2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

Enforcing Signal Temporal Logic Specifications in Multi-Agent Adversarial Environments: A Deep Q-Learning Approach

Devaprakash Muniraj, Kyriakos G. Vamvoudakis, and Mazen Farhood

Abstract— **This work addresses the problem of learning optimal control policies for a multi-agent system in an adversarial environment. Specifically, we focus on multi-agent systems where the mission objectives are expressed as signal temporal logic (STL) specifications. The agents are classified as either defensive or adversarial. The defensive agents are maximizers, namely, they maximize an objective function that enforces the STL specification; the adversarial agents, on the other hand, are minimizers. The interaction among the agents is modeled as a finite-state team stochastic game with an unknown transition probability function. The synthesis objective is to determine optimal control policies for the defensive agents that implement the STL specification against the best responses of the adversarial agents. A multi-agent deep Q-learning algorithm, which is an extension of the minimax Q-learning algorithm, is then proposed to learn the optimal policies. The effectiveness of the proposed approach is illustrated through a simulation case study.**

Index Terms— **multi-agent system, signal temporal logic, deep Q-learning.**

I. Introduction

Safety-critical autonomous systems typically operate in uncertain, dynamic, and adversarial environments. Additionally, many autonomous vehicle applications involve executing time-constrained tasks such as reaching a particular region within a specified time or visiting a particular region every few minutes. The problem setup considered in this work is that of a group of autonomous agents with unknown stochastic dynamics tasked to perform a time-constrained mission, which is expressed as a temporal logic specification, in an uncertain, adversarial environment. The agents in the multi-agent system are classified as either *defensive* agents or *adversarial* agents. The objective of the defensive agents is to complete the specified mission, while the objective of the adversarial agents is to act in a manner so as to prevent the defensive agents from achieving the mission.

The interaction among the agents of the multi-agent system is modeled as a team stochastic game, which is an extension of one-shot matrix games to environments modeled as Markov decision processes [1]. Temporal logic enables one to specify the mission objectives without any ambiguity (see [2] and the references therein). Signal temporal logic (STL) [3], which can be used to specify temporal properties of real-valued signals over finite time intervals, is used as the temporal logic framework to specify the mission objective. The STL specification is enforced by maximizing the expected robustness degree, which is a real number that denotes how well a given signal satisfies or violates the STL specification. The control synthesis objective for the defensive agents is to find optimal policies that maximize the expected robustness degree of the STL specification against the best responses of the adversarial agents.

In this work, a model-free reinforcement learning (RL) approach is adopted to learn the optimal policies of the defensive agents. The objective function that enforces the STL specification is not in the form of a sum of discounted rewards, and therefore RL methods are not directly applicable. Following an approach similar to [4], the objective function is approximated using an extended state space and is written as a sum of discounted rewards. Since the number of states of the team stochastic game grows exponentially with the number of agents and the horizon length of the STL formula, conventional multi-agent RL methods [5]–[7] lose their viability as they use a tabular representation to store data pertaining to the action-value function. Recent advances in deep reinforcement learning reported in [8], [9] overcome many limitations of conventional RL algorithms by representing the action-value function as a deep neural network. In this work, we propose a minimax deep Q-learning algorithm that combines ideas from deep Q-learning [9] and minimax Q-learning [5] to learn the optimal policies.

Related work: The problem of finding optimal policies for multi-agent systems with mission objectives specified as temporal logic specifications has been studied in [10]–[12]. Most of the existing approaches require precise knowledge of the system dynamics. In our proposed approach, the system model and the environment are assumed to be unknown and the agents learn the optimal policies by interacting with the environment through a sequence of observations, actions, and rewards. Minimax Q-learning is a model-free RL technique proposed in [5] to learn the optimal policies for two-player zero-sum stochastic games. In this work, an extension of the minimax Q-learning algorithm is proposed, wherein the Q-function is represented using a convolutional neural network. Although extensions of deep RL algorithms to multi-agent systems have been proposed in [13]–[15], to the best of the authors' knowledge, the problem of finding optimal control policies that enforce an STL specification in an adversarial environment has not been studied before.

Structure: The remainder of the paper is structured as follows. A brief overview of STL and stochastic games is

Devaprakash Muniraj and Mazen Farhood are with the Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24061, USA. Kyriakos G. Vamvoudakis is with the School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0150, USA. Email: devapm@vt.edu, kyriakos@gatech.edu, farhood@vt.edu.

This work was supported in part by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation (NSF) sponsored industry/university cooperative research center (I/UCRC) under NSF Grant Nos. IIP-1539975 and CNS-1650465, along with significant contributions from C-UAS industry members, NSF CAREER under Grant No. CPS-1750789, and NATO under Grant No. SPS G5176.

provided in Section II. In Section III, the control synthesis problem is stated and then reformulated as a multi-agent RL problem. In Section IV, the proposed minimax deep Q-learning algorithm is presented. Section V presents a simulation case study that illustrates the effectiveness of the proposed approach. Finally, conclusions are presented in Section VI.

