To appear in the IEEE Transaction on Robotics, 2015

Extending the Applicability of POMDP Solutions to Robotic Tasks

Devin K. Grady, *Student Member, IEEE,* Mark Moll, *Senior Member, IEEE,* and Lydia E. Kavraki *Fellow, IEEE*

***Abstract*—Partially-Observable Markov Decision Processes (POMDPs) are used in many robotic task classes from soccer to household chores. Determining an approximately optimal action policy for POMDPs is PSPACE-complete, and the exponential growth of computation time prohibits solving large tasks.**

This paper describes two techniques to extend the range of robotic tasks that can be solved using a POMDP. Our first technique reduces the motion constraints of a robot, and then uses state-of-the-art robotic motion planning techniques to respect the true motion constraints at runtime. We then propose a novel task decomposition that can be applied to some indoor robotic tasks. This decomposition transforms a long time horizon task into a set of shorter tasks.

We empirically demonstrate the performance gain provided by these two techniques through simulated execution in a variety of environments. Comparing a direct formulation of a POMDP to solving our proposed reductions, we conclude that the techniques proposed in this paper can provide significant enhancement to current POMDP solution techniques, extending the POMDP instances that can be solved to include large, continuous-state robotic tasks.

***Index Terms*—Robotics, Autonomous Vehicles, Uncertainty, Plan Execution, POMDP**

I. Introduction

PARTIALLY-Observable Markov Decision Processes (POMDPs) [1] represent a planning problem where an agent performs actions and obtains sensor observations with the goal of maximizing total long-term reward. POMDPs can address noise in both the sensors and actuators of a robotic agent. Solving a POMDP is the process of computing an action policy that maximizes the total accumulated reward from an arbitrary reward function. The optimal action policy consists of the optimal action for any possible sequence of observations, such that the expected total reward is maximized under the POMDPs model of sensing and action uncertainty. POMDPs have been established as a tool to solve a variety of tasks in robot soccer [2], household robotics [3], coastal survey [4], and even nursing assistance [5].

Although POMDPs have been successfully used and are relatively well understood, they are not the *de-facto* solution for robotic planning. This is because the twin curses of history and dimensionality make solving a POMDP PSPACE-complete, even for ϵ-optimal solutions [6]. Robotic tasks must be very carefully designed and defined to reduce the size of the resulting POMDP as much as possible so that a solution can be found. Traditionally, the applicability of POMDPs to robotics has been limited by the fact that they can only be solved for small tasks. We increase the applicability of POMDPs by investigating techniques to extend the size of the task that can be considered.

Recent advances in POMDP solvers, particularly the introduction of belief point-based solvers (e.g., [7], [8]), have extended the size of solvable POMDP instances. Point-based solvers have an anytime property, so that a partial solution is returned even if the solution is not optimal. These solvers can successfully solve some robotic tasks involving action and sensing noise. Their primary limitation is the size of their reachable state spaces (from an initial belief). In the literature, the world tends to be described by 10×10 discrete grids to solve tasks only requiring 10–20 actions (e.g., [9], [10]). The POMDP's relatively small state space and short time horizon (number of actions required) restrict the classes of robotic tasks that can be solved. Although anytime solvers provide partial, near-optimal solutions for an increased number of tasks [7], [8], they cannot successfully solve some large robotic tasks within the time limits of our empirical evaluation.

Consequently, this paper proposes two reduction techniques to increase the size of robotic tasks that can be solved using POMDPs. Our reduction techniques are orthogonal to advances in POMDP solvers, although our evaluation relies on the anytime property. Both proposed reductions of the input POMDP typically improve the runtime of the POMDP solver, as the POMDP itself is simpler. After the reduced POMDP is solved, any simplifications made in the reduction must be addressed. The output of the solver, a policy, cannot be executed as-is and must be modified to lift the reduced policy to be a solution on the input problem. Although both techniques we present cannot guarantee theoretical optimality, our empirical evaluation shows that, with a fixed time budget, our reductions provide superior solutions compared to solving the more complex input POMDP.

This paper is a significant extension of the preliminary findings presented in [11]. We show the applicability of the algorithm introduced in [11], POMDP+Online, to task classes more complex than navigation. POMDP+Online can address more complex task classes because it reduces the complexity of the state space given to the POMDP solver. We will additionally present a task decomposition and subtask composition algorithm, MDP+POMDP, that is designed to

Manuscript received on **date**

Devin Grady, Mark Moll and Lydia Kavraki are with the Dept. of Computer Science, Rice University, Houston, TX 77005, USA. {devin.grady, mmoll, kavraki}@rice.edu

This work was partially supported by NSF 1139011, NSF IIS 1317849, and the U. S. Army Research Laboratory and the U. S. Army Research Office under grant number W911NF-09-1-0383. Equipment funded in part by the Data Analysis and Visualization Cyberinfrastructure funded by NSF under grant OCI-0959097. D. Grady was additionally supported by an NSF Graduate Research Fellowship.

Digital Object Identifier

attack the computational complexity caused by a long time horizon. MDP+POMDP will be applied to larger and longer tasks than could previously be addressed.

In Section II, we cover relevant research into related techniques. Then, in Section III, we describe the details of our two proposed techniques. We describe the task classes our techniques are applied to in Section IV. The specification of parameters and environments for the instances of the tasks we use are given in Section V. The performance data in these environments and comparison to a POMDP without our reduction techniques will be presented in Section VI. Finally, we summarize our findings and provide an evaluation of areas for future research in Section VII.

II. Related Work

Hierarchical POMDP methods [5], [12], [13] utilize many smaller POMDP policies at multiple levels. The time horizon is made relatively short for each subtask by splitting an input task into smaller subtasks combined with a top level POMDP. The MDP+POMDP method has significant computational benefits because it uses a fully-observable Markov Decision Process (MDP) on top of many smaller POMDPs. Although our framework shares some common ideology with hierarchical methods to improve computational speed, it is clearly separate.

Point-Based Policy Transformation (PBPT) [14] and online POMDP methods [15], [16] propose methods to decrease the amount of computation by considering low-probability events, and by fixing missing parts of the policy later. The key difference between the methods proposed in this paper to PBPT and online POMDP techniques, is that both POMDP+Online and MDP+POMDP break out of the POMDP framework entirely to avoid the exponential computation cost. Unlike the online POMDP which modifies the policy during execution, our online execution applies replanning to a reduced, static POMDP policy. Online POMDP methods do not capture the reduction in dimensionality that our motion model reduction takes advantage of. Finally, PBPT cannot apply to the reduction we propose because it requires a bijection between the original and reduced state spaces and a bijection between the original and reduced action spaces. The motion model reduction we propose uses state spaces that differ in dimensionality and our action spaces have very different constraints.

Recently, significant contributions have been made to problems for which all beliefs are approximately Gaussian [17], [18]. In general, however, the belief state of the world cannot effectively be collapsed into a unimodal distribution. However, we investigate examples that have a binary selection of obstacle/observer locations not captured well by a Gaussian. Therefore, although these Gaussian methods are effective, they are not applicable to the task classes we are presenting here.

Other POMDP solvers that might seem applicable include the popular POMCP [19], DESPOT [20], and MCVI [21] algorithms. The algorithm we chose to use, Monte-Carlo Value Iteration (MCVI) [21], was explicitly designed to operate in continuous state spaces, which is more broadly applicable to robotic systems. However, any discrete solver could be used on a continuous space with an appropriate sampling strategy, and likewise, a sampling-based continuous solver could be used for a discrete system, and thus the algorithms we will present in this paper are not restricted to using MCVI. More detail on MCVI is presented in Section VI.

A navigation task with explicit reward for belief variance minimization has been proposed as a basis for increasing the size of robotic tasks in the POMDP formulation [22]. User-designed policies that decrease uncertainty in the current belief or drive the system to a goal state are required as input policies. The policy computed switches between these input policies. Although high-level abstract actions such as these input policies could be used, we prefer not to require complex user defined input policies. These abstract actions are sometimes called macro-actions when composed of other, more basic actions. In our evaluation of the MDP+POMDP algorithm in Section V, we did implement very simple macro-actions as described in Section IV-C, and could also incorporate more complex expert-defined policies if they were available.

The MDP+POMDP algorithm is representationally similar to using a Dynamic Bayesian network (DBN) representation of hierarchical POMDPs [23]. The task solved in the cited paper is to *build* a model using inference of the world model using exogenous inputs. The algorithms in this paper are for a *given* model, and the problem at hand is to produce an action policy that the robot will execute. The fundamental difference with a DBN as compared to an MDP is that a DBN assumes inference is required to determine the current state, while the MDP is guaranteed to know the (in the terminology of the cited paper) abstract state.

