Chance-Constrained Motion Planning

Nikitha Gollamudi[1] Apurva Patil[2]

***Abstract*— In this project, we use reinforcement learning to solve the chance-constraint motion planning problem. Chance-constrained motion planning is a method to synthesize an optimal path in the presence of a noisy environment and robot dynamics, that satisfies a user-specified threshold of failure probability. Failure occurs when the robot hits the obstacles in the domain or the boundary of the domain. Through Lagrangian relaxation, we convert the chance-constrained (risk-constrained) motion problem to a risk-minimizing problem. We show that by choosing an appropriate Lagrange multiplier, we can synthesize a policy which has an appropriate level of safety. The video of the project presentation can be found at https://youtu.be/vvBVWfXRAlsm. Source code can be found at https://github.com/Nikitha2497/RL-Project.**

I. Introduction

Safety-critical systems are required to be controlled within their predefined safe regions over the entire operation horizon. For the optimal path-planning of such systems, one must balance the trade-off between the travel cost and safety. When deployed in real-life environments, these systems are subject to uncertainties that arise due to modeling errors, uncertain localization, and disturbances. While some studies consider robust control under set-bounded uncertainties [1], in many cases, modeling uncertainties with unbounded (e.g. Gaussian) distributions is advantageous over a set-bounded approach [2]. In the case of unbounded uncertainties, it is generally difficult to guarantee safety against all realizations of noise. An alternative approach is to plan a path subject to the constraint that the probability of collision is bounded by a user-specified threshold. This problem is known as the *chance-constrained* or *risk-constrained motion planning problem* [3]. Unfortunately, the probability of collision over the entire path is in general challenging to evaluate and optimize against since it involves a multidimensional integration [4]. Many methods have been proposed in the literature to solve the chance-constrained motion problem such as iterative risk allocation [2], and sampling-based approach [5], both based on the assumption that the probability of collision over the entire path can be expressed as the sum of collision probabilities at each time-step. This assumption does not hold in general because the joint disjunctive probability is not equivalent to the summation of probabilities of individual constraints. In fact, the summation is usually an upper bound of the joint probability, and optimizing for the summed constraints will likely lead to conservative paths. In this project, we wish to take a similar approach that Apurva has been using for her research in the controls domain, however, with the RL algorithms. We will convert the risk-constrained problem into a risk-minimizing one using a Lagrange multiplier and augment the cost function of the MDP with the cost of failure due to collision. The Lagrange multiplier balances the trade-off between the travel cost and the collision probability. Moreover, the Lagrange multiplier can be tuned appropriately to achieve a desired level of safety. We will use the notion of exit-time [6] from the continuous-time stochastic calculus along with the existing RL algorithms (such as Q-learning and semi-gradient SARSA [7]) to solve the risk-minimizing problem with the augmented cost function.

In addition to constrained MDP-based approaches like ours, there exist works that improve safety by generating more samples in the risky region to bootstrap performance in critical scenarios [8]; by using a safety layer at the end of a deep neural network to verify the safety of the resulting policy and replacing with a backup safe action if needed [9]; by proposing a reachability-based trajectory safe guard to ensure the safety of a policy [10], etc.

The report is organized as follows: we consider a discrete state-space problem in Section II and continuous state-space in Section III. In both these sections, we present the problem formulation, algorithms to compute an optimal policy and its safety, and experimental results. Finally, we present the conclusions in Section IV.

II. Discrete State Space

First, we consider a discrete state and action-space problem.

A. Problem Formulation

In this section, we define the environment, chance constraint and the problem formulation of optimal policy synthesis.

1) Environment: We consider a grid-world with an obstacle placed in between the start and the goal state as shown in Fig. 1 The state-space is represented by

$$\mathcal{S} = \{x_0, x_1, \dots x_{63}\}. \tag{1}$$

I, G represent the start and the goal states, respectively, as shown in Fig. 1. The unsafe states are shown in red color and the set of such states is represented by $\mathcal{S}_u$. The states corresponding to the boundary of the domain and the obstacle are unsafe. The set of safe states is represented by $\mathcal{S}_s = \mathcal{S} \backslash \mathcal{S}_u$. There are four actions available, North (N), West (W), South (S), and East (E). The action space is shown as follows:

$$\mathcal{A} = \{N, W, S, E\}. \tag{2}$$

[1]Department of Computer Science, University of Texas at Austin. nikithag@utexas.edu. [2]Walker Department of Mechanical Engineering, University of Texas at Austin. apurvapatil@utexas.edu.