II. Preliminaries

A. Signal Temporal Logic

In this work, the mission objective is formally specified in STL [3], which is an extension of linear temporal logic (LTL) [16] to real-valued signals. STL formulas are defined using the following syntax:

$$\varphi := \chi^{\eta} \mid \neg\chi^{\eta} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid F_{[a,b]}\varphi \mid G_{[a,b]}\varphi,$$

where χ^{η} is an atomic predicate whose truth value depends on the value of the function $\eta : \mathbb{R}^m \to \mathbb{R}$, and φ is an STL formula. In this work, η is assumed to be a function of the form $\eta := d - f(\mathbf{x})$, where $\mathbf{x} : \mathbb{R}_+ \to \mathbb{R}^m$ denotes the signal and d is a real-valued constant. The Boolean operators $\neg$, $\vee$, $\wedge$ have their usual meanings, and a, b are non-negative scalars used to denote time bounds. The operators F and G are analogous to the LTL operators $\Diamond$ (`Eventually`) and $\Box$ (`Always`), respectively.

The signal $\mathbf{x}$ here is a continuous-time signal, however, in practical applications, measurements of the signal might be available only at discrete time instances. In this work, we assume that it is always possible to obtain a continuous-time signal from discrete measurements. For instance, given a sequence of discrete measurements x_k of the signal $\mathbf{x}$ that correspond to time instances t_k, where k takes values in the set of natural numbers, a possible reconstruction of the continuous-time signal $\mathbf{x}$ is $\mathbf{x}(t)=x_k$, if $t \in [t_k, t_{k+1})$. $(\mathbf{x}, t)$ denotes the trace of the signal starting at time t and going forward. For given STL formulas ϕ and ψ, the Boolean semantics of STL are defined as follows:

$$\begin{aligned}
(\mathbf{x},t) &\models \chi^{\eta} && \Leftrightarrow \eta(x_t) \geqslant 0,\\
(\mathbf{x},t) &\models \phi \vee \psi && \Leftrightarrow (\mathbf{x},t) \models \phi \text{ or } (\mathbf{x},t) \models \psi,\\
(\mathbf{x},t) &\models \phi \wedge \psi && \Leftrightarrow (\mathbf{x},t) \models \phi \text{ and } (\mathbf{x},t) \models \psi,\\
(\mathbf{x},t) &\models G_{[a,b]}\phi && \Leftrightarrow (\mathbf{x},\hat{t}) \models \phi \ \forall \hat{t} \in [t{+}a, t{+}b], \text{ and}\\
(\mathbf{x},t) &\models F_{[a,b]}\phi && \Leftrightarrow \exists \hat{t} \in [t{+}a, t{+}b] \quad \text{s.t. } (\mathbf{x},\hat{t}) \models \phi.
\end{aligned}$$

In addition to the Boolean semantics, STL formulas can also be interpreted using quantitative semantics, which are defined using the robustness degree. For a given signal $\mathbf{x}$ and an STL specification ϕ, the robustness degree denoted by $\rho^{\phi}(\mathbf{x}, t)$ is defined as follows:

$$\begin{aligned}
\rho^{\chi^{\eta}}(\mathbf{x},t) &= \eta(x_t), \ \rho^{\neg\chi^{\eta}}(\mathbf{x},t) = -\eta(x_t),\\
\rho^{\phi\vee\psi}(\mathbf{x},t) &= \max\ (\rho^{\phi}(\mathbf{x},t), \rho^{\psi}(\mathbf{x},t)),\\
\rho^{\phi\wedge\psi}(\mathbf{x},t) &= \min\ (\rho^{\phi}(\mathbf{x},t), \rho^{\psi}(\mathbf{x},t)),\\
\rho^{G_{[a,b]}\phi}(\mathbf{x},t) &= \min_{\hat{t}\in[t+a,t+b]} \rho^{\phi}(\mathbf{x},\hat{t}), \text{ and}\\
\rho^{F_{[a,b]}\phi}(\mathbf{x},t) &= \max_{\hat{t}\in[t+a,t+b]} \rho^{\phi}(\mathbf{x},\hat{t}).
\end{aligned}$$

The signal $\mathbf{x}$ satisfies the STL formula ϕ at time t if and only if $\rho^{\phi}(\mathbf{x}, t) \geqslant 0$. The robustness degree provides a quantitative measure of the extent to which a signal trace satisfies an STL specification. Another definition which will be used in the sequel is the horizon length of an STL formula ϕ, denoted by $H(\phi)$, which is the duration of time required to verify whether a signal satisfies the formula ϕ. For instance, the horizon length of the STL formula $G_{[2,5]}\phi$ is 5. The horizon length of any other STL formula is computed as in [17].

B. Stochastic Games

Stochastic games, also known as Markov games, are an extension of Markov decision processes to multiple agents [1]. An n-player discounted stochastic game is defined by the tuple $G{=}\langle S, A_1, \ldots, A_n, P, R_1, \ldots, R_n, \gamma\rangle$, where S is the set of states of the multi-agent system and A_i is the finite set of actions available to agent i. P: $S{\times}A_1{\times}\cdots{\times}A_n{\times}S{\to}[0,1]$ is the transition probability function that gives a probability distribution for the next state of the system given the current state and the joint action of the agents. For instance, $P(s, a_1, a_2, s')$ is the probability of ending in state s', given that the agents start in state s and each agent $i \in \{1, 2\}$ takes action $a_i \in A_i$. R_i: $S{\times}A_1{\times}\cdots{\times}A_n{\times}S \to \mathbb{R}$ is the reward function of agent i, and $0{\leqslant}\gamma{<}1$ is the discount factor.

