Solving Continuous POMDPs: Value Iteration with Incremental Learning of an Efficient Space Representation

Sebastian Brechtel, Tobias Gindele, Rüdiger Dillmann BRECHTEL,GINDELE,DILLMANN@KIT.EDU
Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany

Abstract

Discrete POMDPs of medium complexity can be approximately solved in reasonable time. However, most applications have a continuous and thus uncountably infinite state space. We propose the novel concept of learning a discrete representation of the continuous state space to solve the integrals in continuous POMDPs efficiently and generalize sparse calculations over the continuous space. The representation is iteratively refined as part of a novel Value Iteration step and does not depend on prior knowledge. Consistency for the learned generalization is asserted by a self-correction algorithm. The presented concept is implemented for continuous state and observation spaces based on Monte Carlo approximation to allow for arbitrary POMDP models. In an experimental comparison it yields higher values in significantly shorter time than state of the art algorithms and solves higher-dimensional problems.

1. Introduction and Related Work

Solving decision problems in the real world is aggravated by incomplete and noisy perception and uncertain knowledge about how the world behaves over time. Partially observable Markov decision processes (POMDPs) (Sondik, 1971; Kaelbling et al., 1998) provide a principled and powerful framework to model sequential decision problems under these constraints. Solving POMDPs exactly is computationally intractable. Recent developments in approximate solutions made discrete POMDPs applicable for many tasks. Point Based Value Iteration (PBVI) approaches are efficient because they only explore a sampled subset of beliefs (Pineau et al., 2003). In combination with exploration and pruning heuristics, higher-dimensional discrete POMDPs can be efficiently approximated (Spaan & Vlassis, 2005; Hoey & Poupart, 2005; Kurniawati et al., 2008; Poupart et al., 2011).

Most real world problems are of continuous nature, though. In this case, the belief space is not only high- but infinite-dimensional and known approaches for discrete state domains are not applicable. To solve POMDPs expected values need to be calculated. In continuous POMDPs these are defined by integrals, for which in general no closed form exists. For some problems it is possible to find a suited symbolic representation by hand, but this procedure is time-consuming and error-prone. In general, a good representation is not known a priori. To reduce the dimension of the belief space automatically, more efficient belief representations were proposed. In (Roy et al., 2005; Brooks et al., 2006; van den Berg et al., 2012) sufficient statistics, like the family of Gaussian distributions, were used. These are not able to represent multiple modes or sharp edges in the belief distribution. Both abilities are essential, e.g., for path planning: A robot should be able to pass a gap between the two modes of the distribution of an obstacle's position. Also, collision borders need to be clearly represented. By reducing the Kullback-Leibler divergence between the belief state and its compression, more general representations can be found (Zhou et al., 2010). Under certain conditions beneficial properties, e.g. that the value function is piecewise linear and convex (PWLC), can be maintained and exploited (Porta et al., 2006). In this case α-functions, which are defined on the state space, can efficiently represent the value function, which is defined on the belief space.

These approaches share the idea to represent beliefs rather than policies as precisely and efficiently as possible. However, it is not necessary to represent every detail, if it has no influence on the optimal policy. Research in MDPs showed that optimal state abstraction is more than just an efficient description of the state

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

(Munos & Moore, 2002). The same is true for belief representations in POMDPs. According to (Poupart & Boutilier, 2002) a representation of the state space of a POMDP can be called lossless, if it preserves enough information to select the optimal policy. In discrete POMDPs, states with same value can be aggregated without influencing the policy (Feng & Hansen, 2004). This leads to a problem specific compression of the belief space that is refined during the solving process. Unfortunately, these ideas cannot be applied to infinite state spaces directly. We show how they can help to solve continuous-state POMDPs.

2. Concept and Implementation

This paper presents a method to solve continuous-state POMDPs efficiently without constraining the reward, process or observation models. Inspired by (Feng & Hansen, 2004), we utilize the fact that the value function comprises all problem relevant information about interrelations of states, rewards and observations: States need to be distinguished iff that distinction leads to a different decision. Using α-functions to represent the value function, these states have different α-values. For continuous spaces we combine this idea with a novel heuristic: In most problems, states which are close in the state space will usually lead to similar outcomes and thus have similar α-values. Hence, most results can be generalized over the state space without loss in the resulting policy. E.g., in path planning, collisions induce discontinuities in the α-function: For decision making a region in the state space with low values, because these states will end in a collision in the future, must be differentiated from the high value region, where the obstacle can be avoided. Actually, in path planning a robots precise position is mostly not important until it approaches, e.g., a narrow passage. Value functions even underly a hidden pattern in the state space that is more complex than just proximity.

Our approach exploits these insights by automatically learning a low-dimensional, discrete representation of the continuous space during the process of solving. The resulting representation is not only efficient because it is refined in accordance to the problem's needs during the solving process. Moreover, we create it by applying machine learning algorithms. This enables the solver to reveal the hidden pattern of a POMDP and encode it in one space representation shared by all beliefs and α-functions. This way, results of the value calculation can be generalized even to regions which have not been explored, yet.

The main contribution of this paper is the incorporation of this learning process into a Monte Carlo (MC) Value Iteration Bellman backup. The results of the continuous backup (Sec. 2.2) are α-functions represented by state-samples and their according value, from which the learning procedure (Sec. 2.3) extracts an efficient discrete α-representation (examples are shown in Fig. 1). This procedure allows to back-propagate representation refinements over time together with the value. We show that the representation implicitly defines a low-dimensional discrete POMDP and that the solving process thus converges under certain constraints.

