Abstract
Planning in Partially Observable Markov Decision Processes (POMDPs) inherently gathers the information necessary to act optimally under uncertainties. The framework can be extended to model pure information gathering tasks by considering belief-based rewards. This allows us to use reward shaping to guide POMDP planning to informative beliefs by using a weighted combination of the original reward and the expected information gain as the objective. In this work we propose a novel online algorithm, Information Particle Filter Tree (IPFT), to solve problems with belief-dependent rewards on continuous domains. It simulates particle-based belief trajectories in a Monte Carlo Tree Search (MCTS) approach to construct a search tree in the belief space. The evaluation shows that the consideration of information gain greatly improves the performance in problems where information gathering is an essential part of the optimal policy.

Index Terms — Continuous POMDP, Planning, Monte Carlo Tree Search, Particle Filter Tree, IPFT, Reward Shaping, Information Gathering.

1. Introduction
Uncertainty is decisive in many decision making problems. While information retrieval is the sole objective of some problems, in others it is only a means to disambiguate the current situation and select optimal actions. These types of problem can be modeled with POMDPs. The optimal policy for many POMDPs can be found only approximately since finding the exact solution is computationally intractable (Patrascu & Tsitsiklis, 1987). In real-world applications, planning algorithms frequently have to deal with (1) problems on continuous domains and (2) a limited time available for online planning.

Online algorithms for large or continuous state spaces are derived from MCTS, the most prominent ones being POMCP (Silver & Veness, 2010), DESPOT (Somani et al., 2013), and ABT (Kurniawati & Yadav, 2016). Besides having continuous state spaces, many problems in robotics are modeled with continuous observation spaces. The further development of the mentioned algorithms resulted in the state-of-the-art solvers POMCPOW and PFT-DPW (Sunberg & Kochenderfer, 2018) and DESPOT-α (Garg et al., 2019), which can deal with continuous state and observation spaces and therefore solve the first issue.

While MCTS-based algorithms are capable of solving problems online, they can suffer from suboptimal behavior when limited planning time is available. This problem arises when rewards are sparse, e.g. only indicate success or failure at the end of each episode. Hence, the planning algorithm is required to explore many suboptimal paths until finding promising actions, especially in problems with high uncertainty. To resolve this issue, Potential-Based Reward Shaping (PBRS) can be used to implicitly guide the agent to large future rewards (Eck et al., 2016). The potentials are functions over beliefs and reflect how much reward can potentially be gained in a certain situation. This additional guidance considerably speeds up planning and thus addresses the second issue.

In this paper, we propose to combine PBRS with MCTS for solving POMDPs on continuous spaces. This tackles the two issues mentioned for planning in real-world problems. Previous research has only considered reward shaping for problems on discrete spaces. We use potentials based on information measures to guide the agent to informative beliefs. Since information gathering is typically part of the optimal policy, this significantly helps to solve the problem. The resulting belief-based reward functions can be modeled within the ρPOMDP framework, for which solution methods only exist for discrete problems (Araya-López et al., 2010). For this reason, we develop a novel online algorithm, Information Particle Filter Tree (IPFT), for solving ρPOMDPs.
on continuous spaces, which enables us to use PBRS. We approximate beliefs with small particle sets and solve an MDP on the belief space with MCTS.

2. Background and Notation

A sequential decision making problem with uncertain system dynamics can be framed as a Markov Decision Process (MDP), represented by the tuple \((S, A, T, R, \gamma)\), where \(S\) is the state space, \(A\) is the action space, \(T\) is the transition model that defines the distribution of the successor state \(s'\), given the current state \(s\) and the chosen action \(a\). In a continuous state space this amounts to a probability density. The reward model \(R\) represents the reward obtained when executing an action \(a\) in state \(s\), and \(\gamma \in [0, 1)\) is a factor discounting future rewards, which ensures a finite cumulative reward.