Finally, we note several recent methods to improve POMDP solution times by decreasing the size of the state space [24]–[26] and action space [27] through variable resolution decompositions. Although these decompositions are relevant to our goal of decreased computation time without sacrificing total accumulated reward, the decomposition we propose in this paper is across both the state space and time as opposed to the state space alone or the action space. Each of these methods is orthogonal to our proposed algorithms and could potentially be combined to further improve on our results. In particular, FIRM [24] is philosophically aligned with MDP+POMDP in breaking the problem down into smaller belief sets and computing an MDP solution over this small discrete set of beliefs. FIRM is designed to generate a policy specifically for a task where the goal is necessarily a region of the state space around a particular point in state space. This goal point assumption is a critical part of FIRM: associated with every point in the FIRM roadmap is a belief stabilizing controller that can drive the system near that state with high probability. MDP+POMDP, rather than finding a policy to get to a specific state with high probability, uses the general MDP objective of optimizing a reward function that could yield, e.g., an infinite patrolling strategy. Additionally, FIRM selects controllers to drive between beliefs using a roadmap, while our algorithm solves a full POMDP between predefined beliefs. Not requiring a belief-stabilizing controller makes modifications to the robot model simple to test, particularly for cases where it may be difficult or impractical to write such a controller.

III. ALGORITHMS

Both of the two algorithms we propose will reduce an input problem specified as a POMDP before it is run through an off-the-shelf POMDP solver to reduce the computational complexity. The computed solution to this reduced problem is then used to solve the original input problem. In POMDP+Online, the input motion model is reduced to that of an unconstrained holonomic robot. The policy for a holonomic robot cannot be executed on the true, car-like robot model, and online replanning is used to address this reduction and approximately follow the solution policy. The second algorithm, MDP+POMDP, requires a decomposition of the input POMDP into a set of POMDP subtasks. After solving each subtask, an MDP is constructed using empirically simulated transition probabilities for each subtask. The MDP policy, combined with the POMDP subtask policies, comprise a global solution policy. Our proposed methods do not apply to every general POMDP. Instead, they apply to and exploit the structure of several classes of robotic POMDPs to improve solution speed. These classes of tasks may be quite broad, but it is not clear how to characterize exactly which classes of robotic tasks are amenable to our algorithms. Although we will consider a car-like robot model, the algorithms presented are not specific to this model. There are restrictions on the robot model due to our choice of approximation as a holonomic point robot; however, the precise restriction depends on the merits of the planning system.

A. POMDP+Online

The first algorithm we propose, POMDP+Online, operates in two phases and is detailed in Algorithm 1. The input, a POMDP model of the robotic task denoted P, has the motion model reduced to that of a holonomic robot in Line 1. For example, a car-like model for P can be replaced with a rigid body that can move in a grid along the cardinal and ordinal directions. Line 2 takes this reduced POMDP model and computes an *approximately* optimal policy. This approximation is due to the reduction introduced in the motion model as well as the limited offline planning budget. In Lines 3–5, the robot online replanning system is initialized. As discussed in the Experimental Evaluation section, we use OMPL [28] for planning, but other robotic planning systems could be substituted. The loop on Lines 6–11 continues replanning until the sensor returns a terminal observation. Line 7 plans from the current observed state to the target policy state, where the current state is inferred from observation in a robot-dependent fashion. In our experimental evaluation, we have chosen to simply plan from the state inferred by the last observation without implementing a robust state estimator to simplify the analysis of the output as well as the implementation of the experimental platform. In line 8, we update the *state* variable to be the expected state after following *path*. We will assume the policy is a deterministic decision tree, where each node is the current optimal action to execute, and an edge is selected based on an observation. Thus the observation generated on Line 9 informs the policy node update on Line 10. An observation is sampled from the true underlying world state as a simplification of a sensor inference. Although in a physical implementation, a state estimator would need to be implemented, we preferred to use an abstract model with specifically controllable, predictable, and reproducible noise. It is very important to note that the true state is in fact only partially-observable, for otherwise the task at hand is not a POMDP. The online planning does not explicitly use the reward model to plan motion (as this depends on modeling stochastic observations/actions and would amount to solving the POMDP), but, instead, follows the highly-rewarding policy that was computed on Line 2. During the online operation, the task is implicitly encoded in the policy. Therefore, the online planning system must follow each step of the policy as closely as possible (Line 7). This requirement is due to the fact that the online planning system in this algorithm addresses each discrete time segment (action) from the POMDP model as a separate and independent planning task, without concern for reward. Only the POMDP policy encodes a plan to maximize reward, so if the online planner does not approximate the actions specified in the POMDP policy, the robot cannot hope to achieve a high reward except in trivial cases.

Algorithm 1 POMDP+Online

Require: ***P* is a POMDP encoding a robotic task**

1: $P' \leftarrow$ MakeHolonomic(P)
2: *policy* $\leftarrow$ Solve(P') # Offline Planning
3: *state* $\leftarrow$ SampleInitState(P') # Initialization
4: *policyNode* $\leftarrow$ *policy*.root(P')
5: *obs* $\leftarrow$ SampleObs$(state, policyNode, P')$
6: **repeat** # Online replanning loop
7: *path* $\leftarrow$ Plan$(obs.\text{state}(), policy, policyNode, P')$
8: *state* $\leftarrow$ *path*.endPoint()
9: *obs* $\leftarrow$ SampleObs$(state, policyNode, P')$
10: *policyNode* $\leftarrow$ *policyNode*.child(obs)
11: **until** TerminalObservation(obs)

The comparisons in Section V will show the performance benefits of reducing the input POMDP and fixing the output policy with replanning. POMDP+Online, by using a simpler robot model in the offline computation, reduces the time required by the POMDP solver.

In particular, a simpler robot model reduces the size of the state space and action space, both of which contribute exponential terms in the worst-case solve time. Additionally, the simpler dynamics model is chosen to reduce the complexity of the system motion, so a shorter execution time is required to get between state space points, which allows us to search for solutions with a shorter time horizon. The time horizon is present as another exponential term in worst-case solve time, so the overall effect from this simplification can be very large. However, this simplification comes at the cost of additional computation online because the robot motion model being executed is not the same as the reduced motion model used in the resulting policy. The reduced models that will work for a given online execution model are hard to characterize in advance, however it is a topic worth investigating in the future. Although the dynamics simplification provides significant computational benefits, this algorithm can only be applied to systems that can be approximated well with

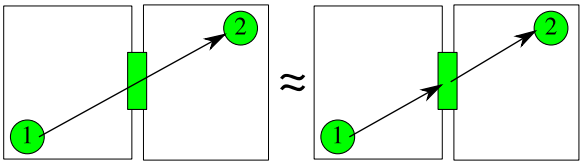


Fig. 1: A task that requires passing through well-observed boundary regions can be naturally decomposed into smaller subtasks. In this example, the task is to navigate from region 1 to region 2, passing through an intermediate region. The solution to the global task can be approximated through a subtask from region 1 to the intermediate region, followed by a subtask from the intermediate to region 2. The intermediate region is assumed to be a small region where additional sensing information is available.

simpler ones. In particular, we have not constructed adversarial environments where there exists no solution for the simplified system but a solution for the full system exists (or vice versa). In general, if the obstacles are assumed to be conservatively defined in the simple system such that the complex system is guaranteed to be able to track each simple action without hitting an obstacle, then this algorithm will be applicable. Small-time local controllability is sufficient to satisfy this assumption, however, it is not necessary, and small violations of this assumption may be acceptable for some specific problem instances. This assumption is relatively easy to check on the bounded-curvature models we will be using, though it is admittedly intractable in the general case.

B. MDP+POMDP

Some tasks with a long time-horizon may be intractable even using the reduced robot model proposed in POMDP+Online. If we assume that the sensor model samples observations from a bounded distribution in some regions of the state space (as opposed to the common Gaussian or other unbounded distributions), we can build a decomposition of the task into subtasks based on these well-observed boundary regions. We propose that an MDP can connect these regions into a solution to the global task. Our proposed decomposition into subtasks is orthogonal to the reduction of the motion model used by POMDP+Online which reduces the complexity of the state space. Decomposition into subtasks has the benefits of reducing the time horizon required for each policy and achieving performance improvements, but the tradeoff is that we will solve many smaller POMDPs and necessarily sacrifice global optimality.

Combining small POMDP subtasks into a global solution will allow the consideration of sensor noise during robot execution (as each subtask policy is still a POMDP policy), while directly attacking the curse of history (as the history is discarded in between the subtasks). The combination of the resulting POMDP policies is accomplished through an MDP in an algorithm we call MDP+POMDP (Algorithm 2). The decomposition into subtasks using a natural set of boundary regions is illustrated by Figure 1.

If the task is comprised of several rooms joined by bounded localization regions, as in Figure 1, then we can decompose the task into natural subtasks. Doors could provide bounded observability easily, as a doorframe is an easy to recognize separator. A separation by bounded observability is critical because an MDP is assumed to have perfect knowledge of which state it is in. At a high level, the task is approximated as an MDP between the bounded observability regions. This MDP is defined as $(S_{MDP}, A_{MDP}, T_{MDP}, R_{MDP}, \gamma_{MDP})$. S_{MDP} is the discrete set of bounded observation regions that our sensor model described above can perfectly observe. A_{MDP} consists of one action per state adjacency pair. That is, if two bounded observation regions are connected at the POMDP subtask level, the MDP will contain an action to traverse between the bounded observation regions. There is an additional implicit assumption that the global POMDP can be well approximated by a sequence of POMDPs that can each be solved independently.

