Monte Carlo Information-Oriented Planning

Vincent Thomas, Gérémy Hutin, Olivier Buffet

▶ **To cite this version:**

Vincent Thomas, Gérémy Hutin, Olivier Buffet. Monte Carlo Information-Oriented Planning. 24th ECAI 2020 - European Conference on Artificial Intelligence, Aug 2020, Santiago de Compostela, Spain. hal-02943028

HAL Id: hal-02943028
https://inria.hal.science/hal-02943028

Submitted on 18 Sep 2020

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

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

Monte Carlo Information-Oriented Planning

Vincent Thomas [1] and **Gérémy Hutin** [2] and **Olivier Buffet** [3]

Abstract. In this article, we discuss how to solve information-gathering problems expressed as ρ-POMDPs, an extension of Partially Observable Markov Decision Processes (POMDPs) whose reward ρ depends on the belief state. Point-based approaches used for solving POMDPs have been extended to solving ρ-POMDPs as belief MDPs when its reward ρ is convex in $\mathcal{B}$ or when it is Lipschitz-continuous. In the present paper, we build on the POMCP algorithm to propose a Monte Carlo Tree Search for ρ-POMDPs, aiming for an efficient on-line planner which can be used for any ρ function. 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. An asymptotic convergence proof to ϵ-optimal values is given when ρ is continuous. Experiments are conducted to analyze the algorithms at hand and show that they outperform myopic approaches.

1 INTRODUCTION

Many state-of-the-art algorithms for solving Partially Observable Markov Decision Processes (POMDPs) rely on turning the problem into a "fully observable" problem—namely a belief MDP—and exploiting the piece-wise linearity and convexity of the optimal value function in this new problem's state space (here the belief space $\mathcal{B}$) by maintaining generalizing function approximators. This approach has been extended to solving ρ-POMDPs as belief MDPs—*i.e.*, problems whose performance criterion depends on the belief (*e.g.*, active information gathering)—when the reward ρ itself is convex in $\mathcal{B}$ [2] or when ρ is Lipschitz-continuous [10].

In this paper, we propose two new algorithms for solving ρ-POMDPs which do not rely on properties of ρ such as its convexity or its Lipschitz-continuity, but are based on Monte Carlo sampling and inspired by POMCP [21]: (1) *ρ-beliefUCT* applies UCT to the belief MDP; and (2) *ρ-POMCP* uses particle filters, *i.e.*, samples particles during trajectories, to estimate the visited belief states and their associated rewards. We prove ρ-POMCP's asymptotic convergence and empirically assess these algorithms on various information-gathering problems.

The paper is organized as follows. Section 2 discusses related work on information-oriented control. Sec. 3 presents background (i) first on POMDPs, (ii) then on the Partially Observable Monte Carlo Planning (POMCP) algorithm, a Monte Carlo Tree Search approach for solving POMDPs, and (iii) finally on ρ-POMDPs. Sec. 4 describes our contribution: ρ-beliefUCT and ρ-POMCP, for solving ρ-POMDPs with MCTS techniques, and provides a proof of ρ-POMCP's convergence. Sec. 5 presents the conducted experiments and analyzes the results before concluding and giving some perspectives.

[1] Université de Lorraine, INRIA, CNRS, LORIA, France, vincent.thomas@loria.fr
[2] École Normale Supérieure de Lyon, France, geremy.hutin@ens-lyon.fr
[3] Université de Lorraine, INRIA, CNRS, LORIA, France, olivier.buffet@loria.fr

2 RELATED WORK

Early research on information-oriented control (IOC) involved problems formalized either (i) as POMDPs (as Egorov et al. [9] did recently, since an observation-dependent reward can be trivially recast as a state-dependent reward), or (ii) with belief-dependent rewards (and mostly ad-hoc solution techniques as [11, 17]).

Araya-López et al. [2] introduced ρ-POMDPs to easily formalize most IOC problems. They showed that a ρ-POMDP with convex belief-dependent reward ρ can be solved with modified point-based POMDP solvers exploiting the piece-wise linearity and convexity (PWLC property), with error bounds that depend on the quality of the PWLC-approximation of ρ. More recently, Fehr et al. [10] applied the same approach as Araya-López et al., but for Lipschitz-continuous (LC)—rather than convex—belief-dependent rewards, demonstrating that, for finite horizons, the optimal value function is also LC. Then, deriving uniformly improvable lower- (and upper-)bounds led to two algorithms based on HSVI [22].

Spaan et al.'s POMDP-IR framework allows describing IOC problems with linear rewards which are provably equivalent to "PWLC" ρ-POMDPs (*i.e.*, when ρ is PWLC) [19], and also leads to modified POMDP solvers. For its part, the general ρ-POMDP framework allows formalizing more problems—*e.g.*, directly specifying an entropy-based criterion.

We here consider ρ-POMDPs, but not relying on generalizing (PWLC or LC) value function approximators as in previous work, so that any belief-dependent reward can be used. For instance, this allows addressing problems where the objective is to minimize the quantity of information of an adversary with known behaviour or where the objective is to gather information on specific state variables while maintaining a low quantity of information on confidential ones (like in medical or domotic fields). None of these problems can be solved by previous approaches on ρ-POMDP nor be modelled by POMDP-IR, which requires a PWLC reward function in $\mathcal{B}$. To circumvent this difficulty, we build on Monte Carlo Tree Search (MCTS) approaches, in particular Silver and Veness's Partially Observable Monte Carlo Planning (POMCP) algorithm 2010. Moreover, MCTS and POMCP present several other benefits: (i) they require a simulator rather than a complete model; (ii) unlike heuristic search, they do not require optimistic or pessimistic initializations of the value function; and (iii) they have been proven to be very efficient for solving problems with huge state, action and observation spaces.

Similar works have been conducted for information gathering control with an MCTS-based approach [3, 15], but Lauri et al. [15] propose a POMCP algorithm for information-oriented *open-loop* planning (which corresponds to finding a sequence of actions to a ρ-POMDP while ignoring observations for action selection during planning), and Bargiacchi [3], whose approach is similar to our advanced approach (cf Sec. 4.3), present only results on small problems (on

specific ρ functions, namely negentropy and max-belief, whereas this article extends the approach for any ρ function). In both cases, there is no proof of asymptotic convergence and the continuity assumption of ρ is not discussed. Also, our experiments are conducted on a wider range of problems, and compare variants of the proposed algorithm with reference algorithms. We believe that these results demonstrate that there is interest in MCTS approaches for information-oriented control, and that our work provides a more in-depth (theoretical and empirical) look into them, making for a useful contribution.

3 BACKGROUND

3.1 POMDPs

A POMDP [18] is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, P, r, \gamma, b_0 \rangle$, where $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{Z}$ are finite sets of states, actions and observations; $P_{a,z}(s, s')$ gives the probability of transiting to state s' and observing observation z when applying action a in state s ($P_{a,z}$ is an $\mathcal{S} \times \mathcal{S}$ matrix); $r(s, a) \in \mathbb{R}$ is the reward associated to performing action a in state s; $\gamma \in [0, 1)$ is a discount factor; and b_0 is the initial belief state—*i.e.*, the initial probability distribution over possible states. The objective is then to find a policy π that prescribes actions depending on past actions and observations so as to maximize the expected discounted sum of rewards (here with an infinite temporal horizon).

