A Risk-Averse Preview-Based $Q$-Learning Algorithm: Application to Highway Driving of Autonomous Vehicles

Majid Mazouchi, Subramanya P. Nageshrao, and Hamidreza Modares, Member, IEEE

Abstract—A risk-averse preview-based $Q$-learning planner is presented for navigation of autonomous vehicles (AVs). To this end, the multilane road ahead of a vehicle is represented by a finite-state nonstationary Markov decision process (MDP). A risk assessment unit module is then presented, which leverages the preview information provided by sensors along with a stochastic reachability module to assign reward values to the MDP states and update them as scenarios develop. A sampling-based risk-averse preview-based $Q$-learning algorithm is finally developed, which generates samples using the preview information and reward function to learn risk-averse optimal planning strategies without actual interaction with the environment. The risk factor is imposed on the objective function to avoid fluctuation of the $Q$ values, which can jeopardize the vehicle’s safety and/or performance. The overall hybrid automaton model of the system is leveraged to develop a feasibility check unit module that detects unfeasible plans and enables the planner system to react proactively to the changes of the environment. Finally, to verify the efficiency of the presented algorithm, its implementation on two highway driving scenarios of an AV in a varying traffic density is considered.

Index Terms—Autonomous vehicle (AV), log-expected-exponential bellman inequality, risk-averse $Q$-learning.

I. INTRODUCTION

There are generally two coupled requirements that every autonomous vehicle (AV) must satisfy: the vehicle’s performance and its safety. Vehicle’s performance captures the cost incurred in achieving its goals (i.e., the time of arrival, the energy consumption, and the passenger’s comfort). Vehicle’s safety requirements impose constraints on its maneuvers to ensure that its trajectories never enter the unsafe set, while it traverses toward its goals. The unsafe set is more general than the failure set: the set of states for which there is no control action to prevent them from eventually entering the failure set. In some real-world applications, the conflict between safety and performance might arise as scenarios develop. This is because operating with an optimal performance might increase the vehicle’s risk of getting into an unsafe set in some scenarios (e.g., to have a minimum arrival time, one needs to increase the speed to the maximum allowable speed). This is exacerbated in the presence of uncertainties under which the performance level that can be achieved without safety violation is not known in advance. Standard practice typically designs plans that optimize the expected value of vehicle’s performance under uncertainties while ignoring its variance. However, while optimizing the expected value can lead to a good performance on average, not all system responses would be satisfactory. This is because the expected value criterion does not account for the risk of possibility of low-performance trajectories, which can cause an oscillatory behavior around the desired trajectory. This, in turn, increases the risk of getting into an unsafe set.

Since many sources of uncertainty in AVs are aleatory (i.e., random), it is reasonable to design a planner to assure that the cost of its implementation is acceptable for all system trajectories realized from probability distributions of the random sources. Risk-averse strategies have been widely and successfully used to control stochastic systems to avoid performance fluctuation and reduce the risk of occurrence of catastrophic events, yet low probability [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. Risk measure can be inferred as a mapping from random variables to scalars [14], [15], [16] and risk-averse optimization interpolates between the expectation-optimization approach and the robust worst case approach by hedging against low probability events without being overly conservative. Several risk measures are considered in the literature, including value-at-risk (VaR) measure [17], [18], [19], [20], [21] and entropic risk measures [15]. The former measure optimizes the expected cost while assuring that its variance is under an acceptable threshold. The latter simultaneously optimizes the average cost and its variance. Risk-averse planning for AVs has also been developed in [22] and [23]. However, they are limited to stationary environments and/or known environments. Besides, they do not leverage the preview information provided by sensors to reduce conservatism. Finally, under epistemic...
 uncertainties (i.e., lack of knowledge), they require direct interaction with the environment to learn about it. However, planning without requiring extensive direct exploration or online actual interaction is significantly beneficial to autonomous driving, for which online actual interaction might be expensive or dangerous.

Reinforcement learning (RL) [24], [25], [26], [27], [28], [29], [30], [31], as the main tool for solving sequential decision-making problems under epistemic uncertainties, has been widely leveraged to learn an optimal control policy for uncertain systems. To account for both aleatory and epistemic uncertainties, risk-averse RL algorithms have been considered to solve risk-averse stochastic optimal control (RASOC) problems in [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], and [44], mainly for Markov decision processes (MDPs). Value iteration and policy iteration algorithms are developed in [33] and [36] for risk-sensitive decision-making in an MDP framework. Convex analytic approaches are developed in [34] and [35] for risk-aware MDPs. Shen et al. [37] presented iterative risk measures, which only depend on the current state rather than on the whole history. Mihatsch and Neuneier [38] and Borkar et al. [39] investigated a risk-sensitive RL algorithm by applying utility functions to the temporal difference error. The most widely used RL algorithm for the RASOC problem is the policy gradient (PG) method [25]. Prashanth [40] and Tamar et al. [41] employed the PG method to risk-aware MDPs with condition VaR measure appearing in either the objective or constraints. PG has been utilized to learn the solution to VaR setting [42], [43] and entropic utility setting [44]. However, PG methods generally result in nonconvex optimization problems and are sensitive to fixed gradient step size, an inappropriate choice that can easily make it divergent. Moreover, existing results on risk-averse RL require online interaction with the environment to learn about the value functions or their gradients. For safety-critical systems, however, it is highly preferred to develop risk-averse RL algorithms capable of learning a policy based only on available data collected from applying a series of control actions to the system.

In this article, we design a risk-averse high-level planner for the navigation of AVs between lanes around static and moving obstacles. A hierarchical control structure is leveraged: a risk-averse high-level planner chooses a sequence of waypoints and their corresponding high-level actions, and a low-level controller drives the AV to assigned waypoints. From the high-level planner’s perspective, the multilane road ahead of a vehicle is represented by a finite-state nonstationary MDP. As scenarios develop, a risk assessment unit (RAU) module leverages preview information of what lies ahead of the ego vehicle, such as other vehicles and road conditions to map probabilistic reward values to the MDP. The RAU module uses a probabilistic reachability module to calculate the probability of MDP states being occupied by other vehicles or obstacles. The probabilistic reward information of MDP is then obtained based on both probabilities of safety violation as well as a goal-reaching objective. A convex program-based preview-based $Q$-learning planner is then developed that turns the preview probabilistic reward information into a noniterative risk-averse planning engine. The optimizing-based risk-averse $Q$-learning requires sampling state-action-reward tuples from the MDP. One advantage of the presented approach is that these samples can be generated using the preview reward function without actual interaction with the environment. When new preview information arrives or the infeasibility of the current plan is certified, replanning is considered. To check for infeasibility, the overall hybrid automaton model of the system is derived to develop the feasibility check unit (FCU) module. The developed scheme is evaluated for a case study of risk-averse AV navigation between lanes around obstacles. The contributions of this article compared to existing planning approaches are listed as follows.

1) Compared to sample-based planning approaches such as rapidly exploring random tree (RRT) [23], [45], [46], [47] and its variants such as RRT* [48], more realistic objective functions can be optimized by the presented approach, rather than just distance to the goal. Moreover, in addition to the expected value, the variance of the cost is also optimized. Finally, in sharp contrast to RRT, transferring to new scenarios can be performed based on the similarity of the reward functions with past learned scenarios to significantly accelerate the search for an optimal path.

2) In sharp contrast to existing (deep) RL algorithms for planning [49], the presented approach accounts for the risk or variance of the utility of outcomes, which leads to a better tradeoff between safety and optimality. Moreover, preview information along with reachability analysis is leveraged to generate imagined samples (i.e., preview samples) without requiring actual interaction with the environment. Utilizing these imagined samples and the FCU module enables the planner to proactively avoid risky exploration of the actual environment and react fast to the changes in the environment. Finally, the presented approaches account for the difficulties of the lower level controller in traversing between waypoints.

The rest of this article is organized as follows. Section II provides an overview of the proposed risk-averse learning planner framework and structure. Section III provides the problem statement and formulation. The proposed risk-averse $Q$-learning planner is presented in Section IV. The implementation of the proposed control scheme for two highway driving scenarios of an AV in varying traffic density is given in Section V. Section VI concludes this article with a note on our future research.

**Notation:** Throughout this article, we use $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{N}$ to denote the sets of real numbers, nonnegative real numbers, and nonnegative integers, respectively. $\mathbb{R}^n$ and $\mathbb{R}^{n \times m}$ denote the $n$-dimensional real vector space and the $n \times m$ real matrix space, respectively. For the given variables $x$ and $y$, $(x, y)$ denotes the inner product of $x$ and $y$. For a random variable $X$, $\mathbb{E}(X)$ denotes the expectation and $\text{Var}(X) = \mathbb{E}(X^2) - \mathbb{E}(X)^2$ refers to the variance of $X$. $\exp(\cdot)$ is an exponential operator...
such that \( \exp(x) = e^x \). For convenience, \( \mathbb{E}_\xi \) refers to the expectation over the random variable \( \xi \).

II. OVERVIEW OF THE PRESENTED RISK-AVERSE LEARNING ALGORITHM

The proposed hierarchical risk-averse control framework is shown in Fig. 1. We consider an AV acting in an uncertain and ever-changing environment. The control system of the AV is hierarchical. A high-level risk-averse learning planner provides a sequence of waypoints to traverse in the discrete-time domain, and a low-level continuous-time controller provides control actuation to drive the AV to the waypoints. The high-level risk-averse planner leverages a preview-based learner in the sense that it learns an optimal plan using the preview information provided by its embedded sensors before a scenario actually becomes apparent. To this end, the RAU module first provides the preview reward function for the current environment. The RAU module is composed of: 1) a predictive traffic submodule that uses embedded sensors along with appropriate sensory fusion and prediction algorithms [50] to predict traffic situations; 2) a predictive stochastic reachability submodule to predict the probability that a traffic participant would occupy a state in the future within a predefined horizon; and 3) a preview reward mapping submodule to assign reward values to the MDP states. A risk-averse preview-based \( Q \)-learning planner then turns this preview reward function into a risk-averse waypoint-generation engine. The presented risk-averse \( Q \)-learning algorithm is a noniterative optimization-based algorithm for which its samples can be imagined and generated from the AVs’ perspective once the reward function is previewed. The risk-averseness accounts for the uncertainty in the reward function (which results from the uncertainty from the behavior of other participants, road conditions, and so on) as well as the difficulties of the lower level controller in traversing between waypoints. Upon the arrival of new preview information and before the new scenario becomes apparent, the FCU module uses a reachability analysis to verify the feasibility and safety of the planned waypoints along with the corresponding low-level control actuation and triggers replanning if needed. The proposed risk-averse learning planner framework is agnostic to the type of low-level controller and accounts for its difficulty in traversing between waypoints.