The second contribution of this paper is a general concept to maintain consistency for the value representation when using sparse backups (see Sec. 2.4). Due to the nature of inductive learning, value results are over-generalized. The reason for this is the absence of better knowledge because an MC backup cannot explore every detail of the space. We formulate a criterion to reveal conflicting generalizations during the process of solving and an algorithm to resolve them.

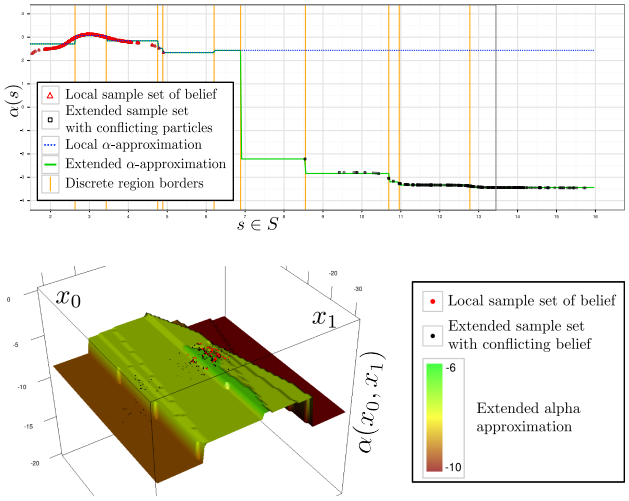

Fig. 1. 1D and 2D α-generalization and conflict resolution.

We embed the concept of value-directed representation learning and conflict correction into a continuous PBVI algorithm. We present the theory to embed discrete representations with a finite number of arbitrary basis functions into continuous Value Iteration. In the implementation we choose a disjunct space representation with piecewise constant beliefs and α-functions. As we use decision trees to represent the space partitioning, this choice is computationally efficient and can be refined by splitting nodes. The implementation uses exploration heuristics similar to 'explore' described in (Smith & Simmons, 2004) for discrete POMDPs to select which belief-point to backup next. Therefore approximate upper and lower bounds of the value must be maintained for every belief.

2.1. Preliminaries on Continuous POMDPs

A POMDP models a system with only partially observable states $s \in S$. The process model $p(s'|s,a) = p(s_{t+1}|s_t,a)$ is stochastic and depends on the taken action $a \in A$. The knowledge about the system's state is represented by the belief distribution $b \in \mathcal{B}$ with $b : S \to \mathbb{R}_{\geq 0}$ and $\int_{s\in S} b(s)\mathrm{d}s = 1$. It can be updated after observing $o \in O$ using the observation model $p(o|s') = p(o|s_{t+1})$ analogously to Bayesian filtering. In every time step the system receives a real-valued reward $r : S \times A \to \mathbb{R}$. The goal of a POMDP solver is to find a policy $\pi : \mathcal{B} \to A$ which maximizes the value $V : \mathcal{B} \to \mathbb{R}$ for an initial belief b_0. The value is defined as the expected total sum of rewards for a belief b_0 over time t discounted by $\gamma \in [0,1)$:

$$V_\pi(b_0) = E\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(b_t))\right]. \tag{1}$$

Value Iteration is widely used to solve discrete POMDPs. It is based on the idea of dynamic programming (Bellman, 1957). Bellman backups propagate the value function back in time and their recursive application finally leads to convergence in the optimal value. (Porta et al., 2006) formalized it for continuous problems. Analogously to the discrete case proven in (Sondik, 1971), they showed that the optimal continuous-state value function is PWLC for discrete observations and actions. Hence, the optimal n-th step value function can be described as the supremum of a finite set Γ^n of linear α-functions:

$$V^n(b) = \sup_{\alpha\in\Gamma^n} \langle \alpha, b\rangle = \sup_{\alpha\in\Gamma^n} \int_{s\in S} b(s)\alpha(s)\mathrm{d}s, \tag{2}$$

with $S \subseteq \mathbb{R}^N$ and $\alpha : S \to \mathbb{R}$. In the discrete case the α-function reduces to an α-vector and the expectation operator $\langle\cdot,\cdot\rangle$ to a dot-product. As the expectation operator is linear, α-function backup can be performed:

$$\alpha_a^n(s) = r(s,a) + \gamma \int_{s'\in S} p(s'|s,a) \int_{o\in O} p(o|s')\alpha_{a,o}^{n-1}(s')\mathrm{d}o\ \mathrm{d}s', \tag{3}$$

with $\alpha_{a,o}^{n-1} \in \Gamma^{n-1}$, $\alpha_{a,o}^{n-1} = \arg\sup_\alpha \langle \alpha, b'_{a,o}\rangle$. Only α-functions dominating the next belief $b'_{a,o}$, reached by choosing action a in belief b and observing o, are used:

$$b'_{a,o}(s') \sim p(o|s') \int_{s\in S} p(s'|s,a)b(s)\mathrm{d}s\ . \tag{4}$$

Repeatedly applying Eq. (3) ensures convergence of the continuous Value Iteration, if all equations are well-defined (Porta et al., 2006). In discrete POMDPs this is unproblematic as these equations are defined by sums over a finite space. The reason why continuous POMDPs are so much harder to solve is the necessity to compute continuous integrals over the state space. A closed form for Eq. (3) can only be found for some special problems: E.g., if all models in the POMDP are a linear combination of Gaussians, the α-functions can also be represented by linear combinations of Gaussians (Porta et al., 2006). Unfortunately, the number of components grows exponentially with every Value Iteration step. Thus, even if a closed form exists, for the sake of computational feasibility approximations are necessary. Notice that approximations can prevent Value Iteration from converging to the optimal result.

