IEEE CSS

Received 12 February 2024; revised 14 May 2024; accepted 11 June 2024. Date of publication 24 June 2024; date of current version 5 July 2024. Recommended by Guest Editor Ahmad Taha.

Digital Object Identifier 10.1109/OJCSYS.2024.3418306

Concurrent Learning of Control Policy and Unknown Safety Specifications in Reinforcement Learning

LUNET YIFRU [1] AND ALI BAHERI [2]

(Intersection of Machine Learning with Control)

[1]West Virginia University, Morgantown, WV 26505 USA
[2]Rochester Institute of Technology, Rochester, NY 14623 USA

CORRESPONDING AUTHOR: ALI BAHERI (e-mail: akbeme@rit.edu).

ABSTRACT Reinforcement learning (RL) has revolutionized decision-making across a wide range of domains over the past few decades. Yet, deploying RL policies in real-world scenarios presents the crucial challenge of ensuring safety. Traditional safe RL approaches have predominantly focused on incorporating predefined safety constraints into the policy learning process. However, this reliance on predefined safety constraints poses limitations in dynamic and unpredictable real-world settings where such constraints may not be available or sufficiently adaptable. Bridging this gap, we propose a novel approach that concurrently learns a safe RL control policy and identifies the unknown safety constraint parameters of a given environment. Initializing with a parametric signal temporal logic (pSTL) safety specification and a small initial labeled dataset, we frame the problem as a bilevel optimization task, intricately integrating constrained policy optimization, using a Lagrangian-variant of the twin delayed deep deterministic policy gradient (TD3) algorithm, with Bayesian optimization for optimizing parameters for the given pSTL safety specification. Through experimentation in comprehensive case studies, we validate the efficacy of this approach across varying forms of environmental constraints, consistently yielding safe RL policies with high returns. Furthermore, our findings indicate successful learning of STL safety constraint parameters, exhibiting a high degree of conformity with true environmental safety constraints. The performance of our model closely mirrors that of an ideal scenario that possesses complete prior knowledge of safety constraints, demonstrating its proficiency in accurately identifying environmental safety constraints and learning safe policies that adhere to those constraints. A Python implementation of the algorithm can be found at https://github.com/SAILRIT/Concurrent-Learning-of-Control-Policy-and-Unknown-Constraints-in-Reinforcement-Learning.git.

INDEX TERMS STL mining, safe learning, specification-guided reinforcement learning (RL).

I. INTRODUCTION

Reinforcement learning (RL) has risen as a key computational paradigm involving training intelligent agents to make sequential decisions, aiming to maximize some notion of expected return [1]. It has been instrumental in solving complex dynamic problems across a wide range of applications such as autonomous driving, robotics, aviation, finance, etc. [2], [3], [4], [5]. However, deploying RL in practical settings introduces the critical concern of safety, especially in domains, such as autonomous driving and healthcare, where unsafe actions can lead to catastrophic outcomes. Safety refers to the need for systems to operate within acceptable risk parameters, and pertaining to RL, safety is defined by the system's ability to attain the environmental objectives while adhering to safety constraints.

Traditionally, safe RL methods base policy design on either modifying the optimality criterion to include cost as one of the objectives or altering the exploration process [6]. For instance, a prominent approach is the integration of formally defined safety constraints, such as STL, into reward

functions encapsulating critical limits within which RL agents must operate. This approach is particularly appealing, because, unlike classical machine learning models, which are often black-box and obscure, temporal logic formalism offers a precise, human-interpretable language for system behavior. Logically constraining RL has shown promise for generating safe, high-performance policies, however, the effectiveness of this approach hinges on the availability and quality of the predefined safety constraints. Defining such safety constraints can be effectively approached through the utilization of expert knowledge, manually designed by domain experts, or derived from data using computational techniques. However, the reliance on expert knowledge for defining safety constraints can be restrictive and often infeasible, as experts are not always readily available. Furthermore, given the dynamic nature of environments, experts may have a limited perspective, potentially leading to safety constraints that do not fully encapsulate the true environmental conditions resulting in either overly conservative behavior or overlooked risks. On the other hand, computational approaches for mining temporal logic safety specifications depend on the availability of extensive historical datasets, which may not always be accessible, or its acquisition could pose significant risks in safety-critical domains. Overall, specifying exact safety constraints in RL environments is a challenging task, and static, predefined constraints may not be sufficiently adaptable to address the complexities of dynamic real-world environments.

Traditional safe RL methods fall short in designing safe policies in the absence of predefined safety constraints, leaving a critical gap that impedes the broader integration of safe RL into areas lacking such predefined constraints. To bridge this gap, we propose an approach that enables the learning of safe control policies in environments where safety constraints are not explicitly defined *a priori*. Our approach, given a parametric STL (pSTL) specification and two categories of small initial datasets, one populated with safe trajectories and another populated with unsafe trajectories, *concurrently* identifies the pSTL safety parameters that accurately model the environmental constraints and derives an optimal safe RL policy constrained by the learned STL. Our concurrent learning process is facilitated through the input of a human expert who iteratively provides labels to rollout traces generated by executing the learned policy. This allows the extension of the small initial dataset and efficient refining of the pSTL parameter values, steering them towards the accurate constraints, in turn, guiding the RL algorithm towards the optimal safe policy.

Our Contribution: Key contributions of our paper are:

1) We propose a novel framework for concurrently learning safe RL policies and STL safety constraint parameters in an environment where safety constraints are not defined *a priori*.
2) We modify the TD3-Lagrangian constrained RL algorithm to use STL as a constraint specification during policy synthesis.
3) Through extensive evaluations in various safety-critical environments, and comparisons to baseline models, we prove that our framework is able to obtain safe RL policies that maximize rewards while upholding safety constraints, performing comparably to baseline models equipped with predefined safety constraints.

The remainder of this paper is structured as follows. Section II provides a review of related work in the domain of safe RL and STL synthesis, Section III outlines the foundational concepts used in deriving our proposed approach. The problem statement is articulated in Section IV and our methodology is detailed in Section V. Section VII is dedicated to the performance evaluation of our results as compared with baselines and discussion of the implications of our findings as well us the limitations of our work.

II. RELATED WORK

Our work is related to two key areas of research, namely safe RL policy synthesis and formal safety specification learning with additional emphasis on parameter synthesis of pSTL specifications.

Safe RL: In recent years, a diverse range of approaches for safe RL has been proposed, including constrained RL [7], [8], safety layers or shielding [9], [10], [11], and formal methods [12]. A comprehensive overview of safe RL methods is given in [13], [14], [15]. Primal-dual policy optimization [16], [17], [18], a method prominent to our approach, is based on the Lagrangian relaxation procedure and solves a saddle-point problem to iteratively optimize the policy (primal) while adjusting the dual variable. Taking inspiration from primal-dual methods, constrained policy optimization methods outlined in [7], [19] develop trust region methods, which approximately enforce constraints in every policy update by evaluating the constraint based on samples collected from the current policy.

Safe RL methods based on using safety certificates as constraints are also explored. An example of such methods is to constrain the agent's actions by applying the control law of Lyapunov functions, mathematical approaches guaranteeing stability and safety, and then excluding unsafe actions from the action set [20], [21]. Lyapunov method, however, requires prior knowledge of a Lyapunov function, which can be challenging to obtain. Another form of safety certificates are Barrier functions that divide unsafe states and safe states by finding a barrier and starting from a given initial state, ensure that the system will not enter the unsafe set. Some works use barrier certificates as constraints [22], [23].

Another direction explored in a safe RL framework is utilizing Gaussian Process (GP) models. For instance, the SNO-MDP framework optimizes costs within a safe region and maximizes rewards in areas with undefined safety constraints by using GP models to predict unobserved states [24]. Similarly, some research have used GP models to approximate unknown functions for safe exploration [25], and to represent unknown reward and cost functions, ensuring safety with a certain probability and optimizing reward [26].

Conversely, other approaches synthesize safe policies based on reward shaping techniques informed by temporal logic

formulae as constraints. For instance, a safe RL method using linear temporal logic (LTL) as a constraint during policy generation has been suggested in [27], [28]. Signal temporal logic (STL) has been widely used in this context due to its expressiveness in capturing complex time-dependent properties. For example, one notable approach approach employs STL to specify safety and performance requirements and utilizes a constrained policy optimization technique to learn policies that adhere to these requirements [29]. These works demonstrate the effectiveness of using STL as a formal specification language to encode safety constraints and guide the learning of policies that satisfy them. Other methods of modifying the reward function involve replacing the reward function in RL environments with the robustness degree of an STL constraint [30], [31], partial signal rewarding mechanism based on the robustness of a given safety STL specification [32], the τ-CMDP approach that uses Lagrangian relaxation to solve a constrained optimization by using an STL specification as a constraint [33]. Although temporal logic-based methods deliver impressive safety performance, the logical constraints need to be *predetermined* to ensure their success.

Safety Specification Mining: The learning of STL specification can be divided into two categories: learning of the formula template along with the parameters, and learning of the parameters given the formula template/pSTL specifications.

Recently research directions have focused on mining complete STL specifications (both template and parameters) from data. The work in [34], by defining a partial order over the set of reactive STL (rSTL), proposes a passive learning approach that infers an STL specification which serves as a classifier from positive and negative examples. This approach was later extended to an online setting [35] and an unsupervised approach [36]. The authors in [37] propose another passive learning method that uses grid-based signal discretization, clustering of similar signals by similarity of covered cells, translating clusters into equivalent STL formulas, and constructing an STL at the disjunction of cluster STL formulas. Decision trees are another widely explored alternative for STL mining and they could be based on offline supervised learning from positive and negative examples [38], [39], online supervised learning [40], and offline unsupervised learning [41]. An approach for mining STL specifications from positive examples guided by robustness metrics is introduced by the work in [42]. Evolutionary algorithms have also been explored in this domain [42], [43], [44], using genetic operators to evolve candidate formulas into effective classifiers.

