2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Exploiting Domain Knowledge in Planning for Uncertain Robot Systems Modeled as POMDPs

Salvatore Candido, James Davidson, and Seth Hutchinson

Abstract— **We propose a planning algorithm that allows user-supplied domain knowledge to be exploited in the synthesis of information feedback policies for systems modeled as partially observable Markov decision processes (POMDPs). POMDP models, which are increasingly popular in the robotics literature, permit a planner to consider future uncertainty in both the application of actions and sensing of observations.**

With our approach, domain experts can inject specialized knowledge into the planning process by providing a set of local policies that are used as primitives by the planner. If the local policies are chosen appropriately, the planner can evaluate further into the future, even for large problems, which can lead to better overall policies at decreased computational cost. We use a structured approach to encode the provided domain knowledge into the value function approximation.

We demonstrate our approach on a multi-robot fire fighting problem, in which a team of robots cooperates to extinguish a spreading fire, modeled as a stochastic process. The state space for this problem is significantly larger than is typical in the POMDP literature, and the geometry of the problem allows for the application of an intuitive set of local policies, thus demonstrating the effectiveness of our approach.

I. INTRODUCTION

When robotics research began to shift its focus from the study of mechanical devices to the study of automated robotic systems, it became apparent that uncertainty would be a fundamental barrier to the effective operation of autonomous robot systems. A number of tools have been introduced to cope with uncertainty in a robot's representation of its own state. These include geometric methods based on bounded uncertainties, e.g. [1], [2], and probabilistic methods such as the Kalman filter [3] and particle filters [4]. More recently, Bayesian methods have been combined with computational tools for dealing with uncertainty in some robotic applications, e.g. [5]–[11]. However, a commonly-used general framework for planning in robotics that considers uncertainty in both motion and sensing and optimization criteria has yet to emerge.

One of the most powerful models for stochastic processes is the partially observable Markov decision process (POMDP). It combines both the effects of uncertainty in process and sensing. While providing an amenable framework for modeling the uncertainty, the problem of finding optimal solutions for POMDPs is intractable [12]. Though progress has been made using randomized sampling algorithms to find anytime solutions for POMDPs, e.g. [13]–[17], the dimensionality and size of the state space of common robotic tasks places a huge computational burden on these algorithms.

Consider the example of 10 robots on a workspace of a 25×25 two-dimensional grid, trying to optimize a cost function; a problem with small size in many robotics contexts. If every robot can move to any of its neighboring cells, the set of controls has 2^{30} possibilities to evaluate. If we have no additional way to prune this set, a naive or brute-force approach will require checking 2^{30} controls to compute a 1-step optimal policy. In fact, if we sample this set uniformly without replacement we will need 2^{28} samples in order to achieve only a 0.25 probability of finding the optimal one-step action. While the problem is already expensive for one stage, if planning is to occur over multiple stages we must consider an exponential branching of possibilities due to the exponentially-growing set of possible sequences of observations. Additionally, analysis must be performed in an expanded space of probability functions over the state space, *beliefs*. Problems of even modest size that require a policy to consider a significant horizon length in order to generate interesting and efficient behavior are, using a POMDP model, computationally difficult in part due to these reasons.

We propose to utilize domain knowledge about specific problems, using the idea of a local *self-stopping policies*, to restrict the set of feedback policies considered, and temporal abstraction, using *hyper-particle filtering* [18], to approximate the result of using those policies. A *self-stopping policy*, or *option* [19], is the combination of a belief-feedback policy and a termination condition that determines when the policy should no longer be continued. Hyper-particle filtering, particle filtering in the belief space, is used to evaluate the cost and evolution of the POMDP using particular self-stopping policies at a particular belief. Our method is a hierarchical approach. We use a set of local (self-stopping) policies and what is referred to as a terminal policy. The terminal policy is often chosen to be the greedy policy (policy minimizing expected cost at the next stage), but that association is not required. Using sampling, we construct an approximation of the value function under our limited set of policies. This approximation of the value function is used to synthesize what we refer to as a *macro-policy*, a belief-feedback policy composed of set of self-stopping policies and switching logic to choose between them. This creates a hybrid system [20] with the modes of operation differing only by the policies closing the control loop. Our method is formulated as an anytime algorithm, whose solution from any belief can only improve with increasing number of iterations. The main idea

S. Candido and J. Davidson are in the Department of Electrical and Computer Engineering at the University of Illinois. candido,jcdavdsn@illinois.edu

S. Hutchinson is a professor of Electrical and Computer Engineering at the University of Illinois. seth@illinois.edu

is that the combination of hyper-particle filtering and local policies allows us to consider, instead of single controls, sequences of controls, and explore further into the reachable belief space.

Our method utilizes external information, in the form of local policies, to be able to generate plans that are forward-looking enough to solve some problems in the robotics domain. This approach has both the advantage and shortcoming of relying on a set of local policies. In many cases, local policies chosen by domain experts can provide a significant performance boost via reduction of the search space. The role of our algorithm is to decide when to apply different strategies known to be effective, in some circumstances and evaluate the (stochastic) long-term consequences of those policies. However, if the local policies are inappropriately chosen or no knowledge of solutions is available, our algorithm will typically not be able to deduce effective solutions. This is due to the exact reason we are able to attack large problems: effects of local policies are evaluated over long sequences and we do not attempt to determine optimal sequences of single controls or to explore the space in small increments.

To demonstrate our approach applied to a problem in the robotics domain, we consider the multi-robot task of a set of robots on a grid workspace. Their goal is to coordinate to manipulate a stochastic process inspired by the spread of a fire evolving on the robots' workspace. With this example, we hope to elucidate our method, show how it is applied, and demonstrate the feasibility of its use for planning for some problems in robotics.

