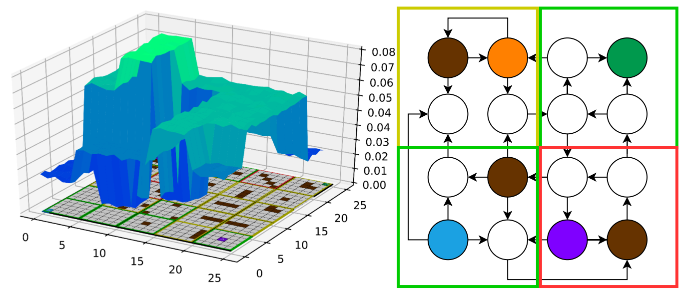
Local-Global Interval MDPs for Efficient Motion Planning with Learnable Uncertainty

Jesse Jiang, Ye Zhao, and Samuel Coogan

Abstract— **We study the problem of computationally efficient control synthesis for Interval Markov Decision Processes (IMDPs), that is, MDPs with interval uncertainty on the transition probabilities, against tasks specified in linear temporal logic. To address the scalability challenge when synthesizing this control policy in a holistic way, we propose decomposing the monolithic global IMDP into a collection of interconnected local IMDPs. We focus on the problem of robotic motion planning. Specifically, we assume a setting in which the transition probabilities can be learned and their interval uncertainty reduced by observing the dynamics of the system at runtime. This creates an objective of exploration to ensure that the planning task can be completed with sufficient probability of success. We perform decoupled exploration and learning on the local IMDPs and then combine local control policies to guarantee global task satisfaction. In a simulation-based case study, we show that, compared to existing approaches, our proposed decomposition leads to faster learning and satisfaction of the planning task and provides a feasible controller when other methods are infeasible.**



(a) **(b)**

Fig. 1. Visualization of the problem setup with robotic motion planning motivation. **(a)**: The unknown terrain elevation which results in motion perturbation is overlaid over the 2D x–y plane which depicts the grid used for IMDP abstraction and motion planning. **(b)**: The graph representation of the LG-PIMDP is shown. Computations are performed on the local regions enclosed in boxes, with PIMDP states represented as circles. Interactions between the regions are captured by the edges which cross region borders.

I. Introduction

Markov Decision Processes (MDPs) have been widely used for robotic motion planning tasks [1], [2]. In particular, abstraction-based planning methods utilizing MDPs are applied to tasks specified using the language of Linear Temporal Logic (LTL) [3], [4]. More recently, Interval MDPs (IMDPs) have been explored in the literature to model stochastic or uncertain dynamics [5]–[7]. Gaussian process (GP) [8] learning of uncertainties has proven especially well-suited for IMDP planning approaches [9]. In robotics, GP learning has been applied effectively to quantify environmental uncertainty such as terrain variation [10], [11].

A major challenge with abstraction-based control synthesis techniques is the curse of dimensionality arising from the poor scaling of these algorithms with the size of the state space [12], [13]. One approach that has been proposed to address this problem for MDPs is the use of partitioning [14] to reduce computational burden. Another approach is the hierarchical MDP [15], [16], which further abstracts the MDP to improve scaling with problem size.

In this work, we propose a novel local-global IMDP framework to enable computationally efficient motion planning on robotic systems with formal guarantees on system behaviors with respect to complex specifications. We consider a scenario in which unknown environmental features create control error and thus uncertainty in the system dynamics, and the objective is to learn the uncertainties and satisfy LTL specifications with sufficient probability. Our specific contributions are as follows:

- We propose a control policy synthesis algorithm which operates on local regions of the state space for computational efficiency. We then combine these local control policies such that we obtain global guarantees on a robot satisfying LTL task specifications.
- We use local GPs to learn state-dependent motion perturbations. As opposed to using a single GP to learn global motion perturbations, local GPs allow for exploitation of environments with local regions of varying uncertainty in the environment.
- We evaluate the effectiveness of our framework on simulation examples and compare to baseline approaches.

The structure of the remainder of the paper is as follows. In Section II, we introduce the tools used in our approach and briefly summarize our previous work on global control policy synthesis for IMDPs which serves as a baseline of comparison for this work. We then define our problem setup in Section III. In Section IV, we explain the local control policy synthesis algorithm performed on individual regions of the state space, and in Section V we detail a methodology to combine these local control policies and prove that we obtain global system behavior guarantees. Finally, in Section

This work was supported in part by the National Science Foundation under grant #1924978 and by the National Science Foundation Graduate Research Fellowship under grant #DGE-2039655.

Jesse Jiang and Samuel Coogan are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: jjiang@gatech.edu, sam.coogan@gatech.edu). S. Coogan is also with the School of Civil and Environmental Engineering.

Ye Zhao is with the School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: ye.zhao@me.gatech.edu).

VI we show simulation results of our methodology and provide an ablation study to show its effectiveness.

II. Preliminaries

Consider a robotic system with a data-driven controller modeled using the discrete-time dynamics

$$x[k+1] = f(x[k], u[k]) + g(x[k], u[k]) + \nu[k] \qquad (1)$$

where $x \in X$ is the state of the system, $u \in U$ is the control input, f represents the known dynamics of the system, g models perturbations arising from controller imperfections, and ν is stochastic noise with stationary, symmetric, and unimodal distribution ρ_ν which is independent in each dimension, zero mean, and has bounded support. For the remainder of this paper, we will use a mobile robot case study to motivate and explain the methodology.

We assume that the state space X is bounded and partitioned into hyper-rectangular regions $\{X_q\}_{q \in Q}$:

$$X_q = \{x \mid \; a_q \leq x \leq b_q\} \subset X, \qquad (2)$$

where the inequality is taken elementwise for lower and upper bounds $a_q, b_q \in \mathbb{R}^n$ and Q is a finite index set of the regions.

Tasks are specified using Linear Temporal Logic (LTL).