In some problems, the system state cannot be observed directly. A Partially Observable Markov Decision Process (POMDP) addresses this issue by extending the MDP model with an observation space \(O\) and an observation model \(Z\) that models the conditional distribution of receiving an observation \(o\), given state \(s\). In a continuous observation space this distribution can be represented by a density. The probability distribution over the current state \(s_t\) given the history \(h_t = (b_0, a_0, o_1, \ldots, a_{t-1}, o_t)\) of the initial state distribution \(b_0\) and all previous actions and observations is called the belief \(b_t = p(s_t|h_t)\). It can be computed from the transition and observations models using Bayes’ theorem. The belief is the basis for decision making in each step, since the true state is unknown. Therefore, policies in POMDPs map beliefs to actions. Furthermore, a POMDP can be equivalently reformulated as an MDP over the belief space, the so-called belief MDP.

An online algorithm capable of solving large MDPs is obtained by combining tree search with Monte Carlo methods. Monte Carlo Tree Search (MCTS) incrementally builds up an asymmetric tree by running simulations (Browne et al., 2012). Each tree node represents a state, for which the expected return is estimated, and edges represent transitions to successor states. One simulation consists of the phases: tree traversal, node expansion, rollout, and backpropagation. In each simulation a new node is added to the tree and the values of all visited nodes are refined. During the first phase, a popular action selection strategy to balance exploration and exploitation is the Upper Confidence Bound (UCB) (Auer et al., 2002; Kocsis & Szepesvári, 2006).

Silver & Veness developed the online algorithm Partially Observable Monte Carlo Planning (POMCP) by adapting MCTS to partially observable environments (2010). In contrast to MCTS, the tree now branches in both actions and observations. Instead of states, the nodes represent histories \(h\), that is, action-observation sequences. Additionally, the simulations building up the tree are also used to obtain a Monte Carlo belief update in between planning steps. The algorithm only requires a generative model and achieves high performance on large problems with discrete actions and observations.

MCTS as described above cannot be applied to problems with large or continuous action spaces. Since the same action is rarely selected twice, this results in a shallow tree where the values cannot be estimated well. Progressive widening is a strategy to tackle this issue (Couëtoux et al., 2011). It limits the number of child nodes considered in a node to \([kN^\alpha]\) in the \(N\)th visit, where \(k > 0\) and \(\alpha \in (0, 1)\) are parameters. If this limit is reached, one of the previous child nodes is selected. Additionally, this strategy can be applied to the branching induced by stochastic transitions or partial observability and is then referred to as Double Progressive Widening (DPW). Sunberg & Kochenderfer applied DPW in their algorithms Particle Filter Tree (PFT-DPW) and POMCP with Observation Widening (POMCPOW) for solving POMDPs on continuous spaces (2018). POMCPOW simulates single particles and iteratively refines a particle-based belief approximation in every tree node. In contrast, PFT-DPW solves the belief MDP by simulating whole beliefs instead of single particles. A belief is approximated by a fixed number of \(m\) weighted particles which is updated as in a particle filter. The immediate reward in each step is computed as the weighted mean of the particle rewards.

Using PBRS requires to plan with belief-based reward functions \(\rho(b, a)\) instead of the state-based reward \(R(s, a)\). An example which uses a belief-based reward is the belief MDP, where the reward is the expected state-based reward \(\rho(b, a) = \int_{S} R(s, a)b(s)\,ds\). Araya-López et al. proposed \(\rho\)POMDPs as an extension to POMDPs which use arbitrary belief-based rewards (2010). This allows to define pure information gathering problems as well as the utilization of reward functions combining information gain with state-based rewards, as necessary in the domain of active sensing. Analogously to the belief MDP, a \(\rho\)POMDP can be described by an MDP over the belief space with reward \(\rho\). Offline solution methods based on value iteration were proposed to solve \(\rho\)POMDPs, but they require discrete state, action, and observation spaces and a reward function that is piecewise linear and convex (PWLC) on the belief space.

3. Information-Theoretic Reward Shaping

Shaping the reward function can result in a different policy. However, policy invariant reward shaping can be achieved by PBRS (Ng et al., 1999; Eck et al., 2016). In this approach, potentials are used as heuristics for the future reward that can be expected in a belief. The reward is shaped by adding
the potential difference of successive beliefs which acts as an implicit guidance for the agent. In our work, we use potentials based on information measures for multiple reasons.

