U^{\pi^*}_{L_{\pi^*}} - U^{\pi^*}_1)$ as defined in Algorithm 1.*

Proof. Note that $\{U^{\pi^*}_M > \text{VaR}^{\pi^*}_q\} \cap \{\text{VaR}^{\pi^*}_q \geq U^{\pi^*}_{\lfloor qL_{\pi^*} \rfloor} - \varepsilon^{\pi^*}\} \subseteq \{U^{\pi^*}_M \geq U^{\pi^*}_{\lfloor qL_{\pi^*} \rfloor} - \varepsilon^{\pi^*}\}$, where $\text{VaR}^{\pi^*}_q$ denotes the theoretical q-quantile. The events of the intersection depends on independent random variables – a future performance $U^{\pi^*}_M$ and a quantile estimation $U^{\pi}_{\lfloor qL_{\pi^*} \rfloor}$ – which allows to write $Pr\big(U^{\pi^*}_M \geq U^{\pi^*}_{\lfloor qL_{\pi^*} \rfloor} - \varepsilon^{\pi^*}\big) \geq Pr\big(U^{\pi^*}_M > \text{VaR}^{\pi^*}_q\big) \cdot Pr\big(\text{VaR}^{\pi^*}_q \geq U^{\pi^*}_{\lfloor qL_{\pi^*} \rfloor} - \varepsilon^{\pi^*}\big) \geq (1-q) \cdot Pr\big(\varepsilon^{\pi^*} \geq |\text{VaR}^{\pi^*}_q - U^{\pi^*}_{\lfloor qL_{\pi^*} \rfloor}|\big) \geq (1-q)(1-\alpha)$. The last inequality is ensured by the quantile estimation (lines 19-22 in Algorithm 1), and the previous one by the definition of VaR_q. □

Remark. When the risk-sensitive measure used in EvC is VaR_q the lower bound on $U^{\pi^*}_M$ in the Proof of Theorem 1 is maximized. If the risk-sensitive measure is CVaR_q the empirical expected value over the q-fraction of low performing policies is maximized.

4.3 Consequences and applications

The purpose of Offline Learning is that of providing behavioural policies to be applied by real-world automated agents. Thus reducing the risk at the expense of a longer computational phase is not only commendable but compulsory. Will the policy obtained through MC2PS and EvC be *good* or entirely-risk free? This goes beyond the theoretical guarantees provided by the algorithms since its outputs depend not only on the characteristics of the environment and on the

set of candidate policies but also on the quality and variety of the batch. A batch of transitions that is too small or too concentrated in the same region of the state-action space may result in policies that, even if they are guaranteed to handle the risk better than the trivial one, can still be catastrophic.

5 Experiments

We selected three small and hence easy to study stochastic environments with different characteristics: two planning environments without absorbing states, Ring (5 states, 3 actions) and Chain (5 states, 2 actions), the former consisting in the stabilization of the agent in a particular non-absorbing goal with stochastic drift and the latter presenting cycles; and the Random Frozen Lake (RFL) environment, a re-adaptation of Frozen Lake from Open AI Gym suite Brockman et al. [2016] (8×8 grid world with fatal absorbing states). A description of the environments is provided in the Supplementary Material.