Definition 1 (Linear Temporal Logic [17, Def. 2.1]): A *linear temporal logic (LTL)* formula ϕ over a set of observations O is recursively defined as

$$\begin{aligned}\phi =& \top \mid o \mid \neg o \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \bigcirc \phi \mid \\ & \phi_1 \mathcal{U} \phi_2 \mid \Diamond \phi \mid \Box \phi \mid \phi_1 \rightarrow \phi_2 \mid \phi_1 \leftrightarrow \phi_2\end{aligned}$$

where $o \in O$ is an observation and ϕ, ϕ_1, and ϕ_2 are LTL formulas. We define the *next* operator $\bigcirc$ as meaning that ϕ will be satisfied in the next state transition, the *until* operator $\mathcal{U}$ as meaning that the system satisfies ϕ_1 until it satisfies ϕ_2, the *eventually* operator $\Diamond \phi$ as $\top \mathcal{U} \phi$, and the *always* operator $\Box \phi$ as $\neg \Diamond \neg \phi$. Finally, we have that $\phi_1 \rightarrow \phi_2 = \neg \phi_1 \vee \phi_2$ and $\phi_1 \leftrightarrow \phi_2 = (\phi_1 \rightarrow \phi_2) \wedge (\phi_2 \rightarrow \phi_1)$.

The satisfaction of LTL formulas can be checked using deterministic Rabin automata (DRA) [17, Def. 2.7].

Definition 2 (Deterministic Rabin Automaton): A deterministic Rabin automaton is a tuple $\mathcal{R} = (S, s_0, O, \delta, F)$, where

- S is a finite set of states,
- $s_0 \subset S$ is a singleton initial state,
- O is the input alphabet, which corresponds to the set of observations from the LTL formula,
- $\delta : S \times O \rightarrow 2^S$ is a transition map which is either $\emptyset$ or a singleton for all $s \in S$ and $o \in O$, and
- $F = \{(G_1, B_1), \cdots, (G_n, B_n)\}$, where $G_i, B_i \subseteq S, i = 1, 2, \cdots, n$ is the acceptance condition.

The semantics of a Rabin automaton $\mathcal{R}$ are defined over infinite input words in O^ω (the set of infinite sequences of observations). A run of $\mathcal{R}$ over an infinite word $w_O = w_O(1), w_O(2), w_O(3) \cdots \in O^\omega$ is a sequence $w_S(1) w_S(2) w_S(3) \cdots \in S^\omega$, where $w_S(1) = s_0$ and $w_S(k+1) = \delta(w_S(k), w_O(k))$ for all $k \geq 1$. A run w_S admits a set $\inf(w_S) = \{w_S(i) : \forall m \in \mathbb{N} \; \exists k > m \text{ s.t. } w_S(k) = w_S(i)\}$, defined as the set of observations in w_S which appear infinitely often. Then, a run w_S is *accepted* by $\mathcal{R}$ if $\inf(w_S) \cap G_i \neq \emptyset \wedge \inf(w_S) \cap B_i = \emptyset$ for some $i \in \{1, \cdots, n\}$. If w_S is accepted by $\mathcal{R}$, we say that $w_S \models \mathcal{R}$.

We use Interval Markov Decision Processes to abstract the system dynamics.

Definition 3 (Interval Markov Decision Process): An *Interval Markov Decision Process (IMDP)* is a tuple $\mathcal{I} = (Q, A, \check{T}, \hat{T}, Q_0, O, L)$ where:

- Q is a finite set of states,
- A is a finite set of actions,
- $\check{T}, \hat{T} : Q \times A \times Q \rightarrow [0, 1]$ are lower and upper bounds, respectively, on the transition probability from state $q \in Q$ to state $q' \in Q$ under action $\alpha \in A$,
- $Q_0 \subseteq Q$ is a set of initial states,
- O is a finite set of atomic propositions or observations,
- $L : Q \rightarrow O$ is a labeling function.

For our setting, the set of IMDP states Q corresponds to the set of hyper-rectangular regions $\{X_q\}$ of the state space such that IMDP state q abstracts hyper-rectangular region X_q. Then, the set of actions A also corresponds to the hyper-rectangular regions of the state space, *i.e.*, the action α_q targets the center of region X_q. However, due to system uncertainty, the robot may be perturbed from its targeted point. We use methods proposed in [18], [19] to learn the motion uncertainties using Gaussian processes (GP) [8] and map learned bounds of the motion perturbations onto the transition probability intervals of an IMDP.

For control policy synthesis, we combine an IMDP with the DRA of the LTL specification to form a product IMDP.

Definition 4 (PIMDP): Let $\mathcal{I} = (Q, A, \check{T}, \hat{T}, Q_0, O, L)$ be a IMDP and $\mathcal{A} = (S, s_0, O, \delta, F)$ be an DRA. The product IMDP (PIMDP) is defined as a tuple $\mathcal{P} = \mathcal{I} \otimes \mathcal{A} = (Q \times S, A, \check{T}', \hat{T}', Q \times s_0, F')$, where

- $\check{T}' : (q, s) \times A \times (q', s') := \check{T}(q, \alpha, q')$ if $s' \in \delta(s, L(q))$ and 0 otherwise (where $\alpha \in A$ is an IMDP action),
- $\hat{T}' : (q, s) \times A \times (q', s') := \hat{T}(q, \alpha, q')$ if $s' \in \delta(s, L(q))$ and 0 otherwise,
- $(q_0, \delta(s_0, L(q_0))) \in (Q_0 \times S)$ is a set of initial states of $\mathcal{I} \otimes \mathcal{A}$, and
- $F' = Q \times F = \{Q \times (G_1, B_1), \cdots, Q \times (G_n, B_n)\}$, where $G_i, B_i \subseteq S$ is the ith acceptance condition.

We next define the control policies we consider.