Several approaches have been explored, specifically targeting the parameter synthesis of pSTL specifications. While the computation of the exact validity domain of a pSTL specification has been investigated to address parameter synthesis [45], it is evident that this method incurs exponentially increasing computational costs. To mitigate this, this method is extended towards approximating validity domains using run-time verification methods in tandem with search techniques by the same authors in [45]. The method outlined in [42] proposes a passive learning of pSTL parameters from positive examples by introducing the notion of a differentiable tightness metric for STL specification satisfaction, and uses gradient-based methods to search over the parameter space. The STLCG framework bridges pSTL parameter mining with machine learning by presenting a novel integration of computation graphs from the machine learning domain to evaluate the robustness of STL formulas and learn pSTL parameters [46]. Conversely, the works in [47], [48] adopt an active learning strategy to mine pSTL parameters using signals generated by dynamic models through an iterative process that computes candidate STL specifications and utilizes falsification methods to search for counterexamples generated by the model. This approach is limited in that it necessitates the availability of a dynamic model capable of generating new signals. The method of logical clustering combines pSTL parameter inference with unsupervised learning [49]. It projects signals to template parameters within their validity domain, uses clustering to group similar signals, and defines an STL formula for each cluster. For our approach, we took inspiration from the ROGE framework, where the parameter identification is addressed using Bayesian optimization (BO) [43].

Recent research directions suggest methods for learning safety certificates, and similarly to our proposed approach, some simultaneously learn safe control policies and safety certificates [22], [50]. However, to our knowledge, there are no works on concurrently learning a temporal logic based safety specification and a constrained RL control policy.

III. PRELIMINARIES

A. SIGNAL TEMPORAL LOGIC

STL is a formal language for specifying temporal properties of real-valued signals. STL extends Boolean logic with temporal operators, allowing the expression of complex requirements and constraints on system behavior over time. The key operators in STL include negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), until (U_I), eventually (F_I), and always (G_I), where I represents a time interval. For example, consider a robot with position $x(t)$ at time t. We can express the following requirements using STL:

$$\phi_1 = G_{[0,\infty)}(x(t) \geq 0 \wedge x(t) \leq 100) \tag{1}$$

which means the robot's position should always be within [0, 100], and

$$\phi_2 = F_{[0,10]}(x(t) = 50) \tag{2}$$

which means the robot should eventually reach position 50 within 10 seconds. STL provides an expressive way to define temporal specifications for complex systems, enabling the formal verification and monitoring of system behavior.

STL can be interpreted using Boolean semantics (True/False), as well as quantitative semantics (a real-value) [51]. The quantitative semantics of an STL formula introduces the notion of a robustness value $\rho(\phi, s_t)$ that quantifies the degree

TABLE 1. STL quantitative semantics.

Formula	Robustness value
$\rho(s_t, >)$	$\rho_{\max}$
$\rho(s_t, \mu_c)$	$\mu(x_t) - c$
$\rho(s_t, \neg\phi_1)$	$-\rho(s_t, \phi_1)$
$\rho(s_t, \phi_1 \wedge \phi_2)$	$\min(\rho(s_t, \phi_1), \rho(s_t, \phi_2))$
$\rho(s_t, \phi_1 \vee \phi_2)$	$\max(\rho(s_t, \phi_1), \rho(s_t, \phi_2))$
$\rho(s_t, \phi_1 \Rightarrow \phi_2)$	$\max(-\rho(s_t, \phi_1), \rho(s_t, \phi_2))$
$\rho(s_t, \mathrm{F}_{[a,b]}\phi_1)$	$\max_{t' \in [t+a, t+b]} \rho(s_{t'}, \phi_1)$
$\rho(s_t, \mathrm{G}_{[a,b]}\phi_1)$	$\min_{t' \in [t+a, t+b]} \rho(s_{t'}, \phi_1)$
$\rho(s_t, \phi_1 \mathcal{U}_{[a,b]}\phi_2)$	$\max_{t' \in [t+a, t+b]} \Big(\min\{\rho(s_{t'}, \phi_2), \min_{t'' \in [t, t']} \rho(s_{t''}, \phi_1)\}\Big)$

to which a formula ϕ is violated or satisfied by signal s_t, and is given in Table 1.

Parametric Signal Temporal Logic (pSTL): pSTL is an extension of STL where only the structure/template of the STL formula is given, i.e., the STL formula is parameterized and all the time-bounds $[t_1, t_2]$ for temporal operators and the constants μ for inequality predicates are replaced by free parameters [45]. Parameter valuation $v(p)$ represents a mapping that assigns values to all time and space parameters p of the pSTL. For a given pSTL formula ϕ_p with parameters p, the valuation of every parameter assignment $v(p)$ results with a corresponding STL formula $\phi_{v(p)}$. In this paper, we only consider unbounded temporal operators, those with time bounds $[0, \infty]$, and will thereby only be concerned with deriving valuations for space parameters.

B. REINFORCEMENT LEARNING

RL is an optimization problem on a Markov decision process (MDP), a tuple $M = (S, A, \mathbb{P}, R, \gamma)$ that defines an environment with states $s \in S$, actions $a \in A$, transition probabilities $\mathbb{P}(s'|s, a) = \mathbb{P}\{S_{t+1} = s' | S_t = s, A_t = a\}$, a reward function $R(s, a) = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$, and a discount factor $\gamma \in [0,1]$ prioritizing short term rewards [52]. An agent's behavior is defined by a policy π which maps states to a probability distribution over the actions $\pi : S \to P(A)$, and its objective is to maximize the total discounted return $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$. The state-action value function $Q_\pi(s, a)$ is defined as the expected return starting from state s, taking action a and thereafter following policy π

$$Q_\pi(s, a) = \mathbb{E}_\pi \Big[G_t | S_t = s, A_t = a \Big] \tag{3}$$

Q-Learning is a foundational value-based algorithm that operates by iteratively approximating the state-action value function $Q(s, a)$ based on the Bellman optimality equation [53]. The update rule is given by

$$Q(s, a) = Q(s, a) + \alpha \big[R(s, a) + \gamma \max(Q(s_{t+1}, a_{t+1})) - Q(s, a) \big] \tag{4}$$

For continuous control problems, deep Q-networks (DQN) [54] incorporates neural networks to approximate the Q-value $Q_\theta(s, a)$ parameterized by θ. The optimal parameter θ^* can be learned using stochastic gradient descent

$$\theta^* = \arg\min_\theta \mathbb{E} \big[(y_t - Q_\theta(s_t, a_t))^2 \big] \tag{5}$$

where $y_t = r(s_t, a_t) + \gamma \max_a Q_\theta(s_{t+1}, a)$ is the temporal difference (TD) target used to stabilize training and maintain a fixed objective over multiple updates. Policy based methods directly learn a policy π_θ parameterized by θ, that maximizes the expected return from a start state. The parameter θ is updated using gradient ascent

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_\theta J(\pi_\theta)|_{\theta=\theta_t} \tag{6}$$

where α is the learning rate, and $\nabla_\theta J(\pi_\theta)$ is computed following the policy gradient (PG) theorem [52]

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \big[Q^\pi(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t | s_t) \big] \tag{7}$$

TD3 [55], an algorithm relevant to our proposed approach, is a class of *actor-critic* methods proposed to address the overestimation error caused by deep deterministic policy gradient (DDPG) [56]. To achieve that, TD3 implements clipped double Q-Learning, delayed policy and target network updates, and target policy smoothing. With these updates, the TD target to which both Q functions regress is given by

$$y_t = r(s_t, a_t) + \gamma \min_{i=1,2} Q_{\theta_i'}(s_{t+1}, \pi_{\phi'}(s_{t+1}) + \epsilon) \tag{8}$$

where $Q_{1,2}$ are the critic networks, π is the actor network, θ_i' and ϕ' are the target critic and target actor network parameters, respectively, and $i = 1, 2$ represents the i-th target critic networks, and ϵ is the clipped Gaussian noise.

Constrained RL: It is a branch of RL that is concerned with maximizing reward while also satisfying environmental safety constraints. Safe RL is modeled as a constrained MDP (CMDP) [57], which is an extension of the standard MDP with an additional constraint set $\mathcal{C}$. The optimal policy in constrained RL is expressed as

$$\pi^* = \arg\max_{\pi \in \Pi_c} J^{\mathcal{R}}(\pi) \tag{9}$$

where $J^{\mathcal{R}}(\pi)$ is the objective function and Π_c is a set of constraint satisfying policies $\Pi_C = \pi : J^{\mathcal{C}}(\pi) \leq d$.

C. BAYESIAN OPTIMIZATION

BO is a powerful strategy for the optimization of black-box functions that are intractable to analyze and are often non-convex, nonlinear, and computationally expensive to evaluate [58]. It has been widely applied across multiple fields, such as hyperparameter tuning in machine learning models [59], control and planning [60], [61], [62], robotics [63], and materials design [64]. BO offers a principled technique to direct a search of the global optimum of an objective function by building a probabilistic model of the objective function, called the surrogate function, that is then searched efficiently guided by an acquisition function. GPs are nonparametric models

employed in BO to impose a prior over the objective function. GP is used to maintain a belief over the design space simultaneously modeling the predicted mean $\mu(p)$ and the epistemic uncertainty $\sigma(p)$ at any parameter set p in the input space. GPs are defined by their mean function $\mu(\mathbf{p})$, which is initially assumed to be 0, and covariance function $k(p, p')$

$$f(p) \sim \mathcal{GP}(\mu(p), k(p, p')) \tag{10}$$

The covariance function $k(p, p')$ is also called the "kernel", and is often given by a squared exponential function

$$k(p, p') = \exp\left(-\|p - p'\|^2\right) \tag{11}$$

For any new set of parameters p_* for the pSTL, the GP model provides a predictive distribution with mean and variance given by [65]

$$\begin{aligned} \mu(p_*|p) &= \mu(p_*) + \mathbf{K}_*^T \mathbf{K}^{-1}(\mathbf{y} - \mu(p)) \\ \sigma^2(p_*|p) &= \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_* \end{aligned} \tag{12}$$

where $\mathbf{K} = k(p, p)$, $\mathbf{K}_* = k(p, p_*)$, and $\mathbf{K}_{**} = k(p_*, p_*)$.

Acquisition functions guide how the parameter space is explored by observing the predicted mean and variance of a sample parameter set from the GP model, given in (12). We use the expected improvement (EI) as the acquisition function. EI accounts for the size of improvement over the current best observation when choosing the next candidate parameter set. The utility of EI lies in its ability to explicitly encode a trade-off between pursuing regions of high uncertainty (exploration) and regions with a potential for high objective function values (exploitation) by quantifying the expected amount of improvement. The EI for a parameter set p, given the current observations $\mathcal{D}$, is defined as follows [65]

$$EI(p) = \mathbb{E}[\max(0, f_{\min}(p) - f(p))|p, \mathcal{D}] \tag{13}$$

where $f_{\min}$ is the minimum value observed so far. Through iterative implementations of the acquisition function, the GP model refines its predictions, steering the optimization process towards the global optimum of the objective function.

