Hybrid Risk-Aware Conditional Planning with Applications in Autonomous Vehicles

Hybrid Risk-Aware Conditional Planning with Applications in Autonomous Vehicles

Xin Huang*, Ashkan Jasour*, Matthew Deyo, Andreas Hofmann and Brian C. Williams

***Abstract*— In this paper, we address the problem of risk-aware conditional planning where the goal is generating risk bounded motion policies in the presence of uncertainty. The problem is modeled as a chance-constrained Partially Observable Markov Decision Process (CC-POMDP) with one controllable agent and multiple uncontrollable agents, each of which can choose from a set of maneuver actions. The risk is defined as the probability of the controllable agent violating safety constraints. Off-line computations include generating a library of probabilistic maneuvers for the controllable agent and planning an initial motion policy to execute. During runtime, the conditional planner can quickly look up maneuver sequences to ensure risk bounds as the world around our agent evolves. We introduce the iterative RAO* heuristic search algorithm, which iteratively generates risk bounded conditional plans over a finite horizon. We demonstrate the performance of the provided approach on two planning problems of autonomous vehicles.**

I. Introduction

An important challenge in enabling safe motion planning for intelligent robotic systems is to assess risks under uncertainties and account for them in planning. This challenge is seen in the development of autonomous vehicles, which need to deal with uncertainties in road surface conditions, pedestrian behavior, and other drivers on the road. Consider, for example, the scenario where a vehicle enters a busy intersection with the intention of turning right (see Fig. 1), while another vehicle is turning ahead and pedestrians are crossing. The autonomous system must be aware of these other agents and the changing environment, be able to predict possible outcomes, and consider the risks before making decisions.

The Markov Decision Process (MDP) is one of the most widely used frameworks for decision making in situations with uncertainties. A number of extensions to MDP have been proposed to achieve the goal of generating optimal plans while bounding risks under uncertainties. For example, [1] and [2] modeled the risk in a constrained POMDP (CPOMDP) framework by assigning unit costs to states considered too risky. Another extension is proposed in [3], where a more flexible definition of risk allocation is used in a chance-constrained POMDP (CC-POMDP) framework. The framework has been studied in various problem domains

Fig. 1: A motivating vehicle driving scenario.

[4], [5], [6], [7] and proved to be useful when dealing with uncertainties. In this paper, we focus on this framework, which allows us to bound the probability of violating chance constraints across a conditional plan.

Motion planning for autonomous vehicles requires a continuous representation for maneuver actions based on the dynamical system of the vehicle. Our approach takes advantage of a probabilistic representation for vehicle maneuvers, which allows us to model the vehicle trajectories accurately and determine the chance constraints such as collisions efficiently.

Another challenge of our work is to model the behaviors of multiple agents in the planning procedure. Many investigations [8], [9] have focused on coordination between agents to achieve the maximum joint utility. However, in real world applications such as autonomous vehicles, we only have control over our own vehicle and there is no communication between vehicles. In addition, a multi-POMDP model [10] is used to represent each vehicle's behavior as individual decision components, where each uncontrollable component only produces one deterministic action. Such an approach is not robust if the vehicle can take multiple actions with similar likelihoods. In order to address these limitations, we develop a multi-agent model that takes all possible actions of uncontrollable agents into consideration. Using our approach, each agent can be modeled independently according to their specific configurations and driving styles.

Our contributions are four-fold. First, we propose a systematic definition of planning with multiple agents in CC-POMDP domains. We also develop a novel representation for continuous actions. Our third contribution is an online risk-aware planner that takes a receding horizon approach to conditional planning and makes iterative updates to the

This work was partially supported by the Toyota Research Institute (TRI). However, this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

*These authors contributed equally to the paper.

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar St, Cambridge, MA 02139, USA {huangxin,jasour,mdeyo,williams}@mit.edu, hofma@csail.mit.edu

policy during execution. Lastly, we verify our algorithms in a Unity simulator that accurately models the vehicle dynamics as well as the road environment.

II. Related Work

A. POMDP Planning

Several approaches exist for solving POMDP planning problems. For instance, [11] partitions the action space and solves for a sequence of POMDP policies using a hierarchical algorithm. Point-based methods [12], [13] are proven successful in scaling to large problems by approximating the value function. Recently, an anytime approach [14], [15] is proposed for online POMDP problems by searching for approximately optimal policy while maintaining a balance between the policy size and the value estimation accuracy. Despite the success of these algorithms, they fail to deal with uncertainties and risk constraints in many real world problems such as autonomous driving.