Definition 5 (Control Policy): A control policy $\pi \in \Pi$ of a PIMDP is a mapping $(Q \times S)^+ \rightarrow A$, where $(Q \times S)^+$ is the set of finite sequences of states of the PIMDP.

The transition probability intervals in PIMDPs resulting from learnable uncertainties are resolved at planning time using adversaries.

Definition 6 (PIMDP Adversary): Given a PIMDP state (q, s) and action α, an adversary $\xi \in \Xi$ is an assignment of transition probabilities T'_ξ to all states (q', s') such that

$$\begin{aligned}\check{T}'((q, s), \alpha, (q', s')) &\leq T'_\xi((q, s), \alpha, (q', s')) \\ &\leq \hat{T}'((q, s), \alpha, (q', s')).\end{aligned}$$

In particular, we use a *minimizing* adversary, which realizes transition probabilities such that the probability of satisfying the specification is minimal, and a *maximizing* adversary, which maximizes the probability of satisfaction.

A. Global Control Policy Synthesis

We give a brief overview of the global control policy synthesis algorithm for IMDPs detailed in our previous works [18], [19]. We will refer to this baseline as "Global VI, Global GP" later in this work. We consider two distinct objectives. First, we want to synthesize a control policy which allows a robot to traverse its state space without violating LTL specification ϕ, thus sampling and learning system uncertainties using a global GP. Then, once the uncertainties have been learned sufficiently to make ϕ feasible, we want to synthesize a control policy to satisfy ϕ.

For the first objective, a value iteration algorithm [20] is first performed with respect to a maximizing adversary to determine the probability of satisfying ϕ from each state. States which have probability zero are considered violating. Then, a graph pruning algorithm is used to find a global control policy which avoids transitions to violating states.

For the second objective, global value iteration is performed with respect to a minimizing adversary to determine the probability of satisfying ϕ from each state. The control policy is then synthesized by selecting the action at each state which produces the maximum probability of satisfaction.

III. Problem Setup

We now explain the problem we address in the remainder of the paper. We consider a scenario in which a robot modeled by (1) has a task given as LTL specifications. However, there exist state-dependent motion perturbations which we seek to learn with GPs. Additionally, we assume that there exists *a priori* knowledge of local variation in the perturbations which should be leveraged for more efficient learning. This assumption can be relaxed by learning the local uncertainty regions online through hyperparameter tuning of local GPs, although that is not explored in this work. Figure 1(a) illustrates an example of the types of environments we consider.

Our ultimate goal is to synthesize a high-level control policy which allows the robot to satisfy the LTL specifications while learning and accounting for motion perturbations.

Problem 1: Synthesize a computationally efficient control policy for the system (1) to satisfy LTL specifications by learning and reducing system uncertainties.

To improve the speed of the computations and incorporate knowledge of regions with varying perturbation characteristics, we want an algorithm which can perform calculations on sub-regions of the state space using local data:

Subproblem 1.1: Develop an algorithm which reduces computation time by performing calculations on sub-regions of the state space using localized data.

Then, we need to combine these local calculations in a manner which gives global guarantees on LTL task satisfaction:

Subproblem 1.2: Use the local calculations generated by the algorithm solving Subproblem 1.1 to synthesize a control policy with global guarantees on LTL task satisfaction.

Solving Subproblems 1.1 and 1.2 solves Problem 1.

IV. Local-Global PIMDPs

In this section, we detail the local-global PIMDP structure which we use to perform control policy synthesis.

A. Partition Structure

We first define the union and intersection of PIMDPs.

Definition 7 (Union of PIMDPs): Let $\mathbf{P} = \{\mathcal{P}_0, ..., \mathcal{P}_n\}$ be a set of PIMDPs, where each PIMDP $\mathcal{P}_i$ has a set of states $Q_i \times S_i$ and a set of actions A_i. Then, the union $\bigcup \mathbf{P}$ is a PIMDP with tuple $(Q \times S, A, \check{T}', \hat{T}', Q \times s_0, F')$, where

- $Q \times S = \{(q, s) | (q, s) \in Q_i \times S_i\}$ for some $i \in 1, \cdots, n$,
- $A = \{\alpha | \alpha \in A_i\}$ for some $i \in 1, \cdots, n$,
- $\check{T}', \hat{T}', Q \times s_0, F'$ follow from Definition 4 given $Q \times S$ and A.

Thus, the union of PIMDPs can be thought of as a PIMDP which contains the union of the states and actions of the individual PIMDPs. Similarly, we can define the intersection of PIMDPs as a PIMDP which contains the intersections of the states and actions of the individual PIMDPs.

Definition 8 (Intersection of PIMDPs): Let $\mathbf{P} = \{\mathcal{P}_0, \cdots, \mathcal{P}_n\}$ be a set of PIMDPs, where each PIMDP $\mathcal{P}_i$ has a set of states $Q_i \times S_i$ and a set of actions A_i. Then, the intersection $\bigcap \mathbf{P}$ is a PIMDP with tuple $(Q \times S, A, \check{T}', \hat{T}', Q \times s_0, F')$, where

- $Q \times S = \{(q, s) | (q, s) \in Q_i \times S_i\}$ for all $i \in 1, \cdots, n$,
- $A = \{\alpha | \alpha \in A_i\}$ for all $i \in 1, \cdots, n$,
- $\check{T}', \hat{T}', Q \times s_0, F'$ follow from Definition 4 given $Q \times S$ and A.

Finally, given PIMDPs $\mathcal{P}^*$ and $\mathcal{P}'$ which are subsets of a global PIMDP $\mathcal{P}_{global} = ((Q \times S)_g, A_g, \check{T}'_g, \hat{T}'_g, (Q \times s_0)_g, F'_g)$, we define $\mathcal{P}^*$ and $\mathcal{P}'$ to be *neighbors* if there exists a state in $\mathcal{P}^*$ which has a corresponding action in $\mathcal{P}_{global}$ with nonzero upper bound probability of transitioning to a state in $\mathcal{P}'$ and vice versa.