IV. PROBLEM STATEMENT AND FORMULATION

We consider the problem of safe RL policy synthesis in an environment where safety constraints are unknown *a priori*. Our ultimate objective is to concurrently learn accurate parameters of the pSTL specification that define the environmental constraints and an optimal policy such that the policy adheres to the learned STL safety constraint while achieving high returns.

We initialize the problem with a small initial labeled datasets, safe trajectories D_s and unsafe trajectories D_{us}, and a pSTL safety specification template ϕ_p. Within the initial labeled datasets D_s and D_{us}, we find 10 safe traces x_s and 10 unsafe traces x_{us}, respectively, which are initially manually selected by the human expert from historical safe and unsafe runs within the given environment. The small size requirement (only 10 trajectories each for both the safe and unsafe datasets) of these initial datasets relieves the difficulties of acquiring a large pool of expressive historical datasets that is required for learning of accurate pSTL parameters from data, especially in safety-critical environments. Our approach, instead, takes on a data-efficient strategy that starts with a small initial dataset and iteratively adds to it additional expert-labeled data as necessary to strategically acquire high-quality parameter estimates using the smallest applicable volume of labeled data.

The pSTL parameter learning process takes on a supervised learning approach that requires obtaining parameter valuations for the pSTL, such that the synthesized STL formula is satisfied by safe trajectories and is not satisfied otherwise. The goal is to learn sufficiently accurate set of parameters of the pSTL with smallest viable dataset size. The parameter synthesis problem focuses on solving a minimization problem for a black-box objective function with the aim of finding optimal parameters of the pSTL specification using a labeled dataset.

The policy learning step requires solving the optimization problem in CMDPs expressed as

$$\begin{aligned} &\max_{\pi_\theta} J^{\mathcal{R}}(\pi_\theta) \\ &\text{s.t. } J^{\mathcal{C}}(\pi_\theta) \leq d \end{aligned} \tag{14}$$

where $J^{\mathcal{R}}$ is a reward-based objective function, $J^{\mathcal{C}}$ is a cost-based constraint function, and d is the threshold for safety. Within our framework, we formulate the cost objective function $J^{\mathcal{C}}$ by infinite-horizon, undiscounted expected cumulative costs

$$J^{\mathcal{C}}(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{\infty} c_t\right] \tag{15}$$

where c_t is the cost at time-step t and is computed using the STL safety constraint. The choice to employ undiscounted cumulative costs for the cost objective function in (15) is motivated by the substantial real-world implications of actual cost values within the algorithm. Specifically in our algorithm, which will be described in detail throughout Section V, we use the undiscounted sum of costs within each episode computed by (15) to update the Lagrange multiplier. Therefore, opting for a discounted sum would result in a reduced episodic cost value for Lagrange multiplier updates, potentially compromising their effectiveness in controlling constraint violations within the cost limit.

The learned policy is considered optimal if achieves its performance objectives while also generating rollout traces that demonstrate constraint-abiding behavior. *Rollouts* are sequences of actions executed by an agent from a specific state under the learned policy, thus serve as direct indicators of the policy's safety.

V. METHODOLOGY

In our proposed framework, the learning of a safe RL policy in an environment with unknown safety constraint parameters is separated into two components: optimization of the

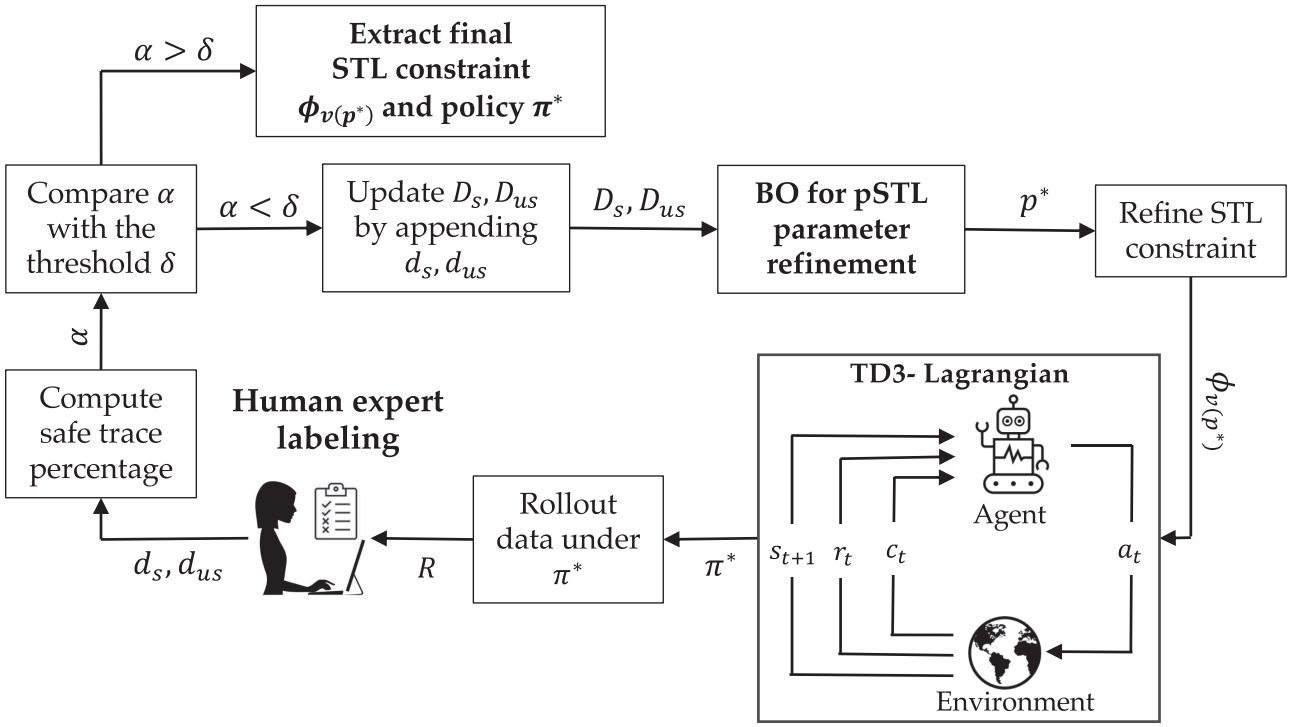


FIGURE 1. Schematic representation of the integrated framework for concurrently learning STL constraint parameters and optimal policies. The framework applies BO for STL parameter mining, TD3-Lagrangian for policy learning, and incorporates human expert for labeling rollout traces to be used in refining the learned constraint parameters and policy. Once the percentage of safe traces in a rollout dataset α is higher than the threshold value δ, convergence is achieved, and the final policy and STL constraint are extracted.

parameters for the given pSTL safety constraint using labeled data, and safe RL policy optimization with logical constraints. These two components are integrated through the assistance of a human expert, who contributes by labeling rollout traces derived from an RL policy. The labeling process involves the human expert designating each rollout trace as "safe" or "unsafe" based on whether each trace adheres to or violates safety constraints in an environment. In other words, a trajectory is labeled "safe" if and only if all the states in the trajectory are safe as deemed by the human expert, and labeled "unsafe" otherwise. This labeling process is crucial, as it yields the labeled dataset required for the iterative refinement of the pSTL parameters.

We frame this concurrent learning problem as a bilevel optimization, an optimization approach that contains two levels of optimization tasks where one optimization task, the lower level, is nested within the other, the upper level [66]. These two levels of optimization each address one each of the two learning components in our framework: the upper level is dedicated to the pSTL parameter synthesis while the lower level is dedicated to the constrained RL policy optimization. The mathematical representation of this bilevel optimization task is given by

$$\begin{aligned} &\arg\min_{p} f(\phi_{v(p)}, \pi^*(\phi_{v(p)})), \\ &\text{s.t.} \quad \pi^*_\theta(\phi_{v(p)}) \in \arg\max_{\pi_\theta \in \Pi_c} J^{\mathcal{R}}(\pi_\theta(\phi_{v(p)})) \end{aligned} \tag{16}$$

where f is the upper-level objective function with optimization variable p, which is a set of parameter values to the pSTL ϕ_p, and π is the lower-level optimization objective with optimization variable θ. $\pi^*(\phi_{v(p)})$ represents the optimal policy under the given STL constraint $\phi_{v(p)}$, and $\phi_{v(p^*)}$ represents the complete STL after the valuation of pSTL ϕ_p with optimal parameters p^*. In (16), the upper-level objective f depends on both the pSTL parameters p, and the solution $\pi^*(\phi_{v(p)})$ of the lower-level objective. A schematic of our framework is illustrated in Fig. 1.

A. STL CONSTRAINT PARAMETER LEARNING

The upper-level of the bilevel optimization framework, a BO process, is designed to obtain the optimal parameters p^* of a given pSTL formula ϕ_p (an STL formula template) through the minimization of an objective function f. The parameter learning process initiates with the pSTL formula ϕ_p and the two initial safe and unsafe datasets D_s, D_{us}. Using these labeled datasets, BO is carried out to learn the best parameter configuration for pSTL ϕ_p such that the final STL best classifies between D_s and D_{us} in terms of robustness degree. The rationale behind designing the objective function for learning the optimal parameters of the pSTL specification ϕ_p is as follows: if a candidate STL $\phi_{v(p)}$ represents the true environmental constraints, any trace labeled "safe" by the human expert x_s should have a positive robustness value and any trace labeled "unsafe" x_{us} should have a negative robustness value

with respect to $\phi_{v(p)}$. Under this consideration, the objective function used for pSTL parameter optimization f is mathematically defined by

$$f(\phi_{v(p)}) = \frac{1}{2}\left(\frac{N_{\rho(\phi_{v(p)})^-|x_s}}{N_{x_s}} + \frac{N_{\rho(\phi_{v(p)})^+|x_{us}}}{N_{x_{us}}}\right) \tag{17}$$

where $\phi_{v(p)}$ is the STL formula obtained after the parameter valuation $v(p)$ of pSTL ϕ_p, x_s and x_{us} are traces sampled from the datasets containing safe and unsafe traces, respectively. N_{x_s} and $N_{x_{us}}$ are the total number of safe and unsafe traces within their respective datasets. The first term within the parenthesis in (17) $\frac{N_{\rho(\phi_{v(p)})^-|x_s}}{N_{x_s}}$ represents the ratio of safe traces x_s with a negative robustness value ρ^- with respect to $\phi_{v(p)}$ (false negative rate), and the second term $\frac{N_{\rho(\phi_{v(p)})^+|x_{us}}}{N_{x_{us}}}$ represents the ratio of unsafe traces x_{us} with a positive robustness value ρ^+ with respect to $\phi_{v(p)}$ (false positive rate). This essentially computes the balanced misclassification rate, derived from the complement of the balanced accuracy score [67], a metric that computes classification accuracy in datasets with imbalanced distribution between classes. It is important to note that in scenarios with extremely small initial datasets or highly imbalanced data, there is a potential for division by zero in (17) if the denominators become zero. To mitigate this issue, a small constant value can be added to the denominators to ensure numerical stability.