2.2. Monte Carlo POMDP Value Backup

In general, there is no closed form for the integrals in the continuous-state value backup in Eq. (3) and the underlying Bayesian filtering problem in Eq. (4). Sequential Monte Carlo (SMC) methods (also known as particle filters) can be applied to numerically approximate these equations. Their idea is to use a finite set of samples to evaluate the continuous integrals. SMC methods are a well-known technique approximating arbitrary Bayesian filtering problems (Doucet et al., 2001). In contrast to, e.g., Kalman filtering, SMC methods are not limited to certain distributions (MacKay, 2003). Their capabilities come at the price of higher computational effort and suboptimal solutions, especially when dealing with very noisy density functions. For highly non-linear and multi-modal problems though, the advantages of SMC outweigh. An example for the need of complex, non-linear models in decision making is given in (Gindele et al., 2010; Brechtel et al., 2011) for the domain of road traffic.

SMC methods have been incorporated into POMDP Value Iteration in (Bai et al., 2010; Porta et al., 2006) among others. In contrast to those approaches, we heavily reuse once sampled particles by reweighting them, in order to avoid the potentially high computational effort of sampling from and evaluating the conditional density functions. Additionally, we want to minimize the expensive discretization of continuous beliefs and particles, which is necessary to look up values in the current bound approximations. Ideally, due to the monotonicity of value backups, an upper and lower bound of the value can be maintained, both converging to the same optimal value. The lower bound defines the policy result and is PWLC. Hence, the α-backup in Eq. (10) can be used. The upper bound is used to estimate the solver's error and guide the exploration. As it is not convex, only the belief backup in Eq. (6) can be applied. Our implementation explores the belief space analogously to (Smith & Simmons, 2004). If a belief has been chosen, the MC value backup is run for all actions a. The set Q of samples

$q \in Q$ is drawn proportional to the proposal joint density $p(s, s', o|b, a)$. Our MC backup uses it as basis for the backup operations in the belief b for a fixed action a. Q is recombined for different purposes.

$$\langle s^q, s'^q, o^q \rangle \sim p(s, s', o|b, a) = p(o|s')p(s'|s, a)b(s) \quad (5)$$

The samples are obtained by drawing s^q from b, then drawing s'^q from $p(s'|s^q)$ and finally drawing o^q from $p(o|s'^q)$. Particles of a backup which are close in the state space are predicted and observed similarly. Thus, recombining them is computationally beneficial, especially because a POMDP solver accounts for every possible observation, not just the one encountered, like in Bayesian Filtering.

Belief Value Backup The sample-tuples in Eq. (5) are first drawn proportional to b and thus the resulting observation samples o^q can be directly used to calculate the next beliefs b'_{a,o^q} according to classical SMC methods. To minimize redundant computations, a set of distinct observations $o \in O_{b,a}$ is used. The observation probability $p(o|b, a)$ equals the probability of getting to the next belief $b'_{a,o}$ and can be extracted by marginalizing the samples over s and s'. The belief $b'_{a,o}$ in Eq. (4) is approximated by the samples s'^q weighted with $w \sim p(o|s'^q)$. The n-th value backup for the belief b results to:

$$V_a^n(b) \approx \left(\sum_{q \in Q} r(s^q, a) \right) + \gamma \sum_{o \in O_{b,a}} p(o|b, a) V^{n-1}(b'_{a,o}), \quad (6)$$

where $V^{n-1}(b'_{a,o})$ is obtained with the value function of the the previous iteration step $n-1$.

α-Function Backup Repeatedly applying α-function backups improves the lower bound and the policy represented by it. An α-function expresses the value an agent receives which is in state s, if he acts according to the best known policy for the belief b. For the backup in Eq. (10) thus the best α-functions for the next beliefs $b'_{a,o}$ are used:

$$\alpha_{a,o}^{n-1} := \underset{\alpha \in \Gamma^{n-1}}{\arg\sup} \left\langle \alpha, b'_{a,o} \right\rangle. \quad (7)$$

For recombining the samples Q, we apply importance sampling to compensate their bias:

$$v_{b,a}^{(k,i)} := \frac{p(s'^k|s^i, a)}{p(s'^k|s^k, a)b(s^k)} \quad (8)$$

$$w_{b,a}^{(j,k)} := \frac{p(o^j|s'^k)}{p(o^j|s'^j)p(s'^j|s^j, a)b(s^j)} \quad (9)$$