III. PROBLEM DESCRIPTION AND FORMULATION

A. Agent and Environment Model

To perform planning in highways, which are actually continuous-state domains, it has become common practice to use discrete planners by uniformly discretizing dimensions of the problem [51], [52]. Grid-world MDPs are the most common discrete models used for planning. However, planning over a long horizon by finding a policy for the entire domain simultaneously leads to an exponential growth in problem size as the number of states of the MDP increases. To sidestep this problem, the environment (i.e., the ground MDP) is decomposed into a series of grid-world MDPs with a finite number of cells in which the local reward and transition functions are evolved in a discrete-time fashion. Therefore, at given predefined discrete times \( \tau_{env} > 0 \), the AV senses only its neighborhood, i.e., a small part of the highway as shown in Fig. 2 and then updates the MDP grid world (environment) through updating the local reward and transition functions of each cell. It is assumed that the changes in the behavior of other participants and the highway conditions and geometry are insignificant in a small-time horizon \( \tau_{env} \), and thus, the MDP grid world is assumed to be fixed and stationary for each \( \tau_{env} \) period of time.

As scenarios develop, the local rewards corresponding to each cell in the environment are evolved and updated, transitioning the vehicle into a new MDP. The reward function is updated based on both probabilities of goal-reaching objectives and safety violations. Depending on the risk level, which will be determined by the RAU module later, the safety state of each of the \( m \times n \) cells in the environment (i.e., the discretized finite portion of the highway around the ego vehicle) is categorized into the following discrete states \( Z = \{\text{un, sa, hr, lr, tg, cp}\} \) (see Fig. 3).

1) Unsafe \{un\}: This kind of state represents the places in the environment that represents failure (i.e., they are occupied with any type of obstacles) or will eventually fail.

2) Safe \{sa\}: This kind of state corresponds to the places in the environment that are not currently in failure states (i.e., unsafe), for which there exist control actions that
composed of a series of fixed local MDPs, i.e.,
of cells. Therefore, the state of each local MDP is limited to the number
the ground MDP change from one MDP to another but remain
states of MDP cells and, consequently, the reward functions of
leveraged by the planner to perform replanning. The safety
between local MDPs, as shown in Fig. 4, which are then
M
{1806 IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, VOL. 31, NO. 4, JULY 2023
as a stationary and fixed grid-world MDP. We now define more
τ
epoch, i.e.,
µ
k

2) The reward function
R
k

4) The local transition probabilities
P
k

5) Terminal Goal [tg]: This kind of state represents the terminal
reaching state in the environment. This kind of state represents the
current position of the AV in the environment.
6) Current Position [cp]: This kind of state represents the
current position of the AV in the environment.

We now formally define the ground MDP \( \mathcal{M} \), which is
composed of a series of fixed local MDPs, i.e., \( \mathcal{M} = \{\mu_1, \mu_2, \ldots, \mu_N\} \). The ground MDP captures the transition
between local MDPs, as shown in Fig. 4, which are then
leveraged by the planner to perform replanning. The safety
states of MPD cells and, consequently, the reward functions of
the ground MDP change from one MDP to another but remain
fixed for each local MDP from the planner’s perspective. Therefore, the state of each local MDP is limited to the number
of cells.

Definition 1 (Environment Ground MDP): The finite-state
MDP \( \mathcal{M} \) for the high-level planner consists of a four-tuple
\( \mathcal{M} = (S_G, \mathcal{E}, \mathcal{A}, \Pi) \), where the following holds.
1) \( S_G \) is the global state space for the ground MDP with
cardinality \( |S_G| = [6]^{m \times n} \), as each cell of each MDP
can take six different safety states.
2) \( \mathcal{E} \) is the set of transitory epochs in which local MDPs change, with \( N_T \leq +\infty \).
3) \( \mathcal{A} \) denotes a finite set of feasible high-level actions.
4) \( \Pi := \{\pi_k : k \in \mathcal{E}\} \) with \( \pi_k := \{a \in \mathcal{A} : l = 0, 1, \ldots\} \),
k \in \mathcal{E}, denotes a finite set of feasible policies, which
maps each global state \( S_k \in S_G \) to a sequence of risk-
averse high-level planned actions.

Based on Definition 1, between each consecutive transition
epoch, i.e., \( \tau_{env} \), the environment of AV (i.e., \( \mathcal{M} \)) can be seen
as a stationary and fixed grid-world MDP. We now define more
formally the environment space for the \( k \)th local MDP,
\( \mu_k \) over a time horizon \( (k + 1)\tau_{env} - k\tau_{env} \) from the high-level
planner’s perspective.

3) High Risk [hr]: This kind of state represents the places in
the environment for which there is a significant chance
that the safety mode cannot avoid crashing, i.e., all
trajectories starting from these states will end in failure
with significant probability.
4) Low Risk [lr]: This kind of state represents the places in
the environment for which safe mode can avoid crashing,
but some possible low-risk actions, such as hard braking
or sudden change of direction, are inevitable.
5) Current Position [cp]: This kind of state represents the
current position of the AV in the environment.

Note that, in the finite-state local MDP \( \mu_k \), \( S_k \) (resp. \( \mathcal{A} \))
is a Borel space (i.e., a Borel subset of a complete and
separable metric space) called the state space (resp. action
space). For each \( s \in S_k \) the measurable set \( \mathcal{A} \) denotes the set of
feasible high-level actions and has the property that the set of
feasible state–action pairs \( \{(s, a) : s \in S_k, a \in \mathcal{A}\} \) is a
measurable subset in \( S_k \times \mathcal{A} \). The canonical sample space is
defined as \( \Omega := (S_k \times \mathcal{A})^\infty \), and all random variables will be
defined on the measurable space \( (\Omega, \mathcal{B}) \), where \( \mathcal{B} \) denotes the
corresponding product σ-algebra.

As an example, one can represent the road ahead as an
\( m \times n \) grid world, where each square (i.e., cell) forward
represents the next second of travel, and the squares to
the left and right represent lane positions and the edges of the
road. For instance, if a vehicle is moving at 100 km/h
and each square forward represents 1 s of the travel, ten
squares represent about 0.35 km in the direction of the travel.
Therefore, the environment of vehicle can be modeled as a
2-D grid world with \( m \times n \) cells over a bounded rectangle
(see Fig. 2) in which the characteristics of each cell evolve in
time. Moreover, considering \( s \in S_k \) as the current state and
$s' \in S_k$ as the subsequent state after applying the action $a$, the rewards $R_k(s, s', a)$ for this example can be defined as follows.

1) Normal distribution with zero mean and very small variance for the high-level action $a$ that causes reaching safe zones, i.e., $s_a$.
2) Normal distribution with a mean of $M$ where $M \gg 0$ and very small variance for the high-level action $a$ that causes reaching unsafe zones, i.e., $s_u$.
3) For the high-level action $a$ that causes reaching a next cell state $s'$ with the safety status $s_a$ (i.e., it traverses to a high-risk zone); a random cost can be assigned with truncated normal distribution with the parent normal distribution with a mean of $M_{hr}$ where $M > M_{hr} \gg 0$.
4) For the high-level action $a$ that causes reaching a next cell state $s'$ with the safety status $s_u$ (i.e., it traverses to a low-risk zone); a random cost can be assigned with truncated normal distribution with the parent normal distribution with a mean of $M_{lr}$ where $M > M_{lr} > 0$.
5) Normal distribution with a mean of $\Gamma$ and very small variance where $-\Gamma \in \mathbb{R}_{>0}$ is chosen sufficiently large, for the high-level action $a$ that causes reaching a next cell state $s'$ with discrete state terminal goal, i.e., $tg$.

It is assumed here for simplicity that the transition probabilities $P_k(s' | s, a)$ of the local MDP are known beforehand and not changing over time, i.e., $P_k(s' | s, a) = P(s' | s, a)$ for $k = 1, \ldots, N_T$. Note that this assumption is reasonable since if the higher-level planner generates a high-level command to move from one waypoint to another one, the low-level controller will follow. The probability of transitions can be learned from the success rate of the low-level controller in performing its high-level actions. For example, moving the vehicle from one waypoint to another one when both are located in a straight line is easier for the low-level controller than moving to the left or right with angles more than 42°. The distributions of the reward functions $R_k$, however, change due to the change in the scenario caused by traffic conditions, road geometry, and so on, that is, a rewarding maneuver in the current situation can jeopardize the system’s safety as the scenario develops (e.g., traffic condition changes) and thus will not be rewarding anymore. Therefore, as MDPs change in the ground MDP, only their probabilistic reward functions are assumed to change.

The planner will learn a policy using stationary and fixed grid-world MDPs, which are called local MDP from now on. This name comes from the fact that the local MDPs of the ground MDP are constructed based on preview information, which can be seen as predicted forms of the actual environment. Now, we define the $k$th local MDP as $\mu_k : (S_k, A, P_k, R_k, \gamma)$, which is used by the high-level planner. To consider future changes in the actual environment that might lead to infeasibility of the planned trajectory, the FCU module is introduced to detect future infeasibility and update the planned trajectory by triggering a replanning request.

The objective of the high-level risk-averse planner is to learn the function $p|l(S_k) : S^G \mapsto \Pi$ by optimizing a risk-averse objective function with the effective horizon for the local MDP.