Upon convergence, this optimization process will identify the optimal set of parameters p^* for the pSTL that minimize the objective function, f in (17), yielding the final STL safety constraint $\phi_{v(p^*)}$, which we denote ϕ_{cost}, the output of the upper-level optimization. The relationship between ϕ_{cost} and the objective function given in (17) is expressed as

$$\phi_{cost} = \phi_{v(p^*)} = \arg\min_{p} f(\phi_{v(p)}) \tag{18}$$

ϕ_{cost} is of type G($\neg(\psi_{cost})$) where ψ_{cost} characterizes unsafe behavior, and ϕ_{cost} conveys *"always-not-unsafe"* i.e. $G(\neg(\psi_{cost}))$, signifying that ϕ_{cost} universally opposes the occurrence of ψ_{cost}.

The pSTL format used in our studies is of the form $G\neg(\psi_{cost})$ where ψ_{cost} varies with each case-study. As mentioned in Section III-A, we only consider the unbounded temporal operator G with time bounds $[0, \infty]$. While we recognize that the format simplifies the STL representation and a simple Boolean formula could suffice in capturing safety constraints in this case, the introduction of STL into our problem definition is pivotal for several reasons. Firstly, it enables parameterization of safety constraints and facilitates learning of unknown parameters from data by sharing the same template across a variety of STL safety constraints. Additionally, it offers quantitative semantics through robustness metrics providing a continuous and informative guiding parameter optimization, and smoothly integrating into the policy optimization framework. Finally, although we employ an unbounded temporal operator, it maintains the interpretability of the STL safety constraint for humans by conclusively indicating that the predicate under the temporal operator must be universally satisfied.

B. POLICY LEARNING

The lower-level of the bilevel optimization framework consists of a logically-constrained, safe RL policy optimization. This phase follows the process of pSTL parameter optimization, detailed in Section V-A, and uses the STL generated therein as its input. For this stage, we solve the optimization problem for CMDP introduced in (14) by utilizing the Lagrangian-variant of the twin delayed deep deterministic policy gradient (TD3) algorithm, TD3-Lagrangian. The background for the TD3 algorithm is given in Section III-B, and throughout this section, we provide an overview of Lagrangian methods, later discussing the development of the TD3-Lagrangian algorithm.

Lagrange multiplier method is used to transform a constrained optimization problem into an equivalent unconstrained optimization problem through Lagrangian relaxation procedure that introduces adaptive penalty coefficients to enforce constraints [68]. Using this method, (14) is transformed into the equivalent unconstrained min-max optimization problem

$$\max_{\theta}\min_{\lambda\geq 0} \mathcal{L}(\theta, \lambda) = J^{\mathcal{R}}(\pi_\theta) - \lambda(J^{\mathcal{C}}(\pi_\theta) - d) \tag{19}$$

where λ is the Lagrange penalty coefficient, $J^{\mathcal{R}}$ is the reward objective function, $J^{\mathcal{C}}$ is the constraint objective function, and d is the maximum allowable cumulative cost. Eq 19 is then solved by gradient ascent on θ and descent on λ to result with the optimal values θ^* and λ^*.

An adaptation of the Lagrange multiplier method to the TD3 algorithm is given in [69], deriving TD3-Lagrangian. TD3-Lagrangian incorporates an additional cost critic network to the original TD3 architecture to estimate the cost value function $Q^{\mathcal{C}}$, and alters the loss function to incorporate a constraint satisfaction component using a Lagrangian multiplier

$$L = -Q^V(\pi_\theta, s) + \lambda Q^{\mathcal{C}}(\pi_\theta, s) \tag{20}$$

where Q^V is the minimum value of the two reward critic network outputs, Q^C is the value of cost critic network, and π is the policy network. The penalty coefficient λ is updated by minimizing the penalty loss $\lambda' = \lambda + \eta(J^{\mathcal{C}}(\pi_\theta) - d)$, where η is the learning rate. When $J^{\mathcal{C}}$ exceeds the constraint threshold d, λ is increased to prioritize cost minimization.

In our approach, we propose a novel modification to the TD3-Lagrangian architecture. While retaining the classical definition of the reward function for each environment, we redefine the cost function logically, using an STL specification ϕ_{cost}. As stated prior, ϕ_{cost} is the STL safety specification derived through the process outlined in Section V-A and within it is ψ_{cost}, the STL specification describing the unsafe behavior at each given time step. Using ψ_{cost} as the safety constraint,

we compute the cost at each step $c(s_t, a_t)$, using the quantitative semantics of STL given in Section III-A, as follows

$$c(s_t, a_t) = \begin{cases} 1, & \text{if } \rho(\neg(\psi_{cost}), s_t) < 0 \\ 0, & \text{if } \rho(\neg(\psi_{cost}), s_t) \geq 0 \end{cases} \tag{21}$$

where $\rho(\neg(\psi_{cost}), s_t)$ is the robustness value of the current state s_t with respect to the STL $\neg(\psi_{cost})$. This equation is interpreted as follows. The cost $c(s_t, a_t)$ is assigned to 1 if $\rho(\neg(\psi_{cost}), s_t) < 0$, indicating the safety constraint has been violated at state s_t, and to 0 otherwise. The usage of $\neg(\psi_{cost})$ instead of ϕ_{cost} at this stage is deliberate as the former describes behavior at a specific time step without employing temporal operators, while the latter incorporates the "always" temporal operator to characterize behavior across the entire duration of an episode. We use this STL robustness-based cost values to compute J^C, which is then used in the policy optimization process by minimizing the loss function in (20). Once we have identified an optimal policy π^*, we generate a dataset of rollout traces R by executing the learned policy, which we then pass on R to the human expert for labeling.

C. HUMAN FEEDBACK MECHANISM

A crucial role is played by a human expert who provides labels to the rollout traces generated through the execution of the RL policy optimized through the process outlined in Section V-B. This labelling process is essential to our framework in gradually and iteratively refining the parameter assignment for the pSTL. The quality of the parameters assigned is reliant on the volume of labeled dataset, and while the richness of data facilitates this process, extensive human labeling effort to amass large datasets is impractical. Hence, our strategy focuses on attaining sufficiently accurate pSTL parameters with the minimal necessary data. This is achieved by the human expert labeling only a small number of traces at each iteration, which are then incrementally added to the existing dataset of labeled data from previous iterations. The phased acquisition of data across different iterations of RL policy rollouts ensures that each new rollout data set is attained from a unique policy and contributes unique and essential information to the learning process, enhancing the overall quality and diversity of the dataset used for pSTL parameter learning.

During our experiments, we have implemented an automated process for labeling the rollout traces, which involves of computing the robustness value of each trace within the rollout set with respect to the True STL safety constraint ϕ_{true}. The human labeling process is given using the the satisfaction relation $\models$ between a trace from the rollout dataset x and an STL formula ϕ_{true} as follows

$$L(x) = \begin{cases} 1, & \text{if } x \models \phi_{true} \\ 0, & \text{if } x \not\models \phi_{true} \end{cases} \tag{22}$$

where $L(x)$ is the label assigned to trace x sampled from the rollout dataset, $\phi_{true} = G_{[0:T]}(\neg(\psi_{true}))$ is the general template we use for any STL safety constraint in which ϕ_{true} is the true environment-specific STL formula exhibiting unsafe behaviour. Eq 22 states that a label of 1 is assigned if a trace from the rollout dataset x satisfies ($\models$) the True STL safety constraint ϕ_{true}, i.e. $\rho(\phi_{true}, x) > 0$, and a label of 0 is assigned otherwise.

It is important to note that the use of the True STL safety constraint for automation purposes is not to be confused with the algorithm having the knowledge of this True STL environmental constraint beforehand. In real-world applications, as per the basis of our problem statement, the actual safety constraint remains unknown to the algorithm and is only used for rapid and efficient experimentation of our framework across various case studies, to which we do not have an actual human expert in those areas to label the traces. Therefore, due to the unavailability of the True environmental constraint in practical implementations of our framework, the involvement of the human expert in manually labeling the traces is integral. After traces are labeled, those identified as safe by the human expert are allocated in the safe dataset d_s, whereas those labeled as unsafe are allocated to the unsafe dataset d_{us}. Finally, the percentage of safe rollout traces α within the rollout dataset R is computed by

$$\alpha = \left(\frac{N_{d_s}}{N_{d_s} + N_{d_{us}}} \right) \tag{23}$$

where α is the percentage of safe traces, N_{d_s} and $N_{d_{us}}$ are the respective number of traces in the safe dataset d_s and unsafe dataset d_{us}. The sum $N_{d_s} + N_{d_{us}}$ is equal to the total number of traces in the rollout dataset N_R.

Our framework is outlined in Algorithm 1. The algorithm requires as input initial datasets, D_s and D_{us}, populated with safe traces and unsafe traces, respectively, and the pSTL specification ϕ_p, along with the number of rollout traces at each iteration n_R, the total RL training steps n_s, and the user-specified threshold for the minimum satisfactory percentage of safe traces within a rollout dataset δ. The algorithm initiates by using BO to optimize parameters to the pSTL from the initial dataset, then optimizes a policy using TD3-Lagrangian algorithm constrained by the learned STL. It then proceeds to generate rollout traces from this policy, which are subsequently labeled by a human expert. A key metric, α, is then calculated representing the percentage of "safe" rollout traces as labeled by the human expert amongst the entire rollout dataset and is compared to δ at every iteration.

This iterative process is repeated until convergence, which is achieved when α is greater than the user-specified, minimum threshold for the percentage of safe rollout traces δ i.e. $\alpha > \delta$. If the convergence criteria has not been met, d_s and d_{us} are appended to the initial datasets D_s and D_{us} respectively, to then serve as inputs for the next iteration to generate new and refined STL parameters. If the convergence criteria has been met, the outputs of the algorithm, an STL with optimal parameter values $\phi_{v(p^*)}$, and the optimal policy π^* with respect to the learned STL, are extracted.

Algorithm 1: Joint Learning of Policy With Constraints.