In the following section, we briefly describe the canonical finite state, discrete time POMDP model. After pointing to research relevant to our work, we explicitly describe our algorithm and discuss sampling methods. Next, we introduce the fire fighting example and use it to demonstrate the efficacy of our approach. Finally, we discuss our method and summarize our current ideas on the future directions of our research.

II. POMDP MODEL

In this section, we briefly describe the POMDP model. For a more thorough discussion see [21] or [22]. A POMDP is defined by three sets and two mappings over those sets. A state space $\mathcal{X}$ is the set of underlying but unobserved states $x \in \mathcal{X}$. A set of control actions $\mathcal{U}$ contains the externally controlled inputs that can be applied to the system. The set of observations $\mathcal{Y}$ contains the possible signals that can be observed externally to the system. A process model $f : \mathcal{X} \times \mathcal{U} \times \mathcal{N} \to \mathcal{X}$ drives the evolution of the system in the state space. A sensor or observation model $h : \mathcal{X} \times \mathcal{U} \times \mathcal{M} \to \mathcal{Y}$ determines the sequence of observations that arrive externally to the system. The sets $\mathcal{N}$ and $\mathcal{M}$ are sets of possible valuations of noise that may enter the process and observation models. We consider a discrete-time, finite POMDP, i.e. $\mathcal{X}$, $\mathcal{U}$, and $\mathcal{Y}$ having finite cardinality and the process evolving over discrete stages.

The sequence of states is an MDP and is characterized by the transition function $x_{k+1} = f(x_k, u_k, n_k)$ where $u_k \in \mathcal{U}$, k is the stage number, and n_k is a random variable drawn from the distribution Γ_{n_k}, a noise distribution that models uncertainty in the process. Based on the characteristics of Γ_{n_k} and the function f, this induces a probability mass function over the random variable $X_{k+1} \in \mathcal{X}$. The quantity

$$p^u_{x^i|x^j} = \mathrm{P}_{\Gamma_{n_k}}\left[f(x_k, u_k, n_k) = x^i | x_k = x^j, u_k = u\right] \quad (1)$$

is called a transition probability. Observations of the process are driven by an observation function $y_k = h(x_k, u_k, m_k)$ where the random variable $m_k \sim \Gamma_{m_k}$ models the uncertainty in measurement of the state. We define observation probabilities with respect to h and Γ_{m_k}. The quantity

$$p^u_{y^i|x^j} = \mathrm{P}_{\Gamma_{m_k}}\left[h(x_k, u_k, m_k) = y^i | x_k = x^j, u_k = u\right] \quad (2)$$

is called an observation probability. We consider time-invariant processes, e.g. those where the noise distributions do not change as a function of stage.

The *information vector* $\mathcal{I}$ is the set of all certain knowledge regarding the state, available to the controller, i.e. $\mathcal{I} = \{\mu_0, u_1, y_1, \cdots, u_k, y_k\}$ where μ_0 is the initial probability distribution on $\mathcal{X}$. A *belief* is a probability mass function over $\mathcal{X}$ conditioned on the information vector. The set $\mathcal{B}(\mathcal{X})$ is referred to as the belief space and is the set of probability distributions over the state space. Every $b \in \mathcal{B}(\mathcal{X})$ for a finite state system is a column vector where the i^{th} component is the probability of being in the state x^i (the i^{th} state) given the current information vector.

The *transition probability function* $T^f(u)$ is an $|\mathcal{X}| \times |\mathcal{X}|$ matrix that, given a particular u, updates the belief according to the model associated with applying the process model. Each entry is the probability of transitioning from the i^{th} state to the j^{th} state, i.e. $t^f_{ij}(u) = p^u_{x^i|x^j}$.The belief vector b represents a prior so the expression $T^f(u)b$ is simply a Bayesian prediction. The *observation probability function* $T^h(y, u)$ performs the same task with respect to a new observation. The matrix $T^h(y, u)$ is an $|\mathcal{X}| \times |\mathcal{X}|$ diagonal matrix. Each entry on the diagonal is $t^h_{ii}(y, u) = p^u_{y|x^i}$. If we define the vector e to be a row vector of length $|\mathcal{X}|$ where every entry is one, then the probability of the observation y at stage k under the distribution b_k is $eT^h(y, u)T^f(u)b_k$. To ensure, that b_{k+1} is a probability distribution, we must normalize by this quantity, which is often denoted by η. This is direct application of Bayes' Rule. The transition function from stage to stage in the belief space, the belief process, is

$$b_{k+1} = \frac{T^h(y, u)T^f(u)}{eT^h(y, u)T^f(u)b_k} b_k = \eta_k T^h(y, u)T^f(u)b_k \quad (3)$$

where $y \in \mathcal{Y}$ and $u \in \mathcal{U}$. This function is referred to as the *belief transition function*.

The true complexity of a POMDP is not clear from just $|\mathcal{X}|$ alone. The sparsity of support on $\mathcal{X}$ of the beliefs in the reachable belief space, the uncertainty present in the observation and process models, the cardinality of $\mathcal{Y}$ and $\mathcal{U}$, the size of the reachable belief space, and the shortest

horizon length to compute an optimal solution are all factors in the amount of computation required to find optimal or nearly-optimal control policies that minimize a cost criterion.

