Analysis and a home assistance application of online AEMS2 planning

Előd Páll, Levente Tamás, Lucian Buşoniu

Abstract— **We consider an online planning algorithm for partially observable Markov decision processes (POMDPs), called Anytime Error Minimization Search 2 (AEMS2). Despite the considerable success it has enjoyed in robotics and other problems, no quantitative analysis exists of the relationship between its near-optimality and the computation invested. Exploiting ideas from fully-observable MDP planning, we provide here such an analysis, in which the relationship is modulated via a measure of problem complexity called near-optimality exponent. We illustrate the exponent for some interesting POMDP structures, and examine the role of the informative heuristics used by AEMS2 in the guarantees. In the second part of the paper, we introduce a domestic assistance problem in which a robot monitors partially observable switches and turns them off if needed. AEMS2 successfully solves this task in real experiments, and also works better than several state of the art planners in simulation comparisons.**

I. Introduction

Partially observable Markov decision processes (POMDPs) model sequential decision-making problems where states cannot be directly measured, but are inferred instead from indirect observations, while a cumulative reward must be maximized. POMDPs are ideal for many robotic tasks affected by uncertainty, e.g. due to limited sensing, interaction with humans, etc. [1]–[6]. Online planning methods solve POMDPs by running a forward search in the tree of future sequences of actions and observations, at each step of robot-environment interaction [7]. Then, an action is chosen and the procedure repeats at the next step. Many online planners have been proposed, e.g. [4], [5], [8]–[13], and successfully applied to simulated robotic tasks [9], [11], [13] and a few real ones [4], [5].

We focus here on Anytime Error Minimization Search 2 (AEMS2) [8], which expands at each iteration a tree node contributing maximally to the uncertainty of an optimistic policy, i.e. one assuming the best possible rewards. Despite its introduction nearly a decade ago, AEMS2 has proven difficult to beat, consistently staying near the top of online planner rankings [7], [13], [14]. While more recent algorithms like DESPOT [12] and POMCP [10] are able to tackle larger-scale problems than AEMS2, as soon as it works, AEMS2 is usually near-optimal.

The first contribution of our paper is to clarify the reasons for this success, by providing a quantitative analysis of AEMS2. While [14] showed that the algorithm achieves ε-optimality within some finite time, our guarantees are novel by tightly linking the computation budget n with the value of ε. Specifically, we extend the analysis of optimistic planning for MDPs [15] to the partially observable case, where a new challenge is to exploit the informed lower and upper bounds used by AEMS2 at tree leaves. The lower bound requires a consistency property, which is satisfied by the often-used blind policy heuristic [16]. The convergence rate of ε to 0 is expressed via a near-optimality exponent, and we prove that good bounds attain an exponent at most as large as in [15]. We study the value of the exponent in some specific POMDP structures, where it is driven by the observation probabilities.

Our second contribution is defining a new robotic task, motivated by safekeeping disabled or elderly people in a known domestic environment. Such people are often affected by memory decline, which raises risks such as fire due to ovens or other electrical devices left on, flooding because of open faucets etc. In this context, our demonstrator is a simplified scenario in which the robot must look after electrical switches and turn off any of them left on. Since the switch states cannot be observed with certainty, a POMDP model is required. We describe this model and our platform, a Cyton Gamma 1500 arm mounted on a Pionner 3-AT mobile base. We first compare in simulations AEMS2 with state-of-the art planners including online DESPOT [12], DHS [13], FHHOP [11], and offline SARSOP [17]. Since AEMS2 obtains better performance for small budgets, it is then applied to the real task, which it solves successfully. We also illustrate numerically the analytical bounds.

This task joins the – not very numerous – ranks of real robotic POMDPs, e.g. [1]–[6]. It is closest to the mobile manipulation tasks of [2], [6] and similarly motivated to the assistive tasks of [1], [3]. Related analytical work includes [10], [12], where probabilistic finite-budget guarantees are given for POMCP and DESPOT. Motivated by the relation of MDP complexity with the packing number of near-optimal solutions [18], we believe that a deep link also exists between the near-optimality exponent of the POMDP and the covering number studied as a measure of complexity by [19], [20].

Next, Section II presents POMDPs and AEMS2, and Section III the analysis. Section IV describes the home assistance task and Section V the results. Section VI concludes.

II. Background

A. Partially observable Markov decision processes

We introduce POMDPs following [7] and adapting the notation to be convenient for the algorithm. States are denoted $s \in S$, actions $u \in U$, and observations $z \in Z$. Upon taking an action u in state s: the state changes to s' with probability $T(s, u, s')$ where $T : S \times U \times S \to [0, 1]$ is the transition function; a reward $r = R(s, u)$ is received, where $R : S \times U \to \mathbb{R}$ is the reward function; and an

The authors are with the Automation Department, Technical University of Cluj-Napoca, Romania (contact: pall.elod@gmail.com). This work was supported by two grants of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project numbers PNII-RU-TE-2012-3-0040 and PNII-RU-TE-2014-4-0942.

observation z is made with probability $O(s', u, z)$ where $O : S \times U \times Z : [0,1]$ is the observation function. We use the generic prime notation to indicate a quantity at the next step. All spaces S, U, Z are taken discrete and finite, and let $|U| = M$, $|Z| = N$ denote the cardinalities of the action and observation spaces. We assume bounded rewards, and by convention rescale them to the interval $[0, 1]$ without changing the solution.

The information accumulated from all actions and observations so far can be represented via a *belief state* x [7]: a distribution $x(s)$ over the underlying states s, belonging to the space X of probability distributions over S. The agent starts from some initial belief x_0, and if at step k it executes u_k and observes z_{k+1}, then it updates its belief with $x_{k+1} = \tau(x_k, u_k, z_{k+1})$, where τ is defined next. First, the probability of observing z given x and u is:

$$\mathrm{P}(z \,|\, x, u) = \sum_{s'} O(s', u, z) \sum_{s} T(s, u, s') x(s) \tag{1}$$

Then:

$$x'(s') = \tau(x, u, z)(s') = \frac{O(s', u, z)}{\mathrm{P}(z \,|\, x, u)} \sum_{s} T(s, u, s') x(s) \tag{2}$$

Define now a new reward function $\rho(x, u) := \sum_s x(s) R(s, u)$, and new probabilities:

$$f(x, u, x') := \begin{cases} \mathrm{P}(z \,|\, x, u), & \text{if } \exists z \in Z \text{ s.t. } x' = \tau(x, u, z) \\ 0, & \text{otherwise} \end{cases}$$

These two quantities define the so-called *belief MDP*, in which the belief state x evolves in a Markov fashion based on the actions u. Then, the agent's behavior is sufficiently represented by a deterministic policy $\pi : X \to U$, and the value of such a policy from initial belief x_0 is:

$$V^{\pi}(x_0) = \mathrm{E}\left\{ \sum_{k=0}^{\infty} \gamma^k \rho(x_k, \pi(x_k)) \right\}$$

where the expectation is taken over the random belief trajectories $x_{k+1} \sim f(x_k, \pi(x_k), \cdot)$. The objective in a POMDP is to act according to the optimal policy $\pi^*(x) = \arg\max_{\pi} V^{\pi}(x)$, which is related to maximizing the expected sum of discounted rewards in the original POMDP. Denote $V^*(x) = V^{\pi^*}(x)$.

