Journal of Artificial Intelligence Research () Submitted ; published

Efficient Planning under Uncertainty with Macro-actions

Ruijie He RUIJIE@MIT.EDU
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139 USA

Emma Brunskill EMMA@CS.BERKELEY.EDU
Electrical Engineering and Computer Science Department
University of California, Berkeley
Berkeley, CA 94709 USA

Nicholas Roy NICKROY@MIT.EDU
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139 USA

Abstract

A wide variety of artificial intelligence systems require the ability to perform planning under uncertainty in a partially observable domain. Such planning is particularly challenging when good control performance requires considering situations far into the future. In this paper, we propose the use of macro-actions in forward search planning to handle such problems. By using macro-actions consisting of fixed-length open-loop policies, we restrict the policy class considered during planning in return for significant computational gains that allow us to perform much longer-horizon forward search. In a certain subset of domains, we also show that it is possible to analytically compute the distribution over posterior beliefs that results from a single macro-action; this distribution captures any observation sequence that could occur during the macro-action, and allows significant additional computational savings. We present a formal analysis of our approach, and demonstrate performance on two simulation experiments: a scientific exploration domain and a target tracking and monitoring domain. We also demonstrate our algorithm with a real robotic helicopter in a target monitoring experiment. This suggests that our approach has practical potential for planning in real-world, long-horizon, partially observable domains.

1. Introduction

Consider an autonomous agent tasked with monitoring the locations of a number of targets. At each time step, the agent must report whether or not each target is inside an area of interest. Due to limited sensing capability, the agent cannot observe the whole environment at once, but instead must move to different locations to observe different regions. The agent is rewarded when it correctly reports that a target is inside one of the key regions, penalized for a wrong report, and gets no reward otherwise. The agent must therefore decide how to move in the environment and what reports to make at each time step.

This problem requires decision-making in an uncertain, partially observable domain, which is a common challenge for artificial agents operating in real-world environments. In addition, this target monitoring problem is particularly challenging because it requires the agent to plan far ahead in order to select good actions. If a target is far away, for example, the agent may have to travel a long distance before it can get information about that target's location. Beyond target monitoring,

there are numerous other problems, such as scientific exploration of extreme environments and autonomous management of retirement portfolios, which require far-lookahead decision-making strategies in partially observable, stochastic environments.

Researchers from multiple fields have sought to address specific instances of the problem of planning under uncertainty. However, most of this work has focused on generic planning under uncertainty, with less attention to problems that specifically require a large amount of foresight to select good actions. For the special case of quadratic costs and linear Gaussian dynamics and sensor models, an optimal infinite-horizon analytic strategy exists, known as the Linear Quadratic Gaussian (LQG) controller (Athans, 1971). The more generically applicable Model Predictive Control/Receding Horizon Control (MPC/RHC) framework from the controls community (Mayne et al., 2000; Kuwata & How, 2004) involves planning over a fixed, short horizon, and then replanning after every action. The sensor resource management community analyzes a similar problem for tracking multiple targets; recently this community has mostly focused on the interesting subclass of problems in which selecting the action to gain the most immediate reward is close to optimal (Krause & Guestrin, 2007). Finally, when the environment and dynamics consist of a discrete set of possible values, many popular approaches within the Partially Observable Markov Decision Process (POMDP) framework seek to compute a (near) globally optimal infinite-horizon control policy in advance of acting (Kaelbling et al., 1998)

Unfortunately, planning techniques of all three research communities typically scale poorly with the horizon length, and are unable to search out to longer horizons for a broad class of sensor, measurement and cost/reward models. MPC/RHC approaches typically optimize policies for a short, finite horizon, and use a heuristic to estimate the cost beyond the fixed horizon. Many POMDP planners construct a set of conditional policies that specify which action an agent should take at each future step, conditioned on the prior actions taken and observations received. The space of conditional plans grows in a double exponential fashion with the number of time steps the agent must look ahead in order to select the best action. This makes planning for long-horizon problems challenging.

In this paper we approach this challenge by restricting the policy space, which allows us to consider a subset of policies out to a longer horizon, instead of considering the full set of possible policies, which can typically be done only for short horizons. Our core contribution is an algorithm for efficiently evaluating the expected reward of policies within a restricted policy space. We restrict the policy space by using macro-actions (Sutton et al., 1999), which we define as fixed-length, open-loop policies: in other words, a macro-action is a finite sequence of primitive actions that is executed without regard to the observations received during the execution of this action sequence. For applicability to partially observable worlds, our macro-action definition is more restrictive than the broader class of macro-actions, also known as options, that is used within the reinforcement learning community. In our target monitoring problem, one macro-action could be for the agent to travel to a key region, which might involve a sequence of individual turns and straight line moves. Macro-actions significantly reduce the computational complexity of planning, since a set of M macro-actions of length L can replace the full set of $|\mathcal{A}|^L$ primitive action sequences, where $|\mathcal{A}|$ is the number of primitive actions. In addition, macro-actions can often be easily identified for a given problem, and hence allow expert domain knowledge to be incorporated naturally.

To our knowledge, using macro-actions is under-explored in the planning under uncertainty literature. Prior work on using macro-actions in the POMDP literature has focused on offline approaches that compute policies for a wide variety of potential beliefs, but do not offer solutions

for handling the wide number of potential future observation trajectories that are still possible in a reduced policy space (Theocharous & Kaelbling, 2003; Yu et al., 2005; Kurniawati et al., 2009). From the sensor management resource literature, Scott et al. (2009) used a maximum likelihood approach to compute the most probable result of a macro-action, ignoring other potential futures. The approach we present in this paper is related to work in the receding horizon control framework that used a geometric set of fixed macro-actions to approximate the control beyond a short horizon (Kuwata & How, 2004; Bellingham et al., 2002); however, in the receding horizon control framework, the initial control was still done using primitive actions, and the macro-actions themselves were not adapted to the particular current beliefs.

We adopt a forward search framework (Ross et al., 2008) for performing long-horizon planning in partially observable domains with macro-actions. Forward search approaches direct computational effort only towards belief states that are reachable from the current belief under different actions. If acting well requires considering potential states H-steps into the future, the resulting forward search tree has on the order of $(|\mathcal{A}||\mathcal{Z}|)^H$ nodes, where $|\mathcal{Z}|$ is the number of observations considered after each action. Although the computational cost still scales exponentially with the horizon, forward search provides a significant reduction compared to the standard doubly exponential dependence of a full policy computation. Forward search approaches can also leverage independence properties, such as factored dynamics and sensor models (Paquet et al., 2005). Factored models often allow belief filtering, the process of incorporating new information into the belief state, to be performed much more efficiently. Forward search can also be used easily for reward models which are either a function of the underlying world state, or a direct function of the belief state.

Using macro-actions inside a forward search planner, the branching factor due to action selection may be substantially reduced. However, at each time step, the agent will still receive one of a large set of possible observations of the different targets' locations. The branching factor of the search due to observations remains an exponential function of the horizon length and the number of primitive actions. In general, the number of potential observations associated with a L-length macro-action is $|\mathcal{Z}|^L$, where $|\mathcal{Z}|$ is the number of possible observations. While this computational cost can be reduced by sampling observation sequences, the number of samples necessary for a good estimate of the expected reward is substantial, and the cost of belief updating per observation sequence can be expensive.

In this paper, we propose two novel algorithms that exploit the structure inherent in specific types of environments in order to perform efficient macro-action forward search. One of our core contributions is an analytic method for computing the distribution over the set of posterior beliefs that could result from a single macro-action, so long as the agent's belief of the world can be represented as a Gaussian distribution (Section 3.3). This posterior belief distribution represents the beliefs that could result given all possible observation sequences that the agent may receive during the macro-action, and therefore removes the need to explicitly enumerate the observation sequences. For a large class of reward models, this distribution of posterior beliefs also allows us to analytically compute the expected reward associated with a macro-action. These techniques form the core of our Posterior Belief Distribution (PBD) algorithm (Section 3.4), which uses a parametric representation of the agent's belief to plan efficiently in partially observable domains, which is especially helpful when it is necessary to look further into the future to choose good actions to take next. The computational complexity of computing the posterior belief distribution scales linearly with the length of the macro-action and cubicly with the number of state-space dimensions, which is often exponentially smaller than the size of the corresponding discrete state space.

When the belief updating and reward calculation cannot be done analytically, we also provide techniques that allow an approximate computation of the posterior belief distribution and expected rewards (Section 3.5). In problem domains that are not well-captured by the parametric assumptions of the PBD algorithm, such as discrete environments, we cannot analytically capture the resulting distribution of beliefs after a macro-action. Instead, we sample observation sequences to produce a particle approximation of the resulting distribution over beliefs: we call this a macro-action Discrete (MAD) forward search planner (Section 3.1). When the underlying environmental models exhibit a factored structure, MAD can be used to perform efficient macro-action forward search, and it significantly outperforms prior approaches on a simulated long-horizon problem.

A formal analysis of both the PBD and MAD algorithms is presented in which we analyze their performance and computational complexity (Section 4). This analysis shows that relative to alternative approaches, PBD possesses a significant reduction in computational complexity due to the use of an analytic expression for the resulting distribution over beliefs (Section 4.3). We also demonstrate that in domains with independent state variables, MAD has a substantially lower computational complexity than a fully joint state representation (Section 4.3.7).

Experimental results highlight the promise of our macro-action planners in problem domains that require long-horizon planning, where past approaches often struggle. We present results on variants of the rocksample POMDP benchmark problem (Smith & Simmons, 2004). and demonstrate that on problems that require a longer-horizon search, our algorithms significantly outperform past approaches (Section 5.1). We also compare the algorithms on our running example of a single-agent, multiple-target monitoring problem, which is related to target tracking problems popular in the sensor resource management community (Section 5.2). Here the problem is naturally represented using continuous-valued states, and the PBD algorithm significantly outperforms other alternate approaches when it is necessary to plan to a long horizon and explicitly reason about the agent's possible beliefs many steps in the future. Finally, we demonstrate the PBD algorithm on a real-world version of the target-monitoring problem, where we use a robotic helicopter platform to monitor multiple ground vehicles (Section 5.4). This demonstration shows that the PBD representations can capture the dynamics and sensor models of real robotic systems, and suggests that our analytic approach of calculating the posterior belief distribution has practical promise for planning in long-horizon planning under uncertainty domains.

2. Background: Planning under uncertainty using forward search

Each research community uses slightly different problem formalisms for planning in partially observable, uncertain environments, and ours follows most closely from the discrete-time Markov decision process framework. Formally, we assume that our decision-making under state-uncertainty problem consists of the following known components:

- $\mathcal{S}$ is a set of states $s \in \mathcal{S}$, which can be either discrete or continuous
- $\mathcal{A}$ is a set of actions (controls) $a \in \mathcal{A}$, which can be either discrete or continuous
- $\mathcal{Z}$ is a set of observations $z \in \mathcal{Z}$, which can be either discrete or continuous
- $p(s'|s, a)$ is a transition function (also known as a dynamics model) which encodes the probability of transitioning to state s' after taking action a from state s. In a model with discrete

states, this is specified by a multinomial distribution. In a linear Gaussian model, this is specified by a linear function with additive Gaussian noise.

- $p(z|s', a)$ is an observation function (also known as a measurement or sensor model) that encodes the probability of receiving observation z after taking action a and transitioning to state s'. This function is specified in a manner similar to the transition function.
- $r(b, a)$ is a reward (or cost) function that describes the utility the agent receives for taking action a when it has a belief b. If the reward is expressed in terms of the state, then $r(b, a)$ is an expectation over the state.
- γ is a discount factor that determines the relative weights of immediate utilities to utilities that will be received at a later time step.

Note that the model described above is slightly more general than in either the classical control or the POMDP frameworks, relaxing the assumption that the reward model can only be a direct function of the state, rather than a function of the beliefs.

The states S are not fully observable. Instead, at every time step, the agent receives an observation after taking an action. The agent must therefore make decisions based on the prior history of observations it has received, $z_{1:t}$, and actions it has taken, $a_{1:t}$, up to time t. As the world states are Markov, instead of maintaining an ever-expanding list of past observations and actions, a sufficient statistic, known as a belief $b_t(s)$, is used to summarize the probability of the world being in each state given its past history and initial belief b_0.

$$b_t(s) = Pr(s_t = s|a_0, z_1, \ldots, z_{t-1}, a_{t-1}, z_t, b_0) \tag{1}$$

The agent can therefore plan based only on the current belief state, rather than on all past actions and observations (Smallwood & Sondik, 1973). For example, in the target monitoring problem introduced in Section 1, the agent maintains a belief over the possible locations of each target. The agent updates its belief at each step, after taking an action a and receiving an observation z (such as a camera image of a far off target), using the Bayes filter:

$$b'(s') = \tau(b, a, z) = \eta\, p(z|a, s') \int_{s \in \mathcal{S}} p(s'|s, a) b(s) ds \tag{2}$$

where $\tau(b, a, z)$ represents the belief update function and η is a normalization constant.

The optimization problem is to compute a policy $\pi : b \rightarrow a$, which is a mapping from belief states to actions, that maximizes the expected sum of discounted utilities over H time steps:

$$\pi = \text{argmax} \left[\sum_{i=1}^{H} \gamma^i E[r(b_i)] \right] \tag{3}$$

where $E[r(b_i)]$ denotes the expectated reward at time step i given the actions specified by π and possible observations received. Each policy π is also associated with a value function $V_\pi : b \rightarrow \mathcal{R}$, specifying the expected total reward of executing policy π starting from b

$$V_\pi(b) = \max_{a \in \mathcal{A}} \big[r(b, a) + \gamma \sum_{z \in \mathcal{Z}} p(z|b, a) V_\pi(\tau(b, a, z)) \big], \tag{4}$$

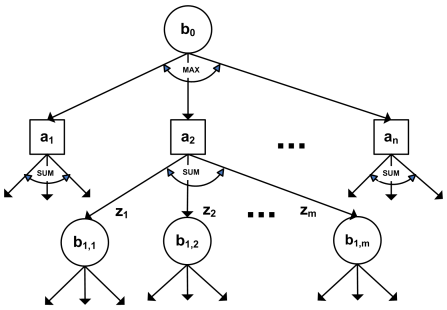


Figure 1: A forward search tree. a are actions, z are observations, and b are beliefs. b_0 is the initial belief, while $b_{i,j}$ refers to the jth belief leaf node at depth i.

Algorithm 1 Forward Search with Macro-Actions

Require: Initial belief b_0, Discount factor γ, Macro-action search depth $\tilde{H}$, Sampling parameter N_s

$t \leftarrow 0$
loop
  Compute set of action seqeunces $\tilde{A}$
  for $\tilde{a}_i \in \tilde{A}$ **do**
    $Q(b_t, \tilde{a}_i) = \textsc{Expand}(\tilde{a}_i, b_t, \gamma, \tilde{H}, N_s)$ {See Algorithm 2-4}
  end for
  Execute first action a_1 of $\tilde{a} = \text{argmax}_{\tilde{a}}\, Q(b_t, \tilde{a})$
  Obtain new observation z_t and reward r_t
  $b_{t+1} = \tau(b_t, a_t, z_t)$
  $t \leftarrow t + 1$
end loop

where $p(z|b,a) = \int_s p(z|s,a)b(s)ds$.

Another useful quantity is the Q-value, which represents the value of executing action a from b

$$Q_\pi(b,a) = r(b,a) + \gamma \sum_{z \in \mathcal{Z}} p(z|b,a) V_\pi(\tau(b,a,z)), \tag{5}$$

Forward search techniques (Ross et al., 2008) direct computational effort only towards belief states that are reachable from the current belief under different actions. In the target monitoring problem, this property enables a forward search planner to compute a meaningful policy in an arbitrarily large environment, since only a subset of the environment is relevant at any point. Forward search techniques alternate between a planning and execution phase, planning online only for the belief at the current time step. During the planning phase, a forward search algorithm creates an AND-OR tree (Figure 1) of reachable belief states from the current belief state. The tree is expanded using action-observation pairs that are admissible from the current belief, and the beliefs at the internal nodes are found using Equation 2. The reward of actions at the root node are found by propagating the belief rewards from the leaf nodes back to the root, repeatedly taking the sum over observation nodes, and a maximization over action nodes. Intuitively, the expected rewards are computed in this manner because the agent can choose which action to take, but must optimize over the expected distribution of observations. A nice feature of forward search is that it is agnostic to the belief representation used, and different representations can even be intermixed as long as belief updates can be performed and expected rewards computed.

After the planning phase, the forward search procedure executes the action at the root with the largest value, and then receives an observation. The forward search planning process is then started again, with the new belief as the root node. Replanning after every time step enables the agent to incorporate and condition on the action selected and observation receieved. This action-selection approach is also known as open-loop feedback control (Bertsekas, 2000), which forms the basis of many control methods, including Model Predictive Control.

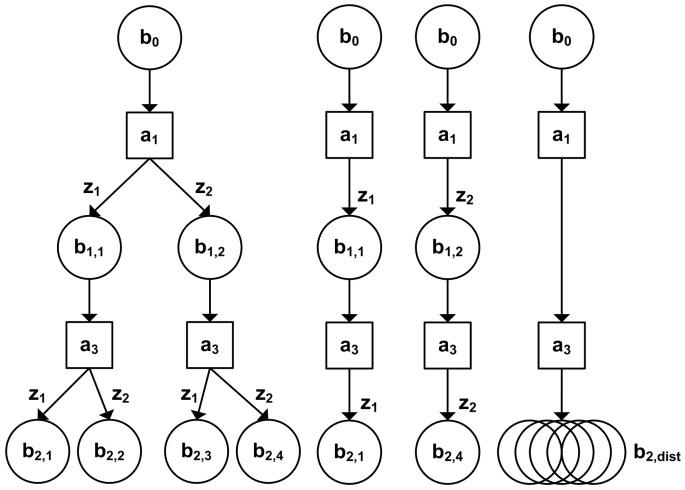
3. Using macro-actions in forward search

To obtain a set of reachable beliefs, naïve forward search algorithms consider all possible action-observation sequences at a specific depth. The computational cost of this procedure scales as $\mathcal{O}((|\mathcal{A}||\mathcal{Z}|)^H)$ where H is the primitive-action forward search planning horizon, or depth of the search. Unfortunately, reasonably large discrete or continuous action and observation sets (large $|\mathcal{A}|$ or $|\mathcal{Z}|$, respectively) severely limit the maximum search depth achievable in real-time.

