Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)
Survey Track

Policy Learning with Constraints in Model-free Reinforcement Learning: A Survey

Yongshuai Liu[1] , **Avishai Halev**[1,2] and **Xin Liu**[1]
[1]University of California, Davis
[2]Total E&P R&T USA
{yshliu, ahalev, xinliu}@ucdavis.edu

Abstract

Reinforcement Learning (RL) algorithms have had tremendous success in simulated domains. These algorithms, however, often cannot be directly applied to physical systems, especially in cases where there are constraints to satisfy (e.g. to ensure safety or limit resource consumption). In standard RL, the agent is incentivized to explore any policy with the sole goal of maximizing reward; in the real world, however, ensuring satisfaction of certain constraints in the process is also necessary and essential. In this article, we overview existing approaches addressing constraints in model-free reinforcement learning. We model the problem of learning with constraints as a Constrained Markov Decision Process and consider two main types of constraints: cumulative and instantaneous. We summarize existing approaches and discuss their pros and cons. To evaluate policy performance under constraints, we introduce a set of standard benchmarks and metrics. We also summarize limitations of current methods and present open questions for future research.

1 Introduction

Deep reinforcement learning (RL) has demonstrated excellent success in a variety of domains, including games [Vinyals *et al.*, 2019], robotic control [Levine *et al.*, 2016], and recommendation systems [Shani *et al.*, 2005], etc. In typical RL settings like these, the fundamental principle of RL – in which the agent aims to maximize long-term reward by trial and error – incentives the agent to explore the entire state space and experiment with all possible actions in unknown environments.

Many real-world applications, however, require the agent to consider constraints that may curtail the agent's freedom to explore [Dulac-Arnold *et al.*, 2019]; representative examples include safe exploration [Garcıa and Fernández, 2015; Ray *et al.*, 2019] and resource allocation [Bhatia *et al.*, 2019; Liu *et al.*, 2020a], which generally employ the constrained RL framework as their main formalism.

Physical examples of this abound. For instance, consider a situation where a robot is asked to accomplish tasks with a limited amount of energy or a finite amount of time – robot behavior is restricted by the total amount of energy available and by the time taken. Another typical example is constraints on the probability of resource overutilization. Assume a group of people is sharing resources and everyone is assigned a certain portion; people in the group are allowed to overuse the resources which are more than he/she is assigned but with limited times (e.g. AT&T family plan). Further examples exist in autonomous and robotic research: an autonomous car is not allowed to take any actions that could cause an accident while optimizing its driving policy; and the goal of a drone is to not only maximize flying speed and distance but also avoid being damaged while doing so.

In this context, the Constrained Markov Decision Process (CMDP) becomes an important model for constrained sequential decision-making problems. In a CMDP, the objective is to maximize long-term reward while keeping certain costs under their respective constraints. In this paper, we divide constraints into two general types relevant to the CMDP problem: cumulative and instantaneous, defined as follows.

- Cumulative constraints require the sum or mean of one constraint variable from the beginning to the current time step to be within a certain limit (e.g., total energy consumption, resource overutilization), as in Eq. (6).
- Instantaneous constraints are constraints that the chosen action needs to satisfy in each step (e.g., accident and damage avoidance), formulated in Eq. (7).

In this paper, we discuss various approaches to addressing the CMDP problem. The CMDP problem is more challenging than standard RL problems, which are able to employ solutions that directly maximize the reward, as value-based [Mnih *et al.*, 2013] or policy gradient [Sutton *et al.*, 1999] methods do. Applying these methods directly to CMDP problems would result in policies that do not respect constraints in general. There are three key facets of this challenge:

- Policy learning becomes a constrained optimization problem which makes the optimization more complex.
- Constraints are diverse, making the problem difficult to formulate.
- The agent will inevitably violate the constraints as model-free RL lacks a priori environment knowledge, necessitating trial and error.

The rest of the paper proceeds as follows. In Section 2, we introduce CMDPs as well as the formulation of each type of constraints. In Sections 3 and 4, we discuss approaches to solving CMDP problems with cumulative and instantaneous constraints, respectively. In Section 5, we suggest benchmarks and metrics for policy evaluation. Finally, we discuss the limitations of current methods and avenues desiring future attention in Section 6 and conclude in Section 7.