B. Algorithm: Anytime Error Minimization Search 2

We describe next AEMS2 [8]. It is in fact equivalent to our optimistic planning for MDPs (OPMDP) [15] applied to the belief MDP, which allows us to exploit the analysis of OPMDP and extend it to the partially observable case. So we introduce AEMS2 in a new way, convenient for this analysis.

At each step k, AEMS2 explores possible solutions (action sequences) from the current belief x_k. It iteratively refines solutions until e.g. exhausting a computational budget n, and then returns an action u_k based on the reward information accumulated. This action is applied to the system and the procedure is repeated from the new belief. We relabel by convention the current time k to 0, while the procedure of course works at any step.

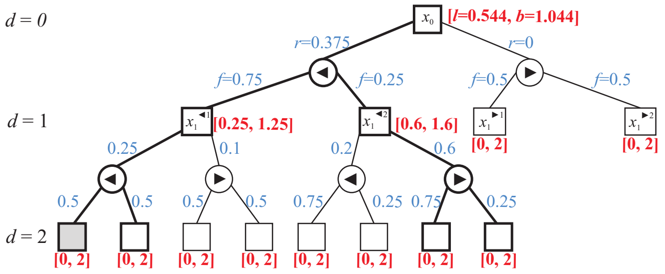

Fig. 1. Example of planning tree for $N = M = 2$. The two actions are symbolically denoted ◀, ▶. The squares are belief state nodes y, labeled by beliefs x, where superscripts index the possible actions and observation outcomes, while subscripts are depths, which increase only with the belief node levels. The actions $u \in \{◀, ▶\}$ are included as circle nodes. Arcs leading to actions are labeled by the rewards $\rho(x, u)$, while arcs leading to beliefs are labeled by the probabilities $f(x, u, x')$, both in blue. The l and b-values are shown near the nodes in red. The thick subtree shows a tree policy. Discount factor γ is 0.5, and we use $\underline{V}_{\mathrm{u}} = 0, \bar{V}_{\mathrm{u}} = \frac{1}{1-\gamma} = 2$. (Figures in this paper are best viewed in color.)

The planning process can be visualized using a tree structure $\mathcal{T}$, exemplified in Figure 1. Each node in the planning tree has MN children. Specifically, a node y labeled by belief x is expanded by adding, for each action u and for each observation z, a new child node y' labeled by the updated belief $x' = \tau(x, u, z)$. Denote the children of x along action u by $\mathcal{C}(y, u)$, the leaves of a tree $\mathcal{T}$ by $\mathcal{L}(\mathcal{T})$, and the belief label of node y by $x(y)$. Note that beliefs may be duplicated if they are encountered along multiple action-observation paths. A tree represents many possible stochastic evolutions of the system, e.g. for the sequence of actions [◀, ▶] there are four possible belief trajectories in the tree of Figure 1: those ending in the 3rd, 4th, 7th, and 8th leaves at depth 2.

The algorithm computes upper and lower bounds b and l on the values of node beliefs, starting from initial bounds $\underline{V}$ and $\bar{V}$ at the leaves, where $\underline{V}(x) \leq V^*(x) \leq \bar{V}(x)$. These initial bounds can be computed in various ways [16], and at worst can be taken to be the uninformed $\underline{V}_{\mathrm{u}} = 0$, $\bar{V}_{\mathrm{u}} = \frac{1}{1-\gamma}$. The bounds are backed up using dynamic programming updates along the tree, e.g. for the b-values:

$$b(y) = \begin{cases} \bar{V}(x(y)), & \text{if } y \text{ is a leaf} \\ \max_u [\rho(x(y), u) + \gamma \sum_{y' \in \mathcal{C}(y,u)} p(y') b(y')], & \text{else} \end{cases} \tag{3}$$

where $p(y') = f(x(y), u, x(y'))$. E.g. in Figure 1, the b-value of the root is $\max\{0.375 + \gamma(0.75 \cdot 1.25 + 0.25 \cdot 1.6), 0 + \gamma(0.5 \cdot 2 + 0.5 \cdot 2)\} \approx \max\{1.044, 1\} \approx 1.044$. The l-values are computed similarly but starting from $\underline{V}$ at the leaves. By a straightforward induction from the leaves to the root, we have $l(y) \leq V^*(x(y)) \leq b(y)$ for any node.

To expand a node, an *optimistic tree policy* $h^{\dagger}$ is constructed, by starting from the root y_0 and recursively adding at each node y the N children along one action that achieves the maximum in (3). In contrast to a simple policy π, a tree policy specifies an action choice for each inner belief node reached under the previous choices, i.e. for all trajectory realizations up to the (varying) depth of the tree. Note that $h^{\dagger}$ is a subtree of $\mathcal{T}$. In Figure 1, $h^{\dagger}$ is shown by thick lines, e.g. the optimistic action branch at the root is ◀ because, as seen in the calculations above, this action achieved the maximum in the b-value backup. Once $h^\dagger$ has been found, one of its leaf nodes is selected by maximizing the *contribution*:

$$c(y) = P(y)\,\gamma^{d(y)}[b(y) - l(y)] \tag{4}$$

where $d(y)$ is the depth and $P(y)$ is the probability of reaching y, the product of the individual transition probabilities along the path. E.g. in Figure 1, the gray node has a probability $P(y) = 0.75 \cdot 0.5 = 0.375$, and contribution $c(y) = P(y)\gamma^2[2-0] = 0.1875$. The reader can verify this contribution is maximal on the optimistic policy $h^\dagger$, and equal to that of its immediate sibling, so one of these two nodes is expanded next. Finally, after n nodes have been expanded, an action that maximizes the expected l-value from the root is returned. Algorithm 1 summarizes AEMS2.

