Data Efficient Safe Reinforcement Learning

Sindhu Padakandla*, Prabuchandran K J†, Sourav Ganguly† and Shalabh Bhatnagar*
* Dept. of Computer Science and Automation, Indian Institute of Science
† Dept. of Computer Science Engineering, IIT Dharwad

***Abstract*—Applying reinforcement learning (RL) methods for real world applications pose multiple challenges - the foremost being safety of the system controlled by the learning agent and the learning efficiency. An RL agent learns to control a system by exploring the available actions in various operating states. In some states, when the RL agent exercises an exploratory action, the system may enter unsafe operation, which can lead to safety hazards both for the system as well as for humans supervising the system. RL algorithms thus must learn to control the system respecting safety. In this work, we formulate the safe RL problem in the constrained off-policy setting that facilitates safe exploration by the RL agent. We then develop a sample efficient algorithm utilizing the cross-entropy method. The proposed algorithm's safety performance is evaluated numerically on benchmark RL problems.**

***Index Terms*—Safe exploration, Constrained RL, Off-Policy**

I. Introduction

Resurgence of artificial intelligence and advancements in it has led to automation of many cyber-physical systems [1], [2]. Such systems have multiple operating states and they evolve based on what the underlying intelligent agent or the controller decides to do. The prime factor for efficient system operation lies in the control decisions of the agent.

The agent interacting with a system and controlling these operating states can be modelled using Markov Decision Process (MDP) framework [3], [4]. The agent learns to control the system by exploring various actions and receiving 'reward or cost' feedback for the choice of actions in an iterative manner. This approach of learning to choose the optimal actions based on indirect feedback is known as reinforcement learning (RL) [6]. The success of the RL approach has been demonstrated on a broad range of applications [5]–[7]. However, a grave concern arises when we speak of these potential applications of RL, which is that of safety [8]. This problem is of profound importance in systems like self-driving cars, where during learning, any decision/action applied to the car has to guarantee safe driving or at least minimize the frequency of unsafe behaviour.

In this paper, we focus on issues arising in safety critical systems [9] controlled by RL algorithms and bring in safe exploration of actions utilizing two important ideas - a) The first enables a RL agent to explore decisions whose average cost violations are well within a specified budget. This is theorized on the constrained RL framework [10] (b) the second enables the agent to learn optimal actions for every state based on data from other safe policies (which may not be optimal). These two ideas lead us to develop a RL algorithm based on cross-entropy method (CEM), which helps in optimization of policies without computing/estimating derivatives. The CEM can handle constraints effectively and at the same time is amenable to off-policy learning. Further, our RL algorithm can utilize the non-linear function approximations provided by neural networks for large scale problems. We summarize our key contributions:

- Develop sample-efficient RL algorithm for safe exploration based on cross entropy method.
- Given safety threshold, i.e., a quantity which indicates the level of safety violations that can be tolerated by the system, our algorithm utilizes the available data to find a policy that is feasible. Also, the safety constraints are kept under check during the learning process as well.
- Our algorithm utilizes budgeted/limited data samples for learning the optimal policy.

II. Background

A Markov decision process (MDP) [4] is formally defined as a tuple $M = \langle S, A, P, R, \gamma \rangle$, where S is the set of states of the system, A is the set of actions (or decisions) that can be taken in these states (assume $|A| < \infty$ and all actions are feasible in every state), $P : S \times A \times S \to [0, 1]$ is the transition probability function that governs the dynamics of the system based on the choice of actions in states, $R : S \times A \to \mathbb{R}$ is a real-valued reward feedback for choosing actions in states, $0 \leq \gamma < 1$ denotes the discount factor and $\beta : S \to \mathbb{R}$ denotes the initial distribution over states. A stationary deterministic policy (SDP) $d : S \to A$ is a state-dependent decision rule that prescribes the action to be taken in state s, $\forall t \geq 0$. For an infinite horizon MDP, the often used performance measure is the expected sum of discounted rewards (value function) of a SDP d which is defined as:

$$V^d = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, d(s_t)) | s_0 \sim \beta(s)\right] \tag{1}$$

A. Safety Formulation

As discussed in Section I, an RL agent has to learn safe policies for safety-critical applications and not mere policies that maximize (1). This brings in a clear case of trade-off in the learning process, i.e., the agent needs to maximize discounted reward return, however, at the same time can only explore actions that do not jeopardize the system functioning. Thus, for safe reinforcement learning, multiple formulations have been attempted in the literature as described below:

(a) Risk based objective functions: To assess average risk of a policy, its performance criterion is specified in terms of measure like conditional value at risk (CVAR) [11] and risk sensitive MDPs [12].

