An Online and Approximate Solver for POMDPs with Continuous Action Space

Konstantin M. Seiler, Hanna Kurniawati, and Surya P. N. Singh

Abstract— **For agile, accurate autonomous robotics, it is desirable to plan motion in the presence of uncertainty. The Partially Observable Markov Decision Process (POMDP) provides a principled framework for this. Despite the tremendous advances of POMDP-based planning, most can only solve problems with a small and discrete set of actions. This paper presents General Pattern Search in Adaptive Belief Tree (GPS-ABT), an approximate and online POMDP solver for problems with continuous action spaces. Generalized Pattern Search (GPS) is used as a search strategy for action selection. Under certain conditions, GPS-ABT converges to the optimal solution in probability. Results on a box pushing and an extended Tag benchmark problem are promising.**

I. INTRODUCTION

Reasoned action in the presence of sensing and model information is central to robotic systems. In all but the most engineered cases, there is uncertainty.

The Partially Observable Markov Decision Process (POMDP) [22] is a rich, mathematically principled framework for planning under uncertainty, particularly in situations with hidden or stochastic states [10]. Due to uncertainty, a robot does not know the exact state, though it can infer a set of possible states. POMDP-based planners represent these sets of possible states as probability distributions over the state space, called *beliefs*. They also represent the non-determinism in the effect of performing an action and the errors in sensing/perception as probability distributions. POMDP-based planners systematically reason over the belief space (i.e., the set of all possible beliefs, denoted as B) to compute the best action to perform, taking the aforementioned non-determinism and errors into account.

The past decade has seen tremendous advances in the capability of POMDP solvers. From algorithms that take days to solve toy problems with less than a dozen states [10] to algorithms that take only a few minutes to generate near-optimal solutions for problems with hundreds of thousands and even continuous state space [2], [15], [21]. These planners have moved the practicality of POMDP-based approaches far beyond robot navigation in a small 2D grid world, to mid-air collision avoidance of commercial aircraft (TCAS) [25], grasping [13], and non-prehensile manipulation [9], to name but a few examples.

While some actions are inherently discrete (e.g., a switch), in an array of applications a continuous action space is natural (e.g., grasping, targeting, cue sports, etc.), particularly when needing accuracy or agility. However, most solvers can only solve problems with a small and discrete set of actions because to find the best action to perform (in the presence of uncertainty), POMDP solvers compute the mean utility over the set of states and observations that the robot may occupy and perceive whenever it performs a particular action, and then find the action that maximizes the mean value. Now, key to the aforementioned advances in solving POMDPs is the use of sampling to trade optimality with approximate optimality for speed. While sampling tends to work well for estimating the mean, finding a good estimate for the max (or min) is more difficult. Therefore, most successful solvers rely on sampling to estimate the mean total utility and, in so doing, enumerate over all possible actions to find the action that maximizes the mean value. When the action space is continuous, such enumeration is no longer feasible.

This paper presents General Pattern Search in Adaptive Belief Tree (GPS-ABT), an approximate and online POMDP solver for problems with continuous action spaces, assuming the action space is a convex subset of $\mathbb{R}^n$. GPS-ABT alleviates the difficulty of solving POMDP with continuous action using direct search methods, in particular the General Pattern Search (GPS). Unlike gradient descent approaches, the most common method for search in continuous space, GPS does not require derivatives, which are difficult to have when the objective function is not readily available and needs to be estimated, as is the case with solving POMDPs. GPS-ABT is built on top of *Adaptive Belief Tree (ABT)* [15], a recent online POMDP planner. GPS-ABT uses GPS to identify a small set of candidate optimal actions, and ABT to find the best action among the candidate actions and estimate the expected value of performing the candidate actions. GPS-ABT computes an initial policy offline, and then improves this policy during runtime, by interleaving candidate optimal actions identification and value estimation. GPS-ABT has been integrated with an open source software implementation of ABT (TAPIR, `http://robotics.itee.uq.edu.au/~tapir`)[11] and is available for download.

Under certain conditions, GPS-ABT converges to the optimal solution in probability. Let $Q(b,a)$ be the expected value of performing action a from belief $b \in B$ and continuing optimally afterwards. Suppose $\mathcal{R}^*(b_0)$ is the set of beliefs reachable from the given initial belief b_0 under the optimal policy. When $Q(b,.)$ is continuous with a single maximum at each belief $b \in \mathcal{R}^*(b_0)$, GPS-ABT converges to an optimal solution in probability. Otherwise, GPS-ABT converges to the local optimal solution in probability.

The remainder of this paper is organised as follows: An introduction to POMDP and solution strategies is given in

Robotics Design Lab, School of Informaton Technology and Electrical Engineering, University of Queensland, QLD 4072, Australia. `{k.seiler, hannakur, spns}@uq.edu.au`

Section II. The proposed GPS-ABT algorithm is detailed in Sections III and IV. Convergence to the optimal solution is proven in Section V. Simulation results are presented in Section VI. The paper concludes with Section VII.

II. RELATED WORK

A. POMDP Background

A POMDP model is a tuple $\langle S, A, O, T, Z, R, b_0, \gamma \rangle$, where S is the set of states, A is the set of actions, and O is the set of observations. At each step, the agent is in a state $s \in S$, takes an action $a \in A$, and moves from s to an end state $s' \in S$. To represent uncertainty in the effect of performing an action, the system dynamic from s to s' is represented as a conditional probability function $T(s,a,s') = f(s'|s,a)$. To represent sensing uncertainty, the observation that may be perceived by the agent after it performs action a and ends at state s', is represented as a conditional probability function $Z(s',a,o) = f(o|s',a)$. The conditional probability functions T and Z may not be available explicitly. However, one can use a *generative model*, which is a black box simulator that outputs an observation perceived, reward received, and next state visited when the agent performs the input action from the input state.