The instantiation of each MDP action is a policy computed by one of the POMDP subtasks. The start region in this POMDP subtask is uniformly sampled, discarding any history of previous actions or observations that could inform this distribution. The transition probabilities, T_{MDP}, are empirically derived from simulations of the policy.

Algorithm 2 MDP+POMDP

Require: ***P* is a POMDP encoding a robotic task**

1: # Offline policy computation
2: Decompose P into subtasks $P_{i,j}$
3: **for all** $P_{i,j} \in$ **P do**
4: $\quad \pi_{P_{i,j}} = \textit{POMDP.Solve}\,(P_{i,j})$
5: **end for**
6: **Construct** M**, an MDP with actions given by** $\pi_{P_{i,j}}$
7: $\pi_{MDP} = \textit{MDP.Solve}\,(M)$
Policies $\pi_{MDP}, \pi_{P_{i,j}}$ compose a global policy
Online execution: current region is denoted by i, initial state is sampled from initial region
8: **repeat** # Subtask update & execute loop
9: $\quad$ # MDP policy selects next destination region
10: $\quad j \leftarrow \pi_{MDP}(i)$
11: $\quad$ # Execute pre-solved POMDP policy to go from i to j
12: $\quad$ execute$(\pi_{P_{i,j}})$
13: $\quad i \leftarrow$ estimate(obs) # Determine MDP region
14: **until** TerminalMDPRegion(i)

Line 1 of Algorithm 2 is inherently vague because the decomposition process is problem dependent. The exact form of the decomposition varies by problem, but we assume there exists some natural decomposition where the boundaries between elements in the decomposition can be perfectly observed. Lines 2–3 solve all subtasks and store the resulting policies. Simulations of these policies are used in Lines 5–7 to construct and solve the high-level MDP using policy iteration [29]. At this stage, we have built a two-level policy composed of π_{MDP} and all the $\pi_{P_{i,j}}$'s.

Lines 8–13 correspond to execution of the two-level policy. The current region where the robot is located is denoted by i. At the start of execution the initial state is sampled from region i (in simulation) or estimated from sensor data. The MDP policy

determines which region the robot should navigate to (line 10). The precomputed POMDP policy $\pi_{P_{i,j}}$ is executed to navigate the robot to neighboring region j (line 12). Due to uncertainty, the robot might actually end up in region $j'(j' \neq j)$, but since the policy already contains the optimal actions to reach the goal, this is merely a delay rather than a failure. Execution of the POMDP policy continues until a region other than i is entered. At that point the current region is updated based on sensor observations (line 13). This process repeats until the robot reaches the terminal MDP region (line 14).

We have presented two algorithms to reduce the computational complexity of robotic tasks specified as a POMDP. Each algorithm "reduces" the input problem before it is run through an off-the-shelf POMDP solver. After the reduced problem is solved, the algorithms also specify the appropriate method to lift the reduced policy to solve the original input task. Each algorithm works for only a subset of general POMDP tasks, which can still be quite general. We will proceed by specifying three task classes and applying the appropriate algorithm to each.

IV. Task Classes

This paper presents three classes of robotic tasks. Each task can be represented directly as a POMDP, however, it may be unsolvable within a reasonable amount of time. Given the exponential growth of the required computation, it is reasonable to think that even as computation power grows, techniques to reduce the input task will always be useful. Although our time limit is arbitrary, memory capacity will present a hard barrier to simply allowing for longer run times. Our tasks vary greatly in memory use, but generally range in the gigabytes per hour of computation. In this section, we will describe the classes of input POMDPs considered in our experimental evaluations before reduction.

A. Exposure Minimization

The class of exposure minimization tasks are defined as navigating a robot through a known environment without being seen. This task is inspired by a stealth transport mission, where a robotic truck would prefer to arrive at the destination unseen, but being seen does not terminate the mission. Several locations exist that may contain an observer. Each observer induces a visibility polygon, which the robot will consider as a region with additional cost to navigate. The number of observers are known but not which of the possible locations they are in. Only noisy sensing is available for observer location.

This task is considered successful when the robot reaches the goal location. When the robot is successful, a large reward is achieved. If the robot hits an observer, crashes into an obstacle, or exceeds the planning time horizon, the task is aborted and a large penalty is incurred. The POMDP framework builds a policy to maximize expected reward. Because the only positive reward available in this task is at the goal, the policy is expected to balance success rate, time to goal, and time spent in the visibility polygons of the observers.

B. Search and Rescue

The search and rescue (SAR) task class requires the robot to navigate to a goal, however, the goal location is not known and must be sensed. Imagine an ejected pilot with an active radio but the signal is too weak to directly communicate. In this case, the robot must use noisy sensing of the radio signal to estimate range to the pilot and determine the true location from a set of probable locations. There are also terminal obstacles with unknown positions that must be sensed with another, independent sensor. This task adds a uniform cost of motion throughout the workspace.

The SAR task is harder to solve than exposure minimization because the goal is no longer a single location but, instead, is selected from several possibilities. The location selection is done from a uniform distribution, as are the obstacles.

The reward structure for SAR task is 'binary', that is, either a large reward is achieved at the (unknown) goal location, or a large penalty is incurred at either known or unknown obstacles. Unlike in the exposure minimization task, there is no intermediate level of reward.

C. Indoor Navigation

We will investigate both a single task and the ability of MDP+POMDP to handle many tasks defined on the same map. An indoor navigation task class requires the robot to navigate to a goal region under action and sensing noise. One example of indoor navigation, wheelchair navigation, may require solving many tasks in the same environment. The smart wheelchair may be overridden by the user at any time. Therefore, deviations from the expected motion model are expected but very difficult to characterize in advance. Given an un-modeled deviation from the expected outcomes, the robot could observe a sensor value that was never seen during computation of the POMDP policy. An unexpected observation is not accounted for in a traditional policy. However, our proposed top-level MDP policy addresses this as MDP policies return an action based only on current state, and can therefore recover from a user taking an un-modeled transition.

The known map of the environment provides the locations of non-terminal obstacles. The cost of motion is zero throughout the state space; the only reward is a terminal reward of one at the goal. Furthermore, as physical implementation cost could be an issue, we operate with minimal sensing: the robot only uses 4 obstacle sensors and a doorframe sensor (e.g., perhaps an upwards looking camera).

The indoor navigation class of tasks is designed explicitly to test the ability of our proposed decomposition to extend the time horizon that can be addressed with a current POMDP solver. The decomposition is orthogonal to the robot model reductions made in POMDP+Online, and our discussion will focus on the effect of the decomposition and MDP+POMDP algorithm in this task class. We have also applied the POMDP+Online algorithm to the policy computed with MDP+POMDP to verify that these algorithms can work together.

V. Experiments

In this section, we will describe the specific environments that are used to instantiate the task classes we have specified,

while experimental results are deferred to Section VI. The first two tasks, exposure minimization and SAR, are not designed with regions of bounded-observability. Therefore, they are used to test only the POMDP+Online algorithm in isolation. The indoor navigation task is more structured and has distinct regions separating subtasks that can be used for bounded-observability. This task is used primarily to test MDP+POMDP in isolation, though is also run with POMDP+Online to show that these two algorithms can be easily combined.

A. Exposure Minimization

We evaluated the exposure minimization task class in two distinct environments called L and Branched (see Figure 2). The input POMDP is a discretized-action version of the true robot model, where the true model is a Reeds-Shepp car [30]. The reduced POMDP will remove the kinematic constraints imposed by this robot model and, instead, model a holonomic point robot. This reduced robot model is simpler in state space ($\mathbb{R}^2 \times \mathbb{S}$ vs. $\mathbb{R}^2$) and system dynamics (bounded-curvature paths vs. holonomic), thereby decreasing two exponential factors in the computational cost.

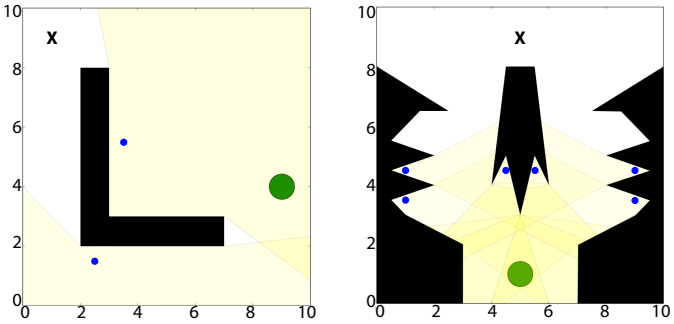
For each action we compute the percentage of time that the robot is observable in order to differentiate between actions that are observed for varying amounts of time. The cost of being observed increases linearly with the number of observers that can see the robot. The overall behavior should find a near-optimal tradeoff between time spent navigating to the goal (discount factor making it worth less over time), and the cost of being observed. Because the locations of the observers are not known at the outset, and the sensor has Gaussian noise, it may take time to determine the appropriate action. If there exist multiple observers/obstacles or goals, only the closest one is sensed. This can lead to significant perceptual aliasing between possible observer positions and adds to the difficulty of this task class. Perceptual aliasing is always a significant factor in the environments used.