Algorithm 1 AEMS2

```
Input: initial belief $x_0$, expansion budget $n$
 1: initialize tree $\mathcal{T}$ with root $y_0$, labeled by $x_0$
 2: for $i = 1, \dots, n$ do
 3:     build optimistic subtree, starting from $h^\dagger \leftarrow y_0$
 4:     while $\mathcal{L}(h^\dagger) \not\subseteq \mathcal{L}(\mathcal{T})$ do
 5:         retrieve a node $y \in \mathcal{L}(h^\dagger) \setminus \mathcal{L}(\mathcal{T})$
 6:         $u^\dagger = \arg\max_u [\rho(x(y), u) + \gamma \sum_{y' \in \mathcal{C}(y,u)} p(y')b(y')]$
 7:         add children $\mathcal{C}(y, u^\dagger)$ to $h^\dagger$
 8:     end while
 9:     select leaf to expand: $y^\dagger \leftarrow \arg\max_{y \in \mathcal{L}(h^\dagger)} c(y)$
10:     expand $h^\dagger$, adding all its children to $\mathcal{T}$
11:     update $b$ and $l$-values on the tree
12: end for
Output: $u_0 = \arg\max_u [\rho(x_0, u) + \gamma \sum_{y' \in \mathcal{C}(y_0,u)} p(y')l(y')]$
```

To get more insight, consider any tree policy h, constructed by assigning actions arbitrarily from the root to the leaves of $\mathcal{T}$; and let $h \in \mathcal{T}$ denote that h was obtained in this way. Define also an *infinitely* long tree policy h_∞, achieved by assigning actions to belief nodes indefinitely, and the expected discounted value $v(h_\infty)$ of such a policy. We have $v^* := \sup_{h_\infty} v(h_\infty) = V^*(x_0)$, motivating us to use tree policies as a local optimal solution at x_0. For each policy h, define lower and upper bounds, and the best value among infinite policies $h_\infty \succ h$ (i.e. starting with h):

$$L(h) = \sum\nolimits_{y \in \mathcal{L}(h)} P(y)\,[r(y) + \gamma^{d(y)} l(y)],$$
$$B(h) = \sum\nolimits_{y \in \mathcal{L}(h)} P(y)\,[r(y) + \gamma^{d(y)} b(y)],$$
$$W(h) = \sum\nolimits_{y \in \mathcal{L}(h)} P(y)\,[r(y) + \gamma^{d(y)} V^*(y)] = \sup_{h_\infty \succ h} v(h_\infty)$$

where $r(y) = \sum_{k=0}^{d(y)-1} \gamma^k r_k$ is the discounted sum of rewards along the path to y. We have $L(h) \leq W(h) \leq B(h)$. The optimistic policy $h^\dagger$ constructed as above maximizes the upper bound B among all tree policies $h \in \mathcal{T}$, so it is the most promising partial solution seen so far. Further, because $B(h) = L(h) + \sum_{y \in \mathcal{L}(h)} c(y) =: L(h) + \delta(h)$, the meaning of $c(y)$ in (4) becomes clear: it is the contribution of leaf y to the uncertainty $\delta(h)$ on the value of h. Thus, finally, the algorithm expands the leaf contributing the most to the uncertainty on the optimistic policy. The action returned at the end in fact corresponds to choosing a tree policy maximizing the lower bound, $\widehat{h} = \arg\max_{h \in \mathcal{T}} L(h)$, a safe choice, and then applying its first action.

The main difference from the description in [8] is that there, contributions are computed for *all* leaves, with:

$$P(y)\,\gamma^{d(y)}[b(y) - l(y)] \prod_{k=0}^{d(y)-1} \mathrm{P}(u_k \,|\, y_k)$$

where $\mathrm{P}(u_k \,|\, y_k)$ is a probability of selecting action u_k from node y_k along the path to y. However, for AEMS2 [8] takes $\mathrm{P}(u_k \,|\, y_k)$ 1 for an action maximizing the upper bound, which means all leaves outside $h^\dagger$ have contribution 0 and will not be selected, just like in Algorithm 1.

In practical implementations, the subtree of the action chosen at the current step is usually taken as a starting point for tree construction at the next step, improving performance. E.g. if action ◀ is chosen in Figure 1, and observation 2 is seen, then the node of $x_1^{◀,2}$ becomes the new root, and all nodes outside its subtree are deleted. Further, bounds must only be backed up along the path from the expanded node to the root, using $O(d(y^\dagger))$ computation, and the next node to expand can be chosen along the same backup, without explicitly constructing $h^\dagger$, see [7].

III. Performance Analysis of AEMS2

We study the near-optimality and convergence rate of AEMS2, by extending the analysis line of OPMDP from [15] to the partially observable case. In [15], the uninformed bounds $\bar{V}_\mathrm{u}, \underline{V}_\mathrm{u}$ were used, and using better bounds turns out to be nontrivial. In particular, $\underline{V}$ requires a consistency property, which is satisfied by the often-used blind policy heuristic [16]. The convergence rate is expressed via a near-optimality exponent β, and we prove that good bounds attain an exponent at most as large as in [15]. Finally, we study the value of β in some interesting POMDPs, where it is driven by the observation function O.

The consistency property prevents that expanding a node and backing up the children bounds leads to a worse bound than the initial one, which would not make sense.

Assumption 1: The lower bound is *consistent*, namely: $\max_u[\rho(x,u) + \gamma \sum_{x'} f(x,u,x')\underline{V}(x')] \geq \underline{V}(x)$.

We start by proving that near-optimality is given by the smallest uncertainty of any optimistic policy refined. This quantity can be computed *a posteriori* by the algorithm. Let subscript i denote quantities at iteration i, e.g. $\mathcal{T}_i$, $h_i^\dagger$, $\widehat{h}_i = \arg\max_{h \in \mathcal{T}_i} L(h)$.

Theorem 2: AEMS2 returns a policy $\widehat{h}_n$ so that $v^* - W(\widehat{h}_n) \leq \delta^*$, where $\delta^* = \min_{i=1}^n \delta(h_i^\dagger)$.

Proof: Consider some iteration i. We have $L(h_i^\dagger) \leq L(\widehat{h}_i) \leq v^* \leq B(h_i^\dagger)$: the first inequality is true because $\widehat{h}_i$ maximizes L, the second by definition, and the third because $B(h_i^\dagger)$ maximizes B among all tree policies, including some policy h' that contains the optimal infinite policy h^* (so that $v^* \leq B(h') \leq B(h_i^\dagger)$). Now, we will show that a consistent

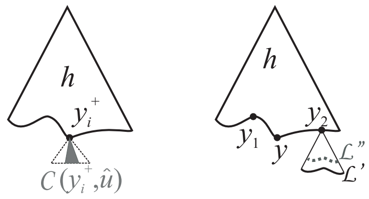

Fig. 2. Illustrations used in the proofs.

bound leads to a better policy choice at the end, $L(\widehat{h}_n) \geq L(\widehat{h}_i)$, which is the main difference from [15]. Then, we will have $v^* - W(\widehat{h}_n) \leq v^* - L(\widehat{h}_n) \leq v^* - L(h_i^\dagger) \leq B(h_i^\dagger) - L(h_i^\dagger)$, where the last difference is $\delta(h_i^\dagger)$. The proposition is then proven, since the inequality holds at any iteration.

