Planification Monte Carlo orientée information

Vincent Thomas, Gérémy Hutin, Olivier Buffet

Planification Monte Carlo orientée information

Vincent Thomas[1] Gérémy Hutin[2] Olivier Buffet[1]

[1] Université de Lorraine, INRIA, CNRS, LORIA, Nancy, FRANCE
[2] École Normale Supérieure, Lyon, FRANCE

{prénom.nom@(inria.fr|ens-lyon.fr)}

Résumé

Dans cet article, nous nous intéressons à la résolution de problèmes de collecte active d'information exprimés sous la forme de ρ-POMDP, une extension des Processus Décisionnels de Markov Partiellement Observables (POMDP) dont la récompense ρ dépend de l'état de croyance. Des approches utilisées pour résoudre les POMDP ont déjà été étendues au cadre ρ-POMDP lorsque la récompense ρ est convexe ou lipschtizienne, mais ces approches ne permettent pas de résoudre toutes les instances de ρ-POMDP. Afin de proposer un algorithme en-ligne efficace qui s'affranchit des contraintes sur ρ, cet article se concentre sur les méthodes à base de recherche arborescente Monte Carlo et cherche à adapter POMCP à la résolution de ρ-POMDP. Comme les récompenses dépendent de l'état de croyance, il est nécessaire de modifier POMCP (i) pour échantillonner plusieurs états lors des trajectoires suivies et (ii) pour éviter les biais dans l'estimation des valeurs. Des expériences ont été conduites pour étudier les propriétés de l'approche proposée.

Mots Clef

Planification dans l'incertain, recherche active d'information, Approche MCTS, POMDP, ρ-POMDP.

Abstract

In this article, we address the issue of solving information-gathering problems expressed as ρ-POMDPs, an extension of Partially Observable Markov Decision Processes (POMDPs) whose reward ρ depends on the belief state. Approaches already used for solving POMDPs have been extended to solving ρ-POMDPs as belief MDPs when the reward ρ is convex in $\mathcal{B}$ or when it is Lipschitz-continuous (LC), using respectively PWLC or LC approximators. In the present paper, we build on the POMCP algorithm to propose a Monte Carlo Tree Search for ρ-POMDPs, aiming for an efficient online planner. Adaptations are required due to the belief-dependent rewards to (i) propagate more than one state at a time, and (ii) prevent biases in value estimates. Experiments are conducted to analyze the approach at hand.

Keywords

Planning under uncertainty, active information gathering, MCTS, POMDP, ρ-POMDP.

1 Introduction

De nombreux algorithmes de l'état de l'art pour la résolution de *processus de décision markoviens partiellement observables* (POMDP) consistent à transformer le problème en un problème «complètement observable»—plus précisément un *belief MDP* (ou MDP sur les états de croyance)—-et à exploiter la linéarité par morceaux et la convexité de la fonction de valeur optimale dans l'espace d'état de ce nouveau problème (ici l'espace des croyances $\mathcal{B}$) en maintenant des approximateurs de fonction généralisant. Cette approche a été étendue à la résolution de ρ-POMDP, c.-à-d. des problèmes dont le critère de performance dépend de la croyance (par exemple en collecte active d'information), quand ρ elle-même est convexe en $\mathcal{B}$ [1] ou quand ρ est lipschitzienne [10]. Dans cet article, nous proposons un nouvel algorithme pour résoudre des ρ-POMDP qui ne nécessite ni la convexité ni la Lipschitz-continuité de ρ, mais repose sur de l'échantillonnage Monte Carlo et est inspiré par POMCP [17]. Cet algorithme utilise des filtres particulaires et échantillonne des particules pendant les trajectoires pour estimer les états de croyance visités B et les récompenses associées $\rho(B)$. Cet algorithme et ses variantes sont évaluées empiriquement sur divers problèmes de collecte d'information.

L'article est organisé comme suit. La section 2 discute de travaux connexes sur le contrôle orienté information. La section 3 présente un état de l'art (i) d'abord sur les processus de décision markoviens (MDP) et les recherches arborescentes Monte Carlo (MCTS), (ii) puis sur les POMDP et l'algorithme *Partially Observable Monte Carlo Planning* (POMCP), une approche MCTS pour la résolution de POMDP et, (iii) enfin sur les ρ-POMDP. La section 4 décrit notre contribution : deux algorithmes, ρ-beliefUCT et ρ-POMCP, pour la résolution de ρ-POMDP avec des approches MCTS. Enfin, la section 5 présente les expérimentations conduites et analyse les résultats avant de conclure et de donner quelques perspectives.

2 Travaux connexes

Les premiers travaux sur le contrôle orienté information (IOC) impliquaient des problèmes formalisés soit (i) comme des POMDP (comme EGOROV, KOCHENDERFER et UUDMAE [9] l'ont fait récemment, puisqu'une récompense dépendant de l'observation peut trivialement être reformulée comme une récompense dépendant de l'état), ou (ii) avec des récompenses dépendant de la croyance (et des méthodes de résolution essentiellement ad-hoc comme [11, 14]).

ARAYA-LÓPEZ et al. [1] proposent les ρ-POMDP pour formaliser facilement de nombreux—si ce n'est la plupart—problèmes IOC. Ils montrent qu'un ρ-POMDP dont la fonction de récompense dépendant de la croyance ρ est convexe peut être résolu avec des solveurs de POMDP à base de points modifiés qui exploitent toujours la propriété PWLC (avec des bornes d'erreurs qui dépendent de la qualité de l'approximation PWLC de ρ).

Le cadre POMDP-IR (*POMDP with Information Reward*) de SPAAN, VEIGA et LIMA [19] permet de décrire des problèmes IOC avec des récompenses linéaires en b qui sont démontrés comme équivalents aux ρ-POMDP «PWLC» (c.-à-d. quand ρ est PWLC) [16]. Dans les deux cas les techniques de résolution proposées sont des solveurs de POMDP modifiés. Pour sa part, le cadre général des ρ-POMDP permet de formaliser plus de problèmes, par exemple en spécifiant directement un critère reposant sur l'entropie de Shannon.

Plus récemment, FEHR et al. [10] ont appliqué la même approche que ARAYA-LÓPEZ et al. [1], mais pour des récompenses croyance-dépendantes lipschitziennes (LC) plutôt que convexes. Ils ont démontré que, quand ρ est lipschitzienne, alors la fonction de valeur optimale est aussi lipschitzienne à horizon fini. Elle peut alors être minorée (et majorée) par des bornes uniformément améliorables, et deux algorithmes reposant sur HSVI ont été proposés pour tirer parti de ces encadrements.

A contrario, le présent article ne repose pas sur des approximateurs de la fonction de valeur généralisant (PWLC ou LC). N'importe quelle récompense croyance-dépendante peut ainsi être considérée. Dans ce but, nous avons décidé de tirer parti des recherche arborescentes Monte Carlo (MCTS), en particulier de l'algorithme Partially Observable Monte Carlo Planning (POMCP) [17] qui ne repose pas sur la propriété PWLC. MCTS et POMCP présentent en outre plusieurs autres intérêts : (i) ils ne requièrent pas un modèle complet mais seulement un simulateur ; (ii) contrairement aux recherches heuristiques, ils ne requièrent pas d'initialisations optimistes ou pessimistes de la fonction de valeur ; et (iii) ils se sont montrés très efficaces pour résoudre des problèmes avec de très grands espaces d'action et d'observation.

3 État de l'art

3.1 MDP et algorithmes par échantillonage

Processus de décision markoviens Un processus de décision markovien (MDP) [15] est défini par un tuple $\langle S, A, P, r\rangle$ où S est l'ensemble fini des états du système ; A est l'ensemble fini des actions possibles ; la fonction de transition P donne la probabilité de transiter d'un état s à un autre s' quand une action a est accomplie : $P_a(s, s') = Pr(s'|s, a)$; et $r(s, a, s')$ est la récompense scalaire instantanée reçue pendant cette transition. Résoudre un MDP consiste à trouver une politique—une application $\pi : S \to A$—maximisant un critère de performance (dans notre cas, la somme γ-atténuée des récompenses).

Algorithmes en-ligne Dans cet article, nous nous concentrerons sur les algorithmes *en-ligne*. À chaque pas de temps, un tel algorithme estime la meilleure action à effectuer. Le système effectue alors cette action et reçoit de l'information de l'environnement pour mettre à jour sa connaissance de l'état courant. Cet algorithme en-ligne est ensuite appelé à nouveau pour décider de la prochaine action à accomplir. Les approches en-ligne ont l'avantage de restreindre les états considérés à ceux qui sont atteignables, et peuvent se concentrer seulement sur ceux qui apparaissent pertinents.
Un premier algorithme naïf consiste à appliquer la programmation dynamique à l'arbre complet des futurs possibles jusqu'à une certaine profondeur. Les valeurs sont alors remontées depuis les feuilles pour estimer la valeur de chaque action à la racine. Toutefois, cet algorithme souffre d'une explosion combinatoire du fait de la croissance exponentielle du nombre de nœuds avec la profondeur considérée.

Algorithmes reposant sur l'échantillonnage Une première amélioration a été faite avec *Sparse Sampling* (SS) [12], lequel n'échantillonne qu'un nombre limité de prochains états après chaque action, et estime les valeurs des actions par une moyenne. Cela réduit grandement le temps de calcul et requiert non pas un modèle complet du processus, mais seulement un simulateur. Toutefois SS échantillonne des trajectoires uniformément sans considérer les valeurs estimées, alors qu'il pourrait être amélioré en se concentrant sur les parties prometteuses de l'arbre, ce qui est le but des recherches arborescentes Monte Carlo (MCTS).

Approches MCTS Dans les approches MCTS [7, 13, 6], l'arbre représentant les futurs possibles à partir d'un état de départ est construit d'une manière incrémentale non-uniforme. Chaque itération consiste en 4 étapes :

- **sélection** : une trajectoire est échantillonnée dans l'arbre selon une stratégie d'exploration jusqu'à ce qu'un nœud n'appartenant pas à l'arbre soit atteint ;
- **expansion** : ce nouveau nœud est ajouté à l'abre ;
- **simulation** : la valeur de ce nouveau nœud est estimée en échantillonnant une trajectoire depuis ce nœud suivant

une politique *rollout* (de «déroulement») ;

rétro-propagation : cette estimation et les récompenses reçues durant l'étape de sélection sont alors rétro-propagées aux nœuds visités pour mettre à jour leurs statistiques (valeur et nombre de visites).

UCT MCTS a été appliqué avec succès pour résoudre des MDP [13]. Dans *Upper Confidence Bound applied to trees* (UCT), la stratégie d'exploration utilisée pour échantillonnée une trajectoire dans l'arbre repose sur *Upper Confidence Bounds* (UCB1). À chaque nœud état, cette stratégie sélectionne l'action qui maximise la valeur estimée augmentée d'un bonus d'exploration $c_{t,s} = C_p\sqrt{\frac{logN(s)}{N(s,a)}}$, où C_p est une constante d'exploration qui doit être choisie correctement, $N(s)$ est le nombre de visites passées du nœud s, et $N(s,a)$ est le nombre de fois où l'action a a été choisie dans le nœud s. Ce bonus d'exploration dépend du nombre de visites et garantit que toutes les actions seront sélectionnées dans le future, même si l'une d'entre elles a une valeur estimée plus grande que les autres. De plus, cette stratégie de sélection d'action fait converger asymptotiquement les valeurs d'actions estimées vers la valeur optimale.

3.2 POMDP

Un POMDP [3] est défini par un tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, P, r, \gamma, b_0 \rangle$, où $\mathcal{S}$, $\mathcal{A}$ et $\mathcal{Z}$ sont des ensembles finis d'états, d'actions et d'observations ; $P_{a,z}(s,s')$ donne la probabilité de transiter vers l'état s' en observant z quand l'action a est appliquée en l'état s ($P_{a,z}$ est une matrice $\mathcal{S} \times \mathcal{S}$) ; $r(s,a) \in \mathbb{R}$ est la récompense associée à l'accomplissement de l'action a dans l'état s ; $\gamma \in [0;1)$ est un facteur d'atténuation ; et b_0 est l'état de croyance initial—c.-à-d. la distribution de probabilité initiale sur les états possibles. L'objectif est alors de trouver une politique π qui prescrit des actions en fonction des actions et observations passées de façon à maximiser l'espérance de la somme des récompenses atténuées (ici avec un horizon temporel infini).