Some important characteristics of the POMDPs in this task:

- Sensor noise $\sigma = 0.2$·range
- Sensor discretized at a resolution of .1
- Reward of achieving goal location $= 1000$
- Reward of hitting (terminal) obstacle $= -1000$
- Reward of motion $= 0$
- Reward of being observed (per action) $= -10$
- Known obstacles

Prior work has shown that the size of the action space directly affects the solution time [27]. For more complex examples with limited time, this translates into decreased policy reward. The robot vehicle model we have chosen models the state space as rigid body car position and orientations, $SE(2)$, and imposes bounded-curvature path constraints. As required in MCVI, a discrete action set is used in the POMDP, therefore the action space A is discretized into 5 constant-speed actions: turning in either direction 180° or 90°, and proceeding straight ahead.

Sensor observations are $(x, y, \textit{range})$ tuples, discretized to integer values as necessary (or a subset of these variables as relevant to the task at hand). T is noise-free, so we can focus all computation and discussion on the sensing. The sensor model, Ω, implements a range-dependent Gaussian noise model for the range measurement. The reward function R (characterized in the list above) is independent of A and only depends on robot position. The action and sensing models from the exposure minimization tasks are also used in the search and rescue tasks.



(a) L (b) Branched

Fig. 2: Environments used in the exposure minimization tasks. In L, one observer location is selected out of two (blue dots), while in Branched, two are selected out of six. X denotes starting location, green circle denotes goal region.

B. Search and Rescue

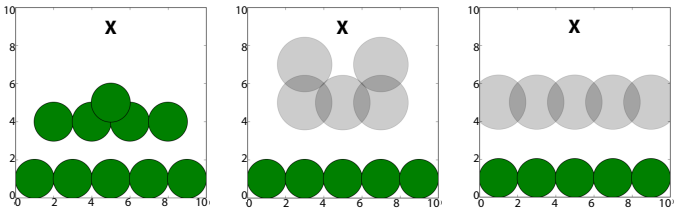
In the SAR tasks, some circular regions are unknown obstacles. The sensor model for these obstacles is the same as described above for the observers. In addition to the unknown obstacles, the goal location is now also unknown. Each task instance has several goal possibilities that are randomly sampled to select one goal. Range sensing of the goal is identical to, and independent from, the sensing of the obstacles. Neither of these sensor models can provide bounded localization, and so cannot be easily decomposed into subtasks. The action model is the same as before. Three environments tested in this task class are shown in Figure 3, called Empty, Symmetric, and Line.

Similar to the previous tasks, we list some of the crucial characteristics of the POMDPs used in this task class:

- Sensor noise $\sigma = 0.2$·range
- Sensor discretized at a resolution of .1
- Reward of achieving goal location $= 1000$
- Reward of hitting (terminal) obstacle $= -1000$
- Reward of motion $= -1$
- Independent, identical goal range sensor added
- Unknown, circular obstacles

The Empty environment, depicted in Figure 3(a), is a useful starting point to see the best case scenario. The robot must utilize sensing to disambiguate the 10 possible goal locations. Analyzing the failures in this example shows that they are due to observation histories seen during execution that were not part of the policy.

The Symmetric example is difficult because only the closest obstacle can be sensed. Only after significant motion can the obstacle and goal locations be determined with any significant confidence. A long path exists around the obstacles, but the particular reward structure makes this path sub-optimal. The



(a) Empty (b) Symmetric (c) Line

Fig. 3: Environments used for the SAR tasks. In the Empty environment, one goal (solid green circles) is selected from the 10 shown locations. In the Symmetric and Line environments, one goal is selected from the five possible locations, and three out of the five obstacles (transparent grey circles) are present. In all environments, the robot starts at the small X at the top.

optimal policy uses sensing and takes a more direct route while incurring a small expected percentage of collision.

Finally, the Line environment is similar in construction to the Symmetric environment except there is no guaranteed safe path. At the outset, it is unclear if the Line environment will be harder or easier than the Symmetric example. Although there is no safe path, the robot has more free space to maneuver and gather information.

C. Indoor Navigation

The navigation task class will be used to evaluate the proposed decomposition in MDP+POMDP, because it is natural to define a decomposition using features of the workspace of an indoor environment. Specifically, we will use the door frames as our bounded-observability regions that can provide improved localization estimates. The start and goal regions are also assumed to be bounded-observability regions, that is, we assume the robot knows exactly when it has successfully completed a navigation task. The high-level MDP will navigate from start to goal by passing through a sequence of door frames. Each transition at the high-level is instantiated on the robot by executing a low-level policy.

The robot state will be a continuous (X, Y) position, so $|S| = \infty$. The actions in A used in both the global POMDP and all POMDP subtasks are motion in the four cardinal directions for length L. Each of these motions also includes Gaussian noise in the final location. Because we wish to solve large problem instances, macro actions (additional actions composed of several 'normal' actions) were implemented as repetitions of motion in these 4 directions of motion until an obstacle or a region of bounded observability is detected in the direction of motion. See [31] for more information on macro actions. These 8 actions, plus an option to stay still, yield an overall $|A| = 9$.

Observations are very different from the previous task classes. The first sensor detects if there exists an obstacle within a distance equal to one motion step plus noise in a given direction. With 4 directions and 2 possible results per direction (on/off), there are 16 possible observations from this sensor. The second sensor returns a region number if the robot is within a region of bounded localization, and a null value in the rest of the space. In addition, a special value is set if the robot is in an invalid state, or if the robot has reached the goal. This second sensor provides information as is needed to split the overall POMDP into smaller POMDPs joined by a fully-observable MDP. This fully-observable MDP does not suffer from the curse of history; it is the partial-observability of the POMDP that requires considering historical observations. The MDP can be optimally solved in PTIME, so the MDP is computationally trivial compared to the PSPACE-complete subtasks.

The observation model Ω defines noisy obstacle sensing as well as a perfect region sensor. Given the current state of the robot, s, $o = \Omega(s, a)$ returns a value described above, where the range of the obstacle sensor is sampled from $\mathcal{N}(L, \sigma)$. As mentioned above, L is the length of a single action from the action model. We observed in our initial experimental evaluations that when the standard deviation is too high, then the optimal policy will be to ignore all noisy observations. Therefore, the standard deviation used in this sensor is relatively low but still significant, at $\sigma = \frac{L}{10}$. The action model (T) for this task class also includes Gaussian noise along the direction that the robot is moving.

The MDP model has the following characteristics:

- S_{MDP} is the set of bounded observability regions plus a failure state
- The goal and failure states are terminal
- Actions are POMDP subtasks between adjacent states
- POMDP policy failure leads to the failure state
- Reward at the goal state is 1, 0 elsewhere

And for each POMDP subtask for noisy navigation:

- Sense x, y position and bounded observability region
- Position sensor noise $\sigma = 0.1$
- Obstacles are not terminal (action execution just stops)
- Reward of achieving goal location $= 1000$, 0 elsewhere
- Known obstacles

This task class is designed to push the limits of the POMDP solver not in sensing, but in time horizon. As we will see in the Experimental Results section, these tasks can provide a significant challenge due to the long time horizon.

The first environment will be a set of rooms with one start region (uniformly sampled) and one goal region, depicted in Figure 4 and referred to as "Rooms." There are 11 regions of bounded observation: 9 door frames between rooms, the start, and the goal. Assuming a fixed start to goal task, 21 local policies between these regions are needed for this environment (given the particular adjacencies in this task). Rooms that are exact duplicates (three across the bottom) can re-use policies between them. This is why only 21 policies are needed even though there are 33 pairs of adjacent regions.

The second environment, denoted "Office" and depicted in Figure 5, is recreated from existing planning literature [32] and was selected to highlight a potential benefit of the MDP+POMDP algorithm in a multi-query example. There are seven non-doorframe, bounded observability regions (e.g., visual fiducials [33]). Start and goal locations are selected from these seven regions, yielding a total of $7 \cdot 6 = 42$ possible start and goal combinations. In Figure 5, only one start and goal combination is shown. This is the particular task instance we

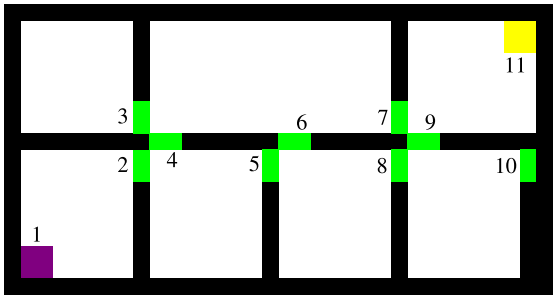


Fig. 4: The Rooms environment for indoor navigation from a single start region to a single goal region. The start region is marked 1 and colored purple. The goal region is marked 11 and colored yellow. All other regions that support bounded observation are marked in green. Each action moves the agent an expected distance equal to the smallest region width. This task can be naturally decomposed into 7 rooms with 11 labeled regions. 3 rooms are exact duplicates of each other. The 5 unique rooms require computing 21 distinct policies from region to region.

will focus on. In this environment, there are 15 regions with bounded observability and 59 adjacent pairs of these regions. Unlike the Rooms example, there are no duplicated rooms that allow policy re-use. This is the worst-case for the number of subtasks that need to be solved. Computing the 59 local policies will include policies to and from every possible start and goal. Therefore, the MDP+POMDP solution can solve all start and goal combinations without solving any additional POMDPs. The high-level MDP has a well defined start and goal, however, and must be re-defined for each of the 42 possible task definitions in this environment. However, as discussed in Section VI-C, the time to construct and solve all 42 MDP instances is negligible.