B. Risk-bounded AO*

Chance-constrained POMDP can be solved using approaches such as linear programming and value iteration [16], [17], [1], [2]. In this paper, we focus on augmenting a conditional planner called Risk-bounded AO* (RAO*) [3] that finds optimal policies with maximum expected reward over a finite horizon while satisfying all chance constraints. Similar to the AO* algorithm [18], RAO* utilizes a value heuristic to guide the search towards optimal policies. It also uses a risk heuristic to prune the search space, removing high-risk branches that violate the chance constraints. RAO* has proven useful in challenging domains like power supply restoration and autonomous science agents. In this paper, we are identifying and addressing four limitations with the existing RAO* planner.

As an offline conditional planner, RAO* is limited in its ability to handle the size of long planning horizons. Conditional planning and probabilistic domains are both known for their search space complexity. Pruning the search space using risk is domain dependent and less effective in safer planning scenarios. The tractability of the search space quickly becomes an issue when trying to reason through driving scenarios where nearby vehicles each have complex driver models to consider.

Secondly, using an offline planner requires more a priori knowledge of the world and confidence in the models used. This limitation is quickly encountered when trying to model driving scenarios in probabilistic domains. In order to generate a finite horizon policy offline, we need to make assumptions about driver models for many different vehicles, which could prove wrong after additional observations are made.

Thirdly, the way that execution risk is calculated can allow low-probability high-risk actions to be included in our policy, essentially discounting future risk in search spaces with high branching factors. This side effect can be illustrated with the simple example shown in Figure 2. A robot must navigate to the goal, two cells to the right, while avoiding the fire in the lower cell. Actions available to the robot are to move right, climb up a row, or descend down a row. Unfortunately, icy conditions in the two blue cells make moving right have nondeterministic results shown by the red arrows, where there is some probability of transitioning to a cell above or below instead of to the right.

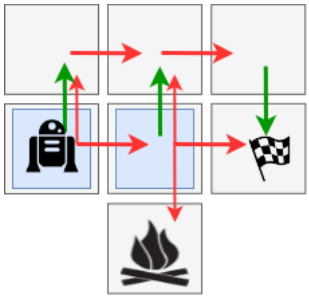


Fig. 2: Execution risk with slippery navigation.

Using RAO* to solve this problem, there are transition probabilities and chance constraints that result in a policy of moving right twice for the most likely outcomes. For example, if the relevant transition probabilities are $Pr(center|start, moveRight)$ = $Pr(goal|center, moveRight)$ = 0.8 and $Pr(fire|center, moveRight)$ = 0.1, and the chance constraint on reaching the fire state is 0.09, then RAO* will result in a policy to move right twice (regardless of whether we have uniform action costs or make climbing more costly). While offline, the execution risk of reaching the fire state is less than the chance constraint, sitting at 0.08. However, after executing the first action of *moveRight*, the most likely next state is *center*, where the offline policy wants us to *moveRight* again though execution risk is now 0.1. This disconnect between offline execution risk calculations and real risks during execution only gets worse as the branching and depth of the policy increases, sometimes resulting in a 100% probability of violating a constraint when the system moves down the most likely path. While those edge cases might only occur with the low probability associated with their offline execution risk, it requires us to depend entirely on our a priori probabilistic models being accurate.

Finally, the original RAO* algorithm solves for problems with discrete domains, thus an extension was necessary for our objective of planning for autonomous vehicles with continuous states and actions.

C. Continuous Maneuver Modeling

It is important to use a continuous representation for maneuver actions of autonomous vehicles. One way to model continuous maneuver actions is with Probabilistic Flow Tubes (PFT) [19], [20], which represent a set of continuous trajectories with common characteristics defined over a time interval $[t_0, t_g]$. It is characterized as a set of cross-sectional regions, where each cross section s_i stores the mean and covariance of the associated common trajectories at time

t_i $(0 \leq i \leq g)$. PFTs are generated from demonstrated trajectories to model realistic human motions. Although PFT allows us to model maneuver actions probabilistically, the method to generate PFTs in [19] only considers the spatial states of the human-generated trajectories and ignores the underlying control inputs, which makes it hard to replicate the motions. We address this limitation by designing a set of PFTs that capture the system dynamics and a set of controllers for the vehicle to follow the desired motions.

Another widely-used approach is with funnels (or regions of finite-time invariance) in [21], but they are not applicable to calculate the collision probability between vehicles because they do not model probabilistic variances of the motions.

III. Problem Formulation

In this work, we consider a planning problem in a multi-agent modeling framework with a controllable agent $\mathcal{R}_0$ and a set of n uncontrollable agents $\mathcal{R}_1, \ldots, \mathcal{R}_n$ with no communication between each other. Each agent can choose from a library of actions modeled with preconditions and effects, as defined in Definition 1.