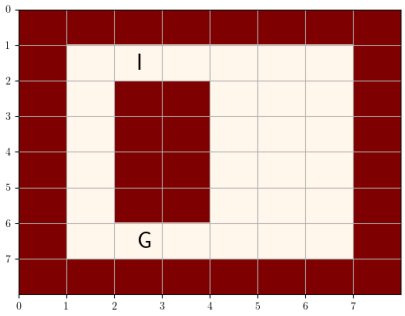


Fig. 1. Grid-world where the red states represent the unsafe states $\mathcal{S}_u$ and the remaining states represent the safe states $\mathcal{S}_s$. The initial and the goal states are shown by I, and G respectively.

We consider the *terminal states* are $\mathcal{S}_u + G$, and the episode ends when the agent reaches one of these terminal states. The environment and the robot dynamics are noisy. If S^N, S^{NW}, and S^{NE} represent the states North, North-West and North-East of the state S, respectively, then the transition dynamics is as follows:

$$\begin{aligned} P(S^N \,|\, S, N) &= 0.9 \\ P(S^{NW} \,|\, S, N) &= 0.05 \\ P(S^{NE} \,|\, S, N) &= 0.05. \end{aligned} \tag{3}$$

Similarly, the transition dynamics for other actions can be written. A reward of $-\lambda$ is given for every transition step and a reward of R^G is given when the goal is reached. The transition and reward functions are not known to the agent.

2) Chance Constraint: We say that the agent fails when it goes into the one of the unsafe states $\mathcal{S}_u$. If the episode starts at time t_0 and ends at t_f, the probability of failure P_{fail} associated with policy π can be written as:

$$P_{fail} = \mathbb{E}_\pi \Big[\mathbb{1}_{S_{t_f} \in \mathcal{S}_u} \,|\, S_{t_0} = I \Big] \tag{4}$$

where S_t represents the state at time t, and $\mathbb{1}_{\mathcal{E}}$ is an indicator function, which returns 1 when the condition $\mathcal{E}$ holds and 0 otherwise. Note that t_f is a random variable that varies from episode to episode. Now, we want P_{fail} to be below a user specified threshold $\Delta \in (0, 1)$. This can be written formally as

$$P_{fail} < \Delta. \tag{5}$$

The constraint (5) is called as a *chance constraint* [2], [3].

3) Optimal Policy Synthesis: Our goal is to find a shortest path from the start state I to the goal state G subject to the constraint (5). Let us first define the return of an episode starting at time t_0 as follows:

$$U_{t_0} = \sum_{t=t_0}^{t_f} R_t \tag{6}$$

where R_t denotes the reward received at time t. Now, the risk-constrained motion planning problem can be formally written as:

Problem 1 (Risk-constrained motion planning problem):

$$\begin{aligned} &\arg\max_\pi \mathbb{E}_\pi \big[U_{t_0} \,|\, S_{t_0} = I \big] \\ &\text{s.t. } P_{fail} < \Delta. \end{aligned} \tag{7}$$

We approach Problem 1 via Lagrangian relaxation. We define the following risk-minimizing motion planning problem:

Problem 2 (Risk-minimizing motion planning problem):

$$\arg\max_\pi \Big\{ \mathbb{E}_\pi \big[U_{t_0} \,|\, S_{t_0} = I \big] - \eta \cdot P_{fail} \Big\}. \tag{8}$$

Here $\eta \geq 0$ is the Lagrange multiplier that balances the trade-off between the expected cumulative reward function and the failure probability. By tuning η, we can achieve the desired level of risk threshold defined by Δ. It is possible to establish the exact correspondence between η and Δ, however, it is not in the scope of this project. In what follows, we will treat η as a constant environment parameter, and focus on Problem 2. In Section II-D, we demonstrate how different values of η achieve different π^* and failure probabilities.