The expected upper bound on the number of actions that might be required for a good solution, called the planning horizon, is an input to the POMDP solver, MCVI. The maximum planning horizon for each of the small rooms in both problems is estimated as $L = \frac{l+w}{m} \cdot 2 \approx 40$ time steps, where l and w are the length and width of the environment respectively and m is the nominal motion length. This approximation is from the knowledge that in a navigation task, we are unlikely to need to travel longer than twice the width plus the length of the room. The larger room is thus given $L = 80$ time steps, and the whole environment uses $L = 200$ time steps. As the number of reachable belief points are upper bounded by the number of possible histories, for each of the three task sizes, the number of reachable belief points is at most

$$|B| = (|A| \cdot |O|)^L \approx \begin{cases} 10^{110} & \text{for } L = 40 \\ 10^{220} & \text{for } L = 80 \\ 10^{552} & \text{for } L = 200 \end{cases}$$

This can be intuited as the maximum number of possible histories that can be observed. Clearly, $9 \cdot 10^{110} + 12 \cdot 10^{220} \ll 10^{552}$ (using the "Rooms" environment as an example), indicating that we expect to see significant performance benefit from this two-stage approach.

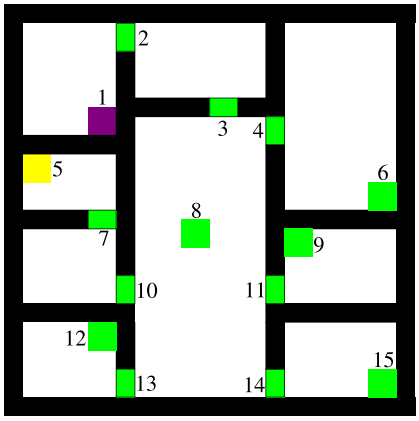


Fig. 5: The Office environment for navigation between multiple start and goal regions. One such combination of start (region 1, purple) and goal (region 5, yellow) is shown. All other regions that support bounded observation are marked in green as in the prior example. This task can be naturally decomposed into 9 rooms with 15 regions, where all rooms unique. The 9 rooms require computing 59 distinct policies from region to region. The square regions (1,5,6,8,9,12,15) are the possible start and goal regions.

VI. Experimental Evaluation

For each problem instance presented in Section V, we solve two POMDPs. The first POMDP directly describes the tasks as the tasks are discussed in Section IV. The second POMDP is a reduction of the direct description using our proposed algorithms. The two policies computed are compared to evaluate the effectiveness of the proposed reductions.

Both POMDP+Online and MDP+POMDP utilize a POMDP solver; the particular solver we used for our experiments is Monte-Carlo Value Iteration (MCVI) [21], as mentioned in Section II. MCVI is a recent point-based solver that uses the extremely successful particle filter to represent a non-Gaussian, potentially multi-modal distribution over the possible states the robot is in (a belief state). The experiments we perform represent challenging tasks at the limits of MCVI's solution ability. Pushing MCVI to its limitations emphasizes the ability of the reduction techniques we have proposed to extend the size of problem instances that can be solved.

Although many methods to drive a car-like robot to a specific goal without hitting obstacles can be used, we use the Open Motion Planning Library (OMPL) [28] for online planning in our experiments. Specifically, we use an implementation of Transition-based Rapidly-exploring Random Trees (TRRT) [34] as available in OMPL. TRRT extends the popular RRT [35] algorithm with rejection sampling. Rejection of collision-free connections is based on a Metropolis criterion for connections that increase cost; connections that decrease cost are always added. This simple change effectively searches for low-cost paths. An increasing 'temperature' parameter (nomenclature taken from simulated annealing literature) slowly raises the effective maximum cost increase after a number of failed expansions. The regions of space near the expected execution of the policy are decreased in cost, while the rest of the space

is left at 0 cost. This has the effect of encouraging the planner not to leave the policy, or, when it must, it will prefer regions that are part of the policy where possible. The POMDP reward function is not used in the TRRT cost function because TRRT does not know about the discrete actions and sensing; the robot needs to follow the solution policy as computed as closely as possible and not head myopically towards the goal.

TRRT is assumed to be run in an interleaved fashion during execution of the prior plan, so the robot performs a continuous motion. We assume the actions in the POMDP each take enough time to execute that the TRRT planner can run, and that an observation taken shortly before the end of the current POMDP action is sufficient to infer with (equivalent noise) the state from which TRRT should plan the next cycle from. For all experiments presented here, we will use 10s as a hard limit on TRRT time in each replanning cycle and assume that execution of the prior action took at least that long. TRRT is simply an instance of the Plan function that can return an approximate solution within the time limit; many other instances could be applied here. The choice of a sampling-based planner was influenced primarily by the desire for a robot-agnostic solution for fast experimentation. For any specific system, a controller could be used to navigate between states. Alternatively, a plan could also be produced by D*Lite [36] (optionally, post-processed by a trajectory optimization algorithm).

For the tests with online replanning, each data point will be plotted as the mean and 95% confidence interval calculated over 500 runs of the online phase. Both the POMDP directly describing the task, denoted P, and the reduced POMDP we propose, denoted P', are solved using MCVI and the policies are executed with online planning. Many runs are necessary not only because the true world states are sampled, but also because TRRT is a randomized algorithm. When simulating the transition probabilities in the MDP, similar to the tests with MDP+POMDP, 500 trials of the policy are executed to find an approximation of the true transition probabilities in the MDP problem specification. Solving the global MDP is deterministic, and therefore it is only run once.

A. Exposure Minimization

The online planner, OMPL, needs to respect the additional constraints in P, causing the online replanning time (Figure 6, top) to be higher than when the policies computed with P' are executed with OMPL. The online planner cannot choose to ignore constraints given in the POMDP policy, otherwise the policy update step may fail. While the policies computed for P fail to achieve positive reward, the reduced POMDP, P', achieves greater reward even though it is defined for a holonomic motion model far removed from the true bounded-curvature constraints (see Figure 6, center). The improved reward shows that replanning can successfully bridge the gap between the reduced solution and the true Reeds-Shepp [30] kinematics. This is consistent with our expectation that this reduction allows MCVI to generate a better solution given the same amount of time as a non-reduced POMDP. The POMDP solution time is 10,000 seconds (approximately 2.7 hours) for both P and P' as MCVI did not converge to optimal in either case.

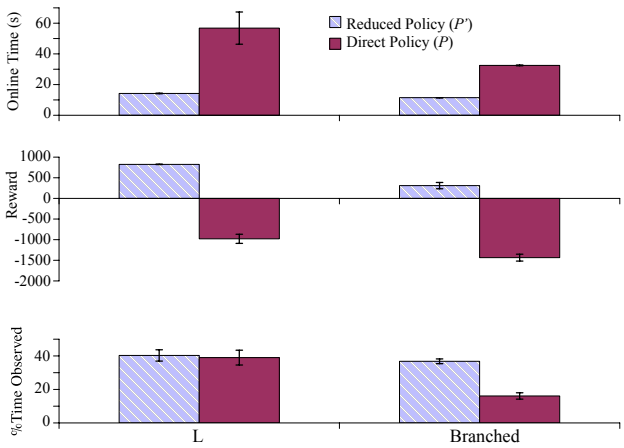


Fig. 6: Experimental results comparing the solution of a direct description of the task (P, solid) with the solution of our reduction of the task (P', striped) in the exposure minimization tasks, with $\gamma = 0.99$ over 500 trials each. Policies from both P and P' were run in replanning with a Reeds-Shepp car.

Note that, in the Branched environment, a lower percentage of time observed (Figure 6, bottom) is not correlated with higher reward as might be expected. This is because the dominant factor in this task is if the robot gets to the goal in the end. The observers play a significant role; however, being observed is less important if the policy cannot reach the goal a significant percentage of the time. The dominant reward factor is task-dependent, based on the particular values given to the reward for the goal and the penalty for being observed. Similarly, in the L environment, although the time observed is approximately equal, the rewards achieved are very different. Another factor to consider is exactly how the percentage of time observed is calculated here. We have taken the average over all successful runs, with different path length solutions being equally weighted. The reward for our reduced model is significantly higher; if we wanted the percentage of time observed to be guaranteed to be negatively correlated we could incorporate this in our reward function. However, the task we have proposed is to get to the goal at any cost, minimizing that cost when possible. The upper bound on possible reward in the Branched environment, given the minimum time to the goal of 10 actions, is $\gamma^{10} \cdot 1000 \approx 904$ when $\gamma = 0.99$. If we assume that applying the success rate in that environment, 69%, will account for the noise present in the system, the analysis has an expected maximum reward to $(1000 \cdot 0.69 + (-1000 \cdot (1 - 0.69))) \cdot \gamma^{10} \approx 343$. We consider the achieved reward of ≈ 310 to be near optimal, given that the maximum reward analysis assumed the robot takes the shortest possible path (which is not optimal due to the observer penalty). This simple evaluation in one environment is purely to show evidence that the method is not too far from optimal, and makes no attempt to rigorously account for the noise present.