By restricting our action space to a set of length L macro-actions, we can reduce the computational complexity component $|\mathcal{A}|^H$ to $|\tilde{\mathcal{A}}|^{\tilde{H}}$ where $\tilde{\mathcal{A}}$ is the set of length L macro-actions, $M = |\tilde{\mathcal{A}}|$ is the number of macro-actions, and $\tilde{H} = \frac{H}{L}$ is the macro-action horizon. In general restricting the action space will result in significant computational savings. Algorithm 1 presents the general framework for using macro-actions for planning under uncertainty with forward search. Based on its current belief at every time step, the planner first generates a number of macro-actions that the agent can execute from its current belief. The planner then performs a rollout of each of the macro-actions through the EXPAND subroutine, in order to obtain the expected reward of taking each macro-action from the current belief. The best action sequence $\tilde{a}$ is chosen, and the first action a_1 of this action sequence is executed. The agent then updates its current belief (Equation 2) based on the action taken and observation obtained, and the cycle repeats. Replanning after each time step is particularly advantageous in the context of macro-actions, as it allows the agent to consider a wider range of policies than allowed by the macro-actions, and allows the plan to condition on observations in a manner that it might not during planning (as we will see shortly). Planning and execution interleave until a termination condition is reached. Possible termination conditions include reaching a maximum number of execution steps in the environment, or receiving an observation that indicates that the agent should stop acting. The EXPAND subroutine has significant impact on the performance and computational complexity of the forward search techniques, and is the main focus of this paper.

3.1 Sampling observation sequences

Since there are still $|\mathcal{Z}|$ potential observations for each primitive action taken within a macro-action (see Figure 2(a)), the total number of potential observations for a sequence of $\tilde{H}$, L-length macro-actions is still the same: $|\mathcal{Z}^{\tilde{H}L}| = |\mathcal{Z}^H|$. Therefore, using macro-actions alone does not in itself break the exponential growth in computational cost as the horizon length increases, though it does reduce the computational complexity from $\mathcal{O}((|\mathcal{A}||\mathcal{Z}|)^H)$ to $\mathcal{O}(|\tilde{\mathcal{A}}|^{\tilde{H}}|\mathcal{Z}|^H)$. A simple way to reduce the impact of the dependence on the number of potential observations is to sample observation sequences for each L-length macro-action, instead of enumerating all $O(|\mathcal{Z}|^L)$ possible observations that could be received during a single macro-action: compare Figures 2(a) and 2(b) for an illustration. Sampling observation sequences depends on the assumption that for most real-world problems, some observation sequences are much less likely than others, and hence has little impact on the evaluation of the policy. For each observation sequence, the planner alternates between a transition update based on the macro-action $\tilde{a}$, sampling an observation based on the current belief, and updating the belief based on the sampled observation. N_z trajectories are sampled for the same macro-action, and each of these posterior belief samples are used to generate a new set of macro-actions. The planner performs a rollout of each of the macro-actions to determine the subsequent macro-action with the largest expected reward. The process repeats up to a search depth of $\tilde{H}$. Algorithm 2 depicts the EXPAND() routine when posterior beliefs are obtained by sampling observation sequences.



(a) Exhaustive (b) Sampled (c) Analytic

Figure 2: Three methods to represent the resulting set of beliefs after a single macro-action. (a) All possible observations are expanded. (b) A subset of possible observation trajectories are sampled. (c) Compute an analytic distribution over the posterior beliefs, which could have been generated via an exhaustive enumeration of all possible observation sequences. b_0 is the initial belief, while $b_{i,j}$ refers to the jth belief leaf node at depth i.

When the agent's belief is represented as a belief over a discrete set of states, we refer to this approach of sampling observation sequences as the macro-action discrete-state (MAD) forward search planning algorithm. Over a continuous state space, we refer to the observation-sampling forward search algorithm as macro-action continuous-state (MAC). Note that belief updating must be performed at each step of the observation sequence for each sampled observation sequence. In general, performing this number of belief updates can be computationally expensive. However, in factored discrete-state domains, belief updates can be performed very quickly. We will demonstrate in Section 5 that MAD can significantly outperform prior approaches in factored domains which require long-horizon planning.

3.2 Computing the posterior belief distribution directly

As we mentioned at the start of the prior subsection, for a single L-length macro-action, there are still $|\mathcal{Z}|^L$ potential observation sequences that could be received during that macro-action. If we exhaustively enumerated all potential observation sequences, and performed belief updating along each sequence, that would yield a set of $|\mathcal{Z}|^L$ possible posterior beliefs after a single macro-action. When considering the immediate expected reward from a macro-action, or the potential future reward after taking a macro-action, we must remember that these rewards are a function of the set of potential posterior beliefs resulting from a macro-action, rather than being a function of a single belief.

A natural question then is how to obtain the set of posterior beliefs that could result after a single macro-action. Exhaustive enumeration of observation sequences will be intractable when

Algorithm 2 EXPAND() (Sampling observation sequences)

Input: Macro-action $\tilde{a}$, Belief state b_t, Discount factor γ, Macro-action search depth $\tilde{H}$, No. sampled observation trajectories N_z
if $\tilde{H} = 0$ **then**
  return 0
else {Expand Macro-Action $\tilde{a}$=$\{a_1, \ldots, a_L\}$}
  $V = 0$
  for $i = 1$ to N_z **do**
    $V_i = 0$
    for $j = 1$ to L **do**
      $V_i = V_i + \gamma^j r(b_t, a_j)$
      Perform transition update $b_j^a = \int_{s \in S} p(s'|s,a) b_{j-1}(s) ds$
      Sample observation z_j based on belief b_j^a
      Perform observation update $b_j(s') = \eta p(z_j|s',a) b_j^a(s')$
    end for
    Generate next set of action sequences $\tilde{A}^{next}$
    for $\tilde{a}_i^{next} \in \tilde{A}^{next}$ **do**
      $Q(b_i, \tilde{a}_i^{next})$ = EXPAND($\tilde{a}_i^{next}$,b_i,γ,$\tilde{H} - 1$,N_z)
    end for
    $V = V + \frac{1}{N_z}(V_i + \gamma^L \operatorname{argmax}_{\tilde{a}_i^{next}} Q(b_i, \tilde{a}_i^{next}))$
  end for
  return V
end if

there are a large number of observations, or when the macro-action itself consists of a long list of primitive actions. In the prior subsection we presented one alternative, which is to instead sample observation sequences for a given macro-action. Sampling observation trajectories yields a particle approximation of the true posterior distribution of beliefs. However, as we will discuss further in Section 4.3, sampling can still be computationally intensive, due to the number of belief updates and expected reward calculations that must occur at each step of each sampled observation sequence.

We now show that for specific kinds of world models, we can analytically compute the distribution over beliefs that arise from a macro-action without sampling observations: a graphical depiction of this process is shown in Figure 2(c). By analytically computing a distribution over beliefs, we completely avoid both the exponential computational dependence on the original horizon length, and also the costly step of performing belief updating along each observation trajectory associated with the macro-action.

Specifically, when the agent's belief is a Gaussian, then the distribution over posterior beliefs is itself a Gaussian over Gaussian beliefs, as illustrated in Figure 3. We will show that when the Gaussian belief update is exact, all posterior beliefs in this distribution have the same covariance, and the posterior means are normally distributed over the continuous state space. Given an action sequence, the posterior distribution over beliefs is therefore a joint distribution over the posterior means and the corresponding distribution over states. As we will discuss later, the computational complexity of forward search is reduced to being an efficient function of the macro-action horizon

$\tilde{H}$: for macro-actions of length 2 or more ($L \geq 2$) this means that it is significantly faster to search to long horizons.

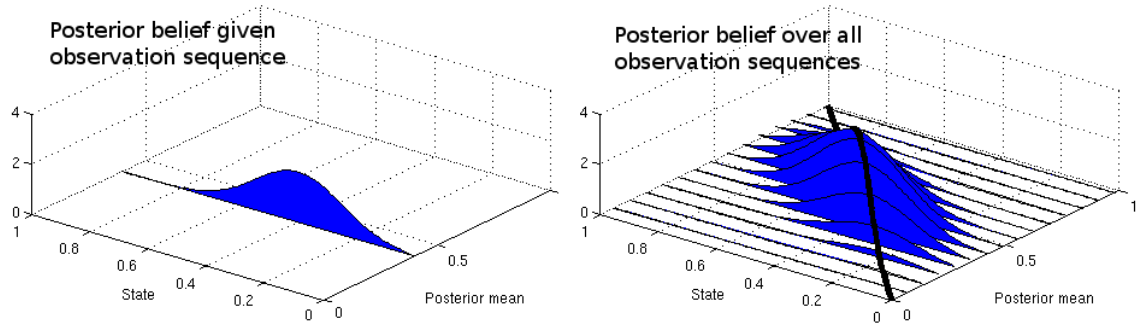


Figure 3: Distribution of posterior beliefs. a) A Gaussian posterior belief results after incorporating an observation sequence. b) Over all possible observation sequences, the distribution of posterior means is a Gaussian (black line), and for each posterior mean, a Gaussian (blue curve) describes the agent's posterior belief.

3.3 Exact computation of Posterior Belief Distribution

Let us assume for now that the agent's belief can be exactly represented as a Gaussian distribution over a continuous state space, and that the observation and transition models are both linear-Gaussian. Formally, the state transition and observation models can be represented as follows:

$$s_t = A_t s_{t-1} + B_t u_t + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, P_t) \tag{6}$$

$$z_t = C_t s_t + \delta_t, \qquad \delta_t \sim \mathcal{N}(0, Q_t) \tag{7}$$

where A_t and B_t are dynamics matrices, C_t is the observation matrix, P_t is the covariance of the Gaussian dynamics process and Q_t is the covariance of the measurement noise.

When the state-transition and observation models are normally distributed and linear functions of the state, the Kalman filter (Kalman, 1960) provides a closed-form solution for the posterior belief $\mathcal{N}(\mu_t, \Sigma_t)$ given a prior belief $\mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$,

$$\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t \qquad \mu_t = \overline{\mu}_t + K_t(z_t - C_t \overline{\mu}_t) \tag{8}$$

$$\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + P_t \qquad \Sigma_t = (C_t^T Q_t^{-1} C_t + \overline{\Sigma}_t^{-1})^{-1}, \tag{9}$$

where $\mathcal{N}(f, F)$ is a D-dimensional Gaussian with mean f and covariance matrix F,
$K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1}$ is the Kalman gain and $\overline{\mu}_t$ and $\overline{\Sigma}_t$ are the mean and covariance after an action is taken but before incorporating the measurement.

We are interested in computing an expression for the distribution over the mean of the posterior belief under any possible observation. To do this, we first re-express the observation model as

$$p(z_t | s_t) \sim \mathcal{N}(C_t s_t, Q_t) \tag{10}$$

which we can use to compute an expression for the probability of an observation given the belief mean, $p(z_t|\overline{\mu}_t)$, by marginalizing over $s_t \sim \mathcal{N}(\overline{\mu}_t, \overline{\Sigma}_t)$:

$$\begin{aligned} p(z_t|\overline{\mu}_t) &= \int p(z_t|s_t, \overline{\mu}_t) p(s_t|\overline{\mu}_t) ds_t. && (11) \\ &\sim \mathcal{N}(C_t\overline{\mu}_t, C_t\overline{\Sigma}_t C_t^T + Q_t) && (12) \end{aligned}$$

Using this expression, and performing linear transformations, we can obtain an expression for the distribution of posterior means, under any potential observation:

$$\begin{aligned} p(z_t|\overline{\mu_t}) &\sim \mathcal{N}(C_t\overline{\mu}_t, C_t\overline{\Sigma}_t C_t^T + Q_t) && (13) \\ p(z_t - C_t\overline{\mu}_t|\overline{\mu_t}) &\sim \mathcal{N}(0, C_t\overline{\Sigma}_t C_t^T + Q_t) && (14) \\ p(K_t(z_t - C_t\overline{\mu}_t)|\overline{\mu_t}) &\sim \mathcal{N}(0, K_t(C_t\overline{\Sigma}_t C_t^T + Q_t)K_t^T) && (15) \\ p(\overline{\mu}_t + K_t(z_t - C_t\overline{\mu}_t)|\overline{\mu}_t) &\sim \mathcal{N}(\overline{\mu}_t, K_t(C_t\overline{\Sigma}_t C_t^T + Q_t)K_t^T) && (16) \\ p(\mu_t|\overline{\mu}_t) &\sim \mathcal{N}(\overline{\mu}_t, K_t(C_t\overline{\Sigma}_t C_t^T + Q_t)K_t^T) && (17) \\ p(\mu_t|\overline{\mu}_t) &\sim \mathcal{N}(\overline{\mu}_t, \overline{\Sigma}_t C_t K_t^T) && (18) \end{aligned}$$

where Equation 18 is computed by substituting the definition of the Kalman gain.

Equation 18 shows that the distribution of posterior belief means under any observation is independent of any particular observation. Sampling a mean from this distribution is equivalent to selecting a particular observation. The distribution is normally distributed about $\overline{\mu}_t$, with a covariance that depends on the prior covariance $\overline{\Sigma}_t$ and the observation model parameters.

We can use Equation 18 to compute the posterior distribution over belief means for a particular action sequence and any possible observation sequence: this will allow us to compute an analytic expression for the posterior distribution over beliefs that could result from a single macro-action. To compute the posterior distribution over belief means, we first combine the process and measurement updates for a single primitive action belief update in order to get an expression for the posterior belief means in terms of the prior belief mean. We do this by marginalizing out $\overline{\mu}_t$, the posterior belief after the transition update but before the observation update, using $p(\mu_t|\mu_{t-1}) = \int p(\mu_t|\overline{\mu}_t) p(\overline{\mu}_t|\mu_{t-1}) d\overline{\mu}_t$. As $\overline{\mu}_t$ is a deterministic function of μ_{t-1} (see Equation 8a), then $p(\overline{\mu}_t|\mu_{t-1})$ is simply a delta function, which means that $p(\mu_t|\mu_{t-1})$ is identical to Equation 18 after substituting $\overline{\mu}_t$ using Equation 8a:

$$p(\mu_t|\mu_{t-1}) \sim \mathcal{N}(A_t\mu_{t-1} + B_t u_t, \overline{\Sigma}_t C_t K_t^T). \quad (19)$$

In a one-step belief update, the belief mean at the prior time step, μ_{t-1}, was assumed to be a fixed value. However, for a macro-action, once the first primitive action has been taken, the posterior belief mean will depend on the received observation. In absence of the knowledge of that received observation, we will instead have a distribution over the belief means, which may be represented as a Gaussian $\mu_{t-1} \sim \mathcal{N}(m_{t-1}, \Sigma_{t-1}^{\mu})$ where m_{t-1} and Σ_{t-1}^{μ} are random variables. In order to compute the probability distribution over μ_t, we must integrate over this distribution of prior belief means μ_{t-1}:

$$p(\mu_t|m_{t-1}, \Sigma_{t-1}^{\mu}) = \int_{\mu_{t-1}} p(\mu_t|\mu_{t-1}) p(\mu_{t-1}|m_{t-1}, \Sigma_{t-1}^{\mu}) d\mu_{t-1} \quad (20)$$

which shows that the posterior belief mean is a Gaussian distribution over a function of the prior mean of the belief means and covariance:

$$\mu_t \sim \mathcal{N}(A_t m_{t-1} + B_t u_t, \Sigma^{\mu}_{t-1} + \overline{\Sigma}_t C_t K_t^T). \tag{21}$$

Equation 21 can now be used to perform a prediction of the posterior mean distribution after a multi-step action sequence. Assuming that the agent is currently at time t and has a particular prior mean μ_{t-1} (which we can also express as a Gaussian with zero covariance, $\mathcal{N}(\mu_{t-1}, 0)$), the posterior mean after an action sequence of T time steps is is distributed as follows:

$$\mu_{t+T} \sim \mathcal{N}(f(\mu_{t-1}, A_{t:t+T}, B_{t:t+T}, u_{t:t+T}), \sum_{i=t}^{t+T} \overline{\Sigma}_i C_i K_i^T) \tag{22}$$

where $f(\mu_{t-1}, A_{t+1:t+T}, B_{t+1:t+T}, u_{t+1:t+T})$ is the deterministic transformation of the means according to $\mu_{t+k} = A_{t+k}\ \mu_{t+k-1} + B_{t+k}\ u_{t+k}$. Since an observation on its own does not shift the mean value m_{t+k} of the distribution of posterior means, m_{t+k} is dependent only on the state-transition model parameters and can be calculated via a recursive update along the action sequence.

The posterior covariance of the agent's belief after an action is taken and an observation is obtained, can be calculated using Equation 9. These covariance update equations are only dependent on the model parameters of the problem, and are independent of the observations that may be obtained. Formally, this property exists because the Fisher information associated with the observation model is independent of the specific observations. The posterior covariance can therefore be computed in closed form, and is independent of the posterior mean.

We define b_{dist} as the posterior distribution over beliefs after a macro-action:

$$b_{dist}(\mu_{t+T}, \Sigma) = \mathcal{N}(f(\mu_{t-1}, A_{t:t+T}, B_{t:t+T}, u_{t:t+T}), \sum_{i=t}^{t+T} \overline{\Sigma}_i C_i K_i^T)\delta(\Sigma, \Sigma') \tag{23}$$

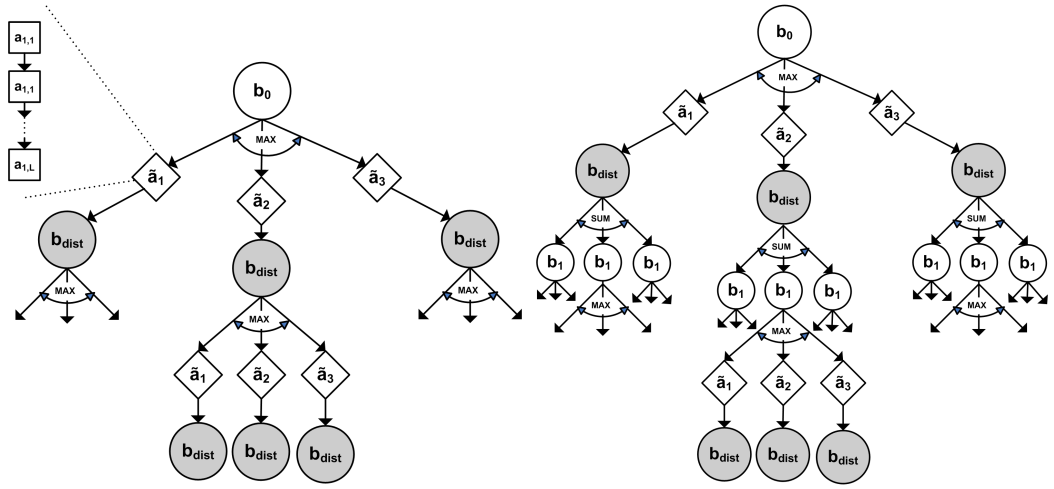
where Σ' is computed by iteratively applying Equation 9. This expression shows that for problems with linear-Gaussian state-transition and observation models, we can exactly calculate the distribution of posterior beliefs associated with a macro-action.