Using (4), Problem 2 can be converted to finding π^* such that

$$\begin{aligned} \pi^* &= \arg\max_\pi \; \mathbb{E}_\pi \big[U_{t_0} - \eta \cdot \mathbb{1}_{S_{t_f} \in \mathcal{S}_u} \,|\, S_{t_0} = I \big] \\ &= \arg\max_\pi \; \mathbb{E}_\pi \big[U'_{t_0} \,|\, S_{t_0} = I \big]. \end{aligned} \tag{9}$$

where,

$$U'_{t_0} = U_{t_0} - \eta \cdot \mathbb{1}_{S_{t_f} \in \mathcal{S}_u}. \tag{10}$$

(10) suggests that the agent gets a reward of $-\eta$ at the end of the episode if that episode ends in one of the unsafe states $\mathcal{S}_u$. This is in addition to the rewards of $-\lambda$ for every transition step, and R^G for the goal state. Note that unlike [2], and [5], the formulation in (9) doesn't make a conservative approximation of the failure probability over the entire trajectory as the sum of failure probabilities at each time-step.

B. Algorithm

In this section, we modify the standard Q-learning algorithm [7] to solve the problem (9). We use the ϵ-greedy behaviour policy and ϵ is decreased over the episodes. The step size α is also decreased over the episodes. The pseudo code of the modified Q-learning algorithm is provided in Algorithm 1.

At the end of Algorithm 1, we estimate optimal Q values: Q^*. We can then find the optimal policy as a greedy policy:

$$\pi^*(s) = \arg\max_a \; Q^*(s, a), \quad \forall \; s \in \mathcal{S}.$$

C. Risk Estimation of π^*

In this section, we find how risky the computed optimal policy π^* (derived using Algorithm 1) is.

1) Problem Formulation: The problem of risk estimation of π^* can be states as follows:

Problem 3 (Risk Estimation): Find the probability of failure (P_{fail}) defined as (4), given a policy π^*.

We note that the value-function of the start state I under policy π^* can be written as

$$v_{\pi^*}(I) = \mathbb{E}_{\pi^*} \big[U_{t_0} \,|\, S_{t_0} = I \big]. \tag{11}$$

Algorithm 1: Modified Q-learning

Parameters: step size $\alpha \in (0,1]$, $\epsilon > 0$, rewards $\lambda, R^G, \eta > 0$, $\gamma = 1$.

Initialize $Q(s,a),\ \forall\ s \in \mathcal{S}, a \in \mathcal{A}$, arbitrarily except that $Q(s,.) = 0,\ \forall\ s \in \mathcal{S}_u$, and $Q(G,.) = 0$.

Loop for each episode:

  $S_{t_0} \leftarrow I$

  Loop for each step of the episode:

    Choose A_t from S_t using ϵ-greedy policy derived from Q

    Take action A_t and observe R_{t+1}, S_{t+1}

    if $S_{t+1} = G$ **then**

      $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big[R_{t+1} + R^G - Q(S_t, A_t)\big]$

      break

    end

    else if $S_{t+1} \in \mathcal{S}_u$ **then**

      $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big[R_{t+1} - \eta - Q(S_t, A_t)\big]$

      break

    end

    else

      $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\big]$

      $S_t \leftarrow S_{t+1}$

    end

  end

end

Now, assume that after we find the policy π^*, we change our reward function. We set

$$\lambda = 0, \quad R^G = 0, \quad \eta = 1. \tag{12}$$

Under this reward function we can write

$$U_{t_0} = \mathbb{1}_{S_{t_f} \in \mathcal{S}_u} \tag{13}$$

and

$$v_{\pi^*}(I) = \mathbb{E}_{\pi^*}\big[\mathbb{1}_{S_{t_f} \in \mathcal{S}_u} \mid S_{t_0} = I\big]. \tag{14}$$

From (4), we recognize that computing the value-function defined in (14) is in fact the risk estimation problem (Problem 3). Now, we find (14) using TD(0).

2) Algorithm: We modify the standard prediction algorithm TD(0) [7] to find $v_{\pi^*}(I)$. The step size α is decreased over the episodes. The pseudo code of the modified TD(0) is provided in Algorithm 2.

Algorithm 2: Modified TD(0)

Input : π^* synthesized from Algorithm 1

Parameters: step size $\alpha \in (0,1]$, $\gamma = 1$

Initialize $V(s),\ \forall\ s \in \mathcal{S}$ arbitrarily except that $V(s) = 0, \forall\ s \in \mathcal{S}_u$, and $V(G) = 0$.