Our goal is to evaluate the behavior of the system under various policies $\phi^1 : \mathcal{B}(\mathcal{X}) \times \mathbb{T} \rightarrow \mathcal{U}$ where $\mathbb{T}$ is a set of stages measuring the length of time ϕ has been used. We use a cost metric that evaluates the effectiveness of ϕ probabilistically, based on the cost of possible trajectories and the likelihood of them occurring while using ϕ. Although different formulations of cost metrics are available, we use an infinite-horizon total cost criterion with retirement option. Thus, the cost is

$$J^{\phi}(b_0) = \mathrm{E}_{\Gamma_{m_1} \times \cdots \times \Gamma_{m_T}} \left[\sum_{k=1}^{T} c_b(\phi(b_{k-1}))' b_k + c'_{bT} b_T | b_0 \right]$$

where c_b and c_{bT} are column vectors whose entries are running and terminal costs, respectively, paid as a result for residing in the states corresponding with the entries of b. The policy is given the option at every stage to "retire" and T denotes the dynamically chosen retirement stage. The quantity $c'_{bT} b_T$ is the cost paid to retire at stage T with the current belief of the system.

III. RELATED WORK

A number of researchers have investigated POMDP methods. In [24], it was shown that the optimal value function is piecewise-linear and convex and, for any finite horizon, one could construct the optimal value function exactly by considering only a finite number of points. An exact algorithm was given, but the process is too expensive to consider for POMDPs where the dimension of the belief space grows large. The intractability of computing exact solutions was expressed explicitly in [12]. Recently, progress has been made using methods that employ randomized sampling in the reachable belief space and perform standard Value Iteration on the approximated set of beliefs. Some of these methods are MC-POMDP [13], HSVI2 [14], PBVI [15], SARSOP [16], and PERSEUS [17]. Another approach for approximating the value function in the belief space is to use compression methods, e.g. [25]–[27].

Some approaches for MDP's such as [28] and [19], like roadmap methods, are hierarchical. Semi-Markov decision processes, [19] use macro-actions or options, feedback or open-loop policies that persist until a termination condition met, to temporally abstract the problem to reduce the number of controls considered. We perform a similar temporal abstraction, but consider the POMDP case and use hyperparticle techniques to analyze the result of the abstracted controls. Moreover, we use sampling of a parameterized set of macro-policies for multiple agents. Other hierarchical POMDP methods have been explored in the literature. Planning with a pre-defined hierarchy of tasks has been explored in [29] and [30]. Other methods, such as [31]–[33] attempt to discover a hierarchy of tasks to use for planning. Our use of the term hierarchical refers to the method we use for planning, not the structure of the POMDP model. Finally, other methods use sampling to select target locations and construct a graph between these targets. Such methods include the Belief Roadmap [34], the Stochastic Motion Roadmap [35], and Sampling Hyperbelief Optimization Technique [36].

The previously mentioned methods typically address POMDPs representing a single agent or decision maker. Multi-agent, decentralized POMDPs (DEC-POMDPs) have been considered recently by a number of investigators, e.g. [37] and [38]. Our work considers multiple agents but decision making is not completely decentralized.

Often when proposing a new POMDP algorithm, investigators test their algorithms against other algorithms in the literature using benchmark problems specifically designed for POMDPs, e.g. Rock Sample, Hallway, Hallway2, Tag, Tiger-Grid, Fourth Floor, Homecare. Descriptions of these problems can be found in [13]–[17]. These existing benchmark problems are not well suited for evaluating our approach. The core idea of our algorithm is to exploit domain knowledge about problems and use statistical methods to analyze the result of applying local policies generated from that knowledge. Testing our method on the aforementioned benchmark problems would require local policies with specific knowledge about the problem at hand. This would make any comparison against other algorithms that begin with no prior knowledge about the structure of the problem arbitrary and frivolous. Furthermore, the greatest strength of this method is that it allows us to consider problems that require solutions whose controls at the current stage are chosen with respect to consequences many stages in the future (i.e. long horizon). The benchmark problems do not emphasize this criterion because the cost functions are discounted which reduces the importance of costs of beliefs far into the future, and thus solving those problems often do not require a long planning horizon. Finally, we are specifically interested in using the POMDP model for planning policies for multiple robots manipulating stochastic systems. For these reasons, we demonstrate our method on the type of problem for which we envision it being employed in Section V.

IV. ALGORITHM

We iteratively approximate the value function of the POMDP using Algorithm 1. We store a set of reached beliefs B, that acts as an approximation of the the reachable belief space. The algorithm samples from B and the set of local policies, and evaluates the evolution of the POMDP using the sampled policy at the sampled belief. Since observations are not known at the planning stage, we represent that evolution as a probability distribution over $\mathcal{B}(\mathcal{X})$. New beliefs are then added to B, and we form a graph with the elements of B as vertices and groups of edges representing probabalistic transitions under the sampled policy. We then optimize over this graph to find the optimal sequence of local policies, from the set we have explored at the current iteration, from each belief in B.

[1]In the POMDP literature, policies are also denoted by π or γ.

Define a self-stopping policy to be the tuple $\psi = \{\phi, a\}$ where ϕ is a belief-feedback policy and $a : \mathcal{B}(\mathcal{X}) \times \mathbb{T} \rightarrow \{0, 1\}$. The function a is the stopping condition and returns zero when the policy should be used for another step. Define a macro-policy α to be a collection of self-stopping policies with switching logic. Essentially, $\alpha : \mathcal{B}(\mathcal{X}) \times \mathbb{T} \rightarrow \mathcal{U}$ using $\alpha(b, t) = \phi_k(b, t)$ where ϕ_k is the active policy in ψ_k. When $a_k(b) = 1$ then ψ_{k+1} is chosen using a finite state machine whose switching logic is based on the information available to the policy, b and t. Our goal is to find a macro-policy that is effective in the reachable portions of $\mathcal{B}(\mathcal{X})$ starting from b_0. Although local policies may be time dependent, we restrict those dependencies to be relative to the start of the policy and not the absolute stage when the policy is used in the POMDP. In this way, we are able to construct a stage-independent graph in the belief space while still utilizing useful relative stage-based conditions, e.g. the stopping condition for a local policy varies based on the number of stages the policy has been applied consecutively.