At each step, a POMDP agent receives a reward $R(s,a)$, if it takes action a from state s. The agent's goal is to choose a suitable sequence of actions that will maximize its expected total reward, while the agent's initial belief is denoted as b_0. When the sequence of actions has infinite length, we specify a discount factor $\gamma \in (0,1)$, so that the total reward is finite and the problem is well defined.

A POMDP planner computes an *optimal policy* that maximizes the agent's expected total reward. A POMDP policy $\pi\colon B \to A$ assigns an action a to each belief $b \in B$, where B is the belief space. A policy π induces a value function $V(b,\pi)$ which specifies the expected total reward of executing policy π from belief b, and is computed as

$$V(b,\pi) = E[\sum_{t=0}^{\infty} \gamma^t R(s_t,a_t)|b,\pi] \tag{1}$$

To execute a policy π, an agent executes action selection and belief update repeatedly. For example, if the agent's current belief is b, it selects the action referred to by $a = \pi(b)$. After the agent performs action a and receives an observation o according to the observation function Z, it updates b to a new belief b' given by

$$b'(s') = \tau(b,a,o) \quad = \quad \eta Z(s',a,o) \int_{s \in S} T(s,a,s') ds \tag{2}$$

where η is a normalization constant.

An offline POMDP planner computes the policy prior to execution, whilst an online POMDP planner interleaves policy generation and execution. Suppose, the agent's current belief is b. Then, an online planner computes the best action $\pi^*(b)$ to perform from b and executes the action. After the agent performs action $\pi^*(b)$ and receives an observation o according to the observation function Z, it updates b to a new belief b' based on Section II-A. The agent then computes the best action $\pi^*(b')$ to perform from b', and repeats the process.

B. Related POMDP-based Planners

Key to the tremendous advances in both offline and online POMDP-based planning is the use of sampling to trade optimality with approximate optimality in exchange for speed. Sampling based planners reduce the complexity of planning in the belief space B by representing B as a set of sampled beliefs and planning with respect to this set only.

In offline planning, the most successful approach is the point-based approach [17], [23]. Point-based approach represents the belief space with a representative set of sampled beliefs, and generate a policy by iteratively performing Bellman backup on V at the sampled beliefs rather than the entire belief space. Although these solvers were originally designed for discrete state, action, and observation spaces, several pieces of work have extended point-based approach to continuous state space [2], [5], [19] and continuous observation space [3], [8], [19]. A few pieces of work [14], [19] have also extended point-based approach to handle continuous action space under limited conditions.

Many successful online POMDP-based planners rely on sampling, e.g., [7], [15], [20], [21], [24]. Some of these solvers [15], [21], [24] can handle large and continuous state and observation spaces. However, continuous action space remains a challenge. The reason is, these solvers rely on sampling to construct a more compact representation of the large and continuous state and observation spaces. Since the optimal value function is computed as expectation over state and observation spaces, one can use results from probability theory to ensure that even uniform random sampling can compute a good estimate on the expected value over the state and observation space. However, the value function is computed as maximum over the action space. In general, random sampling cannot guarantee convergence to a good estimate of the maximum operator, and hence cause significant difficulties in solving POMDP problems where the action space is large or continuous.

Aside from the aforementioned approaches, several POMDP solvers for problems with continuous state, action, and observation spaces have been proposed, e.g., [4], [16], [18]. The work in [4] restricts beliefs to Gaussian distribution. The work in [16] restricts to find optimal solution in a class of policies. And the work in [18] considers only the most likely observation during planning, constructing a path rather than a policy, and regenerate a new path whenever the perceived observation is not the same as the most likely observation. In this paper, we propose a general online POMDP solver for problems with large and continuous action spaces taking into account the different observations an agent may perceive, and without restricting the type of beliefs that the agent may have.

III. OVERVIEW OF GPS-ABT

Similar to ABT, GPS-ABT constructs an initial policy offline and continues improving the policy during runtime. Algorithm 1 presents the overview of the proposed online POMDP solver. GPS-ABT embeds the optimal policy in a belief tree. Let $\mathcal{T}$ denote the belief tree. Each node in $\mathcal{T}$

Algorithm 1 GPS-ABT (b_0)

```
PREPROCESS (OFFLINE)
GENERATE-POLICY($b_0$).
$b$ = $b_0$.

RUNTIME (ONLINE)
while running do
  while there is still time do
    IMPROVE-POLICY($\mathcal{T}$, $b$).
  $a$ = Get best action in $\mathcal{T}$ from $b$.
  Perform action $a$.
  $o$ = Get observation.
  $b$ = $\tau$ ($b$, $a$, $o$).
  $t = t + 1$.
```

represents a belief. For writing compactness, the node in $\mathcal{T}$ and the belief it represents are referred to interchangeably. The root of $\mathcal{T}$ represents the initial belief b_0. Each edge $\overline{bb'}$ in $\mathcal{T}$ is labelled by a pair of action and observation a–o. An edge $\overline{bb'}$ with label a–o means that when a robot at belief b performs action a and perceives observation o, its next belief will be b', i.e., $b' = \tau(b, a, o)$ where $b, b' \in B$, $a \in A$, and $o \in O$. The function GENERATE-POLICY in Algorithm 1 constructs the initial $\mathcal{T}$, while IMPROVE-POLICY further expands $\mathcal{T}$ and improves the policy embedded in it. The policy $\pi_{\mathcal{T}}(b)$ from b that GPS-ABT embeds in $\mathcal{T}$ is then the action that maximizes the current estimate, i.e.,

$$\pi_{\mathcal{T}}(b) = \underset{\{a \in A : N(b,a) > 0\}}{\arg\max} \hat{Q}(b,a) \; . \tag{3}$$

where $N(b,a)$ is the number of edges from b whose action component of its label is a and $\hat{Q}(b,a)$ denotes the estimated Q-value. Q-value $Q(b,a)$ is the value of performing action a from belief b and continuing optimally afterwards, i.e., $Q(b,a) = R(b,a) + \gamma \sum_{o \in O} \tau(b,a,o) \max_{a' \in A} Q(\tau(b,a,o), a')$.