Loop for each episode:

  $S_{t_0} \leftarrow I$

  Loop for each step of the episode:

    $A_t \leftarrow \pi^*(S_t)$

    Take action A_t and observe S_{t+1}

    if $S_{t+1} = G$ **then**

      $V(S_t) \leftarrow V(S_t) - \alpha\big[V(S_t)\big]$

      break

    end

    else if $S_{t+1} \in \mathcal{S}_u$ **then**

      $V(S_t) \leftarrow V(S_t) + \alpha\big[1 - V(S_t)\big]$

      break

    end

    else

      $V(S_t) \leftarrow V(S_t) + \alpha\big[\gamma\, V(S_{t+1}) - V(S_t)\big]$

      $S_t \leftarrow S_{t+1}$

    end

  end

end

At the end of Algorithm 2, we estimate v_{π^*}. We can then find the failure probability of policy π^* as

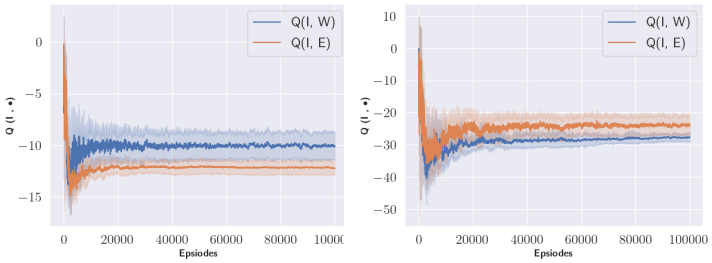
$$P_{fail} = v_{\pi^*}(I). \tag{15}$$

D. Experiments

We performed the experiments on the grid-world described in Fig. 1 in a noisy environment. Recall that the following are the parameters - step size $\alpha \in (0,1]$, $\epsilon > 0$, rewards $\lambda, R^G, \eta > 0$, γ. We assumed the rewards to be non-discounted. So, the γ is always set to 1. We used 100000 episodes for the control learning. The step size α and the ϵ are decreased slowly for every 1000 episodes. We considered the control cost λ to be 1 and the goal reward R^G to be 5. With these values of λ and R^G we are able to show the difference between multiple η values. With the same set of parameters except η, we ran the modified Q-learning algorithm 1 for different η values.

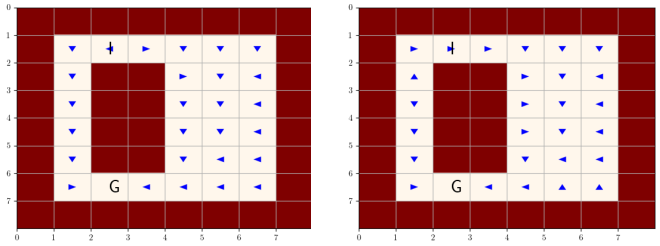
We would expect that in the grid-world described in 1, the agent learns to go **West** for lower η values and the agent learns to take a more safer route, that is **East** for higher η values. We were able to observe the same in our experiments. So we considered two η values, η_1 = 15 and η_2 = 50 for our experiments. For the graphs in Figures 2 and 4 we plotted the confidence interval with one standard deviation for 10 runs.

In Figure 2, we show that the action-values of the initial state I and action E and the Q state action values of the initial state I and action W converge for both the η values. We can also see that the $Q(I, W)$ is greater than the $Q(I, E)$



(a) $\eta = 15$ (b) $\eta = 50$

Fig. 2. Convergence of the Q-values of the initial state I and actions W and E for two different η's. The shaded region represent one standard deviation.



(a) $\eta = 15$ (b) $\eta = 50$

Fig. 3. Optimal Greedy Policy after 100000 episodes

for the smaller η = 15 and $Q(I, E)$ is greater than the $Q(I, W)$ is for the higher η = 50.

At the end of modified Q-learning 1, the agent learns the optimal greedy policy. In Figure 3, we can see the optimal greedy policy learnt by the agent after 100,000 episodes for both the η values 15 and 50.

Figure 4 (a) shows the no noise trajectory for both the η values. We use the optimal deterministic greedy policy learnt from a noisy environment at the end of the 100,000 episodes to plot a trajectory in a no noise environment. We can see that the agent takes the shortest path although riskier for a smaller η and takes a more safer although longer router for a higher η.

Lastly, we estimated the probability of failure P_{fail} using the modified TD prediction algorithm 2. Figure 4 (b) shows the P_{fail} values as the episodes increase for both the η values. We plotted the P_{fail} for 100,000 episodes over 10 runs with a confidence interval of one standard deviation. We can see that the P_{fail} value converges for both η_1 = 15 and η_2 = 50. We can also see that the P_{fail} value for a smaller η is higher than the P_{fail} value for a higher η.