5.1 Setup

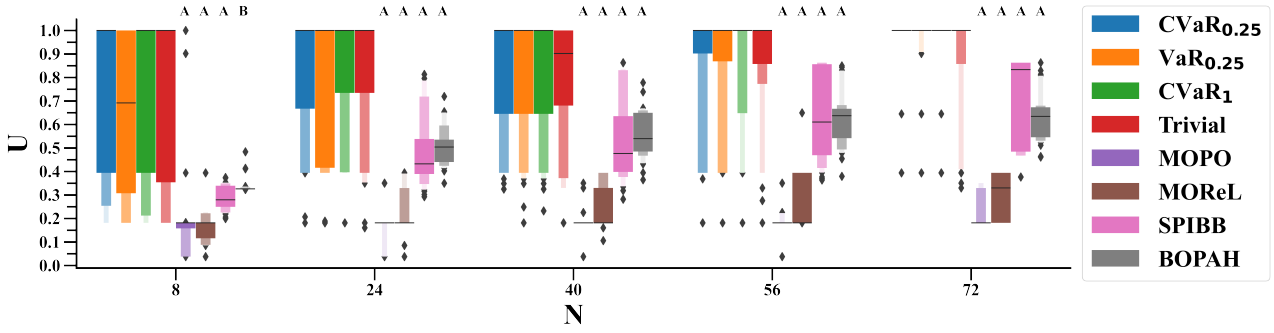
Given $(n, m) \in \mathbf{N}^2$, m trajectories with n steps each are generated following a random policy in each environment. We opted for a random data collecting procedure because we imagine using EvC in a scenario where both the developers and the autonomous agent are completely agnostic about the model dynamics and have no prior knowledge. The true environment is assumed to be known for the a-posteriori evaluation. The most likely transition model is inferred from the batch. The trivial MDP was then solved with the Policy Iteration algorithm and its relative performance in the true environment is obtained by Policy Evaluation. EvC data was computed with $CVaR_{0.25}$, $VaR_{0.25}$ (the first quartile) and $CVaR_1$. For each of these risk-sensitive measures, the following parameters (see Algorithm 2) were used: the set of discount factors $G = \{0.2, 0.4, 0.6, 0.8, 0.9\}$, the significance level $\alpha = 0.01$, the relative tolerance error $\varepsilon_{rel} = 0.01$, and the number of different models sampled from the prior l is given in Table 1. In the experiments, for a given batch size $N \in \mathbf{N}^2$, 50 different batches were generated containing fixed size trajectories. The trajectory sizes used are given in the Table 1. In the evaluation phase, the discount factor is defined as $\gamma_{\text{ev}} = 0.9$. For comparison purposes, we computed the performances of SPIBB, BOPAH, MOPO and MOReL, four state-of-the-art approaches that learn and solve MDPs in the offline setting.In our implementation of these baselines we only used intuitively tunable parameters (e.g. the discount factor) while the uncertainty dependent term is given by an oracle. Thus, EvC is compared with the best possible applications of MOPO and MOReL that are directly applicable without domain-dependent hyperparameter fine tuning. More specifically: (1) In our implementation of MOPO, the penalized MDP is built starting from the trivial MDP with the reward decreased by $V_{max}\frac{\gamma_{ev}}{1-\gamma_{ev}}D_{TV}[T(\cdot|a,s), \hat{T}(\cdot|a,s)]$ where V_{max} is the maximum of the optimal Value Function of the true MDP and $D_{TV}[.]$ is the total variation distance between the transitions of the trivial and the true MDP. This is the best bound that benefits from theoretical guarantees. In the MOPO's original implementation the distributional shift is estimated from the batch and the coefficient is a hyperparameter that has to be tuned. Moreover, since in Yu et al. [2020] a roll-out horizon is introduced as an additional hyperparameter, we will take this aspect into account by solving the penalized MDP with several values of γ, corresponding to different planning horizons; (2) In our implementation of MOReL, we keep the penalization κ fixed. The uncertainty is given by $D_{TV}[T(\cdot|a,s), \hat{T}(\cdot|a,s)]$ and the threshold α_{MOReL} varies between the simulations; (3) SPIBB and BOPAH receive as input the batch collecting policy. Simulation parameters are provided in Table 1.

