Locally Observable Markov Decision Processes

Max Merlin, Neev Parikh, Eric Rosen, George Konidaris

***Abstract*—Real-world robot task planning is computationally intractable in part due to the complexity of dealing with partial observability. One approach to reducing planning complexity is to assume additional model structure such as mixed-observability, factored state representations, or temporally-extended actions. We introduce a novel structured formulation, the *Locally Observable Markov Decision Process*, which assumes that partial observability stems from limited sensor range—objects outside sensor range are unobserved, but become fully observed once they are within sensor range. Plans solving tasks of this type have a specific structure: they must necessarily go through *localities* where objects transition from unobserved to fully observed. We introduce a novel planner that reduces planning time via a hierarchy that structures the plan around these localities, and interleaves online and offline planning. We present preliminary results in a challenging domain that shows that the locality assumption enables robots to plan effectively in the presence of this type of uncertainty.**

I. Introduction

It is unreasonable to expect that a robot can, at every moment, sense all of the aspects of its environment required for decision-making. Task planners that hope to be useful in the real world must therefore cope with the environmental uncertainty caused by such partial observability. Unfortunately, planning in the presence of generic environmental uncertainty is NP-Hard [6], resulting in scalability and performance problems in practice [14].

One approach to avoiding this complexity is to assume additional model structure that can be exploited to enable efficient solutions. An important characteristic of real, physical sensors is that they are *range-limited*; that is, they are only able to sense—often very accurately—objects within a fixed maximum distance. This limited range is a major source of partial observability in real robot tasks. All of the objects the robot wishes to interact with *can* be perceived directly, given appropriate sensor placement. Moreover, the robot is typically configured such that interaction with an object requires that object to be within sensor range, and ensures that it remains observed for the duration of the interaction.

We therefore introduce the Locally Observable Markov Decision Process (LOMDP), a formalism for decision-making under uncertainty that models partial observability stemming from range-limited sensing. The core assumption behind LOMDPs is that an object's state is only uncertain because it has never been within range of the robots sensors, but navigating close enough to the object will bring it into sensor range, thus making it fully observed. We formalize this notion of "close enough" as a *locality*—the set of states from which an object can be observed, which may depend on the object's own (unobserved) attributes. Any plan that solves a LOMDP has a specific structure: it must necessarily navigate through the localities of the objects relevant to the solution, which transitions those objects from partially to fully observed.

We exploit this structure to design a hierarchical planner called **Planning to Find** (P2F). It generates a plan offline to interact with objects that have been observed, but defers offline planning over unobserved objects. During online execution, P2F will plan to observe certain unobserved objects and, once they become observed, how to interact with them.

II. Background and Related Work

A Partially Observable Markov Decision Process (or POMDP) is a sequential decision-making problem where an agent must choose an action to execute at every timestep, but in which the agent cannot always observe all the information necessary to make that decision. Formally, a POMDP is defined as a tuple $(S, A, \Omega, O, T, R, \gamma)$, where S is a set of states, which the agent cannot observe but which govern the dynamics of its environment; A is a set of actions, one of which it must choose to execute at each timestep; Ω is an observation space, describing the agent's sensor space; $O(\omega|s)$ is an observation function describing the probability with which the agent obtains observation $\omega \in \Omega$ when in state s; $T(s'|s,a)$ is the transition function, describing the probability that the agent's choice of action a in state s causes the environment to transition to state s'; $R(s,a,s')$ is the reward received by the agent for transitioning from state s to s' by executing a; and γ is a discount factor, which describes the agent's preference for immediate over future reward. The agent must choose actions at every timestep to maximize its expected discounted sum of rewards, even though it never observes the state s directly, and can only infer it using the information returned by observation function O.