When the action space A is continuous, most online POMDP solvers [15], [21], [24] discretize A and search for the best action to perform among this set of discrete actions. Instead of limiting the search to a pre-defined set of discrete actions, GPS-ABT searches over a continuous action space.

The most common method to search a continuous space is the Gradient Descent approach. However, this approach cannot be applied here because the objective function $Q(b,.)$ is not readily available for evaluation. Instead, in each step its value can only be estimated by the currently available $\hat{Q}(b,.)$ which only converges towards $Q(b,.)$ as the number of search iterations increases. The estimates given by the $\hat{Q}(b,.)$ functions are far too crude to calculate a meaningful estimate for a derivative using finite differences. Instead, GPS-ABT is based on direct search methods that don't require derivatives.

In particular, GPS-ABT uses the *Generalised Pattern Search* (GPS) approach to identify a small subset of A as candidates for the best action from a belief b, and interleaves this identification step with the fast value estimation of ABT [15]. The details on how GPS-ABT uses GPS, constructs the belief tree $\mathcal{T}$, and estimates the value function to find

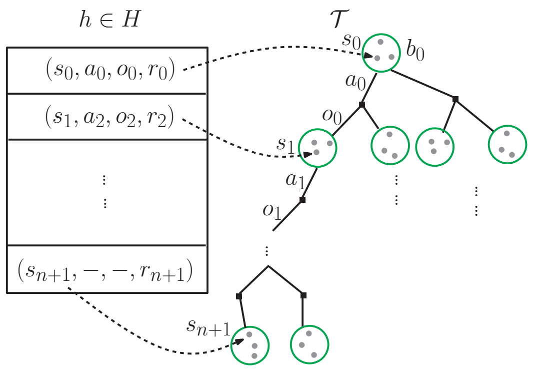

Fig. 1: Illustration of an association between an episode $h \in H$ and a path in the belief tree $\mathcal{T}$.

a good approximation to the optimal policy is presented in Section IV.

GPS-ABT will converge to the optimal policy in probability whenever $Q(b,.)$ is a uni-modal function on A, i.e., it has only one point which is a maximum of an open neighbourhood around itself, for which the utilised GPS method converges to a maximum. When the Q function is multi-modal in A, GPS-ABT will converge to a local maximum. The convergence proof is presented in Section V.

IV. Details of GPS-ABT

A. Constructing $\mathcal{T}$ and Estimating the Value Function

To construct the belief tree $\mathcal{T}$ and estimate the value function, GPS-ABT uses the same method as ABT. We present an overview of the strategies here, but more details on ABT are available in [15].

GPS-ABT samples beliefs using a generative model, sampling sequences of states, actions, observations, and rewards, called episodes, and maintains the set of sampled episodes, denoted as H. To sample an episode, GPS-ABT samples a state $s_0 \in S$ from the initial belief b_0. It then selects an action $a_0 \in A$ and calls a generative model with state s_0 and action a_0 to receive a new state s_1, observation o_0 and reward r_0. The quadruple (s_0, a_0, o_0, r_0) is then added as first element into the episode and the process repeats using s_1 as initial state for the next step. Once no further action is to be selected, $(s_n, -, -, r_n)$ is added as final element to the episode where r_n denotes the immediate reward for being at state s_n. Note that to select an action, GPS-ABT modifies ABT by first selecting a set of candidate optimal actions using GPS, as detailed in Section IV-B.

When an episode h is sampled, its elements are associated with the belief tree $\mathcal{T}$ in the following way: the first element (s_0, a_0, o_0, r_0) is associated with the root of the tree to represent b_0. The next element of h is associated with the vertex b_1 where the edge from b_0 to b_1 is labelled (a_0, o_0). The remaining elements are associated with vertices of $\mathcal{T}$ traversing the tree the same way. The process of associating elements with the belief tree is illustrated in Fig. 1. Through this association process, each episode as a whole is associated with a path ϕ in $\mathcal{T}$. As there can be many episodes that are associated with the same path (i.e. all episodes that have the same sequence of actions and observations), the set of episodes in H that are associated with a particular path ϕ in $\mathcal{T}$ is denoted $H_\phi \subseteq H$. Thus, a belief b contained in $\mathcal{T}$ is represented by a set of particles, the sampled states s of the episode elements that are associated with its vertex.

Algorithm 2 presents a pseudo-code of how GPS-ABT samples an episode and associates it with the belief tree $\mathcal{T}$.

Algorithm 2 GPS-ABT: SAMPLE EPISODES(b_{start})

```
1:  $b$ = $b_{start}$ ; doneMode = UCB
2:  Let $l$ be the depth level of node $b$ in $\mathcal{T}$.
3:  Let $s$ be a state sampled from $b$.
    The sampled state $s$ is essentially the state at the $l^{th}$
    quadruple of an episode $h' \in H$.
4:  Initialize $h$ with the first $l$ elements of $h'$.
5:  while $\gamma^l > \varepsilon$ AND doneMode == UCB do
6:      $A_{active,b}$ = GENERAL-PATTERN-SEARCH($b$).
7:      $A' = \{$actions that labelled the edges from $b$ in $\mathcal{T}\}$.
8:      if $A_{active,b} \setminus A' = \emptyset$ then
9:          $a$ = UCB-ACTION-SELECTION($\mathcal{T}$, $b$).
10:     else
11:         $a$ = Sample uniformly at random from $A_{active,b} \setminus A'$.

12:         doneMode = Default.
13:     $(o,r,s')$ = GenerativeModel($s$, $a$).
14:     Insert $(s,a,o,r)$ to $h$.
15:     Add $h_l.s$ to the set of particles that represent belief
        node $b$ and associate $b$ with $h_l$.
16:     $b$ = child node of $b$ via an edge labelled $a$-$o$. If no
        such child exist, create the child.
17:     $s$ = $s'$; $l = l+1$.
18: if doneMode == Default then
19:     $r$ = Estimate the value of $b$ using DEFAULT-POLICY.
20: Insert $(s,-,-,r)$ to $h$.
21: Add $h_l.s$ to the set of particles that represent belief node
    $b$ and associate $b$ with $h_l$.
22: UPDATE-VALUES($\mathcal{T}$, $h$)
23: Insert $h$ to $H$.
```