Definition 1. (Action Model) An action model is a tuple $A = \langle \textit{Pre}, \textit{Eff}, c, \textit{PFT} \rangle$, where

- *Pre*: is a set of preconditions.
- *Eff* is a set of effects.
- c is the cost associated with the action.
- *PFT* is a probabilistic model representing the continuous evolution of the action with uncertainties (with more details in Section IV-A).

The action model is combined with belief states to represent each agent. The preconditions and effects are dependent on factors internal and external to each agent, allowing us to model different driver types and behaviors. The agent model is defined as:

Definition 2. (Agent Model) An agent model is a tuple $\mathcal{M} = \langle x, x_D, \mathbf{D}, \mathbf{A}, b_0, \textit{pmf} \rangle$, where

- x is a set of continuous state variables.
- x_D is a set of discrete state variables.
- $\mathbf{D}$ is the domain for x_D.
- $\mathbf{A}$ is a set of *action models* that the agent can choose from.
- b_0 is an initial belief state over the state variables.
- *pmf* is a probability mass function calculating the likelihood that uncontrollable agents will execute an action in their library given external and internal factors.

Once we have defined all agent models, an extended version of a POMDP is used to define our problem of planning with multiple agents, called a Chance-Constrained Partially Observable Markov Decision Process with Multiple Agents (*CC-POMDP-MA*):

Definition 3. (CC-POMDP-MA) A CC-POMDP-MA is a tuple $\langle \mathcal{M}, \mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, C, \mathcal{B}_0, h, \mathcal{C}, \Delta \rangle$, inspired by CC-POMDP [3]:

- $\mathcal{M} = [\mathcal{M}_0, \ldots, \mathcal{M}_n]$ is a set of $n+1$ agent models defined in Definition 2.
- $\mathcal{S}$ is a set of composite-states including the states of all agents from $\mathcal{M}$.
- $\mathcal{A}$ is the set of actions for the controllable agent model derived from $\mathcal{M}_0$.
- $\mathcal{O}$ is a set of observations.
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a state transition function between composite-states according to $\mathcal{M}$.
- $O : \mathcal{S} \times \mathcal{O} \to \mathbb{R}$ is an observation function.
- $C : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a cost function.
- $\mathcal{B}_0$ is the initial belief state over composite-states.
- h is the finite execution horizon.
- $\mathcal{C} = [\mathcal{C}^1, \ldots, \mathcal{C}^q]$ is a set of q constraints over composite states $\mathcal{S}$.
- $\Delta = [\Delta^1, \ldots, \Delta^q]$ is a vector of probabilities for q constraints that bound the chance constraints.

We define the execution risk of a policy π measured from a belief state $\mathcal{B}_k$ to horizon h as:

$$er(\mathcal{B}_k, \mathcal{C} | \pi) = 1 - Pr(\bigwedge_{i=k}^{h} \text{Safe}_i(\mathcal{C}) | \mathcal{B}_k, \pi), \tag{1}$$

where Safe_k is a Bernoulli random variable indicating whether the system has not violated constraint $\mathcal{C}$ at time k. So the chance constraints for the entire system starting with an initial belief state are defined as follows.

$$er(\mathcal{B}_0, \mathcal{C}^i | \pi) \leq \Delta^i, \mathcal{C}^i \in 2^{|\mathcal{C}|}, i = 1, 2, \ldots, q. \tag{2}$$

The goal is to find an optimal, deterministic, and chance-constrained policy π^* given a CC-POMDP-MA tuple, such that

$$\pi^* = \arg\min_{\pi} \mathbb{E}\left[\sum_{t=0}^{h} C(s_t, a_t) \Big| \pi \right]. \tag{3}$$

Using the three definitions within a multi-agent framework, we successfully represent a motion planning problem, where the autonomous controllable vehicle drives in an environment with agents that have uncertain motions and partially observable states such as positions and velocities.

IV. Approach

A. Probabilistic Flow Tube with Feedback Control

For each action model, we need to represent the continuous dynamics of the maneuver action with uncertainties. In the following section, we introduce an extension to probabilistic flow tubes (PFT) [19] such that it is able to incorporate vehicle dynamics.