There has been significant work related to planning with uncertainty [5, 8, 11, 2]. Here we review a few key works most similar to our approach. Because generating optimal plans for partially observable environments is intractable, previous work has attempted to leverage modeling assumptions to reduce complexity. We draw inspiration from the Mixed Observability MDP (MOMDP) [9], where the state is partitioned into factors (state variables) that are fully observable and factors that are partially observable. While MOMDPs capture some of the modeling aspects we propose by representing certain factors as fully observable, each factor is always either fully or partially observable and cannot change from one to the other. In contrast, Petrick and Bacchus [10] provides a structure to allow state variables to transition from unknown to known. However, our LOMDP framework provides a more extensible and accessible model rooted in the POMDP formulation.

Other approaches focus on developing specialized planners to handle uncertainty. Weld [15] introduced Partial Order Planner (POP), a least-commitment planning method that introduces the concepts of partial ordering and causal link protection. POP is a goal-regressive planner [12], working from goal states to start states. There are extensions improving on POP [13, 1, 3, 7] that handle stochastic state transitions and partial observability. One particularly prominent work in this line of research is Hierarchical Planning in the Know (HPK) [4], which uses a form of goal regression to hierarchically decompose a task. It makes optimistic assumptions about the state of the world and the outcome of its actions, committing to those assumptions to generate a full plan. Although the optimistic nature of HPK allows it to aggressively commit to high-level subgoals for quick planning and execution, it potentially requires replanning when an optimistic assumption is violated, which could be very costly.

III. Locally Observable MDPs

The POMDP formulation is can model very general hidden state variables and complex observation functions. Since that generality comes with a severe complexity penalty, a promising approach to designing efficient planners is to incorporate structural assumptions specific to real-world robot systems. We focus on the fact that robot sensors are typically range-limited and line-of-sight, which means that their observations can be modeled as highly accurate within a given fixed sensor range, assuming that the object is not occluded, and effectively uninformative beyond that range. Once observed, an object transitions from an uncertain to a known state and, assuming there are no other agents present, will remain in that known state unless directly modified by the robot itself. Additionally, manipulation requires observation: it is the case that a robot can only manipulate an object it can directly observe, and can observe the object for the duration of manipulation.

Consider a home service robot tasked with making a sandwich. Doing so requires the robot to gather the different ingredients from closed cupboards, bring them to a countertop, and then to construct the sandwich. The only source of partial observability in the problem comes from the robot's range-limited, line-of-sight sensors. The robot does not know which cupboard each ingredient is in, because the internal state of each cupboard is occluded by the cupboard itself. Even if the cupboard doors were open, only certain poses would allow the robot to see inside.

Here, observing the ingredients from a nearby robot pose would be sufficient to decisively establish their state; even if the robot possesses noisy sensors, it is not difficult to gather multiple views of an object of interest—once it is within range—to reach practical certainty via active perception.

All the uncertainty in this problem has the structure that the partially-observed states can be completely resolved by having the robot enter certain states, specifically for the purpose of observation. Consequently, rather than solving a general POMDP, the robot should instead model unobserved quantities that it can *choose to observe* by navigating to a pose from which they are visible.

We formalize the LOMDP as a tuple: $(S, A, O, \Omega, L, T, R, \gamma)$, where S is the set of states, A is a set of actions, O is an observation function, Ω is an observation space, T is a transition function, L is the locality function, R is the reward function, and γ is the discount factor. Most of the elements are the same as the standard POMDP formalization, though state and observations have a more object-oriented structure, and one element—L—is novel to LOMDPs. We now consider each element in turn.

Since LOMDPs are inherently focused on objects and observing them, the state space S is factored into n independent objects: $S = f_1 \times ... \times f_n$. For ease of terminology, we assume each factor represents an object with a single attribute or the robot state, though our framework is by no means restricted to this case, and we use the terms "objects" and "factors" interchangeably.