The initial evaluation of the L environment showed that the robot was able to quickly use sensing to determine which topologically distinct path to follow. The Branched environment increases the number of distinct paths, as well as introducing

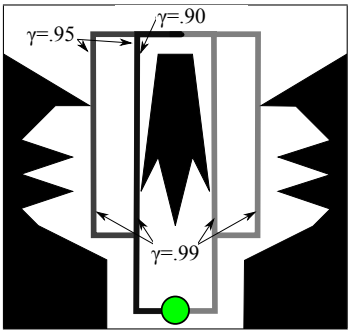


Fig. 7: Qualitative visualization of the different path classes executed by the policies generated in the Branched environment. Only a discount factor of $\gamma \geq 0.99$ matches the human-expected policy. Observer locations and vision polygons omitted here.

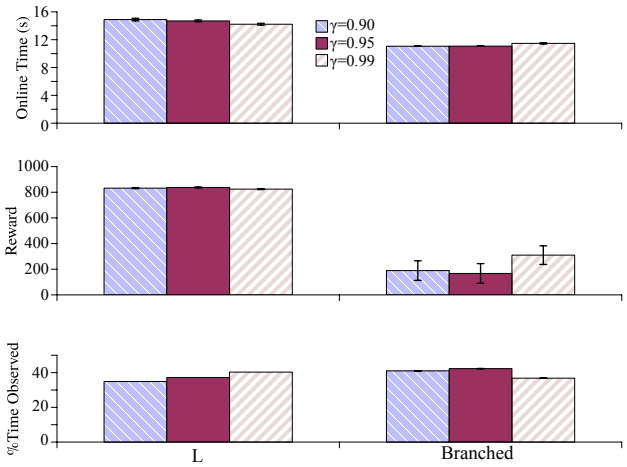


Fig. 8: Experimental results comparing three levels of the discount factor, γ, for the exposure minimization tasks. (500 trials each.)

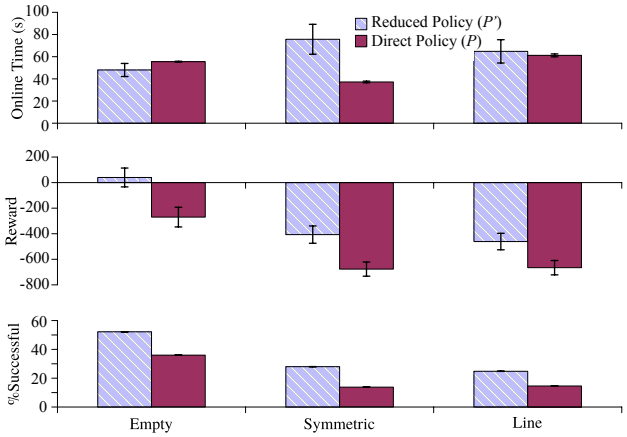


Fig. 9: Evaluation of POMDP+Online applied to the SAR tasks in three environments over 500 trials each.

harder sensing conditions due to selecting two of the six locations for observers. The Branched environment introduces significant masking of any observers farther away because the sensor model only observes the closest one.

Testing with the discount factor $\gamma = 0.90$ showed that only one path was ever taken in the Branched environment. Different levels of γ and the different path classes the policy executed in the Branched environment are shown in Figure 7. Further analysis showed that the time necessary to determine which of the two corridors to traverse was long enough to be suboptimal at $\gamma = 0.90$. Increasing γ to 0.95 still did not split the policy across the two corridors based on sensing, however, it did cause the action policy to choose a longer path in the right hand corridor. This avoided a costly double-exposure region. Finally, increasing γ to 0.99 computed the human-expected policy. The policy with $\gamma = 0.99$ includes the four distinct paths in Figure 7. Because the human-expected policy was found with $\gamma = 0.99$, this value was used throughout all other experiments.

The initial evaluation of γ under different conditions prompted a full set of experimental results for three experimental conditions of γ, shown in Figure 8. In the L environment (Figure 8, left columns), only very minor changes are seen in the exact paths taken. It is important to note that, although the reward is useful to compare these different policies, the reward evaluation in Figure 8 is computed with $\gamma = 0.99$. Each policy is, of course, optimal with respect to the γ that was used during the offline planning.

As seen in our evaluation of γ, determining the correct parameters for a POMDP model is in itself difficult if there is an expected policy to compare to. Careful analysis of the values involved can disambiguate between an error in the modeling of the tasks, or if the POMDP simply did find the optimal policy given the input parameters.

B. Search and Rescue

In our results for the search and rescue tasks, it is clear from overall reward (Figure 9, center row) and success rates (Figure 9, bottom row) that the policies computed by MCVI for both the direct description of the task (again, P) and the reduced POMDP (P') had only moderate success in solving the tasks. MCVI was given the same 10,000 seconds as in Exposure Minimization. The increased variance in online replanning time (Figure 9, top row) is due to some instances being fast to solve and others being solvable but requiring more discrete actions. The policies computed for P do not exhibit this increased variance because only the simpler instances could be solved. Overall, however, the reduced POMDP, P', significantly improved mean reward in these tasks.

The Empty environment serves as a baseline for the maximum expected success rate, and is less than 100% due to some observations being received that did not appear in the computed policy. It is possible that increasing the time allowed and the number of particles used to approximate belief in MCVI will help. In fact, we increased the allocated solution time several times and decided to stop at 10,000 seconds. It is unclear if there would be any benefit of performing these comparisons after a longer runtime.

The Symmetric and Line environments contain unknown obstacles, which pose an additional sensing challenge. As can be seen from the success rates and rewards in Figure 9, the

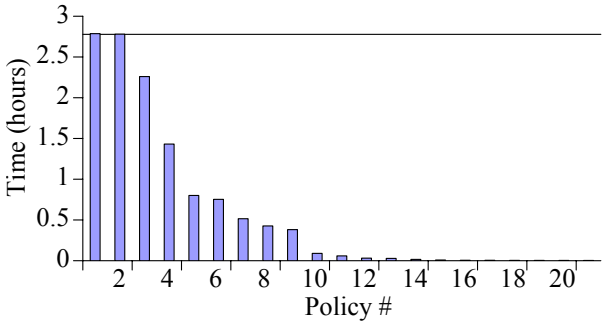


Fig. 10: The time to compute each of the 21 POMDP subtask policies for the Rooms environment, sorted by computation time. The line marks the timeout time. The x-axis is an arbitrary numbering of each unique policy.

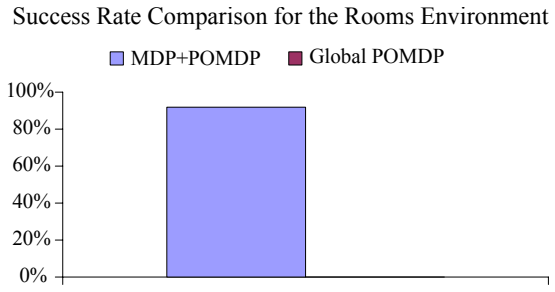


Fig. 11: The success rate of the MDP+POMDP solution compared to the single, global POMDP. The global POMDP was not able to find even an approximately optimal solution, having a 0% success rate and is therefore not visible.

policies were often inadequate to be successful in the task. That said, we note that the policy using our reduction (P') was approximately twice as successful as the policy generated using the direct representation of the task (P).

C. Indoor Navigation

In the indoor navigation tasks, the sensor model is only measuring the obstacles and if the robot is in a special bounded-observability region. This task class was designed to support a task decomposition in time, allowing the consideration of extremely long time horizon tasks. Our initial discussion will assume, $P = P'$, that is, the online replanning is not necessary because the true underlying robot model is assumed to be identical to the one that was used to construct the policy. This will ensure that all computational differences can be attributed to the MDP+POMDP algorithm. We will then validate that the POMDP+Online algorithm can still follow the computed policy using a car-like model.

We will evaluate our proposed MDP+POMDP run on the decomposition of the task as compared to solving one large, global POMDP that directly describes the task with a longer time horizon. The subtask POMDPs are run through a simulated execution (using the simple holonomic robot model) 1000 times to estimate the transition probabilities in the MDP. As the MDP is deterministic, we do not require multiple runs to evaluate the success percentage of the high-level policy.

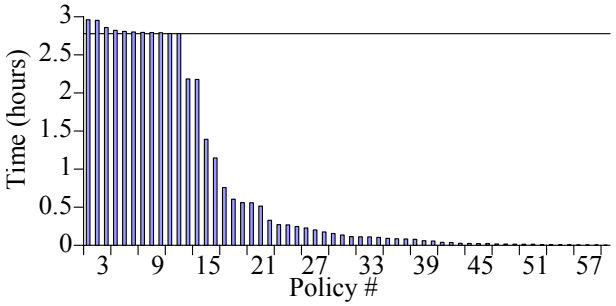


Fig. 12: The time to compute each of the 59 POMDP subtask policies for the Office environment, sorted by time. The line marks the timeout time (only checked once per iteration, therefore some times are longer). The x-axis is an arbitrary numbering of each unique policy.