Similar to ABT, GPS-ABT estimates $Q(b,a)$ as

$$\hat{Q}(b,a) = \frac{1}{|H_{(b,a)}|} \sum_{h \in H_{(b,a)}} V(h,l) \tag{4}$$

where $H_{(b,a)} \subseteq H$ is the set of all episodes associated with a path in $\mathcal{T}$ that starts from b_0, passes through b and then follows action a, l is the depth level of b in $\mathcal{T}$, and $V(h,l)$ is the value of an episode h starting from the l^{th} element. $V(h,l)$ is computed as

$$V(h,l) = \sum_{i=l}^{|h|} \gamma^{i-l} R(h_i.s, h_i.a) \tag{5}$$

where γ is the discount factor and R is the reward function. It may seem odd that Eqs. (4) and (5) can approximate $Q(b,a)$. It turns out one can ensure that as the number of episodes in $H_{(b,a)}$ increases, $\hat{Q}(b,a)$ converges to $Q(b,a)$ in probability, if actions for sampling new episodes are selected using the UCB1 algorithm [1].

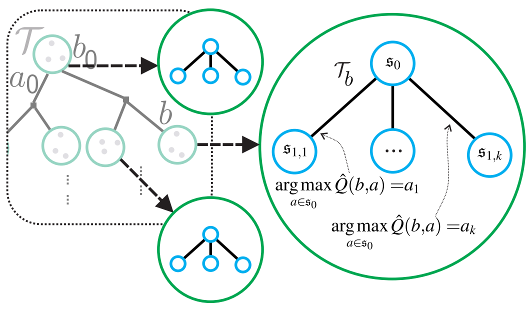

Fig. 2: Illustration of a tree structure that maintains the sets of candidate optimal actions.

The UCB1 algorithm [1] selects an action according to

$$a = \arg\max_{a \in A} \left(\hat{Q}(b,a) + c\sqrt{\frac{\log(|H_b|)}{|H_{(b,a)}|}} \right) \tag{6}$$

where $H_b \subseteq H$ denotes the set of episodes associated with b and $H_{(b,a)} \subseteq H_b$ is the set of episodes that selects action $a \in A$ during the step associated with b. c is a scalar factor that determines the ratio between exploration and exploitation.

Note that Eq. (6) is only applicable when all actions $a \in A$ have been tried for belief b in the past and thus $H_{(b,a)}$ is non-empty for all $a \in A$. Thus, while there are untried actions, Eq. (6) is not used and instead an untried action is selected at random, i.e., an action is drawn from the set

$$\{a \in A : |H_{(b,a)}| = 0\} \ . \tag{7}$$

This implies that the set of actions to be considered in each step must be finite since otherwise UCB1 would forever select random untried actions instead of biasing the search towards parts of the tree where the value function is high.

B. General Pattern Search

At the core of a GPS method is a stencil which is a small set of points $\mathfrak{s} \subseteq A$ to be evaluated. Before the search begins, the stencil is initialised to its base form $\mathfrak{s}_0 \subseteq A$. The search then continues by evaluating the objective function for each of the points in $\mathfrak{s}_0$. Depending on comparisons of function values, the stencil is transformed into $\mathfrak{s}_1$ for the next step. The stencil transformations are often based on translations and/or scaling in size. Once a new stencil is generated, the search process repeats with the new stencil until a convergence criterion is met.

Various GPS methods can be used. Two methods used within this work are *Golden Section Search* for one dimensional search problems and *Compass Search* for higher dimensional searches. These two methods are detailed in Appendices A and B.

Because the objective function of the maximisation process, $Q(b,.)$, is unknown, GPS-ABT uses the available estimates $\hat{Q}(b,.)$ to perform a general pattern search. The estimates for $\hat{Q}(b,.)$ are pointwise converging towards $Q(b,.)$, but because the estimates are constantly updated throughout the search, the decisions made by a GPS algorithm must be revised each time the outcome of the relevant estimates change.

In order to accommodate these changes in the objective function, the state of the search is maintained in a tree structure (illustrated in Fig. 2). There exists one GPS-tree $\mathcal{T}_b$ for each belief b contained in the belief tree $\mathcal{T}$. Each vertex $v \in \mathcal{T}_b$ contains a stencil $\mathfrak{s}_v$. The edges leaving a vertex are labelled by the actions $a_i \in \mathfrak{s}_v$ and lead to a vertex containing the stencil for the case that

$$a_i = \arg\max_{a \in \mathfrak{s}_v} Q(b,a) \; . \tag{8}$$

If within a vertex an action $a \in \mathfrak{s}_v$ fulfils Eq. (8), the corresponding child is denoted the *active child*. In the rare case that several actions achieve the maximum, one of them is selected arbitrarily to be the sole active child.