$\mu_k$, that is, the planner learns a sequence of high-level actions $a_0, a_1, \ldots$ for the local MDP $\mu_k$, providing the planning policy $\pi_k : k \in \mathcal{E}$.

B. RAU and Reward Design Using Preview Information

An RAU module is developed to design reward functions $R_k(s, s', a)$ based on $R^s_{S_k'}$ for the $k$th local MDP, i.e., $\mu_k$, $k \in \mathcal{E}$ as they change over time. To learn the reward function that makes a tradeoff between safety and performance for each scenario, three modules will be used: a probabilistic predictor of traffic participants (PPTP), a predictive stochastic reachability module, and a reward mapping module, as shown in Fig. 5. The PPTP learns the behavior of traffic participants based on traffic observations provided by sensors such as cameras and light detection and ranging (LiDAR). The approaches for PPTP include heuristics methods [53] and deep learning-based methods for trajectory prediction (e.g., using neural networks [54], [55] and Bayesian networks [56]), which have shown great performance in predicting the behavior of other traffic participants. Using this behavior prediction, which can be as brief as the probability of taking some high-level actions, PPTP predicts the probability of each MDP state being occupied by participants and, thus, its probability of being a failure state. However, penalizing possible failure states is not enough for safety guarantee since some states in the MDP are not failure states but are nonetheless high-risk ones in the way that there is a significant chance that the safety mode cannot avoid crashing. Besides the high-risk zones, there also exist low-risk zones in which safe mode can avoid crashing with high probability. Therefore, the stochastic reachability module receives the information of the PPTP module as well as the safety constraints, such as the lane boundaries and the vehicle’s stability limit. It then solves a constraint satisfaction problem over a future time horizon [57], [58], [59] to introduce probabilistic occupancy functions to reason about collision probability and characterize keep-out sets for probabilistic safety. Based on the predefined thresholds and the probability of violating safety, each cell then receives a safety state out of the six safety states in $\mathcal{Z}$ defined before with the corresponding distributions of risk (see Fig. 6). These reachable sets are updated in the receding horizon, as new information about the surrounding environment is previewed, e.g., a change in the road curvature or speed of a vehicle is previewed. The reward mapping submodule then utilizes the safety state categorization of cells and the corresponding distribution of risk, as the output of the stochastic reachability.
module, to assign high reward values for the goal-reaching objective and stochastic penalty values to the actions in each cell state $s$.

Note that the RAU module encodes both safety and performance specifications in the probabilistic reward functions $R_k$. It is shown in [60] that a big enough penalty function can assure that the optimal solution for the original task can be learned safely. Note also that previewing information is highly advantageous and will allow imagining the reward function ahead of time and thus generate samples for the proposed risk-averse $Q$-learning planner and solve it ahead of time before the scenario becomes apparent.

\section*{IV. Sampling-Based Convex Program for Risk-Averse Preview-Based Learning}
\subsection*{A. Risk-Averse $Q$-Learning for Planning}

Consider the local MDP $\mu_k$ defined based on Definition 2. A policy $\pi_k : S_k \mapsto A$ for the high-level planner is a mapping from MDP states $S_k$ to high-level actions $A$. Given the local MDP $\mu_k$, the risk-averse planner’s goal is to learn a sequence of high-level actions to reach a cell state with discrete states type terminal goal, i.e., tg, state from an initial state while optimizing a risk-informed cost function.

Given the finite set of cell states $s \in S_k$ and the set of actions $A$, which are generated by the policy $\pi_k$, and the reward function $R_k$, all defined in the local MDP $\mu_k$, an infinite-horizon discounted cost is first defined in the form of

$$\mathcal{L}^n(s_0) = \sum_{t=0}^{\infty} \gamma^t R_k^t.$$  \hspace{1cm} (1)

Note that even though the cost is defined over an infinite horizon for simplicity of the analysis and design, the effective horizon of the planning, after which the reward can be ignored, is finite due to the discount factor and depends on the value of the discount factor. Note also that safety and performance specifications are encoded in the reward functions $R_k$.

Since the probability transitions $P(s' \mid s, \pi_k(s))$ and the reward functions $R_k$ provided by the RAU are probabilistic, optimizing the expected value of the cost function (1) is not good enough since it can lead to performance fluctuation and even unsafe behaviors. To account for the risk or variance of the performance, which provides high-confidence safety assurance, the entropic-based risk-averse cost function $J$ is used as

$$J_{\pi_k}(s) = \frac{1}{\alpha} \log(\mathbb{E}[\exp(\alpha \mathcal{L}^n) \mid s_0 = s])$$  \hspace{1cm} (2)

where $\alpha \in \mathbb{R}_{>0}$ is the risk-averse factor, and the subscript $l$ is dropped for notational simplicity.

\textbf{Lemma 1:} Consider the infinite-horizon risk-averse cost function $J_{\pi_k}$ given in (2). Then,

$$J_{\pi_k}(s) = \mathbb{E}[\mathcal{L}^n(s)] + \frac{\alpha}{2} \mathbb{V}[\mathcal{L}^n(s)] + O(\alpha^2).$$  \hspace{1cm} (3)

\textbf{Proof:} It is well known that $\exp(f)$ and $\log(1 + f)$, respectively, have Maclaurin series $\exp(f) = 1 + f + f^2/2 + O(f^3)$ and $\log(1 + f) = f - f^2/2 + O(f^3)$, which converge for all $f$. Therefore, using Maclaurin series of $\exp(.)$ and $\log(.)$ and some manipulations yields

$$\log(\mathbb{E}[\exp(\alpha \mathcal{L}^n)]) = \alpha \mathbb{E}[\mathcal{L}^n] + \frac{\alpha^2}{2} \mathbb{V}[\mathcal{L}^n] + (\alpha^2/2)^2 + \ldots$$  \hspace{1cm} (4)

Expanding the squares, dividing both sides of (4) by $\alpha$, and using the same reasoning given in [61], we have

$$\frac{1}{\alpha} \log(\mathbb{E}[\exp(\alpha \mathcal{L}^n)]) = \mathbb{E}[\mathcal{L}^n] + (\alpha/2) \left( \mathbb{V}[\mathcal{L}^n] - (\mathbb{E}[\mathcal{L}^n])^2 \right) + O(\alpha^2)$$

which completes the proof. \hfill $\Box$

\textit{Remark 1:} It is worth noting that the dynamics of MDP are known only probabilistically, which models the aleatory noise that cannot be reduced as more data are collected. The probabilistic reward functions learned based on the preview information characterize the epistemic uncertainty, which can be determined with more certainty as more rich data are available. In this article, we assume that the perception system is not under any drift [63] in the performance of perception-based systems, and the uncertainties are caused by unknown behavior of other participants and randomness in traversing between the waypoints.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
B. Risk-Averse Preview-Based Q-learning

For a given policy \( \pi_k \) and for the cost (2), define the value function as

\[
V^{\pi_k}(s) := J_{\pi_k}(s). \tag{7}
\]

The Q-function associated with the action \( a \in A \) at state \( s \in S_k \) and following the policy \( \pi_k \) afterward is defined as

\[
Q^{\pi_k}(s, a) := g^k_a(s) + \frac{1}{\alpha} \log \left( \mathbb{E}_{s', a' \sim P_{s,a}} \left[ \exp (a' \gamma V^{\pi_k}(s')) \right] \right), \tag{8}
\]

where \( s'_k \) is the realization of the next state starting from the state \( s \) and applying the action \( a \) and

\[
g^k_a(s) := \frac{1}{\alpha} \log \left( \mathbb{E}_{s' \sim P_{s,a}} \left[ \exp (a \gamma \bar{R}_k(s, s', a)) \right] \right) \tag{9}
\]

\[
\tilde{P}^a_{s,a} = \tilde{P}(s' \mid s, a) \quad \text{and} \quad V^{\pi_k}(s) = Q^{\pi_k}(s, \pi_k(s)). \quad \text{Defining the optimal Q-function} \quad Q^*_k(s, a) := Q^{\pi^*_k}(s, a) \quad \text{the optimal Bellman equation}
\]

\[
Q^*_k(s, a) := g^k_a(s) + \frac{1}{\alpha} \log \left( \mathbb{E}_{s' \sim P_{s,a}} \left[ \exp (a \gamma V^*_k(s')) \right] \right) \tag{10}
\]

where \( a' \) denotes the next action under the optimal policy \( \pi^*_k \) and \( V^*_k = \inf_{\pi \in \Pi} V^{\pi_k} = \inf_{a \in A} Q^*_k(s, a) \). The optimal control policy is then given by

\[
\pi^*_k(s) = \arg \min_{a \in A} Q^*_k(s, a). \tag{11}
\]

Using the optimal Q-risk Bellman operator \( \mathbb{T} \) as

\[
\mathbb{T} Q^*_k(s, a) := g^k_a(s) + \frac{1}{\alpha} \log \left( \mathbb{E}_{s' \sim \tilde{P}_{s,a}} \left[ \exp (a \gamma \inf_{a' \in A} Q^*_k(s', a')) \right] \right) \tag{12}
\]

the optimal Bellman equation becomes

\[
Q^*_k(s, a) = \mathbb{T} Q^*_k(s, a). \tag{13}
\]

We use the following definitions in the sequel.

**Definition 3 [64]:** Consider a weight function \( w : X \mapsto \mathbb{R} \) with \( w(x) > 0 \), \( \forall x \in X \), where \( X \) is a finite space. Let \( \mathcal{C}(X) \) be the space of real-valued functions \( f \) on \( X \). The weighted sup-norm is defined as

\[
\|f\|_w = \sup_{x \in X} \frac{|f(x)|}{w(x)} \tag{14}
\]

on \( \mathcal{C}(X) \).