Firstly, in highly uncertain domains the agent has to reduce the uncertainty in the belief before maximizing the reward, even if information gathering as such is not rewarded. This gives rise to the idea that planning can be guided by shaping the reward function to consider the value of information.

Secondly, information retrieval can be rewarded on any domain and does not require a manually designed potential function.

The last motivation for using information-theoretic reward shaping is that the optimal value function \( V^* \) serves as a particularly effective potential (Ng et al., 1999). Although \( V^* \) is unknown, it is known to be convex over the belief space and typically attains higher values on the boundary of the belief space, since more information allows to make better decisions. Information measures, as defined in the next paragraph, are also convex and should attain their maximal values on the boundary of the belief space, which makes them a reasonable heuristic for \( V^* \).

To quantify the information contained in a distribution we define information measures as convex functions on the belief space, denoted by \( I \). The convexity captures the intuitive observation that the information contained in a mixture distribution \( I(\lambda b_1 + (1-\lambda)b_2) \) of beliefs \( b_1, b_2 \) with \( \lambda \in (0, 1) \) cannot exceed the weighted mean information \( \lambda I(b_1) + (1-\lambda)I(b_2) \) of the distributions \( b_1, b_2 \).

Based on the previous considerations, we investigate and compare two slightly different potential-based shaping functions in this work. To begin with, we use the discounted information gain \( \Delta I_\gamma(b, b') = \gamma I(b') - I(b) \) because it guarantees the optimal policy to be invariant under PBRS for infinite horizon problems (Eck et al., 2016). We additionally consider the undiscounted information gain \( \Delta I(b, b') = I(b') - I(b) \) since it resembles the idea of gathering information more intuitively and it is used in the area of active sensing (Mihaylova et al., 2003). The shaped reward function can be written as

\[
\rho(b, a, b') = \int_S R(s, a) b(s) \, ds + \lambda \Delta I(b, b')
\]

(1)

where the parameter \( \lambda \) weights reward maximization and information gathering.

Various information measures are presented in the literature. A widely used uncertainty measure for discrete probability distributions is the entropy \( H = -\sum p(x) \log p(x) \). It can be generalized to continuous distributions by integrating over the continuous domain \( H = -\int p(x) \log p(x) \, dx \) and is then called differential entropy. While this loses some properties (e.g. positivity), it still serves well as an uncertainty measure.

Other information measures we considered are based on \( L_p \)-norms or the distance to the maximum entropy distribution, which represents maximal uncertainty. Similar ideas for discrete spaces are presented in (Mihaylova et al., 2003; Araya-López et al., 2010). Since the results of our evaluations were similar for all information measures considered, we only present the results for negative entropy. The variance of a belief, however, is unsuitable for measuring information. This is because a bimodal distribution with extremely narrow peaks at an arbitrarily large distance contains a lot of information but has unbounded variance.

4. Solving \( \rho \)POMDPs on Continuous Spaces

The information-based reward shaping described in the previous section can be implemented by modeling the problem as a \( \rho \)POMDP with the belief-based reward function in Equation (1). Araya-López et al. introduced an offline algorithm for solving \( \rho \)POMDPs on discrete spaces with PWLC reward function. Since we consider reward shaping for continuous problems and the negative entropy is not PWLC, we develop a novel \( \rho \)POMDP solver for continuous domains.

Our approach adapts the PFT-DPW algorithm presented in Section 2 to belief-based reward functions. Before we present our algorithm, we show how belief-based reward functions can be evaluated on particle-based belief approximations, which are used by PFT-DPW.

4.1. Particle-Based Calculation of Belief-Dependent Rewards

While the expected reward \( \int_S R(s, a) b(s) \, ds \) and the negative entropy on discrete spaces can be easily evaluated for a sample of the belief, there is no straightforward way to compute the differential entropy from a particle set without making additional assumptions. In this section, we derive a method to compute the negative entropy from weighted particle sets based on kernel density estimation (Gisbert, 2003).