Pour cela, un POMDP est souvent transformé en un belief MDP $\langle \Delta, \mathcal{A}, P, r, \gamma, b_0 \rangle$, où Δ est le simplexe des états de croyance possibles, $\mathcal{A}$ est le même ensemble d'actions, et $P_a(b,b') = P(b'|b,a)$ et $r(b,a) = \sum_s b(s)r(s,a)$ sont les fonctions de transition et de récompense induites. Ce cadre permet de considérer des politiques $\pi \colon \Delta \to \mathcal{A}$, chacune étant associée à sa fonction de valeur $V^\pi(b) \stackrel{\text{def}}{=} E[\sum_{t=0}^{\infty} \gamma^t r(b_t, \pi(b_t))|b_0 = b]$. Les politiques optimales maximisent V^π dans tous les états de croyance accessibles depuis b_0. Leur fonction de valeur V^* est le point fixe de l'opérateur *d'optimalité* de Bellman ($\mathcal{H}$) [4] $\mathcal{H}V : b \mapsto \max_a[r(b,a) + \gamma \sum_z \|P_{a,z}b\|_1 V(b^{a,z})]$, et agir de manière gourmande par rapport à V^* fournit une telle politique.

3.3 MCTS pour POMDP

SILVER et VENESS [17] ont proposé l'algorithme *Partially Observable Monte Carlo Planning* (POMCP) pour

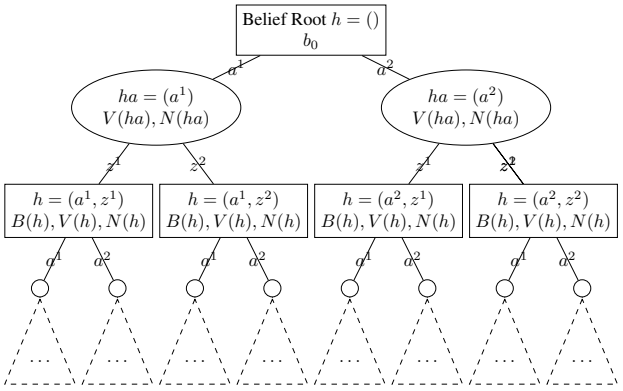

FIGURE 1 – Exemple d'un arbre de croyances avec 2 actions a^1 et a^2, et 2 observations z^1 et z^2. Chaque nœud croyance est représenté par un nœud carré et associé à un historique $h = (a_0, z_0, \ldots, z_t)$. Pendant son exécution, POMCP va calculer $B(h)$, une estimation de son état de croyance, et $V(h)$, une estimation de sa valeur optimale. Il comptera aussi le nombre des visites passées $N(h)$. Chaque nœud-action est représenté par une ellipse et associé à un historique ha. POMCP va calculer une valeur estimée $V(ha)$ et compter le nombre de visites $N(ha)$.

appliquer MCTS à la résolution de POMDP. Un POMDP est abordé à travers le belief MDP correspondant, un arbre d'états de croyance étant fait d'une alternance de nœuds action et de nœuds croyance comme présenté dans la figure 1. Le chemin jusqu'à un nœud croyance à profondeur t suit l'historique action-observation $h_t = (a_0, z_0, a_1, z_1, \ldots, z_t)$ allant de la croyance racine à cette croyance $b(h_t)$.

Appliquer directement UCT au belief MDP échantillonnerait des trajectoires $(b_0, a_0, b_1, a_1, \ldots, b_t)$ dans l'espace des croyances, requiérant ainsi le modèle POMDP complet et un coût computationnel élevé pour calculer des états de croyance exacts. Pour éviter ce coût, POMCP échantillonne des trajectoires dans l'espace d'états, ce qui ne requiert qu'un simulateur comme modèle génératif du POMDP. Étant donné un état s et une action a, ce simulateur $\mathcal{G}$ fournit un état successeur s', une observation z, et une récompense r supposée échantillonnés d'après le vrai modèle sous-jacent $(s', z, r) \sim \mathcal{G}(s, a)$.

Pendant la phase de sélection, POMCP utilise ce modèle génératif pour échantillonner une trajectoire : le premier état est échantillonné à partir de l'estimation de la croyance à la racine, puis on alterne entre (i) choisir une action suivant UCB1 (appliqué aux valeurs d'action estimées dans le nœud croyance courant), et (ii) échantillonner un prochain état, une observation et une récompense selon le modèle génératif $\mathcal{G}$. En accumulant des valeurs dans les nœuds croyance, faire une moyenne sur toutes les trajectoires simulées donne une estimation de la valeur $V(h)$ du nœud croyance h, même si seules des trajectoires sur les états ont été échantillonnées.

En outre, les états sont collectés dans chaque nœud croyance visité de manière à estimer la prochaine croyance à la racine quand une action est effectivement appliquée.

En évitant le calcul des croyances exactes $b(h)$ dans chaque nœud croyance, POMCP permet d'aborder des problèmes plus grands, tout en préservant les garanties de convergence asymptotiques comme UCT.
SILVER et VENESS [17] ont aussi proposé l'algorithme PO-UCT comme une première étape vers POMCP. PO-UCT diffère de POMCP dans la façon de calculer l'état de croyance de la nouvelle racine : au lieu de collecter des états échantillonnés, quand une action est effectivement appliquée et une observation reçue, alors l'état de croyance est calculé avec une mise à jour de Bayes exacte.

3.4 ρ-POMDP

Les ρ-POMDP [1] diffèrent des POMDP en ce que leur fonction de récompense ρ dépendent de l'état de croyance, permettant ainsi de définit non seulement des critères orientés contrôle, mais aussi des critères orientés information, généralisant ainsi les POMDP. La récompense immédiate ρ est naturellement définie comme $\rho(b, a, b')$, la récompense immédiate associée à la transition de la croyance b à la croyance b' après avoir effectué l'action a. Par simplicité et sans perte de généralité, cet article utilise la notation $\rho(b, a)$ (même si les problèmes considérés reposent sur $\rho(b, a, b')$). Il est aisé d'étendre les algorithmes proposés en raisonnant sur $\rho(b, a, b')$ au lieu de $\rho(b, a)$.
Comme présenté dans les travaux connexes, ARAYA-LÓPEZ et al. [1] et FEHR et al. [10] ont exploité les récompenses ρ PWLC et lipschitziennes pour résoudre des ρ-POMDP généraux. Toutefois, si de nombreux problèmes peuvent être modélisés avec des récompenses ρ convexes ou lipschitziennes, cela nous laisse avec de nombreux problèmes qui ne peuvent être résolus avec des approximations similaires.
Ici, nous proposons d'utiliser l'approche MCTS pour aborder le cas ρ-POMDP général sans contrainte sur la fonction ρ. À titre d'exemple, considérons un agent surveillant un musée et dont l'objectif est de localiser un visiteur avec une certitude donnée. Si X dénote la variable aléatoire pour la position du visiteur et b_X la croyance associée, alors la fonction de récompense suivante

$$\rho_X(b, a) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{si } \|b_X\|_\infty > \alpha \\ 0, & \text{sinon} \end{cases}$$

récompense les distributions dont la probabilité maximale est plus grande que $\alpha \in [0, 1]$. Il s'agit d'une fonction seuil, donc ni convexe, ni lipschitzienne du fait de sa discontinuité là où $\|b_X\|_\infty = \alpha$.

4 Algorithmes MCTS pour ρ-POMDP

POMCP et sa variante PO-UCT ne peuvent être appliqués directement à la résolution de ρ-POMDP. PO-UCT ne calcule pas les croyances exactes durant ses étapes de sélection et de rétro-propagation (mais seulement à la racine de l'arbre), et ne peut donc calculer les récompenses croyance-dépendantes le long d'une trajectoire. POMCP génère des trajectoires en ne suivant qu'une séquence d'états et échantillonne des récompenses liées à ces états. Il collecte les états visités dans les nœuds croyance, mais ceux-ci ne sont pas suffisants pour correctement estimer les états de croyance et calculer précisément les récompenses croyance-dépendantes comme requis par les ρ-POMDP.

4.1 Algorithme ρ-beliefUCT

Le premier algorithme proposé dans cet article est appelé *ρ-beliefUCT*. Il consiste à appliquer directement UCT au belief MDP dérivé du ρ-POMDP. Cela requiert d'avoir accès au modèle ρ-POMDP complet et de calculer des croyances exactes pour chaque nœud croyance de l'arbre visité en effectuant la mise à jour de Bayes :

$$b_{haz}(s') \propto \sum_{s \in S} P_{a,z}(s, s').b_h(s).$$

Pendant l'étape de simulation, une politique rollout dépendant de la croyance (par opposition à une politique aléatoire) a besoin de calculer non seulement la croyance à chaque choix d'action, mais aussi les récompenses immédiates $\rho(b, a)$ pour estimer la valeur du nœud ajouté. Cela induit un important coût computationnel.
ρ-beliefUCT peut toutefois tirer parti de plusieurs éléments. D'abord, puisque chaque historique correspond à un unique état de croyance, chaque état de croyance n'a besoin d'être calculé qu'une fois quand le nœud correspondant est ajouté. Deuxièmement, alors que, dans POMCP, $r(b, a)$ est estimé comme une moyenne non biaisée parce que $r(b, a)$ est linéaire en b, dans notre contexte, $\rho(b, a)$ est une fonction déterministe de la croyance : sa valeur exacte peut être mémorisée directement dans le nœud action. Enfin, on peut échantillonner une observation pendant l'étape de sélection sans calculer la probabilité d'observation $P(z|b(h))$. Cela peut être fait sans biais en échantillonnant d'abord un état s de la croyance courante $b(h)$, puis un état s' et une observation z du modèle génératif du POMDP (ou en employant la distribution de probabilité $P_{z,a}(s, s')$ si elle est disponible).
ρ-beliefUCT est un algorithme de référence intéressant. Toutefois, pour éviter (i) le coût associé au calcul exact des croyances, et (ii) le besoin d'un modèle complet du POMDP, nous proposons un autre algorithme, ρ-POMCP, qui ne calcule pas les croyances exactes, mais les estime en collectant des états échantillonnés dans les nœuds croyance.

4.2 Algorithme ρ-POMCP

L'algorithme ρ-POMCP est similaire à POMCP. Pendant l'étape de sélection, les trajectoires sont générées en échantillonnant des états et des observations à l'aide du modèle génératif du POMDP $\mathcal{G}$. Lors de leur visite, chaque nœud croyance h collecte l'état qui a conduit à ce nœud de manière à construire une estimation du vrai état de croyance $b(h)$ correspondant à cette historique.

Appliquer POMCP directement en n'ajoutant que l'état courant de la trajectoire au nœud croyance sans plus de considération conduira toutefois à de très mauvaises estimations des récompenses immédiates, lesquelles amèneraient l'algorithme (i) à dépenser inutilement du temps de calcul en se concentrant sur des branches avec des récompenses surestimées et (ii) à ralentir l'exploration de branches intéressantes dont les récompenses seraient sous-estimées.

Nous proposons ainsi d'ajouter non pas un, mais plusieurs états à chaque nœud visité pendant l'étape de sélection. Ainsi, à chaque transition, quand l'action a est sélectionnée à partir de l'état de la trajectoire s et que l'observation z est échantillonnée, l'algorithme génère un ensemble β de $|\beta|$ états, appelé un *petit sac de particules*, en utilisant le modèle génératif pour faire $|\beta|$ échantillonnages cohérents avec l'observation z. Ensuite, il ajoute toutes les particules de ce petit sac β à $B(haz)$, appelé dans ce contexte le *sac cumulé* de l'historique haz, et stocké dans le nœud croyance correspondant : $B(haz) \leftarrow B(haz) \cup \beta$.

Avec un modèle génératif $\mathcal{G}$, seules les approches par réjection peuvent être considérées pour produire des échantillonnages cohérents. Dans ce cas, un couple $(\tilde{s}', \tilde{z})$ est échantillonné du modèle génératif $(\tilde{s}', \tilde{z}) \sim \mathcal{G}(s, a)$ et $\tilde{s}'$ est gardé dans le sac généré si et seulement si l'observation échantillonnée $\tilde{z}$ correspond à l'observation z de la trajectoire suivie. Ce processus s'accomplit jusqu'à ce que le nombre requis d'échantillonnages cohérents soit atteint (et les états correspondants soient stockés dans β). Ceci peut toutefois être coûteux en temps de calcul, en particulier quand l'observation z a une faible probabilité étant donnés h et a.