In order to evaluate the evolution of the POMDP under a local policy multiple stages into the future, we need to account for probabalistic observations. The evolution of the system into the future is a probability distribution over $\mathcal{B}(\mathcal{X})$, and is known as a *hyperbelief*. In the case of a finite number of controls, observations, and stages, the hyperbelief is comprised of a finite number of support points in $\mathcal{B}(\mathcal{X})$. Ideally, we would use the expected value function over the set of all sequences of observations which is equivalent to the expected value under the hyperbelief distribution. This is somtimes feasible for a typical one-stage value backup, but the number of sequences of observations, i.e. support points in the hyperbelief distribution, increase exponentially with the number of stages we are evaluating.

To approximate the true hyperbelief, we use hyper-particle filtering [18]. Hyper-particle filtering works by using a particle filtering algorithm and at every iteration, each belief is propagated through the probability transition function using the control specified by the policy. Then, each of these resulting beliefs b^i is paired with a number of sampled observations (based on $\mathrm{P}_{b^i}[o]$) and propagated through the probability observation function. These beliefs are appropriately weighted at each stage as to approximate the true pdf over the belief space at each stage. A resample operation is then applied to the set of beliefs. The tallied cost and resulting hyperbelief is then returned. This operation is referred to as HyperFilter in Algorithm 1. The set E is the set of local policy trials and the results of those operations. The quantities β and c refer to hyper-particles and costs, respectively.

The beliefs added to B are the beliefs at the end of each of hyper-particle filtered policy expansion, not every reached belief during the expansion. We establish an initial value (expected cost-to-go) for each of these new beliefs by hyper-particle filtering using a pre-defined terminal policy $\phi_{\text{terminal}}(b, t) : \mathcal{B} \times \mathbb{T} \rightarrow \mathcal{U}$ until the retirement condition for the system is met. A greedy policy can be used as the terminal policy. By continuing to sample simple non-greedy

Algorithm 1: Anytime Optimization

Input: b_0 - initial belief state,
Φ - set of self-stopping policies,
ϕ_{terminal} - terminal policy
Output: data structure containing α_{out}

$c_{\text{terminal}}(b_0)$ =HyperFilter($b_0, \phi_{\text{terminal}}$)
$B = \{b_0\}$
$E = \emptyset$
for $i = 1 : \text{anytime}$ **do**
  b_i = SampleBelief(B)
  ψ_i = SamplePolicy(Ψ, b_i)
  $(\beta_i, c(b_i, \psi_i))$ = HyperFilter(b_i, ψ_i)
  foreach $b_r \in \beta_i$ **do**
    if $\rho(b_r, b_s) < \epsilon$ *for any* $b_s \in B$ **then**
      $b_r = b_s$
    else
      $c_{\text{terminal}}(b_r) = \text{HyperFilter}(b_r, \phi_{\text{terminal}})$
      $B = B \cup \{b_r\}$
  $E = E \cup \{(b_i, \beta_i, \psi_i, c(b_i, \psi_i)\}$
  OptimizeGraph(B, E)

policies and using Value Iteration on the resulting graph, we typically are able to improve the value of the initial belief state using a composite policy generated from the graph and value function over the graph. Essentially, we start with a terminal policy and modify it incrementally to improve the quality of the policy.

The OptimizeGraph operation is a generic graph optimization. Its main purpose is to determine the optimal policy and cost to go from every $b \in B$, with respect to the transition edges in E and starting with a (terminal) cost of $c_{\text{terminal}}(b)$. One possible implementation is standard Value Iteration but, in some cases, more efficient graph search algorithms, e.g. Dijkstra's Algorithm [39], can be used as a substitute.

The SamplePolicy operation is problem dependent and will always depend on the local policies chosen, and the reasons for which they were chosen. The most naive implementation is a uniform sampling of a finite cardinality Ψ, the set of all possibilities available to the sampling algorithm. However, because the number of policies will typically be prohibitively large or infinite, this scheme will usually not be desirable or effective. In the same way that policies are chosen to encode domain knowledge about the problem, allocating resources to attempt application of policies appropriately is also problem dependent. The SampleBelief operation is extremely important and future work will need to focus on improvement of this aspect. We have tested two sampling methods that appear to show promise, but improving this component of our algorithm is an ongoing area of investigation.

The first method assigns a quantity ξ to each belief in B. The initial b_0 has $\xi(b_0) = 1$ and every other belief that is added subsequently is assigned $\xi(b) = 0$. Every time a policy expansion occurs from b^i and a resulting hyperbelief

β^k has some probability reaching b^k, we increase the $\xi(b^k)$ by the quantity $\mathrm{P}_{\beta^k}\left[b^k\right] \cdot \xi(b^i)$. A belief b^i is then sampled from B with probability proportional to $\xi(b^i)$. The main idea behind this method is to provide a heuristic that roughly approximates (and upper bounds) the likelihood of reaching a given belief. Beliefs with a smaller likelihood of being reached will contribute less to the overall expected cost of the starting state, and should be sampled less frequently.