In the following, we assume a belief \( b \) is approximated by a weighted particle set \( \{(s_i, w_i)\}_{i=1}^m \) with normalized weights. As common for particle filters, the integral is approximated as

\[
-\mathcal{H}(b) = \int_S b(s) \log b(s) \, ds \approx \sum_{i=1}^m w_i \log b(s_i).
\]

(2)

Following the insights of Hall & Morton for unweighted particles (1993), the belief \( b \) can be approximated by a KDE \( \hat{b} \) computed from the weighted particle set, which yields the
entropy estimate

\[-\hat{H}(\hat{b}) = \sum_{i=1}^{m} w_i \log \hat{b}(s_i).\]  

(3)

In our work we use a Gaussian kernel and select the bandwidth according to Silverman’s rule of thumb (Silverman, 1986). The computational complexity of evaluating Equation (3) is \(O(m^2D)\) where \(D\) is the dimension of the continuous state space. Details on KDE, bandwidth selection and computational complexity can be found in the supplemental material.

An alternative method to calculate the entropy of a running particle filter can be derived based on Bayes’ theorem (Skoglar et al., 2009; Boers et al., 2010). However, this procedure requires explicit knowledge of transition and observation models, which is often not available.

4.2. The Information Particle Filter Tree Algorithm

Similar to PFT-DPW, our algorithm simulates particle sets in a MCTS fashion to construct a search tree. One iteration of IPFT is shown in Figure 1. For simplicity only belief nodes are shown, not action nodes. Particle sets of fixed size \(m\) are simulated through the tree, until reaching a node with unexplored child nodes. Then a new node is added to the tree and its value is estimated using a rollout policy. All visited nodes are updated with the returned reward.

The principal change necessary in PFT-DPW to solve \(\rho\)POMDPs is the reward computation. Instead of the mean particle reward, the belief-dependent reward model \(\rho(b, a, b')\) is used. Information rewards are calculated using the particle-based approximations derived in the previous section.

Another aspect concerns the particle-based belief approximations. In PFT-DPW particle sets are only generated when a new node is added to the tree. The particles are then saved in the node and reused when it is visited again. This is problematic because a small particle set serves only as a coarse approximation of the continuous belief. Hence, belief-based rewards like the entropy cannot be estimated well from only a small sample. For this reason, in IPFT we simulate new particle beliefs instead of reusing previous samples. The algorithm averages the rewards of different particle approximations of the same belief to obtain a better estimate.

4.2.1. Convergence of Belief-Based Rewards

The belief-based reward in Equation (1) consists of the mean state-based reward and the entropy difference. Regarding the mean particle reward, averaging over multiple particle sets yields the same result as computing the mean reward of the union of all particle sets. However, this might not be true for the entropy-based reward.

In the following, we analyze how averaging entropy estimates from \(K\) different particle sets \(S_k = \{(s_i^{(k)}, w_i^{(k)})\}_{i=1}^{m}\) differs from computing the entropy from the combined sample \(S = \bigcup S_k\). By concatenating the \(K\) particle sets, the union can be formulated as \(S = \{(s_j, \hat{w}_j)\}_{j=1}^{Km}\) where the weights \(\hat{w}_j\) are normalized with the factor \(\frac{1}{K}\). Let \(b_k\) and \(\hat{b}\) denote kernel density estimates for the sample \(S_k\) and the combined sample \(S\), respectively. Averaging the negative entropy estimates from the different particle sets results in

\[-\frac{1}{K} \sum_{k=1}^{K} \hat{H}(\hat{b}_k) = \sum_{k=1}^{K} \sum_{i=1}^{m} \frac{w_i^{(k)}}{K} \log \hat{b}_k(s_i^{(k)}).\]  

(4)

Since \(\hat{b}\) and \(b_k\) are density estimates of the same belief \(b\), they converge to the same density for \(m \to \infty\). For this reason, we assume them to be sufficiently close so that Equation (4) can be approximated by

\[\approx \sum_{k=1}^{K} \sum_{i=1}^{m} \frac{w_i^{(k)}}{K} \log \hat{b}(s_i^{(k)})\]

\[= \sum_{j=1}^{Km} \hat{w}_j \log \hat{b}(s_j) = -\hat{H}(\hat{b}).\]  

(5)

Hence, averaging the entropy estimates of different particle sets is approximately equal to estimating the entropy from the combined particle set. Since the KDE-based entropy estimate (5) converges for \(K \to \infty\), we expect our estimate to converge as well (Hall & Morton, 1993).