Each action is modeled with a discrete-time continuous-state time varying dynamical system modeling the continuous changes of the action:

$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k^w \mathbf{w}_k + B_k \mathbf{u}_k \tag{4}$$

where $\mathbf{x} \in \mathbf{R}^{n_x}$ is the continuous state, $\mathbf{u} \in \mathbf{R}^{n_u}$ is the continuous control input, and $\mathbf{w} \in \mathbf{R}^{n_w}$ is the uncertainty of the controllable agent at discretized time that models disturbances, uncertain system model parameters, and sensor

noises. We assume that $\mathbf{w}$ is a Gaussian process $\mathcal{N}(\mathbf{m}_k^w, \mathbf{\Sigma}_k^w)$ and the initial state $\mathbf{x}_0$ is a Gaussian random variable $\mathcal{N}(\mathbf{m}_0, \mathbf{\Sigma}_0)$.

Library of Maneuver Trajectories $\mathcal{L}_T$: We start by designing a library of maneuvers for the vehicle agent. This library consists of open loop trajectories and associated open loop control inputs that the agent could choose during runtime. Fig. 3 shows the library of maneuvers used for the highway driving problem in Sec. V. This library was generated with spline polynomials and includes maneuvers for merging left, right, and moving forward with different velocities. Although the demonstrated maneuvers are tailored to highway scenario, our approach can design and choose arbitrary maneuvers depending on the problem domain and the driving environment.

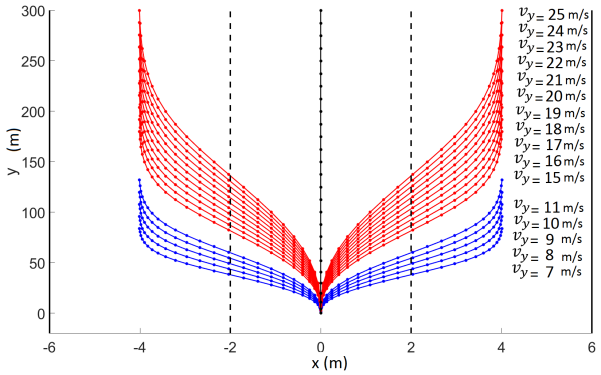


Fig. 3: Library of maneuvers for vehicle.

Library of PFTs $\mathcal{L}_{PFT}$: For a given dynamical system in Eq. (4) and control input $\mathbf{u}_k$, PFTs are defined as probabilistic predictions on the states over the finite horizon. Due to the Gaussian uncertainties $\mathbf{x}_0$ and $\mathbf{w}$, the state of the dynamical system at each time k is Gaussian random variable $\mathbf{x}_k \sim \mathcal{N}(\mathbf{m}_k, \mathbf{\Sigma}_k)$. The evolution of the mean and covariance of states is governed by:

$$\mathbf{m}_{k+1} = A_k \mathbf{m}_k + B_k^w \mathbf{m}_k^w + B_k \mathbf{u}_k \tag{5}$$

$$\mathbf{\Sigma}_{k+1} = A_k \mathbf{\Sigma}_k A_k' + B_k^w \mathbf{\Sigma}_k^w B_k'^w \tag{6}$$

Hence, each PFT over finite horizon N is defined as a tuple of $\langle \bar{\mathbf{m}}, \bar{\mathbf{\Sigma}} \rangle$, where $\bar{\mathbf{m}} = [\mathbf{m}_1, ..., \mathbf{m}_N]'$ and $\bar{\mathbf{\Sigma}} = [\mathbf{\Sigma}_1, ..., \mathbf{\Sigma}_N]'$. Given the control input, $\bar{\mathbf{m}}$ and $\bar{\mathbf{\Sigma}}$ of a PFT are as follows:

$$\bar{\mathbf{m}} = G_{xx} m_0 + G_{xw} \bar{\mathbf{w}} + G_{xu} \bar{\mathbf{u}} \tag{7}$$

$$\bar{\mathbf{\Sigma}} = G_{xx} \mathbf{\Sigma}_0 G_{xx}' + G_{xw} \bar{\mathbf{\Sigma}}^w G_{xw}' \tag{8}$$

where, G_{xx}, G_{xw}, and G_{xu} are matrices calculated by repeated multiplication of the system matrices in (5) and (6) [22], [23].

Once we generate a library of PFTs, we can calculate the collision risks between vehicles using the intersection of cross-sectional Gaussians.

Library of Controllers $\mathcal{L}_C$: For each trajectory in this library, we design a controller for the controllable agent to follow the trajectory while also minimizing the size of the uncertainty. One can use the *LQR* technique to design a closed loop controller to follow the open loop trajectories in the library. In the presence of uncertainty $\mathbf{w}_k$, the $\mathbf{m}_k$ follows the open loop trajectories. However, the uncertainties (e.g., $\mathbf{\Sigma}_k$) increase while tracking. To minimize the rate of uncertainty growth, and to follow the open loop trajectories better, we instead use a covariance minimization technique.