The second method samples based on exploring the graph of beliefs and policies. Let b_s be the belief we are currently considering as a sample, initially set to be b_0. With probability κ, choose to sample b_s. Otherwise, with probability $(1-\kappa)$ choose to sample one of the policies expanded from b_s. (If no local policies have been explored at b_s, choose it with probability one.) Of the set of policies, choose one with probability proportional to the inverse of the expected value of using that policy, i.e. bias toward smaller valued policies. Let β_{s+1} denote the resulting hyperbelief associated with the applying that policy at b_s. Sample b_{s+1} with probability $\mathrm{P}_{\beta_{s+1}}[b_{s+1}]$. This sampling method has the property of exploring sequences of policies more frequently that have already been shown to produce a lower value relative to other sequences of policies.

In limited testing, neither method appears to be strictly superior to the other. However, detailed comparison has not been completed at this time.

The distance function ρ can be any metric, although L_1 is often appropriate. In many cases, a problem-specific distance function will be employed to take advantages of insensitivity in the cost function and transition functions to beliefs separated by small distance for that particular POMDP.

This algorithm produces a data structure $\{B, E, V\}$ that can be used as a macro-policy for the system. To utilize the macro-policy to control the system modeled by the POMDP, choose the closest $b \in B$ to the current belief of the system's state. Apply the local policy specified by the optimal edge in E starting from b. Once the local policy's stopping condition is met, repeat the same procedure. If the edge specifies the terminal policy as the optimal local policy, the terminal policy should be used until the retirement condition for the system is met.

V. RESULTS

In this section, we present a POMDP problem we have formulated and then demonstrate our algorithm's effectiveness. The target application for this method is planning policies for a group of robots attempting to control a random process. We tested our algorithm on a multi-robot task on a grid workspace. The robots' goal is to coordinate to manipulate a stochastic process inspired by the spread of a fire evolving on the robots' workspace. The robots will attempt to contain and extinguish the fire. We first describe the formulation of the problem at a high level and then present the results of application of our algorithm to this POMDP.

A. The Fire Problem

1) System Model: We model the workspace as an $M_1 \times M_2$ grid and place N robots on the grid. Thus, a state $x \in \mathcal{X}$ specifies, for every cell, whether it is on fire and, for every robot, the grid cell in which it resides. This implies the number of states in the $\mathcal{X}$ and the dimension of $\mathcal{B}(\mathcal{X})$ is $2^{(M_1 \cdot M_2)} \cdot N(M_1 \cdot M_2) - 1$. For example, in the case of 10 robots on a 25×25 grid there are approximately 8.7×10^{191} states. We model the robots acting deterministically, but each robot has a limited range of view so, even when sensor readings are combined, the state of the fire typically remains only partially observed. To discuss the problem in a concise manner, we introduce a number of operators to extract specific information from a state. The indicator function $c^{ij}(x)$ specifies if grid cell at (i, j) is on fire for state x. The operator $r^n(x_k)$ returns the location of robot n in the grid. We define $R^{ij}(x)$ to be the indicator function that returns one if only if $r^n(x) = (i, j)$ for some $1 \leq n \leq N$.

Each robot is allowed to move according to the equation $r^n(x_{k+1}) = r^n(x_k) + u_k^n$ where u_k^n contains the vectors that allow movement to the eight-point connected cells on the grid. The fire process spreads probabilistically. If a cell catches fire, it will remain on fire until it is extinguished by a robot. Robots extinguish fires on their current cell with probability one. If a cell is not on fire, it may catch fire with probability related to fire in the four-point connected adjacent grid cells. Specifically,

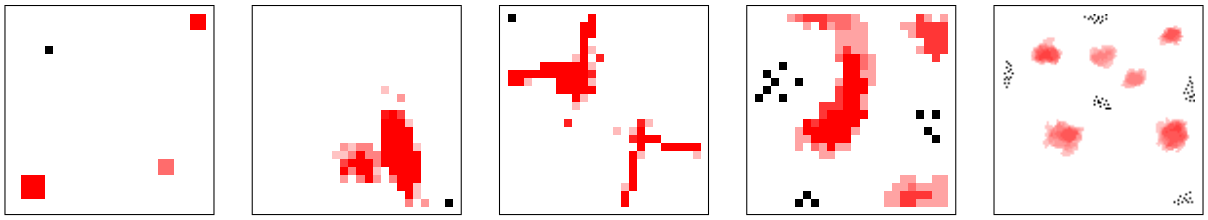
$$\begin{aligned}
&\mathrm{P}_{\Gamma_{n_k}}\left[c^{ij}(x_{k+1}) = 1 | R^{ij}(x_k) = 1\right] = 0 \\
&\mathrm{P}_{\Gamma_{n_k}}\left[c^{ij}(x_{k+1}) = 1 | R^{ij}(x_k) = 0, c^{ij}(x_k) = 1\right] = 1 \\
&\mathrm{P}_{\Gamma_{n_k}}\left[c^{ij}(x_{k+1}) = 1 | R^{ij}(x_k) = 0, c^{ij}(x_k) = 0\right] = \\
&\qquad\qquad \sum_{(a,b)\in\mathcal{G}} \left[s(a, b, i, j) c^{ab}(x_k)\right]
\end{aligned}$$

where $\mathcal{G}$ is the four-point connected neighborhood around cell (i, j). The function s maps two grid locations to $[0, \frac{1}{4}]$ and encodes the rate of spread between neighboring cells. Using the total law of probability we can derive the probability that a cell will be on fire at the next stage, given the state value of the current stage. Combining these probabilities for all grid cells and the movements of the robots allows us to form the probability transition probabilities $p^u_{x^i|x^j}$ and the probability transition function $T^f(u)$ for this problem.