Estimating the entropy from the combined sample would require to maintain all particles sampled in any tree node. In contrast, our approach only needs the currently simulated particle set, which drastically reduces the required memory. Moreover, it is computationally more efficient, because the entropy computation as described in Section 4.1 scales quadratically with the number of particles \(m\).

4.2.2. Description of the Algorithm

Algorithm 1 presents the resulting method. The outer structure of the algorithm is left out due to space constraints. It closely follows the outer structure of the algorithms provided by Sunberg & Kochenderfer (2017): The SIMULATE procedure is called from the initial belief \(b\), an empty action-observation history \(h\), and the maximal tree depth \(d\) as long as computational resources permit, which over time builds up the search tree. Since beliefs are represented by particle sets, the initial belief is sampled at each call of SIMULATE in the root node. In the following, a history \(h\) appended by a new action \(a\) or an action-observation pair \((a, o)\) is denoted \(ha\) or \(hao\), respectively.
Figure 1. One iteration of the IPFT algorithm. Particle sets are simulated through the tree using UCB. At the end a node is expanded and its value estimated with a rollout policy. Then the value is propagated back through all visited nodes.

The first step in SIMULATE is to choose an action. Progressive widening of the action space and action selection according to UCB is covered by the function ACTION_PROG_WIDEN. Then either a new observation is generated or an existing observation child node is sampled, depending on the observation widening parameters, similar as in POMCPW. Here, \( C(\text{ha}) \) denotes the set of child nodes of the tree node corresponding to history \( \text{ha} \), which are identified by the observation \( o \) that leads to the corresponding child node. To generate a new observation, a state is sampled from the belief \( b \) at first, for which a successor state and the observation are generated. Note, that the counter \( M \) is only needed in the case of discrete observations, when the same observation can be sampled multiple times. In continuous observations spaces, \( M \) will almost surely be one for all nodes. Then the posterior belief \( b' \) is computed with a particle filter update \( \mathcal{G}_{PF}(m) \) and the belief-dependent reward is calculated. If a new observation was sampled, it is added as a child node and a rollout is performed to estimate the total reward \( R \) of the belief node. Otherwise, SIMULATE is called recursively on the belief \( b' \) and the successor node \( \text{hao} \). At last, the node statistics are updated.

The computational complexity of the particle filter update is \( O(m) \). For each particle set sampled from the initial belief in the root node, SIMULATE is called in every node while traversing the tree. Hence, the complexity of IPFT is \( O(nmdm) \) for \( n \) particle sets simulated in a tree of depth \( d \). Further, IPFT is well suited for parallelization since the particle filter updates can be distributed on different workers which allows to use larger particle sets. In our experiments small particles sets of size \( m = 20 \) were sufficient.

Algorithm 1

**SIMULATE function of Information Particle Filter Tree**

1: **Input:** belief \( b \), history \( h \), depth \( d \)
2: if \( d = 0 \) then
3:   \( R = 0 \)
4: else
5:   \( a \leftarrow \text{ACTION_PROG_WIDEN}(h) \)
6:   if \( |C(\text{ha})| \leq k_o N(ha)^{p_o} \) then
7:     \( o \leftarrow \text{sample} \ s \ \text{from} \ b \), generate \( o \) from \( (s, a) \)
8:     \( M(ha) \leftarrow M(ha) + 1 \)
9: else
10:    \( o \leftarrow \text{select} \ a \in C(\text{ha}) \) w.p. \( \frac{M(ha)}{\sum_a M(ha)} \)
11: end if
12: \( b' \leftarrow \mathcal{G}_{PF}(m)(b, a, o) \)
13: \( r \leftarrow \rho(h, a, b') \)
14: if \( o \notin C(\text{ha}) \) then
15:   \( C(\text{ha}) \leftarrow C(\text{ha}) \cup \{o\} \)
16:   \( R \leftarrow r + \gamma \cdot \text{ROLLOUT}(b', \text{hao}, d - 1) \)
17: end if
18: \( R \leftarrow r + \gamma \cdot \text{SIMULATE}(b', \text{hao}, d - 1) \)
19: end if
20: \( N(h) \leftarrow N(h) + 1 \)
21: \( N(\text{ha}) \leftarrow N(\text{ha}) + 1 \)
22: \( Q(\text{ha}) \leftarrow Q(\text{ha}) + \frac{R - Q(\text{ha})}{N(\text{ha})} \)
23: end if
24: **Output:** accumulated discounted reward \( R \)