In the covariance control we look for feedback matrices $G_k \in \mathbf{R}^{n_u \times n_x}$ so that control input $\mathbf{u}_k = G_k \mathbf{x}_k$ minimizes the covariance of joint Gaussian distribution over the trajectory $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ as follows:

$$\begin{aligned} &\min_{\{G_k\}_1^{N-1}} \mathbf{Cov}(\mathbf{X}) \\ &\text{s.t} \quad \{G_k\}_1^{N-1} \in \mathbf{G} \end{aligned} \tag{9}$$

where $\mathbf{G}$ is a convex set representing the constraints on the feedback. To solve (9), we implement the approach provided in [24], [25], where a stabilizing controller for uncertain linear dynamical systems via covariance minimization is designed using the convex optimization method:

$$\begin{aligned} &\min_{\{M_k\}_1^{N-1}, \{G_k\}_1^{N-1}} \lambda_{max}(\mathbf{M}) \\ &\text{s.t} \\ &\mathbf{M} = \begin{bmatrix} S_1 + M_1 & -\tilde{A}_1' S_1 & 0 & \dots & 0 \\ -S_1 \tilde{A}_1 & S_2 + M_2 & -\tilde{A}_2' S_2 & \dots & 0 \\ 0 & S_2 \tilde{A}_2 & S_3 + M_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & S_{N-1} + M_{N-1} \end{bmatrix} \\ &\begin{bmatrix} S_k^{-1} & \tilde{A}_k' \\ \tilde{A}_k & M_k \end{bmatrix} \succeq 0, k = 1, ..., N-1 \\ &\tilde{A}_k = A_k + B_k G_k, \\ &\{G_k\}_1^{N-1} \in \mathbf{G} \end{aligned} \tag{10}$$

where λ_{max} is the largest eigenvalue and $S_k = \mathbf{\Sigma}_k'^w$. Hence, the controllers associated with each maneuver in the library can be computed as follows:

$$u_k = G_k \mathbf{x}_k + u_{o_j} \tag{11}$$

where u_{o_j} is open loop controller to follow the j-th trajectory in the library, and G_k is the solution of optimization in (10).

B. Risk-bounded Conditional Planner with Iterative Updates

In this section, we present an iterative RAO* (iRAO*) algorithm that takes a receding horizon approach to conditional planning and makes iterative updates to the policy during execution. The algorithm is detailed in Algorithm 1 and it calls various parts of the RAO* offline planner as described by Santana [3], including the *RAO**, *expand-policy*, and *update-policy* functions.

Algorithm 1 Iterative RAO*

Input CC-POMDP-MA H, library of PFTs $\mathcal{L}_{PFT}$, library of controllers $\mathcal{L}_c$

Output Optimal policies at each step through execution.

$G, \pi, s_0 \leftarrow$ RAO*(H, $\mathcal{L}_{PFT}$)
$s_{current} \leftarrow s_0$
$action \leftarrow$ get-action(π, $s_{current}$)
execute($action$, $\mathcal{L}_c$)
while not at goal **do**
    $s_{new} \leftarrow$ observations()
    $p_{new} \leftarrow Pr(s_{new}|s_{current}, action)$
    $G, \pi \leftarrow$ iterative-step(G, π, s_{new}, p_{new})
    execute(get-action(π, s_{new}), $\mathcal{L}_c$)
    $s_{current} \leftarrow s_{new}$

Algorithm 2 iterative-step

Input Explicit graph G, policy π, state s, probability p.

Output Expanded graph G', updated policy π'.

mark s as new root in G
remove previous root and sibling branches of s
$G' \leftarrow$ update-risk-pruning(G, p)
while π has some leaf shorter than horizon **do**
    $n, G' \leftarrow$ expand-policy(G', π)
    $\pi \leftarrow$ update-policy(n, G', π)
$\pi' \leftarrow \pi$
return G', π'

Algorithm 3 update-risk-pruning

Input Explicit graph G, probability p.

Output Updated graph G'.