To that end, a POMDP is often turned into a belief MDP $\langle \mathcal{B}, \mathcal{A}, P, r, \gamma, b_0 \rangle$, where $\mathcal{B}$ is the belief space, $\mathcal{A}$ is the same action set, and $P_a(b, b') = P(b'|b, a)$ and $r(b, a) = \sum_{s \in \mathcal{S}} b(s) r(s, a)$ are the induced transition and reward functions. This allows considering policies $\pi \colon \mathcal{B} \to \mathcal{A}$, and their value functions $V^\pi(b) \doteq E[\sum_{t=0}^{\infty} \gamma^t r(b_t, \pi(b_t)) | b_0 = b]$. Optimal policies maximize V^π in all belief states reachable from b_0. Their value function V^* is the fixed point of Bellman's *optimality* operator ($\mathcal{H}$) [4] $\mathcal{H}V : b \mapsto \max_a [r(b, a) + \gamma \sum_z \|P_{a,z} b\|_1 V(b^{a,z})]$, and acting greedily with respect to V^* provides such a policy.

3.2 MCTS for POMDPs

MCTS and UCT MCTS approaches [7, 14, 6] are online, sampling-based algorithms, here described in the MDP framework (while they also serve in settings like sequential games). In MCTS, the tree representing possible futures from a starting state is progressively built by sampling trajectories in a non-uniform way. Each iteration consists in 4 steps: **(i) selection**: a trajectory is sampled in the tree according to an exploration strategy until a node not belonging to the tree is reached; **(ii) expansion**: this new node is added to the tree; **(iii) simulation**: this new node's value is estimated by sampling a trajectory from this node according to a rollout policy; **(iv) backpropagation**: this estimate and the rewards received during the selection step are back-propagated to the visited nodes to update their statistics (value and number of visits). Upper Confidence Bound applied to trees (UCT) is an instance of MCTS where, when in a state node, the next action is selected using the Upper Confidence Bounds (UCB1) strategy, *i.e.*, picking an action so as to maximize its estimated value increased by an exploration bonus $c_{t,s} = C_p \sqrt{\frac{\log N(s)}{N(s,a)}}$, with $C_p > 0$ an exploration constant, $N(s)$ the number of past visits of node s, and $N(s, a)$ the number of selections of action a when in node s.

POMCP Silver and Veness [21] proposed the Partially Observable Monte Carlo Planning (POMCP) algorithm to apply MCTS for solving POMDPs. A POMDP is addressed through the corresponding belief MDP, a belief tree being made of alternating action and belief nodes as presented in Figure 1. The path to a belief node at depth t follows the action-observation history $h_t = (a_0, z_0, a_1, z_1, \ldots, z_t)$ leading from the root belief to this belief $b(h_t)$.

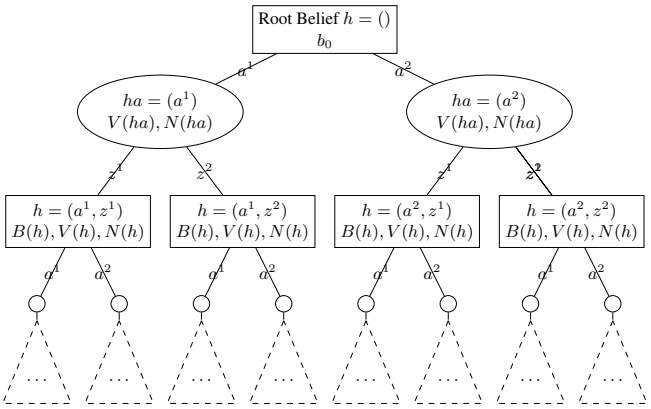

Figure 1: Example of a belief tree with 2 actions: a^1 and a^2, and 2 observations: z^1 and z^2, with various quantities maintained by POMCP.

Applying directly UCT on the belief-MDP would imply sampling trajectories $(b_0, a_0, b_1, a_1, \ldots, b_t)$ in belief space, thus requiring the complete POMDP model and a high computational cost to derive exact belief states. To prevent this cost, POMCP samples trajectories in state space, which only requires a simulator $\mathcal{G}$ as a generative model of the POMDP. During the selection phase, the first state is sampled from the root belief estimate, then one alternates between (i) picking an action according to UCB1 (applied to estimated action values in the current belief node), and (ii) sampling a next state s', observation z and reward r using $\mathcal{G}(s, a)$. By accumulating values in the belief nodes, averaging over all simulated trajectories gives an estimate of the value $V(h)$ of the belief node h.

Moreover, states are collected in each visited belief node in order to estimate the next root belief when an action is actually performed. By preventing the computation of exact beliefs $b(h)$ in each belief node, POMCP allows addressing large problems while preserving UCT's asymptotic convergence.

Silver and Veness also proposed the PO-UCT algorithm as a first step towards POMCP. It differs from POMCP in that it does not collect states, but computes the belief state of the new root with an exact Bayes update.

3.3 ρ-POMDPs

ρ-POMDPs [2] differ from POMDPs in that their reward function ρ is belief-dependent, thus allowing to define not only control-oriented criteria, but also information-oriented ones, thus generalizing POMDPs. The immediate reward ρ is naturally defined as $\rho(b, a, b')$, the immediate reward associated to transiting from belief b to belief b' after having performed action a[4].

As presented in related work, Araya-López et al. [2] and Fehr et al. [10] have exploited PWLC and Lipschitz-continuous reward functions ρ to solve general ρ-POMDPs. However, while many problems can be modeled with convex or Lipschitz-continuous ρ, this leaves us with a number of problems that cannot be solved with similar approximations or with the POMDP-IR approach (for instance, when we seek to

[4] Without loss of generality, the article uses $\rho(b, a)$ (even if the addressed problems rely on $\rho(b, a, b')$).

minimize information or to find a compromise between gathering information and preserving privacy as presented in Sec. 2).

Here, we propose to use the MCTS approach to address the general ρ-POMDP case with no constraints on the ρ function. As an example, let us consider an agent monitoring a museum and whose aim is to locate a visitor with a specified certainty. If X denotes the random variable for the visitor's location and b_X the corresponding belief, then the reward function $\rho_X(b,a) \doteq \mathbf{1}_{\|b_X\|_\infty > \alpha}$[5] rewards beliefs whose maximum probability is greater than $\alpha \in [0,1]$. This is a threshold function, thus neither convex nor Lipschitz-continuous, due to its discontinuity when $\|b_X\|_\infty = \alpha$.

4 MCTS ALGORITHMS FOR ρ-POMDPs

POMCP and its variant PO-UCT cannot be applied directly to solve ρ-POMDPs. PO-UCT does not compute exact beliefs during its selection and backpropagation steps (only at the tree root), and thus cannot compute belief-dependent rewards along trajectories. POMCP generates trajectories following a single sequence of states and samples associated rewards. It collects visited states in belief nodes, which are not sufficient to correctly estimate the belief and thus belief-dependent rewards.