4.2.3. CONVERGENCE OF IPFT

In the following, we want to provide an intuition for the convergence of IPFT. The convergence results for Partially Observable Weighted Sparse Sampling (POWSS) suggest that the particle weighting scheme used in POMCPW is sound (Lim et al., 2020). Since IPFT samples new weighted particles in every simulation rather than reusing a previous sample, it is closely related to POMCPW in this aspect. For this reason, IPFT can also be expected to converge, provided the belief-based rewards are estimated well, which we showed previously in Section 4.2.1 for entropy estimates.

Furthermore, the optimal policy for the infinite horizon problem is invariant under information-theoretic reward shaping with the discounted information gain (Eck et al., 2016). Hence, the policy produced by IPFT is also optimal for the original infinite horizon problem.

5. Evaluation

We evaluate the effects of information-based reward shaping by comparing the proposed IPFT algorithm with state-of-the-art solvers on benchmark problems of varying difficulty using the POMDPs.jl framework (Egorov et al., 2017). Furthermore, we inspect the sensitivity with respect to its hy-
5.1. Benchmark Problems

For every problem we perform 1000 simulations with each solver and average their reward. In the beginning of each simulation the problem is initialized randomly and actions are selected according to the solver, with a maximum computation time of 1 second per step. After an action is executed the agent’s belief is updated with the received observation and the next action is selected until the simulation terminates. The hyperparameters used by the solvers are provided in the supplemental material. In all problems the discount was set to $\gamma = 0.95$. We performed the simulations on an Intel Core i7-6700 CPU with a clock rate of 3.4GHz and 8GB RAM.

5.1.1. LIGHT DARK

Different variations of the Light Dark problem are used in the literature on planning under uncertainty (Perez et al., 2012; Platt, 2013; Sunberg & Kochenderfer, 2018). In our work, we consider the variant used by Sunberg & Kochenderfer, where the agent moves deterministically along a one-dimensional discrete state space.

In this problem, the goal is to reach the origin and generate action 0 to obtain a reward of 100. If action 0 is used anywhere else, the agent receives a reward of $-100$. For moving, the agent has step costs of $-1$. The problem terminates when action 0 is executed. The observations received by the agent are its current state disturbed by an additive Gaussian noise with the standard deviation $\sigma_x(s) = |s - 10|$ grows linearly with the distance from a light source at $s = 10$. Figure 2 shows the probability density of the observation model $Z$ for some states $s$. In addition to the original action space $A_{10} = \{-10, -1, 0, 1, 10\}$, we consider the action space $A_3 = \{-3, -1, 0, 1, 3\}$ as well.

Table 1 reveals the reward and an estimate of its standard deviation for the Light Dark problem. It can be seen that the variant with action space $A_3$ is generally harder to solve. This is because it requires multiple steps to get from the light source to the goal.

For the action space $A_{10}$ the addition of information rewards does not lead to an increased performance over the state-of-the-art solvers. However, POMCPOW fails to solve the version with action space $A_3$. Besides, the difference between the discounted information gain $\Delta I_\gamma$ and the undiscounted information gain $\Delta I_1$ is small.

5.1.2. CONTINUOUS LIGHT DARK

The Light Dark problem is already challenging on a discrete state space. To further increase the difficulty, we remodel the problem on a continuous state space with a stochastic transition and an increased observation noise.