$G' \leftarrow G$, $Z \leftarrow$ set containing root of G'
while $Z \neq \emptyset$ **do**
    $n \leftarrow Z$.pop()
    for each action a from n **do**
        for each child c of n from a **do**
            Multiply execution risk heuristic h_{er} by $1/p$
            Compute new execution risk bounds
        if h_{er} of any c violates risk bound **then**
            Prune all c for action a from n
    Add remaining children of n to Z
return G'

The *observations* function in the main algorithm of iRAO* is domain dependent and not detailed any further in this paper. The *execute* function controls the controllable agent by looking up the specified action in the pre-computed library of controllers. The *iterative-step* function in Algorithm 2 updates the root in the policy, removes unused branches, and expands the planning horizon. The *update-risk-pruning* function in Algorithm 3 is a variation on the *expand-policy* function, but in this case we do not need access to the POMDP models. Instead, the *update-risk-pruning* function performs a quick update to the execution risk heuristic h_{er} using the transition probability of the new state and an update to the risk bounds. The equations used to calculate execution risk bounds are detailed in [3]. While execution risks are model specific, the risk of collision between the vehicle agents in the RAO* function is computed with a Monte Carlo sampling method between maneuver PFTs stored in $\mathcal{L}_{PFT}$. Pruning is complete after all nodes in the explicit graph G have had h_{er} updated and actions that exceed the chance constraints have been removed.

The iterative updates to the policy are not bound to a specific change in the problem or domain. Instead, the updates can come from belief state updates, observed variations in the models, or the addition of new models in the world during execution. This is useful when planning maneuver policies for autonomous vehicles because of the many uncertainties in driving conditions. Expansion of the policy uses the previous search space and execution risk calculations to make efficient updates and allow for quick replanning. The ability to reassess and update the policy during execution ensures that risky actions are replaced as the likelihood of different outcomes evolve. This also enables us to have plans that meet the chance constraints at all times.

V. Results

This section presents two planning problems in the context of autonomous vehicles in order to compare the performance of our algorithm with RAO* and show its online planning capability. Both problems are created in a Unity simulator where vehicles and road environments are simulated using physics engines. In the simulations we use the Dubins vehicle model, where the states include position and velocity $\mathbf{x}_k = [x_k, y_k, v_{x_k}, v_{y_k}]'$. The matrices of the vehicle dynamics in (4) are:

$$A = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 - \frac{\Delta T b}{m} & 0 \\ 0 & 0 & 0 & 1 - \frac{\Delta T b}{m} \end{bmatrix} B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\Delta T}{m} & 0 \\ 0 & \frac{\Delta T}{m} \end{bmatrix} \tag{12}$$

where $m = 1000kg$, $b = 250N.s/m$, $\Delta T = 0.5s$. The control inputs are forces $\mathbf{u}_k = [F_{x_k}, F_{y_k}]'$. The uncertainty parameters are: $\mathbf{m}_k^w = [0, 0, 0, 0]'$, $\mathbf{\Sigma}_k^w = 0.005 I_{4\times4}$, and $B_k^w = I_{4\times4}$. Fig. 3 shows the library of maneuvers.

We solve the convex optimization problem (10) using MATLAB convex programming package *CVX* [26] with constraints $-1000 \leq G_1 = G_2 = ... = G_N \leq 1000$ and $N = 28$ (length of the maneuvers). Obtained feedback is $G_k = \begin{bmatrix} -588.0213 & 0 & -999.9998 & 0 \\ 0 & 0 & 0 & -709.4715 \end{bmatrix}, \forall k$. Fig. 4 shows a library of PFTs under the designed feedback control in Eq. (11) obtained by Eq. (7) and (8).

A. Lane Changing

In the first scenario as shown in Fig. 5, the controllable vehicle in yellow is approaching a highway exit while driving in lane 3. There are two uncontrollable vehicles in red: the fast vehicle in lane 1 and the slow vehicle in lane 2. Each vehicle can choose from a set of actions modeled

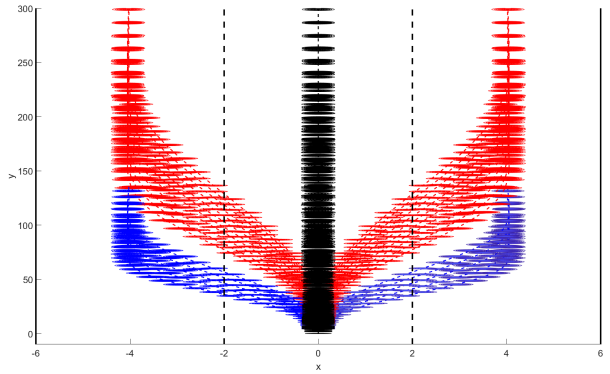


Fig. 4: Library of maneuver PFTs under the feedback control. Each PFT is a sequence of Gaussian probability distributions.