A (stationary) control policy π_i for agent i is a mapping π_i: $S{\to}PD(A_i)$ from the set of states to the set of discrete probability distributions over the set of control actions of agent i. As in [6], given the joint policy $\pi{=}(\pi_1, \ldots, \pi_n)$, the action-value function $Q_i^{\pi}(s, a_1, \ldots, a_n)$ can be defined as

$$\begin{aligned}
Q_i^{\pi}(s, a_1, \ldots, a_n) = \sum_{s'\in S} P(s, a_1, \ldots, a_n, s'){\times}\\
[R_i(s, a_1, \ldots, a_n, s') + \gamma V_i^{\pi}(s')].
\end{aligned}$$

$V_i^{\pi}(\cdot)$ in the above equation is the state-value function under π and is defined as

$$V_i^{\pi}(s){=}\sum_{j=0}^{\infty} \gamma^j E^{\pi}(R_i(s_j, a_1, \ldots, a_n, s_{j+1})),$$

where $E^{\pi}(\cdot)$ is the expectation operator over π. The Q-functions are used to relate the equilibrium in stage games to the equilibrium of a stochastic game [18], [19]. Specifically, if the Q-function at each state is considered as the payoff function for the stage game, then the stage policies are in equilibrium if and only if the multistage policies are in equilibrium.

Definition 1: (Nash equilibrium) A Nash equilibrium (NE) in a stochastic game G is an n-tuple of strategies $(\pi_1^*, \ldots, \pi_n^*)$ such that $\forall s{\in}S$ and $i{\in}\{1, \ldots, n\}$, the following inequalities hold:

$$V_i^{(\pi_1^*,\ldots,\pi_i^*,\ldots,\pi_n^*)}(s) \geqslant V_i^{(\pi_1^*,\ldots,\pi_i,\ldots,\pi_n^*)}(s) \quad \forall \pi_i \in \Pi_i,$$

where Π_i is the set of strategies available to player i.

If the reward functions of the agents are in conflict with each other, then a special case of NE called the adversarial equilibrium can be defined.

Definition 2: (Adversarial equilibrium) An adversarial equilibrium in a stochastic game G is an n-tuple of strategies $(\pi_1^*, \ldots, \pi_n^*)$ such that $\forall s{\in}S$ and $i{\in}\{1, \ldots, n\}$, the following are true:

$$\begin{aligned}
&V_i^{(\pi_1^*,\ldots,\pi_i^*,\ldots,\pi_n^*)}(s) \geqslant V_i^{(\pi_1^*,\ldots,\pi_i,\ldots,\pi_n^*)}(s) \quad \text{and}\\
&V_i^{(\pi_1^*,\ldots,\pi_i^*,\ldots,\pi_n^*)}(s) \leqslant V_i^{(\pi_1,\ldots,\pi_i^*,\ldots,\pi_n)}(s) \quad \forall \pi_i \in \Pi_i.
\end{aligned}$$

If an agent unilaterally deviates from the adversarial equi-

librium, it not only decreases its expected payoff, but also increases the expected payoffs of the other agents. If a stochastic game has only two players and their reward functions are such that $R_1(s,a_1,a_2,s') = -R_2(s,a_1,a_2,s')$ $\forall s{\in}S$, $s'{\in}S$, $a_1{\in}A_1$, and $a_2{\in}A_2$, then the stochastic game is called a zero-sum stochastic game. All Nash equilibria in a zero-sum stochastic game are adversarial equilibria [20]. The concepts of Nash equilibrium and adversarial equilibrium are important in multi-agent reinforcement learning, as they are used as learning goals by most of the RL algorithms.

III. Problem Formulation

A. System Model and Control Objective

The multi-agent system that we consider in this work consists of two teams of agents, namely the team of defensive agents and the team of adversarial agents. The interaction among the agents is modeled as a team stochastic game, which is an n-player stochastic game consisting of a team of p maximizers and $(n-p)$ minimizers [21]. During the game, the defensive agents (maximizers) and the adversarial agents (minimizers) choose actions that respectively maximize and minimize their expected discounted return.

Let the team stochastic game be denoted by the tuple $\bar{G}{=}\langle S, \bar{A}_1, \bar{A}_2, P, R, \gamma\rangle$, where $\bar{A}_1{=}A_1{\times}\cdots{\times}A_p$ and $\bar{A}_2{=}A_{p+1}{\times}\cdots{\times}A_n$ are the joint action sets of the defensive agents and the adversarial agents, respectively. The state space S is composed of discrete partitions of the continuous state space. For example, if the continuous state space is a two-dimensional grid, then $s_k \in \mathbb{R}^{2n}$ and is of the form $s_k{=}(x_1,y_1,\ldots,x_n,y_n)^T$, where (x_i,y_i) denotes the centroid of the cell that contains the i^{th} agent. All the agents share the same reward function R. A state trajectory of the system is of the form $(s_0 s_1 \ldots s_N)$, where s_k corresponds to the state of the system at time t_k, and for $k=1,\ldots,N$, $t_k{-}t_{k-1}{=}\Delta t$ with Δt being the time step. The state trajectory of the system during the time interval $[0,T]$ is denoted by $s_{0:T}$ and is given by $(s_0 s_1 \ldots s_N)$, where $T = N\Delta t$.

In this work, it is assumed that the defensive agents and the adversarial agents are fully cooperative among themselves, by which we mean that the agents within a team either maximize or minimize the reward function together. Therefore, the agents within a team also share the same Q-function. We do not make any assumptions on the transition probability function P, instead, it is considered as an unknown. Thus, the proposed approach obviates the need for a system model.