4.1 ρ-beliefUCT

The first proposed algorithm, *ρ-beliefUCT*, consists in directly applying UCT to the belief MDP built from the ρ-POMDP. This requires accessing the complete ρ-POMDP model and computing exact beliefs for each visited belief node by performing Bayes updates: $b_{haz}(s') \propto \sum_{s \in S} P_{a,z}(s,s').b_h(s)$.

During the simulation step, a belief-based rollout policy (in contrast with a random one) needs to compute the belief at each action step not only to make action choices, but also to compute immediate rewards $\rho(b,a)$ and estimate the value of the added node. This induces an important computational cost. However, ρ-beliefUCT can take advantage of several elements. First, since each history corresponds to a unique belief state, each belief state needs to be computed only once, whenever a new belief node is added. Secondly, while, in a POMCP approach, $r(b,a)$ is estimated using averages of r(s,a) without bias because $r(b,a)$ is linear in b, in our context, $\rho(b,a)$ is a deterministic function of the belief: its exact value can be directly stored in the action node. Lastly, during the selection step, an observation can be sampled without computing observation probability $P(z|b(h))$ and without bias by sampling a state s from current belief $b(h)$, then a state-observation pair (s',z) from $\mathcal{G}$.

ρ-beliefUCT is an interesting reference algorithm. As a direct adaptation of UCT, it inherits its convergence properties only assuming that ρ is bounded [12]. However, to avoid (i) the cost associated to exact belief computations, and (ii) the need for the complete model of the POMDP, we also propose another algorithm, ρ-POMCP, which does not compute exact beliefs but estimates them.

4.2 ρ-POMCP

As a first approximation, the ρ-POMCP algorithm is similar to POMCP. During the selection step, trajectories are generated by sampling states and observations using generative model $\mathcal{G}$. When visited, each belief node h collects the state that has led to this node in order to build an estimate of the true belief state $b(h)$.

[5] with $\mathbf{1}$ denoting the indicator function.

However, applying directly POMCP by only adding the current state of the trajectory to the belief node (as proposed in [3]) may lead to poorly estimated immediate rewards during the first steps of the algorithm, thus causing the MCTS algorithm (i) to unefficiently spend time focusing on branches with over-estimated rewards and (ii) to slow down the exploration of branches with under-estimated rewards.

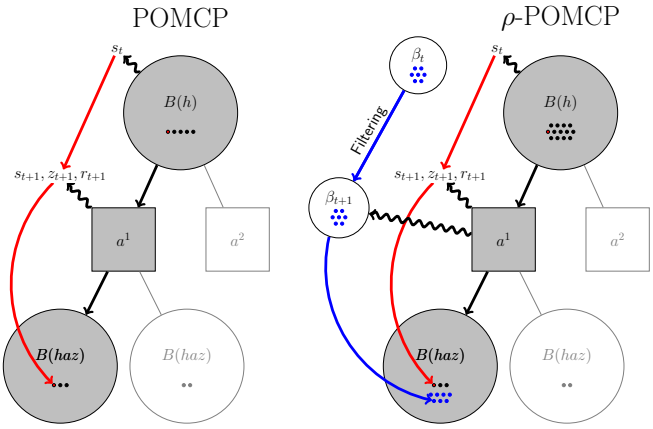

Figure 2: Difference between a POMCP and a ρ-POMCP descent: during a ρ-POMCP descent, at each simulation step, a bag of particles β_{t+1} is generated from the bag β_t, the selected action and the received observation. The particles of this bag β_{t+1} are added to the node $B(h_{t+1})$ to produce a better estimation of $b(h_{t+1})$ than with only the sampled trajectory state.

To add more states at each visit of a belief node, we propose using a particle filter, as illustrated in Figure 2. Thus, at each transition, when action a_t is selected from trajectory state s_t and observation z_{t+1} is sampled, the algorithm generates a set β_{t+1} of $|\beta|$ states, called a *small bag of particles*, using $\mathcal{G}$ to make $|\beta|$ samplings consistent with observation z_{t+1}. Then, it adds this small bag β_{t+1} to $B(haz)$,[6] the *cumulative bag* for history haz, stored in the corresponding belief node: $B(haz) \leftarrow B(haz) \cup \beta_{t+1}$.

Rejection sampling With a generative model $\mathcal{G}$, only rejection approaches (also referred as logic sampling [13]) can be considered to produce consistent samples. In this case, a pair $(\tilde{s}', \tilde{z})$ is sampled from $\mathcal{G}(s,a)$, and $\tilde{s}'$ is kept in β if and only if $\tilde{z}$ corresponds to the observation z of the sampled trajectory. This process repeats until enough consistent samples have been obtained, which can be computationally expensive when $P(z|h,a)$ is small.

Importance sampling To avoid the computational cost of rejection sampling, we instead use importance sampling [8], which assumes that the observation function is available and leads to weighting each particle. After performing action a_t and receiving observation z_{t+1}, a new small bag β_{t+1} is generated from β_t by using the generative model and following these steps: (1) sample a state $\tilde{s}$ from β_t; (2) sample a state $\tilde{s}'$ by using the generative model, $\tilde{s}' \sim \mathcal{G}(\tilde{s}, a_t)$; (3) store this particle $\tilde{s}'$ in β_{t+1} with an associated weight of $P(z_{t+1}|\tilde{s}, a_t, \tilde{s}')$ corresponding to the probability of having generated observation z_{t+1}. The weights in two small bags associated to the same node can be compared, and one can thus accumulate such bags in the corresponding belief node. To save memory, when a small bag β is added to a *cumulative bag* $B(h)$, particles corresponding to the same state are

[6] with $a = a_t$ and $z = z_{t+1}$

merged, their weights being added up. Note that, to prevent particle depletion, the state s_{t+1} in trajectory $h_t = (s_0, a_0, \ldots, a_t, s_{t+1})$ is also included in small bag β_{t+1} with its corresponding weight, so that the observation generated during the sampled trajectory can always be explained by at least one particle in β_{t+1}. While this induces a bias in this trajectory's small bags, since trajectory states are obtained by sampling from the root belief, the distribution of the states in the *cumulative bags* $B(h)$ is unbiased.

In the simulation step, belief estimates are required at least to estimate instant rewards, and possibly to make decisions. The sequence of small bags β thus needs to be perpetuated.

Finally, during the back-propagation step, the *cumulative bag* $B(h)$ of belief node h is used to estimate true belief state $b(h)$—by normalizing the weights—and compute $\rho(B(h))$. Weights are stored un-normalized in $B(h)$ so that new weighted particles can be added without introducing a bias.

The above algorithm description leads to Algorithm 1. In this algorithm, the notation T corresponds to the belief tree and I to the initial belief. $s, \beta \overset{1+n}{\sim} I$ indicates that $1+n$ states are sampled from I, 1 being stored in s and the n others in β. The function *PF* corresponds here to the importance sampling particle filtering process previously described and $w_{s'} = P(z|s, a, s')$.

Algorithm 1: ρ-POMCP Text in red highlights differences with POMCP.