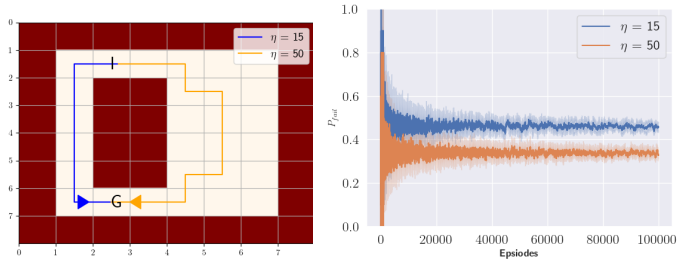
III. Continuous State Space

We consider the similar problem as Section II, however we make our state-space continuous.

A. Problem Formulation

We consider a continuous-space domain with an obstacle as shown in Fig. 5. The state-space is represented by

$$\mathcal{S} = \left\{ \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T, s.t. \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \in \mathbb{R}^2 \right\}. \quad (16)$$



(a) (b)

Fig. 4. (a) No noise trajectories for $\eta = 15$ and $\eta = 50$ (b) Convergence of P_{fail} for $\eta = 15$ and $\eta = 50$ with one standard deviation.

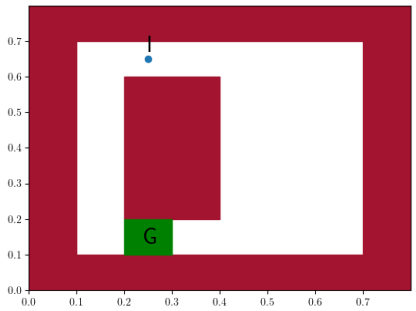


Fig. 5. Continuous state space where the red region represent the unsafe states $\mathcal{S}_u$ and the white region represents the safe states $\mathcal{S}_s$. The green region represents the goal region $\mathcal{S}_G$ and the initial state is shown by I.

As in the discrete state-space, the unsafe region is shown in red hatching and represented by $\mathcal{S}_u$. The safe region is represented by $\mathcal{S}_s = \mathcal{S} \backslash \mathcal{S}_u$. The start state is I and the goal region $\mathcal{S}_G$ is shown by a blue rectangle. Similar the Section II-A.1, there are four actions available and the action space is same as (2). The terminal states are $\mathcal{S}_u + G$. The environment and the robot dynamics are noisy. If the action E is taken at time t then the agent's states change as follows:

$$\begin{aligned} x_1(t+1) &= x_1(t) + \beta_1 + n_1(t), & n_1(t) &\sim \mathcal{N}\left(0, \sigma^2\right), \\ x_2(t+1) &= x_2(t) + n_2(t), & n_2(t) &\sim \mathcal{N}\left(0, \sigma^2\right). \end{aligned} \quad (17)$$

where β_1 is some environment parameter, and n_1, n_2 are Gaussian white noises having covariance σ^2, also an environment parameter. Similarly, if the action W is taken then the dynamics is same as (17) except, we will have $-\beta_1$ instead of $+\beta_1$. If the action N is taken then we get

$$\begin{aligned} x_1(t+1) &= x_1(t) + n_1(t), & n_1(t) &\sim \mathcal{N}\left(0, \sigma^2\right), \\ x_2(t+1) &= x_2(t) + \beta_2 + n_2(t), & n_2(t) &\sim \mathcal{N}\left(0, \sigma^2\right). \end{aligned} \quad (18)$$

and for action S we get the same dynamics as (18) except $-\beta_2$. Similar to (17), β_2 is an environment parameter. The reward function is same as Section II-A.1, and the transition and the reward functions are not known to the agent.

The chance constraint and the problem formulation of optimal policy synthesis are same as the Sections II-A.2, and II-A.3 respectively. Here our goal is to find a shortest path

from the start state I to the goal region $\mathcal{S}_G$, subject to the constraint (5). As before, we will solve the risk-minimizing motion planning problem defined as Problem 2.

B. Function Approximation

In this project, we consider linear state and action-value function approximations as follows:

$$\hat{v}(s, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x}(s)$$
$$\hat{q}(s, a, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x}(s, a)$$

where $\boldsymbol{x}(s)$, $\boldsymbol{x}(s, a) \in \mathbb{R}^d$ are d-dimensional feature vectors and $\boldsymbol{w} \in \mathbb{R}^d$ is weight vector. We considered two types of features, polynomials and tile coding.