Input: $D_s, D_{us}, \phi_p, n_R, n_s, \delta$
Output: p^*, π^*
Compute α from initial datasets D_s, D_{us} using Eq. (23)
while $\alpha < \delta$ **do**
  $p^* \leftarrow$ Employ BO to find optimal parameters of ϕ_p using D_s, D_{us} by minimizing Eq. (17)
  $\phi_{v(p^*)} \leftarrow$ pSTL valuation using optimal parameters p^*
  for $i \leftarrow 1$ to n_s **do**
    $\pi^* \leftarrow$ Employ TD3-Lagrangian to optimize safe policy under STL constraint $\phi_{v(p^*)}$
  end for
  $R \leftarrow$ Generate n_R rollout traces under π^* and generate dataset of the traces
  for Trace $\in R$ **do**
    Human expert provides a "safe"/"unsafe" label to Trace
    if Trace is labeled safe **then**
      Store in safe dataset d_s
    else if Trace is labeled unsafe **then**
      Store in unsafe dataset d_{us}
    end if
  end for
  $\alpha \leftarrow$ Compute percentage of safe traces in R from datasets d_s, d_{us} following Eq. (23)
  Append d_s, d_{us} **to** $D_s, D_{us} \leftarrow$ Extend initial dataset with new labeled rollout traces
end while

VI. CASE STUDIES

We implement our concurrent learning framework across a series of case studies described throughout this section. The case studies are presented with progressively increasing levels of complexity resulting from the corresponding number of learnable parameters. All case studies were performed on Safety-Gymnasium environments [70], a safe RL benchmark that comprises of several safety-critical tasks in continuous control environments where agents and tasks are inherited from safety-gym [71] and MuJoCo physics simulator [72].[1]

A. CASE STUDY 1: SAFE VELOCITY - HALF CHEETAH

The Half Cheetah environment provided by [72], [73] features a two-dimensional, half-body of a cheetah consisting of 9 body parts and 8 joints connecting them as shown in 2. The state space of this environment includes the positions, angles, velocities, and angular velocities of the cheetah's joints and segments whereas the action space is defined by the torques applied to these joints. The primary objective for this agent is to apply torque on the joints to make the cheetah run in the forward direction to achieve maximum speed, u. There is, however, a control cost penalty applied to restrict the agent from taking too large actions. Overall, the reward allocated to the agent is based on the forward movement and control cost penalty, which is calculated as the weighted sum of the squares of the actions (torques)

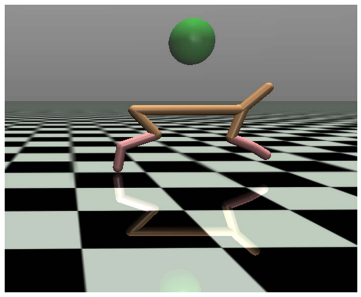


FIGURE 2. Safe velocity test environment with the half cheetah agent.

$$r_t = \left(w_f \cdot \frac{x_{t-1} - x_t}{d_t}\right) - \left(w_c \cdot \sum(a_t^2)\right) \tag{24}$$

where w_f if the forward reward weight, x_{t-1} and x_t are the x-coordinates of the agent before and after applying action a_t, respectively, d_t is the time between actions, and w_c is the control cost weight. A refined adaptation of the Half Cheetah environment is detailed in [70], introducing an additional constraint on the agent's maximum allowable x-velocity. We use this adaptation for our experimentation on this environment with a the pSTL safety constraint given by

$$\phi_{\text{cost}} = G(\neg(u_a > u_{\max})) \tag{25}$$

where u_a is the agent's x-velocity, and u_{max} is the the maximum allowable (safe) x-velocity for the agent. The pSTL given in (25) provides one parameter to be learned using our framework, u_{max}.

B. CASE STUDY 2: SAFE NAVIGATION - CIRCLE

Safety navigation-circle offers a scenario in which an agent is situated randomly within a given x and y bounds at the start of an episode. The objective of the agent is to move in a circular motion within the circle area, while also attempting to stay at the outermost circumference of the circle. In doing so, the agent must also avoid going outside safety boundaries that intersect with the circle area as depicted in Fig. 3. We use level 1 of this environment as given by [70], which consists of 2 boundaries, situated on the left and right side of the of the center, respectively, and the *point* agent [71], a simple robot constrained to a 2D plane with two actuators, one for rotation and the other for forward/backward movement. The reward function for this environment is given as [70]

$$r_t = \frac{1}{1 + |r_a - r_c|} \cdot \frac{(-u \cdot y + v \cdot x)}{r_a} \tag{26}$$

where r_t is the current time-step reward, u, v, are the $x - y$ axis velocity components of the agent, x, y are the $x - y$ axis coordinates of the agent, r_a is the Euclidean distance of the

[1]Our implementation for this research is available at our GitHub repository: https://github.com/SAILRIT/Concurrent-Learning-of-Control-Policy-and-Unknown-Constraints-in-Reinforcement-Learning.git

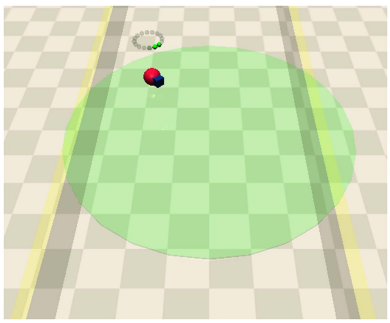


FIGURE 3. Circular navigation environment with 2 boundaries in the x direction (in yellow) and the safe navigation area (in green).

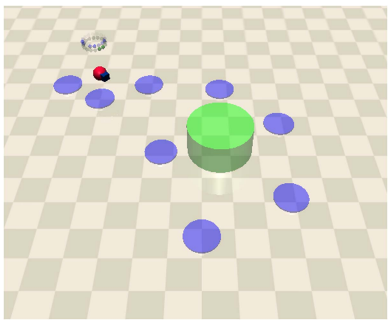


FIGURE 4. Goal navigation environment with eight hazards (in blue), and one goal location (in green).

agent from the origin, r_c is the radius of the circle geometry. Intuitively, the agent moves as far out as it can in outermost circumference of the circle, and the faster the speed, the higher the reward. The predefined pSTL that represents the STL safety constraint template of this environment is given by

$$\phi_{\text{cost}} = G\left(\neg\left(\left(x_a < x_{\mathcal{T}^-}\right) \vee \left(x_a > x_{\mathcal{T}^+}\right)\right)\right) \tag{27}$$

where x_a represents the agent's x position, $x_{\mathcal{T}^+}$ and $x_{\mathcal{T}^-}$ represent the x threshold locations where the boundaries in the positive and negative x directions, respectively, are located, measured from the center.

The pSTL specification in (27), intuitively describes that the agent's x location should never move past the boundaries in either direction of the center. The (initially unknown) safety constraint parameters for this environment are the threshold values $x_{\mathcal{T}^+}$ and $x_{\mathcal{T}^-}$, which provides us with two learning parameters for this pSTL to obtain an STL safety specification.

C. CASE STUDY 3: SAFE NAVIGATION - GOAL

Safe Navigation-Goal is another environment introduced in [70] that offers a scenario in which an agent is randomly positioned at the start of an episode, with the objective of navigating to a designated goal location within the environment while circumventing circular hazard locations. Upon reaching the designated target location, this location is reassigned randomly to a new goal location and the agent continues to navigate towards the updated target. This process continues until the maximum episode steps is reached. We implement level 1 of this environment, which comprises of 8 hazard locations and one goal location. Similarly to Section VI-B, we use the "*point*" agent within this environment. A snapshot of the environment is shown in Fig. 4. The reward function for this environment is defined as [70]

$$r_t = (d_{\text{t-1}} - d_{\text{t}}).\beta \tag{28}$$

where r_t represents the reward at the current time step, $d_{\text{t-1}}$ and d_{t} represent Euclidean distances between the agent a and the goal g at the previous time step $t-1$ and the current time step t, respectively, and β is a discount factor. When $d_{\text{t-1}} > d_{\text{t}}$, it indicates that the agent is moving closer to the goal, and $r_t > 0$ as a result and vice versa.

The pSTL safety constraint that is provided for this environment is given by

$$\phi_{\text{cost}} = G\left(\neg\left(\bigvee_{i=1}^{8}\left(\sqrt{(x_a - x_{h,i})^2 + (y_a - y_{h,i})^2} < r_h\right)\right)\right) \tag{29}$$

where $i = 1, 2, \ldots, 8$ represents each of the 8 hazards, $x_{h,i}$ and $y_{h,i}$ are the x and y coordinates of hazard i's centroid, respectively, x_a and y_a are the agent's current x and y location, respectively, and r_h represents the radius of the hazards.

The pSTL expression provided in (29) can be interpreted as follows. The Euclidean distance between the agent and any of the eight hazards should never be less than the hazard's radius. The initially unknown safety constraint parameters for this environment are the x and y coordinates of the centroids of the hazards: $x_{h,i}$ and $y_{h,i}$, where $i = 1, 2, \ldots, 8$ represents each of the 8 hazards, adding up to 16 unknown parameters to learn.

We evaluate key performance metrics of our approach through the two primary tasks: 1) optimization of safe policies, and 2) synthesis of pSTL parameters. In regards to safe policy optimization, we first show convergence during policy optimization for all case-studies alongside a comparative analysis of cumulative rewards and costs per episode at the end of training against established baselines. Furthermore, we rigorously evaluate the policies by examining the safety of rollout traces generated under each policy. Pertaining to the synthesis of pSTL formula parameters, our evaluation focuses on comparing the learned parameters against the True environmental safety parameters, which are unknown *a priori* to the algorithm. We also evaluate the classification accuracy of the learned STL safety constraints with respect to labeled data. We compare our algorithm with:

- *Baseline 1:* Unconstrained RL policy optimization In this baseline, we perform RL policy optimization without any explicit consideration of safety constraints. The objective is to maximize the expected cumulative reward, and the agent is allowed to freely explore the environment and learn a policy that achieves this goal.
- *Baseline 2:* Constrained RL policy optimization with known STL safety constraints In this baseline, we

TABLE 2. Training hyper-parameters.

Hyper-parameter	Value
Actor learning rate	$5 \cdot 10^{-6}$
Critic learning rate	10^{-3}
Discount factor	0.9
Batch size	256
Policy update delay	2
Exploration noise	0.1
Policy noise	0.2
Policy noise clip	0.5
Actor/Critic activation function	ReLU
Total steps (case-study 1,2,3)	$10^6, 1.5 \cdot 10^6, 10^6$
Steps per epoch (case-study 1,2,3)	$5 \cdot 10^2, 10^3, 10^3$
Cost limit	0.0
λ learning rate	$5 \cdot 10^{-7}$
λ optimizer	Adam

perform constrained reinforcement learning policy optimization with a priori knowledge of the true environmental STL safety constraints. The objective is to maximize the expected cumulative reward while satisfying the given STL safety constraints.