While the tree $\mathcal{T}_b$ is in theory infinite, in practise it is bounded by several factors. First, new child vertices for a vertex v are only created if they are active and there is a sufficient estimate for all values of $\hat{Q}(b,a), a \in \mathfrak{s}_v$. Thus, only those parts of the search tree are created that are actually needed. Second, the search is limited by a convergence criterion such that no further children are created if the search converged sufficiently. In case of the Golden Section Search, the interval size is lower bounded whereas for Compass Search the radius is used as an indicator. Last, there is also an artificial limit on the maximal depth of the tree such that

$$\text{depth}(\mathcal{T}_b) \leq \log_{c_{depth}}(|H_b|) = \frac{\ln(|H_b|)}{\ln(c_{depth})} \tag{9}$$

where c_{depth} is an exploration constant. A new child node is only created if it doesn't violate Eq. (9). This limit prevents the tree from growing too fast in the beginning and results in more time being spent refining the relevant estimates $\hat{Q}(b,.)$.

The active children define a single path in $\mathcal{T}_b$ from the root to one of the leaves by starting at the root and always following the edge to the active child until a leave is reached. The set of actions that are associated with a vertex on the active path is denoted $A_{active,b} \subseteq A$, the set of active actions for the belief b. Algorithm 3 contains pseudo-code to compute the set $A_{active,b}$.

When sampling new episodes, action selection is performed based on restricting UCB1 to the set of active actions. Thus, Eq. (6) becomes

$$a = \arg\max_{a \in A_{active,b}} \left(\hat{Q}(b,a) + c\sqrt{\frac{\log(|H_b|)}{|H_{(b,a)}|}} \right) \tag{10}$$

Similarly, in order to follow the policy contained in the belief tree $\mathcal{T}$, the action $a \in A_{active,b}$ with the largest estimate $\hat{Q}(b,.)$ is executed. Thus, Eq. (3) becomes

$$a = \arg\max_{\{a \in A_{active,b} : |H_{(b,a)}| > 0\}} \hat{Q}(b,a) \; . \tag{11}$$

Note that this method accommodates well for hybrid action spaces where the action space is the union of a continuous space A_{cont} and a finite space A_{disc}. Then the patters search operates on A_{cont} as above and actions A_{disc} are always considered active when evaluating Eqs. (10) and (11). An example of such a hybrid action space is shown in the continuous tag problem in Section VI-B.

Algorithm 3 SAMPLE GENERAL-PATTERN-SEARCH(b)

```
1: active = $\emptyset$
2: $v$ = $\mathcal{T}_b$
3: hasMore = TRUE
4: **while** hasMore **do**
5:     active = active $\cup\, \mathfrak{s}_v$
6:     $a_{\max}$ = $\arg\max_{a \in \mathfrak{s}_v} \hat{Q}(b,a)$
7:     **if** $v$ has child for $a_{\max}$ **then**
8:         $v$ = child for $a_{\max}$
9:     **else**
10:        **if** child creation condition for $a_{\max}$ ok **then**
11:            add child for $a_{\max}$ to $\mathcal{T}_b$
12:            $v$ = child for $a_{\max}$
13:        **else**
14:            hasMore = FALSE
15: **return** active
```

V. CONVERGENCE TO OPTIMAL SOLUTION

It is shown in [15], [21], [12], that ABT with a finite action space A converges towards the optimal policy in probability. The proof is based on an induction on the belief tree $\mathcal{T}$ going backwards from a fixed planning horizon. The planning horizon defines the maximum depth of $\mathcal{T}$ and induction starts at the leaves of $\mathcal{T}$. For a belief b associated with a leave of the belief tree, the value function $\hat{V}^*(b)$ converges towards $V^*(b)$ as it depends only on the immediate reward received for states $s \in \text{support}(b)$ causing convergence in probability as new episodes that reach b are sampled.

Using a similar argument, it is then shown that Q-value estimates and value estimates $\hat{Q}(b,.)$ and $\hat{V}^*(b)$ converge towards the optimal functions for any belief $b \in T$ in probability provided that the value functions of all its children are known to converge.

The proof relies on the observation that the bias term of UCB1 (i.e. the second summand in Eq. (6)) ensures that each action will be selected infinitely often at each belief almost certainly, thus guaranteeing the necessary convergence. For full details of the proof for ABT with finite action space, please see the references above.

To extend this proof to GPS-ABT, it is necessary to ensure that the induction step still applies. It needs to be shown that the value function $\hat{V}^*(b)$ converges in probability if all its children's value functions $\hat{Q}(b,.)$ converge whenever they are reached by episodes infinitely often.

Lemma 1: $\hat{V}^*(b)$ converges towards $V^*(b)$ if the used GPS method (Compass Search or Golden Section Search) converge towards the maximum when applied to the Q-value $Q(b,.)$.

Proof: Note that for true convergence, the convergence criterion needs to be shown for both the GPS method and the tree T_b.

Our proof is by induction on $\mathcal{T}_b$. Let $\lambda_{v,n}$ be the relative frequency a vertex $v \in \mathcal{T}_b$ has been active during an UCB1 selection step. That is

$$\lambda_{v,n} = \frac{m_{v,n}}{n} \tag{12}$$

where $m_{v,n}$ is the number of times v was part of the active path during the first n UCB1 steps.

To start the induction, let v be the root of $\mathcal{T}_b$. Then $\lambda_{v,n} = 1$ for all n since the root is always part of the active path. For the induction step, however, it is sufficient to assume that $\lambda_{v,n}$ does not converge to zero for $n \to \infty$.

Since the relative frequency does not converge to zero, the actions $a \in \mathfrak{s}_v$ contained in the stencil of v are each active infinitely often. Thus, they will be selected by UCB1 infinitely often. Moreover, they will be chosen by UCB1 at least as often as dictated by the bias term of Eq. (6). (note that the bias term dictates a guarantee for the action to be chosen with a logarithmic frequency whereas $\lambda_{v,n}$ guarantees the action to be available for selection (active) with a linear frequency).