We prove the inequality over successive steps, $L(\widehat{h}_{i+1}) \geq L(\widehat{h}_i)$. Consider the set H_i of policies h belonging to the tree at i. If a policy $h \in H_i$ does not contain the expanded node $y_i^\dagger$, then h is also in H_{i+1}. Otherwise, construct from h a policy $h' \in H_{i+1}$ that chooses an action $\widehat{u}$ achieving the maximal lower bound in the backup, $\max u[\rho(x(y_i^\dagger), u) + \gamma \sum_{y' \in \mathcal{C}(y_i^\dagger, u)} p(y') \underline{V}(x(y'))] = l(y_i^\dagger)$; see Figure 2, left. We then have $L(h') = L(h) - \underline{V}(x(y_i^\dagger)) + l(y_i^\dagger)$, which due to Assumption 1 is larger than $L(h)$. Thus, for any policy removed from H_{i+1} one with a larger L is added, leading immediately to $L(\widehat{h}_{i+1}) \geq L(\widehat{h}_i)$. ■

Note that $v^* - W(\widehat{h}_n)$ in Theorem 2 upper-bounds the sub-optimality $v^* - W(u_0)$ of the action choice u_0, which is a good measure of the performance of an online planner.

We next establish that the blind policy lower bound [7], [16] is consistent, which is important because this bound is widely used in AEMS2. The blind policy iterates $V_t^u(s) = R(s,u) + \gamma \sum_{s'} T(s,u,s') V_{t-1}^u(s')$ an arbitrary number of times, starting from $V_0^u = 0$, and then chooses $\underline{V}(x) = \max_u \sum_s x(s) V_t^u(s)$. We must show that the l.h.s. of the consistency inequality:

$$\max_u [\rho(x,u) + \gamma \sum_{x'} f(x,u,x') \underline{V}(x')]$$
$$= \max_u [\rho(x,u) + \gamma \sum_z \mathrm{P}(z \,|\, x,u) \max_{u'} \sum_{s'} x'(s') V_t^{u'}(s')]$$

where $x' = \tau(x,u,z)$, is larger than the r.h.s.:

$$\underline{V}(x) = \max_u \sum_s x(s)[R(s,u) + \gamma \sum_{s'} T(s,u,s') V_{t-1}^u(s')]$$
$$= \max_u [\rho(x,u) + \gamma \sum_s x(s) \sum_{s'} T(s,u,s') V_{t-1}^u(s')]$$

Thus, it suffices to compare the two terms multiplying γ in the two expressions. The former term can be written, using (2) to express τ and noticing that $\mathrm{P}(z \,|\, x,u)$ simplifies:

$$\sum_z \max_{u'} \sum_{s'} O(s',u,z) \Big(\sum_s T(s,u,s') x(s) \Big) V_t^{u'}(s')$$
$$= \sum_z \max_{u'} \sum_{s'} \sum_s O(s',u,z) T(s,u,s') x(s) V_t^{u'}(s')$$
$$= \sum_z \max_{u'} \sum_s x(s) \sum_{s'} O(s',u,z) T(s,u,s') V_t^{u'}(s')$$
$$\geq \max_{u'} \sum_z \sum_s x(s) \sum_{s'} O(s',u,z) T(s,u,s') V_t^{u'}(s')$$
$$= \max_{u'} \sum_s x(s) \sum_{s'} \sum_z O(s',u,z) T(s,u,s') V_t^{u'}(s')$$
$$= \max_{u'} \sum_s x(s) \sum_{s'} T(s,u,s') V_t^{u'}(s')$$
$$\geq \sum_s x(s) \sum_{s'} T(s,u,s') V_{t-1}^u(s')$$

where we used $V_t^u \geq V_{t-1}^u$, manipulated some summation indices, and exchanged a summation with a maximization preserving the desired direction of the inequality. This implies the original consistency inequality.

We move on to an *a priori* guarantee. For any node y, define first a measure of its impact, as the largest diameter of any (finite) policy to which it maximally contributes:

$$\alpha(y) = \sup_{h \in H(y)} \delta(h), H(y) = \{h \,|\, y \in \arg\max_{y' \in \mathcal{L}(h)} c(y')\}$$

Then, for any $\varepsilon \geq 0$ define the set of nodes with (i) at least ε impact to (ii) a near-optimal policy, as follows:

$$Y_\varepsilon = \{y \in \mathcal{T}_\infty | (i) \alpha(y) \geq \varepsilon, \\ (ii) \exists h_\infty \ni y, v^* - v(h_\infty) \leq \alpha(y)\} \quad (5)$$

Intuitively, these nodes are important to examine, and indeed it turns out that AEMS2 only expands such nodes, in the following sense.

Lemma 3: All nodes $y_i^\dagger$ expanded by AEMS2 belong to Y_{α^*}, where $\alpha^* = \min_{i=1}^n \alpha(y_i^\dagger)$.

Proof: First, we show $y_i^\dagger \in Y_{\alpha(y_i^\dagger)}$. Condition (i) is true because $\varepsilon = \alpha(y_i^\dagger)$. For (ii), take h_∞ as the infinite policy that starts with the optimistic one $h_i^\dagger$ and then takes optimal actions starting from its leaves. Then, $v(h_\infty) = W(h_i^\dagger)$, and since $W(h_i^\dagger) \geq L(h_i^\dagger)$ and $v^* \leq B(h_i^\dagger)$ (see again the proof of Theorem 2), we have $v^* - v(h_\infty) \leq \delta(h_i^\dagger)$. Finally, since $y_i^\dagger$ was expanded, it maximizes the contribution on the leaves of $h_i^\dagger$, so that $h_i^\dagger \in H(y_i^\dagger)$, leading to $\delta(h_i^\dagger) \leq \alpha(y_i^\dagger)$, and condition (ii) holds.

Second, observe that $Y_{\varepsilon'} \subseteq Y_\varepsilon$ when $\varepsilon' \geq \varepsilon$, because condition (i) is more restrictive when ε is larger, while (ii) is unaffected. Thus $Y_{\alpha(y_i^\dagger)} \subseteq Y_{\alpha^*}$, completing the proof. ■

To characterize the size of Y_ε, and thereby the complexity of the planning problem at hand, define the *near-optimality exponent* as the smallest number $\beta \geq 0$ so that $|Y_\varepsilon| = \tilde{O}(\varepsilon^{-\beta})$ for $\varepsilon > 0$, i.e. so that $\exists a > 0, b \geq 0, |Y_\varepsilon| \leq a[\log(1/\varepsilon)]^b \varepsilon^{-\beta}$. We assume here $\underline{V}(x_0) < \bar{V}(x_0)$, to ensure β makes sense. We can now provide the final *a priori* bound.

Theorem 4: The near-optimality $v^* - W(\widehat{h}_n)$ of AEMS2 is $\tilde{O}(n^{-1/\beta})$ when $\beta > 0$, or $O(\exp[(-n/a)^{1/b}])$ when $\beta = 0$.