Échantillonnage selon l'importance Pour éviter ce coût computationnel, nous utilisons l'échantillonnage selon l'importance [8] au lieu de l'échantillonnage par réjection, ce qui suppose que la fonction d'observation est disponible, et conduit à pondérer chaque particule. Après avoir effectué l'action a_t et reçu l'observation z_t pendant l'étape de sélection, un nouveau petit sac β_{t+1} est généré à partir de β_t en utilisant le modèle génératif et en suivant les étapes suivantes $|\beta|$ fois : (1) échantillonner un état $\tilde{s}$ de β_t ; (2) échantillonner un état $\tilde{s}'$ en appliquant le modèle génératif sur $\tilde{s}$, $\tilde{s}' \sim \mathcal{G}(\tilde{s}, a_t)$; (3) stocker cet état $\tilde{s}'$ dans β_{t+1} et lui associer le poids correspondant à la probabilité $P(z|\tilde{s}, a, \tilde{s}')$ d'avoir généré l'observation z. Les poids dans deux petits sacs associés au même nœud peuvent être comparés, et on peut donc accumuler de tels sacs dans le nœud croyance correspondant. Pour économiser de la mémoire, quand un petit sac β est ajouté à un *sac cumulé* $B(h)$, les particules correspondant au même état sont fusionnées et leurs poids additionnés. Enfin, pendant les diverses descentes d'une exécution de ρ-POMCP, le nombre de particules accumulées dans le *sac cumulé* $B(h)$ stocké dans le nœud croyance h croit, donnant une meilleure estimation de la vraie croyance $b(h)$. On n'a alors qu'à normaliser les poids pour obtenir une distribution de probabilité valide dès que nécessaire.

Notons que l'état s_t employé dans la génération de la trajectoire $h_t = (s_0, a_0, \ldots, a_{t-1}, s_t)$ au pas de temps t est aussi inclus dans le petit sac β_t pendant la génération de particules, de sorte que l'observation générée avec la trajectoire peut être expliquée au moins par une particule dans le nouveau sac. Alors que cela introduit un biais dans les petits sacs β de cette trajectoire, la distribution des états de la trajectoire dans les *sacs cumulés* n'est pas biaisée puisque ces états sont obtenus par échantillonnage depuis la croyance initiale.

Lors de l'étape de simulation, des estimations de croyance sont requises ne serait-ce que pour estimer les récompenses instantanées, et éventuellement pour prendre des décisions. En conséquence, la séquence de petits sacs β doit être perpétuée.

Enfin, pendant l'état de rétro-propagation, le *sac cumulé* de particules $B(h)$ présent dans un nœud croyance est utilisé comme estimation du vrai état de croyance $b(h)$ pour calculer la récompense ρ. Les poids ne sommant pas à 1 dans le *sac cumulé*, l'estimation de la croyance requiert une normalisation ; mais les poids restent intacts dans le sac de sorte que de nouvelles particules peuvent être ajoutées sans introduire de biais.

La description ci-dessus conduit à l'algorithme 1. Dans celui-ci, la notation $\{s\} \cup \beta \overset{1+n}{\sim} I$ implique que $1 + n$ états sont échantillonnés de I, dont un est stocké dans s et les n autres dans β. La fonction ***PF*** correspond au processus de filtrage particulaire décrit précédemment, et $w_{s'} = P(z|s, a, s')$.

On notera que, comme dans POMCP ou dans des filtres particulaires standards, quand le système évolue effectivement (une action étant accomplie réellement), l'état réel peut ne pas être dans le sac cumulé de la nouvelle racine, et il peut arriver que l'observation reçue ne puisse être expliquée par aucun état contenu dans ce sac. Nous ne discuterons pas plus avant de cette situation, mais il reste possible de ré-estimer la croyance courante en simulant le processus depuis la croyance initiale jusqu'au pas de temps en question.

4.3 Variantes de ρ-POMCP

La différence courante entre POMCP et ρ-POMCP se trouve dans la façon dont les particules sont collectées afin d'estimer la récompense obtenue à chaque transition, alors que les mises à jour des valeurs restent identiques. Celles-ci peuvent toutefois être améliorées, conduisant à plusieurs variantes de ρ-POMCP puisque, lors de son exécution, ρ-POMCP construit des estimations de la croyance $b(h)$ de qualité croissante dans les nœuds croyance visités. Pour ce faire, nous revenons d'abord sur ce qui est calculé dans les mises à jour effectuée par POMCP (et la version de base de ρ-POMCP), puis proposons plusieurs variantes.

Calculs effectués par POMCP Si on ignore l'initialisation du nœud dans POMCP, alors, pour un couple nœud-action ha, la valeur stockée dans $V(ha)$ fait la moyenne,

Algorithm 1: ρ-POMCP

/* Le texte en rouge souligne les différences avec POMCP. */

Fct SEARCH (h)
  repeat
    if $h = empty$ **then** $\{s\} \cup \beta \overset{1+n}{\sim} I$
    else $\{s\} \cup \beta \overset{1+n}{\sim} B(h)$
    SIMULATE $(s, \beta, h, 0)$
  until TIMEOUT()
  return $\arg\max_b V(hb)$

Fct ROLLOUT (s, β, h, δ)
  if $\gamma^\delta < \epsilon$ **then return** 0
  $a \sim \pi_{rollout}(h, \cdot)$
  $(s', z) \sim \mathcal{G}(s, a)$
  $\beta' \leftarrow$PF$(\beta, a, z) \cup \{(s', w_{s'})\}$
  return $\rho(\beta, a) + \gamma$.ROLLOUT $(s', \beta', haz, \delta + 1)$

Fct SIMULATE (s, β, h, δ)
  if $\gamma^\delta < \epsilon$ **then return** 0
  if $h \notin T$ **then**
    forall $a \in \mathcal{A}$ **do**
      $T(ha) \leftarrow (N_{init}(ha), V_{init}(ha), \emptyset)$
    return ROLLOUT (s, β, h, δ)
  $a \leftarrow \arg\max_b V(hb) + c\sqrt{\frac{\log N(h)}{N(hb)}}$
  $(s', z) \sim \mathcal{G}(s, a)$
  $\beta' \leftarrow$PF$(\beta, a, z) \cup \{(s', w_{s'})\}$
  $B(h) \leftarrow B(h) \cup \beta$
  $R \leftarrow \rho(B(h), a) + \gamma$.SIMULATE $(s', \beta', haz, \delta + 1)$
  $N(h) \leftarrow N(h) + 1$
  $N(ha) \leftarrow N(ha) + 1$
  $V(ha) \leftarrow V(ha) + \frac{R - V(ha)}{N(ha)}$
  return R

sur $N(ha)$ échantillons (/visites), de :

- $\sum_{s \in \beta_{ha}} r(s, a)$: la récompense totale sur les états s qui ont été échantillonnés quand l'action a a été choisie dans h (β_{ha} dénote cet ensemble d'états) ;
- $\sum_{z \in \mathcal{Z}_{ha}} \mathbf{1}_{N(h,a,z) \geq 1} Rollout(haz)$: le retour total des rollouts générés à partir de chaque observation z échantillonnée après avoir choisi a dans h (ensemble $\mathcal{Z}_{ha}$), où $N(h, a, z)$ est le nombre de fois où l'action a a été suivie de l'observation z dans le nœud h (alors que $N(haz)$ est le nombre de mises à jour du nœud haz) ;
- $\sum_{z \in \mathcal{Z}_{ha}} \sum_{a' \in \mathcal{A}} N(haza')V(haza')$: la somme des valeurs de tous les nœuds fils, pondérée par leurs nombres de visites (ce qui inclut aussi les rollouts effectués depuis ces nœuds).

En introduisant les estimations de valeurs des nœuds croyance $V(h)$, initialisées avec les valeurs des rollouts (pour $N(h) = 1$), cela conduit aux formules suivantes :

$$V(h) \leftarrow \frac{1}{N(h)} \left[Rollout(h) + \sum_{a' \in \mathcal{A}} N(ha')V(ha') \right] ;$$

$$V(ha) \leftarrow \frac{1}{N(ha)} \left[\sum_{s \in \beta_{ha}} r(s, a) + \gamma \sum_{z \in \mathcal{Z}_{ha}} N(haz)V(haz) \right] .$$

Last-reward-update ρ-POMCP Ainsi, dans POMCP, $V(ha)$ est une moyenne mobile qui *«contient»* une estimation de $r(b(h), a)$, cette estimation étant calculée comme $\frac{\sum_{s \in \beta_{ha}} r(s,a)}{N(ha)}$.
Dans ρ-POMCP de base, la valeur de r au pas de temps courant est remplacée par l'estimation de $\rho(b(h))$ comme $\rho(\hat{B}_{N(h)}(h))$, où $\hat{B}_{N(h)}(h)$ est le sac cumulé après les $N(h)$ premières visites. Dans ce cas, $V(ha)$ inclut ainsi (entre autres choses) une moyenne d'estimations successives (c.-à-d. qu'il calcule $\sum_{i=1}^{N(ha)} \rho(\hat{B}_{\phi(i)}(h), a)$, où $\phi(i)$ est la $i^{\text{ème}}$ visite de h où a a été sélectionnée). Il semblerait pourtant plus approprié d'employer à la place la récompense associée à la dernière estimation de la croyance, $\rho(\hat{B}_{\phi(N(ha))}(h), a)$, laquelle est une estimation moins biaisée.
Cette variante, appelée *last-reward-update ρ-POMCP* (ou *"lru-ρ-POMCP"*), corrige cela assez simplement en remplaçant la mise à jour de $V(ha)$ dans l'algorithme 1 par

$$V(ha) \leftarrow \frac{N(ha) - 1}{N(ha)} [V(ha) - \rho^{prev}(h, a)] + \rho(B(h), a) + \frac{1}{N(ha)} \gamma.\text{SIMULATE}(s', haz, \delta + 1),$$

où $\rho^{prev}(h, a)$ est la précédente valeur de la récompense quand ha a été rencontrée pour la dernière fois (valeur qui doit donc être mémorisée).

Last-value-update ρ-POMCP De la même manière, $V(ha)$ «contient» aussi des estimations des récompenses moyennes des futurs pas de temps, lesquelles souffrent du

même problème. Pour réparer cela, SIMULATE devrait retourner non pas un échantillon de retour, mais une estimation du retour moyen.
Les mises à jour dans l'étape de rétro-propagation consistent ainsi à ré-estimer toutes les valeurs des nœuds croyance visités en utilisant la récompense obtenue à chaque transition et le rollout initial qui doit avoir été mémorisé auparavant. Ceci est accompli dans la variante *last-value-update ρ-POMCP* (ou *"lvu-ρ-POMCP"*) avec la formule suivante :

$$V(h) \leftarrow \frac{1}{N(h)}\left[Rollout(h) + \sum_a N(ha)V(ha)\right],$$
$$V(ha) \leftarrow \rho(B(h),a) + \frac{\gamma}{N(ha)}\sum_z \left[N(haz).V(haz)\right].$$

Le processus résultant est montré dans l'algorithme 2.

Algorithm 2: Last-value-update ρ-POMCP

Fct SEARCH (h)
  /* Untouched/Identical */
Fct ROLLOUT (s, β, h, δ)
  /* Untouched/Identical */
Fct SIMULATE (s, β, h, δ)
  if $\gamma^\delta < \epsilon$ **then return** 0
  if $h \notin T$ **then**
    forall $a \in \mathcal{A}$ **do** $T(ha) \leftarrow (0, 0, \emptyset)$
    $N(h) \leftarrow 1$
    $V(h) \leftarrow rollout(h) \leftarrow$ ROLLOUT (s, β, h, δ)
    return $V(h)$
  $a \leftarrow \arg\max_b V(hb) + c\sqrt{\frac{\log N(h)}{N(hb)}}$
  $(s', z, r) \sim \mathcal{G}(s, a)$
  $\beta' \leftarrow$**PF**$(\{s\} \cup \beta, a, z)$
  $V(haz) \leftarrow$SIMULATE $(s', \beta', haz, \delta + 1)$
  $B(h) \leftarrow B(h) \cup \beta$
  $N(h) \leftarrow N(h) + 1$
  $N(ha) \leftarrow N(ha) + 1$
  $V(ha) \leftarrow \rho(B(h), a) + \frac{\gamma}{N(ha)}[\sum_z N(haz)V(haz)]$
  $V_h \leftarrow \frac{1}{N(h)}.[rollout(h) + \sum_a N(ha)V(ha)]$
  return V_h

5 Expérimentations

5.1 Bancs d'essai

Les POMDP étant une sous-classe des ρ-POMDP, les premiers bancs d'essai que nous considérons sont le problème du tigre (tel que décrit sur la page POMDP de Cassandra) et de petites instances du problème *Rock Sampling* [18] avec une grille de 4×4 cellules, 4 pierres à échantillonner et une récompense de 100 (respectivement -100) pour l'échantillonnage d'une bonne (resp. mauvaise) pierre.
Il est connu que, dans de nombreux problèmes de collecte d'information, une simple stratégie myope donne souvent de très bons résultats [5]. Pour évaluer les algorithmes proposés, nous considérons des problèmes où celles-ci rencontrent des difficultés.