If there is a single best element a_{max} in the stencil maximising $Q(b,.)$, then, due to convergence, a_{max} also maximises $\hat{Q}(b,.)$ all but finitely many times. Thus, after finitely many steps the 'correct' child v_c stays active indefinitely. This implies that the convergence behaviour of $\lambda_{v_c,n}$ is identical to that of $\lambda_{v,n}$ and the induction continues by considering v_c.

In cases where several actions $a_1,...,a_k \in \mathfrak{s}$, $k > 1$ maximise $Q(b,.)$, the convergence to a single active child is not guaranteed. Instead it can happen that different children belonging to the maximising actions $a_1,...,a_k$ are set active infinitely often. Then there must be at least one child v_c where $\lambda_{v_c,n}$ does not converge to zero. This is guaranteed because

$$\lambda_{v,n} = \sum_{v_c \text{ child of } v} \lambda_{v_c,n} \,. \tag{13}$$

Thus, the induction continues using all children v_c where $\lambda_{v_c,n}$ does not converge to zero.

Note that in cases where several actions maximise $Q(b,.)$, the behaviour of the underlying GPS is also arbitrary. The convergence of the GPS, however, is guaranteed no matter which of the maximising actions the stencil transformation is based on. Let Φ_{conv} be the set of all paths that could be chosen by the GPS applied to the true Q-value function $Q(b,.)$. The above induction implies that all vertices $v \in \mathcal{T}_b$ where $\lambda_{v,n}$ does not converge to zero are part of a path in Φ_{conv}. This completes the proof. ■

VI. SIMULATION RESULTS

Two simple applications that illustrate GPS-ABT are presented here.

A. Push Box

Push Box is a problem where a robot has to manoeuvre a box into a goal region solely by bumping into it and pushing it away (loosely analogous to air hockey). The robot and the box are considered to be discs with a diameter of 1 unit length, thus their state can be fully described by the location of the center point. Subsequently, the continuous state space S consists of four dimensions, two for the coordinates (x_r, y_r) that represent the robot's position as well as a second set of coordinates (x_b, y_b) that represents the coordinates of the box. The action space is the two dimensional interval $A = [-1,1] \times [-1,1] \subseteq \mathbb{R}^2$.

In case the robot does not touch the object during the move, the transition function T is defined as

$$T\big((x_r,y_r,x_b,y_b),(x_a,y_a)\big) = (x_r + x_a, y_r + y_a, x_b, y_b) \tag{14}$$

for $(x_r,y_r,x_b,y_b) \in S$ and $(x_a,y_a) \in A$.

If the robot touches the box at any point during its move along the straight line from (x_r,y_r) to (x_r+x_a, y_r+y_a), the box is pushed away. The push is calculated as

$$\begin{pmatrix}\Delta x_b\\ \Delta y_b\end{pmatrix} = (1+r_s)\, c_s \left(\vec{n}\begin{pmatrix}x_a\\ y_a\end{pmatrix}\right)\left(\vec{n} + \begin{pmatrix}r_x\\ r_y\end{pmatrix}\right)\,, \tag{15}$$

where $\vec{n}$ is the unit vector from the center of the robot to the center of the box at the moment of contact. It defines the direction of the push. The distance of the push is set proportional to the robot's speed in the direction of $\vec{n}$ as represented by the scalar product $\vec{n}\binom{x_a}{y_a}$. The factor c_s is a constant that defines the relative intensity of a push. The factors r_s, r_x and r_y are random numbers to represent the move uncertainty of a push. These dynamics result in a push that is similar the dynamics of the game of 'air hockey'.

The robot has a bearing sensor with a coarse resolution to localise the current position of the box. The bearings from 0°to 360°are assigned to 12 intervals of 30°each. In addition, the robot receives information about whether it pushed the box or not. Thus, an observation is calculated by

$$o = \text{floor}\left(\frac{\text{atan2}(y_b - y_r, x_b - x_r) + r_o}{30}\right) + p \tag{16}$$

where r_o is a random number representing measurement uncertainty (the sum of the angles in understood to overflow and only create values between 0°and 360°. p is either 0 or 12 depending on whether a push of the box occurred or not. Thus, the total observation space O consists of 24 possible observations.

The reward function R awards a reward of 1000 when the box reaches the goal and a penalty -10 for every move as well as -1000 if either the box of the robot are on forbidden coordinates (obstacles). Any state where either the box or the robot is on a forbidden field or the box reached the goal is considered a terminal state.

The trade-off of the push box problem is that greedy actions (long shots towards the goal) are dangerous due to the uncertainty in the transition function (r_s, r_x and r_y) as well as an uncertainty about the knowledge of the position of the box. Careful actions on the other hand spend more time moving around to localise the box and perform smaller, but less risky pushes. It was indeed observed that good solutions required several pushes in order to avoid collisions.

The environment map that was used during simulations is shown in Fig. 3 and the parameters for Eqs. (15) and (16) were as follows:

- $c_s = 5$
- r_s, r_x and r_y are drawn from a truncated normal distribution with standard deviation 0.1, truncated at ± 0.1.

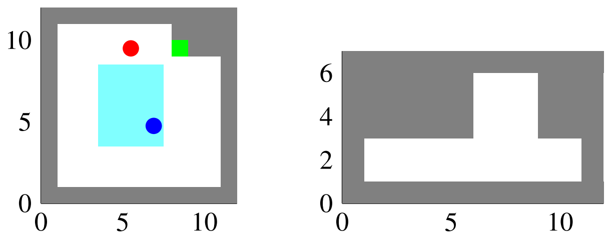

Fig. 3: *Left:* the map used for Push Box simulations. The robot (*red*) starts at a fixed position whereas the box (*blue*) is placed anywhere within the light blue region at random. The goal is marked green. Gray areas are forbidden. *Right:* the map used for Tag simulations. The initial positions of the robot and the target are chosen at random.