The rationale behind the selection of these two baseline approaches to compare to ours is as follows: baseline 1, involving unconstrained policy optimization, underscores the criticality of clearly defining safety constraints and elucidates the safety risks associated with deploying algorithms trained in the absence of appropriate safety constraints. In contrast, baseline 2 represents an optimal scenario wherein all environmental safety constraint parameters are known a priori, facilitating a comparative analysis to gauge the proximity of our framework's results to this ideal benchmark.

We chose to consider α as the principal convergence metric because it evaluates success in both the upper and lower level optimization problems, i.e. it serves as a qualitative indicator of the effectiveness of the learned STL safety specification in guiding the cost assignment during policy optimization as judged by the human expert and the ability of the RL algorithm to generate a policy that adheres to STL safety constraints.

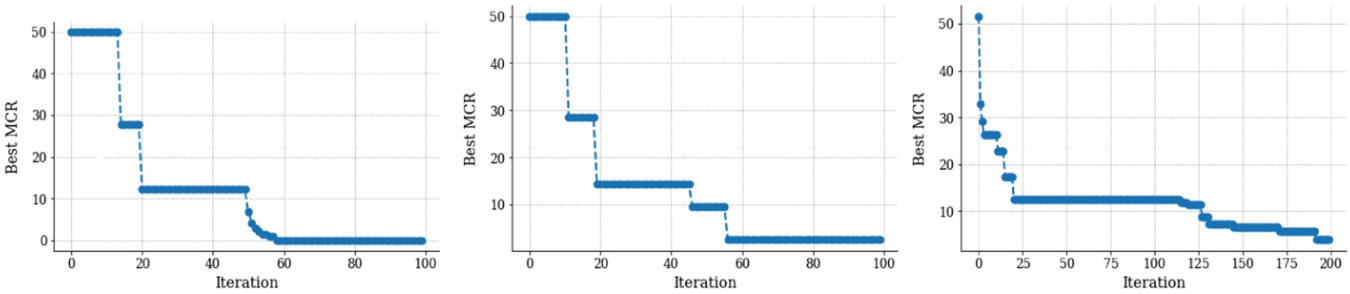
We implemented our proposed algorithm and the two baseline methods to optimize a policy within the each of the specified case studies. The experiments were conducted under consistent environmental settings, with the primary distinction being in the computation of costs for each method at each step. The training parameters used are given in Table 2.

Specifically for our algorithm, the cost assignment during policy optimization is based on a learned STL safety constraint following the process detailed in Section V-B. This contrasts with baseline 2, where the cost assignment stems directly from the actual STL safety constraint, a value which, in practice, is unknown. Finally, baseline 1, an unconstrained optimization approach, does not incorporate cost considerations due to the absence of known safety constraints in the context. For experimentation within our framework, the convergence threshold, δ, was set to 75% for case study 1 and to 90% for case studies 2 and 3, indicating the algorithm terminates once the rollout trace from a policy attains the specified percentage of safe traces. These numbers were decided based on the complexity of the environment, specifically in relation to the quantity of safety parameters required for learning. Convergence was attained when implementing our framework, on average, after 6 iterations on case study 1, 9 iterations for case study 2, and 17 iterations for case study 3. The initial dataset of labeled data contained 10 safe traces and 10 unsafe traces in D_s and D_{us}, respectively, and 50 rollout samples are provided to the human expert for labeling at each iteration. With this setup, the human ultimately ended up labeling 300, 450, and 850 datasets for case studies 1, 2, and 3, respectively. Though these quantities are still substantial, we have found through our experiments that labeling 50 rollout traces per iteration effectively manages the human expert's workload while ensuring that the learning process remains unimpeded. However, further experimentation is necessary to identify the optimal number of datasets a human should label at each iteration to balance expert workload and convergence speed. The safety threshold (cost limit) in the constrained optimization setting is set to 0 for all of our case studies, indicating that no violations of safety constraints are permissible at any point in the trajectory for it to be deemed safe.

VII. RESULTS AND DISCUSSION

We first delve into the results obtained from the BO process employed for pSTL parameter learning. To illustrate these outcomes, we have included a average learning curve of the BO from the final iteration of our algorithm across the three random seeds, which is depicted in Fig. 5 and we present the learned STL safety specification, valuated with the optimal parameters obtained through the BO process averaged across the three random seeds, alongside the True STL specifications for easy comparison in Table 3. Using the Learned STLs given in Table 3 as a constraint, the policy learning curve of the TD3-Lagrangian RL algorithm is shown in Fig. 6. Supplementary to Fig. 6, we provide numeric values of key performance metrics: cumulative reward per episode and the cumulative cost per episode in Table 4. These metrics are derived from the data collected at the conclusion of the training phase averaged over three runs with three random seeds.

Subsequently, we display a graphical illustration depicting the percentage of safe traces within a set of rollout traces generated by executing the trained policy across each case study. This analysis includes a comparison between baseline 1, baseline 2, and our implementation on the various case studies and is depicted in Fig. 7. The primary objective of this evaluation is to provide a quantifiable measure of safety for policies generated through each approach, effectively gauging the potential rate of unsafe incidents that might occur if any of these policies were to be deployed in the respective case studies. Such a visual and statistical comparison is instrumental in assessing the relative safety efficacy of each approach.



(a) Half Cheetah Velocity (b) Point Circle (c) Point Goal

FIGURE 5. BO learning curve for parameter learning of pSTL specifications provided in case studies. 5(a) depicts the learning curve for optimizing one parameter, 5(b) depicts the learning curve for optimizing two parameters, and 5(c) shows the learning curve for optimizing 16 parameters. The minimization metric is given as the balanced misclassification rate (MCR) of the STL at sequentially generated candidate points.

TABLE 3. Learned STL specifications alongside true environmental safety constraint for each case study.

	Learned STL Specification (ϕ_{cost})	True STL Specification (ϕ_{true})
Safe Velocity Half Cheetah	$G\left(\neg\left(u_a > 3.3521\right)\right)$	$G\left(\neg\left(u_a > 3.2096\right)\right)$
Safe Navigation Circle	$G\left(\neg\left(\left(x_a < \text{-}0.93\right) \vee \left(x_a > 1.064\right)\right)\right)$	$G\left(\neg\left(\left(x_a < \text{-}1.20\right) \vee \left(x_a > 1.0\right)\right)\right)$
Safe Navigation Goal	$G\Bigg(\neg\Bigg(\Big(\sqrt{(x_a - \text{-}0.714)^2 + (y_a - 0.91)^2} < 0.4\Big)$ $\vee \Big(\sqrt{(x_a - 1.07)^2 + (y_a - 0.04)^2} < 0.4\Big)$ $\vee \Big(\sqrt{(x_a - \text{-}1.51)^2 + (y_a - 0.11)^2} < 0.4\Big)$ $\vee \Big(\sqrt{(x_a - \text{-}0.53)^2 + (y_a - \text{-}0.31)^2} < 0.4\Big)$ $\vee \Big(\sqrt{(x_a - 0.43)^2 + (y_a - 0.87)^2} < 0.4\Big)$ $\vee \Big(\sqrt{(x_a - 0.16)^2 + (y_a - \text{-}1.79)^2} < 0.4\Big)$ $\vee \Big(\sqrt{(x_a - \text{-}2.3)^2 + (y_a - 1.04)^2} < 0.4\Big)$ $\vee \Big(\sqrt{(x_a - 0.83)^2 + (y_a - \text{-}1.01)^2} < 0.4\Big)\Big)\Bigg)$	$G\Bigg(\neg\Bigg(\Big(\sqrt{(x_a - \text{-}0.75)^2 + (y_a - 1.0)^2} < 0.4\Big)$ $\vee \Big(\sqrt{(x_a - 1.0)^2 + (y_a - 0.2)^2} < 0.4\Big)$ $\vee \Big(\sqrt{(x_a - \text{-}1.4)^2 + (y_a - 0.7)^2} < 0.4\Big)$ $\vee \Big(\sqrt{(x_a - \text{-}0.5)^2 + (y_a - \text{-}0.3)^2} < 0.4\Big)$ $\vee \Big(\sqrt{(x_a - 0.25)^2 + (y_a - 0.9)^2} < 0.4\Big)$ $\vee \Big(\sqrt{(x_a - 0.0)^2 + (y_a - \text{-}1.5)^2} < 0.4\Big)$ $\vee \Big(\sqrt{(x_a - \text{-}1.9)^2 + (y_a - 1.0)^2} < 0.4\Big)$ $\vee \Big(\sqrt{(x_a - 1.0)^2 + (y_a - \text{-}1.0)^2} < 0.4\Big)\Big)\Bigg)$

TABLE 4. Metrics from the conclusion of training averaged over three random seeds per environment.

	Baseline 1		Baseline 2		Ours	
	$\overline{\mathcal{J}}_R$	$\overline{\mathcal{J}}_c$	$\overline{\mathcal{J}}_R$	$\overline{\mathcal{J}}_c$	$\overline{\mathcal{J}}_R$	$\overline{\mathcal{J}}_c$
Safe Velocity Half Cheetah	10371.1	957.6	2676.1	1.67	2114.7	0.62
Safe Navigation Circle	111.3	390.3	54.90	1.41	57.02	8.39
Safe Navigation Goal	28.2	48.8	11.5	4.9	16.5	24.3

In our final analysis, we conduct a comprehensive evaluation focusing on the performance of the upper-level optimization, specifically the learning of the STL constraint parameters. This evaluation entails calculating the misclassification rate of the STL post-training against a labeled data set to assess the accuracy of our learned STL. In order to establish a benchmark for this metric, we compare it with that of the MCR of STL constraint used in baseline 2, which is the true environmental constraint, against the same dataset. A close alignment in these rates would indicate a high degree of accuracy of our learned parameters relative to the *true* parameters.