2 Problem formulation

2.1 Constrained Markov Decision Process

A Markov Decision Process (MDP) is defined as a tuple $(S, A, R, P, \mu, \gamma)$ [Sutton and Barto, 2018], where S is the set of states, A is the set of actions, $R : S \times A \times S \mapsto \mathbb{R}$ is the reward function, $P : S \times A \times S \mapsto [0, 1]$ is the transition probability function , $\mu : S \mapsto [0, 1]$ is the initial state distribution and γ is the discount factor for future reward. A policy $\pi : S \mapsto \mathcal{P}(A)$ is a mapping from states to a probability distribution over actions and $\pi(a_t|s_t)$ is the probability of taking action a under state s in time step t. We denote a policy π as π_θ to emphasize its dependence on a parameter θ . The standard goal of an MDP is to learn a policy π_θ that maximizes the discounted cumulative reward:

$$\max_\theta \ J_R^{\pi_\theta} = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})], \tag{1}$$

where $\tau = (s_0, a_0, s_1, a_1...)$ denotes a trajectory, and $\tau \sim \pi_\theta$ denotes trajectories sampled from the policy π_θ.

A Constrained Markov Decision Process (CMDP) – defined as a tuple $(S, A, R, C, P, \mu, \gamma)$ – extends the definition of the traditional MDP [Altman, 1999] by introducing a set of cost functions C. The cost functions $C_i \in C$, $C_i : S \times A \times S \mapsto \mathbb{R}$ are constrained in some way, depending on the type of constraint (Taxonomy 1).

We call an action a *feasible* if $a \in A$ satisfies all of its necessary constraints. In a CMDP, the objective is to select a policy π_θ that maximizes the long-term reward while satisfying all constraints. Formally, we have

$$\begin{aligned} &\max_\theta \quad J_R^{\pi_\theta}, \\ &\text{s.t.} \quad a_t \text{ is feasible.} \end{aligned} \tag{2}$$

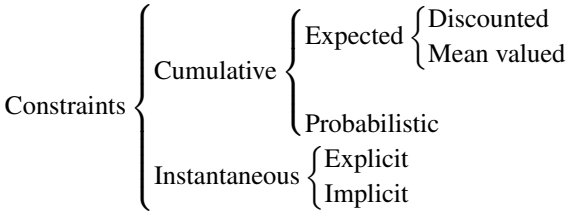
2.2 Cumulative Constraints

Cumulative constraints require the sum or mean of a given cost signal to be within a certain limit, where the sum or mean are computed from the beginning to the current time, such as total revenue and network throughput. We consider two different kinds of cumulative constraints: those on the expectation of some cost signal and those on the probability of a cost signal. The first group includes discounted cumulative constraints and mean valued constraints [Altman, 1999].

A **discounted cumulative** constraint is of the form

$$J_{C_i}^{\pi_\theta} = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t C_i(s_t, a_t, s_{t+1})] \leq \epsilon_i. \tag{3}$$

where $\tau = (s_0, a_0, s_1, a_1...)$ is a trajectory, $\tau \sim \pi_\theta$, and ϵ_i is the limit for each cumulative constraint. These constraints satisfy the Recursive Bellman Equation.



Taxonomy 1: Constraint categories.

A **mean valued** constraint is of the form

$$J_{C_i}^{\pi_\theta} = \mathbb{E}_{\tau\sim\pi_\theta}[\frac{1}{T}\sum_{t=0}^{T-1} C_i(s_t, a_t, s_{t+1})] \leq \epsilon_i, \tag{4}$$

where T is the total number of time steps in each trajectory.

The second group concerns the probability that the cumulative costs violate a constraint [Geibel, 2006]. **Probabilistic** constraints are of the form

$$J_{C_i}^{\pi_\theta} = P(\sum_t C_i(s_t, a_t, s_{t+1}) \geq \eta) \leq \epsilon_i. \tag{5}$$

where $\eta \in \mathbb{R}$ is the cumulative cost threshold for each trajectory and $\epsilon_i \in (0, 1)$ is the probability limit. Intuitively, these constraints require that the probability of cumulative costs exceeding η be less than ϵ_i.

In a CMDP equipped with cumulative constraints, the goal is to find a policy π_θ that maximizes the discounted cumulative reward while satisfying these cumulative constraints. We formulate the cumulative constraint problem as:

$$\begin{aligned} &\max_\theta J_R^{\pi_\theta} \\ &s.t. \quad J_{C_i}^{\pi_\theta} \leq \epsilon_i, \end{aligned} \tag{6}$$

2.3 Instantaneous Constraints

Instantaneous constraints are constraints on the actions, states, or cost functions that selected actions must ensure the satisfaction of in every step. They can be further divided into explicit and implicit varieties. An **explicit** constraint has a closed-form expression that can be numerically checked, such as computing or storage capacity. Explicit constraints can accurately (and relatively easily) be evaluated for each action, for example the constraints are on the actions themselves [Bhatia *et al.*, 2019]. Such constraints are usually on available resources in a general sense. An **implicit** constraint is a constraint that does not have an accurate closed-form formulation due to the complexity of the system [Dalal *et al.*, 2018], such as network latency. These constraints generally concern the outcome of actions; their unknown dependence on actions necessitates modeling or learning this dependence using existing data and/or during exploration. For example, probabilistic instantaneous constraints are one kind of implicit constraint: $\widetilde{C}_i(s_t, a_t, s_{t+1}) = P(C_i(s_t, a_t, s_{t+1}) \geq \eta) \leq \epsilon_i$, where we define a new cost function $\widetilde{C}_i(s_t, a_t, s_{t+1})$.