Algorithm	$\|A\|$	c_{depth}	horizon	mean reward
ABT	9		3	-79.3
ABT	36		3	-5.0
ABT	81		3	-73.6
GPS-ABT		5	3	238.2
GPS-ABT		7	3	223.0
ABT	9		4	-36.0
ABT	36		4	49.3
ABT	81		4	-61.2
GPS-ABT		5	4	146.1
GPS-ABT		7	4	99.6

TABLE I: Simulation results for the Push Box problem. Normal ABT is by discretising the action space A into finitely many actions. The parameter c_{depth} is explained in Eq. (9). The horizon denotes the planning horizon.

- r_o is drawn from a truncated normal distribution with standard deviation 10, truncated at ± 10.

The simulations were performed using the online planning feature of ABT. Thus, the initial policy creation phase was limited to 1 second and before each step the algorithm was allowed another second to refine the policy. Due to this high frequency the planning horizon had to be limited. Setting of 3 and 4 steps have been tested.

In order to compare the effect of planning in continuous action space compared to discretising A, the action space was discretised into 9, 36 and 81 actions respectively to test the performance achieved by normal ABT. For GPS-ABT different settings for c_{depth} in Eq. (9) were tested. All simulations were run single-threaded on an Intel Xeon CPU E5-1620 with 3.60GHz and 16GB of memory.

Statistical results of the simulation are presented in Table I. It can be seen that GPS-ABT tends to perform better than ABT with a discretised action space. This is not very surprising due to the advantage gained by being able to shoot the box towards the goal from larger distances with higher precision. When the action space is discretised, either not enough actions are available to perform the moves that are required, or the search tree branches too far to be explored sufficiently in the limited time that available for planning. It seems that the enough/not too many actions trade-off is met best by the variant using 36 actions. This, however, is a problem specific number and even in this case ABT is out-performed by all tested variations of GPS-ABT.

Algorithm	$\|A\|$	c_{depth}	ε_{tag}	mean reward	99% confidence
ABT	3+1		0.5	-14.3	$\pm$ 0.5
ABT	4+1		0.5	-9.3	$\pm$ 0.4
ABT	6+1		0.5	-9.9	$\pm$ 0.3
ABT	10+1		0.5	-9.5	$\pm$ 0.4
ABT	20+1		0.5	-13.7	$\pm$ 0.4
ABT	50+1		0.5	-14.4	$\pm$ 0.5
GPS-ABT		4	0.5	-9.3	$\pm$ 0.5
GPS-ABT		5	0.5	-9.2	$\pm$ 0.5
GPS-ABT		6	0.5	-9.1	$\pm$ 0.6
GPS-ABT		7	0.5	-8.2	$\pm$ 0.6

TABLE II: Simulation results for the Tag problem. Normal ABT discretises the action space A into finitely many actions plus the TAG action.

B. Continuous Tag

The tag problem is a widely used benchmark problem for POMDP solvers. The usual version has, however, a discrete action space A with only 5 actions. Thus, a continuous version was derived to test the performance overhead required for GPS-ABT in relation to ABT with discretised actions. The objective for the robot is to get itself close enough to a moving target and then 'tag' the target.

The state space S contains four dimensions to encode the robot's position (x_r, y_r) as well as the target's position (x_t, y_t). The action space $A = [0, 360) \cup \{TAG\}$ consists of all angular directions the robot can move into and an additional *TAG* action. In each step the robot moves 1 unit length into the chosen direction while the target moves away from the robot. The target moves 1 unit length into the direction away from the robot but varies its direction within a range of 45°at random. The robot also has an unreliable sensor to detect the target: if the target is within $\pm 90°$ the target is detected with probability $p = 1 - \frac{\alpha}{90°}$. The observation received is binary an only assumes the values 'detected' and 'not detected'. Instead of moving the robot can decide to attempt to TAG the target. If the distance between the robot and the target is smaller ε_{tag}, a reward of 10 is received and the simulation ends, otherwise a penalty of -10 is received and the simulation continues. In addition, each move receives a penalty of -1. The environment map used for the simulations is depicted in Fig. 3.

It can be seen from the results that while GPS-ABT does not have a big advantage over ABT with discretised actions, it is also not penalised for its overhead and non-exhaustive search of the action space. In addition, with GPS-ABT it is not necessary to fix a problem specific discretisation resolution to balance the trade-off between a sufficient amount actions and a small enough action space to avoid excessive branching.

VII. SUMMARY

The past decade has seen tremendous advances in the capability of POMDP solvers, moving the practicality of POMDP-based planning far beyond robot navigation in a small 2D grid world, into mid-air collision avoidance of commercial aircraft (TCAS) [22], grasping [10], and non-prehensile manipulation [7], to name but a few examples. Despite such tremendous advancement, most POMDP solvers can only solve problems with a small and discrete set of actions. Most successful solvers rely on sampling to estimate the mean total utility and enumerate over all possible actions to find the action that maximizes the mean value, which is infeasible when the action space is continuous.

This paper presents GPS-ABT, an approximate and online POMDP solver for problems with continuous action spaces, assuming the action space has a metric space. It interleaves Generalized Pattern Search with ABT to find the best actions among the set of candidate actions and estimate the Q values. When $Q(b,.)$ is continuous with a single maximum at each belief $b \in \mathcal{R}^*(b_0)$, GPS-ABT converges to the optimal solution in probability. Otherwise, GPS-ABT converges to the local optimal solution in probability.

References

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, May 2002.

[2] H. Bai, D. Hsu, W. Lee, and A. Ngo. Monte Carlo Value Iteration for Continuous-State POMDPs. In *WAFR*, 2010.

[3] H. Bai, D. Hsu, and W. S. Lee. Integrated perception and planning in the continuous space: A POMDP approach. In *RSS*, Berlin, Germany, June 2013.