Proof: Using Lemma 3, we have $n \leq |Y_{\alpha^*}| = \tilde{O}(\alpha^{*-\beta})$. By inverting the relationship we obtain $\alpha^* = \tilde{O}(n^{-1/\beta})$ if $\beta > 0$, or $O(\exp[(-n/a)^{1/b}])$. The result follows by Theorem 2 and by noticing that $\delta^* \leq \alpha^*$. ■

The smaller β, the faster AEMS2 converges to an optimal solution. In particular, when $\beta = 0$, i.e. when the problem is easy, convergence is (stretched-)exponential.

Recall that this analysis line extends that of OPMDP in [15], where uninformed bounds $\underline{V}_{\mathrm{u}}, \bar{V}_{\mathrm{u}}$ were used together with a slightly different definition of important nodes:

$$\tilde{Y}_\varepsilon = \{y \in \mathcal{T}_\infty | (i) \tilde{\alpha}(y) \geq \varepsilon, \\ (ii) \exists h_\infty \ni y, v^* - v(h_\infty) \leq N\tilde{\alpha}(y)/\gamma\} \tag{6}$$

Here, $\tilde{\alpha}(y) \geq \alpha_{\mathrm{u}}(y)$ conservatively estimates the impact. Subscript 'u' denotes quantities with uninformed bounds, which are tighter than those obtained with $\tilde{\alpha}$ and (6); the latter are denoted by a tilde. Exponent $\tilde{\beta}$ is defined so that $\tilde{Y}_\varepsilon = \tilde{O}(\varepsilon^{-\tilde{\beta}})$. We expect AEMS2 with good bounds to be faster than uninformed OPMDP, and to illustrate we show that, indeed, good bounds cannot lead to a larger β than $\tilde{\beta}$.

Proposition 5: If $\underline{V} \geq \underline{V}_{\mathrm{u}}, \bar{V} \leq \bar{V}_{\mathrm{u}}$, then $\beta \leq \tilde{\beta}$.

Proof: We first relate $\alpha(y)$ with $\alpha_{\mathrm{u}}(y)$ for any node y. Take any policy $h \in H(y)$. We will construct another policy $h_{\mathrm{u}} \in H_{\mathrm{u}}(y)$, the set with uninformed bounds. There can be two types of nodes on $\mathcal{L}(h)$. Type 1, exemplified by y_1 in Figure 2 right, satisfies $c_{\mathrm{u}}(y_1) \leq c_{\mathrm{u}}(y)$, so we add it to the leaves of h_{u}. The second type, y_2, has larger contribution than $c_{\mathrm{u}}(y)$ so it cannot be a leaf of any policy in $H_{\mathrm{u}}(y)$. Instead, we expand its descendants until we reach a frontier $\mathcal{L}'$, with parents $\mathcal{L}''$, for which $c_{\mathrm{u}}(y') \leq c_{\mathrm{u}}(y) < c(y''), \forall y' \in \mathcal{L}', y'' \in \mathcal{L}''$. We add $\mathcal{L}'$ to h_{u}. Processing all leaves of h in this manner, we construct a complete h_{u}. For type 2 nodes, using the formulas of the uninformed bounds we have $\sum_{y' \in \mathcal{L}'} c_{\mathrm{u}}(y') = \gamma \sum_{y'' \in \mathcal{L}''} c_{\mathrm{u}}(y'') \geq \gamma c_{\mathrm{u}}(y_2) \geq \gamma c(y_2)$. Noticing also that $c_{\mathrm{u}}(y_1) \geq c(y_1) \geq \gamma c(y_1)$ for type 1 nodes, we get $\delta_{\mathrm{u}}(h_{\mathrm{u}}) \geq \gamma \delta(h)$. Since this is true for any $h \in H(y)$, we get $\alpha(y) \leq \alpha_{\mathrm{u}}(y)/\gamma$. Recall $\alpha_{\mathrm{u}}(y) \leq \tilde{\alpha}(y)$.

Consider now sets Y_ε (5) and $\tilde{Y}_{\gamma\varepsilon}$ (6). Using the impact inequalities above, condition (i) $\alpha(y) \geq \varepsilon$ in Y_ε implies (i) $\tilde{\alpha}(y) \geq \gamma\varepsilon$ in $\tilde{Y}_{\gamma\varepsilon}$. Also, $\alpha(y) \leq N\tilde{\alpha}(y)/\gamma$, so (ii) in Y_ε implies (ii) in $\tilde{Y}_{\gamma\varepsilon}$. So, finally, $Y_\varepsilon \subseteq \tilde{Y}_{\gamma\varepsilon}$ and $|Y_\varepsilon| = \tilde{O}((\gamma\varepsilon)^{-\tilde{\beta}}) = \tilde{O}(\varepsilon^{-\tilde{\beta}})$, the desired result. ■

It would be interesting to establish a deeper relation between β and bound quality, which we leave for future work. Here, we exploit instead Proposition 5 to illustrate the complexity of some interesting POMDP structures.

We focus on the role of the observation function O, and take for simplicity $O(s', u, z) = o(z), \forall s', u$ and $T(s, u, s') = t(s'), \forall s, u$. Then, using (1), we get the following transition probabilities $\mathrm{P}(z \,|\, x, u)$ in the belief MDP:

$$\sum_{s'} O(s', u, z) t(s') \sum_{s} x(s) = o(z) \sum_{s'} t(s') = o(z), \forall x, u$$

Exponents $\tilde{\beta}$ were studied in [15] for such types of MDPs, so using them we can get an upper bound and some insight on β. Two particular edge cases are most informative. In the first, o is the uniform distribution, leading to a difficult problem with high uncertainty; such probabilities are exemplified in boldface in Figure 3, for $N = 2$. In the second case, o is a Bernoulli distribution $q, 1 - q$ with q close to 1, i.e., one observation among the two has high probability, leading to an easier problem with informative observations. These probabilities are in parentheses in Figure 3. To isolate the effect of o we take for now all the rewards equal to 1.

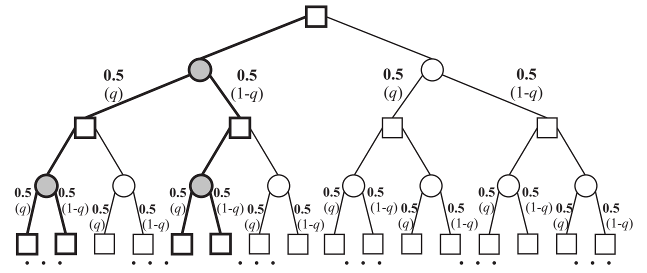

Fig. 3. Example tree structures for different values of β.

When o is uniform, we find that $\beta \leq \tilde{\beta} = \frac{\log NM}{\log 1/\gamma}$, which is also the largest possible value. Here NM is the branching factor of the tree that AEMS2 explores, and the value tells us that the entire tree may have to be explored, so the algorithm converges slowly, cf. Theorem 4. For Bernoulli observations, we consider the limit case in which $q \to 1$. Then, $\tilde{\beta} \to \frac{\log M}{\log 1/\gamma}$, so only the high-probability branches of the tree are explored, a deterministic M-action problem results, and AEMS2 converges faster.