We formalize the instantaneous constraint problem as:

$$\begin{aligned} &\max_{\theta} \quad J_R^{\pi_\theta} \\ &s.t.\ C_i(s_t, a_t, s_{t+1}) \leq \omega_i, \end{aligned} \tag{7}$$

where ω_i is the limit for each respective instantaneous constraint. These limits can be either a predefined constant or a function of the current state s_t [Henri *et al.*, 2020].

We discuss how to address different kinds of constraints in the following section; Table 1 summarizes the body of work that we consider in this article. We compare the methods along the following dimensions: aspects of performance, scalability, computational complexity and theoretical support, and score them according: ✓,−, and ×, in descending order.

3 Cumulative Constraints

3.1 Lagrangian Relaxation

Lagrangian relaxation is the most common approach to address cumulative constraints. Initially introduced in [Altman, 1999], this method works for both expected and probabilistic constraints. The general form of Lagrangian relaxation is to reduce the problem to an unconstrained problem via Lagrange multipliers. These adaptive Lagrange multipliers are then used to penalize constraint violation:

$$\min_{\lambda_i \leq 0} \max_{\theta} L(\theta, \lambda_i) = J_R^{\pi_\theta} - \sum_i \lambda_i (J_{C_i}^{\pi_\theta} - \epsilon_i), \tag{8}$$

where L is the Lagrangian and λ_i are the Lagrange multipliers for each cumulative constraint. The Lagrange multipliers are then updated in the dual problem to satisfy the constraints.

The Lagrangian relaxation method can achieve high performance, measured by having high long-term rewards and low cumulative costs [Liu *et al.*, 2020b; Chow *et al.*, 2017]. However, this approach is sensitive to the initialization of the Lagrange multipliers and the learning rate. The learning curve variation is large and the policy obtained during training does not consistently satisfy the constraints, as discussed in [Achiam *et al.*, 2017; Chow *et al.*, 2019]. Moreover, the Lagrangian multipliers are solved on a slower time-scale, which makes it difficult to optimize in practice.

More scalable versions of the Lagrangian-based methods have been proposed over the years in attempts to accelerate the learning process [Liang *et al.*, 2018; Chen *et al.*, 2021; Stooke *et al.*, 2020]. RCPO [Tessler *et al.*, 2018] follows a two-timescale primal-dual approach, giving guarantees for convergence to a fixed point. [Bohez *et al.*, 2019] solves the constrained problem with value-based methods (Q-learning) instead of policy gradient methods. [Le *et al.*, 2019] describe a batch policy algorithm with PAC-style guarantees for cumulative constraints using a similar game-theoretic formulation.

3.2 Constrained Policy Optimization (CPO)

CPO [Achiam *et al.*, 2017] extends the TRPO algorithm [Schulman *et al.*, 2015] to handle expected cumulative constraints. They approximate the complex constrained optimization problem with a quadratic constrained optimization problem which can be solved by using the Karush–Kuhn–Tucker (KKT) conditions.

CPO monotonically improves the policy during training and demonstrates stable empirical performance [Chow *et al.*, 2019; Liu *et al.*, 2020b]. However, CPO is computationally expensive as it uses conjugate gradients for approximating the Fisher Information Matrix, whose approximation error affects the overall performance. In addition, it requires a backtracking line-search procedure to determine feasible actions. Both of these aspects make CPO complicated to implement and compute. Furthermore, CPO only supports constraints that satisfy the Recursive Bellman Equation (i.e. discounted sum constraints); on top of this, it is difficult to accommodate multiple constraints at once.