1) Polynomial Features: For the state-value function, we choose the following four dimensional polynomial feature vector

$$\boldsymbol{x}(s) = \begin{bmatrix} 1 & x_1 & x_2 & x_1 x_2 \end{bmatrix}^T. \tag{19}$$

The initial 1 feature allows the representation of affine functions in the original state variables x_1, x_2, and the final product feature, $x_1 x_2$, enable interactions to be taken into account. The polynomial feature vector $\boldsymbol{x}(s, a)$ can also be constructed similarly whose dimension will be 16, since our action space is four dimensional.

2) Tile Coding: In the tile coding, we use one tiling of size $[0, 1] \times [0, 1]$ having a uniform grid. This is similar to state aggregation. The width of each tile is 0.1. The feature vector $\boldsymbol{x}(s)$ will have 100 components, all of which will be 0 except for one corresponding to the tile that s falls within. Similarly, the feature vector $\boldsymbol{x}(s, a)$ can be constructed whose dimension will be 400. All of these 400 components will be 0, except for one.

C. Algorithm

In this section, we modify the standard semi-gradient SARSA [7] to solve the problem (9) for continuous state-space. Unlike the discrete state-space, here, we use SARSA instead of Q-learning to avoid the deadly triad. Since SARSA is an on-policy control algorithm both behaviour and target policies are the same. We use ϵ-greedy policy and ϵ is decreased over the episodes. Similarly, step size α is also decreased over the episodes. Since we are using a linear function approximation, $\nabla \hat{q}(s, a, \boldsymbol{w}) = \boldsymbol{x}(s, a)$. The pseudo code of the modified semi-gradient SARSA is provided in Algorithm 3.

At the end of Algorithm 3, we estimate the optimal weight vector $\boldsymbol{w}^*$. We can then find the optimal policy as a greedy policy:

$$\pi^*(s) = \arg\max_a \ {\boldsymbol{w}^*}^T \boldsymbol{x}(s, a), \quad \forall \ s \in \mathcal{S}.$$

D. Risk Estimation of π^*

In this section, we find how risky the optimal policy π^* (derived using Algorithm 3) is. The problem of risk estimation of π^* is same as P0roblem 3. Again, under the reward function (12), the value function of the start state I under policy π^* can be written as (14), which is the failure probability of π^*. Now, we modify the standard semi-gradient TD(0) [7] in order to find $v_{\pi^*}(I)$. The step size α is decreased over the episodes. Since we are using a linear function approximation, $\nabla \hat{v}(s, \boldsymbol{w}) = \boldsymbol{x}(s)$. The pseudo code of the modified semi-gradient TD(0) is provided in Algorithm 4.

Algorithm 3: Modified semi-gradient SARSA

Input : a differentiable linear action-value function parameterization $\hat{q}(s, a, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x}(s, a)$

Parameters: step size $\alpha \in (0, 1]$, $\epsilon > 0$, rewards $\lambda, R^G, \eta > 0$, $\gamma = 1$.

Initialize value-function weights $\boldsymbol{w}_{t_0} \in \mathbb{R}^2$ arbitrarily (e.g.. $\boldsymbol{w}_{t_0} = \boldsymbol{0}$)
Loop for each episode:
  $S_{t_0} \leftarrow I$
  Choose A_{t_0} from S_{t_0} using ϵ-greedy policy derived from $\hat{q}(S_{t_0}, ., \boldsymbol{w}_{t_0})$
  Loop for each step of the episode:
    Take action A_t and observe R_{t+1}, S_{t+1}
    if $S_{t+1} \in \mathcal{S}_G$ **then**
      $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha\big[R_{t+1} + R^G - \hat{q}(S_t, A_t, \boldsymbol{w}_t)\big]\boldsymbol{x}(S_t, A_t)$
      break
    end
    else if $S_{t+1} \in \mathcal{S}_u$ **then**
      $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha\big[R_{t+1} - \eta - \hat{q}(S_t, A_t, \boldsymbol{w}_t)\big]\boldsymbol{x}(S_t, A_t)$
      break
    end
    else
      Choose A_{t+1} from S_{t+1} using ϵ-greedy policy derived from $\hat{q}(S_{t+1}, ., \boldsymbol{w}_t)$
      $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha\big[R_{t+1} + \gamma\, \hat{q}(S_{t+1}, a_{t+1}, \boldsymbol{w}_t) - \hat{q}(S_t, A_t, \boldsymbol{w}_t)\big]\boldsymbol{x}(S_t, A_t)$
      $S_t \leftarrow S_{t+1}$
      $A_t \leftarrow A_{t+1}$
    end
  end
end