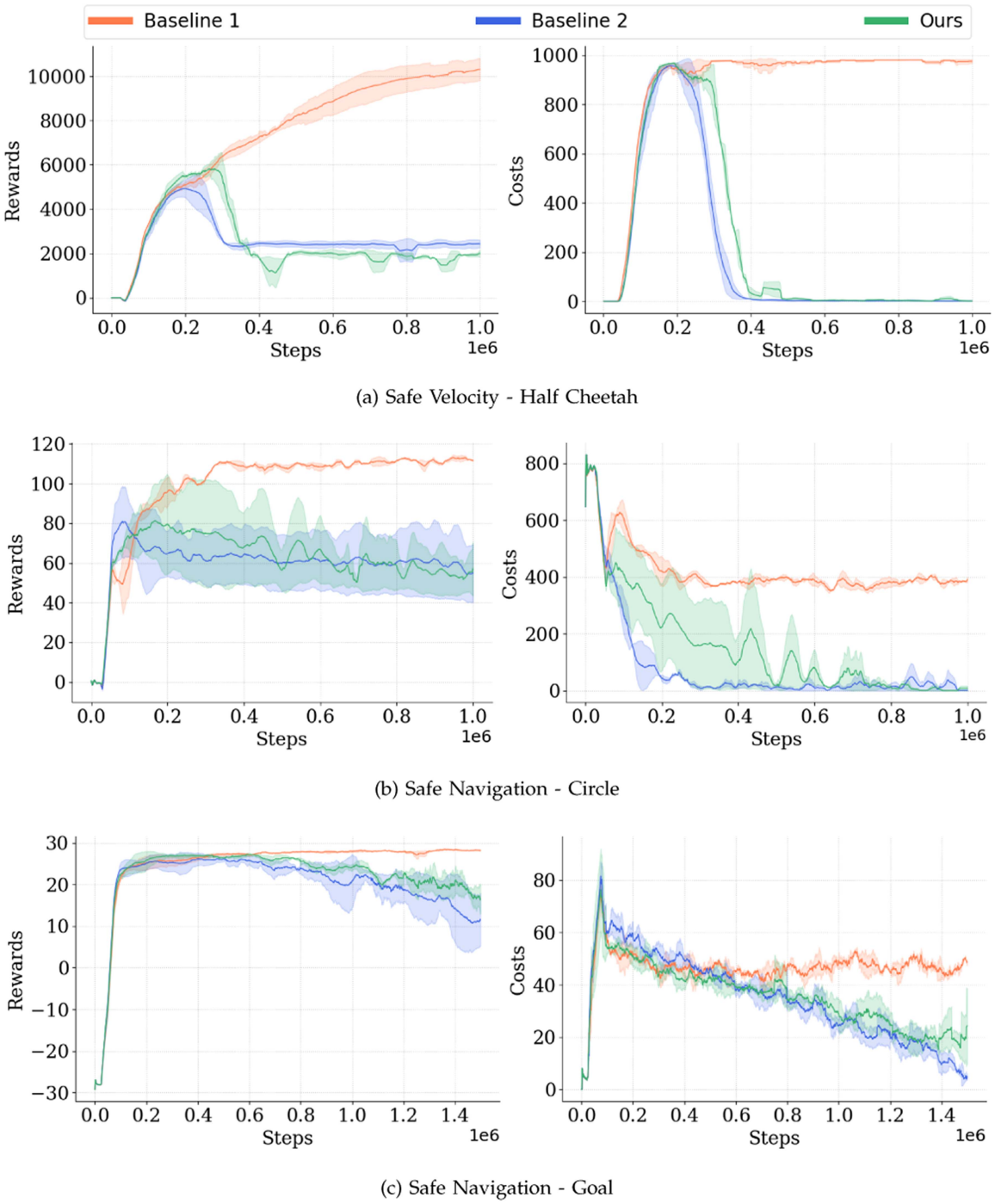


FIGURE 6. **Policy learning curve for our algorithm and baselines 1 and 2 over 1e6 total environment interactions for 6(a) and 6(b) and 1.5e6 total environment interactions for 6(c). The plots on the right display the cumulative rewards per episode, while those on the left display the cumulative costs (quantified as the total number of constraint violations) per episode throughout the training.**

In the analysis presented in Fig. 6, a trade-off between rewards and costs is observed across all case studies. This observation substantiates that the tasks in all of the case studies are not "trivially-safe", i.e., maximizing rewards in these settings consistently leads to constraint violations to some degree. Notably, baseline 1 achieves the highest reward in all case studies, yet it concurrently incurs the highest cost at the end of training. This pattern suggests that the agent continues to engage in unsafe actions, prioritizing only reward maximization. In contrast, our algorithm exhibits a reduction in rewards compared to baseline 1; however, it succeeds in reducing costs substantially across all case studies, and even achieves the threshold of zero violations per episode by the end of training in two out of the three case studies. This

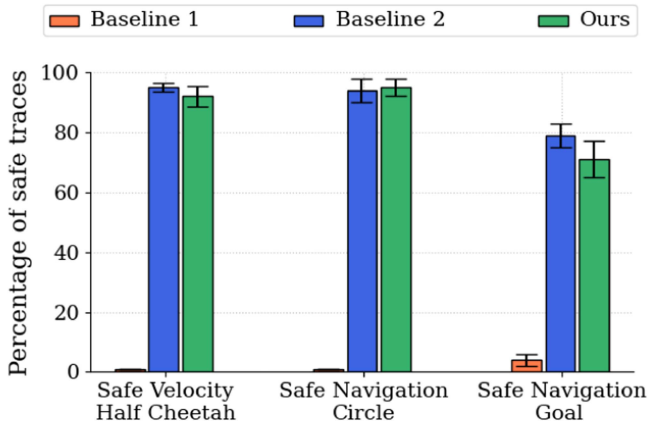


FIGURE 7. Percentage of safe traces in final policy rollouts across different case-studies implementing proposed and baseline algorithms.

improvement upon baseline 1 is a direct result of applying our algorithm in scenarios with initially unknown safety constraints by allowing the learning and adhering to safety constraints, even in the absence of prior knowledge of the constraints. Baseline 2 represents an ideal training scenario, assuming complete availability of STL safety constraint information. The performance of our algorithm closely mirrors that of baseline 2, a result that indicates close similarity between the learned STL in our approach and the true STL.

The data provided in Table 4 offers a quantitative counterpart to the results depicted in Fig. 6. This tabular representation offers a more detailed numerical articulation of the result metrics at the end of training, complementing the plots displayed in the figure. Consequently, the interpretation of the results in the table aligns closely with that of Fig. 6, i.e. while baseline 1 achieves the highest cumulative rewards per episode at the conclusion of training, it also incurs the highest number of constraint violation per episode. In contrast, the adoption of our proposed method demonstrates a significant improvement in cost-efficiency by efficiently directing the agent to act in accordance to the learned environmental constraints, which closely mirror the actual environmental constraints.

In Fig. 7, it is evident that the policy optimized under baseline 1 fails to produce safe trajectories in case studies 2 and 3, with only a few safe trajectories in case study 2. In contrast, the policy optimized through our framework yields a number of safe trajectories comparable to baseline 2, which had complete knowledge of the safety constraints from the start. This demonstrates the effectiveness of our approach in learning the safety constraints and ensuring safety during policy deployment, even with less initial information on safety constraints.

In Table 5, we exhibit results that underscore the quality of the learned STL using our approach. We assessed the STL's quality by its ability to accurately classify labeled data, and then benchmarked these results against the performance of the True STL used in baseline 2. While the True STL safety

TABLE 5. MCR comparison between the learned STL (ours) and the true STL (baseline 2).

	MCR	
	Baseline 2	Ours
Safe Velocity Half Cheetah	0.0	0.0
Safe Navigation Circle	0.0	0.0251
Safe Navigation Goal	0.0	0.0534

specification, by definition, should classify all traces with a misclassification rate (MCR) of zero, it is noteworthy that the MCR of the STL derived through our algorithm closely parallels this standard. In scenarios such as case studies 1 and 3, characterized by a limited number of learning parameters for the pSTL, the MCR is close to zero, mirroring the performance of the True STL, whereas in more complex settings, such as that of case study 2 with 16 learning parameters, the MCR, while higher, still remains within reasonable bounds considering the large number of learning parameters. This not only highlights the precision of our STL learning process but also indicates that the parameters we derived are remarkably close to the real environmental constraints. Overall, our results demonstrate the precision of our algorithm in adapting to and respecting the environmental safety constraints, thereby offering a balanced approach in terms of performance and cost during training and implementation.

Limitations: Despite the successful results, there are limitations to our approach which must be acknowledged. Firstly, our approach relies on pre-existing datasets of safe and unsafe trajectories, however small, as well as an STL safety specification template. The availability of these elements is required for the initialization of our process and the overall performance. The second limitation is the requirement for human expert manual labeling of trajectories. While human expertise is invaluable for providing a better understanding of safety, this requirement imposes considerable demands on human resources.

VIII. CONCLUSION

This research tackles the challenge of ensuring safety in RL, particularly when predefined safety constraints are unavailable. Traditional methods in safe RL often rely heavily on static, predefined safety constraints, thus limiting their applicability. To address this limitation, we proposed an approach that concurrently learns an optimal control policy and identifies the STL safety constraint parameters of a given environment. Our approach implements a bilevel optimization framework, where the upper level is dedicated to optimizing parameters of pSTL safety constraint, and the lower level aims to find an optimal safe policy, constrained by the learned STL safety specification. Our process also leverages input from

human experts who assign safety labels to the RL policy rollout traces to be used to refine safety specification parameters. Various case studies demonstrate the efficacy of our approach, showing that our algorithm substantially reduces constraint violations compared to traditional unconstrained reward maximization methods, while maintaining similar levels of performance. Additionally, it closely mirrors the results of scenarios with complete initial knowledge of true environmental constraints, thereby underscoring the close alignment of our learned STL parameters with actual safety parameters. We believe our approach takes a step closer towards understanding the "unknown unknown" uncertainty in RL, referring to unforeseen safety requirements not initially specified in the template, which may emerge during interactions or through human feedback.

Future directions of our work include conducting a comprehensive analysis to determine how we can minimize the total number of rollout traces to be labeled without significantly impacting the convergence speed, as well as incorporating active learning techniques to select the most informative traces for labeling to optimize human efforts. In addition, extending our methodology to encompass the synthesis of the entire STL safety constraint, which includes both the structural formula and its parameters, is of great importance. Finally, assessing the scalability and efficiency of our framework through further testing, especially in environments characterized by a large number of safety parameters, will yield valuable insights.

REFERENCES

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

[2] B. R. Kiran et al., "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, Jun. 2021.

[3] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.

[4] P. Razzaghi et al., "A survey on reinforcement learning in aviation applications," 2022, *arXiv:2211.02147*.

[5] B. Hambly, R. Xu, and H. Yang, "Recent advances in reinforcement learning in finance," *Math. Finance*, vol. 33, no. 3, pp. 437–503, 2023.

[6] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 42, pp. 1437–1480, 2015.

[7] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 22–31.

[8] Y. Liu, J. Ding, and X. Liu, "IPO: Interior-point policy optimization under constraints," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 4940–4947.