3.3 Interior-point Policy Optimization (IPO)

IPO [Liu *et al.*, 2020b] is a first-order constrained optimization method for expected cumulative constraints. Inspired by the interior-point method [Boyd and Vandenberghe, 2004], IPO augments the objective function (Eq. 6) with logarithmic barrier functions as penalty functions to accommodate the constraints. Intuitively, IPO aims to have a penalty function such that 1) if a constraint is satisfied, the penalty added to the reward function is zero, and 2) if the constraint is violated, the penalty added goes to negative infinity. This motivates the objective function

$$\max_{\theta} J_R^{\pi_\theta} + \sum_i \frac{1}{t_i} \log(-J_{C_i}^{\pi_\theta} + \epsilon_i) \tag{9}$$

where t_i is a hyperparameter. Note this differs from Lagrangian methods, where the Lagrangian multiplier is much harder to learn. Furthermore, IPO builds upon PPO [Schulman *et al.*, 2017], which inherits the trust-region property of TRPO, to achieve policy monotonical improvement and better guarantee constraint satisfaction during training.

Preliminary results suggest that IPO has desirable properties, e.g. easy-to-tune hyperparameters and ability of handling multiple constraints, etc. Furthermore, IPO has shown promising performance in handling multiple and general cumulative constraints. One drawback, however, is that policies must be feasible upon initialization; solutions to this issue are proposed in a follow-up application paper [Liu *et al.*, 2020a; Liu *et al.*, 2021].

3.4 Projection-based Constrained Policy Optimization (PCPO)

PCPO [Yang *et al.*, 2020] is an iterative method for optimizing policies under expected cumulative constraints. It includes two stages. The first stage maximizes reward using TRPO [Schulman *et al.*, 2015]) without constraints, giving an intermediate policy that may not satisfy the constraints. The second stage handles the constraints by projecting the policy back onto the closest feasible policy. This allows the agent to improve the reward while ensuring constraint satisfaction simultaneously.

PCPO provides a way to recover from an infeasible set and presents theoretical performance bounds. Since PCPO uses TRPO to update the policy with a quadratic approximation in a fashion similar to CPO, it also inherits the drawbacks of expensive computation and limited generality that CPO has.

Constraint	Method	References	Performance	Scalability	Computation	Theorem
Cumulative	Lagrangian	[Altman, 1999]	✓	✓	×	×
	CPO	[Achiam *et al.*, 2017]	−	✓	×	✓
	IPO	[Liu *et al.*, 2020b]	✓	✓	✓	−
	PCPO	[Yang *et al.*, 2020]	✓	✓	×	✓
	Lyapunov	[Chow *et al.*, 2019]	✓	✓	×	×
	BMC	[Satija *et al.*, 2020]	✓	✓	✓	✓
	State Augmentation	[Xu and Mannor, 2011]	−	×	✓	−
Instantaneous	Lagrangian	[Bohez *et al.*, 2019]	✓	✓	×	×
	Safety layer	[Dalal *et al.*, 2018]	✓	✓	✓	×
	GP	[Wachi and Sui, 2020]	✓	✓	✓	✓
	Human	[Saunders *et al.*, 2017]	✓	×	✓	×

Table 1: Overview of the representative approaches for Constrained RL.

3.5 Lyapunov-based Approaches

Lyapunov-based approaches [Chow *et al.*, 2018; Chow *et al.*, 2019] are methods of solving the expected cumulative constraint problem by constructing Lyapunov functions, which are a type of scalar potential functions used to compute the stability of a system [Drazin and Drazin, 1992]. They provide an effective way to guarantee cumulative constraint satisfaction via a set of instantaneous, linear constraints. Empirical results show that Lyapunov-based approaches work well and are compatible with different RL algorithms; these methods, however, require solving a Linear Program (LP) at every interaction step to construct the Lyapunov constraints.

3.6 Backward Markov Chain (BMC)

BMC [Satija *et al.*, 2020] proposes converting expected cumulative constraints into instantaneous constraints; the original cumulative constraints are satisfied by satisfying the instantaneous constraints. The instantaneous constraints are defined via Backward Value Functions [Morimura *et al.*, 2010], which act as estimators of the expected cost collected by the agent so far. Using backward value functions makes the system Markovian, which in turn allows the agent to derive analytical solutions for instantaneous constraints. The optimization problem in this method can be easily solved with little approximation.

3.7 State Augmentation

Optimizing CMDPs with probabilistic constraints is an NP-hard problem [Xu and Mannor, 2011]. Nevertheless, [Xu and Mannor, 2011] propose a pseudo-polynomial algorithm that converts the probabilistic constraints to cumulative constraints. In their method, a new CMDP is constructed by doing state augmentation. Given a policy π of the new CMDP, the expected values of the cumulative reward and cost in the new CMDP equal the expected cumulative reward and probability of cumulative cost meets the target in the original MDP, respectively. While the method works well for small-scale and finite CMDPs, it does not scale well.