3.4 Planning with the Posterior Belief Distribution

We can use this approach for directly computing the posterior belief distribution in order to perform efficient macro-action forward search. Starting at the root belief, our Posterior Belief Distribution Exact (PBDE) algorithm computes an analytic expression for the posterior distribution over beliefs for each macro-action, and then branches on all possible macro-actions after each macro-action. This is repeated out to search depth $\tilde{H}$ in order to obtain the best sequence of macro-actions that the agent could execute: see Figure 4(a) for an illustration. PBDE uses the EXPAND() routine presented in Algorithm 3.

PBDE avoids a direct dependence on the time step horizon H and is only a function of the macro-action horizon $\tilde{H}$. In Section 4 we provide a detailed analysis of the performance and computational complexity of the PBDE algorithm.

However, the PBDE algorithm suffers from an important limitation: although it accurately captures the distribution of possible posterior beliefs after multiple macro-actions are executed, it never



(a) PBDE (b) PBD

Figure 4: Planning with posterior belief distributions. a) In PBDE, macro-actions $\tilde{a}_i$ are chained directly. b) In PBD individual beliefs b are sampled from the posterior distribution over beliefs b_{dist}, implicitly sampling a particular observation trajectory. Then the best macro-action is selected for each sampled posterior belief. A sum is taken over all the sampled beliefs, again corresponding to a sum over the implicitly sampled observation sequences. Here, b_i refers to beliefs at macro-action depth i.

Algorithm 3 EXPAND() (PBDE)

Input: Macro-action $\tilde{a}$, Belief state b_t or posterior distribution over belief b_{dist}, Discount factor γ, Macro-action search depth $\tilde{H}$,
if $\tilde{H} = 0$ **then**
  return 0
else {Expand Macro-action $\tilde{a}$=$\{a_1, \ldots, a_L\}$}
  $V = 0$
  $b_{dist} = b_t$
  for $j = 1$ to L **do**
    $V = V + r(b_{dist}, a_j)$
    Update the posterior distribution of beliefs b_{dist} using Equation 23
  end for
  Generate next set of macro-actions $\tilde{A}^{next}$
  for $\tilde{a}_i^{next} \in \tilde{A}^{next}$ **do**
    $Q(b_{dist}, \tilde{a}_i^{next})$ = EXPAND($\tilde{a}_i^{next}$,b_{dist},γ,$\tilde{H} - 1$)
  end for
  $V = V + \gamma^L \operatorname{argmax}_{\tilde{a}_i^{next}} Q(b_{dist}, \tilde{a}_i^{next}))$
  return V
end if

conditions on a particular sequence of observations that might be obtained. In other words, the macro-actions will be selected to maximize the expected reward assuming that any possible observation sequence might occur. However, when the agent acts in the world, it will in fact only receive one observation sequence while executing a macro-action. The received observation sequence will provide information about the underlying belief, and in general it will be helpful to condition future macro-actions on the particular sequence of observations that was actually received. We expect that algorithms that do condition on the potential set of observations received are likely to outperform PBDE and other unconditional policies.

To remedy this, ideally we would partition the distribution of beliefs after each macro actions into subsets with the same next best macro-action. This is hard to do, so instead we sample beliefs from the posterior distribution over beliefs that arises after a macro-action, as is shown in Figure 4(b). We continue the forward search from each sampled belief separately by finding the optimal next macro action. This means that the next macro-action followed can be different for different beliefs arising after a single macro-action.

Each belief sampled is implicitly sampling a particular L-length observation sequence possible given the macro-action, but this is done without having to actually perform the belief updates associated with each (implicitly) sampled observation trajectory. These sampled beliefs essentially form a non-parametric, particle estimate of the posterior distribution of beliefs that is present after taking the macro-action. As the number of samples N_s goes to infinity, the sampled distribution will become an arbitrarily good approximation of the full posterior distribution of beliefs.

Even though this algorithm has lost some of the computational benefits of using only a closed form evaluation of the posterior belief distribution, sampling directly from the posterior distribution still provides a significant computational advantage over performing belief updates for each observation trajectory. As the covariance is a Dirac delta distribution, sampling is needed only for the posterior mean distribution, generating posterior belief samples by associating each posterior mean sample with the posterior covariance Σ_{t+T}. Forward search then proceeds from each of these sampled beliefs. We label this procedure as the standard Posterior Belief Distribution (PBD) algorithm (Algorithm 4). While belief sampling adds an additional computational cost, the resulting computational complexity is still independent of the primitive action horizon. More importantly from a performance perspective, conditioning the future actions selected based on the (implicit) observation sequences received will yield significant performance gains in most domains of interest. This is also true for the MAC and MAD algorithms, though these, as we will shortly see, incur much larger computationally costs than PBD.

3.5 Approximate computation of posterior belief distribution

Unfortunately, most planning under uncertainty problems have observation functions that are non-Gaussian. When the linear-Gaussian assumption does not hold, the traditional Kalman filter model no longer provides an exact belief update, and for the PBD/PBDE algorithms, the distribution of posterior beliefs cannot be calculated exactly. Nevertheless, a more general approximate version of the Kalman filter update exists for problems with observation models that belong to a larger class of parametric distributions, known as the exponential family of distributions (Barndorff-Nielsen, 1979). Examples of exponential family distributions include the Gaussian, Bernoulli, gamma and Poisson distributions, among others. The probability distributions in this class share a certain algebraic representation, and are chosen for mathematical convenience as well as generality.

Algorithm 4 EXPAND() (PBD)

Input: Macro-action $\tilde{a}$, Belief state b_t, Discount factor γ, Macro-action search depth $\tilde{H}$, No. posterior belief samples per macro-action N_s
if $\tilde{H} = 0$ **then**
  return 0
else {Expand Macro-action $\tilde{a}$=$\{a_1, \ldots, a_L\}$}
  $V = 0$
  $b_{dist} = b_t$
  for $j = 1$ to L **do**
    $V = V + r(b_{dist}, a_j)$
    Update the posterior distribution of beliefs b_{dist}
  end for
  for $i = 1$ to N_s **do**
    Sample posterior mean n_i according to $\mathcal{N}(m_{t+T}, \Sigma^{\mu}_{t+T})$
    $b_i \leftarrow \mathcal{N}(n_i, \Sigma_{t+T})$
    Generate next set of action sequences $\tilde{A}^{next}$
    for $\tilde{a}_i^{next} \in \tilde{A}^{next}$ **do**
      $Q(b_i, \tilde{a}_i^{next})$ = EXPAND($\tilde{a}_i^{next}$,b_i,γ,$\tilde{H} - 1$,N_s)
    end for
    $V = V + \frac{1}{N_s}\gamma^L \operatorname{argmax}_{\tilde{a}_i^{next}} Q(b_i, \tilde{a}_i^{next}))$
  end for
  return V
end if

Building on statistical economics research for time-series analysis of non-Gaussian observations (Durbin & Koopman, 2000), West, Harrison and Migon (1985) showed that a dynamic generalized linear model provides the exponential family equivalent of the Kalman filter (efKF). This allows a closed-form approximation of the posterior belief for a larger class of observation models. The key idea is to construct linear-Gaussian models that approximate the non-Gaussian exponential family model in the neighborhood of the conditional mode, $s_t|z_t$. The approximate linear-Gaussian observation model can then be used in a traditional Kalman filter. For completeness we include West et al.'s derivation of the filter in Appendix A, and we present the main equations here.

Constructing the approximate linear-Gaussian model requires computation of the first two moments of the distribution and the linearization around the mean estimate at every time step. An exponential family observation model can be represented as follows,

$$p(z_t|\theta_t) = \exp(z_t^T\theta_t - \beta_t(\theta_t) + \kappa_t(z_t)), \qquad \theta_t = W(s_t) \tag{24}$$

where s_t is the hidden state of the system, θ_t and $\beta_t(\theta_t)$ are the canonical parameter and normalization factor of the distribution, and $W(.)$ maps the states to canonical parameter values. $W(.)$ is also known as the canonical link function, and depends on the particular member of the exponential family.

The first two moments of the distribution (West et al., 1985) are,

$$E(z_t|\theta_t) = \dot{\beta}_t = \frac{\partial\beta_t(\theta_t)}{\partial\theta_t}\Big|_{\theta_t=W(\overline{\mu}_t)} \quad Var(z_t|\theta_t) = \ddot{\beta}_t = \frac{\partial^2\beta_t(\theta_t)}{\partial\theta_t\partial\theta_t^T}\Big|_{\theta_t=W(\overline{\mu}_t)} \quad \theta_t = W(s_t), \tag{25}$$

where $\dot{\beta}_t$ and $\ddot{\beta}_t$ are the derivatives of the exponential family distribution's normalization factor, both linearized about $\overline{\theta}_t = W(\overline{\mu}_t)$.

Given an action-observation sequence, the posterior mean of the agent's belief in the efKF can then be updated according to

$$\overline{\mu}_t = A_t\mu_{t-1} + B_t u_t \qquad \mu_t = \overline{\mu}_t + \tilde{K}_t(\tilde{z}_t - W(\overline{\mu}_t)), \tag{26}$$

$$\overline{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + P_t \qquad \Sigma_t = (\overline{\Sigma}_t^{-1} + Y_t\ddot{\beta}_t Y_t^T)^{-1}, \tag{27}$$

where $\tilde{K}_t = \overline{\Sigma}_t Y_t(Y_t\overline{\Sigma}_t Y_t + \ddot{\beta}_t^{-1})^{-1}$ is the efKF Kalman gain, and $\tilde{z}_t = \overline{\theta}_t - \ddot{\beta}_t^{-1} \cdot (\dot{\beta}_t - z_t)$ is the projection of the observation onto the parameter space of the exponential family observation model. $Y_t = \frac{\partial \theta_t}{\partial s_t}\big|_{s_t=\overline{\mu}_t}$ is the gradient of the exponential family distribution's canonical parameter, linearized about $\overline{\mu}_t$.

The belief covariance comes from Equation 27 and Equation 22 can be modified based on the efKF equations:

$$\mu_{t+T} \sim \mathcal{N}(f(\mu_{t-1}, A_{t:t+T}, B_{t:t+T}, u_{t:t+T}), \sum_{i=t}^{t+T} \overline{\Sigma}_i Y_i \tilde{K}_i^T) \tag{28}$$

However, it is worth noting that in contrast to Equations 9 and 22 , which are exact and completely independent of the observations, Equations 27 and 28 are no longer independent of the observations obtained. This lack of independence arises because the observation model parameters are linearized about the prior mean $\overline{\mu}_t$. Hence while the parameters are independent of the observation that will be obtained for a macro-action sequence of length 1, for a longer macro-action, the observation model parameters depend on the prior observations obtained. Nevertheless, we approximate this update by linearizing about the mean of the prior mean distribution m_t at each step along the action sequence, rather than the true prior belief mean μ_t. We still obtain good experimental results using this approximation.

This approximate method for computing the posterior distribution over beliefs can be used as a substitute for exactly calculating the posterior distribution over beliefs in either the PBD or PBDE algorithm.

4. Analysis

Here we provide a formal analysis of the accuracy and computational complexity of our macro-action forward search algorithms. We first show that under certain assumptions, the computed value of an action under the macro-action forward search approach will be a lower bound of its optimal value. Next we loosen some of the prior assumptions, and consider the computational complexity of macro-action forward search planning under these new cases. We then analyze the computational complexity of these algorithms. For comparison, we briefly consider some alternate representations and approaches and analyze their respective accuracy and computational cost.

4.1 Performance

First note that the value of the action at the start of a L-length macro action, where L is greater than one, is necessarily a lower bound of the optimal L-step value of that action. This is simply because

the macro-action pre-determines a set of subsequent actions, whereas the optimal value involves taking the maximum over the reward of subsequent action choices.

For completeness, we now prove the known result that the error between the optimal infinite-horizon expected belief-action value and the H-horizon expected belief-action value is bounded as a function of the horizon H:

Lemma 1 *Let the possible rewards be bounded by 0 and R_{max}. Then the error between the infinite-horizon optimal expected belief-action value $Q^*(b,a)$ and the H-length horizon expected belief-action value $Q_H(b,a)$ is a bounded function of H:*

$$Q^*(b,a) - Q_H(b,a) \leq \frac{R_{max}\gamma^{H+1}}{1-\gamma}. \tag{29}$$

Proof Let R_i be the expected reward on the i-th time step. Then

$$\begin{aligned} Q^*(b,a) - Q_H(b,a) &= \sum_{i=0}^{\infty} \gamma^i R_i - \sum_{i=0}^{H} \gamma^i R_i & (30)\\ &\leq \sum_{i=H+1}^{\infty} \gamma^i R_{max} & (31)\\ &= \sum_{i=0}^{\infty} \gamma^i R_{max} - \sum_{i=0}^{H} \gamma^i R_{max} & (32)\\ &= \frac{R_{max}}{1-\gamma} - \frac{R_{max}(1-\gamma^{H+1})}{1-\gamma} & (33)\\ &= \frac{R_{max}\gamma^{H+1}}{1-\gamma}. & (34) \end{aligned}$$

where the first inequality follows since the additional rewards are at most R_{max} for the subsequent time steps, the second equality just re-expresses the sum as a difference of two sums, the third equality takes the geometric sum, and the final equality is our desired result. □

Theorem 4.1 *Assume that the dynamics and observation models are linear Gaussian, and the reward model is either a weighted sum of Gaussians or a polynomial function of the state space. Then the H-horizon belief-action value of the best root action a^{PBDE} computed by the PBDE algorithm is a lower bound on the infinite-horizon optimal value of a^{PBDE}.*

Proof We will first show that we can compute the the PBDE H-horizon belief-action values exactly, and then combine this result with Lemma 1 to prove the theorem.

First, let the primitive action a^{PBDE} be the first action in the L-length macro-action $\tilde{a}^{PBDE}$. The value of a^{PBDE} is the expected L-horizon value of the macro-action $\tilde{a}^{PBD}$ plus the expected future value after the macro-action. Let $\tilde{Q}_h$ be used to represent values when macro-actions are used, and the subscript will be used to represent the value horizon either in terms of primitive actions (H) or in terms of the number of macro-actions if a tilde is used ($\tilde{H}$). More specifically, $\tilde{Q}_{\tilde{H}}(b_{dist}, a)$ is the value of taking $\tilde{H}$ macro-actions starting from a distribution of beliefs b_{dist} and primitive action a, which is the first action in macro-action $\tilde{a}$. Note that b_{dist} could be a delta function over a single

belief b. $\tilde{Q}_L(b_{dist}, \tilde{a})$ is the value of the single macro-action $\tilde{a}$ taken from the distribution of beliefs b_{dist}. Then

$$\begin{aligned} \tilde{Q}_{\tilde{H}}(b_{dist}, a^{PBDE}) &= \tilde{Q}_{\tilde{H}}(b_{dist}, \tilde{a}^{PBDE}) && (35) \\ &= \tilde{Q}_L(b_{dist}, \tilde{a}^{PBDE}) + \max_{\tilde{a} \in \tilde{A}} \tilde{Q}_{\tilde{H}-1}(b_{dist}^{\tilde{a}^{PBDE}}, \tilde{a}), && (36) \end{aligned}$$

where $b_{dist}^{\tilde{a}^{PBDE}}$ is the distribution over beliefs after macro-action $\tilde{a}^{PBDE}$. The first equality holds trivially as the value of the action at the start of a macro-action under PBDE must be the same as the value of the macro-action itself. The second equality holds because the value of $\tilde{a}$ under the PBDE algorithm is the expected reward of following $\tilde{a}$ plus the value of following the best next macro-action, given the distribution over beliefs after taking $\tilde{a}$. This definition is recursive, and so the PBDE value of a macro-action $\tilde{a}$ is simply the value of the best $\tilde{H}$ sequence of macro-actions that lead to the highest expected reward given the initial belief distribution b_{dist}. Note that there is no sampling performed (of beliefs or observations) and so this is essentially an $L\tilde{H}$-length open loop policy. This expected value computation is therefore identical to finding the value of a single best macro-action $\tilde{a}^H$ that starts with a^{PBDE} and consists of $\tilde{H}$, L-length macro-actions in a row (for a total of $H = L\tilde{H}$ primitive actions):

$$\tilde{Q}_{\tilde{H}}(b_{dist}, a^{PBDE}) = r(b_{dist}, a^{PBDE}) + \sum_{z \in Z} p(z|b, a^{PBDE}) \tilde{Q}_{H-1}(b^{a^{PBDE}, z}, a_2^H) \quad (37)$$

where a_2^H is the second primitive action in the macro-action $\tilde{a}^H$.

To prove this is a lower bound to the optimal value of a^{PBDE} we first need to show that PBDE can exactly compute the expected reward for any given primitive action. Since the dynamics and observation models are linear Gaussian, then the well-known Kalman filter can be used to exactly perform belief updating. Similarly, the posterior distribution over beliefs b_{dist} after each action can be computed exactly. Calculating the expected reward of a_i for a distribution of beliefs b_{dist} requires integrating over this posterior distribution:

$$r(a_i, b_{dist}) = \int_s \int_b r(s, a_i) b(s) b_{dist}^{a_i}(b) db ds. \quad (38)$$

Recall from the prior section that the posterior distribution over beliefs can be factored into a Gaussian distribution over the belief means (Equation 22), and a Dirac delta distribution over the belief covariances (since all beliefs will have identical covariances):

$$b_{dist} = \mathcal{N}(\mu | m_i, \Sigma_i^\mu) \delta(\Sigma, \Sigma_i) \quad (39)$$

where m_i is the mean of the belief means after action a_i, Σ_i^μ is the covariance of the belief means after action a_i, and Σ_i is the covariance of a belief state after action a_i.