We define a Local-Global extension of $\mathcal{P}_{global}$ as follows.

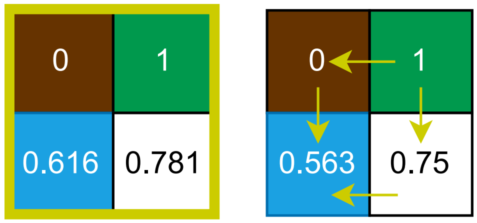
Definition 9 (Local-Global PIMDP): A Local-Global PIMDP (LG-PIMDP) extension of a PIMDP $\mathcal{P}_{global}$ is a tuple $\mathbb{P} = (\mathbf{P}_{sub}, \mathcal{N})$, where

- $\mathbf{P}_{sub}$ is a set of PIMDPs such that $\bigcup \mathbf{P}_{sub} = \mathcal{P}_{global}$ and $\bigcap \{\mathcal{P}', \mathcal{P}^*\} = \emptyset \ \forall \mathcal{P}', \mathcal{P}^* \in \mathbf{P}_{sub}$, $\mathcal{P}' \neq \mathcal{P}^*$,
- $\mathcal{N} \subseteq \mathbf{P}_{sub} \times \mathbf{P}_{sub}$ is a set of neighbor PIMDPs such that $(\mathcal{P}', \mathcal{P}^*) \in \mathcal{N} \implies (\mathcal{P}^*, \mathcal{P}') \in \mathcal{N}$, *i.e.* the edges are undirected.

Figure 1(b) shows an example of a LG-PIMDP.

B. Local Control Policy Synthesis

Given a LG-PIMDP $\mathbb{P}$, we now detail a local control policy synthesis methodology for PIMDPs in its set of regions $\mathbf{P}_{sub}$. Following the exposition in [18], [19], we consider two control policy objectives. Initially, the given LTL specification ϕ may be infeasible due to the high level of motion uncertainties throughout the environment. Thus,



(a) Global VI (b) Local VI

Fig. 2. Toy example comparing local and global control policy synthesis strategies. The blue state is the start, the brown state is a hazard, and the green state is the goal. Numbers on states indicate the final calculated satisfaction probability. **(a)**: The global value iteration (VI) evaluates the satisfaction probabilities of all states simultaneously through VI with a 4x4 matrix of transition probabilities. **(b)**: The local VI evaluates the satisfaction of the states sequentially beginning from the goal state and proceeding recursively through the neighbor states.

the robot must first traverse the environment using a control policy π_{samp} without violating ϕ. Once the robot has learned its motion uncertainties sufficient to make ϕ feasible, a satisfying control policy π_{sat} must be synthesized.

For each local region $\mathcal{P}_i \in \mathbf{P}_{sub}$ with tuple $((Q \times S)_i, A_i, \check{T}'_i, \hat{T}'_i, (Q \times s_0)_i, F'_i)$, we synthesize a local safe sampling control policy $\pi_{samp,i}$, *i.e.*, one which avoids states which are guaranteed to violate ϕ, as follows. We first define an augmented PIMDP $\mathcal{P}'_i$ which has states

$$\begin{aligned}(Q \times S)'_i = &\{(q,s)|(q,s) \in (Q \times S)_i\} \bigcup \\ &\{(q',s')|\check{T}'_g((q^*,s^*),\alpha,(q',s')) > 0\} \\ &\text{for some } (q^*,s^*) \in (Q \times S)_i, \alpha \in A_i.\end{aligned} \tag{3}$$

Intuitively, $\mathcal{P}'_i$ augments $\mathcal{P}_i$ with states it can potentially transition to in one step given the set of corresponding actions in $\mathcal{P}_{global}$. The external states are set to self-transition with probability 1, *i.e.*, they are sink states.

Then, the sampling policy $\pi_{samp,i}$ can be obtained by first running value iteration with a maximizing adversary on $\mathcal{P}'_i$ and identifying violating states $(q,s)_{viol}$ with satisfaction probability $P^{\phi}_{sat} = 0$. Next, we iteratively prune any state-action pairs $((q,s),\alpha) \in ((Q \times S)_i, A_i))$ with $\hat{T}'((q,s),\alpha,(q,s)_{viol}) > 0$ until none remain. The remaining actions at each state $(q,s) \in (Q \times S)_i$ are safe and the union of all such actions is $\pi_{samp,i}$. The maximizing adversary allows the robot to target states which may have $P^{\phi}_{sat} = 0$ under other adversaries, a design choice allowing the robot to maximize information gain by exploring uncertain states.

For the local control policy $\pi_{sat,i}$ in region $\mathcal{P}_i$ to satisfy the LTL specification, we can perform a similar operation, forming the augmented PIMDP $\mathcal{P}'_i$ and performing value iteration on the augmented PIMDP using a minimizing adversary. The control policy for each state $(q,s) \in (Q \times S)_i$ selects the optimal action

$$\alpha^* = \arg\max_{\alpha \in A_i} P^{\phi}_{sat}((q,s),\alpha), \tag{4}$$

where $P^{\phi}_{sat}((q,s),\alpha)$ is the probability of satisfying ϕ from state (q,s) after taking action α. Here, the minimizing adversary ensures that $P^{\phi}_{sat}((q,s),\alpha^*)$ is a lower bound on the true P^{ϕ}_{sat} at each state.

For each PIMDP, the transition probability intervals are generated according to the uncertainty from a GP as in Theorem 1, [18]. In particular, we use local sparse GPs trained on data collected within their respective PIMDP regions. This differs from the approach in [18], [19] where a single global GP was used to learn uncertainties.

Theorem 1 provides behavior guarantees on the local control policy synthesis algorithms detailed in this section, solving Subproblem 1.1.