4 Instantaneous Constraints

4.1 Lagrangian Relaxation

Lagrangian relaxation methods work for instantaneous constraints as well. In these methods [Bohez *et al.*, 2019; Bhatia *et al.*, 2019], the reward is augmented with the sum of the constraint penalty weighted by the Lagrange multipliers in each step. This new reward is defined as

$$R_i(s_t, a_t, s_{t+1}) - \sum_i \lambda_i (C_i(s_t, a_t, s_{t+1}) - \omega_i), \qquad (10)$$

This method solves the instantaneous constraint problem to some degree. However, it can be difficult to optimize in practice as the optimization need to happen in each RL step. In addition, it is inevitable that constraints will be violated in the training process.

4.2 Safety Layer

To satisfy instantaneous constraints, a natural approach is to correct action at each step by projecting actions onto a feasible space. This can be done by introducing a so-called safety layer to the end of the policy network that performs this projection. In a given state, the unconstrained policy outputs an action and then passes it to the safety layer, which projects the action to the nearest feasible action. The role of the safety layer is to solve

$$\begin{aligned} &\min_{a'_t} \frac{1}{2} \left\| a'_t - a_t \right\|^2 \\ &s.t.\ C_i(s_t, a'_t, s'_{t+1}) \leq \omega_i \end{aligned} \qquad (11)$$

where a'_t is the new feasible action, a_t is the infeasible primitive policy output, $C_i(s_t, a'_t, s'_{t+1})$ is the instantaneous constraint signal under new action a' given state s at time t, and ω_i is the predefined constraint. A representative application is to restrict motions of robot arm [Pham *et al.*, 2018]. The safety layer idea can apply to both explicit and implicit constraints. One challenge is that for implicit instantaneous constraints the function $C_i(\cdot)$ may be unknown. To address this problem, one can use another neural network to learn the constraint function $C_i(\cdot)$ [Dalal *et al.*, 2018]. If the constraints are of linear nature [Dalal *et al.*, 2018; Chow *et al.*, 2019] then the Euclidean distance projection can be approximated by a linear projection.

Global Sum Constraints [Bhatia *et al.*, 2019] are a special case of explicit instantaneous constraints that can be satisfied with a softmax safety layer. They appear frequently in resource allocation scenarios [Bhatia *et al.*, 2019]. Assuming that an action is a vector with k entries, denoted

as $a = [a_1, a_2, ..., a_k]^T$, and we have the specific explicit instantaneous constraint $\sum_k a_k = 1$. A softmax safety layer is added right after the output layer of the policy network π_θ to project the arbitrary action a to a feasible action $a' = [a'_1, a'_2, ..., a'_k]^T$:

$$a'_i = \frac{e^{a_i}}{\sum_{j=1}^{k} e^{a_j}}. \tag{12}$$

One thing to note is that when the policy is parameterized with a Gaussian distribution – a standard choice in Deep RL to handle continuous environments – the action projection can lead to a biased gradient [Chou *et al.*, 2017; Fujita and Maeda, 2018] in policy gradient algorithms.

4.3 Gaussian Processes

Since implicit instantaneous constraints are unknown in the model-free environment, some works attempt to model them as Gaussian Processes (GPs) [Turchetta *et al.*, 2016; Wachi *et al.*, 2018; Wachi and Sui, 2020] and then use these models to predict the constraint values of neighboring states. This method requires two main assumptions: 1) the agent starts in a feasible initial state set and 2) regularity of the instantaneous cost, so that similar states have similar cost values. By modeling unknown costs with GPs, an agent partitions the state space into feasible, uncertain, and infeasible regions. The algorithm optimizes the cumulative reward by exploring the feasible regions and the goal is to identify the largest possible feasible regions.

4.4 Human in The Loop

Instantaneous constraints require constraint satisfaction at each step. However, all of the above methods cannot guarantee that they achieve zero violation. This is especially true for implicit instantaneous constraints, where visiting infeasible regions is inevitable as the cost value is unknown to the agent.

For model-free reinforcement learning, having a human "in the loop" who is ready to intervene is currently the only way to prevent all infeasible actions. In [Saunders *et al.*, 2017], a human constantly watches the interface between the RL agent and environment and blocks any infeasible action before it happens. Then a scheme uses a supervised imitation learner to imitate the human intervened policy. Although incorporating learning with human intervention can significantly reduce constraint violation, it is difficult to scale to complex environments because the human time-cost would be prohibitive.

5 Evaluation

5.1 Benchmarks

In order to evaluate the efficacy of various constrained RL approaches, a set of standard benchmarks and metrics is necessary for comparison. In this section, we outline popular environments that provide test cases for benchmarking algorithms that solve the constrained RL. All the environments are configurable so that the user can design their own depending on the type of constraints they hope to study.