As the belief state itself is a Gaussian,

$$b(s) = \mathcal{N}(s | \mu, \Sigma), \quad (40)$$

we can re-express the reward as

$$\begin{aligned} r(a_i, b_{dist}) &= \int_s \int_{\mu, \Sigma} r(s, a_i) \mathcal{N}(s|\mu, \Sigma) \mathcal{N}(\mu | m_i, \Sigma_i^\mu) \delta(\Sigma_i, \Sigma) ds d\mu d\Sigma && (41) \\ &= \int_s \int_\mu r(s, a_i) \mathcal{N}(s|\mu, \Sigma_i) \mathcal{N}(\mu | m_i, \Sigma_i^\mu) d\mu ds. && (42) \end{aligned}$$

Expanding out the formula for $\mathcal{N}(s|\mu, \Sigma_i)$ we see it is identical to the formula for $\mathcal{N}(\mu|s, \Sigma_i)$:

$$\begin{aligned}\mathcal{N}(s|\mu, \Sigma_i) &= \frac{1}{\sqrt{2\pi}|\Sigma_i|^{N_d/2}} exp(-\frac{1}{2}(s-\mu)\Sigma_i^{-1}(s-\mu)^T) &(43)\\ &= \frac{1}{\sqrt{2\pi}|\Sigma_i|^{N_d/2}} exp(-\frac{1}{2}(\mu-s)\Sigma_i^{-1}(\mu-s)^T) &(44)\\ &= \mathcal{N}(\mu|s, \Sigma_i). &(45)\end{aligned}$$

Therefore, we can substitute the equivalent expression to yield

$$r(a_i, b_{dist}) = \int_s \int_\mu r(s, a_i)\mathcal{N}(\mu|s, \Sigma_i)\mathcal{N}(\mu|m_i, \Sigma_i^\mu) d\mu ds. \quad (46)$$

Completing the square in the exponent, we re-express the product of the above two Gaussians as

$$r(a_i, b_{dist}) = \int_s \int_\mu r(s, a_i)\mathcal{N}(s|m_i, \Sigma_i + \Sigma_i^\mu)\mathcal{N}(\mu|c, C) d\mu ds. \quad (47)$$

where $C = (\Sigma_i^{-1} + (\Sigma_i^\mu)^{-1})^{-1}$ and $c = C(m_i(\Sigma_i^\mu)^{-1} + \mu\Sigma_i^{-1})$. We then integrate over μ to get

$$r(a_i, b_{dist}) = \int_s r(s, a_i)\mathcal{N}(s|m_i, \Sigma_i + \Sigma_i^\mu) ds. \quad (48)$$

If the reward model is a weighted sum of Gaussians,

$$r(a, s) = \sum_{j=1}^{W} w_j \mathcal{N}(s|\zeta_j, \Upsilon_j), \quad (49)$$

then the integral in Equation48 can be evaluated in closed form as

$$\begin{aligned}r(a_i, b_{dist}) &= \int_s \sum_{j=1}^{W} w_j \mathcal{N}(s|\zeta_j, \Upsilon_j)\mathcal{N}(s|\mu_i, \Sigma_i + \Sigma_i^\mu) ds &(50)\\ &= \sum_{j=1}^{W} w_j \mathcal{N}(\zeta_j|\mu_i, \Upsilon_j + \Sigma_i + \Sigma_i^\mu) \int_s \mathcal{N}(s|c_1, C_1), &(51)\end{aligned}$$

where we have again completed the square in the exponent, and defined new constants $C_1 = (\Upsilon_j^{-1} + (\Sigma_i + \Sigma_i^\mu)^{-1})^{-1}$ and $c_1 = C_1(\zeta_j\Upsilon_j^{-1} + \mu_i(\Sigma_i + \Sigma_i^\mu)^{-1})$. Integrating we obtain an analytic expression for the expected reward of a primitive action under a distribution of beliefs:

$$r(a_i, b_{dist}) = \sum_{j=1}^{W} w_j \mathcal{N}(\zeta_j|\mu_i, \Upsilon_j + \Sigma_i + \Sigma_i^\mu). \quad (52)$$

A similar closed-form expression is available if the reward model is a polynomial functions of the state,

$$r(a_i, s) = \sum_{j=1}^{W} w_j s^j, \quad (53)$$

instead of a weighted sum of Gaussians. Substituting Equation 53 into Equation 48 yields

$$\begin{aligned} r(a_i, b_{dist}) &= \sum_{i=1}^{L} \int_s \sum_{j=1}^{W} w_j s^j \mathcal{N}(s|\mu_i, \Sigma_i^{\mu} + \Sigma_i) ds \\ &= \sum_{i=1}^{L} \sum_{j=1}^{W} w_j \int_s s^j \mathcal{N}(s|\mu_i, \Sigma_i^{\mu} + \Sigma_i) ds. \end{aligned} \tag{54}$$

Therefore, evaluating the expected reward involves calculating the first W moments of a Gaussian distribution. Each of these moments is an analytic expression of the Gaussian mean and covariance.[1] So, for reward models that are either a weighted sum of Gaussians, or which are polynomial functions of the state space, the expected reward of a macro-action can be exactly computed.

Equations 52 and 54 prove that we can exactly compute the expected reward $r(b_{dist}, a)$ for any primitive action, which allows us to exactly evaluate Equation 37, the value of a $L\tilde{H}$-length macro-action that starts with action a^{PBDE}. This value is necessarily a lower bound on the optimal value $Q^*_{L\tilde{H}}(b, a^{PBDE})$ which allows any future actions instead of only those in the macro-action. In addition, Lemma 1 guarantees the $L\tilde{H}$-step finite horizon optimal value is a lower bound to the infinite horizon optimal value. Putting these results together, we immediately derive the theorem result. □

4.2 Performance analysis of PBD

While Theorem 4.1 provides formal bounds on the values computed by the non-sampled PBD variant (PBDE), PBDE is unlikely to perform well in practice because it selects a sequence of macro-actions in an open loop fashion, without conditioning on observation nodes in the forward search tree. In contrast, the PBD, MAD and MAC algorithms follow a conditional instead of unconditional/open-loop plan.

We now turn our attention to the PBD algorithm. The $L\tilde{H}$-horizon value of a root action a^{PBD} coming from macro action $\tilde{a}^{PBD}$ in the standard PBD algorithm is:

$$\tilde{Q}_{\tilde{H}}(b, a^{PBD}) = \tilde{Q}_L(b, a^{PBD}) + \frac{1}{N_s} \sum_{i=1}^{N_s} \max_{\tilde{a}^* \in \tilde{A}} \tilde{Q}_{\tilde{H}-1}(b_i, \tilde{a}^*) \tag{55}$$

where the beliefs b_i are sampled from the posterior distribution of beliefs arising after macro-action $\tilde{a}^{PBD}$.

In general, if the set of sampled beliefs constitutes an particle representation that is identical to the true posterior distribution of beliefs, then Equation 55 is guaranteed to be a lower bound of the belief-action value of a^{PBD}. This is because the expectation over the sampled beliefs will match the true distribution over the potential observation sequences. The value is a lower bound because the macro-actions restrict the policy class considered.

However, it is unlikely that the sampled belief distribution is identical to the true posterior distribution over beliefs, and it would be hard to know for any finite number of belief samples if this set exactly captured the true posterior distribution over beliefs. As the number of samples N_s goes

1. The Gaussian distribution is completely described by its first two moments; all higher order moments are simply functions of the first two moments.

to infinity, the sampled posterior distribution will becomes arbitrarily close to the true distribution, and Equation 55 will become a lower bound on the optimal belief-action value. However, for a finite number of belief samples N_s, Equation 55 cannot be guaranteed to be an upper or lower bound on the true action value. This is because even if the value of each sampled belief is guaranteed to be a lower bound, the distribution over sampled beliefs may disproportionately include beliefs that lead to higher rewards.

Therefore, the belief-action values of both the PBD algorithm which samples beliefs from the posterior distribution over beliefs after a macro-action, and the MAC/MAD algorithms that sample observation sequences, are unknown to be upper or lower bounds on the optimal values of the root actions. However, we do expect them to generally perform better than PBDE, since PBD, MAC and MAD all condition, implicitly or explicitly, on the received observations when constructing a policy, unlike PBDE. This intuition was confirmed in informal experimentation.

Similarly, for a significant class of observation and dynamics models, it will not be possible to exactly update the belief state and maintain a Gaussian representation over the resulting posterior. Instead, as we described in Section 3.5, we take the popular approach of approximating the resulting distribution as a Gaussian by linearizing about the resulting reward function. Linearization makes the posterior distribution of beliefs $\hat{b}_{dist}$ an approximation of the true posterior distribution b_{dist}. The degree of error in the reward estimate depends critically on the interaction of the domain-specific reward formulation, and the encountered belief states:

$$|r(a_i, \hat{b}_{dist}) - r(a_i, b_{dist})| = \left| \sum_{i=1}^{L} \int_s \int_b r(s, a_i) b(s) (b_{dist} - \hat{b}_{dist}) dbds \right|. \tag{56}$$

In general, this error could be significant. However, our experimental results demonstrate that using sampling and linearizing within a macro-action forward search can still outperform prior approaches where considering further horizons can improve action selection.

In essence, our approach trades the exact solutions of exhaustive forward search procedures for a more restricted policy class that allows us to efficiently search to longer horizons and approximately compute the values of those longer-horizon policies. The potential benefit of this tradeoff is made clearer by reconsidering the bounds of Lemma 1, which provides an upper bound to the error between the optimal infinite-horizon value and the H-step horizon value which is a directly function of the horizon H. Consider the case when the discount factor is 0.95. If H is 3 steps, then this upper bound evaluates to $15.48R_{max}$ or $0.77V_{max}$ where V_{max} is defined as $R_{max}/(1-\gamma)$. In contrast, if H is 20, then this upper bound evaluates to $0.34V_{max}$, which is less than half the upper bound for $H = 4$. This simple example illustrates the potential benefit in accuracy of the computed value functions obtained by searching out to longer horizons. As our underlying assumption is that better estimates of belief-action values will lead to better policies, computing the value of longer-horizon policies may have significant performance advantages. Even approximately estimating the values of long-horizon policies can outperform full forward search, which often becomes computationally intractable after a short horizon. This claim will be supported experimentally later in this paper.

4.3 Computational Complexity

One of the central contributions of our work is providing an efficient macro-action forward search algorithm that can scale to long horizons and large problems. We first analyze the computational complexity of searching using an analytic parametric distribution of the result of a macro-action;

performance bounds were provided in Theorem 4.1. We then consider how this computational cost changes when observation sequences are sampled and a discrete representation is used, such as in the MAD algorithm approach.

4.3.1 Complexity of Gaussian Belief Updating for a Length L Macro-action

The computation for the resulting posterior distribution over beliefs after a macro-action was presented in Equation 28, and consists of a set of matrix multiplications and inversions. Matrix multiplication is an $\mathcal{O}(D^2)$ computation, where D is the state space dimension. Matrix inversion can be done in $\mathcal{O}(D^3)$ time. Therefore the computational cost of performing a single update of the posterior over belief states is an $\mathcal{O}(D^3)$ operation. This update must be performed for each primitive action in a length-L macro-action $\tilde{a}$, resulting in a computational cost of

$$\mathcal{O}(LD^3) \tag{57}$$

for a single macro-action.

4.3.2 Complexity of Analytically Computing the Expected Reward of a Length L Macro-action

The second component of the computational load comes when we evaluate the expected reward of a macro-action. If the reward is a weighted sum of W Gaussians, as specified by Equation 49, this operation involves evaluating the value of WL Gaussians at particular fixed points. Evaluating a D-dimensional Gaussian at a single point is an $\mathcal{O}(D^3)$ operation, due to the inverse covariance that must be computed. The cost for performing this operation WL times is simply $\mathcal{O}(WLD^3)$. Therefore the total cost for evaluating the expected reward of a macro-action when the belief can be represented exactly as a Gaussian and the reward model is a weighted sum of W Gaussians is:

$$\mathcal{O}(LD^3(W+1)). \tag{58}$$

If instead the reward model is a W-th degree polynomial function of the state, then the expected reward calculation consists of the cost of calculating the W-moments of a D-dimensional Gaussian distribution (Equation 54). Assume without loss of generality that we are computing the W-th central moment of a D-dimensional Gaussian: a non-central moment can always be converted into a central moment by adding and subtracting a mean term. Let the W-th central moment denote moments of the form $E[(s_1 - E[s_1])^2(s_2 - E[s_2]) \dots (s_D - E[s_D])]$ or $E[(s_2 - E[s_2])^W]$, and σ_{ij} denote the ij-th entry of the covariance matrix. From Triantafyllopoulos (2003) we know that if W is odd, the central Wth moments are zero, and if W is even ($W = 2k$) the Wth central moments can be decomposed into a sum over products of L covariance terms:

$$= \sum \sigma_{i_1 j_1} \sigma_{i_2 j_2} \dots \sigma_{i_k j_k} \tag{59}$$

where the sum is taken over all permutations of product pairs. This sum yields $(W-1)!/(2^{k-1}(k-1)!)$ terms which consist of covariance elements to the power of at most k. For a particular central moment, this cost is independent of the dimension of the state space. Therefore the cost is dominated by the number of terms, which grows at slightly less than $\mathcal{O}(W!)$. There will also be an additional cost if the original polynomial was not a central moment calculation, which will involve at most

W D-dimensional matrix multiplications, yielding a cost of $\mathcal{O}(WD^2)$. In summary, the cost of computing the expected reward when the reward is a polynomial function will be

$$\mathcal{O}(L(D^3 + W! + WD^2)). \tag{60}$$

4.3.3 COMPLEXITY OF OPEN LOOP PLANNING (THE PBDE ALGORITHM)

Performing forward search requires both updating the posterior distribution of beliefs, and computing the expected reward, recursively out to a fixed macro-action depth horizon $\tilde{H}$. Consider first the computational complexity when we can only select a single macro-action ($\tilde{H} = 1$). In this case, we only need to compute the expected posterior distribution and reward of each of M macro-actions. From Equations 57, 58, and 60, the computational complexity for calculating the posterior distribution over beliefs and expected reward for a single action is $\mathcal{O}(LD^3)$, so for M macro actions the total cost is simply $\mathcal{O}(MLD^3)$. More generally, for a horizon of $\tilde{H}$ macro-actions, PBDE planning consists of considering all $M^{\tilde{H}}$ potential sequences of macro-actions, with an associated computational complexity of

$$\mathcal{O}(M^{\tilde{H}}LD^3W). \tag{61}$$

Therefore the cost scales as a cubic function of the number of state dimensions, and exponentially with the macro-action search depth, where the base is the number of macro-actions. This cost is independent of the number of observations, and scales linearly with the number of primitive actions (L) per macro-action. For long-horizon searches, if the number of macro-actions is small, this will be a significant savings over standard full primitive-action forward search approaches.

4.3.4 COMPLEXITY OF CONDITIONAL MACRO-ACTION PLANNING WITH PBD

PBDE does not provide the opportunity to improve the selected plan by conditionally select different macro-actions given particular observation sequences. PBD does provide this capability by sampling beliefs from the posterior distribution over beliefs at each depth, which is equivalent to sampling observation sequences, and searching forward based on each sample. We next consider the additional computational cost incurred in performing belief sampling.

Sampling beliefs from the posterior distribution over beliefs requires sampling from a multivariate Gaussian over the distribution of belief means, which can be done by first computing the Cholesky decomposition of the covariance matrix, $AA^T = \mu_i \sum_{j=1}^{i} \bar{\Sigma}_j C_j K_j^T + \Sigma^i$. This is an $\mathcal{O}(D^3)$ operation. Each belief mean is generated by first constructing a D-dimensional vector q, consisting of D independent samples from a standard (scalar) normal distribution. A sample from the desired multivariate Gaussian $\mathcal{N}(s|\mu_i, \Phi^i \sum_{j=1}^{i} \bar{\Sigma}_j C_j K_j^T + \Sigma^i)$ is simply $\phi^i + Aq$. Sampling N_s times involves the one-time cost of computing the Cholesky decomposition plus the matrix-vector multiplication for each sample, yielding a cost of

$$\mathcal{O}(D^3 + N_s D^2). \tag{62}$$

This procedure is performed at every branch point in the forward search tree (in other words, after all macro-action nodes except those at the tree leaves). For concreteness, consider a horizon of two macro-actions ($\tilde{H} = 2$). After expanding out each of the M actions, we will sample N_s beliefs. From each resulting belief state, we will again expand each of the M macro-actions: refer back to

Figure 4(b) for an illustration. The computational complexity is now the sum of the cost at horizon one and two:

$$\mathcal{O}(M(LD^3W + N_sD^2 + D^3) + M^2N_sLD^3W) = \mathcal{O}(M(N_sD^2 + D^3) + M^2N_sLD^3C), \quad (63)$$

where the second expression is derived by considering only the higher order terms. In general, the computational complexity of selecting an action using PBD when considering a future horizon of $\tilde{H}$ macro-actions is

$$\mathcal{O}(M^{\tilde{H}-1}N_s^{\tilde{H}-2}(N_sD^2 + D^3) + M^{\tilde{H}}N_s^{\tilde{H}-1}LD^3C). \quad (64)$$

4.3.5 Complexity of PBD with Arbitrary Reward Models

For arbitrary reward models it will not be possible to analytically compute the expected reward. Instead the expected reward for each primitive action a_i within the macro-action can be approximated by sampling D-dimensional states and estimating the expected reward by averaging the reward of each sampled state.[2] The cost of sampling N_s states involves sampling N_s times from a multivariate Gaussian, which is an $\mathcal{O}(D^3 + D^2)$ operation (from Equation 62). Assuming that calculating the reward for each sample takes time linear in the state dimension, then sampling rewards adds an additional

$$\mathcal{O}(D^3 + ND^2D) = \mathcal{O}(D^3(N + 1)) \quad (65)$$

cost to each primitive action within a macro-action, yielding a total complexity of PBD planning with reward sampling of:

$$\mathcal{O}(M^{\tilde{H}}N_s^{\tilde{H}}LD^3 + M^{\tilde{H}}N_s^{\tilde{H}}LD^2). \quad (66)$$

4.3.6 Complexity of macro-actions with Observation Sequence Sampling & Parametric Beliefs

An alternate to analytically computing the posterior distribution over beliefs after a macro-action is to use a particle approximation of this distribution by sampling a set of observation trajectories. If observation sequences are sampled, then even for a single macro action, a belief update must be performed for each of the N_z sampled L-length observation trajectories, yielding a computational cost of $\mathcal{O}(MN_zLD^3)$ for a single macro-action, assuming that the expected reward can be calculated analytically. At longer horizons ($H > 1$) the number of observation trajectories will scale exponentially with the horizon. The computational complexity of sampling observation sequences when the expected reward can be analytically evaluated is

$$\mathcal{O}(M^{\tilde{H}}(N_z)^{\tilde{H}}LD^3). \quad (67)$$

2. Note that if the rewards are bounded, for a given ϵ and δ, sampling a sufficient number of samples $N_s = f(\epsilon, \delta)$, guarantees the estimate of the expected reward of a primitive action is is ϵ-close to the true expected value, with probability at least $1 - \delta$. The proof of this is a simple application of Hoeffding's inequality (Hoeffding, 1963). If N_s is set such that the estimated reward of each primitive action is $\frac{\epsilon}{L}$ close to the true expected primitive action reward with probability at least $1 - \frac{\delta}{\epsilon}$, then the triangle inequality and union bound guarantee that the expected reward of the entire length-L macro-action is ϵ-close to the true expected reward for the macro-action with probability at least $1 - \delta$.