[4] J. Berg, S. Patil, and R. Alterovitz. Efficient approximate value iteration for continuous gaussian POMDPs. In *AAAI*, 2012.

[5] E. Brunskill, L. Kaelbling, T. Lozano-Pérez, and N. Roy. Continuous-state POMDPs with hybrid dynamics. In *International Symposium on Artificial Intelligence and Mathematics*, 2008.

[6] I. Griva, S. Nash, and A. Sofer. *Linear and Nonlinear Optimization: Second Edition*. Society for Industrial and Applied Mathematics, 2009.

[7] R. He, E. Brunskill, and N. Roy. PUMA: planning under uncertainty with macro-actions. In *AAAI*, 2010.

[8] J. Hoey and P. Poupart. Solving POMDPs with continuous or large discrete observation spaces. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI, pages 1332–1338, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[9] M. Horowitz and J. Burdick. Interactive Non-Prehensile Manipulation for Grasping Via POMDPs. In *ICRA*, 2013.

[10] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *AI*, 101:99–134, 1998.

[11] D. Klimenko and J. S. and. H. Kurniawati. TAPIR: A software toolkit for approximating and adapting POMDP solutions online. Submitted to ACRA2014, `http://robotics.itee.uq.edu.au/~hannakur/papers/tapir.pdf`.

[12] L. Kocsis and C. Szepesvri. Bandit based monte-carlo planning. In *In: ECML-06. Number 4212 in LNCS*, pages 282–293. Springer, 2006.

[13] M. Koval, N. Pollard, and S. Srinivasa. Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. In *RSS*, Berkeley, USA, July 2014.

[14] H. Kurniawati, T. Bandyopadhyay, and N. Patrikalakis. Global motion planning under uncertain motion, sensing, and environment map. *Autonomous Robots: Special issue on RSS 2011*, 30(3), 2012.

[15] H. Kurniawati and V. Yadav. An online POMDP solver for uncertainty planning in dynamic environment. In *ISRR*, 2013.

[16] A. Y. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 406–415. Morgan Kaufmann Publishers Inc., 2000.

[17] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, pages 1025–1032, 2003.

[18] R. Platt, R. Tedrake, T. Lozano-Perez, and L. Kaelbling. Belief space planning assuming maximum likelihood observations. In *RSS*, 2010.

[19] J. Porta, N. Vlassis, M. Spaan, and P. Poupart. Point-Based Value Iteration for Continuous POMDPs. *JMLR*, 7(Nov):2329–2367, 2006.

[20] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online planning algorithms for POMDPs. *JAIR*, 32:663–704, 2008.

[21] D. Silver and J. Veness. Monte-Carlo Planning in Large POMDPs. In *NIPS*, 2010.

[22] R. Smallwood and E. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[23] T. Smith and R. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *UAI*, July 2005.

[24] A. Somani, N. Ye, D. Hsu, and W. S. Lee. DESPOT: Online POMDP planning with regularization. In *NIPS*, pages 1772–1780, 2013.

[25] S. Temizer, M. J. Kochenderfer, L. P. Kaelbling, T. Lozano-Pérez, and J. K. Kuchar. Collision avoidance for unmanned aircraft using markov decision processes. In *Proc. AIAA Guidance, Navigation, and Control Conference*, 2010.

Appendix

A. Golden Section Search

The Golden Section Search method is applicable for uni-modal, one-dimensional objective functions f. The stencil of Golden Section Search contains only two points, thus the stencil in step i has the form $\mathfrak{s}_i = \{a_i, b_i\}$.

Given a search interval $(l,u) \subseteq \mathbb{R}$, the initial stencil is set to $a_0 = \phi l + (1-\phi)$ and $b_i = (1-\phi)l + \phi u$, where ϕ denotes the inverse of the golden ratio: $\phi = \frac{\sqrt{5}-1}{2}$.

New stencils are created by setting $a_{i+1} = a_i$ and $b_{i+1} = (1+\phi)a_i - \phi b_i$ if $f(a_i) > f(b_i)$, otherwise it is set as $a_{i+1} = (1+\phi)b_i - \phi a_i$ and $b_{i+1} = b_i$.

Effectively the stencil splits the search interval into three sub-intervals. During each step one of the two outer intervals is identified as a region that cannot contain the optimum and is eliminated. A new point is added to split the remaining intervals into three parts again and the process repeats. Convergence of the method follows directly from the assumption that f is uni-modal.

B. Compass Search

Compass search is a GPS method that is applicable to multi-dimensional spaces. The stencil used by compass search contains $2n+1$ points where n is the number of dimensions of the search space. The stencil is defined by its center point p and a radius r. The remaining $2n$ points of the stencil have the form

$$p \pm r e_d \tag{17}$$

for $d = 1, ..., n$ where e_d is the n-dimensional vector with 1 at the d-th position and 0 otherwise, i.e.

$$e_1 = (1,0,0,...)\,, \quad e_2 = (0,1,0,...)\,, \; \dots\,. \tag{18}$$

The stencil thus forms a shape similar to a compass rose in two dimensions which give the search algorithm its name.

The search progresses by evaluating the objective function for all $2n+1$ point in the stencil $\mathfrak{s}_i$. The point that yields the largest value of the objective function becomes the new center point $p_{i+1} \in \mathfrak{s}_{i+1}$. If $p_{i+1} = p_i$, the radius halved to produce $r_{i+1} = 0.5r$, otherwise $r_{i+1} = r$. The search continues until a convergence criterion is met.

Compass Search is known to converge to the optimal solution if the following assumptions are met:

1) the set $\{x : f(x) \leq f(p_0)\}$ is bounded.
2) ∇f is Lipschitz continuous for all x, that is,

$$|\nabla f(x) - \nabla f(y)| \leq L|x-y| \tag{19}$$

for some constant $0 < L < \infty$.

A proof for convergence under these conditions is presented in [6, chapter 12.5.3].