Traditional Mujoco Environments

The tasks presented here, Circle and Gather [Achiam *et al.*, 2017] are chosen based on their intuitive definition and their inclusion of a clear constraint component. These tasks can be trained on agents of various complexity: a point-mass ($S \subseteq \mathbb{R}^9, A \subseteq \mathbb{R}^2$), an 'ant' (a quadruped robot, $S \subseteq \mathbb{R}^3 2, A \subseteq \mathbb{R}^8$), and a humanoid ($S \subseteq \mathbb{R}^{102}, A \subseteq \mathbb{R}^{10}$). Each agent-task combination is available with the tasks outlined below with the exception of Humanoid-Gather:

Circle: In the circle task, agents are rewarded for running in a wide circle of predefined radius, yet constrained to stay within a safe region smaller than the radius of the circle.

Gather: In the gather task, agents are rewarded for collecting apples while being constrained to avoid bombs. Each apple collected results in a reward while collecting a bomb results in a cost.

Safety Gym

Recently, OpenAI developed the Safety Gym benchmark [Ray *et al.*, 2019] which is inspired by safety concerns that have been observed in robotics control problems. Here, each environment consists of a robot that is asked to traverse a cluttered environment to accomplish one of a variety of tasks, while satisfying constraints on its interactions with the objects that clutter its environment. Like the tasks outlined above, these tasks allow for mixing and matching various components to allowing for varying levels of difficulty and complexity when testing algorithms. The agent, task, and constraint options are highly configurable; we outline the premade options below.

Agents: Safety-gym includes three robots of increasing complexity:

1. **Point** is a simple 2D-robot with two actuators – one for turning and one for moving forward or backward.
2. **Car** has three wheels, the front two of which are independently driven and a free-rolling rear wheel. Moving this robot requires coordination of the actuators that drive the front wheels.
3. **Doggo** is a quadrupedal robot with bilateral symmetry, designed so that a uniform random policy keeps the robot upright and generates some movement. Each of its four legs has three controls – two at the hip for azimuth and elevation relative to the torso and one controlling angle in the knee.

Tasks: These tasks are mutually exclusive, and can be configured to have either sparse or dense rewards.

1. The **Goal** task involves moving the robot to a progression of goal positions.
2. The **Button** task asks the robot to press a series of buttons. The buttons are immobile and the agent is tasked with press the currently highlighted one.
3. The **Push** task asks the robot with moving a box to a series of goal positions.

Constraints: Each of the elements below provides a different form of dynamics for the agent to understand and learn to avoid. These elements can be mixed and matched as the user

desires. At each step, the environment gives a cost signal for each of the unsafe elements, in addition to an aggregate cost signal that reflects the overall interaction with the elements.

1. **Hazards** are non-physical circles on the ground; the agent is penalized for entering them.
2. **Vases** represent fragile objects; the agent is penalized for touching or moving them.
3. **Pillars** are rigid barriers that the agent should not touch.
4. **Buttons** imitate goals; pressing one of these fake buttons results in a penalty.
5. **Gremlins** are moving objects that the agent should not touch.

5.2 Metrics

Along with the above benchmarks, we propose a set of metrics to use when evaluating the efficacy of a constrained RL algorithm. These metrics aim to provide a uniform answer to the following question: how high of a cumulative discounted reward can we achieve while simultaneously ensuring that constraints are satisfied? As such, we suggest the following metrics:

Reward: The expected cumulative return (Eq. 1).

Cumulative constraints:

- The expected cumulative cost (Eq. 3 and 4) or probabilistic cumulative cost (Eq. 5).
- The sum of all costs divided by the total number of interaction steps, a proxy for regret.

Instantaneous constraints:

- The total number of constraint violations.
- The violation rate: the total number of constraint violations divided by the total number of interaction steps.

6 Discussions

6.1 Theoretical Guarantees

In addition to algorithm development, it is useful to develop a rigorous understanding of the developed algorithms. Performance measures for learning algorithm performance include 1) eventual convergence to optimality, 2) the speed of convergence to optimality, and 3) regret analysis [Kaelbling *et al.*, 1996]. While it is difficult, if not impossible, to analyze the performance of a general deep neural network, researchers have analyzed learning-algorithm performance under more restrictive conditions. Convergence has been studied under the assumption of convexity [Le *et al.*, 2019], and regret has been studied in both finite [Azar *et al.*, 2017; Jin *et al.*, 2018; Qiu *et al.*, 2020; Efroni *et al.*, 2020] and infinite horizons [Singh *et al.*, 2020].