To understand the influence of the rewards, consider again the uniform o but now let a single action gets rewards of 1 everywhere on the tree, while the other actions get 0, see again Figure 3 where the high-reward action nodes are gray. In this case, the exponent is around $\frac{\log N}{\log 1/\gamma}$ (see [15] for details), so the branching due to the actions is eliminated, and the algorithm only explores the single optimal policy, shown as a thick subtree in Figure 3.

IV. Robotic home assistance application

We consider a scenario in which a robot must look after electrical switches and turn off any of them left on. Since the switch states cannot be observed fully accurately, not all the switches can be seen at once, and the motion of the robot must be controlled, a POMDP model is ideal for this task.

A. POMDP model of the home assistance task

The states of the home assistance task can be divided in two classes: locations and switch states. The robot location in the environment is discretized into a grid coordinate $\lambda \in \Lambda$, which is fully observable. The state of each switch i, $w_i \in \{\text{on}, \text{off}\}$, is partially observable. The state space consists of all combinations of grid locations and switch states, $S = \Lambda \times W$. For switch i, we define its location λ_i, as well 3 sets of locations $\Lambda_{i,j}$, $j \in \{A, B, C\}$ from which it is observable with different probabilities, as explained below. Figure 4 illustrates this type of model for our real lab.

The possible actions are $U = \{\text{north}, \text{east}, \text{south}, \text{west}, \text{stay}, \text{flip}, \text{observe}\}$, out of which the first five are deterministic. The four motion actions navigate the robot between cells relatively to the map. If an action would lead to leaving the map or hitting an obstacle, then the robot does not move. Action 'stay' keeps the robot in the same cell. Action 'flip' is stochastic and succeeds only with 80% probability, a value which we computed from experiments, and which is due to the limited arm accuracy of about 1 cm (because of the arm construction and imprecise visual feedback).

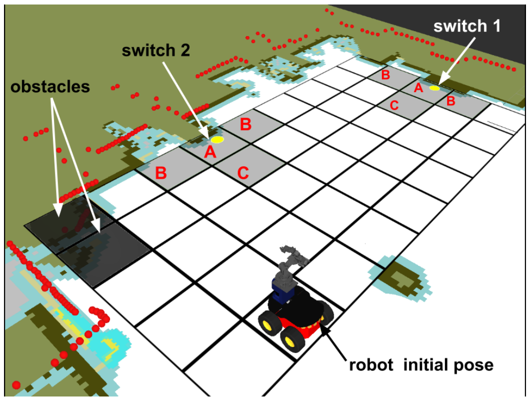

Fig. 4. Discretized map of size (5×9), where the yellow circles are the location of the switches. The rightmost corner is (1, 1) and the initial location of the robot is (1, 8). Dark gray cells are obstacles and light gray cells denote the observation ranges of three types A, B, C.

TABLE I

SENSING PROBABILITIES FOR ONE SWITCH.

Location type	switch on		switch off	
	TP	FN	TN	FP
A	0.82	0.18	0.78	0.22
B	0.84	0.16	1.00	0.00
C	0.89	0.11	0.93	0.07

The observation is limited to the switches: $z_i \in \{\text{on}, \text{off}, \text{unobserved}\}$. Observation probabilities depend on the robot's location relative to the switch, and on the switch state. Table I shows the probabilities of sensing true positives TP and false negatives FN when the switch is on, and true negatives TN and false positives FP when it is off, for each type of location $j \in \{A, B, C\}$. These sensing probabilities were estimated experimentally. Using them, the observation function for a single switch i is defined:

$$O_i(s', \text{observe}, \cdot) = \begin{cases} [TP_j - \frac{\epsilon}{2}, FN_j - \frac{\epsilon}{2}, \epsilon], & \text{if } w'_i = \text{on}, \lambda' \in \Lambda_{i,j} \\ [FP_j - \frac{\epsilon}{2}, TN_j - \frac{\epsilon}{2}, \epsilon], & \text{if } w'_i = \text{off}, \lambda' \in \Lambda_{i,j} \\ [0, 0, 1], & \text{otherwise} \end{cases}$$

Here, $\epsilon = 0.001$ is a small probability of not observing the switch even in the observable range, which we introduced to cover rare situations when the camera view is obstructed. For example, if action 'observe' is taken in cell A with the switch on, the probabilities of seeing 'on', 'off', and 'unobserved' are $0.82 - \frac{\epsilon}{2}, 0.18 - \frac{\epsilon}{2}$, and ϵ, respectively. We assume the observation ranges of different switches are disjoint, so the overall observation function O is easy to compute.

The reward function is defined as follows:

$$R(s, u) = \begin{cases} 1 & \text{if } u = \text{flip}, \lambda = \lambda_i, w_i = \text{on} \\ 0 & \text{if } u = \text{flip}, \lambda = \lambda_i, w_i = \text{off} \\ 0.601 & \text{if } u = \text{stay (for any } s) \\ 0.6 & \text{otherwise} \end{cases} \tag{7}$$

Recall that reward values must be normalized between 0 and 1. Maximum reward 1 is given for turning off an on switch, and the smallest reward 0 (maximum penalty) is given for flipping an off switch to on. In other cases, a better than neutral reward is given, 0.6, to avoid a misleading solution that infinitely toggles a switch (getting average reward 0.5). A marginally greater reward was chosen for 'stay' to reduce energy consumption.

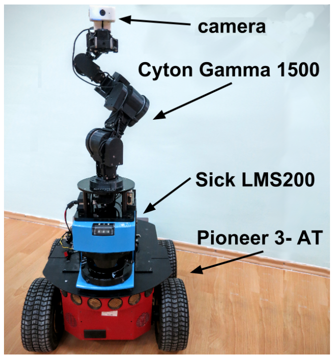

Fig. 5. Assistive robot.

B. Hardware and software

To accomplish the task defined above, a mobile robot capable of manipulation and localization is needed. We chose a Pioneer 3-AT wheeled robot as the base platform. We mounted a Sick LMS200 laser scanner, which is used together with the odometry sensors built into the mobile base to localize the robot. On top of the laser scanner we placed the Cyton Gamma 1500 light-weight robotic arm with 7 DOF. A camera mounted above the arm's gripper is used for switch state sensing and for vision feedback in arm trajectory control. The overall configuration is shown in Figure 5.

The application is written in C++ and Python, and runs under the Robot Operating System (ROS). The two most important ROS packages we use are amcl (`wiki.ros.org/amcl`), to perform probabilistic localization for 2D navigation, and MoveIt! (`moveit.ros.org`), for mobile manipulation with motion planing algorithms. We implemented a centralized controller that calls the low-level motion and sensing procedures needed by the POMDP solver to execute actions and make measurements. Among these procedures, we describe switch state sensing and flipping, which are more involved.