Observations of the system come in the form of a value for each grid cell, $y = [\{y^{ij}\} \text{ for all } (i, j)]$. Each robot can see, deterministically, a limited distance o_c around itself. An observation is then the union over all (i, j) of the randomized operator

$$y^{ij}(x_k) = \begin{cases} c^{ij}(x_k) & \text{if } ||(i, j) - (i', j')||_\infty < o_c \\ & \text{for any } (i', j') \text{ s.t. } R^{i'j'}(x_k) = 1 \\ 1 & \text{otherwise w.p. } \frac{1}{2} \\ 0 & \text{otherwise w.p. } \frac{1}{2} \end{cases}$$

The observation space is also large, i.e. on the order of $2^{(o_c^2-1)N}$.



(a) b_0 for Experiment 1 (b) b_0 for Experiment 2 (c) b_0 for Experiment 3 (d) b_0 for Experiment 4 (e) b_0 for Experiment 5

Fig. 1. Initial Belief States for Experiments

2) Cost Function: Qualitatively, we want the robots to exhibit two behaviors that achieve two tasks. The robots should, if possible, completely extinguish the fire. Second, the robots should protect more valuable regions of the grid while completing the first task (or focus solely on this goal in the case where the first task cannot be completed). Quantitatively, we express this with respect to the state process as

$$J_{\mathcal{X}}(x_0) = \left[\sum_{k=1}^{T}\left(\sum_{(i,j)} a_1^{ij} c^{ij}(x_k)\right)\right] + c_T(x_T) \qquad (4)$$

where the constants a_1^{ij} weight the cost of the grid cell being on fire. The terminal penalty is $c_T(x) = 0$ for all $x \in \mathcal{X}$ such that $c^{ij}(x) = 0$ for all (i,j). It is chosen to be a large number in all other x, states where fire has not been extinguished. For the case of the simulations presented in this paper $c_T(x) = (M_1 \times M_2)/(1-\gamma)$ with $\gamma = 0.99999$ for all x where fire remains. Since the quantities $c^{ij}(x_k)$ are unknown, we optimize the expected value of (4) which is $J(b_0) = \mathrm{E}_{\Gamma_{\mathcal{SP}}}[J_{\mathcal{X}}(x_0)|b_0]$ where $\Gamma_{\mathcal{SP}} = \Gamma_{n_1} \times \Gamma_{m_1} \times \cdots \times \Gamma_{n_T} \times \Gamma_{m_T}$.

3) Belief Representation: We cannot work with the full belief vector as typically its dimension will be too large (the size of the state space), and the support of the belief vector grows exponentially (but occasionally can be reduced by observations and controls). Each state in the belief represents a set of cells that are on fire in the workspace. Thus, we use a particle filtering approximation [4] to contain the growth of the support of our belief vector, while still maintaining a representative set of samples. The only modifications to the standard particle filtering algorithm is to take multiple samples from the process model in the transition probability phase and sample without replacement. We can still compute the probability that a cell is on fire given a weighted set of possible states. The operator

$$c^{ij}(b) = \sum_{x \in X} c^{ij}(x) b(x)$$

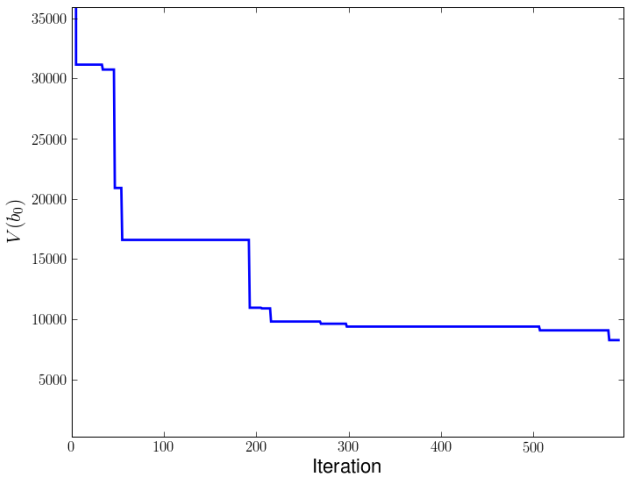
maps a belief to [0,1] and corresponds to the marginalized probability for cell (i,j) under b where $b(x)$ is the weight on state x in the belief vector. This also becomes an approximation with our particle filtered representation. We will use this set of marginalized probabilities to both construct our policies and evaluate (4).

4) Policies: The policies we have chosen are designed to be nearly decentralized, i.e. each robot acts with minimal need to check the controls of other robots, and are simple to compute. However, policy sampling operations such as choosing targets for robots are designed to create high level cooperation between robots when a new policy is applied, e.g. coordinating which robots will attack different portions of the fire.