**Definition 4 [64]:** An operator \( \mathbb{T} : \mathcal{C}(X) \mapsto \mathcal{C}(X) \) is monotone if

\[
(\mathbb{T} f_1 - \mathbb{T} f_2, f_1 - f_2) \geq 0, \quad \forall f_1, f_2 \in \mathcal{C}(X). \tag{15}
\]

Now, to compute the fixed point of (13), a convex optimization approach is developed inspired by [65]. The presented convex optimization approach relies on the Bellman inequality version of (13), i.e.,

\[
Q_k(s, a) \leq \mathbb{T} Q_k(s, a) \tag{16}
\]

which implies that \( Q_k \leq Q^*_k \) because of the optimal Bellman equation (13) and the monotonicity and contractivity properties. Therefore, to learn the optimal solution to (13) (i.e., the fixed point of \( \mathbb{T} \), one can search for the greatest \( Q_k \) that satisfies \( Q_k \leq \mathbb{T} Q_k \), that is, the following optimization problem solves the Bellman equation (13):

\[
\sup_{Q_k \in \mathcal{C}(S_k \times A)} \sum_{s_k \times A} c(s,a) Q_k(s,a) \leq \frac{1}{\alpha} \log \left( \mathbb{E}_{s' \sim P_{s,a}} \left[ \exp (a \gamma \inf_{a' \in A} Q_k(s', a')) \right] \right) \tag{17}
\]

where \( c(\cdot, \cdot) \) is a nonnegative measure (e.g., probability measure [65]) with finite moments that assigns nonnegative mass to all subsets of \( S_k \times A \). It should be noted that \( c(\cdot, \cdot) \) is typically used as a probability measure [65], [66]. For example, a Gaussian distribution can be used when the state space is unbounded, while a truncated normal distribution or uniform distribution can be used when it is compact. The infimum operator in the optimal Bellman operator \( \mathbb{T} \) makes (17) hard to solve. It is common practice to remove the infimum operator and instead relax \( a' \) in the constraint (17) to all \( a' \in A \). This makes the problem convex at the cost of increasing the dimension of constraint. Therefore, instead of using the optimal Bellman operator \( \mathbb{T} \) and directly solving (17), we use an equivalent convex program [will be given later in (25)] by using the following easier-to-solve Bellman operator, which is developed by dropping the infimum operator in \( \mathbb{T} \):

\[
\hat{Q}_k(s, a) := g^k_a(s) + \frac{1}{\alpha} \log \left( \mathbb{E}_{s' \sim \tilde{P}_{s,a}} \left[ \exp (a \gamma Q_k(s', a')) \right] \right). \tag{18}
\]

**Remark 2:** Note that the standard policy iteration algorithm cannot be applied to solve the risk-aware optimal control problem (17). This is because, in contrast to their risk-neutral counterparts, the risk-aware Bellman operator (12) is a nonlinear operator, while the standard policy iteration significantly relies on the linearity of Bellman equations. In contrast, the standard value iteration algorithm solves the Bellman equation recursively as

\[
Q_{k+1}(s, a) = g^k_a(s) + \frac{1}{\alpha} \log \left( \mathbb{E}_{s' \sim \tilde{P}_{s,a}} \left[ \exp (a \gamma Q_k(s', a')) \right] \right). \tag{19}
\]

Since the \( Q \)-function is for the previous iteration on the right-hand side and for the current iteration on the left-hand side, the nonlinearity of the right-hand side does not cause any issue here, that is, this equation is a recursion rather than a Lyapunov equation and can be easily solved for a nonlinear operator. Therefore, the standard value iteration can also be used to solve the formulated risk-averse problem. Nevertheless, standard value iteration is an iterative approach, and theoretically, it takes an infinite number of iterations for it to converge [67]. In contrast, the presented algorithm here is a one-shot optimization algorithm that needs to be
solved only once. Since the infimum operation is removed in (18), furthermore, the risk-averse optimal control problem (17) with the relaxed Bellman operator in (18) can be solved more efficiently than (12), which is used in standard policy iteration or value iteration algorithms. Besides, to learn a risk-averse planning solution for an environment that evolves as a series of MDPs, collecting data for every new MDP is required, which is time-consuming and not practical since a plan must be made before the decision horizon is finished. This is especially true for autonomous driving for which scenarios keep changing, which in turn changes the MDP model. The following presents an optimization-based preview-based $Q$-learning that leverages the Bellman operator (18) to solve the optimal planning problem in a tractable manner by resolving both nonlinearity and nonstationarity issues. The algorithm uses preview information collected by sensors, providing cues about the scenario (e.g., traffic situation and road geometry), to generate samples of environment interaction without any actual interaction and leverage these samples to generate an optimal plan by solving a static optimization problem before actually, the scenario becomes apparent.

Since the presented sampling-based preview-based learning relies on the operator $\hat{T}$, it is shown in the following proposition and theorem show that this operator is monotone and contractive.

**Proposition 1:** The Bellman operator $\hat{T}$ in (18) is monotone.

**Proof:** See Appendix A.

We now show that the Bellman operator with a reward function that is not necessarily bounded by a constant is a contraction mapping operator with respect to the weighted norm defined in Definition 3.

**Assumption 1:** There exists a nondecreasing function $\varphi : \mathbb{R} \mapsto [1, \infty)$ such that $\varphi(s, a, a') \in S_k \times A^2$

\[
\sup_{a' \in A} \mathbb{E}_{s' \sim p_{s,a}} \{w(\varphi, a')\} \leq \gamma \varphi(s, a) \tag{20}
\]

for some constant $\gamma \in (0, 1/\alpha)$.

**Theorem 1:** Under Assumption 1, the Bellman operator in (18) is a contraction mapping operator with respect to the weighted norm $w$, i.e.,

\[
\|\hat{T}Q_k - \hat{T}Q^*_k\|_w \leq \gamma \|Q_k - Q^*_k\|_w \tag{21}
\]

\(\forall Q_k, Q^*_k \in C(S_k \times A)\), where $\gamma \leq 1$.

**Proof:** See Appendix B.

**Proposition 2:** Let $Q_k \in C(S_k \times A)$ be an arbitrary $Q$-function. Then, the following property holds:

\[
Q_k = \lim_{l \rightarrow \infty} \hat{T}_l Q_k. \tag{22}
\]

**Proof:** Proposition 1 and Theorem 1 show that the Bellman operator satisfies monotonicity and contraction properties. Therefore,

\[
\|\hat{T}_l Q_k - \hat{T}_l Q^*_k\|_w = \|\hat{T}_{l-1} Q_k - \hat{T}_{l-1} Q^*_k\|_w \leq \gamma \|\hat{T}_{l-1} Q_k - \hat{T}_{l-1} Q^*_k\|_w \leq \cdots \leq (\gamma \gamma')^l \|Q_k - Q^*_k\|_w. \tag{23}
\]

Now, invoking Banach’s theorem [68] implies that there exists a unique fixed-point solution $Q^*_k$ of the Bellman operator. This completes the proof.

**Proposition 3:** A $Q$-function that satisfies the Bellman inequality $Q_k \leq \hat{T}_l Q_k$ provides a lower bound to $Q^*_k$.

**Proof:** Considering $Q_k \leq \hat{T}_l Q_k$ and applying the operator $\hat{T}$ on both sides repeatedly, using the results of Proposition 2 yields

\[
Q_k \leq \hat{T}_l Q_k \leq \hat{T}_l^2 Q_k \ldots \leq \hat{T}_l^\infty Q_k = \lim_{l \rightarrow \infty} \hat{T}_l Q_k = Q^*_k. \tag{24}
\]

This completes the proof.

This proposition is used to develop a preview-based convex problem for learning the risk-averse planning solution using only preview information in a sampling-based optimization framework, that is, the following preview-based convex optimization is first presented, and its convergence to the fixed point of $\hat{T}$ operator (18) is also shown. Since solving the actual optimization problem can be computationally demanding, a sampling-based finite-horizon convex optimization is then presented, for which the samples are generated using the preview information obtained from the RAU module, to learn a near-optimal solution.

**Problem 2 (Preview-Based Convex Program):**

\[
\begin{align*}
\sup_{Q_k \in C(S_k \times A)} \sum_{s_k \in S_k \times A} c(s, a)Q_k(s, a) \\
Q_k(s, a) \leq g_k(s) \\
+ \frac{1}{\alpha} \log \left(\mathbb{E}_{s' \sim p_{s,a}}[\exp(\alpha g_k(s', a'))]\right) \\
\forall (s, a, a') \in S_k \times A^2.
\end{align*} \tag{25}
\]

**Corollary 1:** The solution to the optimization problem (25) coincides with the optimal value function $Q^*_k$ in (13).

**Proof:** See Appendix C.

After finding the approximate optimal risk-averse state–action function $\hat{Q}^*_k$ from solving Problem 2, one can find the greedy approximate optimal control policy by solving

\[
\pi^*_k(s) = \arg \min_{a \in A} \{\hat{Q}^*_k(s, a)\}. \tag{26}
\]

Note that the size of the decision variables is finite and is equal to the number of state–action pairs. The number of constraints is also finite since the numbers of states and actions of MDP are finite. However, to avoid requiring any online interaction to form the constraints, and to avoid computational intractability, the preview reward function as the output of the RAU module is leveraged to generate samples. Note that the transition probabilities for the high-level planner are known and depend on the difficulty of the low-level controller in executing and traversing between waypoints. Therefore, optimization is formed using only preview information and solved for the MDP ahead before the scenario actually becomes apparent. Algorithm 1 provides the details of the risk-averse preview-based $Q$-learning algorithm.

**Remark 3:** Even though not considered in this article, it has the potential to leverage transfer learning based on similarity measurement developed for MDPs [69], [70], [71] to quickly...
Algorithm 1: One-Shot Preview-Based Risk-Averse $Q$-Learning Algorithm

Result: The approximate optimal risk-averse state-action values $Q_k$.