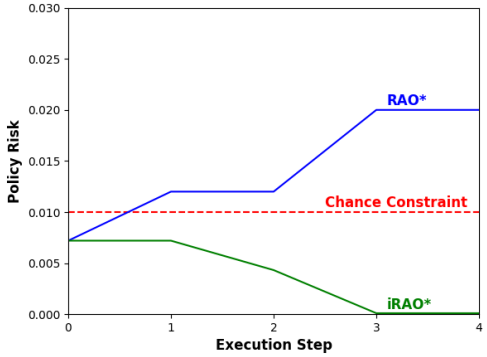


Fig. 6: iRAO* vs RAO* execution risk during execution

using Definition 1. For example, the merging left action requires the left lane to the vehicle to be open and has a consequence of the vehicle in the left adjacent lane. The cost is defined as the execution time, and the means of PFTs are depicted in dashed arrow lines in Fig. 5. For each agent, the continuous state variables are the positions and velocities, and the discrete state is the lane number. The goal of the controllable vehicle is to continue down the road as soon as possible and a chance constraint of 1% is used to bound the risk of collision.

Although all contingencies are considered by the planner, the important probabilities for calculating risk in this example are $Pr(forwardFast|fastDriver)$ = 0.6 and $Pr(mergeRight|fastDriver)$ = 0.02 because of the possible outcome where the fast vehicle accelerates ahead to merge into lane 2, shown in the right-most frame of Figure 5.

RAO* generates a policy with the most-likely maneuver sequence of continuing forward to pass the careful vehicle before merging left into lane 2 to avoid the highway exit. The execution risk for this policy is 0.0072, so the chance constraint is not violated before execution. The policy and a progression of the most-likely states are shown across the frames in Figure 5. The offline policy from RAO* violates the chance constraint after the first action during execution, where we consider the most-likely state transitions.

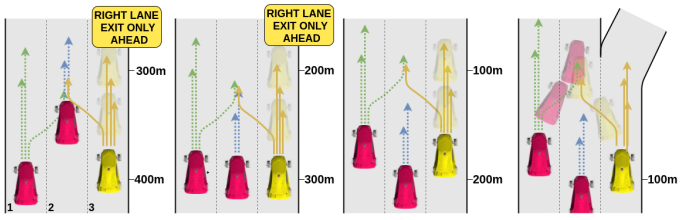
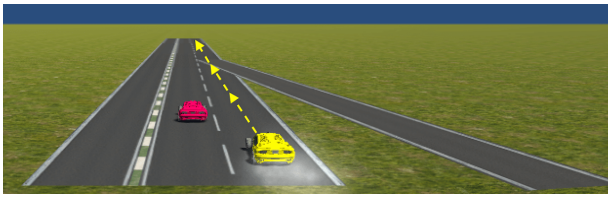


Fig. 5: Lane changing scenario with the offline policy.

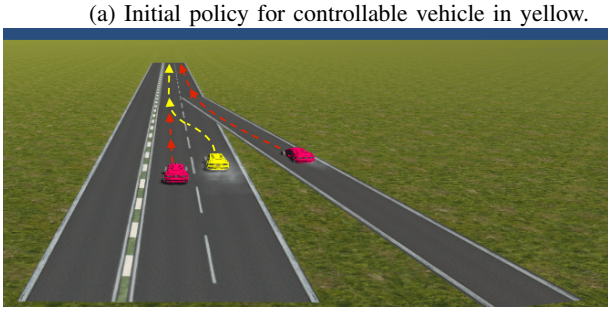
In contrast, the iRAO* planner is able to maintain the chance constraint through execution, shown in Fig. 6. After the first action, the observed state of the aggressive vehicle, which executed the $forwardFast$ maneuver, indicates an increase in the likelihood of the aggressive vehicle merging into lane 2. The updated policies from iRAO* have the autonomous vehicle slowing down after the first maneuver and then safely merging into lane 2 behind the other vehicles. This new policy was not selected at first due to action model costs for maneuvers that are slower than desired velocities.

B. Highway On-Ramp

In the second scenario, the controllable vehicle is driving in the right lane and there is a slow vehicle on the left. The goal is to continue down the highway and a chance constraint of 1% is used to bound the risk of collision. The initial policy generated by iRAO* is to take forward maneuvers shown in Fig. 7a. After the controllable vehicle continues forward according to the policy, an additional vehicle starting in the on-ramp lane is added to our problem model to test the capability of our planner during execution. As shown in Fig. 7b, iRAO* successfully finds a new optimal policy that has the vehicle merge to the left lane after re-evaluating the risks in the new environment.



(a) Initial policy for controllable vehicle in yellow.



(b) Updated policy after seeing the on-ramp vehicle.

Fig. 7: Online policy evolution of iRAO*.