At each timestep, an agent solving a LOMDP receives an observation where each object is either accurately observed or completely unobserved. The observation space for a LOMDP is therefore $\Omega = \{R \cup \phi\}^n$, a vector of n elements where n is the number of factors in S, and each element in Ω is either a real number—that factor's true value—or ϕ (i.e., not observed). Therefore, when an observation $O(s)$ gives accurate information about factor i, the value at $O(s)[i]$ is equal to the value of f_i in the current state s.

We next turn to the observation function, which in a POMDP is of the form $O(o \mid s', a)$, returning a probability distribution over observations given a state and action. In LOMDPs observations are deterministic, so it is unnecessary to return a probability distribution. Instead, the observation function deterministically maps a given state and action to an observation: $O : (s', a) \rightarrow \omega \in \Omega$.

In addition to these standard POMDP elements, LOMDPs have a new element, the *locality function*, which takes in a state factor and outputs the set of states from which that factor can be perfectly observed. This can be derived from the observation function. If the robot is in the locality of a given i, the observation it receives for that factor is that factor's true value v. If the robot is outside of the locality of i, it observes ϕ, (i.e. no observation). We therefore define a locality as:

$$L(i, v) = \{s \in S \mid O(s)[i] = s[i] \wedge s[i] = v\} \quad (1)$$

$$L(i) = \{s \mid \exists v, s \in L(i, v)\}, \quad (2)$$

where $L(i, v)$ is the locality from which state variable i can be observed if it has value v (Fig. 1.A), and $L(i)$ is the global locality for state factor i (Fig. 1.B).

The locality function allows us to redefine the observation function in a more structured way:

$$o[i] = \begin{cases} s[i] & \text{if } s \in L(i) \\ \phi & \text{otherwise.} \end{cases} \quad (3)$$

The definition of the observation function is dependent on the locality function, which is in turn dependent on the observation function; for any specific domain one must be given.

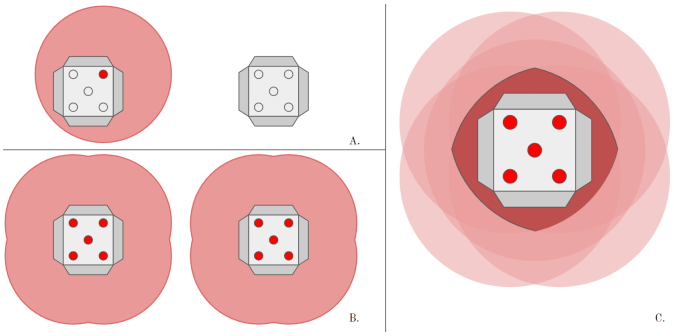

Fig. 1. A domain where a red ball can be in several locations within one of two boxes. *(A)* The larger red area shows the robot states $L(i, v)$ of one possible v for the ball i. *(B)* Shows the union of $L(i, v)$ for all possible v. This is the same as $L(i)$. *(C)* $L(i, v)$ for the 5 possible v for one box. The darkest red shows the intersection, from which the ball could be seen if it is any of those five values.

LOMDPs assume additional structure in the relationship between actions and the observation function. We assume that any object interaction requires observing the factor corresponding to that object, and that the robot can observe that factor for the duration of the interaction.

Because interaction requires observation, a robot must navigate to the locality of an object before interacting with it. Unfortunately, we cannot simply navigate to any state in $L(i)$ to interact with object i, since the states from which we can observe i depend on its unobserved value. We therefore require a representation of localities that enables the robot to navigate to the locality of a given factor even when its value is unknown. To do so, it is helpful to identify discrete sets of states (which we term *locales*) from which a range of values of factor i can be observed, and to find the collection of locales that collectively cover any setting of state factor i. The object must be observable from at least one of these locales, so navigating to all of the locales of a factor in any sequence guarantees that it becomes known.