Nous proposons ainsi le *museum problem* (problème du musée) inspiré par [16], dans lequel un agent doit déterminer la position d'un visiteur dans un environnement torique (4×4 dans nos expérimentations). L'état correspond à la position inconnue du visiteur. À chaque pas de temps, le visiteur reste immobile avec probabilité $0,6$, et bouge vers l'une de ses 4 cellules voisines avec probabilité $0,4$ (en choisissant uniformément au hasard). En guise d'action, l'agent peut activer une caméra dans l'une des cellules possibles. Il reçoit alors une observation déterministe qui peut être *"present"* quand le visiteur est à cette position, *"close"* quand il est sur une cellule voisine, et *"absent"* si le visiteur est plus loin. En outre : la récompense immédiate correspond à la néguentropie de la distribution de probabilité sur la position du visiteur $\rho(b, a, b') = -H(b') = \sum_s b'(s).log(b'(s))$; la croyance initiale sur la position du visiteur est une distribution uniforme sur toutes les cellules ; et γ est fixé à $0,95$. L'intérêt de ce problème repose sur le grand nombre d'actions (une par cellule). Nous avons aussi employé une variante, *Museum Threshold*, comme proposé en section 3.4, où la récompense repose sur une fonction seuil de l'état de croyance (avec $\alpha = 0,8$). La récompense est alors nulle dans la plupart de l'espace de croyance (c.-à-d., partout où $max_s b(s) \leq 0,8$).

Les prochains problèmes appartiennent à la classe des problèmes de (auto-)localisation. L'agent est placé dans une grille torique discrète où les cellules sont colorées en noir ou blanc. À chaque pas de temps, l'agent peut se déplacer vers une cellule voisine. Il reçoit alors une observation correspondant à la couleur de la cellule atteinte. En outre : observations et transitions sont déterministes ; la croyance initiale de l'agent est uniforme sur toutes les cellules blanches de la grille ; la récompense reçue correspond à la différence d'entropie entre b et b' : $\rho(b, a, b') = -H(b') + H(b)$; et $\gamma = 0,9$. Plusieurs configurations ont été étudiées, comme présenté sur la figure 2 : (i) *MazeCross*, où les cellules noires forment une croix, rendant facile l'autolocalisation ; (ii) *MazeLines*, où la grille est divisée en deux régions : la première région contient des rayures et l'autre ne contient qu'une cellule noire ; l'agent ne peut se localiser dans la régions rayée du fait des ambiguïtés et doit planifier plusieurs pas de temps à l'avance pour chercher le point dans la région vide ; (iii) *MazeHole*, où des cellules noires sont séparées par des cellules blanches et une cellule noire manque ; l'agent doit raisonner un pas à l'avance pour chercher la cellule noire manquante dans cette configuration régulière : un agent myope sur une cellule noire ne voit aucun intérêt à se déplacer dans une direction plutôt qu'une autre puisqu'un seul pas ne lui apportera pas d'information ; et (iv) *MazeDots*, qui est la même configuration que MazeHole mais où les cellules noires sont séparées par plusieurs cellules blanches.

Les problèmes de *localisation active* emploient les mêmes

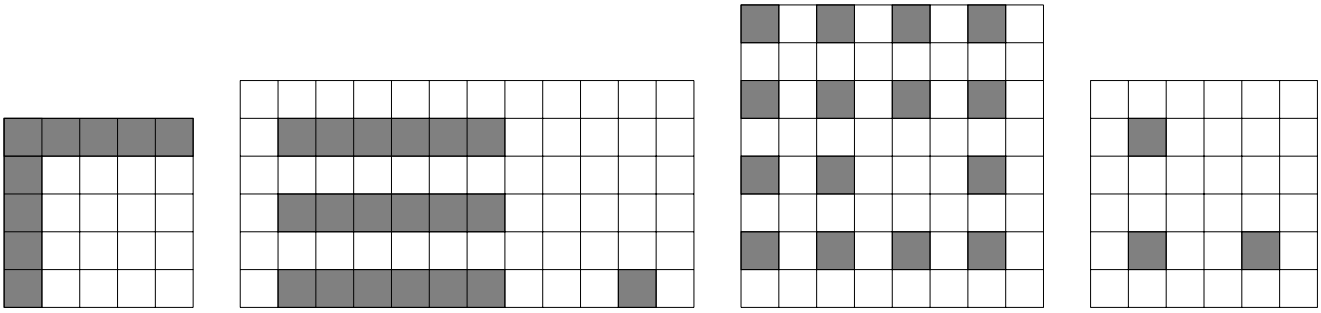

FIGURE 2 – Les diverses configurations de cellules employées pour les problèmes de localisation passive ou active ; de gauche à droite : *MazeCross*, *MazeLines*, *MazeHole* et *MazeDots*.

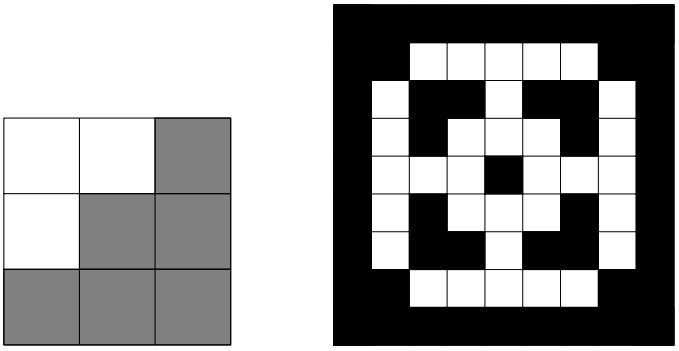

FIGURE 3 – (gauche) la configuration de cellules employée pour *GridX* et *GridNotX*, et (droite) la configuration d'obstacles pour *SeekAndSeek*.

labyrinthes, récompenses et γ que les problèmes de localisation, mais l'agent ne reçoit une information concernant la couleur de sa cellule courante non plus quand il se déplace, mais quand il effectue une action *«observer»* spécifique. La localisation active est ainsi difficile pour les stratégies myopes qui ne voient aucun bénéfice (immédiat) à se déplacer.

GridX et *GridNotX* [10] diffèrent des premiers problèmes de localisation en ce que : (i) la position initiale peut être n'importe quelle cellule (pas juste les blanches) ; (ii) les déplacements (n,s,e,w) réussissent avec une probabilité $0,8$, sinon l'agent reste immobile ; (iii) $\gamma = 0,95$; et (iv) en notant b_x (resp. b_y) la croyance sur l'abscisse x (resp. l'ordonnée y), la fonction de récompense est $\rho(b) = +\|b_x - \frac{1}{3}\mathbf{1}\|_1$ (resp. $-\|b_x - \frac{1}{3}\mathbf{1}\|_1$) pour *GridX* (resp. *GridNotX*). *GridNotX* est un problème particulièrement intéressant parce que son objectif est de maximiser l'incertitude, ce qui ne peut être modélisé avec des récompenses dépendant de l'état.

Dans le problème *SeekAndSeek*, un objet est perdu dans un labyrinthe avec des obstacles restreignant les déplacements possibles de l'agent. L'agent peut se déplacer dans le labyrinthe et reçoit des observations spécifiques selon que l'objet est à la même position (*present*), à côté de celle-ci (*close*), ou dans les autres cas (*absent*). La récompense immédiate reçue est la néguentropie sur les positions possibles de l'objet $\rho(b, a, b') = -H(b') + H(b)$. L'objet étant immobile, quand un agent myope est dans un cul-de-sac, il n'a aucune motivation pour explorer l'environnement puisque revenir en arrière sur ses pas ne lui apporte pas d'information.

Nous avons aussi abordé le problème de diagnostic *CameraClean* [2] qui a été conçu pour empêcher les stratégies myopes d'être efficaces. Dans ce problème une caméra robot peut être orientée pour prendre une photo dans l'une des quatre zones existantes et doit trouver un objet immobile dans l'une d'elles. L'état contient la position de l'objet, la zone cible actuelle, et un booléen spécifiant si la lentille est propre ou non. Les actions consistent pour la caméra à tourner (de manière déterministe) vers la prochaine zone, photographier la zone courante, ou nettoyer sa lentille. Quand la caméra prend une photo, sa qualité dépend du statut de la lentille. Si elle est propre, la probabilité de bien observer la présence de l'objet dans la zone cible est de $0,8$ (la probabilité de faux-positifs et faux-négatifs est de $0,2$), et cette probabilité est de $0,5$ quand la lentille est sale. La récompense est la différence de néguentropie entre les croyances de départ et d'arrivée $\rho(b, a, b') = -H(b') + H(b)$

Toutes les expérimentations ont été conduites sur des cœurs Intel Xeon Gold 6130 à 2.1GHz avec 512Mo.

5.2 Influence des rollouts

Des expérimentations préliminaires ont été conduites avec des rollouts aléatoires. Dans de nombreux problèmes, la politique de rollout aléatoire requiert un important temps

de calcul. Pour un même nombre de descentes, la durée observée pour ρ-POMCP avec des rollouts aléatoires (interrompus quand $\gamma^{depth} < 0.01$) est entre 5 et 10 fois plus élevée que sans aucun rollout (en fixant la valeur des nouveaux nœuds à 0). De plus, dans la majorité des problèmes, utiliser une politique de rollout complètement aléatoire n'a pas d'avantage significatif. Dans certains problèmes, les performances observées sont même dégradées (comme dans le problème du *Tigre*). Nous avons donc préféré nous concentrer d'abord sur ρ-POMCP et ρ-beliefUCT sans rollouts, et observerons l'influence de la politique de rollout en section 5.6.

5.3 Comparaison avec des stratégies myopes

La table 1 présente de premiers résultats comparant la stratégie *aléatoire pure*, les stratégies *look-ahead*, ρ-beliefUCT, et ρ-POMCP sur les problèmes proposés.

Random correspond à une stratégie dans laquelle les actions sont sélectionnées d'une manière complètement aléatoire. Les algorithmes *look-ahead-H* effectuent une programmation dynamique sur tous les futurs possibles pour un horizon temporel fini H en employant le modèle POMDP complet. Ainsi, la stratégie myope pure, dans laquelle l'agent accomplit une action pour maximiser la récompense immédiate, correspond à *Look-ahead-1*, alors que *Look-ahead-3* correspond à une stratégie qui tient compte des toutes les conséquences trois pas de temps à l'avance. La colonne ρ-POMCP correspond à l'algorithme ρ-POMCP avec un nombre fixe de descentes $nb_{descents} = 10\,000$, sans aucun rollout (en fixant la valeur des nouveaux nœuds à 0), avec échantillonnage selon l'importante et $|\beta| = 50$. Enfin, la colonne ρ-beliefUCT correspond à l'algorithme ρ-beliefUCT avec $10\,000$ descentes. ρ-POMCP et ρ-beliefUCT utilisent tous deux une constante UCB spécifique pour chaque problème comme spécifié dans la table 1 (usuellement correspondant à $(R_{max} - R_{min})/(1-\gamma)$).

(a) La première chose à souligner est que, à part pour *GridX* et *Museum*, les problèmes posés requièrent une stratégie plus élaborée qu'une stratégie myope simple puisque *Look-ahead*-3 obtient de meilleurs résultats que *Look-ahead*-1. La stratégie myope donne aussi de bons résultats pour *Museum Threshold* : même si les récompenses non-nulles sont rares, l'existence, pour chaque action, d'une observation qui désambiguise complètement l'état (quand le visiteur est observé par la caméra) est suffisante pour guider une stratégie myope telle que *Look-ahead-1*.

(b) Quand le problème requiert plus qu'une simple stratégie myope, ρ-POMCP et ρ-beliefUCT donnent de meilleurs résultats que *Look-ahead*-1 et des résultats similaires à *Look-ahead*-3. Dans la plupart des cas, comme le montre l'erreur standard, la différence entre *Look-ahead*-1 et *ρ-POMCP* est significative. Dans certains problèmes, la différence entre *Look-ahead*-3 et *ρ-POMCP* est aussi significative (*SeekAndSeek* et *RockSampling* de manière évidente, mais aussi *GridNotX*, *Mazeline* et *ActiveLocLines*).