Fct SEARCH (h)
  repeat
    if $h = empty$ **then**
      $s, \beta \overset{1+n}{\sim} I$
    else
      $s, \beta \overset{1+n}{\sim} B(h)$
    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(β, 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(β, 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

Note that, as in POMCP or in standard particle filters, when the system actually evolves, the actual observation may not be explained by any state contained in the new root's cumulative bag. We leave this case for discussion, but it is still possible to re-estimate the current belief by simulating the process from the initial belief.

Asymptotic convergence The following theorem states the asymptotic convergence of ρ-POMCP, a notable difference with ρ-beliefUCT being the need for ρ to be continuous in b and bounded by ρ_{max}.

Theorem 1 *Let ρ be continuous in b and bounded by ρ_{max}, and $\epsilon > 0$, then the root action values computed by ρ-POMCP converge asymptotically to ϵ-optimal action values.*

Proof 1 *Since ρ is bounded and the criterion is γ-discounted, the ϵ-convergence allows reasoning with a finite horizon only, even though the problem horizon is infinite (with an infinite belief space). To do so, let us consider the tree of depth $\delta = \lceil \frac{\ln \frac{\epsilon(1-\gamma)}{\rho_{max}}}{\ln \gamma} \rceil$, and assume that the root belief estimate is exact. Due to UCB1, all nodes in that tree are visited infinitely often. Each belief-node h then collects infinitely many particles ($|\beta|$ at each visit) and, due to the use of particle filters applied from the root node, the probability distribution induced by cumulative bags $B(h)$ converges to the true belief state $b(h)$. Let us prove by induction that, at any depth $d \leq \delta$, the action value estimates are bounded by $\gamma^{-d}\epsilon$. Trivially, any node's value estimate is absolutely bounded by $\frac{\rho_{max}}{1-\gamma}$, so that, in particular, the bound is $\gamma^{-\delta}\epsilon$ at depth δ (cf. def of δ). Let us now assume that the induction property holds at depth $d \in \{1, \ldots, \delta\}$. Then for any node h at depth $d-1$, (i) due to ρ's continuity, $\rho(b(h))$ is correctly estimated, [7] (ii) since each action is infinitely selected, by induction, the action values converge to $\gamma^{d-1}\epsilon$-optimal values, and (iii) UCB1 selects the optimal action infinitely more often than other actions, thus $V(h)$ converges to a $\gamma^{d-1}\epsilon$-optimal value.* □

4.3 ρ-POMCP variants

Proposed variants The current difference between POMCP and ρ-POMCP lies in the way particles are collected in order to estimate the reward obtained at each transition. But value estimates are updated in the same way. However, these updates can be improved, leading to several ρ-POMCP variants, since, during its execution, ρ-POMCP builds increasingly better estimates of the true belief states in visited belief nodes. To do so, we first discuss what is computed by updates performed by POMCP (and vanilla ρ-POMCP), and then propose several variants (also proposed by [3]).

Computations performed by POMCP If we ignore the node initialization in POMCP, then, when considering a node-action pair ha, the value stored in $V(ha)$ averages, over $N(ha)$ samples/visits:

- $\sum_{s \in \beta_{ha}} r(s, a)$: the total reward over the states s that were sampled while action a was picked in h (β_{ha} denotes this set of states);
- $\sum_{z \in \mathcal{Z}_{ha}} \mathbf{1}_{N(h,a,z) \geq 1} Rollout(haz)$: the total return of the rollouts generated from each observation z sampled after picking a in h (set $\mathcal{Z}_{ha}$), where $N(h, a, z)$ is the number of times action a was followed by observation z in node h (while $N(haz)$ is the number of updates of node haz);
- $\sum_{z \in \mathcal{Z}_{ha}} \sum_{a' \in \mathcal{A}} N(haza')V(haza')$: the sum of the values of the children nodes, weighted by their visit counts (which also includes the rollouts performed from these nodes).

By introducing belief node value estimates $V(h)$, initialized with rollout values (for $N(h) = 1$), this leads to the following formulas:

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

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

[7] It is only required that the bias of the $\rho(B(h))$ estimator vanishes when the number of particles grows, which is guaranteed by ρ's continuity.

Last-reward-update ρ-POMCP Thus, in POMCP, $V(ha)$ is a moving average that "contains" an estimate of $r(b(h), a)$, that estimate being computed as $\frac{\sum_{s\in\beta_{ha}} r(s,a)}{N(ha)}$.

In vanilla ρ-POMCP, the sampled r at a current time step is replaced with an estimate of $\rho(b(h))$ as $\rho(\hat{B}_{N(h)}(h))$, where $\hat{B}_{N(h)}(h)$ is the cumulative bag after the first $N(h)$ visits. In this case, $V(ha)$ includes thus (among other elements) an average of successive estimates (*i.e.*, it computes $\sum_{i=1}^{N(ha)} \rho(\hat{B}_{\phi(i)}(h), a)$, where $\phi(i)$ is the i^{th} visit of h where a was selected). However, it would seem more appropriate to instead use the reward associated to the last estimated belief $\rho(\hat{B}_{\phi(N(ha))}(h), a)$, which is a less biased estimate.

This proposed variant, called *last-reward-update ρ-POMCP* (or *"lru-ρ-POMCP"*), fixes this rather easily by replacing the update of $V(ha)$ in Algorithm 1 by

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

where $\rho^{prev}(h,a)$ is the previous value of the reward when ha was last experienced (thus needs to be stored).

Last-value-update ρ-POMCP But then, $V(ha)$ also "contains" estimates of average rewards for future time steps, which suffer from the same issue. To fix this, SIMULATE should not return a sample return, but an estimate of the average return.

The updates in the backpropagation step consist then in re-estimating all the values of the visited belief nodes by using the reward obtained at each transition and the initial rollout, which needs to have been previously stored. This is done in *last-value-update ρ-POMCP* variant (or *"lvu-ρ-POMCP"*) through the following formulas:

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

5 EXPERIMENTS

5.1 Benchmark Problems

POMDPs being a subclass of ρ-POMDPs, first benchmark problems we consider are the classical *Tiger*, *Tiger-Grid* and *Hallway2* problems [16] (as per Cassandra's POMDP page) and instances of *Rock Sampling* [23] with several grids of $n \times n$ cells, and n rocks to sample (where n equals 4, 6 and 8). A reward of $+100$ (resp. -100) is given for sampling a good (resp. bad) rock.

Then, a known issue with information-gathering problems is that a simple myopic strategy may often give very good results [5, 20]. In order to assess the proposed algorithms, we had to provide problems where myopic strategies encounter difficulties.