Data collection:
for $i = 1, \ldots, |S_k|$ do
    for $j = 1, \ldots, |A|$ do
      for $l = 1, \ldots, N_l$ do
        Obtain $(s_i, a_j, s'_{l,i}, r_{l,i}, g_{l,i}(s_i))$
        $S'_{l,i} := \{s'_{l,i}, l = 1, \ldots, N_l\}$
        $R_{l,i} := \{R_{l,i}(s_i, s'_{l,i}, a_j), l = 1, \ldots, N_l\}$
        Create $\mathcal{D}^k := \{(s_i, a_j, s'_{l,i}, r_{l,i}, g_{l,i}(s_i))\}_{i=1,j=1}^{|S_k|,|A|}$
      Select $\{(s_i, a_j, s'_{l,i}, r_{l,i}, g_{l,i}(s_i))\} \in \mathcal{D}^k$
      Select $a'_z \in A$
      Create $\mathcal{D}^k := \{(s_i, a_j, s'_{l,i}, r_{l,i}, g_{l,i}(s_i), a'_z)\}_{i=1,j=1}^{|S_k|,|A|}$

Function evaluation:
Solve
\[
\sup_{Q \in C(S_k, A)} \sum_{s,a} c(s,a) Q(s,a)
\]
\[
Q(s_i, a_j) \leq g_{l,i}(s_i) + \frac{1}{\alpha} \log \left( \mathbb{E}_{s_{l,i}, a_j \sim P_{s_{l,i}}} \exp(\alpha Q(s_i', a_j')) \right)
\]
\[\forall(s_i, a_j, s'_{l,i}, r_{l,i}, g_{l,i}(s_i), a'_z) \in \mathcal{D}^k\]

V. FEASIBILITY CHECK UNIT

Upon planning high-level actions for the next $n \tau_{\text{env}}$ seconds at the transition epoch $k$ by the high-level $Q$-learning planner based on the local MDP $\mu_k$, a desired continuous-time reference trajectory for low-level control to track is computed for the next $n \tau_{\text{env}}$ seconds. To continuously monitor the feasibility of the generated desired reference trajectory for the next $n \tau_{\text{env}}$ seconds of the travel, which can be jeopardized due to the change in environment, an FCU module is utilized. If the FCU module infers that feasibility violation is imminent at any time in the next $n \tau_{\text{env}}$ seconds with a probability above a threshold, it triggers a replanning request (see Fig. 7). Replanning can be reactive to just get around infeasible areas and proactive by triggering the preview-based risk-averse $Q$-learning with new information to relearn a new temporally risk-averse plan.

Since the FCU module needs to consider both continuous-time transitions (low-level controller outcome) of the AV and discrete-time stochastic transitions of the environment MDP $\mathcal{M}$, a hybrid automaton is used to formalize the characteristics of the FCU module.

Before formally defining the hybrid automata, let us first recall and define some update functions acting on different global states to produce different layers of discrete transitions based on the environment ground MDP $\mathcal{M}$. Recall that the planner function $pl(.)$ maps each global state $S_k \in S_G$ in the ground MDP $\mathcal{M}$ to a sequence of risk-averse high-level planned actions $a \in A$. We now define the following functions $\mathbf{AE}(.)$ and $\mathbf{EN}(.)$ that predict the changes in MDPs based on a change in the actions of the AV itself and a change in the surrounding environment, respectively. These functions are only defined to define the hybrid automata and the presented approach does not need to learn them or know them.

1) The update function $\mathbf{AE}(.)$ maps each global state $S_k \in S_G$ in the ground MDP $\mathcal{M}$ to a sequence of risk-averse high-level planned actions $a \in A$ based on the planner function $pl(.)$.
2) The update function $\mathbf{EN}(.)$ captures the evolution of local MDPs within the ground MDP $\mathcal{M}$ for a future finite-time horizon based on environmental changes.

Note that, based on the $pl(.)$ function and environmental changes, the $\mathbf{EN}(.)$ and $\mathbf{AE}(.)$ functions predict and calculate the environment behavior and conditions, such as new locations of the other vehicles for a future finite-time horizon in the vision horizon of the AV and also predict and calculate the corresponding safety state of zones, new terminal goal, and position of the AV. For notational simplicity in the sequel, the subscript $k$ is dropped when it is clear from the context.

1) The update function $\mathbf{AE}(S_k, pl(S_k)) : S_G \times A \mapsto S_G$
2) The update function $\mathbf{EN}(S_k) : S_G \mapsto S_G$

The hybrid automaton is now defined as a four-tuple $\mathcal{H} = (Q, \Theta, \Pi, T)$, where $Q = S_G \times S_p \times \Delta$ is the hybrid state space of the automaton, $\Pi$ denotes the set of all possible planners, $\Theta$ is the initial state, and $T$ is a set of continuous-time transitions.
trajectories. Moreover, $S_o$ is the set of vehicle states from the low-level controller perspective, which can change continuously, and $\Delta := \{+, -, \}$, where $+$, indicating that a sequence of planned high-level actions is not feasible anymore as scenarios develop and $-$, indicating that a sequence of high-level actions is feasible. Let $q \in Q$ denote a hybrid automaton system state. We denote the components of $q$ as $q \cdot S_o, q \cdot S, q \cdot D$. Moreover, $\Theta \subseteq Q$ is the set of initial states. A trajectory $\xi \in T$ is a map from $Q$ to $Q$ for a finite-time interval $t \in \text{dom}(\xi)$ based on the corresponding high-level planned actions. For the entire time period of the trajectory $\xi_l(k) \in T$, i.e., $t \in \text{env}$, the global environment state $S \in S_G$ remains unchanged. That is, $\forall t \in \text{env}$, $\xi_l(k) \cdot S = \xi_l(k)0 \cdot S$. However, $\xi_l(k) \cdot S_o$ changes continuously with time and is based on the interaction of the AV and low-level controller dynamics. For the sake of simplicity, in this article, $\xi_l(k) \cdot S_o$ is assumed to be a solution of a known differential equation (i.e., the autonomous overall low-level closed-loop system) as follows:

$$\frac{d}{dt}(\xi_l(k) \cdot S_o) = f_{q_0}a(\xi_l(k) \cdot S_o), \quad q_0 \in Q. \quad (30)$$

**Assumption 2:** Let the high-level planner action $a \in A$ command the vehicle to move from the cell state $s \in S_o$ to $s' \in S_o$. Then, the AV has a low-level controller that can track a trajectory $\xi_l \in T$ in finite time $\tau_{env}$ such that $\xi_l(k)0 \cdot S_o = s$ and $\xi_l(k) \cdot S_o = s'$.

Three types of discrete transitions are considered: environment transition, risk-averse planner transition, and replanning transition, denoted by the finite-state set $\mathcal{D}_pl = \{\text{env, rpl, pl}\}$. Let $D \subseteq Q \times D_{pl} \times Q$ denote the set of discrete transitions. Let $q \in Q$ and $q' \in Q$ denote a state and the poststate of the transition updates of the hybrid automaton system $\mathcal{H}$. Let $\text{Trans}(q, \ell, q')$ be a discrete transition map between $q$ and $q'$, where $\ell \in \mathcal{D}_{pl}$. Then, these discrete transitions are defined as follows.

1) **Environment Transition**, $\ell = \text{env}$: If the time elapsed since the last environment transition is equal to $\tau_{env}$, then $\ell = \text{env}$ and the environment transition is enabled, i.e., $\text{Trans}(q, \ell, q') \in D$, and as a result, the discrete state of the environment set is updated according to the environment update function $\text{En}(.)$, i.e., $q \cdot S_o \Rightarrow q' \cdot S_o$ and $q' \cdot S_o = q \cdot S_o$. Moreover, the agent (i.e., the low-level closed-loop system) and the current environment states get updated according to the joint AV/environment update function $\text{AE}(S, pl(S_o))$, i.e., $(q' \cdot S_o, q \cdot S_o) = \text{AE}(q \cdot S_o, q \cdot S_o)$.

2) **Risk-Averse Planning Transition**, $\ell = \text{pl}$: If, at the state $q$, the time elapsed since the last planning transition is equal to $\tau_{pl} = n \tau_{env}$ and $\Delta = -$, then $\ell = \text{pl}$ and the risk-averse planner transition is enabled. As a result, the risk-averse planner $Q$-learning $pl(.)$ gets updated by the risk-averse planner based on the preview probabilistic reward information provided by the RAU module.

3) **Replanning Transition**, $\ell = \text{rpl}$: If, at the state $q$, $q \cdot S_o$ stays optimal risk-averse during actual operation. It is also worth mentioning that the design parameter $n$ is related to the computational burden of the developed algorithm. For example, for $n = 1$ (i.e., very short future time horizon), we need to replan very fast with the frequency of $1/\tau_{env}$, which increases our computational cost, and if one chooses $n \gg 1$ to decrease the computational cost, then we might lose some optimality. However, for most traffic scenarios, the flow of traffic is smooth enough (e.g., drivers do not do sudden lane changing or break), and to decrease the computation burden, $n$ can be reasonably increased.

**Remark 5:** It is worth noting that Vallon and Borrelli [72] provided a data-driven hierarchical learning architecture for predictive control in unknown environments. Vallon and Borrelli [72] first designed the target sets from past data and
then incorporated them into a model predictive control (MPC) scheme with a shifting horizon that ensures the safety of the closed-loop system when performing the new task. Even though adaptation to the situations for MPC is considered in [72], which is promising, in contrast to this work, the constraints are adapted in contrast to the reward function. Risk-averse MPC is also reported in [73], [74], [75], [76], and [77]. An online learning-based stochastic MPC framework is proposed in [73], in which CVaR constraints on system states are required to hold for a family of distributions called uncertainty sets. Based on generic cost functions, the author derives Lyapunov-type risk-averse stability conditions for MPC problems with constrained nonlinear Markovian switching systems in [74]. A stochastic nonlinear MPC approach was developed in [75] in order to systematically find an optimal decision for dynamical systems involving uncertainty. An MPC framework for linear systems with multiplicative uncertainty is provided in [76]. While MPC results in a nonconvex optimization problem for nonlinear systems and/or nonquadratic costs (which are the cases of the risk-aware cost), our risk-aware approach results in a convex optimization approach. Moreover, MPC with a fixed cost function cannot guarantee a viable solution in ever-changing environments for which the reward function must be adjusted to the situation to trade off safety and optimality.