(c) On notera aussi que ρ-POMCP et ρ-beliefUCT donnent les mêmes résultats pour ce nombre de descentes et prennent beaucoup plus de temps que les simples algorithmes *Look-ahead* (sauf quand le nombre d'actions devient important comme dans *RockSampling* et *Museum*). ρ-beliefUCT est généralement plus rapide que ρ-POMCP, mais cela dépend plus précisément du problème puisque les coûts de calcul de ces deux algorithmes viennent de différentes opérations. Dans ρ-beliefUCT, ce coût est dû au calcul des croyances exactes à chaque fois qu'un nouveau nœud croyance est ajouté à l'arbre. Dans ρ-POMCP, ce coût est dû à la génération de petits sacs β à chaque transition le long des trajectoires. C'est pourquoi, alors que le temps requis par ρ-POMCP est plus régulier (sauf pour le cas non élucidé de *GridX*), le temps requis par ρ-beliefUCT dépend fortement de la croyance considérée et du nombre d'états possibles (par exemple, le calcul de la croyance est rapide dans les problèmes *Tiger* et *CameraClean*, mais requiert plus de temps dans les problèmes de *localisation active* où les états de croyance b incluent de nombreux états s de probabilité non-nulle $b(s) \neq 0$). La section 6 discute du temps de calcul élevé de ρ-POMCP en proposant de possibles améliorations.

Ces premiers résultats suggèrent donc d'abord que les bancs d'essai choisis sont appropriés, et que ρ-POMCP et ρ-beliefUCT sont des approches intéressantes. La prochaine étape a été d'étudier l'influence de la taille des sacs de particules échantillonnés $|\beta|$ pour un budget temps fixe.

5.4 Influence de $|\beta|$

Des expériences ont été conduites pour évaluer l'influence de $|\beta|$, la taille des sacs de particules générés le long des trajectoires. Nous avions l'intuition que, quand $|\beta|$ est petit avec un nombre fixe de descentes, les états de croyance seraient mal approchés et ρ-POMCP donnerait de moins bons résultats qu'avec une grande valeur de $|\beta|$. Toutefois, avec un nombre fixe de 10 000 descentes, il s'avère que $|\beta|$ n'a pas d'influence notable. Avec tant de descentes, la petite valeur de $|\beta|$ est compensée par le grand nombre de trajectoires échantillonnées.

Avec un plus petit nombre de descentes, l'influence de $|\beta|$ apparaît, comme visible table 2, où le nombre de descentes a été fixé à $nb_{descents} = 1000$. Il est difficile d'interpréter précisément ces résultats, mais nous allons essayer de fournir des pistes d'explications générales.

(a) Dans certains problèmes (comme *GridX*, *Museum* et quelques labyrinthes : *MazeCross*, *MazeHole*, *MazeDots* et *ActiveLoc Cross*), on n'observe aucune influence de $|\beta|$ sur la performance. Pour ceux-ci, la performance avec un petit $|\beta|$ est déjà proche de la performance observée table 1 avec une valeur de $|\beta|$ de 50 et 10 000 descentes. Ce sont des problèmes faciles pour lesquels *Look-ahead-1* fournit déjà de bonnes performances ; ils ne requièrent donc pas de bonnes estimations de nœuds croyance profonds pour obtenir une stratégie efficace.

(b) Au contraire, $|\beta|$ a une influence significative dans des problèmes où *Look-ahead-1* a des difficultés, comme

TABLE 1 – Résultats obtenus sur des épisodes de 40 actions en exécutant l'algorithme en-ligne correspondant avant chaque action. V correspond à la performance γ-atténuée moyenne sur tous les épisodes ; Err est l'erreur standard ; nb_{xp} est le nombre d'expériences effectuées ; et $t(s)$ est la durée moyenne d'un épisode.

	Random			Look-ahead-1			Look-ahead-3			ρ-beliefUCT			ρ-POMCP		
Problem (UCB cst)	$V \pm Err$	nb_{xp}	$t(s)$	$V \pm Err$	nb_{xp}	$t(s)$	$V \pm Err$	nb_{xp}	$t(s)$	$V \pm Err$	nb_{xp}	$t(s)$	$V \pm Err$	nb_{xp}	$t(s)$
Tiger (360)	-122.13 ± 2.68	200	0.00	1.86 ± 0.13	200	0.01	2.06 ± 0.12	200	0.04	2.03 ± 0.12	200	9.90	2.05 ± 0.12	200	97.31
CameraClean (14)	0.24 ± 0.01	100	0.02	0.19 ± 0.01	100	0.26	0.55 ± 0.03	100	4.84	0.58 ± 0.03	100	12.44	0.56 ± 0.02	100	92.64
SeekAndSeek (69.3)	-41.04 ± 1.97	100	0.02	-44.02 ± 2.04	100	14.57	-39.86 ± 2.36	100	17.53	-24.66 ± 1.60	100	55.93	-25.29 ± 1.69	100	112.45
Museum entropy (1)	-26.39 ± 0.38	100	0.01	-16.98 ± 0.42	100	0.11	-16.44 ± 0.42	100	103.42	-17.26 ± 0.42	100	38.90	-15.70 ± 0.42	100	73.28
Museum threshold (1)	1.70 ± 0.08	200	0.01	6.36 ± 0.17	200	0.11	6.34 ± 0.16	200	105.55	6.55 ± 0.17	200	46.20	6.32 ± 0.18	200	119.81
GridX (26)	16.38 ± 0.23	100	0.01	21.68 ± 0.06	100	0.02	21.67 ± 0.06	100	0.28	21.69 ± 0.05	100	88.99	21.63 ± 0.05	100	527.64
GridNotX (26)	-16.77 ± 0.14	200	0.01	-4.06 ± 0.16	200	0.02	-3.61 ± 0.15	200	0.24	-3.19 ± 0.14	200	17.93	-3.16 ± 0.15	200	89.39
MazeCross (3.2)	1.63 ± 0.03	200	0.00	2.21 ± 0.01	200	0.02	2.22 ± 0.01	200	0.25	2.20 ± 0.01	200	28.73	2.20 ± 0.01	200	72.78
MazeLines (4.3)	1.52 ± 0.04	200	0.01	2.34 ± 0.03	200	0.03	2.63 ± 0.03	200	0.80	2.73 ± 0.03	200	84.02	2.74 ± 0.03	200	76.99
MazeHole (4.2)	1.27 ± 0.03	200	0.01	1.73 ± 0.05	200	0.03	2.02 ± 0.04	200	0.70	2.04 ± 0.04	200	98.06	1.99 ± 0.04	200	88.26
MazeDots (3.6)	1.35 ± 0.05	200	0.01	1.98 ± 0.04	200	0.03	2.09 ± 0.04	200	0.41	2.16 ± 0.04	200	55.79	2.06 ± 0.04	200	79.26
ActivlocCross (3.2)	0.87 ± 0.03	200	0.01	1.45 ± 0.01	200	0.02	2.07 ± 0.02	200	0.20	2.08 ± 0.02	200	37.61	2.10 ± 0.02	200	49.77
ActivlocLines (4.3)	0.63 ± 0.03	200	0.01	1.17 ± 0.04	200	0.04	2.01 ± 0.04	200	0.63	2.21 ± 0.04	200	97.85	2.25 ± 0.04	200	60.81
ActivlocHole (4.2)	0.40 ± 0.04	200	0.01	0.97 ± 0.07	200	0.02	1.42 ± 0.05	200	0.31	1.37 ± 0.05	200	56.65	1.44 ± 0.05	200	63.73
ActivlocDots (3.6)	0.65 ± 0.03	200	0.01	1.19 ± 0.05	200	0.03	1.79 ± 0.04	200	0.57	1.65 ± 0.03	200	84.42	1.75 ± 0.04	200	68.95
RockSampling 44 (100)	-8.58 ± 4.17	100	0.00	0.00 ± 0.00	100	0.04	0.00 ± 0.00	100	38.00	108.25 ± 7.08	100	20.97	109.96 ± 6.53	100	79.31

TABLE 2 – Influence de $|\beta|$ pour un nombre fixe de 1000 descentes. Chaque colonne correspond aux résultats obtenus sur 200 expériences en exécutant ρ-POMCP avec différentes valeurs de $|\beta|$ sur des épisodes de 40 actions. V est la valeur γ-atténuée cumulée moyenne, Err l'erreur standard, et $t(s)$ la durée moyenne d'une expérience.

	$\lvert\beta\rvert = 0$		$\lvert\beta\rvert = 1$		$\lvert\beta\rvert = 2$		$\lvert\beta\rvert = 5$		$\lvert\beta\rvert = 10$		$\lvert\beta\rvert = 50$		$\lvert\beta\rvert = 100$	
Problem (UCB cst)	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$
Tiger (360)	0.18 ± 0.23	0.55	1.76 ± 0.12	0.71	-0.11 ± 0.15	0.84	-3.95 ± 0.03	1.23	-4.00 ± 0.00	1.84	-4.00 ± 0.00	6.63	-4.00 ± 0.00	12.01
CameraClean (14)	0.17 ± 0.01	0.57	0.21 ± 0.01	0.66	0.32 ± 0.01	0.76	0.49 ± 0.01	1.05	0.55 ± 0.02	1.55	0.58 ± 0.02	6.81	0.56 ± 0.02	12.88
SeekAndSeek (69.3)	-33.79 ± 1.57	0.70	-33.45 ± 1.59	0.75	-35.33 ± 1.44	0.91	-33.96 ± 1.47	1.31	-32.51 ± 1.45	1.94	-31.17 ± 1.27	8.06	-29.01 ± 1.28	14.51
Museum entropy (1)	-17.06 ± 0.31	0.57	-17.18 ± 0.30	0.60	-16.57 ± 0.31	0.58	-17.21 ± 0.32	0.79	-17.15 ± 0.30	1.15	-16.87 ± 0.29	5.53	-17.16 ± 0.31	10.38
Museum threshold (1)	6.05 ± 0.16	0.61	5.95 ± 0.19	0.70	6.08 ± 0.19	0.78	6.29 ± 0.18	0.98	5.82 ± 0.16	1.48	6.01 ± 0.18	6.62	6.25 ± 0.18	12.28
GridX (26)	21.64 ± 0.04	0.47	21.61 ± 0.04	0.60	21.70 ± 0.04	0.71	21.68 ± 0.04	0.99	21.61 ± 0.04	1.49	21.69 ± 0.04	6.31	21.67 ± 0.04	11.41
GridNotX (26)	-3.78 ± 0.17	0.46	-3.83 ± 0.16	0.58	-3.72 ± 0.16	0.70	-3.76 ± 0.15	0.95	-3.58 ± 0.16	1.45	-3.56 ± 0.16	6.03	-3.35 ± 0.15	11.22
MazeCross (3.2)	2.20 ± 0.01	0.51	2.18 ± 0.01	0.60	2.20 ± 0.01	0.66	2.21 ± 0.01	0.99	2.18 ± 0.01	1.51	2.22 ± 0.01	5.66	2.20 ± 0.01	10.40
MazeLines (4.3)	2.46 ± 0.03	0.59	2.48 ± 0.03	0.70	2.53 ± 0.03	0.82	2.54 ± 0.03	1.14	2.64 ± 0.03	1.73	2.67 ± 0.03	6.03	2.64 ± 0.03	11.08
MazeHole (4.2)	1.91 ± 0.05	0.65	1.87 ± 0.05	0.76	1.93 ± 0.04	0.90	1.98 ± 0.04	1.23	2.02 ± 0.04	1.84	1.98 ± 0.04	6.66	2.01 ± 0.04	12.17
MazeDots (3.6)	2.01 ± 0.04	0.54	2.02 ± 0.04	0.67	2.01 ± 0.04	0.77	2.05 ± 0.04	1.06	2.05 ± 0.04	1.62	2.05 ± 0.04	6.12	2.04 ± 0.04	11.31
ActivlocCross (3.2)	2.09 ± 0.02	0.50	2.09 ± 0.02	0.55	2.09 ± 0.02	0.62	2.08 ± 0.02	0.83	2.12 ± 0.02	1.19	2.12 ± 0.02	3.85	2.10 ± 0.02	7.03
ActivlocLines (4.3)	1.83 ± 0.04	0.62	1.93 ± 0.04	0.71	1.95 ± 0.04	0.82	2.07 ± 0.04	1.10	2.08 ± 0.04	1.50	2.10 ± 0.04	5.29	2.08 ± 0.04	9.33
ActivlocHole (4.2)	1.34 ± 0.02	0.67	1.40 ± 0.03	0.79	1.41 ± 0.03	0.94	1.49 ± 0.03	1.36	1.57 ± 0.04	1.89	1.63 ± 0.03	6.12	1.67 ± 0.03	10.42
ActivlocDots (3.6)	1.22 ± 0.05	0.59	1.14 ± 0.05	0.72	1.24 ± 0.05	0.82	1.31 ± 0.05	1.09	1.36 ± 0.05	1.57	1.43 ± 0.05	5.10	1.52 ± 0.06	9.49
RockSampling 44 (100)	100.43 ± 4.76	0.53	90.86 ± 4.67	0.52	96.86 ± 4.50	0.61	94.65 ± 4.81	0.93	86.29 ± 4.70	1.50	72.59 ± 4.43	7.37	70.51 ± 4.46	13.09
RockSampling 44 (800)	56.29 ± 3.72	0.53	60.71 ± 3.89	0.52	50.18 ± 3.75	0.60	64.28 ± 4.15	0.92	65.59 ± 4.24	1.53	60.75 ± 4.00	7.41	50.26 ± 3.72	13.22

dans les problèmes de localisation active *ActivLoc Lines*, *ActivLoc Dots*, *GridNotX*, *SeekAndSeek* et *CameraClean*. Dans ce cas, le petit nombre de descentes ne permet pas à ρ-POMCP de construire de bonnes approximations des vrais états de croyance et des valeurs des nœuds croyance. En ce qui concerne plus spécifiquement *GridNotX*, ρ est concave et de petites valeurs de $|\beta|$ pourraient empêcher MCTS de se concentrer sur une action intéressante. Dans ce cas, les récompenses, et donc les valeurs, vont être initialement sous-évaluées et, dans certaines situations, cela empêchera l'algorithme d'essayer l'action optimale même si, sur le long terme, ce phénomène sera compensé par le bonus d'exploration.