We first proposed the *Museum problem*, inspired by [19], where an agent has to continuously localize a visitor in a toric grid environment (4×4 in our experiments). The state corresponds to the visitor's unknown location. At each time step, the visitor stays still with probability 0.6, and moves to 1 of his 4 neighboring cells with probability 0.4 (chosen uniformly). The agent acts by activating a camera in any location. It then receives a deterministic observation: *"present"* if the visitor is at this location, *"close"* if s/he is in a neighboring cell, and *"absent"* if s/he is further away. Additionally: the immediate reward corresponds to the negentropy of the belief: $\rho(b,a,b') = -H(b') = \sum_s b'(s).log(b'(s))$; the initial belief b_0 is uniform over all cells; and $\gamma = 0.95$. The interest of this problem lies in the large number of actions (one per cell). We also used a variant, *Museum Threshold*, as proposed in Sec. 3.3, where the (non-continuous) reward is based on a threshold function on the belief state (with $\alpha = 0.8$) which is null in most of the belief space.

In *active-localization* problems, an agent is in a toric grid with white and black cells. At each step, it can move to a neighboring cell or observe the color of its current cell. Active localization is difficult for myopic strategies as they see no (immediate) benefit in moving. Additionally: observations and transitions are deterministic; b_0 is uniform over all cells; the reward corresponds to the entropy difference between b and b': $\rho(b,a,b') = -H(b') + H(b)$; and $\gamma = 0.95$. Several configurations have been studied (*cf.* Fig. 3): (i) *MazeCross*, where black cells make it easy to localize oneself; (ii) *MazeLines*, where the agent cannot localize itself within the striped region and has to plan several steps ahead to look for the spot in the empty region; (iii) *MazeHole*, which requires the agent to reason one step in advance to search for the missing black cell in this regular configuration; and (iv) *MazeDots*, identical to MazeHole except that black cells are separated by several white cells. *MazeDots_nxn* corresponds to variations of the *MazeDots* configuration where n denotes the size of the grid.

GridX and *GridNotX* [10] differ from the first localization problems in that: (i) moves (n,s,e,w) succeed with probability 0.8, otherwise the agent stays still; (ii) denoting b_x (resp. b_y) the belief over the x (resp. y) coordinate, the reward function is $\rho(b) = +\|b_x - \frac{1}{3}\mathbf{1}\|_1$ (resp. $-\|b_x - \frac{1}{3}\mathbf{1}\|_1$) for *GridX* (resp. *GridNotX*). *GridNotX* is an interesting problem since the objective consists in maximizing uncertainty, which can not be modeled with state-depend rewards and POMDP-IR.

In the *SeekAndSeek* problem, an object is lost in a (toric grid) maze with obstacles limiting the possible moves of the agent. The agent can move in the maze and receives specific observations whenever the object is on its location (*present*), next to it (*close*), or otherwise (*absent*). The immediate reward received is the entropy difference over the possible object locations $\rho(b,a,b') = -H(b') + H(b)$. The object being at a fixed location, a myopic agent surrounded by already visited cells has no incentive to explore any of them since they do not give any information.

We also addressed Araya-López's *CameraClean* Diagnosis problem [1]. Here, a robot camera can be oriented to shoot one of four different zones and must find an object put in a fixed location. Its state is its current target zone, and a boolean specifying whether its lens is clean or dirty. As actions, the robot camera can reorient itself (deterministically) to the next zone, shoot the current zone, or clean its lens. When the camera takes a picture, the probability to properly detect the presence or absence of the object in the target zone is 0.8 if the lens is clean, and 0.55 if dirty. The reward is the entropy difference between starting and arrival belief states $\rho(b,a,b') = -H(b') + H(b)$.

Finally, we proposed the *LostOrFound* problem to consider non-convex rewards. An agent is in a toric corridor with one colored cell. It can move left or right (with probability 0.3 to stay), stay or change its status (*lost* or *found*). Then, it observes the color of its cell. The reward depends on the agent's status. If its status is *found*, the agent tries to localize itself and $\rho(b,a,b') = H_{max} - H(b')$, where $H_{max} = log(|cells|)$ denotes the maximum entropy to ensure positive rewards. If its status is *lost*, it tries to get lost and receives $\rho(b,a,b') = 3.H(b')$. To act optimally, the agent needs to change its status to *lost* and to perform moving actions to lose information regarding its location.

All experiments were conducted on Intel Xeon Gold 6130 cores @

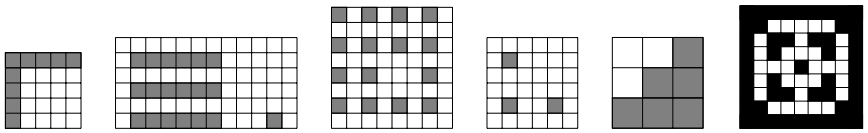

Figure 3: The various cell configurations used for active localization problems; from left to right: (a) *MazeCross*, (b) *MazeLines*, (c) *MazeHole* and (d) *MazeDots*; (e) corresponds to the cell configuration used for *GridX* and *GridNotX*, and (f) the obstacle configuration for *SeekAndSeek*.

2.1 GHz with 512 MB. Source code is available at `https://gitlab.inria.fr/vthomas/ecai_2020_source`.

5.2 Results

Influence of rollouts Preliminary experiments have been conducted with random rollouts. For the same number of descents, ρ-POMCP with random rollouts (stopped when $\gamma^{depth} < 0.01$) lasts between 5 and 10 times longer than without any rollout (setting the value of new nodes to 0). Moreover, the gain from using a pure random rollout policy was also rarely significant. In some problems (*Tiger*), it even reduced the observed performance. We thus focus on ρ-POMCP and ρ-beliefUCT without rollouts.

Comparison with myopic strategies Table 1 presents results comparing the purely *Random* strategy, *Look-ahead* strategies, ρ-beliefUCT, and ρ-POMCP on the benchmark problems. The *look-ahead-H* algorithms perform dynamic programming over all possible futures for a fixed finite horizon H by using the complete ρ-POMDP model. The pure myopic strategy, where the agent maximizes its immediate reward, corresponds to *Look-ahead-1*, whereas *Look-ahead-3* corresponds to anticipating all consequences 3 time-steps in advance[8]. Both ρ-MCTS algorithms (ρ-POMCP and ρ-beliefUCT) use a fixed number of descents $nb_{descents} = 10\,000$, without any rollout (setting the value of new nodes to 0) and use a specific constant UCB for each problem as specified in Tab. 1 (usually $(R_{max} - R_{min})/(1-\gamma)$). For ρ-POMCP, $|\beta| = 50$ and importance sampling was used.

(a) Tab. 1 shows that *Look-ahead-1* is close to the best value only on *Tiger*, *GridX* and *Museum entropy*. Regarding *museum* problems, the myopic strategy is less efficient than *look-ahead-3* in *Museum Threshold* due to the sparsity of non-zero rewards.

(b) It is known that *Look-ahead-1* often gives good results and usually constitute a very good baseline [5, 20]. But, we have also compared our approaches to the more challenging *Look-ahead-3* strategy. In this case, ρ-POMCP and ρ-beliefUCT give better results than *Look-ahead-1* and similar results to *Look-ahead-3*. In most cases, the difference between *Look-ahead-1* and *ρ-POMCP* is significant. *ρ-POMCP* also improves on *Look-ahead-3* in *SeekAndSeek*, *RockSampling44*, *GridNotX ActiveLocLines*, *Hallway2* and *TigerGrid*.