5.2 Results and discussion

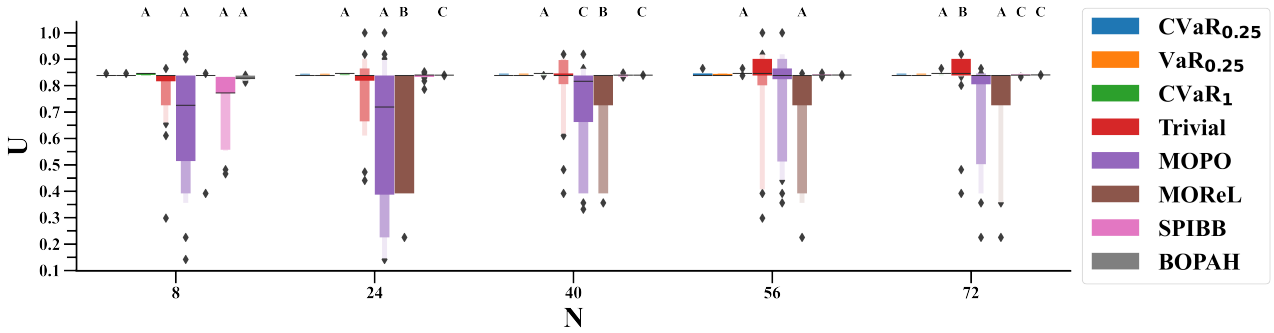
We plot the performances $U^{\pi}_{\beta,\sigma}$ of the policies π obtained with a specific algorithm (Eq. 3 using the utility function defined in Eq. 7) normalized by the optimal one in a Letter-Value fashion Heike et al. [2017]. The results are shown in Figure 2. The N on the x-axis is the total number of transitions in each batch ($N = nm$). Statistics are performed over 50 batches for each N. In the case of RFL, we generated 7 different environments. We then plot the aggregate results where the statistics are performed over 350 batches for each N. The median of the distribution is displayed as a horizontal black line. Different quantiles are plotted as boxes of different sizes and shades, becoming smaller and more transparent the more extremal the quantile is. Outliers are shown as diamonds. In this way we keep track: (i) of the typical behaviour of an algorithm; (ii) of the whole distribution over the analyzed batches; and, (iii) of the minimal performance achieved over the said batches, hence visualizing how well each algorithm handles the risk. Notice that in the original papers of SPIBB and BOPAH the $CVaR$ over several batches is just used as an evaluation metric, while in EvC it can define the risk-sensitive utility function according to model uncertainty (Eq. 7). In any tested scenario, the policies obtained by the baselines perform almost always worse than the one computed using the trivial MDP with the exception of BOPAH in Chain that is as performing as the trivial approach. The high variance in the results is caused by the size and variety of the starting batch (and in RFL also by the difference in the generated maps). Taking into account that the distributional shift was oracle given to MOPO and MOReL these results scale back the validity of the

Table 1: Parameters and hyperparameters used during the simulations: n is the number of steps in each trajectory contained in a batch; l is the number of different models sampled from the prior in EvC (Algorithm 2); $\{\alpha_{\text{MOReL}}\}$ is the set of different thresholds tested with MOReL; κ is the absolute value of the penalization of the absorbing state in MOReL; $\{\gamma_{\text{MOPO}}\}$ is the set of different discount factors tested with MOPO; $\{N_\wedge\}$ is the set of different thresholds used to test SPIBB; **fold** and **DOF** are the fold and degree of freedom hyper-parameters used to test BOPAH. Bold values are displayed in the plots.

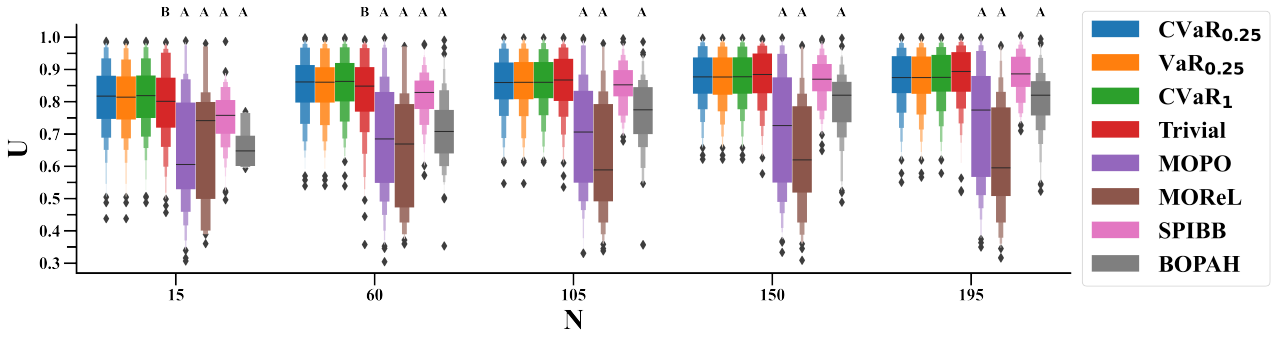
Environment	n	l	$\{\alpha_{\text{MOReL}}\}$	κ	$\{\gamma_{\text{MOPO}}\}$	$\{N_\wedge\}$	**fold**	**DOF**
Ring	8	3	$\{0.1, 0.2, 0.3, 0.4, \mathbf{0.5}\}$	100	$\{0.2, \mathbf{0.4}, 0.6, 0.8, 0.9\}$	$\{\mathbf{1}, 2, 3, 5, 7, 10, 20\}$	2	20
Chain	8	3	$\{0.1, 0.2, 0.3, 0.4, \mathbf{0.5}\}$	100	$\{\mathbf{0.2}, 0.4, 0.6, 0.8, 0.9\}$	$\{\mathbf{1}, 2, 3, 5, 7, 10, 20\}$	2	20
RFL (8×8)	15	10	$\{0.1, 0.2, 0.3, 0.4, \mathbf{0.5}\}$	100	$\{\mathbf{0.2}, 0.4, 0.6, 0.8, 0.9\}$	$\{\mathbf{1}, 2, 3, 5, 7, 10, 20\}$	2	20