With these weights we calculate the α-function backup (Eq. (3)) at every sampled point s^i of b. Note, that α can also be evaluated for $s^i : b(s^i) = 0$. This capability is essential for the conflict resolution in Sec. 2.4. The evaluated α-samples $\alpha_{b,a}^n(s^i)$ form a sparse representation of the continuous α-function. Every backup operation extends the set $\Gamma^n = \Gamma^{n-1} \cup \alpha_{b,a}^n(s^i)$:

$$\alpha_{b,a}^n(s^i) \overset{(3)}{\approx} r(s^i, a) + \gamma \sum_{k \in Q} \sum_{o \in O_{b,a}} \sum_{\{j \in Q | o^j = o\}} w_{b,a}^{(j,k)} v_{b,a}^{(k,i)} \alpha_{a,o^j}^{n-1}(s'^k). \quad (10)$$

As we evaluate the continuous expectation operator $\langle \cdot, \cdot \rangle$ in Eq. (7) using the discrete representation, every continuous sample s'^q has to be discretized only once, even if the used α-functions change (see Sec. 2.3). Moreover, the distributions $b(s^k)$, $p(s'^k|s^i, a)$ and $p(o^j|s'^k)$ need to be evaluated only once. In both cases the results can be saved and reused, if a belief is iterated a second time. The recombination of weights is quite expensive, but a perfect candidate for massive parallelization, e.g., with GPUs because it is independent of the specific POMDP.

The procedure is the same for continuous and discrete observations. However, while for discrete observations it is likely to sample the same observation several times, it is very improbable in noisy, continuous domains. In such domains the number of different observations $|O_{b,a}|$ can get very high and we apply an idea presented in (Hoey & Poupart, 2005) for discrete POMDPs. The linearity of the backup allows to cluster all observations and with them beliefs b' that are dominated by the same α-function in Eq. (7) without affecting the result.

2.3. Value Representation and Refinement

In order to converge, continuous MC Value Iteration needs the ability to generalize the results calculated in one belief by the MC backup to other beliefs. A naive implementation of Eq. (2) does not work in continuous state spaces: The MC backup in Eq. (10) approximates the continuous α-function with a finite set of samples and thus is undefined for most $s \in S$. In (Bai et al., 2010) this problem is circumvented by using policy graphs as an implicit α-function description. However, this approach scales very poorly with long time horizons and is not applicable for continuous observation spaces.

In contrast to that, we propose to find a low-dimensional representation shared by all continuous beliefs and α-functions. The key idea is to find a representation suitable for the value rather than for the (possibly high-dimensional) beliefs as in previous approaches. Note, that this line of action imposes a lossy compression of beliefs, but lossless compression of the

policy. We employ inductive machine learning with the MC α-backup results as training data to obtain the representation. Learning aims to find the patterns generating the value functions. Thus, the discrete representation effectively abstracts from the given samples and generalizes their result.

During the solving process, a new space $S_d \subset \mathbb{N}$ is build and continuously refined. We describe its relation to the continuous space S with a conditional density function $p(s_d|s)$. The discretized belief b_d can then be obtained from b by marginalization:

$$b_d(s_d) = \int_{s\in S} p(s_d|s)b(s)\mathrm{d}s \ . \tag{11}$$

We assume that the continuous α-function can be represented by the vector $\alpha_d(s_d)$ as a finite, linear combination of normalized, but otherwise arbitrary, basis functions $p(s_d|s)$:

$$\alpha(s) = \sum_{s_d\in S_d} p(s_d|s)\alpha_d(s_d) \ . \tag{12}$$

As shown in (Porta et al., 2006), the continuous operator $\langle\cdot,\cdot\rangle$ is linear in the continuous belief space. Hence, the value function $V_a^n(b)$ in Eq. (6) is PWLC for discrete observations and actions. With above definitions Lemma 1 holds and the discrete expectation operator $\langle\cdot,\cdot\rangle_d$, a simple and computationally inexpensive dot-product, equals the continuous expectation operator:

Lemma 1. *The continuous and the discrete expectation operator are equal.*

Proof.

$$\begin{aligned}
\langle\alpha, b\rangle &\overset{(2)}{=} \int_{s\in S} \alpha(s)b(s)\mathrm{d}s \\
&\overset{(12)}{=} \int_{s\in S} \sum_{s_d\in S_d} \alpha_d(s_d)p(s_d|s)b(s)\mathrm{d}s \\
&= \sum_{s_d\in S_d} \alpha_d(s_d) \int_{s\in S} p(s_d|s)b(s)\mathrm{d}s \\
&\overset{(11)}{=} \sum_{s_d\in S_d} \alpha_d(s_d)b(s_d)\mathrm{d}s = \langle\alpha_d, b_d\rangle_d
\end{aligned}$$ □

Under these premises and assuming the MC α-backup in Eq. (10) to be exact, the α-function recursion is an isotonic contraction and converges. Further, if Lemma 1 holds, $p(s_d|s)$ defines a discrete problem dual to the continuous-state POMDP. Solving the discrete POMDP in S_d, solves the original problem.

Finding a small and thus efficient set of basis functions $p(s_d|s)$ that fulfill Eq. (12) for every α-function is a difficult task. Also, the error induced by the MC backup can only be minimized by using a a sufficient number of particles, but not prevented. The following subsections outline how the presented theory is implemented in a computationally efficient manner and discuss some details.