[9] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2669–2678.

[10] A. Baheri, S. Nageshrao, H. E. Tseng, I. Kolmanovsky, A. Girard, and D. Filev, "Deep reinforcement learning with enhanced safety for autonomous highway driving," in *Proc. IEEE Intell. Veh. Symp.*, 2020, pp. 1550–1555.

[11] A. Baheri, "Safe reinforcement learning with mixture density network, with application to autonomous driving," *Results Control Optim.*, vol. 6, 2022, Art. no. 100095.

[12] S. Bansal, "Specification-guided reinforcement learning," in *Proc. Int. Static Anal. Symp.*, 2022, pp. 3–9.

[13] S. Gu et al., "A review of safe reinforcement learning: Methods, theory and applications," 2022, *arXiv:2205.10330*.

[14] Y. Liu, A. Halev, and X. Liu, "Policy learning with constraints in model-free reinforcement learning: A survey," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, 2021, pp. 4508–4515.

[15] W. Zhao, T. He, R. Chen, T. Wei, and C. Liu, "State-wise safe reinforcement learning: A survey," 2023, *arXiv:2302.03122*.

[16] D. Ding, X. Wei, Z. Yang, Z. Wang, and M. Jovanovic, "Provably efficient safe exploration via primal-dual policy optimization," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 3304–3312.

[17] S. Qiu, X. Wei, Z. Yang, J. Ye, and Z. Wang, "Upper confidence primal-dual reinforcement learning for CMDP with adversarial loss," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 15277–15287.

[18] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *J. Mach. Learn. Res.*, vol. 18, no. 167, pp. 1–51, 2018.

[19] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, "Projection-based constrained policy optimization," 2020, *arXiv:2010.03152*.

[20] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh, "Lyapunov-based safe policy optimization for continuous control," 2019, *arXiv:1901.10031*.

[21] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A Lyapunov-based approach to safe reinforcement learning," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 8103–8112.

[22] Y. Luo and T. Ma, "Learning barrier certificates: Towards safe reinforcement learning with zero training-time violations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 25621–25632.

[23] Y. Yang, Y. Jiang, Y. Liu, J. Chen, and S. E. Li, "Model-free safe reinforcement learning through neural barrier certificate," *IEEE Robot. Automat. Lett.*, vol. 8, no. 3, pp. 1295–1302, Mar. 2023.

[24] A. Wachi and Y. Sui, "Safe reinforcement learning in constrained Markov decision processes," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 9797–9806.

[25] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration in finite markov decision processes with Gaussian processes," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4312–4320.

[26] A. Wachi, Y. Sui, Y. Yue, and M. Ono, "Safe exploration and optimization of constrained MDPs using Gaussian processes," *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018.

[27] M. Hasanbeig, A. Abate, and D. Kroening, "Cautious reinforcement learning with logical constraints," in *Proc. 19th Int. Conf. Auton. Agents MultiAgent Syst.*, 2020, pp. 483–491.

[28] M. Cai, S. Xiao, J. Li, and Z. Kan, "Safe reinforcement learning under temporal logic with reward design and quantum action selection," *Sci. Rep.*, vol. 13, no. 1, 2023, Art. no. 1925.

[29] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 6565–6570.

[30] N. Hamilton, P. K. Robinette, and T. T. Johnson, "Training agents to satisfy timed and untimed signal temporal logic specifications with reinforcement learning," in *Proc. Int. Conf. Softw. Eng. Formal Methods*, 2022, pp. 190–206.

[31] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3834–3839.

[32] A. Balakrishnan and J. V. Deshmukh, "Structured reward shaping using signal temporal logic specifications," in *Proc. Int. Conf. Intell. Robots Syst.*, 2019, pp. 3481–3486.

[33] J. Ikemoto and T. Ushio, "Deep reinforcement learning under signal temporal logic constraints using lagrangian relaxation," *IEEE Access*, vol. 10, pp. 114814–114828, 2022.

[34] Z. Kong, A. Jones, A. M. Ayala, E. A. Gol, and C. Belta, "Temporal logic inference for classification and prediction from data," in *Proc. 17th Int. Conf. Hybrid Syst.: Comput. Control*, 2014, pp. 273–282.

[35] Z. Kong, A. Jones, and C. Belta, "Temporal logics for learning and detection of anomalous behavior," *IEEE Trans. Autom. Control*, vol. 62, no. 3, pp. 1210–1222, Mar. 2017.

[36] A. Jones, Z. Kong, and C. Belta, "Anomaly detection in cyber-physical systems: A formal methods approach," in *Proc. IEEE 53rd Conf. Decis. Control*, 2014, pp. 848–853.

[37] P. Vaidyanathan et al., "Grid-based temporal logic inference," in *Proc. 56th Annu. Conf. Decis. Control*, 2017, pp. 5354–5359.

[38] G. Bombara, C.-I. Vasile, F. Penedo, H. Yasuoka, and C. Belta, "A decision tree approach to data classification using signal temporal logic," in *Proc. 19th Int. Conf. Hybrid Syst.: Comput. Control*, 2016, pp. 1–10.

[39] G. Bombara and C. Belta, "Offline and online learning of signal temporal logic formulae using decision trees," *ACM Trans. Cyber-Phys. Syst.*, vol. 5, pp. 1–23, Mar. 2021.
[40] G. Bombara and C. Belta, "Online learning of temporal logic formulae for signal classification," in *Proc. Eur. Control Conf.*, 2018, pp. 2057–2062.
[41] G. Bombara and C. Belta, "Signal clustering using temporal logics," in *Proc. 17th Int. Conf. Runtime Verification*, 2017, pp. 121–137.
[42] S. Jha, A. Tiwari, S. Seshia, T. Sahai, and N. Shankar, "TeLEx: Learning signal temporal logic from positive examples using tightness metric," *Formal Methods Syst. Des.*, vol. 54, no. 3, pp. 364–387, 2019.
[43] L. Nenzi, S. Silvetti, E. Bartocci, and L. Bortolussi, "A robust genetic algorithm for learning temporal specifications from data," in *Proc. 15th Int. Conf. Quantitative Eval. Syst.*, 2018, pp. 323–338.
[44] F. Pigozzi, E. Medvet, and L. Nenzi, "Mining road traffic rules with signal temporal logic and grammar-based genetic programming," *Appl. Sci.*, vol. 11, no. 22, 2021, Art. no. 10573.
[45] E. Asarin, A. Donzé, O. Maler, and D. Nickovic, "Parametric identification of temporal properties," in *Proc. Int. Conf. Runtime Verification*, 2012, pp. 147–160.
[46] K. Leung, N. Arechiga, and M. Pavone, "Back-propagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods," in *Proc. 14th Workshop Algorithmic Found. Robot.*, 2021, pp. 432–449.
[47] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia, "Mining requirements from closed-loop control models," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 11, pp. 1704–1717, Nov. 2015.
[48] H. Yang, B. Hoxha, and G. Fainekos, "Querying parametric temporal logic properties on embedded systems," in *Proc. Int. Conf. Testing Softw. Syst.*, 2012, pp. 136–151.
[49] M. Vazquez-Chanlatte, J. V. Deshmukh, X. Jin, and S. A. Seshia, "Logical clustering and learning for time-series data," in *Proc. Int. Conf. Comput. Aided Verification*, 2016, pp. 305–325.
[50] H. Ma, C. Liu, S. E. Li, S. Zheng, and J. Chen, "Joint synthesis of safety certificate and safe control policy using constrained reinforcement learning," in *Proc. 4th Annu. Learn. Dyn. Control Conf.*, 2022, pp. 97–109.
[51] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Proc. Int. Symp. Formal Techn. Real-Time Fault-Tolerant Syst.*, 2004, pp. 152–166.
[52] R. S. Sutton et al., *Introduction to Reinforcement Learning*, vol. 2. Cambridge, MA, USA: MIT Press, 1998.
[53] R. Bellman, "On the theory of dynamic programming," *Proc. Nat. Acad. Sci.*, vol. 38, no. 8, pp. 716–719, 1952.
[54] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[55] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.
[56] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[57] E. Altman, *Constrained Markov Decision Processes*. London, U.K.: Chapman and Hall, 1999.
[58] P. I. Frazier, "A tutorial on Bayesian optimization," 2018, *arXiv:1807.02811*.
[59] A. H. Victoria and G. Maragatham, "Automatic tuning of hyperparameters using Bayesian optimization," *Evolving Syst.*, vol. 12, pp. 217–223, 2021.
[60] A. Baheri and C. Vermillion, "Waypoint optimization using Bayesian optimization: A case study in airborne wind energy systems," in *Proc. Amer. Control Conf.*, 2020, pp. 5102–5017.
[61] A. Baheri, S. Bin-Karim, A. Bafandeh, and C. Vermillion, "Real-time control using Bayesian optimization: A case study in airborne wind energy systems," *Control Eng. Pract.*, vol. 69, pp. 131–140, 2017.
[62] A. Baheri and C. Vermillion, "Combined plant and controller design using batch Bayesian optimization: A case study in airborne wind energy systems," *J. Dyn. Syst., Meas., Control*, vol. 141, no. 9, 2019, Art. no. 091013.
[63] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty," *Ann. Math. Artif. Intell.*, vol. 76, no. 1, pp. 5–23, 2016.
[64] Y. Zhang, D. W. Apley, and W. Chen, "Bayesian optimization for materials design with mixed quantitative and qualitative variables," *Sci. Rep.*, vol. 10, no. 1, 2020, Art. no. 4924.
[65] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. D. Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, Jan. 2016.
[66] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 276–295, Apr. 2018.
[67] V. García, R. A. Mollineda, and J. S. Sánchez, "Index of balanced accuracy: A performance measure for skewed class distributions," in *Proc. Iberian Conf. Pattern Recognit. Image Anal.*, 2009, pp. 441–448.
[68] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. New York, NY, USA: Academic, 2014.
[69] J. Ji et al., "Omnisafe: An infrastructure for accelerating safe reinforcement learning research," 2023, *arXiv:2305.09304*.
[70] J. Ji et al., "Safety-gymnasium: A unified safe reinforcement learning benchmark," 2023, *arXiv:2310.12567*.
[71] A. Ray, J. Achiam, and D. Amodei, "Benchmarking safe exploration in deep reinforcement learning," 2019. [Online]. Available: https://cdn.openai.com/safexp-short.pdf
[72] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. Int. Conf. Intell. Robots Syst.*, 2012, pp. 5026–5033.
[73] G. Brockman et al., "Open AI gym," 2016, *arXiv:1606.01540*.