As mentioned earlier, the mission objective is specified as an STL formula and is denoted as Φ. The control synthesis problem is then to find the optimal policies for the defensive agents such that the STL specification Φ is enforced by maximizing the expected robustness degree against the best responses of the adversarial agents. Since a team stochastic game can be considered as a two-player zero-sum stochastic game, therefore, without any loss of generality, we assume that $p{=}1$ and $n{=}2$ and model the interaction between the agents as a two-player zero-sum stochastic game. The control objective for the case of two agents is now formally stated.

Control objective: Given an STL specification Φ with $H(\Phi){=}T$, a zero-sum stochastic game denoted by the tuple $G{=}\langle S, A_1, A_2, P, R, \gamma\rangle$ with unknown P, and an initial partial state trajectory $\tilde{s}_{0:t'}$ for some $0{\leqslant}t'{<}T$, the control objective is to find a control policy π_1^* such that

$$\pi_1^* = \arg\max_{\pi_1\in\Pi_1}\min_{\pi_2\in\Pi_2} E^{(\pi_1,\pi_2)}[\rho^{\Phi}(s_{0:T})], \tag{1}$$

where Π_1 and Π_2 denote the set of policies available to the maximizing agent and the minimizing agent, respectively. $s_{0:T}$ denotes a state trajectory of the system during the interval $[0,T]$ with $s_{0:t'}{=}\tilde{s}_{0:t'}$. $E^{(\pi_1,\pi_2)}[\rho^{\Phi}(s_{0:T})]$ is the expected robustness degree of $s_{0:T}$ with respect to Φ under the policies π_1 and π_2. In order to ensure that the minimum is achieved in (1), we assume that the policy of the adversarial agent, π_2, is deterministic [6].

B. Reformulation of the Control Synthesis Problem

The objective function in the control synthesis problem posed in the previous subsection is not in the form of a sum of discounted rewards, and so RL methods are not directly applicable. Moreover, if the STL specification Φ contains a nested operator such as $\Phi = F_{[0,a]}G_{[0,b]}\phi$, the control policy of the agent should take into account not only the current state of the system but also a sufficient length of the state history. To overcome these two issues, we adopt an approach similar to the one proposed in [4] to redefine the state space based on Φ. We assume that the STL specification Φ is of the form $\Phi = G_{[0,T]}\phi$ or $\Phi = F_{[0,T]}\phi$, where ϕ is any STL formula as defined in Section II. Let τ be a non-negative integer defined as $\tau = \lceil H(\phi)/\Delta t\rceil + 1$, where $\lceil\cdot\rceil$ is the ceiling function and Δt is the time step. The state space of the zero-sum stochastic game is redefined as S^τ and an extended zero-sum stochastic game is formed.

Definition 3: (Extended zero-sum stochastic game) Given a zero-sum stochastic game denoted by the tuple $G{=}\langle S, A_1, A_2, P, R, \gamma\rangle$ and a non-negative integer τ, an extended zero-sum stochastic game is defined as the tuple $G^\tau{=}\langle S^\tau, A_1, A_2, P^\tau, R^\tau, \gamma\rangle$, where

- each state $s^\tau \in S^\tau$ corresponds to a state trajectory with τ samples; in the case that the length of the trajectory k is less than τ, then s^τ is formed by appending the empty string ϵ to the trajectory $\tau - k$ times;
- P^τ: $S^\tau{\times}A_1{\times}A_2{\times}S^\tau{\to}[0,1]$ is the transition probability function and is related to P as follows: given $s_i^\tau{=}s_a s_b \ldots s_c$ and $s_j^\tau{=}s_d \ldots s_f s_g$, $P^\tau(s_i^\tau, a_1, a_2, s_j^\tau){>}0$ if and only if $P(s_c,a_1,a_2,s_g) \in [0,1]$ and $s_d \ldots s_f = s_b \ldots s_c$;
- A_i is the finite set of actions for agent i; and
- R^τ: $S^\tau{\times}A_1{\times}A_2{\times}S^\tau \to \mathbb{R}$ is the reward function.

The optimal policy of the extended stochastic game is given by the map $\pi_1^* : S^\tau \to PD(A_1)$. Given a state trajectory $s_{0:T}$, the corresponding trajectory in the extended state space can be written as $s^\tau_{(\tau-1)\Delta t:T} = s^\tau_{\tau-1}\ldots s^\tau_N$, where $T{=}N\Delta t$ and $s^\tau_k = s_{(k-\tau+1)\Delta t:k\Delta t}$ for $(\tau-1){\leqslant}k{\leqslant}N$. Using this relation and the quantitative semantics of the robustness degree provided in Section II, the robustness degree of $s_{0:T}$ with respect to Φ can be written as

$$\rho^{\Phi}(s_{0:T}) = f(\rho^{\Phi}(s^\tau_{\tau-1}),\ldots,\rho^{\Phi}(s^\tau_T)), \tag{2}$$

where the function $f(\cdot)$ depends on the temporal operator used in the STL specification Φ. If $\Phi = F_{[0,T]}\phi$, then $f(\cdot)$ is the maximum function, and if $\Phi = G_{[0,T]}\phi$, $f(\cdot)$

is the minimum function. The maximum function can be approximated by the log-sum-exp function as

$$\max(z_1,\dots,z_n) \approx \frac{1}{\beta}\log\sum_{i=1}^{n} e^{\beta z_i}, \tag{3}$$