Representation Implementation For our implementation we choose S_d to represent a partitioning of S using a mapping $c^m : S \to S_d$. While this choice imposes limitations regarding the representation capabilities, it allows for an efficient implementation and refinement during the solving process to account for more details. Therefore m indicates the *version* of c. It is also able to represent discontinuities in the value function and resembles the idea of differentiating states only if it is relevant to solve the problem.

$$p^m(s_d|s) := \begin{cases} 1, & \text{if } c^m(s) = s_d \\ 0, & \text{else} \end{cases} \tag{13}$$

and consequently $\alpha(s) \overset{(12)}{:=} \alpha_d(c^m(s))$. We allow a representation error $|\alpha(b) - \alpha_d(b_d)| < e_{\mathrm{LB}}(b)$ for every α-function to limit the number of partitions.

The solver must always be able to evaluate α-functions created with arbitrary level of refinement. Implementation-wise, using arbitrary basis functions, it can be challenging to store $p(s_d|s)$ for all refinement versions m. We use a decision tree to implement the mapping c because it allows quick lookups and can be refined incrementally by splitting a leaf node s_d. Further, a single decision tree can hold all versions of $p(s_d|s)$ as former leaves remain in the tree as nodes. The tree needs to be traversed only once for every particle in a belief to lookup all α-functions in the lower bound and all belief-value-pairs in the upper bound (see Sec. 2.3), even if they were created with different versions m. It is sufficient to compute an overlapping belief representation $b_{d,\mathrm{overlap}}$ containing all non-zero $p^m(s_d|s)b(s) > 0$ regardless of their version m.

Refinement Implementation A continuous backup following Eq. (10) results in a sample-based representation of the continuous α-function. This backup operations often reveal new details of the policy at new places in the state space. Thus, the discrete representation must be constantly refined to describe new α-functions.

Our implementation uses piecewise constant basis functions $p(s_d|s)$ represented by a decision tree to define Eq. (12). At the beginning of the solving process there is only the root node in the decision tree, covering the whole continuous space S. Regions are split during the process of α-approximation, if the value of their contained samples differs. In that case the two subregions resulting from the split need to be distinguished to describe the policy of the newly generated

continuous α-function correctly. Otherwise no approximation error would be induced.

To build and refine the decision tree a greedy, recursive procedure similar to C4.5 (Quinlan, 1993) is used. The classical C4.5 is only formulated for discrete spaces. We extended it with the ability to create oblique splits in the continuous space, by sampling several candidates and choosing the one that maximizes information gain. The precise choice of the learning algorithm is not crucial to the concept of Value Iteration presented in this paper. As the sampled α-function values do not suffer from significant noise, overfitting can be neglected in this context.

Algorithm 1 Value Iteration with Conflict Resolution

Require: lower bound Γ_{LB}, space representation p
1: **function** VALUE ITERATION STEP(b)
2: **for all** $\alpha_{C,d} \in \Gamma_{\mathrm{LB}}$ **do**
3: **if** $V(b) < \alpha_{C,d}(b_d)$ **then** ▷ check (14)
4: BACKUP AND RESOLVE(b_C)
5: **end if**
6: **end for**
7: BACKUP AND RESOLVE(b)
8: $\mathcal{B}_{explored} \leftarrow \mathcal{B}_{explored} \cup b_C$
9: **end function**
10: **function** BACKUP AND RESOLVE(b_C)
11: $\alpha_c \leftarrow$ CONTINUOUS BACKUP($b_C, \Gamma_{\mathrm{LB}}$) ▷ (10)
12: $p \leftarrow$ REFINE REPRESENTATION(α_c, p) ▷ Sec. 2.3
13: $\alpha_d \leftarrow$ DISCRETIZE(α_c, p)
14: **for all** $b_A \in \mathcal{B}_{explored}$ **do**
15: **if** $V(b_A) < \alpha_d(b_{A,d})$ **then** ▷ check (14)
16: $b_C \leftarrow b_C \cup b_A|_{w=0}$ ▷ extend b_C particles
17: **goto** 11
18: **end if**
19: **end for**
20: $\Gamma_{\mathrm{LB}} \leftarrow \Gamma_{\mathrm{LB}} \setminus \alpha_{C,d}$ ▷ remove old $\alpha_{C,d}$ of b_C
21: $\Gamma_{\mathrm{LB}} \leftarrow \Gamma_{\mathrm{LB}} \cup \alpha_d$ ▷ add new consistent α_d
22: **end function**

Lower and Upper Bound The lower bound at backup step n is a set Γ^n_{LB} of PWLC α-functions, represented by sparse vectors $\alpha_d : S_d \to \mathbb{R}$ using the discretization $p^m(s_d|s)$ of a potentially different version m. If $p^m(s_d|s)$ defines a partitioning c^m (see Eq. (13)), α_d is non-overlapping and non-empty in S. Consequently, the discretization of version m is implicitly stored with α_d. To evaluate $\langle \alpha_d, b_d \rangle_d$ we can calculate the sparse dot-product with the overlapping representation $b_{d,\mathrm{overlap}}$ of b. Zero-entries $\alpha_d(s_d) = 0$ are undefined and do neither affect the result nor increase processing time.

For the upper bound we use the 'sawtooth'-interpolation method proposed in (Hauskrecht, 2000). The upper bound is represented by a set $(b^p_d, V_{b^p_d}) \in \Gamma^n_{\mathrm{UB}}$ of pairs of beliefs and their value. It can be evaluated for a belief b by finding the belief-value-pair (BVP) with the minimal interpolation $V_{\mathrm{UB}}(b) = \min_p V^p(b)$ in Γ^n_{UB}. Using $p^m(s_d|s)$, this can be effectively carried out in S_d. Ideally, both bounds converge to the optimal value function and thus share their optimal representation. They can benefit from each other discovering 'interesting' splits.