In order to construct the locales, we apply a $mask(s, i)$ to the states in $L(i, v)$ to obtain the set of navigable states from which we could observe i if the value of i is v: $\{mask(s, i) \mid s \in L(i, v)\}$. The $mask(s, i)$ takes the state s and outputs the set of states where factor i can take on any value in range, but all other factors remain the same value. This resulting set of states represents all states which are identical to the original state s except for the value of factor i. This allows us to use the states in $L(i, v)$ as subgoal states despite not knowing the actual value of i. Doing this for all values of i, however, we may find that the resulting masked states for many different v's have a significant number of shared states, and hence certain states can confirm or deny many possibilities at once. We can therefore use a subset of these states that fully spans all possible values of i as our locales, which is sufficient for planning.

This is illustrated in Fig. 1.C, where masking out the factor for the ball location leaves us with the set of states represented by the larger, translucent red circles. These circles have significant overlap, and the subset represented by the intersection (darkest red) can be used to observe the ball in all five v values in this box. This is a perfect example of what we would use as a locale. Since navigating to any of the other states around the box would be redundant, we only ever need to plan to a state within that locale (the darkest red area).

Our planner only requires a set of locales that give complete coverage over all values for factor i, but the problem of selecting an optimal spanning set is potentially NP-hard. For the purposes of this paper, we provide the locales to the planner as part of the domain.

IV. Planning

Now that we have a set of locales,the robot can reach the localities of objects that we need to interact with. It need only identify which localities it needs to enter over the course of its plan. The hierarchical nature of goal regression planning is a perfect fit for this, as it can determine which objects we need for our plan before fully resolving how to interact with them. Even though the robot has not fully completed the task plan, it can compute which objects must be observed and can execute the portion of its plan that transitions them to observed (searching through their locales). Once the object has become observed, the remainder of the plan regarding that object can be calculated based on its true observed state and the current state. This eliminates the need to reason about the possible values that unobserved objects could take, and therefore it only generates plans which interact with observed factors and search for unobserved factors.

We contribute a novel planner as an extension to POP [15] called P2F (Planning to Find), that exploits the properties of LOMDPs to generate hierarchies that reduce planning time. We extend a goal regression method because it allows the robot to recursively evaluate actions that are required by its goal state. If any action the plan requires has preconditions that are unknown, a generic *find* action can be inserted into the plan that will resolve that precondition online. After inserting the *find* action, P2F considers that precondition suspended and continues with plan resolution. While POP requires all preconditions be satisfied before returning a plan, P2F only requires that all unsuspended preconditions be satisfied, after which it can begin executing the resulting incomplete plan. When a *find* action comes up in plan execution, the *find* subroutine extracts which object state the suspended precondition is over and plans to navigate to one of the possible locales of that object. If the object is not observed from that locale, another locale is selected to plan towards. This continues until the object is observed. Once the object state is known, a new instance of P2F is recursively called within the *find* subroutine to achieve the suspended precondition, after which can resume the overall plan. P2F therefore interleaves online and offline planning, since each subplan is resolved during execution.

V. Results

Our test domain is the peanut butter and jam (PBJ) domain. As in the example sandwich domain, the robot begins in a kitchen and is given the goal of making a sandwich.

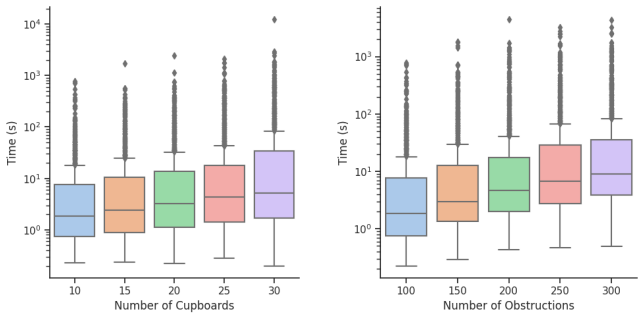

Fig. 2. Task completion time for PBJ domain using P2F over 1000 trials. Left: With 100 obstructions, varying the number of cupboards. Right: With 10 cupboards, varying the number of obstructions.