(c) ρ-POMCP and ρ-beliefUCT provide the same results for this number of descents and take usually a lot more time than the *Look-ahead* algorithms. ρ-beliefUCT is usually faster than ρ-POMCP, but this depends on the problem since computational costs of these two algorithms come from different operations. In ρ-beliefUCT, this cost is due to the computation of exact beliefs when a new belief node is added. In ρ-POMCP, it is due to the generation of small bags β at each transition. That is why, whereas the time needed by ρ-POMCP is more regular (except for the yet-unexplained case of *GridX*), the time needed by ρ-beliefUCT largely depends on $|\mathcal{S}|$. For instance, belief computation is quick in *Tiger* or *CameraClean*, but requires more time in *Active Localization* problems, where belief states include many states with a non-zero probability. Sec. 6 proposes tentative solutions to ρ-POMCP's high computational cost.

(d) In problems with larger state or observation spaces, like *Hallway2* and *TigerGrid*, ρ-MCTS algorithms are faster than *Look-ahead-3* (by a factor between 8 and 20 depending on the problem) while achieving a higher performance. It shows that ρ-MCTS algorithms manage to deal with problems even with a high branching factor (where *Look-ahead-3* cannot compete) by focusing their descents on interesting branches.

(e) Finally, results from *Rocksampling* are difficult to analyze since they highly depend on the rock locations (different for $v1$, $v2$ and $v3$). The random action selection in *Look-head* when no reward is visible gives a good exploration policy for reaching rocks, whereas ρ-MCTS require more time to reach the interesting rewards (cf Tab. 2).

Results with fixed time-budget Table 2 presents results regarding the influence of $|\beta|$ with a fixed time-budget of 1 s by action (except last column). When $|\beta|$ increases, the number of descents performed by ρ-POMCP naturally decreases (since trajectory generation is slower), but this has no clear impact on the γ-discounted cumulated value. Several problems exhibit different behaviors. In *CameraClean* and *Tiger*, there is a significant performance gap between $|\beta| = 0$ and $|\beta| = 5$. This might come from the highly stochastic observation process which requires better belief state estimates to act correctly. In *Rocksampling*, small values of $|\beta|$ give better results. This may be due to overestimation which favors exploitation, and more descents allowing to reach a greater depth. Finally, results obtained with *LostOrFound* need to be commented in detail. With a UCB constant of 35, ρ-POMCP needs an large bag to give good results. In this problem, convex rewards (*found* status) are compared with concave rewards (*lost* status). However, when we try to maximize information (convex reward), a poorly estimated belief lead to an optimistic return and ρ-POMCP is attracted by the corresponding branch. On the contrary, with concave rewards, the estimated return is low and will be an incentive not to explore this branch anymore. Both these effects lead ρ-POMCP to fail to find a good policy when $|\beta|$ is low. When the UCB constant increases, this effect disappears due to favored exploration. In (almost) all cases, when the time-budget increases (like 10 s in the last column), ρ-POMCP manages to generate the highest cumulated return (higher than *Look-ahead-3* from Tab. 1).

Comparison between rejection and importance sampling (IS) We have conducted experiments with ρ-POMCP algorithms to test the interest of using importance sampling instead of rejection sampling. All experiments were conducted with 200 runs of 40 actions, 10 000 descents per action, $|\beta| = 20$, and UCB constants from previous tables. The cumulated values are the same and rejection sampling requires approximatively 20% more time on small problems (the table is not presented in this article). However, for problems with larger

[8] To prevent side-effects, when several actions share the same highest value, the performed action is randomly selected among them.

Table 1: Results obtained when executing 200 episodes of 40 actions. V is the averaged γ-discounted performance; Err is the standard error; and $t(\mathrm{s})$ is the average duration of one episode. ρ-MCTS-algorithms were launched with a fixed number of 10 000 descents before performing each action. Significantly best values are highlighted in bold.

	Random		Look-ahead-1		Look-ahead-3		ρbeliefUCT		ρPOMCP $\|\beta\| = 50$	
Problem (UCB cst)	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$	$V \pm Err$	$t(s)$
Tiger (360)	-122.96 ± 2.59	0.00	**1.80** ± 0.13	0.02	**1.77** ± 0.12	0.08	**1.92** ± 0.13	6.44	**2.04** ± 0.11	63.63
CameraClean (14)	0.39 ± 0.01	0.00	0.19 ± 0.01	0.02	**0.82** ± 0.02	0.07	**0.81** ± 0.02	8.46	**0.81** ± 0.02	64.59
SeekAndSeek (69.3)	-42.04 ± 1.43	0.01	-37.70 ± 1.46	0.06	-30.75 ± 1.36	0.43	**-23.25** ± 1.16	32.73	**-25.68** ± 1.20	72.61
Hallway2 (1)	0.02 ± 0.00	0.01	0.03 ± 0.00	0.62	0.15 ± 0.01	723.19	**0.21** ± 0.01	46.17	**0.19** ± 0.01	36.40
TigerGrid (1)	-1.73 ± 0.15	0.01	-0.08 ± 0.04	0.22	0.52 ± 0.06	217.54	**1.93** ± 0.05	20.48	**1.81** ± 0.06	25.29
Museum entropy (1)	-26.31 ± 0.23	0.02	**-16.85** ± 0.30	0.16	-16.95 ± 0.27	70.02	**-16.09** ± 0.30	24.11	**-16.77** ± 0.28	51.83
Museum threshold (1)	1.71 ± 0.07	0.02	6.30 ± 0.16	0.18	**6.78** ± 0.17	71.53	**6.58** ± 0.17	28.63	**6.60** ± 0.17	83.29
ActivLocCross (3.2)	1.38 ± 0.04	0.01	2.33 ± 0.02	0.04	**2.58** ± 0.01	0.32	**2.57** ± 0.01	25.70	**2.57** ± 0.01	39.99
ActivLocLines (4.3)	1.11 ± 0.04	0.01	1.99 ± 0.04	0.08	2.78 ± 0.03	0.87	**2.99** ± 0.03	72.46	**2.96** ± 0.03	50.59
ActivLocHole (4.2)	0.92 ± 0.03	0.01	1.59 ± 0.04	0.07	**2.36** ± 0.04	0.79	**2.28** ± 0.04	64.34	**2.29** ± 0.04	60.18
ActivLocDots (3.6)	0.76 ± 0.05	0.01	1.56 ± 0.05	0.05	**2.09** ± 0.04	0.48	**2.15** ± 0.04	41.61	**2.15** ± 0.04	54.69
ActivLocDots 8x8 (4.2)	0.48 ± 0.03	0.01	1.15 ± 0.04	0.06	1.65 ± 0.03	0.81	**1.77** ± 0.04	67.12	**1.74** ± 0.04	83.33
ActivLocDots 10x10 (4.6)	0.33 ± 0.03	0.02	0.73 ± 0.04	0.10	1.16 ± 0.04	1.28	**1.34** ± 0.05	93.83	**1.25** ± 0.04	121.00
ActivLocDots 12x12 (5)	0.27 ± 0.03	0.02	0.60 ± 0.04	0.12	**0.98** ± 0.04	1.69	**0.98** ± 0.05	130.12	**0.96** ± 0.04	160.93
GridX (26)	16.69 ± 0.14	0.01	**21.62** ± 0.04	0.03	**21.72** ± 0.04	0.34	**21.68** ± 0.04	82.12	**21.73** ± 0.04	448.01
GridNotX (26)	-16.91 ± 0.15	0.01	-4.57 ± 0.20	0.04	-3.95 ± 0.15	0.37	**-3.22** ± 0.14	13.89	**-3.04** ± 0.14	70.26
RockSampling44 (100)	-20.95 ± 4.15	0.01	18.10 ± 1.82	0.05	99.12 ± 4.49	2.09	**115.67** ± 4.92	12.48	**109.15** ± 4.76	61.34
RockSampling66 (100)	-5.66 ± 1.70	0.02	5.26 ± 0.73	0.17	**96.04** ± 4.32	10.08	82.59 ± 4.14	39.46	78.21 ± 3.74	107.09
RockSampling88 v1(100)	-3.83 ± 1.72	0.09	6.33 ± 1.06	1.11	158.49 ± 4.87	46.01	**206.05** ± 5.46	107.79	177.31 ± 5.95	131.09
RockSampling88 v2 (100)	-4.28 ± 1.20	0.05	6.53 ± 0.83	0.65	**140.38** ± 4.88	49.04	114.66 ± 6.12	169.70	104.53 ± 5.99	178.04
RockSampling88 v3 (100)	-2.73 ± 1.02	0.05	4.79 ± 0.98	0.63	**117.47** ± 4.93	53.26	86.21 ± 3.83	170.42	91.60 ± 4.24	177.57
LostOrFound (35)	24.40 ± 0.69	0.01	31.23 ± 0.00	0.02	61.75 ± 0.28	0.14	**62.97** ± 0.21	53.84	61.82 ± 0.38	495.08
LostOrFound (100)	25.18 ± 0.66	0.01	31.23 ± 0.00	0.03	**61.85** ± 0.27	0.14	31.38 ± 0.15	7.08	31.23 ± 0.00	49.01