If instead the reward must be numerically estimated, then an additional N_s samples are required to evaluate the expected reward at each belief, which, from Equation 62, increases the computational complexity to

$$\mathcal{O}(M^{\tilde{H}}(N_z)^{\tilde{H}}LD^3 + M^{\tilde{H}}(N_z)^{\tilde{H}}N_sLD^2). \tag{68}$$

It is useful to compare these results to the standard PBD algorithm which uses an analytic expression over the posterior distribution of beliefs. The complexity will depend on the relative number of observation sequences sampled versus the number of beliefs sampled. However, an interesting situation to examine is when the number of observation sequences sampled is equal to the number of beliefs sampled in the standard PBD case, which is identical to the number of states sampled to evaluate numerically the expected reward. Let us call this number N_s. Assuming the number of observation trajectories samples is the same as the number of beliefs sampled in the PBD algorithms is particularly intuitive, as both are done to try to instantiate a particle-like approximation of the resulting posterior distribution of beliefs after a macro-action. In this situation the computational complexity of evaluating observation sequences is

$$\mathcal{O}(M^{\tilde{H}}N_s^{\tilde{H}}LD^3) \tag{69}$$

which is approximately[3] N_s larger than the computational complexity of the PBD algorithm

$$\mathcal{O}(M^{\tilde{H}}N_s^{\tilde{H}-1}LD^3). \tag{70}$$

This relationship is also true when the reward must be evaluated numerically: see Table 1. Therefore, if a number of samples are necessary to get good experimental results, we expect that the standard PBD algorithm will run significantly faster for the same N_s than sampling observation sequences.

4.3.7 COMPLEXITY OF MAD: MACRO-ACTIONS WITH OBSERVATION SEQUENCE SAMPLING & DISCRETE BELIEFS

Finally, we consider observation sequence sampling when using a factored-discrete state representation, as in our MAD algorithm. Assuming that each state dimension is independent of the other dimensions, and that each state dimension is divided into g uniformly-spaced grid cells, then the total number of discrete states is gD. Note that unlike in the PBD approach, there is no analytic expression to represent the posterior distribution over the belief state consisting of D g-length vectors after a sequence of primitive actions corresponding to a macro-action. As for sampling observation sequences in the parametric case, the computational cost includes performing belief updates and evaluating the reward along each of the N_z sampled length-L observation sequences. The cost of performing a belief update is a quadratic function of the state space, but can be done independently for each dimension, leading to a cost of $\mathcal{O}(Dg^2)$. Assuming the reward model is also factored, computing the expected reward consists of D dot products between g-length vectors. The final cost is similar to sampling when using a parametric representation (Equation 67) but with a different cost per primitive action. Therefore MAD has a computational cost of

$$\mathcal{O}(M^{\tilde{H}}N_z^{\tilde{H}}Lg^2D). \tag{71}$$

3. The exact relation involves considering the lower order terms of the computational complexity.

Algorithm	Eqn.	Computational Complexity
PBDE	61	$\mathcal{O}(M^{\tilde{H}} L D^3 W)$
PBD, analytic expected reward	64	$\mathcal{O}(M^{\tilde{H}} N_s^{H-1} L D^3 W)$
PBD, numeric expected reward	66	$\mathcal{O}(M^{\tilde{H}} N_s^{\tilde{H}} L D^3 + M^{\tilde{H}} N_s^{\tilde{H}} L D^2)$
MAC, analytic reward	67	$\mathcal{O}(M^{\tilde{H}} N_s^{\tilde{H}} L D^3 W)$
MAC, numeric reward	68	$\mathcal{O}(M^{\tilde{H}} N_s^{\tilde{H}} L D^3 + M^{\tilde{H}} N_s^{\tilde{H}+1} L D^2))$
MAD with factored state	71	$\mathcal{O}(M^{\tilde{H}} N_s^{\tilde{H}} L g^2 D)$
MAD with non-factored states	72	$\mathcal{O}(M^{H} N_s^{H} L g^{2D})$

Table 1: Computational complexity of selecting an action using PBD algorithm and closely related alternatives. D is the number of state dimensions, H is the forward search horizon (as measured in macro-actions), and g is the number of discrete states per state dimension. To make the comparison between the approaches clearer, we have expressed the complexity as a function of N_s which is used interchangeably to represent the number of beliefs sampled per macro-action, the number of states sampled to numerically estimate the expected reward, and the number of observation sequences sampled per macro-action. To simplify the expressions we have only kept the dominant terms.

Note that in high-dimensional problems, this is significantly less than using a fully dependent model, which would result in g^D states, and an associated

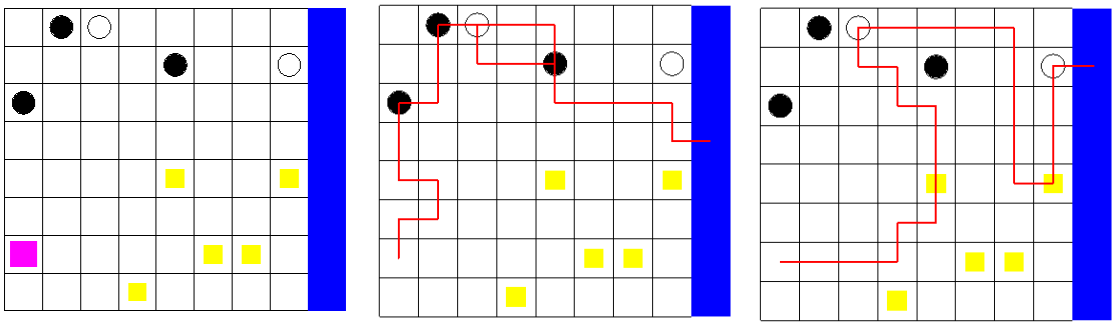
$$\mathcal{O}(M^{\tilde{H}} N_z^{\tilde{H}} L g^{2D}) \tag{72}$$

computational complexity for planning. In continuous domains, the quality of a discrete approximation will depend on the grid cell granularity: by making the tiling sufficiently fine, it is possible to make the discretized dynamics, reward and observation models accurate to any specified precision. However, finer grids have a direct impact on the computational cost of planning, particularly in fully dependent (non-factored) representations.

5. Experimental results

In this section we test our algorithm in planning under uncertainty problems from two different research communities: a scientific exploration problem from the POMDP literature, as well as our running target-tracking example, which originates from the sensor resource management domain. Despite the different origins and state space representations of these problems, they both require searching over a long horizon in order to generate good policies. Our macro-action forward search approach outperforms existing approaches in both settings. We also demonstrate our algorithm in a target-tracking problem on an actual helicopter platform, underscoring the applicability of our algorithm to real-world domains.

In both problems we compare the PBD algorithm to state-of-the-art approaches from the relevant research community. We also compare our macro-action forward search algorithms which sample observation trajectories (MAC and MAD). The PBDE algorithm was left out of the comparisons as we found that the algorithm performed poorly, due to the reasons discussed in Section 3.4.



(a) ISRS(8,5) (b) SARSOP policy (c) PBD policy

Figure 5: Information Search Rocksample (ISRS) problem. An agent (pink square) explores and samples rocks (circles) in the world. White circles correspond to rocks with positive value, black otherwise. Yellow squares indicate locations of the rock information beacons. Red lines indicate paths taken by an agent executing the (b) SARSOP and (c) PBD policies.

5.1 Rocksample

The scientific exploration Rocksample problem is a benchmark POMDP problem first proposed by (Smith & Simmons, 2004), and subsequently extended to the FieldVisionRockSample (FVRS) problem by (Ross & Chaib-draa, 2007). Initial experiments in these domains revealed that a short lookahead search was sufficient to obtain good policies. As our interest is on domains which require long-horizon lookahead, we created a new variant of the rocksample problem called the Information Search Rocksample (ISRS) problem, shown in Figure 5(a). In ISRS an agent explores and samples k rocks in a $n \times n$ grid world. The positions of the agent (pink square) and the rocks (circles) are fully observable, but the value of each rock (good or bad) is unknown to the agent. At every time step, the agent receives a binary observation of the value of each rock. The accuracy of this observation depends on the agent's proximity to rock information beacons (yellow squares) that each correspond to a particular rock (for example, these could be mountain tops that offer a particularly good view of a far off geologic formation). A key problem characteristic is that the rock information beacons are not at the same locations as the rock themselves. Unlike previous rocksample formulations, information gathering and reward exploitation require different actions.

The agent gets a fixed positive reward for collecting a good rock (white circle), a negative reward for collecting a bad rock (black circle), and a smaller positive reward for exiting the problem. A discount factor γ encourages the agent to collect rewards sooner. All other actions have zero rewards.

The observation model is a Bernoulli distribution:

$$p(z_{i,t}|m_i, r_t, RB_i) = \begin{cases} 0.5 + (m_i - 0.5)2^{\frac{-\|r_t - RB_i\|_2}{D_0}} & z_{i,t} = 1 \\ 0.5 - (m_i - 0.5)2^{\frac{-\|r_t - RB_i\|_2}{D_0}} & z_{i,t} = 0 \end{cases} \tag{73}$$

where $z_{i,t}$ is a binary $\{0 \text{ or } 1\}$ observation for the value of the rock i at time t,
m_i is the true value $\{0 \text{ or } 1\}$ of the rock,

	ISRS[8,5]		
	Ave rewards	Online time (s)	Offline time (s)
QMDP	1.11 $\pm$ 0.43	0.0001	3.03
HSVI2	6.76 $\pm$ 2.55	0.017	150
HSVI2	6.78 $\pm$ 2.46	0.051	1000
SARSOP	4.47 $\pm$ 2.74	0.007	150
SARSOP	8.25 $\pm$ 2.35	0.045	10000
SARSOP	8.46 $\pm$ 2.46	0.070	25000
RTBSS (d5, s10)	9.78 $\pm$ 1.69	17.64	0
MAC (d3,s50)	13.68 $\pm$ 1.86	15.39	0
PBD (d3,s50)	14.49 $\pm$ 1.73	1.26	0
MAD (d3,s50)	**15.88** $\pm$ 1.58	4.81	0
Fully observable	21.37	N.A.	N.A.

Table 2: ISRS results. HSVI2 and SARSOP were executed offline for a range of durations. For the forward search algorithms, the numbers in brackets represent the search depth (d) and number of posterior beliefs obtained (s) at the end of each action/macro-action. Online time indicates the average time taken by the planner to return a decision at every time step. Standard error values are shown.

r_t is the agent's position at time t,
RB_i is the location of the information beacon associated with rock i,
D_0 is a tuning parameter that controls how quickly the accuracy of the observations decrease with greater distance between the agent and the beacon.

For our PBD and MAC algorithms, we represent the agent's belief of each rock's value as a Gaussian distribution over the [0,1] state space, and take advantage of the efKF presented in Section 3.5 to represent the Rocksample problem's Bernoulli observation model (Equation 73: see Appendix B for details).

Given domain knowledge, we manually chose macro-actions that consist of the sequence of actions that will get the agent to a rock, an information beacon, or to the nearest exit. If the agent is currently on a rock, the action sequences have the additional option of sampling the rock as the first action in the action sequence, resulting in twice as many macro-actions.

As the Rocksample family of problems originates from the POMDP literature, we compared our macro-action algorithms to existing state-of-the-art POMDP solvers: a fast upper bound QMDP (Littman et al., 1995), the point-based offline value-iteration techniques HSVI2 (Smith & Simmons, 2005) and SARSOP (Kurniawati et al., 2008), as well as RTBSS (Paquet et al., 2006), an online, factored, forward search algorithm. Since all approaches, including our own, are approximations, we also include as an upper bound the value of the fully observable problem.

Table 2 compares the performance of the different algorithms in the ISRS problem. Each algorithm was tested on 10 sets of hidden rock values, and each scenario was tested 20 times. The HSVI2 and SARSOP algorithms were executed offline for a range of durations, while the forward search algorithms were allowed to search out to a pre-defined depth. Here, depth refers to the primitive action depth in the RTBSS algorithm, and the macro-action depth in the macro-action algorithms

	Depth 1, Samples 50		Depth 2, Samples 50		Depth 3, Samples 50		Depth 4, Samples 20	
	Ave rewards	Online time (s)	Ave rewards	Online time (s)	Ave rewards	Online time (s)	Ave rewards	Online time (s)
MAC	4.61	0	**9.63**	0.022	13.68	15.39	15.55	660.50
PBD	4.61	0	7.73	**0.002**	14.49	**1.26**	15.54	**75.06**
MAD	4.61	0	7.51	0.0083	**15.88**	4.81	**18.27**	225.74

Table 3: Performance of macro-action algorithms with different macro-action depth on ISRS. At depth 4, a smaller number of posterior beliefs were sampled or computational reasons

(MAC, PBD and MAD). In addition, a pre-defined number of samples were used to obtain posterior beliefs after every action/macro-action. We abuse notation here slightly by using samples to refer to observations in the RTBSS algorithm, observation sequences in the MAD and MAC algorithms, and to samples from the posterior belief distribution in the PBD algorithm.

The three macro-action algorithms do significantly better than the other benchmark solvers, demonstrating that approaches that can search deeper can yield better policies on this domain. Figure 5(b) and 5(c) compare the policies generated by the SARSOP algorithm and the PBD algorithm in the ISRS problem. Both SARSOP and HSVI2 explore parts of the belief space guided by an upper bound on belief-action values. As it takes a long looakahead to realize that visiting beacons and then rocks has a higher value that visiting rocks, it will take a long time for SARSOP and HSVI2 to sample the beliefs that will lead to them computing a higher-value policy. In the considerable offline computation time provided, Both SARSOP and HSVI2 did not discover that it is valuable for the agent to make a detour to the information beacons before approaching the rocks. Instead, they directly approach the rocks and make decisions based on the noisy observations that are obtained due to the large distance from the information beacons.

In contrast, macro-actions allow our forward search approaches, such as PBD, to uncover the potential value of moving to an information beacon. This allows our macro-action forward search approaches to perform much better than prior primitive-action approaches. Figure 5(c) shows that a PBD agent's policy involves visiting some of the information beacons to gather information about which of the rocks are good (white circles), before traveling to those rocks to sample them. In this domain, MAD does particularly well since it exploits the fully-factorable, discrete-state problem specification, whereas the parametric approaches must approximate the world models during planning. Nevertheless, the difference amongst the macro-action algorithms is not significant.

Table 3 compares the different rewards obtained by the macro-action algorithms for different macro-action depths, as well as the time taken by the planner to return a decision at every time step. The sharp performance jump that occurs when the macro-action search depth is increased from 2 to 3 emphasizes the need to search to a longer horizon in the ISRS problem before a good policy can be generated. However, the computational cost of the algorithms also increases exponentially with the macro-action search depth.

Next we examine the relative performance and computational cost of PBD, MAC and MAD, as the number of samples change (Table 4). Sampling allows the macro-action forward search algorithms to consider different conditional plans based on the prior macro-action and potential observation sequence implicitly received. PBD scales best of the three algorithms as the number of samples increases, as predicted by our earlier computational complexity analysis, since it does not

	5 Samples		50 Samples		100 Samples		500 Samples	
	Ave rewards	Online time (s)	Ave rewards	Online time (s)	Ave rewards	Online time (s)	Ave rewards	Online time (s)
MAC	12.76	0.15	13.68	15.39	12.47	58.90	12.94	1732.52
PBD	12.92	**0.035**	14.49	**1.26**	14.56	**4.52**	15.36	**108.64**
MAD	**15.31**	0.056	**15.88**	4.81	**15.57**	20.72	**16.32**	552.64

Table 4: Performance of macro-action algorithms in ISRS with different number of samples

have to perform belief updates along each sampled trajectory explicitly. In general, performance improves with more samples, although the improvement was not statistically significant in the ISRS problem. However, when a decision-making under uncertainty problem requires a large number of posterior beliefs to be sampled after every macro-action, the PBD algorithm results in a lower time complexity for the same number of samples.

The macro-action forward search nature of our algorithm allows us to scale to much larger versions of the Rocksample problem, since unlike offline techniques, it is unnecessary to generate a policy that spans the entire belief space. We implemented the macro-action algorithms on an ISRS problem with a 100 by 100 grid and 30 rocks, a problem domain that far exceeds any problem that can be solved by a traditional POMDP solver.[4] Table 5 compares the results of the three macro-action algorithms to the fully observable value, which provides a strict upperbound of the maximum possible reward for the problem. Such large problems also underscore the value of having macro-actions to limit the branching factor of the forward search.

	ISRS[100,30]	
	Ave rewards	Online time (s)
MAC(d3,s5)	42.64	310.05
PBD(d3,s5)	43.68	**60.81**
MAD(d3,s5)	**51.70**	101.92
Fully obs.	66.61	N.A.

Table 5: Performance of macro-action algorithms on a larger ISRS problem.

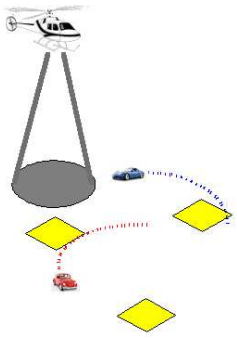


Figure 6: TARGETREPORT problem. A helicopter must track multiple targets moving with noisy dynamics. The field-of-view of the agent's sensor (shaded circle) increases with the agent's altitude.

4. Standard error values for the larger rocksample problems are not presented due to the small number of trials that were performed. These results are primarily meant to illustrate the applicability of the macro-action algorithms on much larger POMDP problems.

5.2 Target monitoring

We next consider a target-tracking problem related to those studied in the sensor resource management literature. In this problem (Figure 6), a helicopter agent has to track multiple targets that are moving independently with noisy dynamics. The helicopter operates in 3D space, while the targets move on the 2D ground plane. The helicopter is equipped with a downward-facing camera for monitoring the environment, and if a target is within the field-of-view of the camera sensor, the agent receives a noisy observation of the location and orientation of the target. We assume for simplicity that the observations of each target are unique, and therefore ignore the data association problem that has been addressed elsewhere.

The noise associated with the agent's observation of a target depends on the agent's position relative to the target. When the helicopter is close to the ground it can only observe a small region, but can determine the position of objects within that small region to a high level of accuracy. When the helicopter flies at a higher altitude, it can view a wider region of the environment, but its measurements will be less precise. Similarly, the closer the helicopter is to a particular target, the more accurate the helicopter's observation of that target is expected to be. Reflecting this intuition, we use a Gaussian observation model where the noise covariance is a function of the position of the helicopter and target: details of this sensor model are provided in Appendix C. One desirable attribute of our sensor model is that if the helicopter is very uncertain about a target's location, even if the helicopter is close to the target's mean location, a single observation is unlikely to localize the target. If the target location is very uncertain, there is a low probability that the target is within the helicopter's good field of view.