Making the sandwich requires spreading peanut butter on bread, spreading jam on bread, and then putting the two halves together. In order to perform each spread action, the jelly or peanut butter must be on the table along with the bread and the knife. However, all of these ingredients are distributed in closed cupboards and cannot be observed at the start of the task. In order to see each objects the robot must be in one of its locales, which we defined as the robot being in front of the cupboard that the object is in and that cupboard door being open. Additionally, there are some number of obstacle objects that are randomly distributed within the cupboards, and the robot may need to take obstacles out of the cupboard and place them on the counter below the cupboard to reach an object that it needs.

A forward planner would need to hypothesize about every possible combination of objects in each cupboard, and what it may need to do in order to make a specific item reachable in each scenario. Instead, P2F suspends planning interactions with unknown object until after the required object is actually observed. As a direct consequence of locales, once a requisite object is observed, the other objects in that same cupboard are observed as well, allowing P2F to plan only over the true locations of the relevant obstacles.

We are able to scale up the number of cupboards and number of obstacles in our domain to observe the performance of our planner (Fig. 2).

VI. Conclusion

LOMDPs are a compact model of environments that enables agents to explicitly represent states while accounting for limited sensor range with accurate sensor outputs. While planning in partially-observed environments is typically intractable, LOMDPs provide the structure to plan to observe objects, after which they are known and can be reasoned about as such. To exploit the structure of LOMDPs we propose a novel planner, P2F, which combines hierarchical goal regression with the localities defined in our LOMDP framework to delay planning over unknown factors. We plan to demonstrate the power of our model and planner by comparing it to existing state-of-the-art planners in a sandwich-making domain. Future work will consist of implementing the planner on a real robot, and extending to additional domains.

References

[1] Patrick Bechon, Magali Barbier, Guillaume Infantes, Charles Lesire, and Vincent Vidal. HIPOP: Hierarchical Partial-Order Planning. In *Proceedings of the 7th European Starting AI Researcher Symposium*, pages 51–60, 2014.

[2] Jilles Steeve Dibangoye, Guy Shani, Brahim Chaib-Draa, and Abdell-Illah Mouaddib. Topological order planner for POMDPs. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[3] Denise Draper, Steve Hanks, and Daniel Weld. A probabilistic model of action for least-commitment planning with information gathering. In *Uncertainty Proceedings*, pages 178–186. Elsevier, 1994.

[4] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Planning in the know: Hierarchical belief-space task and motion planning. In *Workshop on Mobile Manipulation IEEE Intl. Conf. on Robotics and Automation*, 2011.

[5] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227, 2013.

[6] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[7] Nicholas Kushmerick, Steve Hanks, and Daniel Weld. An algorithm for probabilistic least-commitment planning. In *AAAI-94: Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1073–1078, 1994.

[8] Andrew Kachites McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1996.

[9] Sylvie CW Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 29(8):1053–1068, 2010.

[10] Ronald PA Petrick and Fahiem Bacchus. Extending the knowledge-based approach to planning with incomplete information and sensing. In *International Conference on Automated Planning and Scheduling*, 2004.

[11] Joelle Pineau, Nicholas Roy, and Sebastian Thrun. A hierarchical approach to pomdp planning and execution. In *In ICML Workshop on Hierarchy and Memory in Reinforcement Learning*, 2001.

[12] John L Pollock. The logical foundations of goal-regression planning in autonomous agents. *Artificial Intelligence*, 106(2): 267–334, 1998.

[13] Oscar Sapena, Alejandro Torreño, and Eva Onaindía. Parallel heuristic search in forward partial-order planning. *The Knowledge Engineering Review*, 31(5):417–428, 2016.

[14] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, Cambridge, Mass., 2005.

[15] Daniel S Weld. An introduction to least commitment planning. *AI Magazine*, 15(4):27–27, 1994.