Theorem 1 (Local Behavior Guarantees): A robot with LTL specification ϕ and executing control policy $\pi_{samp,i}$ generated by the algorithms in Section IV-B traverses only nonviolating states while in the local region $\mathcal{P}'_i$. When executing the policy $\pi_{sat,i}$, the robot maximizes its probability of satisfying ϕ from states in $\mathcal{P}'_i$.

Proof: We first examine $\pi_{samp,i}$. The algorithm first initializes the probability of satisfaction for states in $\mathcal{P}_i$ to be 0 unless the state is contained in the accepting region F'_i, in which case the probability of satisfaction is fixed at 1. Then, the algorithm identifies an augmented PIMDP $\mathcal{P}'_i$. Assume that the external states $\{(q',s')|\check{T}'_i((q^*,s^*),\alpha,(q',s')) > 0\} \setminus (Q \times S)_i$ have valid probabilities of satisfaction:

$$\max_{\alpha \in A_i} P^{\phi}_{sat}((q',s'),\alpha) \leq P^{\phi}_{opt}(q',s'), \tag{5}$$

where $P^{\phi}_{opt}(q',s')$ is the true probability of satisfaction. This assumption is addressed in Section V. Let $\hat{P}^{\phi}_{sat,k}$ be the probability of satisfaction calculated for an arbitrary state in $\mathcal{P}'_i$ at the kth step of value iteration given this assumption, and let $\hat{P}^{\phi}_{opt,k}$ correspond to the case when the external states are initialized at the optimal values

$$\max_{\alpha \in A_i} P^{\phi}_{sat}((q',s'),\alpha) = P^{\phi}_{opt}(q',s'), \tag{6}$$

Then, consider the value iteration algorithm as described in [20]. Initially (for step $k=0$) it holds trivially that $\hat{P}^{\phi}_{sat,0} \leq \hat{P}^{\phi}_{opt,0}$ for all states in $\mathcal{P}'_i$. If at the kth step $\hat{P}^{\phi}_{sat,k} \leq \hat{P}^{\phi}_{opt,k}$, it holds that $\hat{P}^{\phi}_{sat,k+1} \leq \hat{P}^{\phi}_{opt,k+1}$ at the next step since the value iteration operations are monotone with respect to the current state values. By induction, at any step of the value iteration it holds for all states $(q,s) \in \mathcal{P}'_i$ that

$$\max_{\alpha \in A_i} P^{\phi}_{sat}((q,s),\alpha) \leq P^{\phi}_{opt}(q,s), \tag{7}$$

i.e. the calculated probabilities of satisfaction are underapproximations of the true values. It follows that this algorithm identifies all the true violating states $(q,s)_{viol}$ with $P^{\phi}_{opt} = 0$. Then, by pruning actions transitioning to the violating states, the set of allowable actions $A_{q,s} \subseteq \pi_{samp,i}$ at any state $(q,s) \in \mathcal{P}'_i$ has the property

$$\hat{T}'((q,s),\alpha,(q,s)_{viol}) = 0 \ \forall \alpha \in A_{q,s} \tag{8}$$

Therefore, when executing $\pi_{samp,i}$ in $\mathcal{P}'_i$, the robot is guaranteed to remain in nonviolating states.

The proof of the guarantees for $\pi_{sat,i}$ follows similarly. Assume the external states $\{(q',s')|\check{T}'_i((q^*,s^*),\alpha,(q',s')) > 0\} \setminus (Q \times S)_i$ in the augmented PIMDP $\mathcal{P}'_i$ satisfy

$$\max_{\alpha \in A_i} P^{\phi}_{sat}((q',s'),\alpha) \leq P^{\phi}_{opt}(q',s'). \tag{9}$$

Then, at any step of value iteration with a minimizing adversary, the relation

$$\max_{\alpha \in A_i} P_{sat}^{\phi}((q,s),\alpha) \leq P_{opt}^{\phi}(q,s). \tag{10}$$

holds for all states $(q,s) \in \mathcal{P}_i'$. Therefore, the computed P_{sat}^{ϕ} for any state in $\mathcal{P}_i'$ is valid, and by construction $\pi_{sat,i}$ selects the probability-maximizing action at each state. ■

We now present a toy example to illustrate our algorithms:

Example 1 (Simple Grid World): Consider the system shown in Figure 2, which has four states: the blue start state, the brown hazard state, the green goal state, and an unlabeled state. The LTL specification ϕ is to reach the goal state while avoiding the hazard state. Each state has two actions, each of which targets one of its two border states. Each action has a lower bound and upper bound probability $[0.75, 1]$ of reaching its target state, and probability $[0, 0.2]$ of reaching each one of the other three states. In Figure 2(a), global value iteration (as in [18], [19]) is used which involves repeated matrix-vector multiplication of dimension 4. The P_{sat}^{ϕ} for each state is taken as the final computed satisfaction probability for the state when the value iteration converges. In Figure 2(b), local value iteration as proposed in this work is used with each state as its own region for control policy synthesis. For each local region, a P_{sat}^{ϕ} is computed by selecting the probability-maximizing action and taking its corresponding satisfaction probability, an operation which requires only a single matrix-vector multiplication.

V. Global Control Policy Synthesis

Given the LG-PIMDP $\mathbb{P}$ and local control policy synthesis algorithms established in Section IV, it remains to develop an algorithm to combine local control policies to provide global guarantees of task satisfaction.