Table 2: Influence of $|\beta|$ for a fixed time budget with a budget of 1s per action (except last column with a budget of 10s for reference). Each column corresponds to the results obtained for 200 runs of 40 actions by ρ-POMCP and several values of $|\beta|$. V is the average γ-discounted cumulative value, Err the standard Error and nb_d the number of descents performed. Significantly best values are highlighted in bold.

	$\|\beta\| = 0$		$\|\beta\| = 1$		$\|\beta\| = 5$		$\|\beta\| = 10$		$\|\beta\| = 100$		$\|\beta\| = 10, t = 10s$	
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
Tiger (360)	-13.43 ± 0.00	30945	-13.43 ± 0.00	29763	**2.14** ± 0.10	19639	**2.12** ± 0.12	13900	0.38 ± 0.12	2766	**1.95** ± 0.13	102553
CameraClean (14)	0.28 ± 0.02	31976	0.25 ± 0.02	28418	**0.77** ± 0.02	19092	**0.78** ± 0.02	13942	**0.79** ± 0.02	3289	**0.80** ± 0.02	92671
SeekAndSeek (69.3)	**-28.39** ± 1.25	17342	**-27.46** ± 1.21	14574	**-28.04** ± 1.23	10877	**-27.82** ± 1.19	8505	**-27.07** ± 1.18	1740	**-26.37** ± 1.13	65322
Hallway2 (1)	0.19 ± 0.01	30226	0.18 ± 0.01	23153	0.19 ± 0.01	17522	0.21 ± 0.01	14052	0.18 ± 0.01	2561	**0.25** ± 0.01	106838
TigerGrid (1)	**1.73** ± 0.06	58236	**1.81** ± 0.06	48902	**1.84** ± 0.06	38756	**1.79** ± 0.06	31980	**1.82** ± 0.06	9641	**1.80** ± 0.06	184639
Museum entropy (1)	**-16.51** ± 0.29	29418	**-16.81** ± 0.27	24674	**-16.74** ± 0.29	19007	**-16.20** ± 0.29	15034	-17.01 ± 0.29	3052	**-16.64** ± 0.28	65596
Museum threshold (1)	6.15 ± 0.17	31232	6.15 ± 0.17	26468	**6.23** ± 0.18	19334	**6.59** ± 0.18	14958	**6.70** ± 0.16	3196	**6.41** ± 0.18	81593
ActivLocCross (3.2)	**2.58** ± 0.01	20602	**2.59** ± 0.01	18893	**2.56** ± 0.01	15391	**2.57** ± 0.01	12475	**2.57** ± 0.01	2837	**2.58** ± 0.01	99308
ActivLocLines (4.3)	2.89 ± 0.03	11859	2.89 ± 0.03	11188	2.90 ± 0.03	9208	**2.90** ± 0.03	7697	2.87 ± 0.03	1693	**2.98** ± 0.03	58659
ActivLocHole (4.2)	**2.36** ± 0.04	12737	**2.39** ± 0.04	12025	2.26 ± 0.04	9556	2.27 ± 0.04	8152	**2.33** ± 0.04	1709	**2.34** ± 0.04	57199
ActivLocDots (3.6)	**2.15** ± 0.04	15897	**2.18** ± 0.04	14965	**2.16** ± 0.04	12031	**2.21** ± 0.04	9288	**2.18** ± 0.04	2035	**2.14** ± 0.04	70543
GridX (26)	**21.69** ± 0.04	44487	21.67 ± 0.04	35916	**21.79** ± 0.04	23823	21.67 ± 0.04	17583	21.68 ± 0.04	3718	21.61 ± 0.04	101912
GridNotX (26)	-3.46 ± 0.15	40786	**-3.30** ± 0.14	33989	-3.54 ± 0.14	23351	**-3.17** ± 0.13	16605	**-3.41** ± 0.14	3627	**-3.01** ± 0.15	123545
RockSampling44 (100)	**114.00** ± 4.68	21967	**110.78** ± 5.08	18991	**106.79** ± 4.69	13059	**109.00** ± 4.87	9952	103.89 ± 4.33	2198	**118.46** ± 4.59	66347
RockSampling66 (100)	**110.12** ± 4.37	14033	**108.86** ± 4.38	11981	**103.77** ± 4.37	8915	88.08 ± 4.13	6416	50.00 ± 3.98	1112	**109.88** ± 4.02	55186
RockSampling88 v1(100)	172.69 ± 6.10	6228	155.01 ± 6.11	5195	152.48 ± 5.57	3791	123.84 ± 5.32	2946	50.02 ± 4.42	542	**188.54** ± 5.07	23907
RockSampling88 v2(100)	130.21 ± 5.85	4615	113.39 ± 5.67	4378	108.54 ± 5.28	2990	99.43 ± 5.26	2286	41.93 ± 3.89	420	**149.96** ± 5.38	26922
RockSampling88 v3(100)	116.15 ± 5.03	5732	118.31 ± 5.12	4634	109.51 ± 4.59	3181	102.24 ± 4.67	2403	68.04 ± 4.04	466	**171.84** ± 4.56	24677
LostOrFound (35)	31.23 ± 0.00	32734	31.23 ± 0.00	27317	31.87 ± 0.35	19088	39.68 ± 1.12	14826	**59.50** ± 0.32	4019	37.92 ± 0.96	74955
LostOrFound (100)	50.32 ± 0.79	35012	59.86 ± 0.22	33141	58.27 ± 0.27	22519	55.42 ± 0.37	15824	31.23 ± 0.00	3462	**63.05** ± 0.16	114450

observation spaces or with highly stochastic observation process, like *TigerGrid*, *Hallway2* or *Museum* problems, rejection sampling requires much more time (from to 2 to 8 times more than IS, i.e., from 48 s to 381 s for *TigerGrid* problem) because of a high reject rate.