(a) Normalized performance for the Ring Environment.



(b) Normalized performance for the Chain Environment.



(c) Normalized performance for the aggregate of 7 different RFL 8x8.

Figure 2: Letter-Value plot of the normalized performance U of EvC with $CVaR_{0.25}$, $VaR_{0.25}$, $CVaR_1$ compared to state-of-the-art approaches. N is the total number of transitions in a batch. Statistics are performed over 50 batches (350 in the case of RFL) for each N. The median is a horizontal line, the largest box contains the central 50% of data, thus limited by the first and third quartiles, the other boxes above and below are delimited by the first and the seventh octiles, etc. Diamonds are outliers. The significance of the Wilcoxon post hoc test between $CVaR_{0.25}$ and the other baselines is displayed on top of each box: a p-value $p < 0.001$ is denoted by A, $p < 0.01$ is denoted by B, and $p < 0.05$ by C.

said methods in the considered environments. In fact, even though the theoretical guarantees of MOReL and MOPO work in any MDP, it has to be said that (1) they highly rely on hyperparameter domain-dependent fine tuning which we did not do to fulfill the offline learning obligation; (2) they have usually been tested on continuous state MDPs driven by a deterministic dynamics, while here we are tackling non-deterministic environments. SPIBB and BOPAH yield policies that result in performance improvement over the batch collecting strategy, unfortunately they are still not better than the trivial MDP optimal policy.

The empirical results are validated by a non-parametric statistical analysis. A Friedman test of differences among repeated measures was conducted for every batch size N and rendered F values ≥ 20 which were significant ($p < 0.001$). The significance of the test points out that there is at least one algorithm that outputs a distribution of performances that very significantly differs from the others. Two post hoc pairwise tests were performed to assess which baselines were significantly different: a multi comparison Conover test and a Wilcoxon signed-rank test, both with Bonferroni-Holm correction for the p value. Sometimes the differences between two paradigms were not significant ($p > 0.05$) with respect to the Conover test but they were significant for the Wilcoxon one. Indeed, a multi comparison mean-ranks test as the Conover one can lead to different results when evaluating two groups depending on the samples in the other groups, hence in this case a test that depends only on the pair of approaches considered and not on all the groups as the Wilcoxon one is more realiable Benavoli et al. [2016]. On average, there is not a statistically significant difference between the several risk-metrics used in EvC. $CVaR_1$ seems to be overall the most performing one, abiding by the non written rule of *optimism in face of uncertainty*. Still, $CVaR_1$ does not provide the same risk management probabilistic guarantees of $VaR_{0.25}$ and $CVaR_{0.25}$ because it maximizes the expected value over the whole distribution (see Eq. 7) without focusing on the worst case scenarios. With this in mind on top of each box in Figure 2 we display the signficance of the pairwise comparison with the Wilcoxon post hoc test using the following legend: A ($p < 0.001$), B ($p < 0.01$), C ($p < 0.05$). Both in Ring and in Chain (Figures 2a, 2b) the median is greater than the one of the trivial optimal policy. However, while the said results are statistically significant with $p < 0.05$ in Chain, this difference in Ring is not statistically significant according to both the Conover and the Wilcoxon tests. In RFL 8×8 (Figure 2c) the EvC policies yield the highest average rewards for a batch made of $N = 15$ transitions and keep handling the risk better than the other baselines up to $N = 150$. BOPAH yield competitive robust policies but perform worse on average. SPIBB is less performing for $N = 15$, but for larger batches the difference with EvC becomes not statistically significant. Both MOPO and MOReL are outperformed by EvC and by the other paradigms in this scenario. For the smallest batch sizes there is a significant difference (Wilcoxon's $p < 0.001$) between all the tested EvC methods and both the Trivial policy and the state-of-the-art approaches. The minimum value of the normalized performance of the EvC methods is almost always higher than all the other ones. This makes EvC the most robust technique between the tested ones. However, in environments characterized by fatal absorbing states, like RFL, using EvC over the trivial optimal policy is not as effective as resorting to EvC in the other cases. We speculate that the information contained in a batch, with trajectories ending in a penalized absorbing state, is already enough to obtain a trivial model that, when solved, results in a good policy. The major drawbacks of EvC are the heavy computational demands and the inefficient scalability to high dimensional environments. Moreover, it is worth recalling that the quality of EvC results is limited by the quality of the candidate policy set Π and that using the policies that solve the trivial MDPs with different planning horizons is just a simple yet effective choice. The application of EvC to real world problems is safer than directly following the trivial optimal policy in terms of the considered risk measure. Yet, whether or not EvC can be used to obtain solutions that are also good and not only safer than the trivial one depends on the set of candidate policies.