E. Experiments

In this section, we discuss the experiments we conducted in the continuous state space shown in the Figure 5. Since this is a continuous state space, we have to use function approximation. As mentioned in the previous section, we used Linear Function Approximation for our experiments. We tried polynomial feature selection and tile coding feature selection. With the polynomial feature extraction, the state action values seem to diverge. We think the following might be the reason. Consider two points which are next to each other but one is in the unsafe region and the other is in the safe region but very close to the goal region. So, we

Algorithm 4: Modified semi-gradient TD(0)

Input : π^* synthesized from Algorithm 3, a differentiable linear state-value function parameterization $\hat{v}(s, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x}(s)$

Parameters: step size $\alpha \in (0, 1]$, $\gamma = 1$

Initialize value-function weights $\boldsymbol{w}_{t_0} \in \mathbb{R}^2$ arbitrarily (e.g., $\boldsymbol{w}_{t_0} = \mathbf{0}$)

Loop for each episode:

  $S_{t_0} \leftarrow I$

  Loop for each step of the episode:

    $A_t \leftarrow \pi^*(S_t)$

    Take action A_t and observe S_{t+1}

    if $S_{t+1} \in \mathcal{S}_G$ **then**

      $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \alpha\, \hat{v}(S_t, \boldsymbol{w}_t)\, \boldsymbol{x}(S_t)$

      break

    end

    else if $S_{t+1} \in \mathcal{S}_u$ **then**

      $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha \big[1 - \hat{v}(S_t, \boldsymbol{w}_t)\big] \boldsymbol{x}(S_t)$

      break

    end

    else

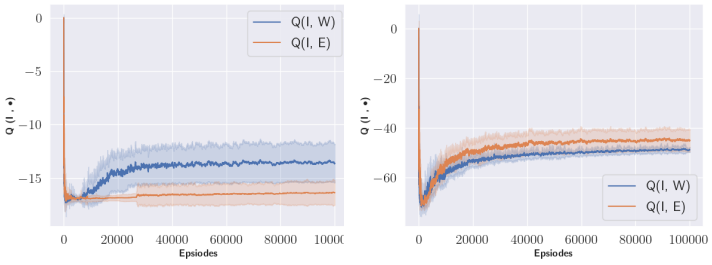
      $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha \big[\gamma \hat{v}(S_{t+1}, \boldsymbol{w}_t) - \hat{v}(S_t, \boldsymbol{w}_t)\big] \boldsymbol{x}(S_t)$

      $S_t \leftarrow S_{t+1}$

    end

  end

end

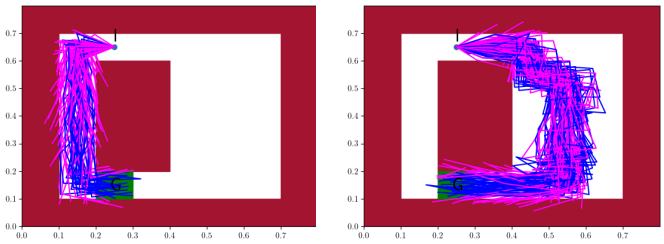


(a) $\eta = 15$ (b) $\eta = 70$

Fig. 6. Convergence of the Q-values of the initial state I and actions W and E for two different η's. The shaded region represent one standard deviation.

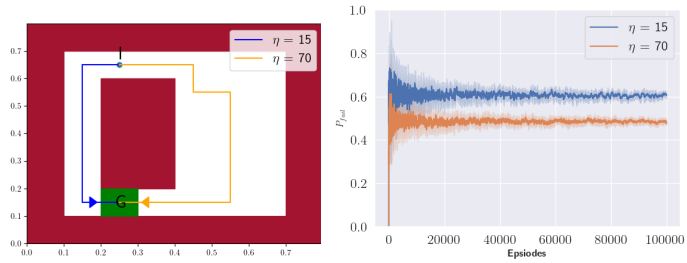
would want these two states to have very different values but we believe the polynomial feature generalization will result in the two states having similar values. So, we present the results using Tile coding in this report.