2.4. Correction of Value-Generalizations

It is not feasible to preserve the bound-properties in general continuous POMDP problems exactly because the MC Value Iteration backup cannot be calculated for the complete state space. If not treated, this will lead to contradictory statements from the different beliefs regarding the value and possibly corrupt the policy. Quantifying these errors is very difficult if not impossible. Instead, we give the solver the novel ability to reflect the reasons for errors and correct them. To maintain consistency for the lower and upper bound, α-functions and BVPs violating the bound property are detected. After identification, these inconsistencies are resolved by extending the sample horizon of the conflicting value approximation. This concept is universally applicable, regardless of the representation of bounds, the problem's dimensionality or nature. It can also be used for discrete POMDP solvers to realize an efficient sparse backup.

Conflict Detection Assuming an exact MC value-backup, the Value Iteration in Sec. 2.2 converges monotonically to the optimal value. Hence, it can be safely stated that the n-th step continuous belief backup $V^n(b_A)$ defined in Eq. (6), using a lower bound $\Gamma^{n-1}_{\mathrm{LB}}$ as knowledge base, must be superior or equal to any direct α-function from the same knowledge base evaluated for b_A:

$$\forall \alpha \in \Gamma^{n-1}_{\mathrm{LB}} : V^n(b_A) \geq \alpha(b_A). \tag{14}$$

If this property is violated, $\Gamma^{n-1}_{\mathrm{LB}}$ is no lower bound because a continuous α-function was overgeneralized. Consequently, the violating α_C must be corrected. Algorithm 1 shows how this concept is embedded into the lower bound Value Iteration for a newly explored belief b. In the examples in Fig. 1, the α-vector generalization in a conflicting belief b_C violated the continuous backup of the accusing belief b_A. The beliefs in those examples are obviously different. There are also POMDP problems, where beliefs close to each other in the state space can lead to significantly different values. Approaches which use a distance metric to differentiate beliefs or approximate beliefs directly with parametric models or kernel densities fail for such

problems. The conflict mechanism in our approach ensures that the representation gets refined to differentiate even such close beliefs.

For the upper bound approximation applies similar: The BVPs in the sawtooth approximation described in Sec. 2.3 must create a higher or equal value for every b than their continuous backup of the upper bound.

We assert these properties continuously by checking every new α-vector or BVP for conflicts with any of the so far explored beliefs. Additionally, we check, if a newly explored belief reveals a conflict with an existing α-function or BVP. If that is the case, we resolve the conflict as described in the next section. As both bounds are formulated sparsely in the discrete space, the conflict detection is not as time-consuming as it might appear at first glance. All beliefs are already given in a discrete representation, which must at worst be updated to the most recent version.

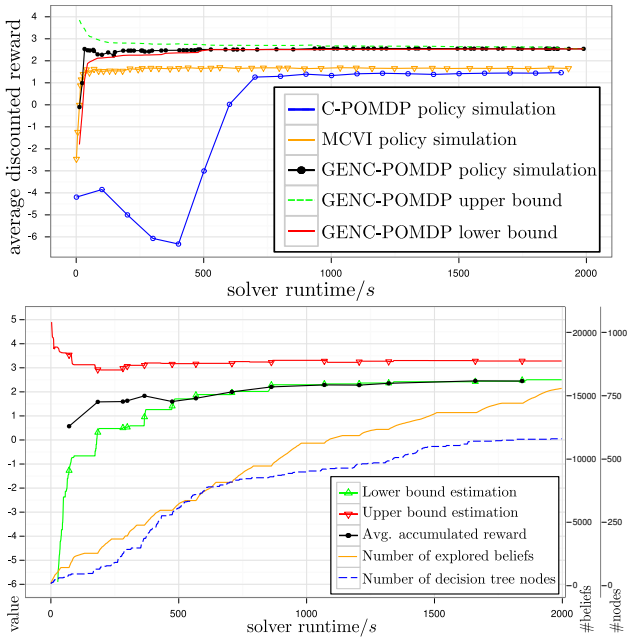

Fig. 2. Convergence comparison of state of the art solvers with the presented GENC-POMDP for the 1D corridor problem and analysis of the 2D problem.

Conflict Resolution The reason for the conflicts induced by an α-vector is that the continuous space is only incompletely covered by sampling. Assumptions about the undefined regions had to be made, which b_A falsified during conflict detection. Extending the sampling base for the approximation with samples from b_A invalidates the reason for the conflict. Therefore we compute the α-backup for every sample s_A^i of b_A acting according to the policy of b_C with Eq. (10). As can be seen in the 1D example in Fig. 1, a new approximation then automatically learns the lower value of those additional $\alpha(s)$-samples combined with the old samples.

The correction procedure for the upper bound approximations is not covered in detail. It is based on the fact that using a more refined representation for b_d^p specifies the meaning of the BVP and increases the value $V_d^p(b_d)$ for any b_d. It is possible and probably beneficial to introduce splits in this process, but the current implementation only uses splits from the lower bound.