6 Conclusion and perspectives

With the aim of achieving a promising solution in the offline setting for Risk-sensitive BMDPs, we developed EvC. EvC finds a solution policy that maximizes the risk-sensitive utility function of Eq. (7) over a set of candidate policies. This set contains the strategies obtained by solving the trivially learnt MDP and other MDPs with transition dynamics sampled from the Bayesian posterior (e.g. the one shown in Eq. 4) using different discount factors. The risk computation in the presented algorithm provides a probabilistic guarantee for the actual performance of the resulting policy described in Theorem 1. Moreover, if the Bayesian posterior efficiently encodes the uncertainty about the true models, then EvC guarantees to output a policy that performs in probability better than (or equal to) the trivial optimal one. Even if the EvC approach was tested in toy environments, it yielded sub-optimal policies that are more robust than the ones obtained with the other methods. Nevertheless, since EvC is based on the parallel resolution of a great number of models sampled from the Bayesian posterior we doubt that it could efficiently scale to solve MDPs with a

great number of states and actions. Anyway, in the future we aim to improve EvC's method of generation of the set of candidate policies. An interesting direction consists in incrementally enriching the set of candidate policies following a heuristic. Another promising line of research concerns the development of scalable algorithms, using function approximators for instance, that can efficiently solve large RBMDPs in the offline setting.

*. References

Philippe Artzner, Freddy Delbaen, Eber Jean-Marc, and David Heath. Coherent Measures of Risk. *Mathematical Finance*, 9:203 – 228, 07 1999. doi: 10.1111/1467-9965.00068.

Bahram Behzadian, Reazul Hasan Russel, Marek Petrik, and Chin Pang Ho. Optimizing Percentile Criterion using Robust MDPs. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1009–1017. PMLR, 13–15 Apr 2021.

Alessio Benavoli, Giorgio Corani, and Francesca Mangili. Should we really use post-hoc tests based on mean-ranks? *The Journal of Machine Learning Research*, 17(1):152–161, 2016.

K. Briggs and F.M. Ying. How to estimate quantiles easily and reliably. *Mathematics Today*, 2018 (February):26–29, 2018.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

Jinglin Chen and Nan Jiang. Information-Theoretic Considerations in Batch Reinforcement Learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of Machine Learning Research*, volume 97, pages 1042–1051, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Erick Delage and Shie Mannor. Percentile optimization for markov decision processes with parameter uncertainty. *Operations research*, 58(1):203–213, 2010.

Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. Benchmarking Batch Deep Reinforcement Learning Algorithms, 2019.

Hofmann Heike, Hadley Wickham, and Karen Kafadar. Letter-Value Plots: Boxplots for Large Data. *Journal of Computational and Graphical Statistics*, 26, 03 2017. doi: 10.1080/10618600.2017.1305277.

Garud N. Iyengar. Robust Dynamic Programming. *Mathematics of Operations Research*, 30(2):257–280, 2005. doi: 10.1287/moor.1040.0129.

Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1181–1189, 2015.

Anders Jonsson. Deep Reinforcement Learning in Medicine. *Kidney Diseases*, 5:1–5, 10 2018. doi: 10.1159/000492670.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL: Model-Based Offline Reinforcement Learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21810–21823, 2020.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11784–11794, 2019.

Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe Policy Improvement with Baseline Bootstrapping. In *International Conference on Machine Learning*, pages 3652–3661. PMLR, 2019.

Byungjun Lee, Jongmin Lee, Peter Vrancx, Dongho Kim, and Kee-Eung Kim. Batch Reinforcement Learning with Hyperparameter Gradients. In *International Conference on Machine Learning*, pages 5725–5735. PMLR, 2020.

Sergey Levine, Aviral Kumar, G. Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems, 2020.

Elita A. Lobo, Mohammad Ghavamzadeh, and Marek Petrik. Soft-robust algorithms for batch reinforcement learning, 2021.

Anirudha Majumdar and Marco Pavone. How Should a Robot Assess Risk? Towards an Axiomatic Theory of Risk in Robotics. In Nancy M. Amato, Greg Hager, Shawna Thomas, and Miguel Torres-Torriti, editors, *Robotics Research*, pages 75–84, Cham, 2020. Springer International Publishing. ISBN 978-3-030-28619-4.

Mausam and Andrey Kolobov. Planning with Markov Decision Processes: An AI perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–210, 2012.

B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker. High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2156–2162, 2018. doi: 10.1109/ITSC.2018.8569448.

Arnab Nilim and Laurent El Ghaoui. Robust Control of Markov Decision Processes with Uncertain Transition Matrices. *Operations Research*, 53(5):780–798, 2005. doi: 10.1287/opre.1050.0216.

Marek Petrik and Reazul Hasan Russel. Beyond Confidence Regions: Tight Bayesian Ambiguity Sets for Robust MDPs. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, volume 32, 2019.

Marek Petrik, Yinlam Chow, and Mohammad Ghavamzadeh. Safe Policy Improvement by Minimizing Robust Baseline Regret. *Advances in Neural Information Processing Systems*, 29:2298–2306, 2016.

Apoorva Sharma, James Harrison, Matthew Tsao, and Marco Pavone. Robust and Adaptive Planning under Model Uncertainty. *Proceedings of the International Conference on Automated Planning and Scheduling*, 29(1):410–418, Jul. 2019.

Malcolm Strens. A Bayesian framework for Reinforcement Learning. In *In Proceedings of the Seventeenth International Conference on Machine Learning*, pages 943–950. ICML, 2000.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior Regularized Offline Reinforcement Learning, 2019.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based Offline Policy Optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14129–14142, 2020.

Exploitation vs Caution: Bayesian Risk-sensitive Policies for Offline Learning Supplementary Material

May 6, 2022

Environments

Ring. This environment is described by five states forming a single loop. Three actions are possible: *a*, *b*, *c*. The agent starts in state 0. *a* will move it to the state $s-1$ if $s=0,1,3$ and with probability 0.5 if it is elsewhere. With *b* the agent will not move with probability 0.8 and move to the left or to the right with probability 0.1 if it is in state $s=0,1,3$, if it is in state 2 or 4 it won't move with probability 1. *c* will move the agent to the right with probability 0.9 and it won't move with probability 0.1 if it is in state $s=0,1,3$. Otherwise the same effects will apply, but with probability 0.5. The agent earns an immediate reward $r=0.5$ if it moves from $2\rightarrow 3$ or $4\rightarrow 3$ and $r=1$ for any transition $3\rightarrow 3$. Elsewhere $r=0$. A graphical representation is shown in Figure 1.

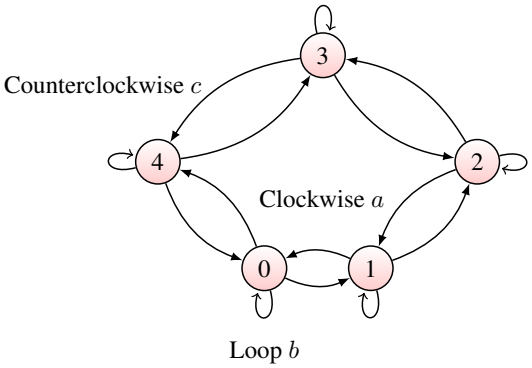


Figure 1: Representation of the Ring environment.