We first detail a global control policy synthesis algorithm π_{samp} to safely traverse the state space while learning uncertainties. The key idea is to order the local partition calculations in such a way as to guarantee the correctness of the set of nonviolating states identified. Our algorithm is as follows. We begin with the regions $\mathcal{P}_i$ which contain accepting states of the LG-PIMDP, i.e. $F_i' \neq \emptyset$. We then form the augmented PIMDP $\pi_{samp,i}$ and perform value iteration as in Section IV, initializing the neighbor states $\{(q',s') | \check{T}_i'((q^*,s^*),\alpha,(q',s')) > 0\} \setminus (Q \times S)_i$ with probability $P_{sat}^{\phi} = 0$. Then, we identify the neighbor set

$$\{\mathcal{P}_j | (\mathcal{P}_i, \mathcal{P}_j) \in \mathcal{N}\} \tag{11}$$

and add these neighboring regions to a first-in-first-out queue $\mathbf{R}$. For each region $\mathcal{P}_j$ in the queue, we repeat the local control policy synthesis algorithm and add its neighbors which have not been processed already to the end of the queue $\mathbf{R}$. We continue this sequeunce until $\mathbf{R} = \emptyset$, at which point the entire LG-PIMDP has been processed. The final control policy is synthesized as

$$\pi_{samp} = \bigcup_{\{i | \mathcal{P}_i \in \mathbf{P}_{sub}\}} \pi_{samp,i} \tag{12}$$

Algorithm 1: Safe Control Policy Synthesis

Input: LG-PIMDP $\mathbb{P}$, Accepting PIMDP set $\mathbf{G}$
Output: Global control policy π_{samp}

Initialize *Processing Queue* $\mathbf{R}$, *Completed Set* $\mathbf{U}$;
for *PIMDP* $\mathcal{P} \in \mathbf{G}$ **do**
  Synthesize local control policy $\pi_{samp,\mathcal{P}}$ as in Section IV-B with external augmented states fixed with satisfaction probability 0;
  Add unprocessed neighbor set $\{\mathcal{P}' | (\mathcal{P}, \mathcal{P}') \in \mathcal{N}\} \setminus \mathbf{U}$ to the end of queue $\mathbf{R}$;
  Add $\mathcal{P}$ to completed set $\mathbf{U}$;
end
while $\mathbf{R} \neq \emptyset$ **do**
  Remove first PIMDP $\mathcal{P}$ from $\mathbf{R}$;
  Synthesize local control policy $\pi_{samp,\mathcal{P}}$ for $\mathcal{P}$ with current information from external augmented states;
  Add unprocessed neighbor set $\{\mathcal{P}' | (\mathcal{P}, \mathcal{P}') \in \mathcal{N}\} \setminus \mathbf{U}$ to the end of queue $\mathbf{R}$;
  Add $\mathcal{P}$ to completed set $\mathbf{U}$;
end
return $\pi_{samp} = \bigcup_{\{i | \mathcal{P}_i \in \mathbf{P}_{sub}\}} \pi_{samp,i}$

Algorithm 1 details the methodology.

The global control policy synthesis algorithm to satisfy ϕ follows a similar structure. We again begin with the accepting regions $\{\mathcal{P}_i | F_i' \neq \emptyset\}$, performing local value iteration and control policy synthesis to obtain $\pi_{sat,i}$ as in Section IV-B. We then add the neighbor set $\{\mathcal{P}_j | (\mathcal{P}_i, \mathcal{P}_j) \in \mathcal{N}\}$ to the end of the queue $\mathbf{R}$ to be processed and continue performing local control policy synthesis until $\mathbf{R} = \emptyset$, at which point we generate

$$\pi_{sat} = \bigcup_{\{i | \mathcal{P}_i \in \mathbf{P}_{sub}\}} \pi_{sat,i}. \tag{13}$$

Theorem 2 proves that the global control policy synthesis algorithms presented in this section give global guarantees on LTL task satisfiability, solving Subproblem 1.2.

Theorem 2 (Global Behavior Guarantees): Executing π_{samp} generated by Algorithm 1 ensures that the robot does not violate the LTL specification ϕ anywhere in the state space if a nonviolating policy exists. Likewise, executing π_{sat} guarantees that the robot maximizes its probability of satisfying ϕ globally.

Proof: The proof follows generally from the proof of correctness of the local control policy synthesis algorithms established in Theorem 1. We first examine the safe control policy π_{samp} generated by Algorithm 1. We initially run the local control policy synthesis algorithm on the accepting regions $\{\mathcal{P}_i | F_i' \neq \emptyset\}$. Since we assume that accepting states (q^*, s^*) are absorbing and thus

$$\max_{\alpha \in A_i} P_{sat}^{\phi}((q^*,s^*),\alpha) = 1, \tag{14}$$

it follows that

$$P_{sat}^{\phi}(q^*,s^*) \leq P_{opt}^{\phi}(q^*,s^*) \tag{15}$$

since probabilities are bounded above by 1. All other states are initialized with probability 0, so Equation (15) holds for all states in $\mathcal{P}_i$ and by Theorem 1 we know that the local value iteration in $\mathcal{P}_i$ produces valid values for all P_{sat}^{ϕ}. Once the initial regions have been processed, we proceed to perform local control policy synthesis on the neighbors $\{\mathcal{P}_j | (\mathcal{P}_i, \mathcal{P}_j) \in \mathcal{N}\}$. By definition, these neighbors transition to states in the initial set $\{\mathcal{P}_i | F_i' \neq \emptyset\}$, so local value iteration on the $\mathcal{P}_j$ satisfies the assumption (5) in Theorem 1 and the local value iterations are valid. Thus, by recursively performing local control policy synthesis on neighbor sets as in Algorithm 1, we guarantee the safety of each local control policy $\pi_{samp,i}$ by using guarantees from the previously computed regions. Since $\pi_{samp,i}$ is valid for all i, it follows that $\pi_{samp} = \bigcup_{\{i | \mathcal{P}_i \in \mathbf{P}_{sub}\}} \pi_{samp,i}$ is also valid. A similar argument holds for the validity of π_{sat}. ■

Example 2 (Simple Grid World Continued): We continue illustrating our methodology using the same setup as in Example 1. We compare the global control policy synthesis algorithms to generate π_{sat} shown in Figure 2. In Figure 2(a), global value iteration is used which takes eight iterations to converge to the values shown in the figure. In Figure 2(b), local value iteration is used which starts at the goal state and then proceeds sequentially to neighbor regions as depicted by the arrows. In this case, only four total computations are needed, two of which are trivial (the goal and hazard states). The final P_{sat}^{ϕ} values in the local VI case are more conservative than in the global VI case, illustrating a tradeoff between computational complexity and accuracy.