We use switches with LED indicators, common in home environments. Switch state sensing is based on LED blob detection and marker pose estimation on a 2D image. A 4×5 chessboard marker is placed above each switch, from which a region of interest on the image is defined containing the LED. We search for a dark red blob corresponding to the turned-off LED in this region. E.g. if the switch is on, then the LED is off and the blob will be found, see also Figure 8.

The 'flip' action is a sequence of plans generated with MoveIt!, using vision feedback. The relative pose of the marker is used to compute goal positions for the end effector, as well as position corrections after execution. Initially, it is necessary to have visual contact with the marker, so the end effector is oriented perpendicular to the marker plane, at a 12 cm distance. Next, the arm is moved vertically to aim for the switch button. Finally, the button is pushed by

TABLE II
PERFORMANCE COMPARISON ON THE HOME ASSISTANCE MODEL.

	τ = 1 ms	τ = 10 ms	τ = 100 ms	τ = 500 ms	τ = 750 ms	τ = 1000 ms
AEMS2	9.858 (9.844, 9.872)	9.897 (9.879, 9.915)	9.903 (9.885, 9.922)	9.920 (9.896, 9.943)	9.874 (9.858, 9.891)	9.876 (9.858, 9.895)
DH	9.424 $--$	9.424 $--$	9.856 (9.840, 9.871)	9.866 (9.847, 9.885)	9.835 (9.812, 9.858)	9.871 (9.850, 9.891)
FHHOP	9.424 $--$	9.424 $--$	9.842 (9.826, 9.859)	9.837 (9.815, 9.859)	9.853 (9.830, 9.876)	9.856 (9.875, 9.836)
DESPOT	9.566 (9.528, 9.603)	9.548 (9.516, 9.579)	9.613 (9.587, 9.639)	9.643 (9.601, 9.685)	9.641 (9.604, 9.678)	9.650 (9.609, 9.691)
SARSOP	9.877 (9.857, 9.898); offline execution time 1360 ms					

constraining the orientation and position of the end effector so that it only translates on one axis.

The application is distributed on two machines: the low-level controllers and sensing run on a laptop placed on the mobile platform, while the computationally intensive POMDP planning algorithm runs on a desktop computer, together with 3D visualisation.

V. PLANNING RESULTS FOR HOME ASSISTANCE

Our goal in this section is to solve the home assistance task using POMDP planning. We start with simulations that compare AEMS2 with several state of the art online planners: DHS [13], FHHOP [11], and DESPOT [12], as well as the SARSOP [17] offline planner as a baseline. The performance of these planners on standard POMDPs is known [7], [10], [12]–[14], so here we only apply them to the new task. We then evaluate the winner, AEMS2, in the real-life task.

We implemented AEMS2, DHS, and FHHOP in the appl package (`http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/`), which originally includes SARSOP; and we used the publicly available appl implementation of DESPOT. For all algorithms, we initialize lower and upper bounds $\underline{V}, \bar{V}$ with the blind policy and Fast Informed Bound techniques [16], respectively. To improve performance, we reuse subtrees and (for AEMS2, FHHOP and DHS) integrate leaf selection with the bounds backup, as described in Section II-B. The experiments are carried out on a desktop with a quad-core Xeon E5-1620 3.70GHz CPU and 16GB RAM (while using the laptop for low-level control).

A. Algorithm comparison in simulation

The algorithms are applied to the model of our real lab, shown in Figure 4. This has $|\Lambda| = 43$ location states, two switches leading to $|W| = 4$ switch states, $|U| = 7$ actions, and $|Z| = 9$ observations. The initial robot location is (1,8), as shown in Figure 4, and both switches are on. The initial belief state x_0 is a uniform distribution over the possible switch states. With reward function (7), DHS and FHHOP get stuck choosing only action 'stay'. So, to make the comparison informative, we change for all algorithms the reward of 'stay' to 0.6, equal to the neutral reward of navigation and observation actions.

We compare the discounted return of the online algorithms over trajectories of 30 steps, as a function of the computational budget, expressed as the time allowed for tree expansion to choose one action, $\tau = 1, 10, 100, 500, 750, 1000$ ms. SARSOP controls the robot using a solution found offline, with a precision of 0.000099 [17]. The discount factor is $\gamma = 0.95$. We ran 100 independent experiments for each setting,

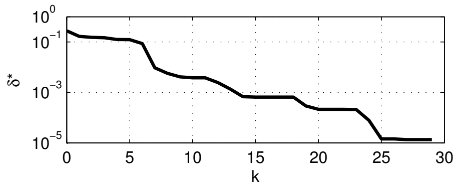

Fig. 6. Evolution of δ^*.

and computed the mean return with its 95% confidence interval. These results are shown in Table II.

AEMS2 has better performance for all $\tau \leq 500$, while for greater values of τ statistically none of the algorithms outperforms the others (but AEMS2 still has the largest mean performance). This confirms the good quality of AEMS2 predicted by our analysis. Note that the different returns correspond to real differences in task-solving performance, e.g. for $\tau = 1$ and 10, DHS and FHHOP still always choose action 'stay', while AEMS2 and DESPOT do not, even though they all use the modified rewards. Returns are usually larger for larger budgets, as the analysis indicates, although there are exceptions, e.g. for $\tau = \{750,\ 1000\}$ AEMS2 has slightly smaller performance than for $\tau = \{100,\ 500\}$. These exceptions stay within the theoretical bounds, and may occur due to nonmonotonic returns in the planning horizon. SARSOP returns an offline, global solution with performance that is statistically similar to AEMS2. The execution time of SARSOP is, however, larger than the planning time used by AEMS2 over the entire 30-step trajectory when $\tau \in \{1, 10\}$ ms, so AEMS2 is computationally preferable for these budgets.

Figure 6 shows the near-optimality bounds δ^* of Theorem 2, for each step k along a representative trajectory. The bound becomes very tight because the subtrees are reused, and also because the local planning problems become simpler as the task is being solved. To illustrate Theorem 4, we created a very large tree using $\tau = 60$ s from the initial state, and computed the average branching factor on this tree, which is related to β as explained in Section III. The result is around 2.23, much smaller than the worst-case branching factor $21 = |U| \times 3$ (recall that at most 3 observations have nonzero probability at any state).

Both DHS and FHHOP combine the leaf selection criterion of AEMS2 with additional criteria, called LSEM in DHS and simply H_L in FHHOP, so it is interesting to examine how many times the different criteria are chosen. For $\tau = 100$, FHHOP chooses the AEMS2 criterion 69% of the time and LSEM 31%, while DHS strongly prefers H_L, picking AEMS2 selection less then 1% of the time.