2.5. Evaluation

The presented solver is evaluated on three problems. The first one is the continuous corridor problem described by (Porta et al., 2005). While it is only 1D it comprises an infinite amount of states and can therefore not be solved discretely. The second problem is a 2D extension of the first and shows the capability of the presented algorithm to find a low dimensional discrete representation. At last, we introduce a new higher-dimensional 8D intersection problem to analyze the solver in a more realistic scenario.

1D Corridor Problem The first problem models a robot moving in a one-dimensional corridor limited by walls. The aim of the robot is to open the right door, while perceiving its own position only uncertainly by a discrete number of Gaussians. In the upper graph of Fig. 2 the evaluation results of the policies after different solving times of the presented solver is compared to MCVI (Release 0.2, 17 May 2012) from (Bai et al., 2010) and C-POMDP (PERSEUS) from (Porta et al., 2006) using their open source implementations and parameters. Therefore we ran 10,000 trials simulating 100 consecutive time steps. It shows that previous solvers only reach suboptimal average rewards after reasonable solving time. After 12 h MCVI did not exceed an average reward of 1.8. In 16 min our algorithm found a much better policy with an average reward of about 2.55, although the smooth nature of the Gaussians is close to a worst case for the proposed piecewise-constant representation. Because of the reuse of samples in the presented MC backup and the particle extension by the conflict mechanism, 300 particles are sufficient to solve this problem. After about 33 min both bounds converged and the algorithm stopped.

2D Corridor Problem The second problem is a 2D extension of the corridor problem. The robot can additionally go up and down. The 12 observations are product densities of $\mathcal{N}(-3,2)$, $\mathcal{N}(3,2)$ and $\mathcal{N}(0,100)$ in x_1 with the 4 original 1D observations in x_0.

The reward only depends on x_0. Basically x_1 only distracts the solver, but still the combinatorial complexity of this problem is much higher due to the increased number of observations. Additionally, the backup steps are slower because 500 particles were used. The second plot in Fig. 1 shows that the splits are mainly parallel to x_1. The solver generalizes the values by recognizing the meaninglessness of x_1 and thus converges after reasonable time, as shown in the lower graph of Fig. 2. Interestingly, the value bounds are not monotonic. The upper bound often drops quickly at the beginning and corrects itself when a conflict detecting belief is explored.

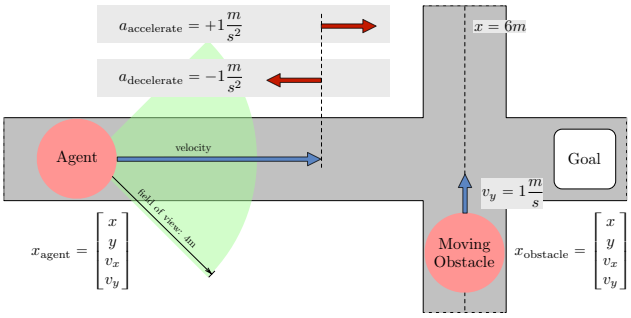

Fig. 3. Sketch of the 8D intersection problem.

8D Intersection Problem In the intersection scenario in Fig. 3 a robot agent and an obstacle move on a 2D plane following a constant velocity model with small white noise. The positions and velocities of both objects result in an 8D joint state space. The observations are continuous 8D measurements of the state with additive white noise. The objects have a circular shape with a radius of 1 m. Their positions are loosely bounded by *roads*. Starting at $(x, y, v_x, v_y) = (0\,\text{m}, 0\,\text{m}, 1\,\text{m/s}, 0\,\text{m/s})$ the agent has to cross the intersection without hitting the moving obstacle by accelerating and decelerating. The agent receives a reward of $+10$ for reaching the goal and -10 in case of a collision. The discount is $\gamma = 0.95$. If the obstacle moves past $y = 8\,\text{m}$ it is respawned at $y = -8\,\text{m}$ so that the obstacle remains a constant threat. The difficulty of the problem is that the y-position of the obstacle is not known in advance and the agent's viewing distance is limited to 4 m. The policy found by the solver after 543 s achieves an average score of 5.0 in simulation using 400 particles to represent the beliefs. The final representation of the space uses a total of 487 states. A fixed discretization of an 8D space with just two regions per dimension would already yield 256 states. The iterative refinement discovered an efficient representation. The final policy is encoded by 41 α-functions describing the following behavior: The agent slowly approaches the intersection to be able to stop in a safe distance where he can perceive the y-position of the obstacle with a small standard deviation of $\sigma_y = 0.32$ (see Fig. 4). Then it accelerates instantly or moves back first to pass the obstacle.

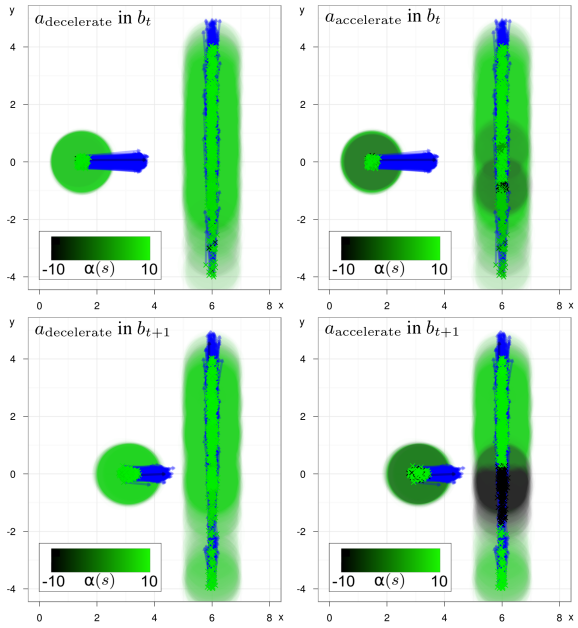

Fig. 4. 8D problem: Dominating α-functions for successive beliefs. Arrows indicate the velocity and color the α-value. Choosing $a_{\text{decelerate}}$ in b_t the agent reaches b_{t+1}.