VI. SIMULATION

As case studies to show the validity of the proposed risk-averse planner scheme, two traffic scenario examples are considered.

Example 1: In the first example, which is the highway driving of an AV in a varying traffic density, the main objective of the AV (so-called ego vehicle) is to avoid unsafe and risky zones as they appear while optimizing a goal-reaching objective. In this scenario, for simplicity, we assume that the preview information is just the positions of the ego vehicle and other vehicles are shown by blue dashed lines.

The model dynamics of ego vehicle is chosen as a single-track bicycle model vehicle given by

\[
\begin{bmatrix}
\dot{Y} \\
\dot{\Psi} \\
\dot{\alpha}_T \\
\dot{\Psi}
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & K & -m_T v_T & v_T \\
0 & 0 & 0 & 1 \\
0 & 0 & -2K & m_T v_T \\
0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\delta \\
\Psi \\
\alpha_T \\
\Psi
\end{bmatrix}
\]

where \( Y(t) \) and \( \delta(t) \) denote the vertical position and steering angle of the controlled vehicle along its lane, respectively. Moreover, \( \Psi \) and \( \Psi \) denote the yaw and yaw rate, respectively, \( \alpha_T \) is the vehicle sideslip angle, \( \dot{X} = v_T + X(0) \) and \( \dot{v}_T = 0 \), where \( X \) and \( v_T \) are the longitudinal positions of the controlled vehicle and vehicle velocity, respectively. Other vehicle parameters are listed in Table I.

The ego-vehicle state vector is \( S_e = \{ X, Y, \Psi, \alpha, \dot{\Psi} \} \in \mathbb{R}^5 \), and its observable state vector is \( S_o = \{ X, Y \} \in \mathbb{R}^2 \) and its control input is \( \delta \) steering angle.

For the low-level controller, a linear quadratic regulator (LQR) state feedback controller is utilized. To this end, given a planned trajectory \( T_p(t) \) based on the planned waypoints \( (X_p, Y_p) \in T_p(t) \) provided by high-level risk-averse \( Q \)-learning planner, the main objective of the low-level LQR state feedback controller is to follow \( T_p(t) \) and stabilize the ego-vehicle yaw, yaw rate, and side slip angle states, i.e., \( s_e = [\Psi, \alpha, \dot{\Psi}]' \), that is, \( e(t) := \{(Y - Y_p, \Psi, \alpha, \dot{\Psi})'\}^2 \rightarrow 0 \). To this aim, the LQR performance index is chosen as \( \int_0^\infty (e^T Q e + \delta^T R \delta) dt \), where \( Q = \text{diag}(3, 1, 1, 1) \) and \( R = 1 \). Note that \( X - X_p \) is removed from \( e(t) \) since \( v_T = 0 \) and the low-level controller controls the lateral dynamics of the ego vehicle.

The ego vehicle and vehicles 2–4 speeds are 16.7 m/s, 2.6 m/s, –2.6 m/s, and –2.6 m/s, respectively. In performing closed-loop experiments, the centers of gravity of ego vehicle and other vehicles are initialized at \( (X(0), Y(0)) = (0, 1) \) m, \( (X_1(0), Y_1(0)) = (28, –4) \) m, \( (X_2(0), Y_2(0)) = (56, 0) \) m, \( (X_3(0), Y_3(0)) = (164.5, 6) \) m, and \( (X_4(0), Y_4(0)) = (157.5, –6) \) m. Fig. 8 represents a three-lane highway in which the positions of the ego vehicle and other vehicles are shown with blue and red squares, respectively.

The width and length of the highway are 22 and 203 m, respectively. We assume that the preview information is just available for the next eight seconds of travel. The environment is discretized and modeled as a 2-D 11 × 39 grid with a total of 429 cells with a width of 2 m where each square forward represents the next 0.2 s of the travel (i.e., length of 3.5 m since the ego vehicle traveling with the constant velocity of 16.75 m/s) and cells can have different characteristics. The dimensions of the ego vehicle are considered as 2 × 3.5 m,
Fig. 9. Three-lane highway represented by $11 \times 39$ grid. Obstacles are red. Ego vehicle is blue. Lanes are shown by blue dashed lines. The green shaded area and green dashed line represent the results between $[10\%, 90\%]$ quantiles and the means across ten independent experiments, respectively.

Fig. 10. Risk-neutral case.

and the dimensions of the other vehicles are chosen as $4 \times 7$ m. Each of these cells in the environment can be in one of the five discrete states $Z = \{ \text{un, sa, hr, lr, tg, cp} \}$. Moreover, for simplicity, it is assumed that the evolution of stochastic occupancy of the road by other vehicles is available, i.e., the risk probabilities of the cells. The edges of the highway occupy one square, and each of the three lanes occupies three squares. The finite set of high-level actions is $A = \{ \text{go straight, left turn, right turn} \}$.

The simulation time is 12 s, all the computations are carried out on an i7 1.9 GHz Intel core processor with 8 GB of RAM running MATLAB R2021a, and the following values are used: $\gamma = 0.3$, $\tau_{\text{env}} \simeq 0.2$ Sec, $n = 39$, and $N_k = 1500$, $\tau = 15$, $M = -10\,000$, $\tau_l = 10\,000$, $\tau_h = 0$, $\sigma = 1$, $\bar{\tau}_l = 10$, $\bar{\tau}_h = 0$, $\bar{\sigma} = 1$, and $\alpha = 0.2$.

The trajectories of $X$ and $Y$ positions of the ego vehicle for both risk-neutral (i.e., $\alpha = 0$) and risk-averse (i.e., $\alpha > 0$) cases are shown in Figs. 9 and 10. Fig. 9 represents two different time segments of the simulation, $[0, 4]$ and $[4, 12]$ s. The shaded green area and dashed green line represent the results of the low-level continuous-time trajectories based on the high-level actions toward the terminal goal between $[10\%, 90\%]$ quantiles and the means across ten independent experiments. The successive frames of the ego vehicle are shown by the blue square to represent the current position of the ego vehicle and the transparent cyan squares to denote some of the planned estimated positions. Moreover, the current positions of the other vehicles are shown with red squares, and high-risk zones and low-risk zones are shown with transparent red and yellow squares, respectively, in the grids. The variance of the planning position is significantly lower for the risk-averse case as expected. Note that at the time of 4 s, i.e., when ego vehicle is on $X = 67.05$ m and $Y = 7$ m position (see Fig. 10), the FCU module detects the infeasibility issue and initiates new planning through the high-level risk-averse $Q$-learning Algorithm 1.

**Example 2:** In the second example, which is the overtaking scenario (which is adopted from [78]), the main objective of ego vehicle is to safely overtake a bicycle before reaching a T-intersection, where another car is approaching. Using $s$ to
represents the volumetric center of the vehicle along a path, \( v \) to represent velocity, and \( a \) to represent absolute acceleration, the longitudinal dynamics can be described as follows:

\[
\dot{s} = v, \quad \dot{v} = \begin{cases} 
5 \cdot \left(1 - \frac{v}{20}\right) \cdot u, & u > 0 \\
5 \cdot u, & u \leq 0 \\
0, & v \leq 0
\end{cases}
\]  

subject to the constraint \( a \leq 5 \) where \( a = (a_N^2 + a_T^2)^{1/2} \), \( a_N = v^2 / \rho(s) \), \( a_T = \dot{v} \), \( \rho(s) \) maps the path coordinate \( s \) to its radius of curvature, and \( a_N \) and \( a_T \), respectively, are the normal and tangential accelerations. We assume that the preview information is just available for the next 3 s of travel. The dimensions of the ego vehicle are considered as \( 2 \times 3.5 \) m, and the dimensions of the bicycle and other vehicle chosen as \( 1 \times 2 \) m and \( 2 \times 2 \) m, respectively. The environment of the ego vehicle is discretized according to [78]. The stochastic occupancy of the road by other vehicle is computed using a probabilistic reachability analysis presented in [78], which is online computationally efficient since most of the intensive computations are performed offline. Using this scenario, it is checked whether an ego car can safely overtake a bicycle, while another car approaches a T-intersection. Four time segments of the simulation along with the risky zones (result from stochastic reachable submodule) of the traffic participants are given in Fig. 11. The simulation time is 10 s, and the following values are used: \( \gamma = 0.22 \), \( \tau_{\text{env}} = 0.45 \) Sec, \( v(0) = 5 \) m/s, \( N_k = 1000 \), \( \tau = 10 \), and \( \alpha = 0.25 \).

As mentioned, high-risk cells indicate high probability risk of collision, whereas low-risk cells represent areas of low probability risk of collision. It is worth noting that when the FCU module detects the infeasibility issue at 3.68 s, it initiates new planning by applying a high-level risk-averse Q-learning Algorithm 1. Using the parallel computing toolbox (i.e., parfor), the total computational time is 1.719 s, and the summation of computational times of the RAU and FCU modules is 0.581 and 0.306 s, respectively. With a similar setup, the total computational time of the method given in [78] is 1.481 s, which indicates that it is less computationally demanding. However, the path provided in [78] is not optimal, and also, the proposed framework in this article has the potential to provide us with the ability to transfer knowledge to other circumstances (e.g., by the similar idea provided in [79] and [72]), which can decrease the computational burden significantly.

VII. CONCLUSION

A risk-averse planner is developed for navigation of AVs between lanes around moving vehicles. The multilane road ahead of a vehicle is represented by a finite-state MDP and the entire planning horizon is modeled as a series of MDPs that evolve over time. An RAU module is leveraged to assess and preview the risk of each state and the reward function. A static convex program-based preview-based Q learning algorithm is then developed that leverages these probabilistic reward values to learn risk-averse optimal planning strategies. Finally, the developed scheme is implemented in some driving scenarios to show the efficiency of the proposed learning framework. Using state-of-the-art vehicle simulators, such as CARLA, we intend to investigate this article’s approach with the ability of transferring knowledge in more depth in the future.