Each target's motion is determined by its translational and rotational velocities. The model provides the agent with a prior over these velocities, but at every time step, the target's true velocities are additive functions of these fixed input controls and Gaussian noise. In the parametric formulation, the agent maintains a Gaussian belief over each target's state, and in order to compare MAD, the discrete formulation of the problem discretizes the continuous state spaces of the agent's and targets' positions, and maintains a probability distribution over each discrete target state.

We focus on a particular decision-theoretic version of the sensor resource management problem, where at each time step the agent must decide if each of the targets is inside an area of interest. These areas of interest are indicated by the yellow squares in Figure 6. The agent receives a positive reward if it successfully reports that a target is in an interest region, a negative reward if it wrongly decides that the target is in the region, and no reward if it decides that the target is not in the region, regardless of the target's actual state. We call this the TARGETREPORT problem.

Macro-actions were generated by computing the sequence of actions that will enable the agent to hover at different altitudes over the means of each target belief, as well as a hovering macro-action that consists of hovering at the agent's current location for a few time steps. We compare the macro-action algorithms to a range of intuitive strategies and prior approaches. The first algorithm is the greedy strategy, which returns the primitive action that results in the largest expected reward in the next step. The next two approaches are the Worst Target (WT) policies, which are hand-coded policies of traveling to the target that has the largest uncertainty of all the targets being tracked. The intuition is that the agent's goal in general is to localize the targets in the environment. The two algorithms differ based on whether the agent chooses a new target to travel to after each time step (WT-single), or replans only after it has reached the target it had initially chosen (WT-macro). Finally, we compared our algorithm to the nominal belief optimization (NBO) algorithm

	1 Target		2 Targets		3 Targets		8 Targets	
	Ave rewards	Online time (s)	Ave rewards	Online time (s)	Ave rewards	Online time (s)	Ave rewards	Online time (s)
Greedy	0.56	0.040	2.00	0.19	3.00	0.391	13.75	2.64
WT-Single	**51.51**	0.00031	30.11	0.00067	25.55	0.00065	53.58	0.00077
WT-Macro	50.40	0.00033	20.74	0.00036	27.98	0.00030	47.33	0.00035
NBO	8.01	0.024	28.98	0.21	10.88	0.57	88.27	8.43
MAD(d2,s3)	4.29	2.53	-6.18	7.28	-11.3	18.67	—	—
MAC(d2,s10)	33.54	2.26	84.81	22.13	70.41	64.51	144.31	1042.98
PBD(d2,s10)	41.76	0.42	**98.87**	4.11	**88.32**	11.82	**160.57**	192.12

Table 6: TARGETREPORT Results. Run for 200 time steps.

proposed by Scott et al. (2009). The NBO algorithm also assumes a Kalman Filter model for the target-tracking problem, but rather than considering the entire distribution of posterior beliefs, only the most likely posterior belief after an action is considered. In this algorithm, the most likely posterior belief for a Gaussian belief update is given by the posterior mean without incorporating any observations, and the covariance by performing the covariance update while linearizing about the most likely mean at each step. Although the original algorithm uses an optimization approach to search for action sequences, here we modify the NBO algorithm by adopting a forward search approach, evaluating a macro-action based on the most likely posterior belief.

Table 6 presents results for the TARGETREPORT problem, comparing the algorithms in scenarios with different number of targets.[5] These results demonstrate that the PBD algorithm, with its closed form representation of the distribution of posterior beliefs after an action, finds a significantly better policy than alternate approaches. Figure 7 demonstrates a typical policy executed by the PBD algorithm. The agent begins in the middle of the grid world, and approaches a target at a high altitude (Figure 7(b)), maximizing the likelihood of localizing that target. If none of the targets seem to be approaching a region of interest, the agent hovers in the same position to conserve energy (Figure 7(c)). When one of the targets may potentially be entering a region of interest, the agent focuses on that target, tracking it carefully to ensure that it knows when the target is exactly in the region of interest (Figure 7(d),(e)). The agent subsequently travels to a high altitude and repeats the process of localizing another target with potential rewards (Figure 7(f)).

Considering the entire distribution of posterior beliefs, rather than just the maximum likelihood posterior belief, is valuable because the agent is able to reason that there is a possibility that the target could be within a region of interest. This is in contrast to the NBO approach, which because it only considers the most likely posterior belief, will in general seek to localize the target only if the mean of its belief appears to be heading into a region of interest. While the consideration of the entire distribution of posterior beliefs necessarily incurs greater computational cost, we demonstrate in Section 5.4 that we are able to track two targets in real-time using an implementation of the PBD algorithm that has not been optimized for speed.

Table 6 also shows that because the PBD algorithm directly computes the distribution of posterior beliefs after a macro-action, the computational cost of the PBD algorithm is significantly lower than the MAC algorithm. The MAC algorithm suffers a greater computation cost as it generates the set of posterior beliefs after a macro-action by sampling observation sequences and explicitly

5. The comparison with MAD for the 8-target scenario was not performed for computational memory reasons

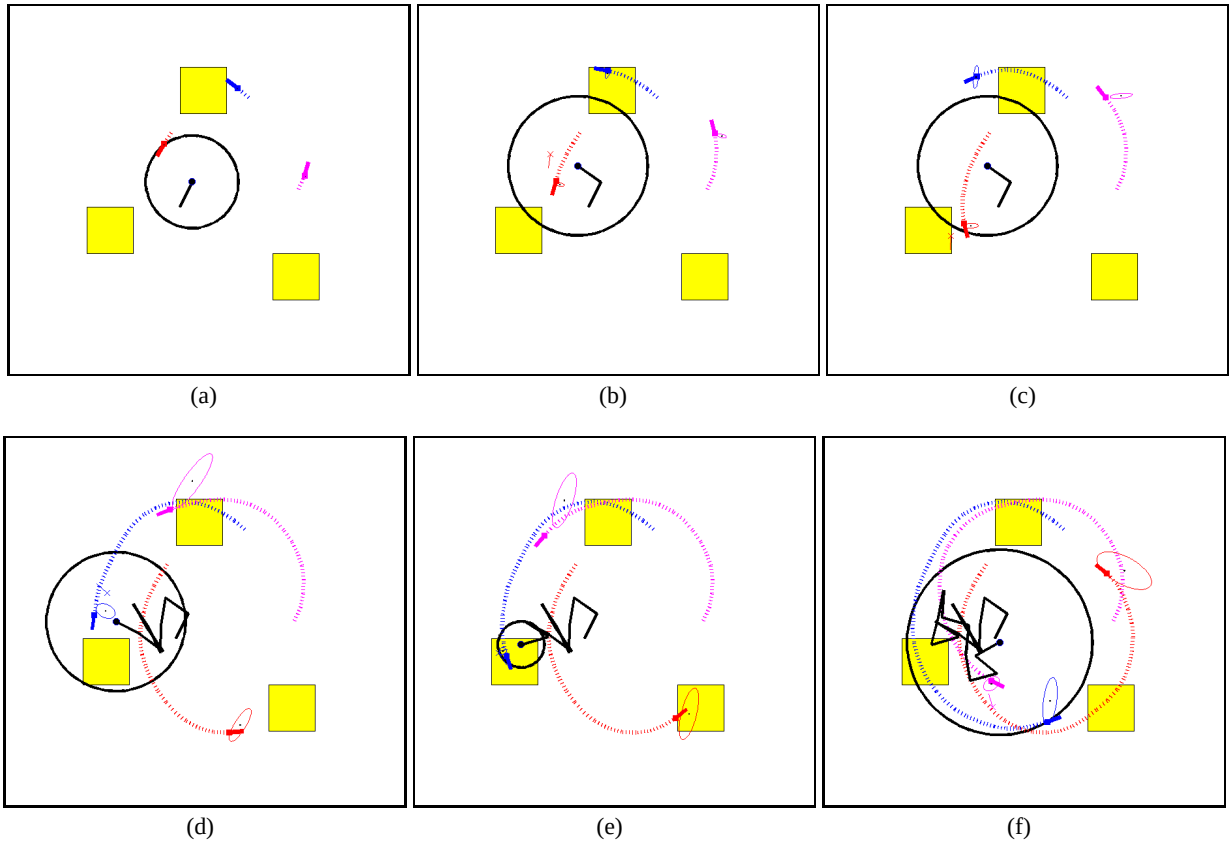


Figure 7: Snapshots of the PBD policy being executed. The black circle indicates the field-of-view of the agent's sensor, which is directly proportionate to the agent's height. The size of the error ellipses indicate the agent's uncertainty associated with each target at each time step. The agent alternates between flying at a high altitude to maximize the likelihood of observing targets (b),(f) and focusing on a single target that is near/has entered an area of interest (e).

performs belief updates along each sample trajectory. In addition, because the TARGETREPORT problem has a state space that is fundamentally continuous, a reasonable discretization of the state space is unfortunately still unable to capture the inherent characteristics of the target-tracking problem, resulting in the poor performance of MAD in the TARGETREPORT problem.

In the single-target case, we also observed the result that the PBD algorithm does worse than the hand-coded policy of the agent traveling to the target with the largest uncertainty (WT-single). When the problem only involves a single target, such a policy equates to having the agent hover over the sole target at every step, which is the optimal policy in the single target case. In contrast, we observe that the MAC and PBD algorithms return policies that result in the agent periodically leaving the target to fly to a higher altitude, resulting in greater noise in the observations and corresponding loss of rewards on average. By restricting the MAC and PBD algorithms to plan with macro-actions, we are restricting the agent to plan with open-loop action sequences in order to perform deeper

search, rather than a conditional plan that is conditioned on the observations after each *primitive* action. Even though the agent replans after every time step, without this conditional plan, an agent executing the MAC or PBD algorithms will execute the "safe" policy and fly to a higher altitude, which maximizes the likelihood of keeping the target well-localized when it is unable to condition its actions based on subsequent observations. This example highlights the tradeoff we make by considering a smaller class of policies (those that can be expressed as chains of macro-actions) compared to the full policy set. While in simple problems, such as a single-target TARGETREPORT problem, the policy restriction can clearly be a limitation, our macro-action algorithms perform significantly better than the other benchmark approaches when there are multiple targets, in scenarios that are arguably more complicated and require more sophisticated planning algorithms.

5.3 Benchmark problems directly from the literature

For completeness, we also examine the performance of our macro-action algorithms on variants of the rocksample and target-tracking problems that are more commonly addressed in their respective research communities. The FieldVisionRockSample (FVRS) problem is equivalent to an instantiation of ISRS where the information beacons are located in the same position as the rock they represent. Table 7 shows that the MAD algorithm performs similarly to existing offline POMDP solvers, while the MAC and PBD algorithms do slightly worse due to the approximations necessary to perform the parametric belief updates. Nevertheless, in the FVRS problem formulation, information gathering actions are similar to reward exploitation actions, making long-horizon planning unnecessary for generating optimal policies. Table 7 shows that a naïve QMDP solver does reasonably well on the FVRS problem, and since QMDP assumes that the problem becomes fully observable after a single action, its good performance suggests that it is unnecessary to explicitly account for uncertainty when making decisions in the FVRS problem. A greedy policy of finding the shortest path to all the rocks becomes a good approximation to the optimal policy.

Within the target-tracking domain, target-surveying is a popular application where an agent is rewarded for maintaining a small uncertainty over the targets positions: the reward model is proportional to the negative sum of the trace of the target belief covariances. Table 8 shows that the MAC and PBD algorithms perform approximately as well as NBO, which only considers the most likely posterior belief. This is not too surprising as the reward function is directly a function of the covariance, which is independent of the actual observations received. In this case, having a distribution over the posterior beliefs, and specifically over the posterior means, which are explicitly computed in the MAD, MAC and PBD algorithms, does not provide an advantage over the simpler NBO approach.

5.4 Real-world Helicopter experiments

Finally, as a proof of concept, we demonstrate the PBD algorithm on a live instantiation of the TARGETREPORT problem. A motivating application for this tracking problem is our involvement (He et al., 2009) in the 1st US-Asian Demonstration and Assessment of Micro Aerial Vehicle (MAV) and Unmanned Ground Vehicle (UGV) Technology (MAV08 competition). The mission was a hostage rescue scenario, where an aerial vehicle had to guide ground units to a hostage building while avoiding an enemy guard vehicle. Our aerial vehicle therefore had to plan paths in order to be able to monitor the different ground objects and report whenever any of them arrived at an area of interest.

(a)

	FVRS[8,5]		
	Ave rewards	Online time (s)	Offline time (s)
QMDP	18.58 $\pm$ 0.49	0	3.00
HSVI2(150s)	18.46 $\pm$ 2.53	0.021	150
SARSOP(150s)	**21.04** $\pm$ 0.076	0.0063	150
RTBSS(d5, s10)	20.12 $\pm$ 0.48	19.04	0
MAC(d2,s200)	17.72 $\pm$ 1.29	0.035	0
PBD(d2,s200)	19.85 $\pm$ 0.43	0.004	0
MAD(d2,s200)	20.69 $\pm$ 0.27	0.015	0
Fully obs.	21.37	N.A.	N.A.

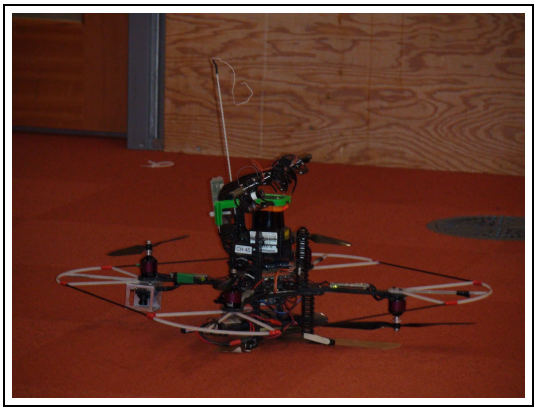
(b)

	FVRS[100,30]	
	Ave rewards	Online time (s)
MAC(d3,s5)	59.51	104.04
PBD(d3,s5)	59.57	**19.48**
MAD(d3,s5)	**65.87**	32.17
Fully obs.	66.61	N.A.

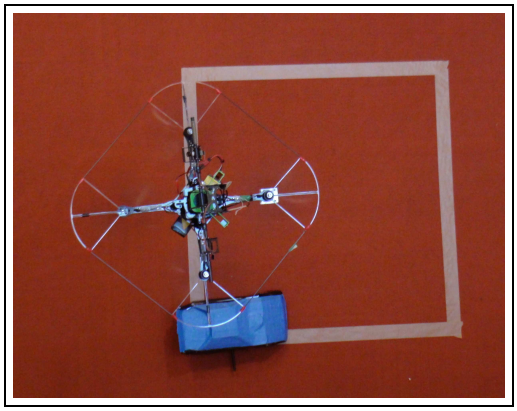
Table 7: FVRS results. For the forward search algorithms, the numbers in brackets represent the search depth (d) and number of posterior beliefs obtained (s) at the end of each action/macro-action. Standard error values for FVRS[8,5] are shown.

	1 Target		2 Targets		3 Targets		8 Targets	
	Ave rewards	Online time (s)	Ave rewards	Online time (s)	Ave rewards	Online time (s)	Ave rewards	Online time (s)
Greedy	-17.42	0.0057	-117.79	0.0082	-172.84	0.0096	-12782.81	0.063
WT-Single	-6.41	0.00095	-111.00	0.00065	-187.81	0.00045	-15210.14	0.00066
WT-Macro	**-6.40**	0.00090	-99.97	0.00019	-178.30	0.00013	-15109.16	0.00036
NBO	**-6.40**	0.046	**-99.74**	0.12	**-165.21**	0.21	-13289.39	3.78
MAD(d2,s3)	-23.03	6.30	-149.32	23.50	-214.25	53.26	—	—
MAC (d2,s10)	**-6.40**	4.77	-102.48	15.66	-165.33	28.22	-13477.92	538.13
PBD (d2,s10)	-6.41	0.40	-102.21	1.17	-165.32	2.00	**-12219.17**	37.62

Table 8: Target survey problem. Run for 200 time steps.

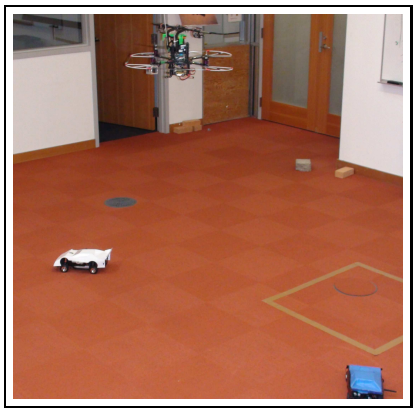


(a) Our Quadrotor helicopter

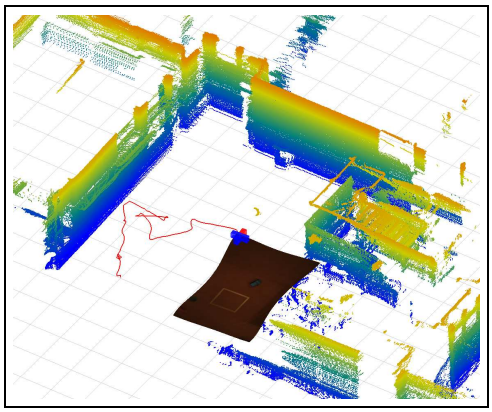


(b) Helicopter tracking car through area of interest

Figure 8: Target monitoring demonstration with helicopter.



(a) Multiple cars being tracked.

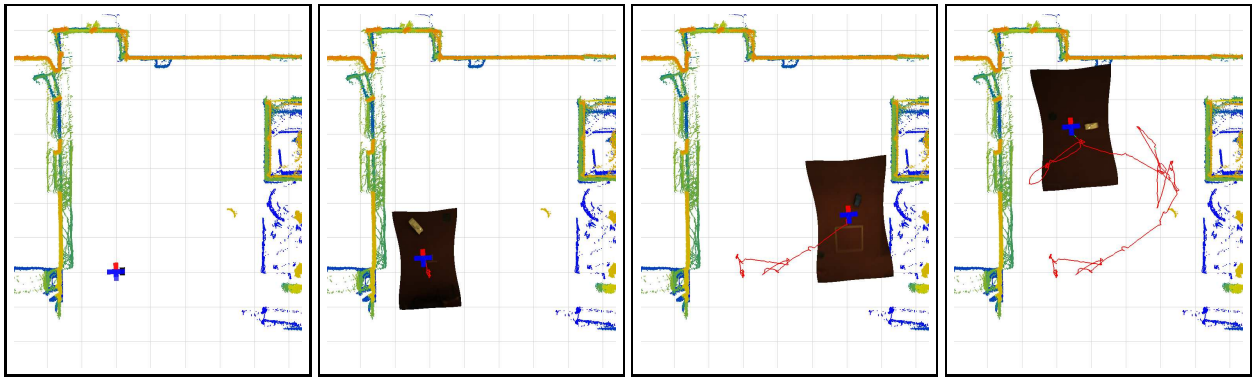


(b) Visualization of helicopter's path in 3D

Figure 9: The helicopter has to track two cars simultaneously, and report whenever either car enters an area of interest (brown square). b) The helicopter (blue/red cross) uses an onboard laser scanner to localize itself. A downward pointing camera is used to observe the ground targets (image projected onto the ground plane here, resulting in a curved image).