where β is a positive constant. Using equations (2) and (3), the objective function in (1) can be written as

$$E^{(\pi_1,\pi_2)}\big[\rho^{\Phi}(s_{0:T})\big] \approx E^{(\pi_1,\pi_2)}\bigg[\frac{1}{\beta}\log\sum_{k=\tau-1}^{N} e^{\beta\rho^{\Phi}(s_k^\tau)}\bigg], \tag{4}$$

for $\Phi = F_{[0,T]}\phi$. Similarly, for $\Phi = G_{[0,T]}\phi$, we can write

$$E^{(\pi_1,\pi_2)}\big[\rho^{\Phi}(s_{0:T})\big] \approx E^{(\pi_1,\pi_2)}\bigg[-\frac{1}{\beta}\log\sum_{k=\tau-1}^{N} e^{-\beta\rho^{\Phi}(s_k^\tau)}\bigg]. \tag{5}$$

The objective function in the control synthesis problem can now be interpreted as a sum of rewards, where the agents receive an immediate reward at every time instant t_k after both agents take their corresponding actions. Using the approximations given in (4) and (5), the control synthesis problem given in (1) is reformulated as follows.

Approximated control synthesis problem: Given an STL specification Φ with $H(\Phi)=T$, where $T=N\Delta t$ for some positive integer N, and a zero-sum stochastic game denoted by the tuple $G=\langle S, A_1, A_2, P, R, \gamma\rangle$, let $G^\tau=\langle S^\tau, A_1, A_2, P^\tau, R^\tau, \gamma\rangle$ be the extended zero-sum stochastic game defined as in Definition 3. For a given initial partial state trajectory $s^\tau_{\tau-1}=s_{0:(\tau-1)\Delta t}$ and $\beta>0$, the control objective is to find a control policy $\pi_1^* : S^\tau \to PD(A_1)$ such that

$$\pi_1^* = \arg\max_{\pi_1\in\Pi_1}\min_{\pi_2\in\Pi_2^D} E^{(\pi_1,\pi_2)}\bigg[\sum_{k=\tau-1}^{N} r(s_k^\tau,\phi)\bigg], \tag{6}$$

$$\text{where}\quad r(s_k^\tau,\phi) = \begin{cases} e^{\beta\rho^{\Phi}(s_k^\tau)} & \text{if } \Phi = F_{[0,T]}\phi \\ -e^{-\beta\rho^{\Phi}(s_k^\tau)} & \text{if } \Phi = G_{[0,T]}\phi \end{cases},$$

and Π_2^D denotes the set of deterministic policies of the adversarial agent.

The control synthesis problem is in the standard form for multi-agent reinforcement learning where the agents receive an immediate reward during each state transition. The optimal control policy π_1^* maximizes the expected robustness degree with respect to the STL specification Φ against the best response of the adversarial agent.

IV. Learning The Optimal Policies

A. Learning Control Policies Using Minimax Q-Learning

Minimax Q-learning, which was first proposed in [5], is a model-free reinforcement learning method for learning optimal control policies in two-player zero-sum stochastic games. The learning objective in minimax Q-learning is to learn the control policy that ensures that the agents are in adversarial equilibrium provided the adversarial agent plays its optimal policy. In minimax Q-learning, the state-value function of the learning agent (agent 1) is given by

$$V_1(s) = \max_{\pi_1(s)\in\Pi_1}\min_{a_2\in A_2}\sum_{a_1\in A_1} Q_1(s,a_1,a_2)\pi_1^{a_1}(s),$$

where $\pi_1^{a_1}(s)$ is the probability with which action a_1 is chosen at state s under the policy π_1. The learning agent learns its optimal policy by solving the stage game at each state with the Q-values $Q_1(s,\cdot,\cdot)$ as the immediate payoffs. The minimax Q-learning algorithm for learning the optimal policy for the approximated control synthesis problem in (6) is presented in Algorithm 1.

In Algorithm 1, $\alpha_j \in (0,1)$ is the learning rate, $\gamma<1$ is the discount factor from the definition of the zero-sum stochastic game, N_p denotes the number of episodes, and $T=H(\Phi)=N\Delta t$. The minimax Q-learning player always plays a safe strategy and therefore does not adapt to the opponent's change in strategies. The following theorem provides conditions for convergence to an adversarial equilibrium for the approximated control synthesis problem defined in (6) under Algorithm 1.

Algorithm 1: Minimax Q-learning algorithm

Initialize $Q_1(s^\tau,a_1,a_2)$, $V_1(s^\tau)$, and $\pi_1(s^\tau)$
procedure
  for $j=1:N_p$ **do**
    Initialize the state $s^\tau_{\tau-1}$
    for $k=\tau-1:N$ **do**
      Select an action a_1 based on policy π_1 with probability ϵ or a random action from A_1 with probability $1-\epsilon$
      At the new state s^τ_{k+1}, observe reward r_1 and opponent's action a_2
      Update $Q_1(s_k^\tau,a_1,a_2)$ using the following rule
$Q_1(s_k^\tau,a_1,a_2)\leftarrow(1-\alpha_j)Q_1(s_k^\tau,a_1,a_2)+\alpha_j(r_1+\gamma V_1(s_{k+1}^\tau))$
      Update the policy by solving a linear program
$\pi_1(s_k^\tau)\leftarrow \arg\max_{\pi_1(s_k^\tau)}\min_{\bar a_2\in A_2}\sum_{\bar a_1\in A_1} Q_1(s_k^\tau,\bar a_1,\bar a_2)\pi_1^{\bar a_1}(s_k^\tau)$
      $V_1(s_k^\tau)=\min_{\bar a_2\in A_2}\sum_{\bar a_1\in A_1} Q_1(s_k^\tau,\bar a_1,\bar a_2)\pi_1^{\bar a_1}(s_k^\tau)$
      $s_k^\tau\leftarrow s_{k+1}^\tau$