In [Wu *et al.*, 2015], the authors analyzed regrets for constrained multi-armed bandit (MAB) problems. MABs are a simple version of MDPs, where agent actions do not change the state of the system. In [Wu *et al.*, 2015], the authors consider constrained contextual bandits, contextual bandits with budget and time constraints. They show that their proposed algorithm achieves logarithmic regret except for certain boundary cases. Moving beyond bandits, researchers have started to develop regret analysis for finite horizon RL [Qiu *et al.*, 2020; Efroni *et al.*, 2020] – also known as episodic RL – in which the agent interacts with the environment in a finite sequence of episodes. Furthermore, [Singh *et al.*, 2020] shed light on the infinite horizon problem by bounding the reward and costs simultaneously during unknown long term runs.

6.2 Zero Constraint Violation

Another issue is constraint violation during the policy training process. Approaches to solving cumulative constraints often violate constraints during training but produce a trained policy that is able to respect the constraints. Explicit instantaneous constraints can achieve zero constraint violation in most cases since they can be checked before taking actions. For implicit instantaneous constraints, only the human in the loop method [Saunders *et al.*, 2017] prevents constraint violation; this method, however, does not scale to complex environments. Ideally, we would like a method that can achieve zero constraint violation during training and not just at the final optimal policy.

It is intractable to avoid all constraint violations in the model-free setting as the agent needs to acquire environment knowledge through trial and error. One possible approach is to use more stringent thresholds when defining constraints in the CMDP; this method, however, is not systematic. An open question is how to provide a priori knowledge efficiently to avoid constraint violation. Another complementary strategy is to design better recovery methods to prevent an agent from getting stuck in an infeasible region.

6.3 Mixed Constraints

In this paper, we discussed cumulative and instantaneous constraints separately, with the knowledge that one can sometimes be converted to the other [Satija *et al.*, 2020; Xu and Mannor, 2011]. In practice, various types of constraints may occur simultaneously. For example, our application paper [Liu *et al.*, 2020a] handles cumulative and instantaneous (explicit and implicit) constraints simultaneously for network slicing resource allocation. Further efforts are needed to combine multiply types of constraints.

7 Conclusion

This article discusses policy learning with constraints in model-free reinforcement learning. We formulate the constraint learning problem as a CMDP and consider two main kinds of constraints: cumulative and instantaneous. In order to evaluate algorithm performance, we briefly introduce benchmarks and metrics for constrained reinforcement learning. We also highlight limitations of current methods and present promising directions for future research: algorithms with theoretical guarantees, ability of achieving zero constraint violation and handling mixed constraints.

Acknowledgments

The work was partially supported by NSF through grants CNS-1718901, IIS-1838207, CNS 1901218, OIA-2040680, and USDA-020-67021-32855.

References

[Achiam *et al.*, 2017] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 22–31. JMLR. org, 2017.

[Altman, 1999] Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

[Azar *et al.*, 2017] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 263–272. JMLR. org, 2017.

[Bhatia *et al.*, 2019] Abhinav Bhatia, Pradeep Varakantham, and Akshat Kumar. Resource constrained deep reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 610–620, 2019.

[Bohez *et al.*, 2019] Steven Bohez, Abbas Abdolmaleki, Michael Neunert, Jonas Buchli, Nicolas Heess, and Raia Hadsell. Value constrained model-free continuous control. *arXiv preprint arXiv:1902.04623*, 2019.

[Boyd and Vandenberghe, 2004] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[Chen *et al.*, 2021] Yi Chen, Jing Dong, and Zhaoran Wang. A primal-dual approach to constrained markov decision processes. *arXiv preprint arXiv:2101.10895*, 2021.

[Chou *et al.*, 2017] Po-Wei Chou, Daniel Maturana, and Sebastian Scherer. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *International conference on machine learning*, pages 834–843. PMLR, 2017.

[Chow *et al.*, 2017] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

[Chow *et al.*, 2018] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8092–8101, 2018.

[Chow *et al.*, 2019] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Mohammad Ghavamzadeh, and Edgar Duenez-Guzman. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.

[Dalal *et al.*, 2018] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

[Drazin and Drazin, 1992] Philip G Drazin and Philip Drazin Drazin. *Nonlinear systems*. Number 10. Cambridge University Press, 1992.

[Dulac-Arnold *et al.*, 2019] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[Efroni *et al.*, 2020] Yonathan Efroni, Shie Mannor, and Matteo Pirotta. Exploration-exploitation in constrained mdps. *arXiv preprint arXiv:2003.02189*, 2020.

[Fujita and Maeda, 2018] Yasuhiro Fujita and Shin-ichi Maeda. Clipped action policy gradient. In *International Conference on Machine Learning*, pages 1597–1606. PMLR, 2018.