3. Conclusion and Future Work

The idea of combining planning with the generalization capabilities of inductive learning shows very promising results that bring the application of POMDPs in real world tasks one step closer to reality. The presented algorithm finds a better solution to the continuous corridor problem in about 30 s than previous approaches can obtain in hours. The novel concept worked even in this noise dominated domain and a higher-dimensional 8D obstacle avoidance problem. The presented algorithm is particularly suited for POMDPs with sparse, multi-modal models.

In future work, a pruning of α-functions as well as beliefs analogously to discrete solvers needs to be incorporated. Non-linear splitting functions, refinement according to the upper bound and more sophisticated error heuristics to control the adaptation would further increase the efficiency of the representation. Finally, finding a better exploration heuristic for continuous belief spaces is still an unsolved problem.

Acknowledgments This research is supported by the *Deutsche Forschungsgemeinschaft*.

References

Bai, H., Hsu, D., Lee, W.S., and V.A., Ngo. Monte Carlo Value Iteration for Continuous-State POMDPs. In *Workshop on the Algorithmic Foundations of Robotics*, pp. 175–191, 2010.

Bellman, R. A Markovian decision process. *Journal of Applied Mathematics and Mechanics*, 6:679–684, 1957.

Brechtel, S., Gindele, T., and Dillmann, R. Probabilistic MDP-behavior planning for cars. In *Int. Conf. on Intelligent Transportation Systems*, pp. 1537–1542, 2011.

Brooks, A., Makarenko, A., Williams, S., and Durrant-Whyte, H. Parametric POMDPs for planning in continuous state spaces. *Robotics and Autonomous Systems*, 54(11):887–897, 2006. ISSN 0921-8890.

Doucet, A., De Freitas, N., and Gordon, N. *Sequential Monte Carlo methods in practice*. Springer, 2001.

Feng, Z. and Hansen, E. An Approach to State Aggregation for POMDPs. In *Workshop on Learning and Planning in Markov Processes*, pp. 7–12, 2004.

Gindele, T., Brechtel, S., and Dillmann, R. A Probabilistic Model for Estimating Driver Behaviors and Vehicle Trajectories in Traffic Environments. In *Int. Conf. on Intelligent Transportation Systems*, pp. 1625–1631, 2010.

Hauskrecht, M. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13(1):33–94, 2000.

Hoey, J. and Poupart, P. Solving POMDPs with continuous or large discrete observation spaces. In *Int. Joint Conf. on Artificial Intelligence*, pp. 1332, 2005.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Journal on Artificial Intelligence*, 101 (12):99 – 134, 1998. ISSN 0004-3702.

Kurniawati, H., Hsu, D., and Lee, W.S. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

MacKay, D.J.C. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

Munos, R. and Moore, A. Variable resolution discretization in optimal control. *Machine learning*, 49 (2):291–323, 2002. ISSN 0885-6125.

Pineau, J., Gordon, G., and Thrun, S. Point-based value iteration: An anytime algorithm for POMDPs. In *Int. Joint Conf. on Artificial Intelligence*, volume 18, pp. 1025–1032, 2003.

Porta, J.M., Spaan, M.T.J., and Vlassis, N. Robot planning in partially observable continuous domains. In *Robotics: Science and Systems*, volume 1, pp. 217. The MIT Press, 2005.

Porta, J.M., Vlassis, N., Spaan, M.T.J., and Poupart, P. Point-based value iteration for continuous POMDPs. *The Journal of Machine Learning Research*, 7:2329–2367, 2006. ISSN 1532-4435.

Poupart, P. and Boutilier, C. Value-directed compression of POMDPs. *Advances in Neural Information Processing Systems*, 15:1547–1554, 2002.

Poupart, P., Kim, K.E., and Kim, D. Closing the Gap: Improved Bounds on Optimal POMDP Solutions. In *Int. Conf. on Automated Planning and Scheduling*, 2011.

Quinlan, J.R. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.

Roy, N., Gordon, G.J., and Thrun, S. Finding approximate pomdp solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1–40, 2005.

Smith, T. and Simmons, R. Heuristic search value iteration for pomdps. In *Uncertainty in Artificial Intelligence*, pp. 520–527. AUAI Press, 2004.

Sondik, E.J. The Optimal Control of Partially Observable Markov Decision Processes. *PhD thesis, Stanford University*, 1971.

Spaan, M.T.J. and Vlassis, N. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24(1):195–220, 2005.

van den Berg, Jur, Patil, Sachin, and Alterovitz, Ron. Efficient Approximate Value Iteration for Continuous Gaussian POMDPs. In *Conf. on Artificial Intelligence*, 2012.

Zhou, E., Fu, M.C., and Marcus, S.I. Solving continuous-state POMDPs via density projection. *Transactions on Automatic Control*, 55(5):1101–1116, 2010.