Chain. The environment proposed in Strens (2000) was adapted. There are five states with the topology of an open chain and two actions. The agent starts from the state most

to the left. With action a the agent moves to the right and receives an immediate reward $r = 0$ with probability 0.8. Once the agent is in the rightmost state, performing the first action lets him stay there and receive a reward $r = 10$ with probability 0.8. It slips back to the origin earning a reward $r = 2$ with probability 0.2. Action b teleports the agent to the origin with probability 0.8 receiving a reward $r = 2$ or let it go right with probability 0.2 earning $r = 0$. The optimal policy consists of applying action b in the first state and action a in the others. A representation is shown in Figure 2.

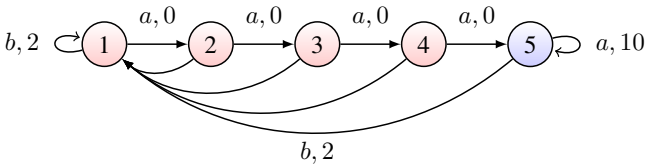


Figure 2: Representation of the Chain environment. Each circle is a state, each arrow is a transition labeled by action, reward.

Random Frozen Lake (RFL). The Frozen Lake Environment of the Open AI Gym suite of Brockman u. a. (2016) was edited. The agent moves in a grid world (8×8) . It starts in the utmost left corner and it must reach a distant absorbing goal state that yields a reward $r = 1$. In the grid there are some holes. If it falls in a hole it's blocked there and can not move anymore, obtaining from that moment an immediate reward $r = 0$. Unfortunately, the field is covered with ice and hence it is slippery. When the agent wants to move towards a nearby state it can slip with fixed probability p and ends up in an unintended place. The grid is generated randomly assuring that there always exists a hole-free path connecting the start and the goal. Moreover, to each couple of action and non terminal state (a, s) is assigned a different immediate reward r sampled at random between $(0, 0.8)$ at the moment of the generation of the MDP. The MDP itself does not have a stochastic reward, but the map and the rewards are randomly generated. A graphical representation (for a 3×3 grid) is shown in Figure 3.

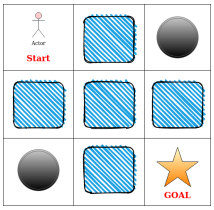
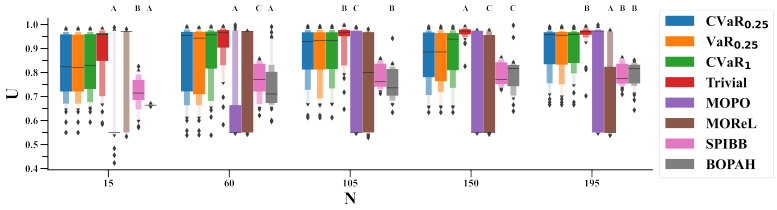
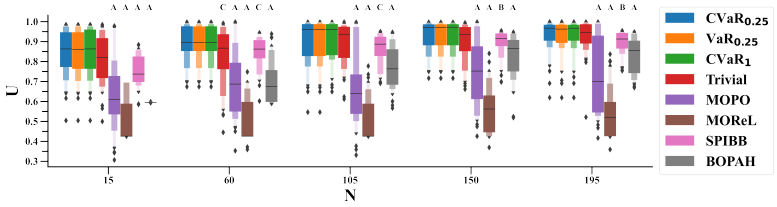


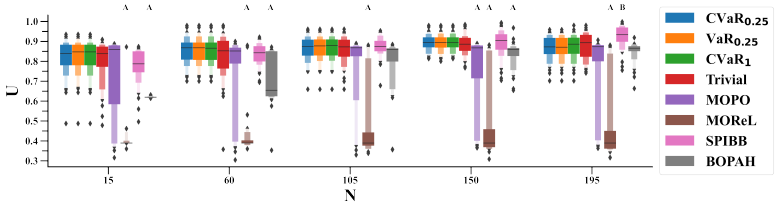
Figure 3: Frozen Lake environment example with grid of size 3×3. The agent has to reach the goal, paying attention to slippery (blue) states and avoiding holes (black).