A. Complete Control Policy Synthesis Framework

We can now present the complete methodology to solve Problem 1. We start by constructing a LG-PIMDP $\mathbb{P}$ of the system (1), environment, and a given LTL specification ϕ. The true motion perturbation is initially unknown but is estimated with local GPs based on conservative estimates known *a priori*. The size of the regions corresponding to local GPs is selected to balance computational tractability and task feasibility. Smaller local regions are more computationally efficient, but result in more conservative satisfaction probabilities as compared to larger local regions.

Next, we perform global control policy synthesis on $\mathbb{P}$ to maximize the probability of satisfying ϕ. If the probability from the initial state is below a desired threshold P_{des}^{ϕ}, we then synthesize a safe sampling control policy π_{samp} using Algorithm 1. We execute this control policy (randomly selecting actions if multiple are valid at a given state) for a predetermined number of steps, allowing the robot to traverse through the environment and collect actual motion perturbation data. We then update the GPs corresponding to traversed regions with the newly collected data and recalculate P_{sat}^{ϕ} globally. We continue sampling the state space and retraining GPs until $\max_{\alpha \in A} P_{sat}^{\phi}(q, s, \alpha) > P_{des}^{\phi}$ at the current state (q, s), or a maximum number of iterations has been reached. This bound on the iteration count ensures termination in the case that a satisfying control policy cannot be synthesized for the given scenario. If ϕ can be satisfied, we execute the control policy π_{sat} until the robot reaches an accepting state.

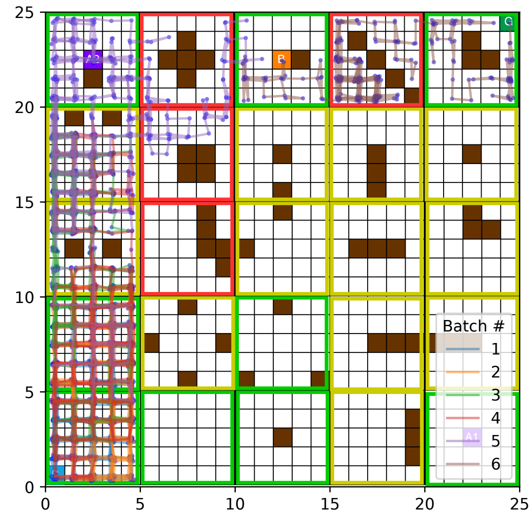


Fig. 3. Sample trajectory for one run of the case study simulation. Green, yellow, and red regions correspond to areas with low, medium, and high uncertainty, respectively. The initial state is in blue in the bottom left, the target state is green in the top right, and the hazard regions are brown. There are intermediate "A1" and "A2" states in purple, "A2" in the top left and "A1" in the bottom right, and the "B" state is orange in the top middle. The robot initially samples close to its start region until it learns the motion perturbations sufficiently well to traverse to the top left area. Once there, the robot traverses across to reach the "A2", "B", and goal states.

VI. Case Study

Consider a mobile robot in a state space depicted in Figure 3 with bounds $x \in X := [0, 25]^2 \subset \mathbb{R}^2$. At each discrete state, the robot has actions corresponding to moving {up, down, left, right} one state. We assume that terrain uncertainty in the environment creates motion perturbations, and we choose to directly learn the state-dependent motion perturbations with GPs. For implementation purposes, the true motion perturbation for regions with low, medium, and high uncertainty (green, yellow, and red regions in Figure 3, respectively) is created by sampling pregenerated local GPs with variance $0.01, 0.03$, and 0.05, and length hyperparameter $7, 5$, and 3, respectively. Intuitively, regions with higher variance and lower length hyperparameter have greater variation in perturbation values and thus require more sampling to characterize. Additionally, there exists stochastic motion perturbation ν sampled from a truncated Gaussian distribution $N(0, 0.01)$ bounded at $\pm 2\sigma$. The objective of the robot is to reach the goal state "G" while avoiding hazard "H" states. Additionally, before it reaches the goal, it must traverse either the "A1" or "A2" states and then the "B" state. This gives the LTL specification

$$\phi = \neg \mathtt{H} \; \mathcal{U} \; \mathtt{G} \; \wedge \; \neg \mathtt{G} \; \mathcal{U} \; \mathtt{B} \; \wedge \; \neg \mathtt{B} \; \mathcal{U} \; (\mathtt{A1} \vee \mathtt{A2}) \tag{16}$$

Given these parameters, under our proposed methodology the robot initially has a probability of 0.19 to satisfy the specification. The robot then executes the algorithm, collecting batches of trajectory data and retraining its GP estimations of the motion perturbation in between each batch before continuing from its current position. In earlier batches, the

TABLE I
ABLATION STUDY OF THE PROPOSED ALGORITHM.

Algorithm	Total Time (sec)	# Steps
Our Method (Local VI, Local GP)	**354**	**2759**
Global VI, Local GP	1284	9499
Local VI, Global GP	Infeasible	Infeasible
Baseline (Global VI, Global GP)	Infeasible	Infeasible

robot remains within the low perturbation areas close to its starting position, learning the local dynamics until it moves towards the "A2" checkpoint in the top left of the state space. Once the probability of satisfaction crosses the desired threshold of 0.99, the robot takes the riskier but shorter direct path across the top of the state space towards the "B" checkpoint and finally the goal region. The robot is able to follow this trajectory owing to its efficient learning of motion perturbations in the top left area of the state space, allowing it to safely pass through the high-uncertainty regions separating the "A2" and "B" checkpoints. The entire algorithm takes a total of 5 minutes 54 seconds to run on a machine with a Ryzen 7700X CPU, 32 GB of RAM, and a RTX 3090 GPU. Figure 3 shows the trajectory of the run discussed.