Theorem 1: Given an STL specification Φ with $H(\Phi)=T$ and a zero-sum stochastic game denoted by the tuple $G=\langle S, A_1, A_2, P, R, \gamma\rangle$, let $G^\tau=\langle S^\tau, A_1, A_2, P^\tau, R^\tau, \gamma\rangle$ be the extended zero-sum stochastic game. Suppose that the optimal Q-function and the optimal policy are denoted by $Q_1^*(s^\tau,a_1,a_2)$ and $\pi_1^*(s^\tau)$, respectively, and that the following assumptions hold:

i) Every state s and action $a_i\in A_i$, for $i=1,2$, are visited infinitely often and
ii) The learning rate α satisfies the conditions $\sum_{j=0}^{\infty}\alpha_j=\infty$ and $\sum_{j=0}^{\infty}(\alpha_j)^2<\infty$.

Then, Algorithm 1 with the update rules

$$Q_1^{j+1}(s_k^\tau,a_1,a_2) = (1-\alpha_j)Q_1^j(s_k^\tau,a_1,a_2)+$$
$$\alpha_j\big(r_1+\gamma\max_{\pi_{1_j}(s_{k+1}^\tau)}\min_{\bar a_2\in A_2}\sum_{\bar a_1\in A_1}Q_1^j(s_{k+1}^\tau,\bar a_1,\bar a_2)\pi_{1_j}^{\bar a_1}(s_{k+1}^\tau)\big),$$
$$\pi_{1_{j+1}}(s_k^\tau) = \arg\max_{\pi_{1_j}(s_k^\tau)}\min_{\bar a_2\in A_2}\sum_{\bar a_1\in A_1}Q_i^j(s_k^\tau,\bar a_1,\bar a_2)\pi_{1_j}^{\bar a_1}(s_k^\tau),$$

$$\text{where } r_1 = \begin{cases} e^{\beta\rho^{\Phi}(s_{k+1}^\tau)} & \text{if } \Phi = F_{[0,T]}\phi \\ -e^{-\beta\rho^{\Phi}(s_{k+1}^\tau)} & \text{if } \Phi = G_{[0,T]}\phi \end{cases},$$

for some $\beta>0$, ensures convergence of $Q_1^j(\cdot,\cdot,\cdot)$ and $\pi_{1_j}(\cdot)$ to $Q_1^*(\cdot,\cdot,\cdot)$ and $\pi_1^*(\cdot)$, respectively, with probability 1 as $j\to\infty$.

Proof. The proof is a generalization of Theorem 3.1 in [22] and is omitted here due to paucity of space. ∎

B. Learning Control Policies Via Minimax Deep Q-Learning

One of the major shortcomings of Algorithm 1, or Q-learning in general, is scalability as the size of the state space increases exponentially with τ and the number of partitions of the continuous state space. The deep Q-learning algorithm proposed in [9] overcomes the issue of scalability by representing the Q-function as a deep neural network, where the network weights are learned during the training process. Inspired by deep Q-learning, we propose an algorithm called the minimax deep Q-learning algorithm, which is an extension of the minimax Q-learning algorithm presented in Algorithm 1, to learn the optimal policies. Similar to the input of the Q-function in [9], the input of the Q-function in Algorithm 1 can be thought of as a series of image frames of length $\tau + 2$, where the first τ image frames correspond to $s^{\tau}_{(\cdot)}$, and the last two frames encode the actions of the agents. This will enable us to use convolutional neural networks to approximate the Q-function; such neural networks have been found to provide promising results for applications that involve recognizing patterns in data [8], [9].

Algorithm 2: Minimax deep Q-learning algorithm

Initialize the replay buffer D of length M; initialize the Q-network Q_1 with random weights θ; initialize the target network $\hat{Q}_1$ with weights $\hat{\theta}{=}\theta$; initialize $\pi_1(s^{\tau})$
procedure
  for $k = 1 : N_p$ **do**
    Initialize the state $s^{\tau}_{\tau-1}$
    for $t = \tau{-}1 : N$ **do**
      Select an action a^t_1 based on policy π_1 with probability ϵ or a random action from A_1 with probability $1-\epsilon$
      At new state s^{τ}_{t+1}, observe reward r_t and opponent's action a^t_2; store transition $(s^{\tau}_t, a^t_1, a^t_2, s^{\tau}_{t+1}, r_t)$ in D
      Sample random minibatch of transitions $(s^{\tau}_j, a^j_1, a^j_2, s^{\tau}_{j+1}, r_j)$ from D; set the target y_j as r_j if episode terminates at $j{+}1$, or else set y_j as

$$y_j \leftarrow r_j + \max_{\pi_1(s^{\tau}_{j+1})} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \hat{Q}_1(s^{\tau}_{j+1}, a_1, a_2; \hat{\theta}) \pi_1^{a_1}(s^{\tau}_{j+1})$$