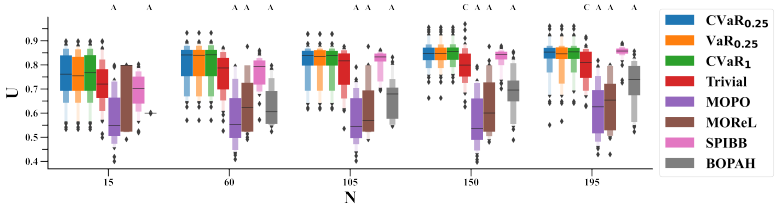
(a) Random Frozen Lake environment 1. Grid size 8×8



(b) Random Frozen Lake environment 2. Grid size 8×8

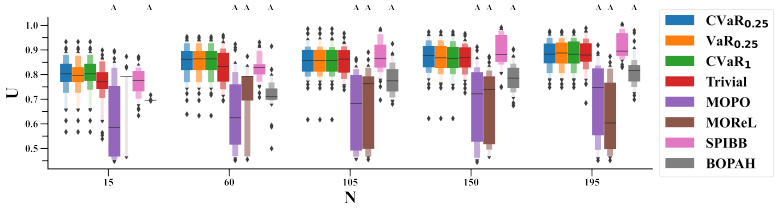


(c) Random Frozen Lake environment 3. Grid size 8×8

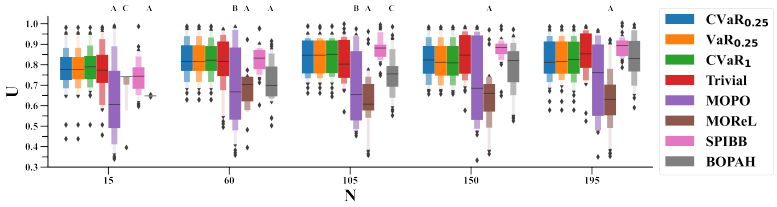


(d) Random Frozen Lake environment 4. Grid size 8×8

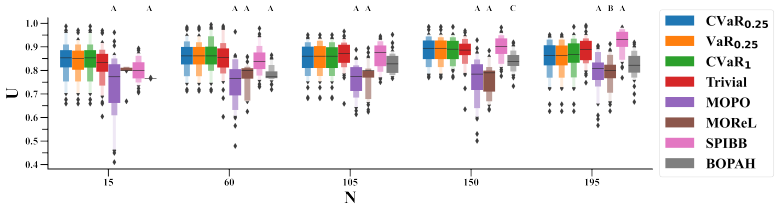
Figure 4: Letter-Value plot of the normalized performance U of EvC with $CVaR_{0.25}$, $VaR_{0.25}$, $CVaR_1$ compared to state-of-the-art approaches. N is the total number of transitions in a batch. Statistics are performed over 50 batches for each N. The median is a horizontal line, the largest box contains the central 50% of data, thus limited by the first and third quartiles, the other boxes above and below are delimited by the first and the seventh octiles, etc. Diamonds are outliers. The significance of the Wilcoxon post hoc test between $CVaR_{0.25}$ and the other baselines is displayed on top of each box: a p-value $p < 0.001$ is denoted by A, $p < 0.01$ is denoted by B, and $p < 0.05$ by C.



(a) Random Frozen Lake environment 5. Grid size 8×8



(b) Random Frozen Lake environment 6. Grid size 8×8



(c) Random Frozen Lake environment 7. Grid size 8×8

Figure 5: Letter-Value plot of the normalized performance U of EvC with $CVaR_{0.25}$, $VaR_{0.25}$, $CVaR_1$ compared to state-of-the-art approaches. N is the total number of transitions in a batch. Statistics are performed over 50 batches for each N. The median is a horizontal line, the largest box contains the central 50% of data, thus limited by the first and third quartiles, the other boxes above and below are delimited by the first and the seventh octiles, etc. Diamonds are outliers. The significance of the Wilcoxon post hoc test between $CVaR_{0.25}$ and the other baselines is displayed on top of each box: a p-value $p < 0.001$ is denoted by A, $p < 0.01$ is denoted by B, and $p < 0.05$ by C.

References

[Brockman u. a. 2016] BROCKMAN, Greg ; CHEUNG, Vicki ; PETTERSSON, Ludwig ; SCHNEIDER, Jonas ; SCHULMAN, John ; TANG, Jie ; ZAREMBA, Wojciech: *OpenAI Gym*. 2016

[Strens 2000] STRENS, Malcolm: A Bayesian framework for Reinforcement Learning. In: *In Proceedings of the Seventeenth International Conference on Machine Learning*, ICML, 2000, S. 943–950