In the Rooms environment, a time limit of 10,000 seconds (≈ 2.7 hours) was provided to each small policy. Although the 21 policies each get 2.7 hours, meaning the maximum computation time is $21 \cdot 2.7 = 56.7$ hours, the small policies often take much less than the time limit, as shown in Figure 10. The total computation time for all 21 policies was 12.4 machine-hours. An added benefit of this approach is that the 21 policies can be computed in parallel; the total time (assuming 21 computers were available) would be only 2.7 hours. To compare to a traditional solution, a single global POMDP instance was constructed and given 27 machine-hours. The global POMDP cannot take advantage of multiple computers with the current implementation of MCVI (though it is multi-threaded). The global POMDP was given a time limit of 27 hours, 10 times the computation time of a single policy, and well over the total time spent on all smaller policies. The MDP only has 12 states (the 11 regions described above, plus one failure state), and takes less than one second to solve for an optimal policy, so is negligible and not included in the time comparison.

The MDP+POMDP policy success rate in the Office environment is compared to the global POMDP solution in Figure 11. Although in theory, the global POMDP should be able to achieve higher overall success rates, the exponential complexity of the computation requires so much time that the policy computed over 27 hours is still insufficient to find a policy with a non-zero success rate; MDP+POMDP, by contrast, provided a very good success rate using a composition of local policies that took 12.4 hours to compute.

In the Office environment, many similar trends are seen. As in the prior results, Figure 12 shows that some policies are computationally difficult, but many are easy. The overall solution time is much less than the maximum $59 \cdot 2.7 \approx 160$ hours. The total time to compute the 59 small POMDP policies in this environment was 44 hours. For a fair comparison, the global POMDP was given 45 hours to solve this task.

As illustrated in Figure 13, the success rate of the solution computed by MDP+POMDP was significantly higher than the success rate of the global POMDP solution. Unlike the rooms example, the global POMDP did find a reasonably successful policy. Although the environment is more complex, the particular start/goal pair (the one depicted in Experiments,

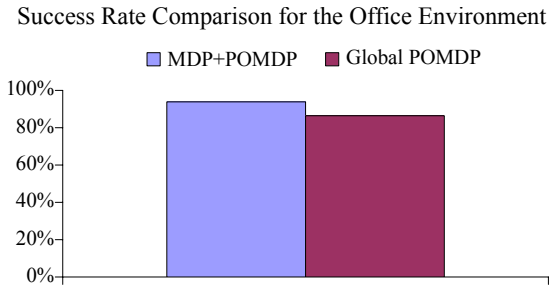


Fig. 13: The success rate of the MDP+POMDP solution compared to the single, global POMDP. The MDP+POMDP framework used 44 hours to compute a solution with 94% success rate; given 45 hours, the global POMDP policy had an 86% success rate.

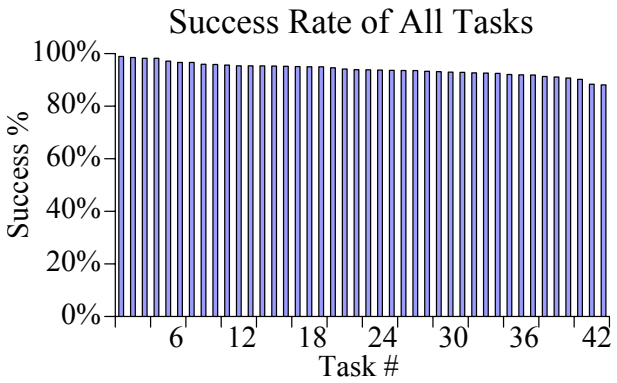


Fig. 14: The success rate of the MDP+POMDP solution across each of the 42 possible start/goal combinations, sorted by success rate. The x-axis is an arbitrary numbering of the possible start/goal combinations that define different tasks.

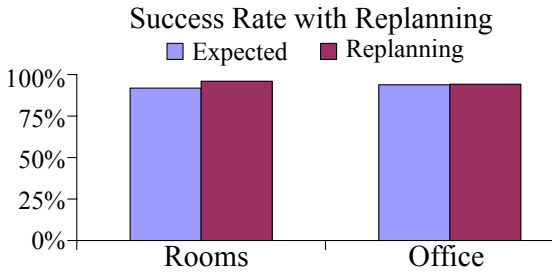


Fig. 15: The expected success rate of the MDP+POMDP solution assuming a holonomic robot, compared to the success rate of the Reeds-Shepp robot using online replanning. The success rate improves because small deviations can be fixed in the online replanning step and hitting an obstacle is non-terminal.

Figure 4) does not require as long of a time horizon to solve. The existing, general heuristics present in MCVI are enough to avoid exploring the significant areas of the belief space that would never be entered under an optimal policy that only passes through 5 out of the 9 rooms. Even though the maximum time horizon specified was the same, MCVI was able to find that it was unnecessary to plan that far into the future.

The global POMDP solution is only for the particular start/goal combination depicted previously. This global POMDP solution does not provide any information to decrease the time to solve any other start/goal pair. Because of the excessive time required (80 machine-days) and the low expected utility of doing so, a global POMDP was not run for 45 hours per each of the 42 start/goal pairs. Rather, we report only the MDP+POMDP success rate for each of these possible tasks in Figure 14. The time to run all 42 tasks at the MDP level was under one half of a second, and the success rate for each task was never below 80%. Computing the solution of all possible tasks may be unnecessary in many applications; however, we expect that for indoor navigation tasks, there may be many queries for a variety of tasks and precomputing all solutions is a significant benefit. To restate this result, MDP+POMDP required 44 hours to solve all 42 task instances and even provided a better solution than the global POMDP given similar total solution time per task instance.

Having determined that the MDP+POMDP algorithm can provide significant computational benefits, we applied the POMDP+Online algorithm with the same Reeds-Shepp robot model used previously. The proof-of-concept evaluation of applying both algorithms in the Rooms and Office environments are shown in Figure 15; for the Office environment only the task depicted in Figure 4 was tested. The success rate is very close to the expected success rates computed with the holonomic model; the improvement occurs because the online replanning system can recover from rare, but large, disturbances in position.

VII. Discussion

The proposed techniques for reduction of POMDPs and execution of policies generated using these reduced POMDPs have been successfully applied to three robotic task classes in this paper. Each task uses navigation toward a goal location, and two task classes extend navigation to more complex task classes requiring additional sensing. Our results over a variety of environments and task classes support our supposition that the reduction of a robotic task in the POMDP formulation can be effectively used for a variety of robotic tasks, which are often constructed from a combination of sensing and navigation. We presented two algorithms, POMDP+Online and MDP+POMDP, which perform a reduction to the input POMDP and provide a method to use the reduced solution in the original problem.

In our evaluation of the POMDP+Online algorithm, we see that the reduction of the motion model considered at the POMDP stage produces a superior policy. The online replanning stage can effectively support execution of the policy, even though it was computed for a different state space with significantly simpler constraints. Although planning on a simplified dynamics model required additional online computation, the time is negligible compared to the offline time savings that, in many cases, allows us to find a significantly better offline policy than was otherwise possible.

In particular, the exposure minimization tasks showed excellent performance improvement when using the POMDP+Online algorithm. For the exposure minimization tasks, we also analyzed the significant effects of varying the discount factor, γ. This type of parameter sweep is rarely seen in POMDP

literature although occasional reference to the difficulty of choosing γ can be found.

The experimental results in the SAR tasks were less successful in absolute terms, though the reduced model continued to provide significant relative improvements. Analysis of the failures pointed out the need for more robust policies. A possible area of future investigation is a more robust policy execution strategy. One possibility may be coupling replanning with an online POMDP process and/or a true state estimator that can fix the policy. There is promising active research in online POMDP planning [37]. However, this robotic POMDP application is at a short-term reactive cycle stage, only looking forward in time at most 4 actions in the examples provided. In this paper, even the smallest tasks we investigated had a discrete time horizon an order of magnitude greater. Therefore, we suspect that a fundamentally different mechanism from the one presented in [37] will be required for these tasks, and believe that the approximation algorithms presented in this paper may serve as a starting point for a new approach.

In our experiments the model reduction in the POMDP+online algorithm was applied to a first-order car. We expect that similar reductions can be created for other small-time locally controllable systems. How well this would work in practice is the subject of future work. It would be interesting to see what the tradeoffs are between solution quality and computation time as the complexity of dynamics (both in the task and in the simplification) increases. For systems with more complex dynamics (such as second-order dynamics) a reduction to a completely holonomic system might result in policies that are difficult to execute. In such cases a reduction to a first-order system with a discrete number of actions might provide a reasonable tradeoff between solution quality and computation time. Furthermore, as the underlying system becomes more complex, there exists the distinct possibility of pathological regions of the state space. For example, a system with complex legs may be able to fold up and fall over such that it cannot escape the local region of the state space and can no longer move effectively, like a turtle on its back. A simplified model may not capture these pathological regions accurately and the policy may require passing through one of them, making the offline solution an infeasible one.