In the Continuous Light Dark problem the agent moves along the real line with steps that are normally distributed around the chosen action with noise $\sigma_T = 0.1$. As in the previous problem, the two action spaces $A_{10}$ and $A_3$ are considered. Since the origin is a null set in the continuous state space, a unit ball around the origin is chosen as the goal area where the agent receives the positive reward. While the observations remain normally distributed, their standard deviation is increased to $\sigma_x(s) = \sqrt{2}|s - 10| + 0.5$, leading to less informative observations.

The results for the Continuous Light Dark problem for each of the two action spaces are also listed in Table 1. While IPFT still performs well, the benchmark solvers cannot
Table 1. Results for the discrete Light Dark and Continuous Light Dark problems for the two actions spaces $A_{10}$ and $A_3$. The table shows the mean reward of 1000 simulations and its estimated standard deviation.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Light Dark problem</th>
<th>Continuous Light Dark problem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>action space $A_{10}$</td>
<td>action space $A_3$</td>
</tr>
<tr>
<td>IPFT($\Delta I_1$)</td>
<td>58.2 ± 0.4</td>
<td>34.8 ± 0.7</td>
</tr>
<tr>
<td>IPFT($\Delta I_3$)</td>
<td>55.4 ± 0.5</td>
<td>27.8 ± 0.8</td>
</tr>
<tr>
<td>POMCPOW</td>
<td>58.6 ± 0.5</td>
<td>-2.6 ± 0.9</td>
</tr>
<tr>
<td>PFT-DPW</td>
<td>57.4 ± 0.5</td>
<td>33.9 ± 0.8</td>
</tr>
</tbody>
</table>

![Exemplary trajectories for the continuous Light Dark problem with action space $A_3$. Both trajectories start in $s_0 = -5$.](image)

Figure 3. Exemplary trajectories for the continuous Light Dark problem with action space $A_3$. Both trajectories start in $s_0 = -5$.

Table 2. Results for the Laser Tag problem. The table shows the mean reward of 1000 simulations and its estimated standard deviation.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Laser Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPFT($\Delta I_1$)</td>
<td>-9.0 ± 0.2</td>
</tr>
<tr>
<td>IPFT($\Delta I_3$)</td>
<td>-8.9 ± 0.2</td>
</tr>
<tr>
<td>POMCPOW</td>
<td>-9.9 ± 0.2</td>
</tr>
<tr>
<td>PFT-DPW</td>
<td>-12.0 ± 0.2</td>
</tr>
</tbody>
</table>

As Table 2 shows, IPFT achieves higher rewards than the benchmark solvers. While it outperforms PFT-DPW, the result of POMCPOW is only slightly worse. In general, the performance gap is not as significant as for the Continuous Light Dark problem. The reason for this is that both terms in the shaped reward function in Equation (1) are maximized by the same actions: Searching for the target increases the chance for high rewards as well as gathers information. Hence, adding the information reward does not provide as much additional guidance as in other problems.

5.2. Empirical Parameter Sensitivity Analysis

We investigate the sensitivity of the IPFT solver with respect to the number of particles $m$ and the weight $\lambda$ balancing information gathering and reward maximization. To this end, we carry out more experiments on the Continuous Light Dark problem with action space $A_3$ where those parameters are varied. The undiscounted information gain $\Delta I_1$ is used for reward shaping.

Figure 4 shows the results on two separate axes. It can be seen that the performance degrades gracefully with varying $\lambda$, while keeping a fixed number of $m = 20$ particles (blue). For $\lambda \rightarrow 0$, IPFT and hence its performance converge to
PFT-DPW. With increasing $\lambda$ the information gain is over-weighted and less rewards are collected. Nonetheless, there is a large region ($\lambda \in [60, 120]$) in between those two extremes where the performance is not very sensitive to the choice of $\lambda$, which makes it easy to find a suitable parameter in practice.

The number of particles $m$ is varied using a constant information weight $\lambda = 60$ (red). For larger particle sets with $m > 100$, the performance degrades because the increased computation time $O(m^2)$ necessary for each entropy calculation outweighs the increased accuracy of the particle set approximation. In this scenario the algorithm also works well with as few as $m = 4$ particles. Since the region where the maximal reward is attained is relatively flat, the reward is not very sensitive with respect to $m$.