Similar to the discrete space, recall that the following are the parameters - step size $\alpha \in (0, 1]$, $\epsilon > 0$, rewards $\lambda, R^G, \eta > 0$, γ. We assumed the rewards to be non-discounted. So, the γ is always set to 1. We used 100000 episodes for the control learning. The step size α and the ϵ are decreased slowly for every 1000 episodes. We considered the control cost λ to be 1 and the goal reward R^G to be 5. With these values of λ and R^G we are able to show the difference between multiple η values. With the same set of



(a) $\eta = 15$ (b) $\eta = 70$

Fig. 7. 100 sample trajectories generated using π^* for $\eta = 15$ and $\eta = 70$. The trajectories are color-coded; magenta paths go into the unsafe region $\mathcal{S}_u$, while blue paths go to the goal region $\mathcal{S}_G$.



(a) (b)

Fig. 8. (a) No noise trajectories for $\eta = 15$ and $\eta = 70$ (b) Convergence of P_{fail} for $\eta = 15$ and $\eta = 70$ with one standard deviation.

parameters except η, we ran the modified semi-gradient Sarsa algorithm 3 for different η values.

In Figure 6, we show that the action-values of the initial state I and action E and the action values of the initial state I and action W converge for both the η values. We can also see that the $Q(I, W)$ is greater than the $Q(I, E)$ for the smaller η = 15 and $Q(I, E)$ is greater than the $Q(I, W)$ is for the higher η = 70. At the end of modified semi-gradient Sarsa 3, the agent learns the optimal greedy policy.

Figure 8 (a) shows the no noise trajectories for both the η values. Similar to the discrete case, we use the optimal deterministic greedy policy learnt from a noisy environment at the end of the 100,000 episodes to plot trajectories in a no noise environment. We can see that the agent takes the shortest path although riskier for a smaller η and takes a more safer although longer route for a higher η.

We estimated the probability of failure P_{fail} using the modified semi-gradient TD prediction algorithm 4. Figure 8 (b) shows the P_{fail} values as the episodes increase for both the η values. We plotted the P_{fail} for 100,000 episodes over 10 runs with a confidence interval of one standard deviation. We can see that the P_{fail} value converges for both $\eta = 15$ and $\eta = 70$. We can also see that the P_{fail} value for a smaller η is higher than the P_{fail} value for a higher η.

Finally, we plotted the sample trajectories using the optimal deterministic greedy policy learnt at the of the control algorithm. In Figure 7, we can see the sample trajectories for both η values 15 and 70. The blue lines represent the trajectories that go into the goal region and the magenta lines

represents the trajectories that go into the unsafe region. This further provides evidence that longer but more safer route is taken for higher eta values.

IV. Conclusion

In this project, we modified the standard RL algorithms to solve the chance-constraint motion planning problem. Through Lagrangian relaxation, we converted the chance-constrained (risk-constrained) motion problem to a risk-minimizing problem. We showed that by choosing an appropriate Lagrange multiplier, we can synthesize a policy which has an appropriate level of safety. We also computed the risk associated with the synthesized policies by modifying the RL prediction algorithms. These algorithms have shown promising results. In the future, we plan to extend the presented results for the continuous action spaces.

References

[1] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996.

[2] L. Blackmore, M. Ono, and B. C. Williams, "Chance-constrained optimal path planning with obstacles," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1080–1094, 2011.

[3] K. Oguri, M. Ono, and J. W. McMahon, "Convex optimization over sequential linear feedback policies with continuous-time chance constraints," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 6325–6331.

[4] K. M. Frey, T. J. Steiner, and J. P. How, "Collision probabilities for continuous-time systems without sampling," in *Proc. Robot.: Sci. Syst.*, 2020.

[5] B. Luders, M. Kothari, and J. How, "Chance constrained rrt for probabilistic robustness to environmental uncertainty," in *AIAA guidance, navigation, and control conference*, 2010, p. 8160.

[6] B. Oksendal, *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[8] O. Andersson, M. Wzorek, and P. Doherty, "Deep learning quadcopter control via risk-aware active learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[9] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–7.

[10] Y. S. Shao, C. Chen, S. Kousik, and R. Vasudevan, "Reachability-based trajectory safeguard (rts): A safe and fast reinforcement learning safety layer for continuous control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3663–3670, 2021.