A. Ablation Study

We now perform an ablation study on the same scenario to characterize the benefits of each component of the proposed algorithm. We compare performance against a baseline algorithm proposed in our previous works [18], [19] which uses global value iteration (Global VI) and global GP learning (Global GP), as well as against modified versions of the algorithm in this work which remove one component each (Global VI/Local GP, Local VI/Global GP). Table I summarizes our findings, comparing total computation time and the number of steps the robot required to satisfy the LTL specification. The baseline case and the "Local VI, Global GP" case are both infeasible due to the high levels of uncertainty created by using global parameters for the GPs, resulting in no valid safe sampling trajectory being found. The "Global VI, Local GP" case finds a successful trajectory, but requires four times the amount of time and number of steps. In this case, the initial probability of satisfaction is 0.98, but the robot is less efficient in exploring towards the goal as the robot tends to explore the space more uniformly rather than moving along a trajectory of local regions towards the goal as in the "Local VI, Local GP" case.

VII. CONCLUSION

In this work, we proposed a novel local-global method to enable computationally efficient IMDP control policy synthesis for robotic systems with system and environmental uncertainty. We developed a methodology which performs local control policy synthesis on sub-regions of the global state space and then combines local policies to obtain global guarantees on the probability of satisfying a given LTL specification. Additionally, we proposed the use of local GPs to learn uncertainties. Finally, we demonstrated the effectiveness of our proposed algorithms compared to the baseline global control policy synthesis algorithms developed in previous works. Future work will implement these local-global algorithms on hardware in order to demonstrate the feasibility of online IMDP-based control policy synthesis.

REFERENCES

[1] K. Sreenath, C. R. Hill, and V. Kumar, "A partially observable hybrid system model for bipedal locomotion for adapting to terrain variations," in *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 137–142.

[2] K. Byl and R. Tedrake, "Metastable walking machines," *I. J. Robotic Res.*, vol. 28, pp. 1040–1064, 07 2009.

[3] B. Lacerda, D. Parker, and N. Hawes, "Optimal and dynamic planning for markov decision processes with co-safe ltl specifications," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1511–1516.

[4] M. Cai, H. Peng, Z. Li, and Z. Kan, "Learning-Based Probabilistic LTL Motion Planning With Environment and Motion Uncertainties," *IEEE Transactions on Automatic Control*, vol. 66, no. 5, pp. 2386–2392, 2021.

[5] M. Dutreix, J. Huh, and S. Coogan, "Abstraction-based synthesis for stochastic systems with omega-regular objectives," *Nonlinear Analysis: Hybrid Systems*, vol. 45, p. 101204, 2022.

[6] R. Majumdar, K. Mallik, A.-K. Schmuck, and S. Soudjani, "Symbolic qualitative control for stochastic systems via finite parity games." *IFAC-PapersOnLine*, vol. 54, no. 5, pp. 127–132, 2021, 7th IFAC Conference on Analysis and Design of Hybrid Systems ADHS 2021.

[7] A. Lavaei, S. Soudjani, A. Abate, and M. Zamani, "Automated verification and synthesis of stochastic hybrid systems: A survey," *Automatica*, vol. 146, p. 110617, 2022.

[8] C. K. I. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in neural information processing systems 8*. MIT press, 1996, pp. 514–520.

[9] J. Jackson, L. Laurenti, E. Frew, and M. Lahijanian, "Strategy Synthesis for Partially-known Switched Stochastic Systems," *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, May 2021.

[10] A. H. Chang, C. M. Hubicki, J. J. Aguilar, D. I. Goldman, A. D. Ames, and P. A. Vela, "Learning terrain dynamics: A gaussian process modeling and optimal control adaptation framework applied to robotic jumping," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 4, pp. 1581–1596, 2021.

[11] S. Vasudevan, F. Ramos, E. Nettleton, H. Durrant-Whyte, and A. Blair, "Gaussian process modeling of large scale terrain," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 1047–1053.

[12] M. L. Littman, T. L. Dean, and L. P. Kaelbling, "On the complexity of solving markov decision problems," in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, ser. UAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, p. 394–402.

[13] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic mdp-behavior planning for cars," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011, pp. 1537–1542.

[14] D. Wingate, *Solving large mdps quickly with partitioned value iteration*. Brigham Young University, 2004.

[15] M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. L. Dean, and C. Boutilier, "Hierarchical solution of markov decision processes using macro-actions," *arXiv preprint arXiv:1301.7381*, 2013.

[16] J. Barry, L. P. Kaelbling, and T. Lozano-Pérez, "Hierarchical solution of large markov decision processes," 2010.

[17] C. Belta, B. Yordanov, and E. GÖL, *Formal Methods for Discrete-Time Dynamical Systems*, ser. Studies in Systems, Decision and Control. Springer International Publishing, 2017.

[18] J. Jiang, Y. Zhao, and S. Coogan, "Safe learning for uncertainty-aware planning via interval MDP abstraction," *IEEE Control Systems Letters*, vol. 6, pp. 2641–2646, 2022.

[19] J. Jiang, S. Coogan, and Y. Zhao, "Abstraction-based planning for uncertainty-aware legged navigation," *IEEE Open Journal of Control Systems*, vol. 2, pp. 221–234, 2023.

[20] M. Lahijanian, S. B. Andersson, and C. Belta, "Formal Verification and Synthesis for Discrete-Time Stochastic Systems," *IEEE Transactions on Automatic Control*, vol. 60, no. 8, pp. 2031–2045, Aug. 2015.