(c) De façon surprenante, dans certains cas (comme *RockSampling 44* et *Tiger*), on observe que de petites valeurs de $|\beta|$ donnent de meilleurs résultats. Nous comptons étudier ce phénomène, mais une explication possible serait que de petites valeurs de $|\beta|$ donnent parfois des sur-estimations importantes des valeurs de nœuds croyance, ce qui favorise l'exploitation des branches associées et donc une recherche plus profonde.

La prochaine étape consiste en l'analyse des résultats avec un budget temps fixe pour voir si de petites valeurs de $|\beta|$ sont compensées par la possibilité d'échantillonner plus de trajectoires.

5.5 Résultats avec budget temps fixe

Les tables 3 et 4 (voir annexe) présentent des résultats concernant l'influence de $|\beta|$ avec un budget temps fixe (respectivement de 1s et 100ms).

Avec un budget temps fixe et un petit $|\beta|$, seules peu de particules sont ajoutées aux sacs cumulés $B(h)$ des nœuds croyance visités lors d'une trajectoire. Toutefois, dans ce cas, ρ-POMCP effectue de nombreuses descentes, ajoutant

de nombreuses particules sur le long terme, et peut donc obtenir de bonnes estimations des états de croyance. À l'inverse, quand $|\beta|$ est grand, chaque descente ajoute de nombreuses particules dans les nœuds croyances rencontrés, les états de croyance sont plus précisément approchés, et la sélection des trajectoires échantillonnées devrait être guidée par des valeurs estimées plus précises, mais l'algorithme effectue moins de descentes.

Dans les deux cas (budget temps de 100ms et 1s), quand $|\beta|$ croît, le nombre de descentes effectuées par ρ-POMCP décroît naturellement, mais cela ne semble pas avoir d'impact sur les valeurs cumulées γ-atténuées à l'exécution. Pour de très grandes valeurs de $|\beta|$ avec 100ms de budget temps, les expériences n'atteignent pas toutes la fin de l'épisode. Cela arrive quand une action réelle génère une observation qui n'a pas été échantillonnée dans l'arbre. Dans ce cas, l'algorithme n'a pas d'estimation de la prochaine croyance racine et s'arrête (voir discussion pour des propositions concernant ce point). Ce phénomène s'observe dans des problèmes avec de nombreuses actions (comme *Museum*) puisque, dans ce cas, le nombre de descentes est très faible par rapport au nombre d'actions à explorer.

Des comportements différents s'observent dans deux problèmes. Dans *CameraClean*, il y a une différence de performance significative entre $|\beta| = 0$ et $|\beta| = 5$ avec un budget temps de 100ms. Cela pourrait venir du processus d'observation hautement stochastique de *CameraClean*, lequel nécessiterait de meilleures estimations de l'état de croyance pour agir correctement. Dans *Rocksampling*, on observe, comme quand ρ-POMCP effectue un nombre fixe de descentes, que de petites valeurs de $|\beta|$ donnent de meilleurs résultats. Cela pourrait être dû à des sur-estimations favorisant l'exploitation et au fait que de petits valeurs de $|\beta|$ avec un budget de temps fixe favorisent un grand nombre de descentes avec une plus grande profondeur. Mais ce cas précis doit encore être étudié plus en détails.

5.6 Variantes de ρ-POMCP

Nous avons enfin regardé les résultats pour différentes variantes de ρ-POMCP avec budget temps fixe. Ces variantes correspondent à différents types de rollouts : les rollouts aléatoires (les actions étant choisies aléatoirement) et les rollouts myopes (les action étant choisies selon une politique *look-ahead-1*—sélectionnant une action maximisant la récompense immédiate). Elles correspondent aussi à *Last-Reward update* et *Last-Value update* comme présenté en section 4.3. Nous nous sommes concentrés sur un budget temps de 500ms afin d'avoir assez de temps pour exécuter ρ-POMCP avec suffisamment de descentes pour les rollouts coûteux (tels que les myopes).

La table 5 (voir annexe) présente les résultats obtenus avec différents rollouts. Comme attendu, le nombre de descentes décroît quand $|\beta|$ croît. Aussi, pour une même valeur de $|\beta|$, le nombre de descentes décroît de *constant-rollout* à *random-rollout* et de *random-rollout* à *myopic-rollout* à cause du temps nécessaire à l'exécution des rollouts complexes. Dans le problème *museum entropy*, du fait du petit nombre de descentes, plusieurs épisodes se terminent sur une observation non-anticipée.

La table 6 (voir annexe) présente des résultats obtenus avec ρ-POMCP de base contre les versions avec mise à jour *lru* et *lvu* pour différentes valeurs de $|\beta|$. Nous nous attendions à ce que les mises à jour *lru* et *lvu* donnent de meilleurs résultats que la mise à jour ρ-POMCP de base, en particulier quand $|\beta|$ est petit, mais, comme le montre cette table, ce phénomène n'est pas observé ici.

6 Discussion

Dans cet article, nous avons proposé deux algorithmes, ρ-POMCP et ρ-beliefUCT, afin d'aborder la résolution de ρ-POMDP sans contrainte sur la fonction de récompense ρ. Des expérimentations simples ont montré que ρ-POMCP et ρ-beliefUCT donnent des résultats intéressants dans ces problèmes, mais qu'on n'observe pas de différences significatives avec plusieurs variantes de ρ-POMCP (modifiant les rollouts ou les mises à jour). Ces résultats ouvrent cependant plusieurs directions de recherche.

$|\beta|$ dynamique pour ρ-POMCP Nous avons observé que ρ-POMCP est coûteux en temps de calcul du fait des nombreuses particules échantillonnées Pourtant, après un certain nombre d'itérations, les nœuds croyance proches de la racine pourraient contenir assez de particules pour estimer leur état de croyance précisément. Une direction prometteuse serait d'économiser du temps en faisant croître les sacs cumulés $B(h)$ sous-linéairement (plutôt que linéairement) avec le nombre de visites $N(h)$. Cela nécessitera de modifier la façon dont les particules sont générées, ce qui n'est pas une tâche aisée puisque les particules sont générées du nœud croyance racine et doivent être propagées jusqu'à la fin de chaque trajectoire de façon (i) à exécuter une politique de rollout non-aléatoire et (ii) à construire de meilleures estimations des états de croyance aux nœuds feuilles.

À propos des transitions réelles imprévues Un autre problème qui concerne aussi POMCP et le filtrage particulaire est de gérer les transitions imprévues. Si, après avoir effectivement accompli une action a, l'observation réelle z a été rarement (voire pas du tout) échantillonnée, la nouvelle racine pourrait n'être dotée que d'une mauvaise estimation de la croyance, voire ne pas exister du tout, de sorte que les calculs en ligne donneront de mauvais résultats s'ils peuvent seulement être accomplis. Dans ce cas, effectuer un filtrage particulaire depuis le père du nouveau nœud racine pourrait fournir une nouvelle croyance racine mais à un coût de calcul élevé. Une autre amélioration complémentaire serait d'échantillonner régulièrement des particules de la croyance initiale du POMDP en suivant la trajectoire réelle suivie par l'agent pour ajouter de nouvelles particules à la racine courante et se prémunir de l'appauvrissement le long des épisodes.

Travaux futurs Nous espérons que les approches MCTS telles que ρ-POMCP fourniront une manière d'affronter des problèmes avec des grands nombres d'états, d'actions ou d'observations. Les travaux futurs incluent le fait des considérer des ρ-POMDP continus (c.-à-d. avec des états, actions ou observations continues), par exemple en partant des travaux de SUNBERG et KOCHENDERFER, SUNBERG et KOCHENDERFER [20, 21], afin d'aborder des problèmes de contrôle orientés information dans un contexte robotique.

Remerciements Les expériences présentées dans cet article ont été menées sur Grid5000, qui bénéficie du support d'un groupe d'intérêt scientifique hébergé par Inria, et incluant le CNRS, RENATER et quelques universités et organisations (voir https://www.grid5000.fr). Enfin, nous remercions les relecteurs pour leurs remarques constructives intégrées dans cet article.

Références

[1] M. ARAYA-LÓPEZ, O. BUFFET, V. THOMAS et F. CHARPILLET. « A POMDP Extension with Belief-dependent Rewards ». In : *Advances in Neural Information Processing Systems 23 (NIPS-10)*. [see also companion techreport]. 2010.

[2] M. ARAYA-LÓPEZ. « Near-Optimal Algorithms for Sequential Information-Gathering Decision Problems ». Thèse de doct. Université de Lorraine, fév. 2013.

[3] K. ÅSTRÖM. « Optimal control of Markov processes with incomplete state information ». In : *Journal of Mathematical Analysis and Applications* 10.1 (1965). ISSN : 0022-247X.

[4] R. BELLMAN. « A Markovian Decision Process ». In : *Journal of Mathematics and Mechanics* 6.5 (1957).

[5] M. BONNEAU, N. PEYRARD et R. SABBADIN. « A Reinforcement-Learning Algorithm for Sampling Design in Markov Random Fields ». In : 2012.

[6] C. BROWNE et al. « A Survey of Monte Carlo Tree Search Methods ». In : *IEEE Transactions on Computational Intelligence and AI in Games* 4.1 (2012).

[7] R. COULOM. « Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search ». In : *Proceedings of the Fifth International Conference on Computer and Games (CG-2006)*. 2006.

[8] A. DOUCET, S. GODSILL et A. CHRISTOPHE. « On sequential Monte Carlo sampling methods for Bayesian filtering ». In : *Statistics and Computing* 10.3 (2000), p. 197–208.

[9] M. EGOROV, M. J. KOCHENDERFER et J. J. UUDMAE. « Target Surveillance in Adversarial Environments using POMDPs ». In : *AAAI-16*. 2016.

[10] M. FEHR, O. BUFFET, V. THOMAS et J. DIBANGOYE. « ρ-POMDPs have Lipschitz-Continuous ϵ-Optimal Value Functions ». In : *Advances in Neural Information Processing Systems 32 (NIPS-18)*. 2018.

[11] D. FOX, W. BURGARD et S. THRUN. « Active Markov Localization for Mobile Robots ». In : *Robotics and Autonomous Systems* 25.3–4 (1998). DOI : http://dx.doi.org/10.1016/S0921-8890(98)00049-9.

[12] M. KEARNS, Y. MANSOUR et A. NG. « A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes ». In : *Machine Learning* 49 (2002).

[13] L. KOCSIS et C. SZEPESVARI. « Bandit based Monte-Carlo Planning ». In : *ECML-06*. 2006.

[14] L. MIHAYLOVA, T. LEFEBVRE, H. BRUYNINCKX et J. D. SCHUTTER. In : *NATO Science Series on Data Fusion for Situation Monitoring, Incident Detection, Alert and Response Management*. T. 198. 2006. Chap. Active Robotic Sensing as Decision Making with Statistical Methods.

[15] M. L. PUTERMAN. *Markov Decision Processes – Discrete Stochastic Dynamic Programming*. 1994.

[16] Y. SATSANGI, S. WHITESON et M. T. J. SPAAN. *An Analysis of Piecewise-Linear and Convex Value Functions for Active Perception POMDPs*. Rapp. tech. IAS-UVA-15-01. IAS, Universiteit van Amsterdam, 2015.

[17] D. SILVER et J. VENESS. « Monte-Carlo Planning in Large POMDPs ». In : *NIPS-10*. 2010.

[18] T. SMITH et R. SIMMONS. « Heuristic Search Value Iteration for POMDPs ». In : *UAI-04*. 2004.