We demonstrate this scenario on an actual helicopter platform tracking multiple ground vehicles in an indoor environment (Figure 9(a)). In previous work (He et al., 2008; Bachrach et al., 2009), we developed a quadrotor helicopter (Figure 8(a)) that is capable of autonomous flight in unstructured and unknown indoor environments. The helicopter uses a laser rangefinder to localize itself in the environment.

We mounted a downward-facing camera that makes observations of the target. Since target detection is not the focus of this paper, each of the ground vehicles had a known, distinctive color, making them easy to detect and distinguish with a simple blob detection algorithm. Given the helicopter's position in the world and the image coordinates of the detected object, we can recover an estimate of the position and orientation of a target observation in global coordinates. Nevertheless, the helicopter only receives an observation of the target when the target is within the camera's field-



(a) (b) (c) (d)

Figure 10: Bird's eye-view snapshots of the helicopter's trajectory (red), based on policy generated by the PBD algorithm. The helicopter (blue/red cross) alternates between observing the white (b,d) and blue (c) cars in order to accurately report when either car is in an area of interest. The field-of-view of the agent's camera sensor varies directly with the height that the agent is flying at.

	# Target entry detections	# True target entries	Flight time (s)	Dist. traveled (m)
WT-Single	1	7	484.15	243.36
NBO	1	4	435.25	247.01
PBD	**4**	6	474.64	282.51

Table 9: Performance of algorithms on real-world helicopter experiment. Ground truth was found using an overhead video camera.

of-view, and although the helicopter platform hovers relatively stably, slight oscillations persist, which result in noisier observations when the helicopter is flying at higher altitudes. Hence, the helicopter has to choose actions that balance between obtaining more accurate observations at low altitudes and a larger field-of-view by flying high.

Two ground vehicles were driven autonomously in the environment with open-loop control, and the helicopter had to plan actions that would accurately localize both targets. To replicate the TARGETREPORT problem, we marked out three areas of interest where the helicopter had to predict at every time step if the targets are within those areas (Figure 8(b)). We applied the PBD algorithm to plan paths for the helicopter that maximize the likelihood that it can accurately report whenever a target is in an area of interest.

Figure 9(b) shows a 3D view of the helicopter performing autonomous target tracking of the ground targets. As the helicopter flies around the environment, it obtains observations of the target, which are then used to update the agent's belief of the targets. Figure 10 provides snapshots of the helicopter executing a plan that is computed online by the PBD algorithm. The helicopter exhibits similar behaviors to those that were observed in the simulation experiments. The helicopter alternates between the two targets in the environment to report when either target is in an area of interest. When the agent had a large uncertainty over a particular target's location, it would also fly

to a higher altitude in order to increase its sensor field-of-view, thereby maximizing the likelihood that it will be able to re-localize the targets.

As a coarse measure of achieved reward, we evaluated how well the helicopter running PBD did at reporting when a target entered an area of interest, and compared it to the WT-Single and NBO algorithms. The ground truth of the number of times the targets actually entered the areas of interests in each trial was found by using a video camera mounted overhead above the environment. Table 9 indicates that the PBD algorithm did a much better job of reporting the targets' positions than both the WT-Single and NBO algorithms. In particular, we observed that both the WT-Single and NBO algorithms seldom took advantage of the ability to increase the agent's sensor field-of-view by having the agent fly to a higher altitude. An agent applying these two algorithms therefore had a higher probability of losing track of the targets completely.

6. Related Work

Decision-making under uncertainty when the states are partially observable is most commonly discussed under the Partially Observable Markov Decision Process (POMDP) framework, though they have also been analyzed in other research domains under similar assumptions. While it is beyond the scope of this paper to provide a comprehensive survey of POMDP techniques, point-based methods such as HSVI2 (Smith & Simmons, 2005) and SARSOP (Kurniawati et al., 2008) are often considered state-of-the-art offline methods, leveraging the piece-wise and convex aspects of the value function to perform value updates at selected beliefs. These approaches assume a discrete-state representation, but offline approaches that use parametric representations have been proposed for continuous-valued spaces (Brooks et al., 2006; Brunskill et al., 2008; Porta et al., 2006).

The ideas in this paper are more closely related to the body of online, forward search POMDP techniques that only compute an action for the current belief: see Ross et al. (2008) for a recent survey. Our PBD algorithm assumes a similar parametric belief representation to some of the prior offline continuous-state approaches, but exploits additional properties of the Gaussian parametric representation to do an efficient forward search.

Macro-actions have been considered in depth within the fully observable Markov decision process community, and are typically known as "options" (Sutton et al., 1999), or posed as part of a semi-Markov decision process (Mahadevan et al., 1997). These prior formalisms for temporally-extended actions include closed-loop policies that persist until a termination state is achieved. Unfortunately, such policies are not tenable in partially observable environments, since in general the agent will only have a probability distribution over whether the agent has reached the terminal state, resulting in a terminal condition that is not well-defined. This may be why there is comparatively less research on macro-actions in POMDPs. However, fixed-length open-loop policies, which we use in this paper, can be easily applied to partially observable domains.

There do exist several offline POMDP approaches that use macro-actions. Pineau, Gordon and Thrun's PolCA (2003) algorithm use a hierarchical approach to solving discrete-state POMDPs. PolCA is guaranteed to be recursively optimal, which means that subtask policies are locally but not globally optimal. Yu et al. (2005) provide an optimal algorithm for planning if no observations were available, but do not provide performance guarantees when some noisy sensor information is received. Foka and Trahanias's (2007) solution involves building a hierarchy of nested representations and solutions. Their focus is on discrete-state problems, particularly navigation applications, and no performance guarantees are provided. Theocharous and Kaelbling's (2003) discrete-state

reinforcement learning approach sample observation trajectories and solve for the expected reward of a discrete set of belief points using function approximation. While their experimental results demonstrate the potential advantages of macro-actions, no bounds are provided on the resulting solution quality. Hsiao, Lozano-Pérez, and Kaelbling (2008) used a form of macro-actions for robot manipulation tasks, but the focus of their work is on robust manipulation under uncertainty, and their work only considers a very short horizon of action trajectories.

Perhaps the most closely related POMDP planner is Kurniawati et al. (2009)'s recent work on using macro-actions to guide the sampling of belief points for use in a point-based POMDP solver, and applied to partially observable motion-planning tasks. The authors prove that if the sampled beliefs cover the reachable belief space sufficiently densely, then the resulting optimal value at the sampled beliefs will be arbitrarily close to the value of nearby beliefs that are not sampled. Macro-actions are generated by sampling states from the state space and generating the set of sequential actions that will get the agent from one state sample to another. Observation sequences are similarly sampled, resulting in a set of reachable beliefs that are then used in the point-based SARSOP POMDP solver. Though experimental results demonstrate significant improvements over prior POMDP solvers, this algorithm is limited to motion planning tasks, and the offline approach also seems likely to struggle with large problems with a high number of state diensions. To our knowledge, macro-actions have not been used in an online, forward search manner for decision-making under uncertainty.

A natural question is whether macro-actions could be used within some of the current state-of-the art point-based solvers, such as SARSOP or HSVI2. HSVI2 and SARSOP both iteratively expand trajectories of belief states until a desired bound on the value function at the initial belief is achieved. HSVI2 and SARSOP also both maintain upper and lower bounds on the value function, and use these bounds to choose action-observation pairs that construct a trajectory of beliefs. Using macro-actions would then remove a key strength of these approaches: the ability to selectively sample actions that are based on the upper bound and lower bounds. These algorithms also rely on an explicit representation of the value function over beliefs by using a set of α-vectors that are created by backing up individual beliefs along the trajectory. Maintaining and updating this value function representation in large problems is expensive: the value function backup process has a computational complexity that scales as the product of the primitive action space, the observation space, the size of the state space squared, and the number of prior α-vectors ($|A||Z||S|^2|\alpha|$). Therefore it seems unlikely that SARSOP or HSVI2 would benefit from using macro-actions in belief space, since that would remove their ability to selectively sample actions but maintain the expense of their value function backups. Developing value function bounds that operate directly on distributions of beliefs, or representing value functions over distributions of beliefs, might be potential ways of incorporating macro-actions within an offline, point-based solver, though that is beyond the scope of the work presented in this paper.

Planning under uncertainty in partially observable domains is also a problem of interest to other research communities, most notably amongst the sensor resource management and controls community. In particular, when the noise model is Gaussian, the dynamics are linear Gaussian, and the reward model is quadratic, then the well-known Linear Quadratic Gaussian (LQG) controller provides an optimal solution. However, most problems do not adhere to all three model assumptions, making the LQG controller inapplicable: our approach provides a bounded solution for a wider range of dynamics and reward models.

In the sensor resource management domain, planning under uncertainty techniques are used in the context of planning sensor placements to track single or multiple targets. In general, due to the computational complexities involved in seeking to compute a policy over the entire state space even for problems of moderate size, researchers here have focused on generating approximate solution techniques by taking a forward search approach. Existing algorithms often adopt a myopic, or greedy strategy when it comes to planning (Krause & Guestrin, 2007), but notable exceptions include Scott et al. (2009) and Kreucher, Hero III, Kastella, and Chang (2004). Kreucher et al. describe a multi-target tracking problem, where non-myopic sensor management is necessary for multi-target tracking. The authors use a particle filter approach to represent the agent's belief of the target's location, and seeks to find paths that will result in the greatest KL divergence in density before and after the measurement. For a non-myopic rollout, this algorithm uses Monte Carlo sampling to sample possible observation outcomes. They also provide an information-directed path searching scheme to reduce the complexity of the Monte Carlo sampling, as well as value heuristics that will help direct the search. It is possible that some of their insights could be used in combination with our macro-action formulation to strengthen both approaches. Scott et al. (2009) directly formulates the target tracking problem as a POMDP, and proposes the Nominal Belief Optimization (NBO) algorithm that computes the most likely belief after an action for deeper forward search. In contrast, our algorithm explicitly computes the entire set of possible posterior beliefs after a macro-action.

Within the controls community, the Model Predictive Control/Receding Horizon Control (MPC/RHC) framework has strong similarities to our approach (see e.g. (Kuwata & How, 2004), (Bellingham et al., 2002), and (Richards et al., 2003)). MPC/RHC involves solving a finite-horizon optimal control problem for the current state, executing the first few controls, and then repeating the optimization based on the latest information available. MPC/RHC algorithms therefore essentially optimize over the forward search space by anchoring the initial condition of the optimization on the current state/belief. The cost beyond the finite horizon is typically estimated by using a cost heuristic computed offline. Several papers (Kuwata & How, 2004; Bellingham et al., 2002) use a geometric set of fixed macro-actions to approximate the control beyond the finite horizon, but these macro-actions are typically not adapted to the particular current beliefs, and are not evaluated as possible action sequences within the finite horizon.

7. Conclusion and future work

In this paper, we have demonstrated the benefits of using macro-actions for performing long-horizon planning in partially observable domains. We have adopted a forward search framework for planning under uncertainty, and in problems where a far lookahead is needed to act well, our macro-action algorithms allow us to restrict the policy space in order to search out to a longer horizon.

We have proposed two novel algorithms that exploit the inherent structure of different problem domains to perform efficient macro-action forward search. Our PBD algorithm exploits the fact that when the agent's belief can be represented as a Gaussian distribution, we can analytically compute the distribution over the set of posterior beliefs that could result from a single macro-action. The analytical computation removes the need to explicitly enumerate the observation sequences, resulting in significant computational savings. Second, in discrete environmental models where we cannot analytically capture the resulting distribution of beliefs after a macro-action, our MAD forward search planner generates observation sequences to produce a particle approximation of the resulting

distribution of beliefs. The MAD algorithm also easily exploits any factored structure that may exist in the problem to perform efficient belief updates, thereby minimizing computational cost.

We have presented theoretical and experimental results evaluating the performance and computational cost of our macro-action algorithms. Our algorithms are applicable in problem domains that span multiple research communities, and consistently perform better than prior approaches in problems where it is necessary to perform far lookahead and/or it is necessary to explicitly account for the distribution of posterior beliefs. Finally, we demonstrated our algorithm on a real robotic helicopter, underscoring the applicability of our algorithm for planning in real-world, long-horizon, partially observable domains.

Nevertheless, there are a number of interesting directions for future work. Our experience with the single-target TARGETREPORT problem suggests that in some scenarios, having access to the set of primitive actions is important, at least for a couple steps of lookahead. One possible extension to this paper would therefore be to perform a short naïve factored full forward search, and then use the macro-actions as a value heuristic at the leaves: an open question here is how to balance between expensive, but more accurate full forward search, and the faster, restricted-policy-class macro-actions. Another direction of interest, inspired by the receding horizon framework, is to explore how to determine what length of macro-actions should be used at every depth of the search. Finally, in this paper we assume that the macro-actions themselves are provided in advance: we are now investigating ways to automatically construct macro-actions.

8. Acknowledgements

We wish to thank Finale Doshi, Alborz Geramifard, Josh Joseph, Brandon Luders, Javier Velez, and Matthew Walter for valuable discussions and feedback. Daniel Gurdan, Jan Stumpf and Markus Achtelik provided the quadrotor helicopter and the support of Ascending Technologies. Abraham Bachrach, Anton De Winter, Garrett Hemann, Albert Huang, and Samuel Prentice assisted with the development of the software and hardware for the helicopter demonstration. We also appreciate early POMDP forward search discussions with Leslie Pack Kaelbling and Tomas Lozano-Perez.

Appendix A: Exponential Family Kalman Filter

Building on statistical economics research for time-series analysis of non-Gaussian observations (Durbin & Koopman, 2000), we present the Kalman filter equivalent for systems with linear-Gaussian state-transitions and observation models that belong to the exponential family of distributions.

The state-transition and observation models can be represented as follows:

$$s_t = A_t s_{t-1} + B_t u_t + \varepsilon_t, \qquad s_{t-1} \sim N(\mu_{t-1}, \Sigma_{t-1}), \qquad \varepsilon_t \sim N(0, P_t) \tag{74}$$

$$p(z_t|\theta_t) = \exp(z_t^T \theta_t - \beta_t(\theta_t) + \kappa_t(z_t)), \qquad \theta_t = W(s_t) \tag{75}$$

For the state-transition model, s_t is the system's hidden state, u_t is the control actions, A_t and B_t are the linear transition matrices, and ϵ_t is the state-transition Gaussian noise with covariance P_t.

The observation model belongs to the exponential family of distributions. θ_t and $\beta_t(\theta_t)$ are the canonical parameter and normalization factor of the distribution, and $W(.)$ maps the states to canonical parameter values. $W(.)$ depends on the particular member of the exponential family. For ease of notation, we let

$$\beta_t(z_t|\theta_t) = -\log p(z_t|\theta_t) = -z_t^T \theta_t + \beta_t(\theta_t) + \kappa_t(z_t) \tag{76}$$

Following the traditional Kalman filter, the process update can be written as

$$\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t, \qquad \overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + P_t \tag{77}$$

where $\overline{\mu}_t$ and $\overline{\Sigma}_t$ are the mean and covariances of the posterior belief after the process update but before the measurement udpate. For the measurement update, we seek to find the conditional mode

$$\mu_t = \arg\max_{s_t} p(s_t|z_t) \tag{78}$$

$$= \arg\max_{s_t} p(z_t|s_t)\overline{b}(s_t) \qquad \text{(Bayes rule)} \tag{79}$$

$$= \arg\max_{s_t} p(z_t|\theta_t)\overline{b}(s_t) \tag{80}$$

$$= \arg\max_{s_t} \exp(-J_t), \qquad \text{where } J_t = -\log p(z_t|\theta_t) + \frac{1}{2}(s_t - \overline{\mu}_t)^T \overline{\Sigma}_t^{-1}(s_t - \overline{\mu}_t) \tag{81}$$

$$\Rightarrow \qquad 0 = \frac{\partial J_t}{\partial s_t}\Big|_{s_t=\mu_t} = \frac{\partial \beta_t(z_t, \theta_t)}{\partial \theta_t}\frac{\partial \theta_t}{\partial s_t} + \overline{\Sigma}_t^{-1}(\mu_t - \overline{\mu}_t), \tag{82}$$

Taking the derivative of $\theta_t = W(s_t)$ about the prior mean $\overline{\mu}_t$, we let

$$Y_t = \frac{\partial W(s_t)}{\partial s_t}\Big|_{s_t=\overline{\mu}_t} \tag{83}$$

Similarly, performing Taylor expansion on $\frac{\partial \beta_t(z_t|\theta_t)}{\partial \theta_t}$ about $\overline{\theta}_t = W(\overline{\mu}_t)$,

$$\frac{\partial \beta_t(z_t|\theta_t)}{\partial \theta_t} = \frac{\partial \beta_t(z_t|\theta_t)}{\partial \theta_t}\bigg|_{\theta_t=\overline{\theta}_t} + \frac{\partial^2 \beta_t(z_t|\theta_t)}{\partial \theta_t \partial \theta_t^T}\bigg|_{\theta_t=\overline{\theta}_t} (\theta_t - \overline{\theta}_t) \tag{84}$$

$$\frac{\partial \beta_t(z_t|\theta_t)}{\partial \theta_t} = \dot{\beta}_t + \ddot{\beta}_t(\theta_t - \overline{\theta}_t) \tag{85}$$

$$\text{where} \qquad \dot{\beta}_t = \frac{\partial}{\partial \theta_t}(-z_t^T\theta_t + \beta_t(\theta_t) - \kappa_t(z_t))\bigg|_{\theta_t=\overline{\theta}_t}, \qquad \text{(Eqn. 76)} \tag{86}$$

$$= \frac{\partial \beta_t(\theta_t)}{\partial \theta_t}\bigg|_{\theta_t=\overline{\theta}_t} - z_t \tag{87}$$

$$\dot{\beta}_t = \dot{\beta}_t - z_t \tag{88}$$

$$\text{and} \qquad \ddot{\beta}_t = \frac{\partial^2 \beta_t(z_t|\theta_t)}{\partial \theta_t \partial \theta_t^T}\bigg|_{\theta_t=\overline{\theta}_t} (\theta_t - \overline{\theta}_t) \tag{89}$$

$$\ddot{\beta}_t = \ddot{\beta}_t \tag{90}$$

Plugging Equations 88 and 90 into Equation 85, and then into Equation 82,

$$Y_t\ (\dot{\beta}_t - z_t + \ddot{\beta}_t(\theta_t - \overline{\theta}_t)) = -\ \overline{\Sigma}_t^{-1}(\mu_t - \overline{\mu}_t) \tag{91}$$

$$Y_t\ \ddot{\beta}_t(\ddot{\beta}_t^{-1}(\dot{\beta}_t - z_t) - \overline{\theta}_t + \theta_t) = -\ \overline{\Sigma}_t^{-1}(\mu_t - \overline{\mu}_t) \tag{92}$$

$$Y_t\ \ddot{\beta}_t((\overline{\theta}_t - \ddot{\beta}_t^{-1}(\dot{\beta}_t - z_t)) - \theta_t) = \overline{\Sigma}_t^{-1}(\mu_t - \overline{\mu}_t) \tag{93}$$

$$Y_t\ \ddot{\beta}_t(\tilde{z}_t - W(s_t)) = \overline{\Sigma}_t^{-1}(\mu_t - \overline{\mu}_t) \tag{94}$$

where $\tilde{z}_t = (\overline{\theta}_t - \ddot{\beta}_t^{-1}(\dot{\beta}_t - z_t))$ is the projection of the observation onto the parameter space of the exponential family distribution, and is independent of s_t. In Equation 94 we substituted θ_t using Equation 75.