      Perform a gradient descent step on $(y_j - Q_1(s^{\tau}_j, a^j_1, a^j_2; \theta))^2$ with respect to θ
    $\pi_1(s^{\tau}_t) \leftarrow \arg\max_{\pi_1(s^{\tau}_t)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} Q_1(s^{\tau}_t, a_1, a_2; \theta) \pi_1^{a_1}(s^{\tau}_t)$
    $s^{\tau}_t \leftarrow s^{\tau}_{t+1}$; every C steps, reset $\hat{Q}_1{=}Q_1$

It is known that using a nonlinear function approximator such as a neural network to represent the Q-function might result in instability during the learning process [23]. The deep Q-learning algorithm proposed in [9] makes use of two heuristics to overcome the issue of instability. In the proposed minimax deep Q-learning algorithm presented in Algorithm 2, we take advantage of these heuristics in order to stabilize the learning process. The first heuristic involves storing the experiences of the learning agent at each time step in a replay buffer $D{=}\{e_1, \ldots, e_t\}$, where $e_t{=}(s^{\tau}_i, a_1, a_2, s^{\tau}_j, r_1)$. During the Q-learning update, the algorithm randomly samples the experience from the replay buffer and uses it to update the parameters of the network. The second heuristic makes use of a separate target Q-network for generating the targets y_j during the Q-learning update. The target network stabilizes the learning process by ensuring that the Q-function to be learned is not continuously changing.

V. Simulation Case Study

In this section, Algorithms 1 and 2 are applied to a simulation case study. The case study involves an agent moving in a $4{\times}4$ spatial grid and tasked to reach a goal and stay within it for a stipulated number of time steps in the presence of an adversary. The agents are allowed to take a step in either the north (N), the east (E), the west (W), the south (S), the north-east (NE), the north-west (NW), the south-east (SE), or the south-west (SW) direction. The uncertainty in the dynamics is modeled as follows: if an agent selects an action a, then the agent executes the selected action with probability 0.95 or executes a randomly selected action from the set $\{a_-, a_+\}$ with probability 0.05. If one visualizes the agents' possible actions to be rays emanating from the center of a circle, a_- and a_+ are actions that are immediately to the left and the right of a, respectively. For instance, if the selected action is N, then a_- and a_+ are the actions NW and NE, respectively. When an agent in state s takes an action that drives it outside the boundary of the grid, the agent remains in the same state s after executing the action. If the actions of the agents in state s are such that they move into the same grid, then both agents remain in s after the transition.

The objective of agent 1 is eventually to reach the region shown in Figure 1 with a green outline within 5 time steps and stay within that region for at least 2 consecutive time steps. The task objective is formally specified using the following STL specification:

$$\Phi = F_{[0,5]} G_{[0,1]} (\text{region } A \wedge \neg \text{region } B), \tag{7}$$

where region A represents $(x_1 > 0 \wedge x_1 < 2 \wedge y_1 > 2 \wedge y_1 < 4)$ and region B represents $(x_1 > 1 \wedge x_1 < 2 \wedge y_1 > 2 \wedge y_1 < 3)$. The specification results in $\tau{=}2$ with a state space of dimension $|S^2| = 7056$. We use Algorithms 1 and 2 presented in Section IV to learn the optimal policies that enforce the STL formula Φ given in (7). During the implementation of Algorithm 2, a convolutional neural network (CNN) is used to represent the Q-function. The input to the CNN consists of $\tau + 2$ frames, where the first τ frames represent the state of the system and the last two frames encode the actions taken by each of the agents. The frames that correspond to the states take values in the set $\{-1, 0, 1\}$, where a value of 1 at a cell corresponds to the presence of the agent and a value of -1 corresponds to the presence of the adversary. A value of 0 indicates that the cell is empty. The frames that encode the agents' actions take values in the set $\{1, \ldots, 8\}$. The output of the network is the value of the Q-function corresponding to the state of the system and the actions executed by the two agents. The CNN used to represent the Q-function is composed of two convolution layers having 15 filters each and a stride of 1, with the second convolution layer followed by a hyperbolic tangent nonlinearity and two fully connected layers. The parameters used in the implementation of Algorithms 1 and 2 are $M{=}1000$, $C{=}70$, $\gamma{=}0.9999$, $\alpha{=}0.99{\times}10^{(\log(0.01)/N_p)}$, $\epsilon{=}0.9$, $\beta{=}7$, and $N_p{=}50000$. The size of the minibatch used in Algorithm 2 is 25.

In this case study, two types of adversaries were considered, namely the random adversary (RA) that plays a

	Algorithm 1		Algorithm 2	
	π_A^*	π_B^*	π_A^*	π_B^*
RA	0.812	0.846	0.938	0.919
OA	0.271	0.394	0.353	0.481

TABLE I: $Pr[s_{[0:6]} \models \Phi]$ computed from 1000 simulations.