VI. Conclusions

In this paper, we present improvements to the conditional planner RAO* in a multi-agent environment that is subject to uncertainties. By modeling our problem in chance-constrained POMDP domains, and representing continuous actions using probabilistic flow tubes, our approach is able to iteratively generate risk-bounded conditional plans over the receding horizon. We present results from our approach using a Unity simulator and driving scenarios that model stressful traffic situations. The results presented show that the planner is able to generate conditional sequences of probabilistic flow tubes that ensure chance constraints are met through execution and can be quickly updated when there are changes in the models.

However, there are still limitations to the work presented. The first is that actions in our multi-agent model are assumed to take an equal amount of time to execute, allowing for synchronized branching on the conditional plans. We plan to address this limitation with further improvements to RAO* to handle durative actions. There is also planned work on improving the probabilistic flow tubes, so that they are able to represent different classes of uncertainties and continuous dynamical systems. Finally, we have plans to test our algorithm in more scenarios such as a simulated four-way intersection or on real vehicles.

References

[1] P. Poupart, A. Malhotra, P. Pei, K.-E. Kim, B. Goh, and M. Bowling, "Approximate linear programming for constrained partially observable markov decision processes." in *AAAI*, 2015, pp. 3342–3348.

[2] A. Undurti and J. P. How, "An online algorithm for constrained pomdps," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3966–3973.

[3] P. Santana, S. Thiébaux, and B. Williams, "Rao*: An algorithm for chance-constrained pomdps," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI16)*, 2016, pp. 3308–3314.

[4] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, 2013.

[5] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams, "A probabilistic particle-control approximation of chance-constrained stochastic predictive control," *IEEE transactions on Robotics*, vol. 26, no. 3, pp. 502–517, 2010.

[6] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints." *J. Artif. Intell. Res.(JAIR)*, vol. 24, pp. 81–108, 2005.

[7] M. Ono, M. Pavone, Y. Kuwata, and J. Balaram, "Chance-constrained dynamic programming with application to risk-aware robotic space exploration," *Autonomous Robots*, vol. 39, no. 4, pp. 555–571, 2015.

[8] C. Guestrin, D. Koller, and R. Parr, "Multiagent planning with factored mdps," in *Advances in neural information processing systems*, 2002, pp. 1523–1530.

[9] F. Wu, S. Zilberstein, and X. Chen, "Online planning for multi-agent systems with bounded communication," *Artificial Intelligence*, vol. 175, no. 2, pp. 487–511, 2011.

[10] K. H. Wray and S. Zilberstein, "Approximating reachable belief points in pomdps," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 117–122.

[11] J. Pineau, N. Roy, and S. Thrun, "A hierarchical approach to pomdp planning and execution," in *Workshop on hierarchy and memory in reinforcement learning (ICML)*, vol. 65, no. 66, 2001, p. 51.

[12] 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*, vol. 2008. Zurich, Switzerland., 2008.

[13] J. Pineau and G. J. Gordon, "Pomdp planning for robust robot control," in *Robotics Research*. Springer, 2007, pp. 69–82.

[14] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," in *Advances in neural information processing systems*, 2013, pp. 1772–1780.

[15] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 454–460.

[16] J. D. Isom, S. P. Meyn, and R. D. Braatz, "Piecewise linear dynamic programming for constrained pomdps." in *AAAI*, 2008, pp. 291–296.

[17] D. Kim, J. Lee, K.-E. Kim, and P. Poupart, "Point-based value iteration for constrained pomdps," in *IJCAI*, 2011, pp. 1968–1974.

[18] N. J. Nilsson, *Principles of artificial intelligence*. Morgan Kaufmann, 2014.

[19] S. Dong and B. Williams, "Motion learning in variable environments using probabilistic flow tubes," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1976–1981.

[20] A. Hofmann and B. Williams, "Exploiting spatial and temporal flexibility for plan execution of hybrid, under-actuated systems," in *AAAI 2006*, 2006.

[21] M. M. Tobenkin, I. R. Manchester, and R. Tedrake, "Invariant funnels around trajectories using sum-of-squares programming," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 9218–9223, 2011.

[22] L. Blackmore and M. Ono, "Convex chance constrained predictive control without sampling," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2009, pp. 7–21.

[23] M. Dahleh, M. A. Dahleh, and G. Verghese, "Lectures on dynamic systems and control," *A+ A*, vol. 4, no. 100, pp. 1–100, 2004.

[24] K. Dvijotham, E. Todorov, and M. Fazel, "Convex control design via covariance minimization," in *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on*. IEEE, 2013, pp. 93–99.

[25] ——, "Convex structured controller design in finite horizon," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 1, pp. 1–10, 2015.

[26] M. Grant, S. Boyd, and Y. Ye, "Cvx: Matlab software for disciplined convex programming," 2008.