Mean Update

Using Equation 94 and substituting μ_t for s_t,

$$\overline{\Sigma}_t^{-1}(\mu_t - \overline{\mu}_t) = Y_t\ \ddot{\beta}_t(\tilde{z}_t - W(\mu_t)) \tag{95}$$

$$= Y_t\ \ddot{\beta}_t(\tilde{z}_t - W(\mu_t)) + W(\overline{\mu}_t) - W(\overline{\mu}_t) \tag{96}$$

$$= Y_t\ \ddot{\beta}_t(\tilde{z}_t - W(\overline{\mu}_t)) - Y_t\ \ddot{\beta}_t(W(\mu_t) - W(\overline{\mu}_t)) \tag{97}$$

Linearizing $W(s_t)$ about $\overline{\mu}_t$,

$$W(s_t) = W(\overline{\mu}_t) + W'(s_t)_{s_t=\overline{\mu}_t}(s_t - \overline{\mu}_t) \tag{98}$$

$$= W(\overline{\mu}_t) + Y_t\ (\mu_t - \overline{\mu}_t) \tag{99}$$

$$\Rightarrow \qquad \overline{\Sigma}_t^{-1}(\mu_t - \overline{\mu}_t) = Y_t\ \ddot{\beta}_t(\tilde{z}_t - W(\overline{\mu}_t)) - Y_t\ \ddot{\beta}_t Y_t\ (\mu_t - \overline{\mu}_t) \tag{100}$$

$$Y_t\ \ddot{\beta}_t(\tilde{z}_t - W(\overline{\mu}_t)) = (\overline{\Sigma}_t^{-1} + Y_t\ \ddot{\beta}_t Y_t\ )(\mu_t - \overline{\mu}_t) \tag{101}$$

$$= \Sigma_t^{-1}(\mu_t - \overline{\mu}_t) \tag{102}$$

$$\Rightarrow \qquad \mu_t - \overline{\mu}_t = \Sigma_t Y_t\ \ddot{\beta}_t(\tilde{z}_t - W(\overline{\mu}_t)) \tag{103}$$

where $\Sigma_t Y_t \ \ddot{\beta}_t = \tilde{K}_t$ is the Kalman gain for non-Gaussian exponential family distributions. Via a standard transformation, the Kalman gain can be written in terms of covariances other than Σ_t,

$$\tilde{K}_t = \overline{\Sigma}_t Y_t \ (Y_t \overline{\Sigma}_t Y_t + \ddot{\beta}_t^{-1})^{-1} \tag{104}$$

$$\text{and} \qquad \mu_t = \overline{\mu}_t + \tilde{K}_t(\tilde{z}_t - W(\overline{\mu}_t)) \tag{105}$$

Covariance Update

Given a Gaussian posterior belief, $\frac{\partial^2 J}{\partial s_t^2}$ is the inverse of the covariance of the agent's belief

$$\Sigma_t^{-1} = \frac{\partial^2 J}{\partial s_t^2} \tag{106}$$

$$= \frac{\partial}{\partial x}(\overline{\Sigma}_t^{-1}(s_t - \overline{\mu}_t) - Y_t \ \ddot{\beta}_t(\tilde{z}_t - W(s_t))) \tag{107}$$

$$= \overline{\Sigma}_t^{-1} + Y_t \ \ddot{\beta}_t Y_t \tag{108}$$

$$\Rightarrow \Sigma_t = (\overline{\Sigma}_t^{-1} + Y_t \ \ddot{\beta}_t Y_t)^{-1} \tag{109}$$

Appendix B. Rock Sample Observation Model

The Bernoulli observation function can be written as follows if we let $d_{i,t} = \| \ r_t - RB_i \ \|_2$,

$$p(Z_{i,t}|RV_{i,t} = m_{i,t}, r_t, RB_i) \tag{110}$$

$$= (0.5 + (m_{i,t} - 0.5)2^{-d_{i,t}/D_0})^{Z_{i,t}}(0.5 - (m_{i,t} - 0.5)2^{-d_{i,t}/D_0})^{1-Z_{i,t}} \tag{111}$$

$$= \exp(Z_{i,t} \ln \frac{0.5 + (m_{i,t} - 0.5)2^{-d_{i,t}/D_0}}{0.5 - (m_{i,t} - 0.5)2^{-d_{i,t}/D_0}} + \ln(0.5 - (m_{i,t} - 0.5)2^{-d_{i,t}/D_0})) \tag{112}$$

$$= \exp(Z_{i,t}\theta_t - \beta_t(\theta_t)) \tag{113}$$

We therefore have the parameters of the exp. family observation model

$$\theta_{i,t} = W(m_{i,t}, r_t, RB_i) \tag{114}$$

$$= \ln \frac{0.5 + (m_{i,t} - 0.5)2^{-d_{i,t}/D_0}}{0.5 - (m_{i,t} - 0.5)2^{-d_{i,t}/D_0}} \tag{115}$$

$$\beta_{i,t} = -\ln(0.5 - (m_{i,t} - 0.5)2^{-d_{i,t}/D_0}) \tag{116}$$

$$= \ln(\exp(\theta_{i,t}) + 1) \tag{117}$$

We can then derive the derivatives $Y_{i,t}$ and $\ddot{\beta}_{i,t}$

$$Y_t = \frac{\partial W(m_{i,t}, r_t, RB_i)}{\partial m_{i,t}}\Big|_{m_{i,t}=\hat{m}_{i,t}} \tag{118}$$

$$= \frac{\partial}{\partial m_{i,t}} \ln \frac{0.5 + (m_{i,t} - 0.5)2^{-d_{i,t}/D_0}}{0.5 - (m_{i,t} - 0.5)2^{-d_{i,t}/D_0}}\Big|_{m_{i,t}=\hat{m}_{i,t}} \tag{119}$$

$$= \frac{2^{-d_{i,t}/D_0}}{0.5 + (\hat{m}_{i,t} - 0.5)2^{-d_{i,t}/D_0}} \cdot \frac{1}{0.5 - (\hat{m}_{i,t} - 0.5)2^{-d_{i,t}/D_0}} \tag{120}$$

$$\Rightarrow \qquad \beta_{i,t} = \ln(\exp(\theta_{i,t}) + 1) \tag{121}$$

$$\ddot{\beta}_{i,t} = \frac{\partial^2 b_{i,t}}{\partial \theta_{i,t}^2}\Big|_{\theta_{i,t}=\hat{\theta}_{i,t}} \tag{122}$$

$$= \frac{\exp(\hat{\theta}_{i,t})}{\exp(\hat{\theta}_{i,t}) + 1} - \frac{\exp(2\hat{\theta}_{i,t})}{(\exp(\hat{\theta}_{i,t}) + 1)^2} \tag{123}$$

Appendix C. Target Tracking Observation Model

We adopt an observation model for target tracking where the target observation obtained has Gaussian noise and the noise covariance is a function of the position of the helicopter and target:

$$\begin{bmatrix} z_{xi} \\ z_{yi} \\ z_{\theta i} \end{bmatrix} = f\left(\begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix}\right) + \mathcal{N}(0, \Sigma_{zi})$$
$$\Sigma_{zi} = f(x_i, y_i, x_a, y_a, h_a)$$

where x_i, y_i, θ_i is the pose of target i, while x_a, y_a, h_a correspond to the agent's position and height in the environment.

The covariance function itself is specified as

$$f(x_i, y_i, x_a, y_a, h_a) = C_1 h_a + C_2 \frac{\left(\begin{bmatrix} x_i \\ y_i \end{bmatrix} - \begin{bmatrix} x_a \\ y_a \end{bmatrix}\right)\left(\begin{bmatrix} x_i \\ y_i \end{bmatrix} - \begin{bmatrix} x_a \\ y_a \end{bmatrix}\right)^T}{h_a} + C_3$$

where C_1, C_2 and C_3 are constants.

In the generic belief update expression where the target position is unknown,

$$b^{az}(s') \propto \mathcal{N}(z|s', \Sigma_{zi}) \int_s p(s'|s, a)b(s)ds \quad s.t. \quad \int_{s'} b^{az}(s')ds' = 1,$$

this means that each possible s' would be associated with a different covariance Σ_{zi}. Performing this integration exactly would not keep the distribution Gaussian. Instead, we approximate the observation model by computing a single expected covariance $\tilde{\Sigma}_{zi}$ given the current belief distribution:

$$E[\Sigma_{zi}] = \int_s b(s)\Sigma_{zi}(s)ds.$$

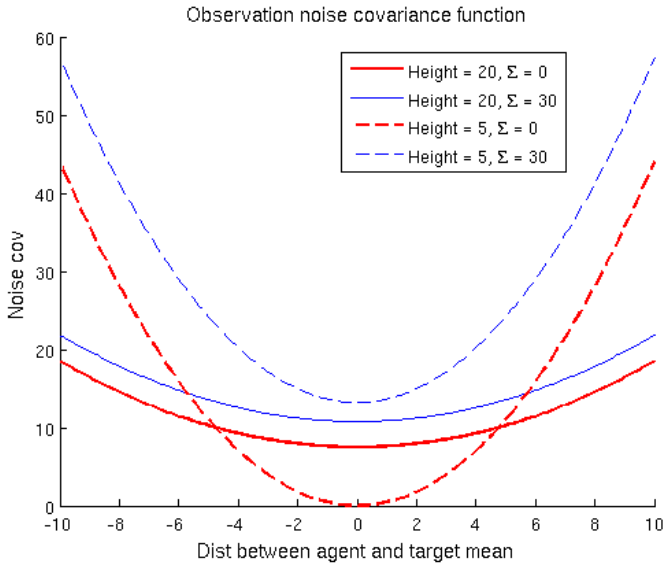


Figure 11: The observation noise covariance is a function of the height of helicopter, the distance between the helicopter and the mean of the target belief, and the covariance of the target belief. At lower altitudes, the helicopter can make better observations of targets close to it, but has a limited field of vision. At higher heights, the helicopter can see a larger area but even close targets are more noisily observed.

Substituting in the exact expressions for the measurement-updated belief and covariance function, we get:

$$E[\Sigma_{zi}] = \int \mathcal{N}\left(\begin{bmatrix} x_i \\ y_i \end{bmatrix} \middle| \hat{\mu}_{txy}, \hat{\Sigma}_{txy}\right) \left(C_1 h_a - \frac{C_2}{h_a}\left(\begin{bmatrix} x_i \\ y_i \end{bmatrix} - \begin{bmatrix} x_a \\ y_a \end{bmatrix}\right)\left(\begin{bmatrix} x_i \\ y_i \end{bmatrix} - \begin{bmatrix} x_a \\ y_a \end{bmatrix}\right)^T + C_3\right) dx_i dy_i.$$

and by adding and subtracting $\hat{\mu}_{txy}$ from the second term, reduces to

$$E[\Sigma_{zi}] \quad = C_1 h_a + \frac{C_2}{h_a}\left(\hat{\mu}_{txy} - \begin{bmatrix} x_a \\ y_a \end{bmatrix}\right)\left(\hat{\mu}_{txy} - \begin{bmatrix} x_a \\ y_a \end{bmatrix}\right)^T + \frac{C_2}{h_a}\hat{\Sigma}_{txy}.$$

In contrast to simpler observation models, our observation model has the desirable characteristic that if a target's location is very uncertain, namely its covariance $\hat{\Sigma}_{txy}$ is very large, then even if the target's mean location is close to the helicopter's mean location, the expected benefit of receiving an observation (in terms of reducing the target's uncertainty) is still small. This property comes out automatically from the above derivation, since $E[\Sigma_{zi}]$ includes the current target covariance $\hat{\Sigma}_{txy}$. Figure 11 provides an illustration of the expected covariance for different locations of the target relative to the agent, agent heights, and target belief covariances.

References

Athans, M. (1971). The role and use of the stochastic linear-quadratic-Gaussian problem in control system design. *IEEE Transactions on Automatic Control, 16*(6), 529–552.

Bachrach, A., He, R., & Roy, N. (2009). Autonomous flight in unstructured and unknown indoor environments. In *Proceedings of the European Micro Aerial Vehicle (EMAV) Conference*.

Barndorff-Nielsen, O. (1979). Information and exponential families in statistical theory. *Bull. Amer. Math. Soc. 1 (1979), 667-668., 273*(0979).

Bellingham, J., Richards, A., & How, J. (2002). Receding horizon control of autonomous aerial vehicles. In *Proceedings of the American Control Conference*.

Bertsekas, D. (2000). Dynamic Programming and Optimal Control, vol. 1 & 2, 2nd. *Ed., Athena Scientific*.

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

Brunskill, E., Kaelbling, L., Lozano-Perez, T., & Roy, N. (2008). Continuous-state POMDPs with hybrid dynamics. In *Symposium on Artificial Intelligence and Mathematics*.

Durbin, J., & Koopman, S. (2000). Time series analysis of non-Gaussian observations based on state space models from both classical and Bayesian perspectives. *Journal of the Royal Statistical Society: Series B (Methodological), 62*(1), 3–56.

Foka, A., & Trahanias, P. (2007). Real-time hierarchical pomdps for autonomous robot navigation. *Robotics and Autonomous Systems, 55*(7), 561–571.

He, R., Bachrach, A., Achtelik, M., Geramifard, A., Gurdan, D., Prentice, S., Stumpf, J., & Roy, N. (2009). On the design and use of a micro air vehicle to track and avoid adversaries. *International Journal of Robotics Research*.

He, R., Prentice, S., & Roy, N. (2008). Planning in Information Space for a Quadrotor Helicopter in GPS-denied Environments. In *Proceedings of the International Conference of Robotics and Automation (ICRA)*.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association, 58*(301), 13–30.

Hsiao, K., Lozano-Pérez, T., & Kaelbling, L. (2008). Robust belief-based execution of manipulation programs. In *the Eighth International Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

Kaelbling, L., Littman, M., & Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence, 101*, 99–134.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering, 82*(Series D), 35–45.

Krause, A., & Guestrin, C. (2007). Near-optimal observation selection using submodular functions. In *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI)*.

Kreucher, C., Hero III, A., Kastella, K., & Chang, D. (2004). Efficient methods of non-myopic sensor management for multitarget tracking. *Atlantis*.

Kurniawati, H., Du, Y., Hsu, D., & Lee, W. (2009). Motion Planning under Uncertainty for Robotic Tasks with Long Time Horizons. In *Proceedings of the 13th International Symposium of Robotics Research (ISRR)*.

Kurniawati, H., Hsu, D., & W.S., L. (2008). SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proceedings of the Robotics: Science and Systems (RSS)*.

Kuwata, Y., & How, J. (2004). Three dimensional receding horizon control for UAVs. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*.

Littman, M., Cassandra, A., & Kaelbling, L. (1995). Learning policies for partially observable environments: Scaling up. *Proceedings of the Twlfth International Conference on Machine Learning*.

Mahadevan, S., Marchalleck, N., Das, T., & Gosavi, A. (1997). Self-improving factory simulation using continuous-time average-reward reinforcement learning. In *Proceedings of the International Conference on Machine Learning*.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., & Scokaert, P. O. M. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, *36*, 789–814.

Paquet, S., Chaib-draa, B., & Ross, S. (2006). Hybrid POMDP Algorithms. In *Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*.

Paquet, S., Tobin, L., & Chaib-draa, B. (2005). An online POMDP algorithm for complex multiagent environments. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 970–977. ACM New York, NY, USA.

Pineau, J., Gordon, G., & Thrun, S. (2003). Policy-contingent abstraction for robust robot control. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI), Acapulco, Mexico*.

Porta, J., Vlassis, N., Spaan, M., & Poupart, P. (2006). Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research*, *7*, 2329–2367.

Richards, A., Kuwata, Y., & How, J. (2003). Experimental demonstrations of real-time MILP control. In *Proceeding of the AIAA Guidance, Navigation, and Control Conference*.

Ross, S., & Chaib-draa, B. (2007). AEMS: An Anytime Online Search Algorithm for Approximate Policy Refinement in Large POMDPs. In *Proceedings of The 20th Joint Conference in Artificial Intelligence (IJCAI 2007), Hyderabad, India*.

Ross, S., Pineau, J., Paquet, S., & Chaib-draa, B. (2008). Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 663–704.

Scott, A., Harris, Z., & Chong, E. (2009). A POMDP framework for coordinated guidance of autonomous UAVs for multitarget tracking. *EURASIP Journal on Advances in Signal Processing, 2009*.

Smallwood, R., & Sondik, E. (1973). The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, *21*(5), 1071–1088.

Smith, T., & Simmons, R. (2004). Heuristic search value iteration for POMDPs. In *Proceedings of the 20th conference on Uncertainty in Artificial Intelligence*.

Smith, T., & Simmons, R. (2005). Point-based POMDP algorithms: Improved analysis and implementation. In *Proceedings of the 21st conference on Uncertainty in Artificial Intelligence*.

Sutton, R., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence, 112*, 181–211.

Theocharous, G., & Kaelbling, L. (2003). Approximate planning in POMDPs with macro-actions. *Advances in Neural Processing Information Systems, 17*.

Triantafyllopoulos, K. (2003). On the central moments of the multidimensional Gaussian distribution. *The Mathematical Scientist, 28*, 125–128.

West, M., Harrison, P., & Migon, H. (1985). Dynamic generalized linear models and Bayesian forecasting. *Journal of the American Statistical Association*, 73–83.

Yu, C., Chuang, J., Gerkey, B., Gordon, G., & Ng, A. (2005). Open-loop plans in multi-robot POMDPs. Tech. rep., Technical Report, Stanford CS Department.