Results with proposed variants To speed up convergence, lru-ρ-POMCP and lvu-ρ-POMCP variants (cf. Sec. 4.3) have been investigated replacing moving averages by up-to-date estimates, which are less biased in our setting. However, up to now, experiments with fixed time budgets (100 ms, 200 ms, 1 s and 10 s) have not shown any significant improvement.

6 DISCUSSION

In this article, we proposed two algorithms, ρ-POMCP and ρ-beliefUCT, to address ρ-POMDPs without constraints on the reward function. ρ-POMCP (and trivially ρ-beliefUCT) is proved to asymptotically converge when ρ is continuous and bounded. ρ-MCTS-algorithms are thus particularly useful when considering non-convex non-Lipschtiz-continuous rewards which cannot be addressed by previous approaches (like [2], [10] or [19]).

Conducted experiments show that both algorithms give better results than the proposed baseline. The difference between ρ-POMCP and direct use of POMCP ($\beta = 0$ in Table 2) is less significant. However, we proposed problems (like *LostOrFound*) where the use of a particle filter generates better results with a fixed-time budget (due to poorly estimated beliefs when using small particle bags).

This advantage of ρ-POMCP over POMCP would probably increase with a better use of the available time budget. We have observed that ρ-POMCP is time consuming due to the many sampled particles. One promising direction would be to save time by letting a cumulative bag $B(h)$ grow sub-linearly (rather than linearly) with the number of visits $N(h)$, *i.e.*, **by considering dynamic** $|\beta|$. This will require modifying the way particles are generated, which is no simple task since particles need to be propagated from the root to the end of each trajectory.

Another issue that even concerns vanilla POMCP and particle filtering is how to deal with **unexpected transitions**. If, after actually performing an action a, the actual observation z has been rarely sampled (if at all), the new root may come with a poor belief estimate or not exist at all, so that the online computations will give poor results if they can be run at all. In this case, performing particle filtering from the root belief's parent might provide a new root belief but with a high computational cost. Another complementary improvement would be to regularly sample particles from the initial POMDP belief to add new particles in the current root and prevent particle depletion during episodes.

Finally, we hope that MCTS approaches such as ρ-POMCP will provide a way to cope with many states, actions or observations. Future work includes considering **continuous ρ-POMDPs** (*i.e.*, with continuous states/actions/observations), *e.g.*, building on Sunberg and Kochenderfer [25] [26], in order to address information-gathering problems in a robotic context.

ACKNOWLEDGEMENTS

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see `https://www.grid5000.fr`).

REFERENCES

[1] M. Araya-López. *Near-Optimal Algorithms for Sequential Information-Gathering Decision Problems*. PhD thesis, Université de Lorraine, February 2013.

[2] M. Araya-López, O. Buffet, V. Thomas, and F. Charpillet. A POMDP extension with belief-dependent rewards, in *NIPS-10*, 2010.

[3] E. Bargiacchi. Dynamic resource allocation for multi-camera systems. Master's thesis, University of Amsterdam, 2016.

[4] R. Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(5), 1957.

[5] M. Bonneau, N. Peyrard, and R. Sabbadin. A reinforcement-learning algorithm for sampling design in Markov random fields, in *ECAI-12*, 2012.

[6] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of Monte Carlo tree search methods. *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-06)*, 2006.

[8] A. Doucet, S. Godsill, and A. Christophe. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[9] M. Egorov, M. J. Kochenderfer, and J. J. Uudmae. Target surveillance in adversarial environments using POMDPs, in *AAAI-16*, 2016.

[10] M. Fehr, O. Buffet, V. Thomas, and J. Dibangoye. ρ-POMDPs have Lipschitz-continuous ϵ-optimal value functions, in *NIPS-18*, 2018.

[11] D. Fox, W. Burgard, and S. Thrun. Active Markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3–4), 1998.

[12] J.-B. Grill, M. Valko, and R. Munos. Blazing the trails before beating the path: Sample-efficient Monte-Carlo planning, in *NIPS-16*, 2016.

[13] M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling, , in J. F. Lemmer and L. N. Kanal, eds., *Uncertainty in Artificial Intelligence*, volume 5 of *Machine Intelligence and Pattern Recognition*, pages 149–163. Elsevier, 1988.

[14] L. Kocsis and C. Szepesvari. Bandit based Monte-Carlo planning, in *ECML-06*, 2006.

[15] M. Lauri and R. Ritala. Planning for robotic exploration based on forward simulation. *Robotics and Autonomous Systems*, 83: 15–31, 2016.

[16] M. Littman, A. Cassandra, and L. Kaelbling. Learning policies for partially observable environments: Scaling up, in *ICML-95*, 1995.

[17] L. Mihaylova, T. Lefebvre, H. Bruyninckx, and J. D. Schutter. Active robotic sensing as decision making with statistical methods, in *Data Fusion for Situation Monitoring, Incident Detection, Alert and Response Management*. 2006.

[18] K. Åström. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1), 1965.

[19] Y. Satsangi, S. Whiteson, and M. T. J. Spaan. An analysis of piecewise-linear and convex value functions for active perception POMDPs. Technical Report IAS-UVA-15-01, IAS, Universiteit van Amsterdam, 2015.

[20] Y. Satsangi, S. Whiteson, F. A. Oliehoek, and M. T. J. Spaan. Exploiting submodular value functions for scaling up active perception. *Autonomous Robots*, 42:209–233, 2018.

[21] D. Silver and J. Veness. Monte-Carlo planning in large POMDPs, in *NIPS-10*, 2010.

[22] T. Smith. *Probabilistic Planning for Robotic Exploration*. PhD thesis, The Robotics Institute, Carnegie Mellon University, 2007.

[23] T. Smith and R. Simmons. Heuristic search value iteration for POMDPs, in *UAI-04*, 2004.

[24] M. T. Spaan, T. S. Veiga, and P. U. Lima. Decision-theoretic planning under uncertainty with information rewards for active cooperative perception. *JAAMAS*, 29(6), 2015.

[25] Z. N. Sunberg and M. J. Kochenderfer. POMCPOW: an online algorithm for POMDPs with continuous state, action, and observation spaces. *CoRR*, abs/1709.06196, 2017.

[26] Z. N. Sunberg and M. J. Kochenderfer. Online algorithms for POMDPs with continuous state, action, and observation spaces, in *ICAPS-18*, 2018.