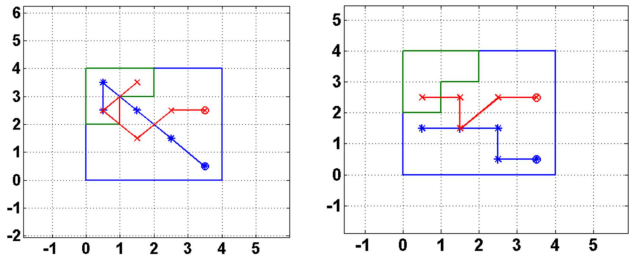

Fig. 1: Sample trajectories using the optimal strategy π_B^* learned using Algorithm 2; the trajectories of the defensive agent and the adversarial agent are shown in blue and red, respectively; the adversary is playing its optimal strategy; the figure on the left shows a simulation where the defensive agent successfully satisfies the specification, whereas in the figure on the right, the agent fails to satisfy the specification.

stationary random strategy, and the optimal adversary (OA) that plays the optimal strategy. Let π_A^* and π_B^* denote the optimal policies of the agent learned against RA and OA, respectively. The optimal strategy of the adversary is first learned against a deterministic hand-built strategy of the agent. The performance measure used to evaluate the learned optimal policies is the ratio of the number of trajectories that satisfy the specification Φ to the total number of simulated trajectories, and is denoted by $Pr[s_{[0:7]} \models \Phi]$. The results are summarized in Table I. It is observed that the optimal policies learned using Algorithm 2 outperform the optimal policies learned using Algorithm 1. The reason for the improvement in performance when applying Algorithm 2 is the use of a convolution network to represent the Q-function. In Algorithm 1, the Q-function is updated only for the state-action combinations that the agents visit. However, in Algorithm 2, the Q-network that represents the Q-function is updated at all state-action combinations and therefore provides a better representation of the Q-values for the state-action combinations that are not often visited. It is noted that in an ideal case when both the agent and the adversary are playing their optimal policies, the performance measure should be close to 0.5 since the agent and the adversary are in an adversarial equilibrium. However, in this case study, the optimal policy used by the adversary is learned against a hand-built policy of the agent and therefore might not have converged to the adversarial equilibrium, thereby explaining the performance values shown in the second row of Table I. Results from two representative simulations where the agent plays the optimal policy π_B^*, which is learned using Algorithm 2, against the optimal adversary are shown in Figure 1.

VI. Conclusions

This work addressed the problem of learning optimal control policies for a multi-agent system in an adversarial environment where the mission objective is expressed as an STL specification. The control synthesis problem was formulated as a zero-sum discounted stochastic game. It was shown that the control synthesis problem is very similar to the deep Q-learning problem where the inputs are in the form of a sequence of image frames. A multi-agent deep Q-learning algorithm was proposed to learn the optimal policies. The effectiveness of the proposed approach was illustrated using a simulation case study.

References

[1] J. van der Wal, "Stochastic dynamic programming," Ph.D. dissertation, Methematisch Centrum, 1980.

[2] C. Belta, B. Yordanov, and E. A. Gol, "Formal methods for discrete-time dynamical systems," 2017.

[3] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *FORMATS/FTRTFT*, vol. 3253, 2004, pp. 152–166.

[4] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-Learning for robust satisfaction of Signal Temporal Logic specifications," 2016, pp. 6565–6570.

[5] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the eleventh international conference on machine learning*, vol. 157, 1994, pp. 157–163.

[6] ——, "Value-function reinforcement learning in Markov games," *Cognitive Systems Research*, vol. 2, no. 1, pp. 55–66, 2001.

[7] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *Journal of Machine Learning Research*, vol. 4, no. Nov, pp. 1039–1069, 2003.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[10] A. Nikou, J. Tumova, and D. V. Dimarogonas, "Cooperative task planning of multi-agent systems under timed temporal specifications," in *Proceedings of the American Control Conference (ACC)*. IEEE, 2016, pp. 7104–7109.

[11] Z. Liu, J. Dai, B. Wu, and H. Lin, "Communication-aware motion planning for multi-agent systems from signal temporal logic specifications," *arXiv preprint arXiv:1705.11085*, 2017.

[12] S. Andersson, A. Nikou, and D. V. Dimarogonas, "Control synthesis for multi-agent systems under metric interval temporal logic specifications," *arXiv preprint arXiv:1703.02780*, 2017.

[13] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, p. e0172395, 2017.

[14] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2017)*, 2017.

[15] G. Palmer, K. Tuyls, D. Bloembergen, and R. Savani, "Lenient multi-agent deep reinforcement learning," *arXiv preprint arXiv:1707.04402*, 2017.

[16] A. Pnueli, "The temporal logic of programs," in *Foundations of Computer Science, 1977., 18th Annual Symposium on*, 1977, pp. 46–57.

[17] A. Dokhanchi, B. Hoxha, and G. Fainekos, "On-line monitoring for temporal logic robustness," in *International Conference on Runtime Verification*. Springer, 2014, pp. 231–246.

[18] J. Filar and K. Vrieze, *Competitive Markov decision processes*. Springer Science & Business Media, 2012.

[19] M. L. Littman, "Friend-or-foe q-learning in general-sum games," in *ICML*, vol. 1, 2001, pp. 322–328.

[20] A. Neyman and S. Sorin, *Stochastic games and applications*. Springer Science & Business Media, 2003, vol. 570.

[21] M. G. Lagoudakis and R. Parr, "Learning in zero-sum team markov games using factored value functions," in *Advances in Neural Information Processing Systems*, 2003, pp. 1659–1666.

[22] C. Szepesvári and M. L. Littman, "Generalized markov decision processes: Dynamic-programming and reinforcement-learning algorithms," in *Proceedings of International Conference of Machine Learning*, vol. 96, 1996.

[23] J. N. Tsitsiklis and B. Van Roy, "Analysis of temporal-diffference learning with function approximation," in *Advances in neural information processing systems*, 1997, pp. 1075–1081.