APPENDIX A

PROOF OF PROPOSITION 1

To show the monotonicity property for \( \hat{T} \), one needs to show that for any \( Q^k_1(s, a) \) and \( Q^j_1(s, a) \), if \( Q^k_1(s, a) \leq Q^j_1(s, a) \), \( \forall(s, a) \in S_k \times A \), then \( \hat{T} Q^k_1(s, a) \leq \hat{T} Q^j_1(s, a) \), \( \forall(s, a, a') \in S_k \times A^2 \). Now, consider \( Q^k_1(s, a) \) and \( Q^j_1(s, a) \). Let \( s' \) be the next state and \( a \) be the current state. Since \( \log(.) \) and \( \exp(.) \) are injective relations (i.e., one-to-one and left total) on \( \mathbb{R}_{>0} \), it follows from \( Q^k_1(s', a') \leq Q^j_1(s', a') \) that:

\[
\frac{1}{\alpha} \left( \log \mathbb{E}_{s' \sim P_{s', a'}} \left[ \exp \left( \alpha g_k^s(s', a') \right) \right] \right) \leq \frac{1}{\alpha} \left( \log \mathbb{E}_{s' \sim P_{s', a'}} \left[ \exp \left( \alpha g_j^s(s', a') \right) \right] \right). 
\]

Adding \( g_k^s(s) \) to both sides yields:

\[
g_k^s(s) + \frac{1}{\alpha} \left( \log \mathbb{E}_{s' \sim P_{s', a'}} \left[ \exp \left( \alpha g_k^s(s', a') \right) \right] \right) \leq g_j^s(s) + \frac{1}{\alpha} \left( \log \mathbb{E}_{s' \sim P_{s', a'}} \left[ \exp \left( \alpha g_j^s(s', a') \right) \right] \right) \]

which implies that \( \hat{T} Q^1_1(s, a) \leq \hat{T} Q^j_1(s, a) \), \( \forall(s, a, a') \in S_k \times A^2 \). Therefore,

\[
\left( \hat{T} Q^1_1(s, a) - \hat{T} Q^j_1(s, a) \right) \geq 0 \quad (35)
\]
∀Q^1_k, Q^2_k \in C(S_k \times A), which satisfies the condition of Definition 4. This completes the proof.

**APPENDIX B**

**PROOF OF THEOREM 1**

Before proceeding with the proof, note that using Definition 3, one has

\[
\alpha \gamma (Q^2_k(s, a) - Q^1_k(s, a)) \leq \alpha \gamma \|Q^1_k(s, a) - Q^2_k(s, a)\|_w w(s, a)
\]

\[
\Rightarrow \alpha \gamma Q^2_k(s, a) - \alpha \gamma Q^1_k(s, a) - \alpha \gamma \|Q^1_k(s, a) - Q^2_k(s, a)\|_w w(s, a) \leq \alpha \gamma Q^1_k(s, a)
\]

\[
\log E_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma Q^1_k(s, a))]
\]

\[
- \log E_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma Q^2_k(s, a))]
\]

\[
\leq \frac{-1}{\alpha} \log E_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma Q^2_k(s, a))]
\]

\[
\Rightarrow \frac{-1}{\alpha} \log E_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma Q^2_k(s, a))]
\]

\[
\geq \frac{-1}{\alpha} \log E_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma Q^1_k(s, a))].
\]

Then,

\[
\hat{T}Q^2_k(s, a) - \hat{T}Q^1_k(s, a)
\]

\[
\leq \sup_{a' \in A} \left( \frac{-1}{\alpha} \log E_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma Q^1_k(s', a'))] + \frac{1}{\alpha} \log E_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma Q^2_k(s', a'))] \right)
\]

\[
\leq \sup_{a' \in A} \left( \frac{-1}{\alpha} \log E_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma Q^2_k(s', a'))] - \frac{1}{\alpha} \log E_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma Q^1_k(s', a'))] \right)
\]

\[
= \sup_{a' \in A} \left( \frac{-1}{\alpha} \log E_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma Q^1_k(s', a'))] - \frac{1}{\alpha} \log E_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma Q^2_k(s', a'))] \right)
\]

where the second inequality follows from (36), the third inequality from \(\log E[f(s) + g(s)] \geq E(f(s)) + E(g(s))\), and the last inequality from Jensen’s inequality. It follows from (37) and Assumption 1 that:

\[
\sup_{\xi' \in \bar{S}_k \times A} \log \left| \frac{\hat{T}Q^2_k(s, a) - \hat{T}Q^1_k(s, a)}{w(s, a)} \right|
\]

\[
\leq \sup_{\xi' \in \bar{S}_k \times A} \left( \gamma \log \left| \frac{Q^2_k(s', a') - Q^1_k(s', a')}{w(s', a')} \right| \right)
\]

\[
\leq \gamma \|Q^2_k - Q^1_k\|_w w(s, a)
\]

(38)

which implies (21). This completes the proof.

**APPENDIX C**

**PROOF OF COROLLARY 1**

Recall that

\[
g^k_a(s) + \frac{1}{\alpha} \log \left( \mathbb{E}_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma \inf_{a' \in A} \{Q^1_k(s', a')\})] \right)
\]

\[
\leq g^a(s) + \frac{1}{\alpha} \log \left( \mathbb{E}_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma Q^2_k(s', a'))] \right)
\]

(39)

∀(s, a, a') ∈ S_k × A^2. This implies that \(Q^*_k\) is a feasible solution to (25). Also, for any \(Q_k \in C(S_k \times A)\) in a feasible region of (25), the inequality

\[
Q_k(s, a) \leq g^k_a(s) + \frac{1}{\alpha} \log \left( \mathbb{E}_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma Q^2_k(s', a'))] \right)
\]

(40)

holds \(∀(s, a, a') \in S_k \times A^2\), which implies that it also holds for specific \(a' \in A\) that minimizes \(Q_k(s', a')\), that is, it also satisfies \(T\) operator, i.e., \(Q_k(s, a) \leq \hat{T}Q_k(s, a) = g^k_a(s) + \frac{1}{\alpha} \log \left( \mathbb{E}_{\xi' \sim \bar{P}'_{\alpha}}[\exp (\alpha \gamma \inf_{a' \in A} \{Q^1_k(s', a')\})] \right)\).

Based on Proposition 3, this indicates that \(Q^*_k\) is a lower bound to \(Q^*_k\). Since we are maximizing the objective function in (25), this means that \(Q^*_k\) is a maximizer of the optimization. This completes the proof.

**REFERENCES**

[1] A. Hakobyan, G. C. Kim, and I. Yang, “Risk-aware motion planning and control using CVaR-constrained optimization,” IEEE Robot. Autom. Lett., vol. 4, no. 4, pp. 3924–3931, Oct. 2019.

[2] J. R. Medina, D. Lee, and S. Hirche, “Risk-sensitive optimal feedback control for haptic assistance,” in Proc. IEEE Int. Conf. Robot. Autom., May 2012, pp. 1025–1031.

[3] E. Fernandez-Gaucherand and S. I. Marcus, “Risk-sensitive optimal control of hidden Markov models: Structural results,” IEEE Trans. Autom. Control, vol. 42, no. 10, pp. 1418–1422, Oct. 1997.

[4] E. Fernandez-Gaucherand and S. I. Marcus, “Risk-sensitive optimal control of hidden Markov models: A case study,” in Proc. 33rd IEEE Conf. Decis. Control, Dec. 1994, pp. 1657–1662.

[5] P. Brezas and M. C. Smith, “Linear quadratic optimal and risk-sensitive control for vehicle active suspensions,” IEEE Trans. Control Syst. Technol., vol. 22, no. 2, pp. 543–556, Mar. 2014.

[6] K. Lin, C. Jie, and S. I. Marcus, “Probabilistically distorted risk-sensitive infinite-horizon dynamic programming,” Automatica, vol. 97, pp. 1–6, Nov. 2018.

[7] M. R. James, J. S. Baras, and R. J. Elliott, “Risk-sensitive control and dynamic games for partially observed discrete-time nonlinear systems,” IEEE Trans. Autom. Control, vol. 39, no. 4, pp. 780–792, Apr. 1994.

[8] Y. G. Liu, Z. G. Pan, and S. J. Shi, “Output feedback control design for strict-feedback stochastic nonlinear systems under a risk-sensitive cost,” IEEE Trans. Autom. Control., vol. 48, no. 3, pp. 509–513, Mar. 2003.

[9] R. K. Boel, M. R. James, and I. R. Petersen, “Robustness and risk-sensitive filtering,” IEEE Trans. Autom. Control, vol. 47, no. 3, pp. 451–461, Mar. 2002.
[10] M. P. Chapman, J. P. Lacotte, K. M. Smith, I. Yang, H. Man, M. Pavone, and C. J. Tomlin, “Risk-sensitive safety specifications for stochastic systems using conditional value-at-risk,” 2019, arXiv:1909.09703.

[11] K. Dvijotham, M. Fazel, and E. Todorov, “Convex risk averse control design,” in Proc. 53rd IEEE Conf. Decis. Control, Dec. 2014, pp. 4020–4025.

[12] L. J. Ratliff and E. Mazumdar, “Inverse risk-sensitive reinforcement learning,” IEEE Trans. Autom. Control, vol. 65, no. 3, pp. 1256–1263, Mar. 2020.

[13] E. Mazumdar, L. J. Ratliff, T. Fiez, and S. S. Sastry, “Gradient-based inverse risk-sensitive reinforcement learning,” in Proc. IEEE 56th Annu. Conf. Decis. Control (CDC), Dec. 2017, pp. 5796–5801.

[14] A. Shapiro, D. Dentcheva, and A. Ruszczynski, Lectures on Stochastic Programming. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2009.

[15] N. Saldi, T. Başar, and M. Raginsky, “Approximate Markov-Nash equilibria for discrete-time risk-sensitive mean-field games,” Math. Oper. Res., vol. 45, no. 4, pp. 1596–1620, Nov. 2020.

[16] N. Bäuerle and J. Ott, “Markov decision processes with average-value-at-risk criteria,” Math. Methods Oper. Res., vol. 74, no. 3, pp. 361–379, Dec. 2011.