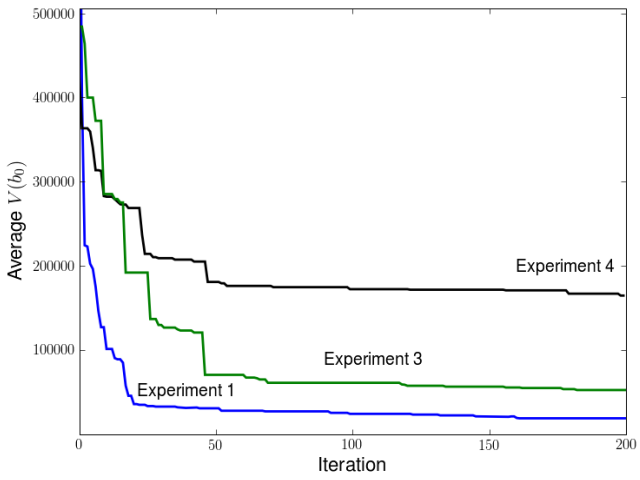
For brevity, we will not give explicit algorithmic specification of our policies, but instead describe them at a high level. We include parameterized policies that direct robots to target regions in the space. Those targets are sampled in a number of different ways, e.g. on the perimeter of the fire or areas of large scaled cost on the grid. Other policies direct robots to the nearest grid cell where there is a nonzero probability of fire, instruct robots to stay in place unless the are able to attack fire in the immediate vicinity, move the robots around the perimeter of the fire, and attempt to cluster the robots at various targets in the space. The terminal policy uses a brushfire expansion [40] on the workspace grid that is computed to grow the potential function away from cells with nonzero probability of fire. Robots follow this gradient to find and extinguish fire. Note that the terminal policy we used is not the typical greedy policy with respect to the cost function. This is because when no robot is directly adjacent to a cell containing fire, there is no gradient in the cost function with respect to the set of controls. This prevents the robots from making progress at a large number of belief states and severely limits the usefulness of the policy. However, our terminal policy is equivalent to the greedy policy at all places where the greedy policy has a gradient in cost. The retirement condition for the system occurs when we are applying the terminal policy and (i) the probability of fire in all cells is zero for the current belief or (ii) the sum of the expected value of fire in every cell exceeds $k_t \cdot M_1 \cdot M_2$. Typically k_t is chosen to be in $[\frac{3}{4}, \frac{9}{10}]$, depending on the size of the grid and the number of robots.

B. Experimental Results

To demonstrate the method, we chose four initial conditions where $M_1 = M_2 = 25$, $N = 14$ (Experiments 1-4) and one where $M_1 = M_2 = 100$, $N = 70$ (Experiment 5). The initial conditions are shown in Figure 1. In these figures, the intensity of the red values in the table represent the probability of fire with red corresponding to probability one. A lighter red corresponds to lower probability values, but



(a) Result of single long run



(b) Averaged result over all trials of Experiments 1, 3 and 4

Fig. 2. Plots of $V(b_0)$ vs. Iteration

for visualization purposes the probability is scaled between intensity of one half and one instead of zero and one. A black cell indicates the cell contains at least one robot. A white cell indicates zero probability of fire and no robot is present. The transition probabilities for fire from cell to cell were uniform throughout the workspace, i.e. $s(i, j, a, b)$ constant for all (i, j), (a, b).

We ran a number of trials on each experiment setup and the results of Experiments 1-4 are reported in Table I. Unfortunately, finding a lower bound on cost by computing the value for the fully-observed, MDP version of the problem is prohibitively expensive. Thus, we use the expected cost with a greedy policy as a baseline for comparison. Plots of the average expected cost versus number of iterations, averaged over all trials, for Experiments 1, 3, and 4 are shown in Figure 2 (b). In Figure 2 (a) we show the expected cost versus the number of iterations for a single trial with an extended number of iterations. The tests were conducted on workstation class computers.

The experiments demonstrate a significant reduction in expected cost from the greedy policy in each of the experiments. Moreover, the starting condition for each of the experiments vary greatly and the observed convergence of the value suggests the method performs as desired. The resulting policies often represent behavior that may be counter-intuitive. For instance, in Experiment 2 the robots start beneath the fire in the right-hand corner. The initial greedy policy attacks the fire by moving directly towards the fire, from the bottom. However, a plan is found that manages to extinguish the fire with a significantly lower cost. The robots are able to achieve this reduction in cost by first circling above the fire on the top and right, and then "pushing" the fire down and into the bottom. This type of behavior requires controls be determined by looking forward a significant number of stages into the future.

Even though the example in the first four experiments represent systems that require significant foresight in the planning algorithm, we also ran an experiment on an example with a 100×100 grid. In this problem, we increased not only the size of the workspace, but also significantly increased the number of robots, size of observation space, and initial amount of fire present in the workspace. For a single trial, over 30 iterations, we noticed a decrease from the greedy policy value of 2.60×10^7 to 1.70×10^7. The average time per iteration was 1774.5 (s), but note that this figure is averaged over a small number of iterations and should be only taken as a relative indication of the computational requirements of scaling the problem up.

TABLE I

CUMULATIVE RESULTS OF EXPERIMENTS 1-4

Experiment	1	2	3	4
Number Trials	10	6	10	10
Number Iterations	200	50	200	200
Avg. Greedy Cost	782,870.8	56,151.1	427,015.0	484,236.9
Avg. Final Cost	18,860.5	34,066.5	164,694.0	52,459.5
Std. Dev. Final Cost	8,370.4	10,918.7	8,500.1	13,071.4
Avg. Time/Iteration[2]	565.2 (s)	514.8 (s)	213.1 (s)	391.9 (s)

VI. DISCUSSION

We have proposed and demonstrated a new algorithm for finding belief-feedback policies for robotic tasks modeled as POMDPs. We use a hierarchical approach to utilize a set of local belief-feedback policies and combine them to synthesize a hybrid policy to reduce the expected cost from an initial belief. We established the utility of our approach by providing results generated by applying our algorithm to a POMDP problem formulated to test the applicability of this algorithm to multiple robots manipulating stochastic systems. Although we have demonstrated performance gains experimentally, our next step will be to perform an analysis of the theoretical performance of the proposed method.

We additionally hope to improve this algorithm on several fronts. In Section IV, we discussed two sampling methods for

[2]Only trials computed on identical hardware used.

B. However, we have not performed a detailed comparison of these or explored alternative sampling algorithms. Careful analysis of sampling methods could potentially provide a huge improvement in the performance of this algorithm, as they are a major factor in guiding the exploration and expansion of the reached belief space.