[Garcıa and Fernández, 2015] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[Geibel, 2006] Peter Geibel. Reinforcement learning for mdps with constraints. In *European Conference on Machine Learning*, pages 646–653. Springer, 2006.

[Henri *et al.*, 2020] Gonzague Henri, Tanguy Levent, Avishai Halev, Reda Alami, and Philippe Cordier. pymgrid: An open-source python microgrid simulator for applied artificial intelligence research. *arXiv preprint arXiv:2011.08004*, 2020.

[Jin *et al.*, 2018] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? In *Advances in Neural Information Processing Systems*, pages 4863–4873, 2018.

[Kaelbling *et al.*, 1996] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[Le *et al.*, 2019] Hoang Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pages 3703–3712. PMLR, 2019.

[Levine *et al.*, 2016] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[Liang *et al.*, 2018] Qingkai Liang, Fanyu Que, and Eytan Modiano. Accelerated primal-dual policy optimization for safe reinforcement learning. *arXiv preprint arXiv:1802.06480*, 2018.

[Liu *et al.*, 2020a] Yongshuai Liu, Jiaxin Ding, and Xin Liu. A constrained reinforcement learning based approach for network slicing. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE, 2020.

[Liu *et al.*, 2020b] Yongshuai Liu, Jiaxin Ding, and Xin Liu. Ipo: Interior-point policy optimization under constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4940–4947, 2020.

[Liu *et al.*, 2021] Yongshuai Liu, Jiaxin Ding, and Xin Liu. Resource allocation method for network slicing using constrained reinforcement learning. In *The International*

Federation for Information Processing (IFIP) Networking 2021, pages 1–3. IEEE, 2021.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[Morimura *et al.*, 2010] Tetsuro Morimura, Eiji Uchibe, Junichiro Yoshimoto, Jan Peters, and Kenji Doya. Derivatives of logarithmic stationary distributions for policy gradient reinforcement learning. *Neural computation*, 22(2):342–376, 2010.

[Pham *et al.*, 2018] Tu-Hoa Pham, Giovanni De Magistris, and Ryuki Tachibana. Optlayer-practical constrained optimization for deep reinforcement learning in the real world. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6236–6243. IEEE, 2018.

[Qiu *et al.*, 2020] Shuang Qiu, Xiaohan Wei, Zhuoran Yang, Jieping Ye, and Zhaoran Wang. Upper confidence primal-dual optimization: Stochastically constrained markov decision processes with adversarial losses and unknown transitions. *arXiv preprint arXiv:2003.00660*, 2020.

[Ray *et al.*, 2019] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 2019.

[Satija *et al.*, 2020] Harsh Satija, Philip Amortila, and Joelle Pineau. Constrained markov decision processes via backward value functions. In *International Conference on Machine Learning*, pages 8502–8511. PMLR, 2020.

[Saunders *et al.*, 2017] William Saunders, Girish Sastry, Andreas Stuhlmueller, and Owain Evans. Trial without error: Towards safe reinforcement learning via human intervention. *arXiv preprint arXiv:1707.05173*, 2017.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Shani *et al.*, 2005] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. An mdp-based recommender system. *Journal of Machine Learning Research*, 6(9), 2005.

[Singh *et al.*, 2020] Rahul Singh, Abhishek Gupta, and Ness B Shroff. Learning in markov decision processes under constraints. *arXiv preprint arXiv:2002.12435*, 2020.

[Stooke *et al.*, 2020] Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pages 9133–9143. PMLR, 2020.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Sutton *et al.*, 1999] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPs*, volume 99, pages 1057–1063. Citeseer, 1999.

[Tessler *et al.*, 2018] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

[Turchetta *et al.*, 2016] Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe exploration in finite markov decision processes with gaussian processes. *arXiv preprint arXiv:1606.04753*, 2016.

[Vinyals *et al.*, 2019] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[Wachi and Sui, 2020] Akifumi Wachi and Yanan Sui. Safe reinforcement learning in constrained markov decision processes. In *International Conference on Machine Learning*, pages 9797–9806. PMLR, 2020.

[Wachi *et al.*, 2018] Akifumi Wachi, Yanan Sui, Yisong Yue, and Masahiro Ono. Safe exploration and optimization of constrained mdps using gaussian processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[Wu *et al.*, 2015] Huasen Wu, R Srikant, Xin Liu, and Chong Jiang. Algorithms with logarithmic or sublinear regret for constrained contextual bandits. In *Proceedings of the Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015.

[Xu and Mannor, 2011] Huan Xu and Shie Mannor. Probabilistic goal markov decision processes. In *Twenty-Second International Joint Conference on Artificial Intelligence*. Citeseer, 2011.

[Yang *et al.*, 2020] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152*, 2020.