Finally, our proposed MDP+POMDP algorithm showed huge performance benefits in the indoor navigation tasks. MDP+POMDP requires the addition of a sensor that bounds localization relatively tightly so that the prior history can be forgotten without significant degradation in overall performance. We propose a doorframe sensor as a model of how this might be accomplished in indoor environments. This sensor, and the subsequent application of an MDP to construct a high level policy together allow the consideration of multi-query POMDP tasks, a problem domain that, to the authors knowledge, has very little existing work outside of reinforcement learning. Given the extreme performance benefits in the multi-query POMDP tasks, this is a very interesting idea to pursue. A natural question that arises is if the state space can always be assumed to have obvious bounded-observable regions. Determining such regions is likely to be at least as hard as solving a POMDP on the underlying state space, but an automated sensor model analysis and studying how much the requirement of bounded-observability can be softened or broken are promising directions for future research. A natural generalization to explore would be if the MDP was formulated as a Semi-Markov Decision Process (SMDP) instead. The SMDP allows us to consider the expected time to execute a policy, which is a random variable, in a principled way, instead of only optimizing for overall success. Future work would also investigate additional task classes. For example, in object manipulation there is a natural subtask decomposition (grasp, transfer, release) in the time domain that seems with clear boundaries between these subtasks, or other tasks that are solved with more complex reward functions than just entering a particular region of state space (or sequence thereof such as grasp/transfer/release).

REFERENCES

[1] E. J. Sondik, "The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs," *Operations Research*, vol. 26, no. 2, pp. 282–304, Apr. 1978.

[2] S. Zhang and M. Sridharan, "Active visual sensing and collaboration on mobile robots using hierarchical POMDPs," in *AAMAS*, 2012, pp. 181–188.

[3] L. P. Kaelbling and T. Lozano-Perez, "Integrated task and motion planning in belief space," *The Int. Jour. of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, Jul. 2013.

[4] H. Kurniawati, Y. Du, D. Hsu, and W. S. Lee, "Motion planning under uncertainty for robotic tasks with long time horizons," *Intl. J. of Robotics Research*, vol. 30, no. 3, pp. 308–323, 2011.

[5] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun, "Towards robotic assistants in nursing homes: Challenges and results," *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 271–281, Mar. 2003.

[6] C. Papadimitriou and J. Tsitsiklis, "The complexity of Markov decision processes," *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.

[7] T. Smith and R. Simmons, "Point-based POMDP algorithms: Improved analysis and implementation," in *Uncertainty in AI*, Jul. 2005.

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

[9] T. Lee and Y. J. Kim, "GPU-based motion planning under uncertainties using POMDP," in *IEEE Int. Conf. on Robotics and Automation*, Karlsruhe, Germany, 2013, pp. 4576–4581.

[10] K. Hsiao, L. P. Kaelbling, and T. Lozano-Perez, "Grasping POMDPs," in *IEEE Int. Conf. on Robotics and Automation*, Apr. 2007, pp. 4685–4692.

[11] D. K. Grady, M. Moll, and L. E. Kavraki, "Combining a POMDP Abstraction with Replanning to Solve Complex, Position-Dependent Sensing Tasks," in *AAAI Fall Symp.*, Arlington, Virginia, 2013.

[12] A. Foka and P. Trahanias, "Real-time hierarchical POMDPs for autonomous robot navigation," *Robotics and Autonomous Systems*, vol. 55, no. 7, pp. 561–571, Jul. 2007.

[13] J. Pineau, N. Roy, and S. Thrun, "A hierarchical approach to POMDP planning and execution," in *Workshop on Hierarchy and Memory in Reinforcement Learning*, 2001.

[14] H. Kurniawati and N. M. Patrikalakis, "Point-based policy transformation: Adapting policy to changing POMDP models," in *Workshop on the Algorithmic Foundations of Robotics*, 2012, pp. 1–16.

[15] M. Wang and R. Dearden, "Improving Point-Based POMDP Policies at Run-Time," in *Workshop of the UK Planning And Scheduling SIG*, Middlesbrough, UK, 2012.

[16] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online Planning Algorithms for POMDPs," *Journal of Artificial Intelligence Research*, vol. 32, no. 2, pp. 663–704, Jul. 2008.

[17] H. Bai, D. Hsu, and W. S. Lee, "Planning how to learn," in *IEEE Intl. Conf. on Robotics and Automation*, May 2013, pp. 2853–2859.

[18] J. van den Berg, S. Patil, and R. Alterovitz, "Efficient approximate value iteration for continuous Gaussian POMDPs," in *AAAI Conf. on Artificial Intelligence*, 2012, pp. 1832–1838.

[19] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Advances in Neural Information Processing Systems 23*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., 2010, pp. 2164–2172.

[20] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 1772–1780.

[21] H. Bai, D. Hsu, W. S. Lee, and V. Ngo, "Monte Carlo value iteration for continuous-state POMDPs," in *Workshop on the Algorithmic Foundations of Robotics*, 2010, pp. 175–191.

[22] S. Candido and S. Hutchinson, "Minimum uncertainty robot navigation using information-guided POMDP planning," in *IEEE Int. Conf. on Robotics and Automation*, May 2011, pp. 6102–6108.

[23] G. Theocharous, K. Murphy, and L. Kaelbling, "Representing hierarchical POMDPs as DBNs for multi-scale robot localization," in *IEEE Intl. Conf. on Robotics and Automation*, vol. 1, April 2004, pp. 1045–1051.

[24] A.-a. Agha-mohammadi, S. Chakravorty, and N. M. Amato, "FIRM: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements," *Intl. J. of Robotics Research*, vol. 33, no. 2, pp. 268–304, Feb. 2014.

[25] H. Kurniawati, T. Bandyopadhyay, and N. M. Patrikalakis, "Global motion planning under uncertain motion, sensing, and environment map," *Autonomous Robots*, vol. 33, no. 3, pp. 255–272, Jun. 2012.

[26] R. Kaplow, A. Atrash, and J. Pineau, "Variable resolution decomposition for robotic navigation under a POMDP framework," in *IEEE Int. Conf. on Robotics and Automation*, May 2010, pp. 369–376.

[27] D. K. Grady, M. Moll, and L. E. Kavraki, "Automated Model Approximation for Robotic Navigation with POMDPs," in *IEEE Int. Conf. on Robotics and Automation*, Karlsruhe, Germany, 2013, pp. 78—–84.

[28] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012, http://ompl.kavrakilab.org.

[29] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*, 3rd ed. Prentice hall Englewood Cliffs, 2009, vol. 74.

[30] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[31] Z. Lim, D. Hsu, and W. S. Lee, "Monte Carlo value iteration with macro-actions," in *Advances in Neural Information Processing Systems*, 2011, pp. 1287–1295.

[32] M. R. Maly, M. Lahijanian, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal motion planning for hybrid systems in partially unknown environments," in *ACM Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*, Philadelphia, PA, USA, 2013, pp. 353–362.

[33] E. Olson, "AprilTag: a robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, May 2011, pp. 3400–3407.

[34] L. Jaillet, J. Cortes, and T. Siméon, "Transition-based RRT for path planning in continuous cost spaces," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sep. 2008, pp. 2145–2150.

[35] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[36] D. Ferguson and A. Stentz, "Field D*: An interpolation-based path planner and replanner," in *International Symposium of Robotics Research*, ser. Springer Tracts in Advanced Robotics, S. Thrun, R. Brooks, and H. Durrant-Whyte, Eds., vol. 28. Berlin, Heidelberg: Springer, 2007, pp. 239–253.

[37] J. Pajarinen and V. Kyrki, "Robotic manipulation of multiple objects as a POMDP," *CoRR*, vol. abs/1402.0649, 2014. [Online]. Available: http://arxiv.org/abs/1402.0649

Devin K. Grady (M'12) received the Ph.D. degree in computer science in 2014 from Rice University in Houston, TX, USA. He is currently a software engineer at Google.

His research interests include robotic decision making, motion planning, sampling-based algorithms for high-dimensional systems and collaborative robotics, among other planning and policy-generation tasks for challenging computational problems.

Mark Moll (M'99—SM'13) received the Ph.D. degree in Computer Science at Carnegie Mellon University in 2002. He is an adjunct assistant professor and research scientist in computer science at Rice University. His current research interests include robot motion planning, modeling of protein motion, and geometric problems in robotics and structural computational biology.

Lydia E. Kavraki (M'93—SM'11—F'12) received the Ph.D. degree in computer science from Stanford University, Stanford, CA, in 1995, and is the Noah Harding Professor of computer science and bioengineering at Rice University, Houston, TX. She is an author of over 200 papers and co-authored a robotics textbook: Principles of Robot Motion (MIT Press, 2005). Her research interests include motion planning for continuous and hybrid systems, mobile manipulation, and applications of robotics methods to biology. Prof. Kavraki is a Fellow of the IEEE, AAAI, AIMBE, ACM and AAAS, as well as an Elected Member of the IoM.