Our algorithm largely depends on the local policies and policy sampling methods developed for specific POMDP problems from domain knowledge. It will be important to understand the relationship between these policies, how to measure the utility of the local policies, and how to choose local policies for the purpose of maximizing the power and usefulness in the context of our algorithm. Hopefully, this will not only lead to an understanding of how a user should design local policies, but also insight into the expected number of iterations before the returns of the algorithm diminish. This could lead to a decision rule regarding how long should be spent planning and when action should be taken, in situations where planning occurs online but cannot be computed in real time. Since generating plans for large POMDPs often takes considerable time, this could be useful to analyze the tradeoff being made when waiting to act as planning continues.

REFERENCES

[1] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *International Journal of Robotics Research*, vol. 3, no. 1, pp. 3–24, 1984.

[2] M. Erdmann, "On motion planning with uncertainty," Master's thesis, Massachusetts Institute of Technology, 1986.

[3] R. E. Kalman, "A new approach to filtering and prediction problems," *Journal of Basic Engineering*, vol. 82D, pp. 35–45, 1960.

[4] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. New York, NY: Springer Verlag, 2001.

[5] W. Burgard, A. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour-guide robot," *Artificial Intelligence*, vol. 114, no. 1-2, 2000.

[6] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, 2000.

[7] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, pp. 391–427, 1999.

[8] M. Montemerlo, S. Thrun, and W. Whittaker, "Conditional particle filters for simultaneous mobile robot localization and people tracking," in *IEEE International Conference on Robotics and Automation*, 2002.

[9] S. Thrun, "A probabilistic online mapping algorithm for teams of mobile robots," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 335–363, 2001.

[10] S. Thrun, D. Fox, and W. Burgard, "A probabilistic approach to concurrent mapping and localization for mobile robots," *Machine Learning and Autonomous Robots (joint issue)*, vol. 31, no. 1-3, pp. 29–53, 1998.

[11] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2000.

[12] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99 – 134, 1998.

[13] S. Thrun, "Monte carlo POMDPs," *Advances in Neural Information Processing Systems*, pp. 1064–1070, 2000.

[14] T. Smith and R. Simmons, "Point-based POMDP algorithms: Improved analysis and implementation," in *Proceedings of Uncertainty in Artificial Intelligence*, 2005, pp. 542–555.

[15] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *Proceedings of the International Joint Conference on Artificial Intelligence*, vol. 18, 2003, pp. 1025–1032.

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

[17] M. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *Journal of Artificial Intelligence Research*, vol. 24, pp. 195–220, 2005.

[18] J. Davidson and S. Hutchinson, "Hyper-particle filtering for stochastic systems," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 2770–2777.

[19] R. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1, pp. 181–211, 1999.

[20] D. Liberzon, *Switching in Systems and Control*. Boston, MA: Birkhäuser, 2003.

[21] S. Thrun, *Probabilistic robotics*. New York, NY: ACM, 2002.

[22] W. Lovejoy, "A survey of algorithmic methods for partially observed Markov decision processes," *Annals of Operations Research*, vol. 28, no. 1, pp. 47–65, 1991.

[23] D. Hsu, W. Lee, and N. Rong, "What makes some POMDP problems easy to approximate?" in *Advances in Neural Information Processing Systems*, 2007, pp. 689–696.

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

[25] N. Roy and G. Gordon, "Exponential family PCA for belief compression in POMDPs," in *Advances in Neural Information Processing Systems*, 2002, pp. 1635–1642.

[26] P. Poupart and C. Boutilier, "Value directed compression of POMDPs," in *Advances in Neural Information Processing Systems*, 2003, pp. 1547–1554.

[27] X. Li, W. Cheung, and J. Liu, "Decomposing large-scale POMDP via belief state analysis," in *IEEE Conference on Intelligent Agent Technology*, 2005, pp. 428–434.

[28] N. Melchior and R. Simmons, "Particle RRT for path planning with uncertainty," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 1617–1624.

[29] E. Hansen and R. Zhou, "Synthesis of hierarchical finite-state controllers for POMDPs," in *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling*, 2003, pp. 113–122.

[30] J. Pineau, G. Gordon, and S. Thrun, "Policy-contingent abstraction for robust robot control," in *Proceedings of Uncertainty in Artificial Intelligence*, 2003, pp. 477–484.

[31] M. Toussaint, L. Charlin, and P. Poupart, "Hierarchical POMDP controller optimization by likelihood maximization," in *Proceedings of Uncertainty in Artificial Intelligence*, 2008.

[32] L. Charlin, P. Poupart, and R. Shioda, "Automated hierarchy discovery for planning in partially observable environments," in *Advances in Neural Information Processing Systems*, 2007, pp. 225–232.

[33] G. Theocharous, K. Murphy, and L. Kaelbling, "Representing hierarchical POMDPs as dbns for multi-scale robot localization," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 1045–1051.

[34] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *International Journal of Robotics Research*, 2009.

[35] R. Alterovitz, T. Simeon, and K. Goldberg, "The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty," in *Robotics: Science and Systems*, 2007.

[36] J. Davidson and S. Hutchinson, "A sampling hyperbelief optimization technique for stochastic systems," in *Workshop on the Algorithmic Foundations of Robotics*, 2008.

[37] J. Dibangoye, A. Mouaddib, and B. Chai-draa, "Point-based incremental pruning heuristic for solving finite-horizon DEC-POMDPs," in *Proceedings of Autonomous Agents and Multiagent Systems*, 2009, pp. 569–576.

[38] D. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of Markov decision processes," *Mathematics of Operations Research*, pp. 819–840, 2002.

[39] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, "Introduction to algorithms," Cambridge, MA, 2001.

[40] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, 2005.