[17] D. Jelitto, M. Pitera, and L. Stettner, “Long-run risk-sensitive impulse control,” SIAM J. Control Optim., vol. 58, no. 4, pp. 2446–2468, Jan. 2020.

[18] G. B. Di Masi and L. Stettner, “Risk-sensitive control of discrete-time Markov processes with infinite horizon,” SIAM J. Control Optim., vol. 38, no. 1, pp. 61–78, 1999.

[19] S. Zhu and M. Fukushima, “Worst-case conditional value-at-risk with application to robust portfolio management,” Oper. Res., vol. 57, no. 5, pp. 1155–1168, Oct. 2009.

[20] M. P. Chapman, R. Bonalli, K. M. Smith, I. Yang, M. Pavone, and C. J. Tomlin, “Risk-sensitive safety analysis using conditional value-at-risk,” 2021, arXiv:2101.12086.

[21] L. A. Prashanth, “Risk-sensitive reinforcement learning—Part I: Constrained optimization framework,” in Proc. 5th Indian Control Conf. (ICC), Jan. 2019, p. 9.

[22] M. Ahmad, M. Ono, M. D. Ingham, R. M. Murray, and A. D. Ames, “Risk-averse planning under uncertainty,” in Proc. Amer. Control Conf. (ACC), Jul. 2020, pp. 3305–3312.

[23] S. Safarzai, B. J. Gravel, V. Renganathan, and T. H. Summers, “Risk-averse RRT* planning with nonlinear steering and tracking controllers for nonlinear robotic systems under uncertainty,” 2021, arXiv:2103.05572.

[24] W. B. Powell, Approximate Dynamic Programming: Solving the curses of dimensionality, vol. 703. Hoboken, NJ, USA: Wiley, 2007.

[25] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. Cambridge, MA, USA: MIT Press, 2018.

[26] D. Bertsekas, Reinforcement and Optimal Control. Nashua, NH, USA: Athena Scientific, 2019.

[27] R. Kamalapurkar, P. Walters, J. Rosenfeld, and W. E. Dixon, Reinforcement Learning for Optional Feedback Control: A Lyapunov-Based Approach. Cham, Switzerland: Springer, 2018.

[28] F. L. Lewis and D. Liu, Reinforcement Learning and Approximate Dynamic Programming for Feedback Control, vol. 17. Hoboken, NJ, USA: Wiley, 2013.

[29] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, “Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers,” IEEE Control Syst. vol. 32, no. 6, pp. 76–105, Dec. 2012.

[30] G. Banjac and J. Lygeros, “A data-driven policy iteration scheme based on linear programming,” in Proc. IEEE 58th Conf. Decis. Control (CDC), Dec. 2019, pp. 816–821.

[31] A. Agarwal, N. Jiang, S. M. Kakade, and W. Sun, Reinforcement Learning: Theory and Algorithms. Princeton, NJ, USA: Princeton Univ. Press, 2020.

[32] E. Fernández-Gaucherand and S. I. Marcus, “Risk-sensitive optimal control of hidden Markov models: Structural results,” IEEE Trans. Autom. Control, vol. 42, no. 10, pp. 1418–1422, 1997.

[33] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, “Risk-constrained reinforcement learning with percentile risk criteria,” J. Mach. Learn. Res., vol. 18, no. 1, pp. 6070–6120, 2017.

[34] W. B. Haskell and R. Jain, “A convex analytic approach to risk-aware Markov decision processes,” SIAM J. Control Optim., vol. 53, no. 3, pp. 1569–1598, Jan. 2015.

[35] W. B. Haskell and R. Jain, “Stochastic dominance-constrained Markov decision processes,” SIAM J. Control Optim., vol. 51, no. 1, pp. 273–303, Jan. 2013.

[36] A. Ruszczynski, “Risk-averse dynamic programming for Markov decision processes,” Math. Program., vol. 125, no. 2, pp. 235–261, Oct. 2010.

[37] Y. Shen, W. Stannat, and K. Obermayer, “Risk-sensitive Markov control processes,” SIAM J. Control Optim., vol. 51, no. 5, pp. 3652–3672, Jan. 2013.

[38] M. Althoff, “Reachability analysis and its application to the safety assessment of autonomous cars,” Ph.D. dissertation, Technische Universität München, Munich, Germany, 2010.

[39] P.-F. Massiani, S. Heim, F. Sölowojw, and S. Trimpe, “Safe value functions,” 2021, arXiv:2105.12204.
N. Saldi, T. Başar, and M. Raginsky, “Approximate Markov-Nash equilibria for discrete-time risk-sensitive mean-field games,” Math. Oper. Res., vol. 45, no. 4, pp. 1596–1620, Nov. 2020.

N. Bäuerle and A. Jaskiewicz, “Stochastic optimal growth model with risk sensitive preferences,” J. Econ. Theory, vol. 173, pp. 181–200, Jan. 2018.

S. Bitzer, H. Park, F. Blankenburg, and S. J. Kiebel, “Perceptual decision making: Drift-diffusion model is equivalent to a Bayesian model,” Frontiers Hum. Neurosci., vol. 8, p. 102, Feb. 2014.

D. P. Bertsekas, “Weighted sup-norm contractions in dynamic programming: A review and some new applications,” Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep. LIDS-P-2884, 2012, p. 5.

D. P. de Farias and B. V. Roy, “The linear programming approach to approximate dynamic programming,” Oper. Res., vol. 51, no. 6, pp. 850–865, 2003.

P. N. Beuchat, A. Georgiou, and J. Lygeros, “Performance guarantees for model-based approximate dynamic programming in continuous spaces,” IEEE Trans. Autom. Control, vol. 65, no. 1, pp. 143–158, Jan. 2020.

D. Bertsekas and J. N. Tsitsiklis, Neuro-Dynamic Programming. Nashua, NH, USA: Athena Scientific, 1996.

J. B. Conway, A Course in Functional Analysis, vol. 96, Chalmers, Switzerland: Springer, 2019.

A. Lazaric and M. Restelli, “Transfer from multiple MDPs,” in Proc. Adv. Neural Inf. Process. Syst., vol. 24, 2011, pp. 1746–1754.

H. B. Ammar et al., “An automated measure of MDP similarity for transfer in reinforcement learning,” in Proc. Workshops 28th AAAI Conf. Artif. Intell., 2014, pp. 1–7.

Á. Visís, J. García, and F. Fernández, “A taxonomy of similarity metrics for Markov decision processes,” 2021, arXiv:2103.04706.

C. Vallon and F. Borrelli, “Data-driven hierarchical predictive learning in unknown environments,” in Proc. IEEE 16th Int. Conf. Autom. Sci. Eng. (CASE), Aug. 2020, pp. 104–109.

C. Ning and F. You, “Online learning based risk-averse stochastic MPC of constrained linear uncertain systems,” Automatica, vol. 125, Mar. 2021, Art. no. 109402.

S. A. Sajadi-Alamdari, H. Voos, and M. Darouach, “Risk-averse stochastic nonlinear model predictive control for real-time safety-critical systems,” IFAC-PapersOnLine, vol. 50, no. 1, pp. 5991–5997, Jul. 2017.

Y.-L. Chow and M. Pavone, “A framework for time-consistent, risk-averse model predictive control: Theory and algorithms,” in Proc. Amer. Control Conf., Jun. 2014, pp. 4204–4211.

M. Schuurmans, A. Katriniok, H. E. Tseng, and P. Patrinos, “Learning-based risk-averse model predictive control for adaptive cruise control with stochastic driver models,” IFAC-PapersOnLine, vol. 53, no. 2, pp. 15128–15133, 2020.

M. Althoff, O. Stursberg, and M. Buss, “Model-based probabilistic collision detection in autonomous driving,” IEEE Trans. Intell. Transp. Syst., vol. 10, no. 2, pp. 299–310, Jun. 2009.

E. Y. Chen, M. I. Jordan, and S. Li, “Transferred Q-learning,” 2022, arXiv:2202.04709.

Majid Mazouchi received the B.Sc. degree from the K. N. Toosi University of Technology, Tehran, Iran, in 2007, and the M.Sc. and Ph.D. degrees from the Ferdowsi University of Mashhad, Mashhad, Iran, in 2010 and 2018, respectively.

He was a Senior Lecturer at Semnan University, Semnan, Iran, from 2017 to 2018. He is currently a Research Associate with the Department of Mechanical Engineering, Michigan State University, East Lansing, MI, USA. His current research interests include safe reinforcement learning, cyber-physical systems, and distributed control.

Dr. Mazouchi serves as a Reviewer for several international journals, including IEEE TRANSACTIONS ON AUTOMATIC CONTROL, Automatica, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, and IEEE/CAA JOURNAL OF AUTOMATICA SINICA.

Subramanya P. Nageshrao received the Ph.D. degree from the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands, in 2016.

He is currently a Research Engineer at Ford Green Field Labs, Palo Alto, CA, USA. Prior to that, he worked as a Post-Doctoral Research Scholar at the University of Michigan at Ann Arbor, Ann Arbor, MI, USA. He is actively leveraging machine learning methods, particularly reinforcement learning techniques to build next-generation state-of-the-art advanced driver assistance technologies.

Hamidreza Modares (Member, IEEE) received the B.S. degree from the University of Tehran, Tehran, Iran, in 2004, the M.S. degree from the Shahrood University of Technology, Shahrood, Iran, in 2006, and the Ph.D. degree from The University of Texas at Arlington, Arlington, TX, USA, in 2015.

He was a Senior Lecturer with the Shahrood University of Technology from 2006 to 2009 and a Faculty Research Associate with The University of Texas at Arlington from 2015 to 2016. He is currently an Assistant Professor with the Mechanical Engineering Department, Michigan State University, East Lansing, MI, USA. His current research interests include cyber-physical systems, reinforcement learning, distributed control, robotics, and machine learning.

Dr. Modares was a recipient of the Best Paper Award from the 2015 IEEE International Symposium on Resilient Control Systems. He is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.