[19] M. T. SPAAN, T. S. VEIGA et P. U. LIMA. « Decision-theoretic planning under uncertainty with information rewards for active cooperative perception ». In : *JAAMAS* 29.6 (2015).

[20] Z. N. SUNBERG et M. J. KOCHENDERFER. « Online Algorithms for POMDPs with Continuous State, Action, and Observation Spaces ». In : *ICAPS-18*. 2018.

[21] Z. SUNBERG et M. J. KOCHENDERFER. « POMCPOW : An online algorithm for POMDPs with continuous state, action, and observation spaces ». In : *CoRR* abs/1709.06196 (2017). URL : http://arxiv.org/abs/1709.06196.

TABLE 3 – Influence de $|\beta|$ pour budget temps fixe de 1 seconde. Chaque colonne correspond aux résultats obtenus par ρ-POMCP pour différentes valeurs de $|\beta|$. Chaque valeur est le résultat de 200 expériences conduites sur des épisodes de 40 actions. V est la valeur cumulée γ-atténuée moyenne, Err l'erreur standard, nb_{xp} le nombre d'expériences accomplissant effectivement les 40 actions, et nb_d le nombre de descentes effectuées en 1 seconde.

	$\|\beta\|=0$			$\|\beta\|=1$			$\|\beta\|=5$			$\|\beta\|=10$			$\|\beta\|=20$			$\|\beta\|=50$			$\|\beta\|=100$			$\|\beta\|=500$		
Problem (UCB cst)	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d
Tiger (360)	-13.33 ± 0.07	200	17764	1.34 ± 0.18	200	16518	1.96 ± 0.12	200	13060	1.82 ± 0.12	200	9496	2.05 ± 0.12	200	6382	0.88 ± 0.13	200	3515	-1.41 ± 0.12	200	2159	-4.00 ± 0.00	200	621
CameraClean (14)	0.23 ± 0.02	200	20136	0.23 ± 0.02	200	17985	0.46 ± 0.02	200	13537	0.54 ± 0.02	200	9623	0.53 ± 0.02	200	6443	0.59 ± 0.02	200	3282	0.56 ± 0.02	200	2020	0.54 ± 0.02	200	545
SeekAndSeek (69.3)	-29.15 ± 1.29	200	11626	-27.51 ± 1.29	200	10971	-27.65 ± 1.27	200	8842	-28.88 ± 1.26	200	6919	-27.90 ± 1.25	200	4608	-30.32 ± 1.19	200	2383	-27.01 ± 1.25	200	1367	-29.63 ± 1.30	200	356
Museum entropy (1)	-16.67 ± 0.29	200	12100	-16.75 ± 0.27	200	10788	-16.58 ± 0.31	200	9007	-16.17 ± 0.28	200	7558	-16.83 ± 0.29	200	5714	-16.08 ± 0.28	200	3329	-16.75 ± 0.29	200	2030	-16.92 ± 0.29	199	599
Museum threshold (1)	6.16 ± 0.17	200	13831	6.00 ± 0.17	200	12580	5.95 ± 0.16	200	11065	6.12 ± 0.16	200	8675	6.13 ± 0.16	200	6300	6.26 ± 0.16	200	3513	6.45 ± 0.17	200	2141	6.29 ± 0.17	200	609
MazeCross (3.2)	2.21 ± 0.01	200	16140	2.20 ± 0.01	200	14946	2.21 ± 0.01	200	12594	2.23 ± 0.01	200	10028	2.19 ± 0.01	200	6806	2.19 ± 0.01	200	3492	2.20 ± 0.01	200	2058	2.21 ± 0.01	200	538
MazeLines (4.3)	2.67 ± 0.03	200	11413	2.68 ± 0.03	200	10161	2.73 ± 0.03	200	8427	2.70 ± 0.03	200	6826	2.69 ± 0.02	200	4981	2.73 ± 0.03	200	2582	2.65 ± 0.03	200	1421	2.69 ± 0.03	200	371
MazeHole (4.2)	2.03 ± 0.04	200	12215	2.05 ± 0.04	200	11596	2.02 ± 0.04	200	9485	2.10 ± 0.04	200	7693	1.92 ± 0.04	200	5384	2.04 ± 0.04	200	2722	1.99 ± 0.04	200	1537	2.03 ± 0.04	200	390
MazeDots (3.6)	2.07 ± 0.04	200	12167	2.09 ± 0.04	200	10553	2.07 ± 0.04	200	8648	2.08 ± 0.03	200	7065	2.04 ± 0.04	200	5018	2.11 ± 0.04	200	2633	2.10 ± 0.04	200	1518	2.07 ± 0.04	200	382
Activloc Cross (3.2)	2.10 ± 0.02	200	13072	2.10 ± 0.02	200	11930	2.13 ± 0.02	200	10613	2.13 ± 0.02	200	9124	2.13 ± 0.02	200	6848	2.09 ± 0.02	200	3835	2.10 ± 0.02	200	2297	2.09 ± 0.02	200	619
ActivLoc Lines (4.3)	2.15 ± 0.04	200	8462	2.09 ± 0.03	200	8086	2.21 ± 0.04	200	6810	2.15 ± 0.03	200	5921	2.11 ± 0.04	200	4469	2.14 ± 0.04	200	2375	2.16 ± 0.04	200	1333	2.11 ± 0.04	200	334
ActivLoc Hole (4.2)	1.66 ± 0.03	200	9261	1.71 ± 0.04	200	8503	1.75 ± 0.04	200	7503	1.65 ± 0.03	200	6379	1.73 ± 0.03	200	4593	1.73 ± 0.03	200	2521	1.69 ± 0.03	200	1403	1.71 ± 0.03	200	356
Activloc Dots (3.6)	1.37 ± 0.05	200	11384	1.45 ± 0.06	200	10485	1.37 ± 0.05	200	9077	1.43 ± 0.05	200	7613	1.42 ± 0.05	200	5468	1.38 ± 0.05	200	3059	1.39 ± 0.05	200	1652	1.47 ± 0.05	200	466
GridX (26)	21.64 ± 0.04	200	24773	21.68 ± 0.04	200	22485	21.65 ± 0.04	200	17109	21.67 ± 0.04	200	12633	21.67 ± 0.04	200	7653	21.66 ± 0.04	200	4220	21.66 ± 0.04	200	2431	21.69 ± 0.04	200	656
GridNotX (26)	-3.29 ± 0.15	200	21832	-3.31 ± 0.15	200	19934	-3.24 ± 0.15	200	15571	-3.34 ± 0.15	199	11177	-3.51 ± 0.15	200	7622	-3.29 ± 0.15	199	4127	-3.41 ± 0.15	200	2472	-3.56 ± 0.16	200	687
RockSampling44 (100)	115.92 ± 4.79	200	11414	107.73 ± 4.67	200	9898	104.46 ± 4.43	200	8591	106.05 ± 5.04	200	6249	108.23 ± 4.82	200	4424	95.38 ± 4.66	200	2483	89.22 ± 4.67	200	1444	59.21 ± 3.95	200	450
RockSampling44 (800)	77.87 ± 3.92	200	12533	66.09 ± 3.65	200	11583	62.69 ± 3.66	200	9689	51.50 ± 3.45	200	7493	57.01 ± 3.40	200	5111	57.20 ± 3.37	200	2800	53.97 ± 3.39	200	1695	61.91 ± 3.77	200	454

TABLE 4 – Influence de $|\beta|$ pour un budget temps fixe de 100 ms.

	$\|\beta\|=0$			$\|\beta\|=1$			$\|\beta\|=5$			$\|\beta\|=10$			$\|\beta\|=20$			$\|\beta\|=50$			$\|\beta\|=100$			$\|\beta\|=500$		
Problem (UCB cst)	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d	$V \pm Err$	nb_{xp}	nb_d
Tiger (360)	0.97 ± 0.18	200	955	1.73 ± 0.13	200	829	0.28 ± 0.12	200	671	-3.31 ± 0.06	200	425	-4.00 ± 0.00	200	272	-3.71 ± 0.07	200	158	-1.31 ± 0.13	200	91	1.37 ± 0.14	198	27
CameraClean (14)	0.23 ± 0.02	200	1509	0.22 ± 0.01	200	1224	0.46 ± 0.01	200	912	0.53 ± 0.02	200	661	0.52 ± 0.02	200	394	0.55 ± 0.02	200	226	0.53 ± 0.02	200	130	0.54 ± 0.02	199	42
SeekAndSeek (69.3)	-34.16 ± 1.44	195	1127	-34.65 ± 1.44	198	973	-31.15 ± 1.41	200	670	-33.57 ± 1.33	200	518	-34.16 ± 1.34	199	347	-31.51 ± 1.40	198	171	-33.02 ± 1.36	196	100	-35.29 ± 1.50	188	30
Museum entropy (1)	-16.95 ± 0.29	200	913	-17.12 ± 0.30	198	875	-17.17 ± 0.30	200	864	-16.83 ± 0.29	200	653	-17.02 ± 0.32	200	475	-16.78 ± 0.30	197	252	-17.75 ± 0.31	195	147	-18.36 ± 0.40	143	45
Museum threshold (1)	6.35 ± 0.16	200	756	5.99 ± 0.16	199	669	6.30 ± 0.17	200	638	6.20 ± 0.17	200	596	6.35 ± 0.16	199	366	5.96 ± 0.16	194	218	6.19 ± 0.18	189	127	5.41 ± 0.19	164	45
GridX (26)	21.64 ± 0.04	200	1480	21.67 ± 0.04	199	1295	21.61 ± 0.04	200	885	21.66 ± 0.04	200	677	21.75 ± 0.04	200	428	21.63 ± 0.04	199	226	21.62 ± 0.04	200	132	21.61 ± 0.05	189	40
GridNotX (26)	-3.70 ± 0.16	200	1535	-3.68 ± 0.15	200	1211	-3.36 ± 0.15	199	958	-3.62 ± 0.15	199	649	-3.60 ± 0.16	199	488	-3.76 ± 0.15	199	223	-3.52 ± 0.16	197	148	-3.42 ± 0.18	181	41
MazeCross (3.2)	2.20 ± 0.01	199	1091	2.19 ± 0.01	199	928	2.23 ± 0.01	200	709	2.19 ± 0.01	200	532	2.22 ± 0.01	200	408	2.21 ± 0.01	199	209	2.20 ± 0.01	200	121	2.21 ± 0.01	194	35
MazeLines (4.3)	2.61 ± 0.03	188	722	2.50 ± 0.03	181	635	2.57 ± 0.03	195	517	2.59 ± 0.03	196	383	2.58 ± 0.03	197	269	2.58 ± 0.03	198	142	2.61 ± 0.03	199	84	2.49 ± 0.03	184	24
MazeHole (4.2)	2.03 ± 0.05	189	879	2.03 ± 0.04	195	808	2.02 ± 0.04	197	592	1.99 ± 0.04	200	451	1.99 ± 0.04	199	316	1.96 ± 0.04	199	164	1.95 ± 0.04	198	100	1.96 ± 0.04	189	28
MazeDots (3.6)	2.01 ± 0.04	190	805	2.06 ± 0.04	194	747	2.07 ± 0.04	199	555	2.07 ± 0.04	200	381	1.96 ± 0.03	198	275	2.15 ± 0.04	199	151	2.03 ± 0.04	196	100	2.05 ± 0.04	189	29
ActivlocCross (3.2)	2.09 ± 0.02	200	1005	2.06 ± 0.02	200	873	2.11 ± 0.02	200	678	2.12 ± 0.02	200	508	2.09 ± 0.02	200	410	2.09 ± 0.02	200	245	2.08 ± 0.02	200	154	2.12 ± 0.02	199	47
ActivlocLines (4.3)	2.03 ± 0.04	200	755	1.95 ± 0.04	198	671	2.02 ± 0.04	200	580	2.07 ± 0.03	200	466	2.09 ± 0.04	200	314	2.07 ± 0.04	200	165	2.01 ± 0.04	200	95	2.05 ± 0.04	183	30
ActivlocHole (4.2)	1.50 ± 0.03	198	856	1.47 ± 0.03	195	810	1.56 ± 0.03	200	697	1.55 ± 0.03	200	535	1.59 ± 0.03	200	358	1.59 ± 0.03	199	177	1.60 ± 0.03	199	103	1.56 ± 0.03	181	34
ActivlocDots (3.6)	1.30 ± 0.05	200	1010	1.42 ± 0.05	199	858	1.45 ± 0.05	200	670	1.40 ± 0.05	200	492	1.39 ± 0.05	200	377	1.34 ± 0.05	200	220	1.41 ± 0.05	199	123	1.31 ± 0.05	188	42
RockSampling 44 (100)	104.56 ± 4.37	200	1055	109.47 ± 5.01	200	902	97.97 ± 4.29	200	730	90.61 ± 4.57	200	516	85.99 ± 4.70	200	315	68.58 ± 4.16	200	154	69.35 ± 4.04	199	82	31.71 ± 2.48	198	24
RockSampling 44 (800)	67.71 ± 4.23	200	1381	64.91 ± 4.28	200	1224	66.05 ± 3.90	200	787	72.71 ± 4.19	200	553	62.15 ± 4.13	200	354	53.58 ± 3.76	199	155	64.06 ± 3.93	198	88	39.45 ± 2.88	197	26

TABLE 5 – Résultats obtenus avec différents rollout sur ρ-POMCP avec un budget temps fixe de 500ms et plusieurs valeurs de $|\beta|$. La colonne *No Rollout* correspond à une valeur de rollout constante (0), les colonnes *Random* à des rollouts aléatoires (actions sélectionnées aléatoirement pendant l'étape de simulation) et les colonnes *Myopic* à des rollouts myopes (actions sélectionnées selon une stratégie *Look-ahead-1*. Chaque valeur correspond au résultat aggrégé de 200 expériences.