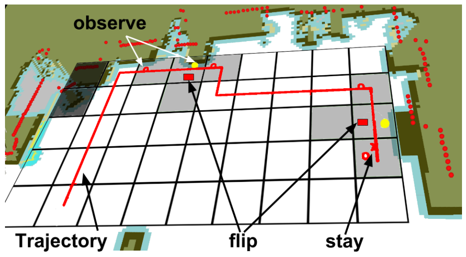

Fig. 7. Real robot trajectory (red line). Red circles are 'observe' actions, red rectangles are 'flip' actions, and x is 'stay'.

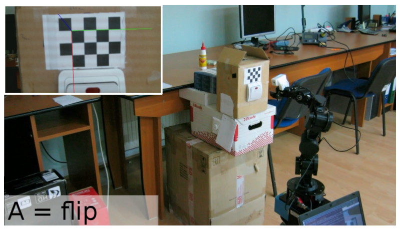

Fig. 8. Robot executing a flip action; the top-left inset is the camera view.

B. *Real-life experiment*

Given the better performance for smaller budgets, AEMS2 is selected for the real experiment. We reinstate the reward 0.601 for the 'stay' action, since reducing power consumption is important in practice. The initial state is the same as above. A budget of $n = 2000$ node expansions is imposed for each action search (leading to execution times $\tau \approx 78\,\text{ms}$), while visualizing navigation and arm manipulation in Rviz.

The trajectory in the real experiment is shown in Figure 7. The robot determines that both switches are on and flips them off, and once it reaches this goal, it chooses 'stay'. An interesting feature related to partial observability is that the robot always performs two consecutive observations to reduce uncertainty, from locations of type B where sensing accuracy is the highest. Furthermore, observations are made before *and* after flipping a switch, to make sure the switch is off, given the 80% success rate of the action.

Figure 8 illustrates the robot in action, see also the full demo video at `https://youtu.be/xc4t6p9dVFM`. Our software is publicly available under BSD licensing at `http://rocon.utcluj.ro/node/171`.

VI. Conclusions and Future Work

The near-optimality of the AEMS2 algorithm was analyzed as a function of the computation budget invested. The algorithm was successfully applied to a new domestic assistance problem, in which a robot monitors partially observable switches and turns them off if needed.

The next step for the application is monitoring the human, to prevent the robot from becoming a nuisance (collisions, turning off lights that are actually needed), revisit old switches in case they were turned back on, etc. On the analytical side, it is important to quantitatively exploit the informed AEMS2 bounds and to understand the complexity of more realistic POMDP structures. It is also interesting to investigate the relation between the near-optimality exponent and the covering number of [19], [20].

References

[1] J. Pineau and S. Thrun, "High-level robot behavior control using POMDPs," in *AAAI Workshop on Cognitive Robotics*, Edmonton, Canada, Aug 2002.

[2] G. A. Hollinger, D. Ferguson, S. S. Srinivasa, and S. Singh, "Combining search and action for mobile robots," in *IEEE International Conf. on Robotics and Automation (ICRA-09)*, Kobe, Japan, 12–17 May 2009, pp. 952–957.

[3] T. Taha, J. V. Miró, and G. Dissanayake, "A POMDP framework for modelling human interaction with assistive robots," in *IEEE International Conf. on Robotics and Automation (ICRA-11)*, Shanghai, China, 9–13 May 2011, pp. 544–549.

[4] P. Monso, G. Alenyà, and C. Torras, "POMDP approach to robotized clothes separation," in *IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS-12)*, Vilamoura, Portugal, 7–12 October 2012, pp. 1324–1329.

[5] C. P. C. Chanel, C. Lesire, and F. Teichteil-Königsbuch, "A robotic execution framework for online probabilistic (re)planning," in *24th International Conf. on Automated Planning and Scheduling, (ICAPS-14*, Portsmouth, US, 21–26 Jun 2014.

[6] N. A. Vien and M. Toussaint, "POMDP manipulation via trajectory optimization," in *IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS-15)*, Hamburg, Germany, 28 Sept - 2 Oct 2015, pp. 242–249.

[7] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, "Online planning algorithms for POMDPs," *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.

[8] S. Ross and B. Chaib-draa, "AEMS: An Anytime Online Search Algorithm for Approximate Policy Refinement in Large POMDPs," in *20th International Joint Conf. on Artificial Intelligence (IJCAI-07)*, Hyderabad, India, 6–12 Jan 2007, pp. 2592–2598.

[9] S. Ross, B. Chaib-draa, and J. Pineau, "Bayesian reinforcement learning in continuous POMDPs with application to robot navigation," in *IEEE International Conf. on Robotics and Automation, (ICRA-08)*, Pasadena, US, 19–23 May 2008, pp. 2845–2851.

[10] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Advances in Neural Information Processing Systems 23*. MIT Press, 2010, pp. 2164–2172.

[11] Z. Zhang and X. Chen, "FHHOP: A Factored Hybrid Heuristic Online Planning Algorithm for Large POMDPs," in *28th Conf. on Uncertainty in Artificial Intelligence (UAI2012)*, Catalina Island, USA, 15 – 17 Aug 2012, pp. 934–943.

[12] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," in *Advances in Neural Information Processing Systems 26*, 2013, pp. 1772–1780.

[13] A. Eck and L. Soh, "Online heuristic planning for highly uncertain domains," in *2014 International Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS '14)*, Richland, US, 5–9 May 2014, pp. 741–748.

[14] S. Ross, J. Pineau, and B. Chaib-draa, "Theoretical analysis of heuristic search methods for online POMDPs," in *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. MIT Press, 2008, pp. 1233–1240.

[15] L. Buşoniu and R. Munos, "Optimistic planning for Markov decision processes," in *15th International Conf. on Artificial Intelligence and Statistics (AISTATS-12)*, La Palma, Canary Islands, Spain, 21–23 Apr 2012, pp. 182–189.

[16] M. Hauskrecht, "Value-function approximations for partially observable Markov decision processes," *Journal of Artificial Intelligence Research*, vol. 13, pp. 33–94, 2000.

[17] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Robotics: Science and Systems IV*, Zurich, Switzerland, 25–28 Jun 2008.

[18] R. Munos, "The optimistic principle applied to games, optimization and planning: Towards foundations of Monte-Carlo tree search," *Foundations and Trends in Machine Learning*, vol. 7, no. 1, pp. 1–130, 2014.

[19] Z. Zhang, M. L. Littman, and X. Chen, "Covering number as a complexity measure for POMDP planning and learning," in *26th AAAI Conf. on Artificial Intelligence*, Toronto, Canada, 22–26 Jul 2012.

[20] Z. Zhang, D. Hsu, and W. S. Lee, "Covering number for efficient heuristic-based POMDP planning," in *31th International Conf. on Machine Learning, ICML-14*, Beijing, China, 21–26 Jun 2014, pp. 28–36.