6. Related Work

Information gathering has been integrated in sequential decision making problems before, in particular in the areas of exploration and active sensing.

Cassandra et al. used POMDPs to solve a localization problem (1996). They guide the search by choosing an information gathering action whenever the entropy exceeds a threshold. Another work used an optimization criterion combining the value function with the one-step expected entropy, resulting in a myopic algorithm (Burgard et al., 1997). Kreucher et al. maximized the Rényi divergence to choose the most informative sensor (2005). They approximated distributions over a continuous state space with a particle filter. Hoffmann et al. estimate information theoretic costs from particle filters and maximize the mutual information of multiple agents in the next time step to localize an object with decentralized planning (2006). In contrast to our work, these approaches result in myopic, information-greedy policies because they only consider the information gained in the next step.

Some works were able to consider information gathering in long-term planning. Roy & Thrun augmented the state with the entropy and solved the resulting problem with dynamic programming (2000). Unlike our work, they used a parametric approach which allows only unimodal beliefs. Another strategy to obtain non-myopic policies is to sample subgoals where the observation distribution has a low entropy and the immediate reward is high (He et al., 2010; Ma & Pineau, 2015). However, the uncertainty in the observation distribution does not necessarily reflect how much information is actually gained. Spaan et al. proposed to implement state-based information rewards by enlarging the action space with so-called commit actions, which result in high rewards if the state can be guessed correctly (2015). Because one such commit action is necessary for every state that has to be distinguished, the action space quickly becomes intractable. Dressel & Kochenderfer implemented belief-dependent rewards in the offline POMDP solver SAR-SOP to solve localization problems (2017). Besides being unsuitable for online planning, this also restricts the reward function to be PWLC. They chose reward functions based on the $\ell_{\infty}$-norm and on the commit actions from Spaan et al. mentioned above.

Reward shaping based on information maximization was used by Mafi et al. in the context of reinforcement learning (2011). Similar to our work, their reward function combines information gain and task execution. Eck et al. applied PBRS to online planning in POMDPs (2016). They describe four types of potential functions that can be used to guide the agent and provide theoretical results. The POMDP-lite algorithm solves the subclass of POMDPs where the un-
observable states are constant or change deterministically (Chen et al., 2016). It measures the information gain as the $L_1$ divergence between consecutive beliefs and adds it as a bonus to the reward. Saborío & Hertzberg develop the idea of partial goal satisfaction to construct a domain-independent heuristic for PBRS. They apply PBRS to states sampled by POMCP instead of planning directly over beliefs. In contrast to our work, all of these approaches consider only discrete state spaces.

The referenced works indicate the benefits of utilizing information measures in decision making. Nevertheless, none of them could efficiently deal with arbitrary belief-based rewards on continuous domains online.

7. Conclusion and Future Work

Decision making under uncertainty is particularly challenging in problems on large spaces where the reward is only obtained after long action sequences of information gathering. In this work, we propose to consider the value of information in the objective function in order to facilitate the search for promising actions. We develop a novel online algorithm that enables information-theoretic reward shaping on continuous spaces. Our algorithm, IPFT, performs MCTS with particle-based belief simulations and utilizes DPW to accommodate continuous action and observation spaces. As a result, the algorithm is capable of solving arbitrary $\rho$POMDPs on continuous domains.

Our evaluation reveals that problems which require information gathering for task execution can be solved much more efficiently using our approach than with state-of-the-art algorithms. Furthermore, the algorithm can be easily used on new problems since our analysis shows that the performance is not very sensitive to its hyperparameters.

In our future research we will use IPFT to investigate more complex scenarios. For instance, search and rescue is an important application which likely benefits from information-based reward shaping. Since we developed a universal $\rho$POMDP solver this allows us to also consider pure information gathering tasks on continuous spaces, as are common in active sensing.

Acknowledgements

The authors gratefully acknowledge the fruitful discussions and review of this work by Martin Lauer on the original version of this document.

References


Information Particle Filter Tree Algorithm for Continuous ρPOMDPs