	No rollout $\|\beta\| = 1$		No rollout $\|\beta\| = 10$		No rollout $\|\beta\| = 50$		Random $\|\beta\| = 1$		Random $\|\beta\| = 10$		Random $\|\beta\| = 50$		Myopic $\|\beta\| = 1$		Myopic $\|\beta\| = 10$		myopic $\|\beta\| = 50$	
Problem (UCB cst)	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d
Museum entropy (1)	-17.33 ± 0.41	5049	-16.88 ± 0.44	2834	-16.63 ± 0.46	1367	-17.27 ± 0.41	819	-17.86 ± 0.49	281	-20.99 ± 0.43	67	- ± -	-	- ± -	-	- ± -	-
SeekAndSeek (69.3)	-30.98 ± 1.87	5013	-25.53 ± 1.88	2911	-29.80 ± 1.73	1096	-36.71 ± 1.92	1076	-29.53 ± 1.97	420	-28.24 ± 1.85	104	-37.91 ± 2.18	182	-31.45 ± 2.05	119	-35.75 ± 1.83	54
CameraClean (14)	0.25 ± 0.03	9679	0.58 ± 0.03	4025	0.57 ± 0.02	1496	0.20 ± 0.02	1862	0.42 ± 0.02	810	0.55 ± 0.02	205	0.21 ± 0.02	424	0.45 ± 0.02	297	0.54 ± 0.02	109
MazeCross (3.2)	2.21 ± 0.01	5753	2.20 ± 0.01	3419	2.17 ± 0.01	1438	2.20 ± 0.01	1675	2.18 ± 0.01	811	2.23 ± 0.02	211	2.20 ± 0.01	425	2.18 ± 0.01	316	2.19 ± 0.01	126
MazeLines (4.3)	2.63 ± 0.04	5484	2.70 ± 0.04	3437	2.66 ± 0.04	1154	2.62 ± 0.04	1670	2.60 ± 0.04	731	2.60 ± 0.04	203	2.45 ± 0.05	427	2.64 ± 0.04	276	2.56 ± 0.04	118
MazeHole (4.2)	2.11 ± 0.06	4362	2.02 ± 0.05	2892	2.07 ± 0.06	1107	1.95 ± 0.06	1696	2.06 ± 0.06	789	2.00 ± 0.06	195	1.96 ± 0.07	411	1.91 ± 0.05	271	1.89 ± 0.06	111
MazeDots (3.6)	2.02 ± 0.05	4023	2.07 ± 0.05	2598	2.00 ± 0.05	1054	2.00 ± 0.05	1590	2.11 ± 0.05	784	2.11 ± 0.06	198	2.01 ± 0.05	419	2.06 ± 0.05	294	2.02 ± 0.06	118
ActivlocCross (3.2)	2.10 ± 0.03	4774	2.12 ± 0.03	3364	2.15 ± 0.03	1635	2.14 ± 0.03	2095	2.13 ± 0.03	1095	2.08 ± 0.03	357	2.10 ± 0.03	513	2.08 ± 0.03	430	2.09 ± 0.03	184
ActivlocLines (4.3)	2.13 ± 0.06	3356	2.10 ± 0.05	2309	2.14 ± 0.05	1057	1.96 ± 0.05	1723	2.07 ± 0.05	980	2.13 ± 0.05	296	1.99 ± 0.05	484	2.13 ± 0.05	385	2.04 ± 0.06	146
ActivlocHole (4.2)	1.63 ± 0.04	3856	1.67 ± 0.04	2656	1.66 ± 0.04	1120	1.58 ± 0.05	1864	1.64 ± 0.05	972	1.54 ± 0.04	308	1.42 ± 0.04	472	1.53 ± 0.04	384	1.53 ± 0.04	148
ActivlocDots (3.6)	1.59 ± 0.08	4514	1.41 ± 0.07	2977	1.40 ± 0.07	1332	1.35 ± 0.07	1973	1.32 ± 0.07	1062	1.31 ± 0.06	317	1.19 ± 0.08	474	1.44 ± 0.07	397	1.30 ± 0.07	161
GridNotX (26)	-3.50 ± 0.22	8651	-3.24 ± 0.22	4293	-3.29 ± 0.21	1806	-3.66 ± 0.18	1082	-3.61 ± 0.22	410	-4.01 ± 0.26	103	-3.60 ± 0.25	124	-3.78 ± 0.25	76	-3.50 ± 0.24	46
GridX (26)	21.67 ± 0.05	10270	21.75 ± 0.06	5089	21.77 ± 0.06	1750	21.64 ± 0.06	1083	21.66 ± 0.07	397	21.46 ± 0.08	104	21.45 ± 0.14	120	21.41 ± 0.07	77	21.43 ± 0.08	48
RockSampling44 (100)	113.53 ± 6.83	3797	115.85 ± 6.75	2253	96.08 ± 5.84	979	51.77 ± 5.04	924	31.92 ± 3.72	337	24.49 ± 3.33	81	16.67 ± 3.20	63	51.82 ± 5.38	43	97.45 ± 6.44	21
RockSampling44 (800)	65.76 ± 5.42	4673	50.28 ± 4.94	2842	55.05 ± 5.35	1205	20.78 ± 2.61	926	20.01 ± 2.74	334	31.24 ± 4.07	82	22.95 ± 3.79	59	47.96 ± 4.54	45	100.72 ± 6.72	22

TABLE 6 – Résultats obtenus avec différentes formules de mise à jour dans ρ-POMCP avec un temps budget fixe de 500ms et plusieurs valeurs de $|\beta|$. Les colonnes *Vanilla* correspondent à la mise à jour de POMCP, les colonnes *Last-reward* aux mises à jour où la dernière récompense reçue remplace son estimation par moyenne mobile, et les colonnes *Last-value* aux mises à jour où les valeurs des nœuds visités sont recalculées. Chaque valeur correspond au résultat aggrégé de 200 expériences.

	Vanilla $\|\beta\| = 1$		Vanilla $\|\beta\| = 10$		Vanilla $\|\beta\| = 50$		Last-Reward $\|\beta\| = 1$		Last-Reward $\|\beta\| = 10$		Last-Reward $\|\beta\| = 50$		Last-Value $\|\beta\| = 1$		Last-Value $\|\beta\| = 10$		Last-Value $\|\beta\| = 50$	
Problem (UCB cst)	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d	$V \pm Err$	nb_d
Museum entropy (1)	-17.33 ± 0.41	5049	-16.88 ± 0.44	2834	-16.63 ± 0.46	1367	-16.16 ± 0.46	4280	-16.15 ± 0.37	2739	-16.87 ± 0.42	1356	-16.88 ± 0.40	5080	-16.26 ± 0.42	2771	-17.36 ± 0.44	1406
SeekAndSeek (69.3)	-30.98 ± 1.87	5013	-25.53 ± 1.88	2911	-29.80 ± 1.73	1096	-28.00 ± 1.91	5176	-33.26 ± 1.84	3024	-28.39 ± 1.80	1057	-30.59 ± 1.84	5263	-28.73 ± 1.77	2870	-26.45 ± 1.89	1074
CameraClean (14)	0.25 ± 0.03	9679	0.58 ± 0.03	4025	0.57 ± 0.02	1496	0.28 ± 0.02	8000	0.59 ± 0.02	3925	0.56 ± 0.02	1458	0.53 ± 0.02	8969	0.55 ± 0.02	4214	0.55 ± 0.02	1444
MazeCross (3.2)	2.21 ± 0.01	5753	2.20 ± 0.01	3419	2.17 ± 0.01	1438	2.16 ± 0.01	5528	2.21 ± 0.01	3472	2.21 ± 0.02	1488	2.20 ± 0.01	7922	2.22 ± 0.02	3805	2.22 ± 0.01	1608
MazeLines (4.3)	2.63 ± 0.04	5484	2.70 ± 0.04	3437	2.66 ± 0.04	1154	2.64 ± 0.04	4727	2.65 ± 0.04	3082	2.62 ± 0.04	1090	2.63 ± 0.04	5516	2.69 ± 0.04	2978	2.72 ± 0.04	1057
MazeHole (4.2)	2.11 ± 0.06	4362	2.02 ± 0.05	2892	2.07 ± 0.06	1107	2.01 ± 0.05	4155	2.09 ± 0.06	2803	2.09 ± 0.06	1112	2.12 ± 0.06	5104	2.00 ± 0.05	3370	1.94 ± 0.05	1138
MazeDots (3.6)	2.02 ± 0.05	4023	2.07 ± 0.05	2598	2.00 ± 0.05	1054	2.14 ± 0.05	4112	2.18 ± 0.05	2540	2.08 ± 0.05	1067	2.17 ± 0.06	5343	2.17 ± 0.06	2933	2.17 ± 0.06	1100
ActivlocCross (3.2)	2.10 ± 0.03	4774	2.12 ± 0.03	3364	2.15 ± 0.03	1635	2.09 ± 0.03	5056	2.06 ± 0.03	3449	2.10 ± 0.03	1586	2.09 ± 0.03	5715	2.04 ± 0.03	3964	2.09 ± 0.03	1570
ActivlocLines (4.3)	2.13 ± 0.06	3356	2.10 ± 0.05	2309	2.14 ± 0.05	1057	1.81 ± 0.05	3078	2.08 ± 0.05	2381	2.15 ± 0.06	1016	2.15 ± 0.05	3908	2.24 ± 0.05	2430	2.18 ± 0.05	1017
ActivlocHole (4.2)	1.63 ± 0.04	3856	1.67 ± 0.04	2656	1.66 ± 0.04	1120	1.58 ± 0.05	3582	1.58 ± 0.04	2681	1.77 ± 0.05	1103	1.71 ± 0.05	3827	1.62 ± 0.04	2524	1.66 ± 0.04	1086
ActivlocDots (3.6)	1.59 ± 0.08	4514	1.41 ± 0.07	2977	1.40 ± 0.07	1332	1.41 ± 0.07	4372	1.57 ± 0.08	2906	1.46 ± 0.08	1331	1.37 ± 0.08	5070	1.45 ± 0.07	3227	1.48 ± 0.07	1292
GridNotX (26)	-3.50 ± 0.22	8651	-3.24 ± 0.22	4293	-3.29 ± 0.21	1806	-3.68 ± 0.22	7961	-3.56 ± 0.17	4504	-3.27 ± 0.22	1811	-3.34 ± 0.23	8478	-3.10 ± 0.21	4173	-3.80 ± 0.21	1665
GridX (26)	21.67 ± 0.05	10270	21.75 ± 0.06	5089	21.77 ± 0.06	1750	21.71 ± 0.06	9303	21.69 ± 0.05	4722	21.64 ± 0.06	1831	21.63 ± 0.05	10273	21.66 ± 0.06	4150	21.70 ± 0.06	1737
RockSampling44 (100)	113.53 ± 6.83	3797	115.85 ± 6.75	2253	96.08 ± 5.84	979	114.13 ± 6.96	3811	114.61 ± 6.78	2108	91.66 ± 5.98	981	110.04 ± 6.90	4800	112.79 ± 6.37	2460	84.70 ± 6.23	907
RockSampling44 (800)	65.76 ± 5.42	4673	50.28 ± 4.94	2842	55.05 ± 5.35	1205	65.62 ± 5.85	4120	54.49 ± 5.54	2654	63.67 ± 5.69	1179	76.05 ± 5.49	5000	54.84 ± 4.84	2870	48.10 ± 4.93	1022