(b) Safety Constraints: Basic formulation of MDPs extended to constrained MDP formulation

$$\max_{d \in D} \; V^d \;\; \text{s.t.} \;\; S_i(\pi) \leq \alpha_i, 1 \leq i \leq l$$

where D is set of all stationary deterministic policies, $S_i \;\; 1 \leq i \leq l$ are expected long term safety constraints and α_i are corresponding threshold values.

Manuscript 374 submitted to 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Received April 28, 2022.

(c) Exploring strategies: The learning algorithm follows varied prior exploration strategies to collect more information about the MDP model [13].
(d) Modelling safety constraint: Safety constraint $f(x)$ is modelled as a Gaussian Process [14], which is estimated from historic data.

Prior works on safe RL (see [15]) utilize one of the aforementioned safety formulations. Constrained policy optimization approaches [16], [17] utilize problem formulation (a) and (b) as described above. A trust-region [16] based method proposes surrogate objective and constraint functions to suit model-free optimization. [17] proposes a constrained RL optimization algorithm adapted from cross-entropy method [18]. In [18], using trajectory data samples, the objective function as well as the safety constraints are computed. Based on the computed values, the policy is learnt. As with [16], the method in [18] also requires exploration of all actions (including unsafe) for evaluating a policy.

III. Proposed Approach

In this section, we describe our safe RL formulation utilizing safety formulations (b) and (c) of Section II-A.

A. Problem formulation

A safety constrained problem is modeled as a constrained MDP (CMDP) such that $M = \langle S, A, P, R, c_1, c_2, \ldots, c_K \rangle$ where S, A, P and R are same as in Section II. The additional terms $c_1, c_2, \ldots, c_K$ denotes K constraints. $c_i : S \times A \rightarrow \mathbb{R}, 1 \leq i \leq K$. For any $n \geq 0$, let $r(n), c_1(n), c_2(n) \ldots c_K(n)$ denote single stage reward and constraint values at instant n. We assume that reward and constraints are (uniformly) bounded. The state evolution of the Markov process under any stationary policy π is denoted $\{X_n\}$ and corresponding action sequence as $\{Z_n\}$. At instant n, when state X_n is observed and action Z_n is taken, a reward $r(n)$ and constraint costs $c_1(n), \ldots, c_K(n)$ are obtained. The process moves to state X_{n+1}. Let $R(s,a) = \mathbb{E}[r(n)|X_n = s, Z_n = a]$ and $c_i(s,a) = \mathbb{E}[c_i(n)|X_n = s, Z_n = a], 1 \leq i \leq K$.

The set of policies we consider is specified by a state-dependent randomized decision rule $\pi : S \rightarrow \mathcal{P}(A)$. Here, $\mathcal{P}(A)$ denotes probability distribution over action space A. Let the probability of picking action a in state s be denoted as $\pi(a|s)$. Additionally, when two policies are involved (say μ and π), we differentiate between them using this notation as $\mu(a|s)$ and $\pi(a|s)$, which give the respective probabilities of taking action a under both policies in state s.

Let $U^\pi(s) = \mathbb{E}\left[\sum_{m=0}^{\infty} \gamma^m r(m)|X_0 = s, \pi\right]$ and $W_i^\pi(s) = \mathbb{E}\left[\sum_{m=0}^{\infty} \gamma^m c_i(m)|X_0 = s, \pi\right]$, which are the long-run expected sum of reward and i$^{\text{th}}$ constraint values, respectively. Let β be initial distribution over states. The value function of policy π w.r.t β is defined as $V^\beta(\pi) = \sum_{s \in S} \beta(s) U^\pi(s)$. Further, we define $G_i^\beta(\pi) = \sum_{s \in S} \beta(s) W_i^\pi(s)$ as the long term cost measure w.r.t constraint function i and initial state distribution β.

Given this specification, our constrained safe RL formulation is the following:

$$\max_{\pi \in \Pi} \; V^\beta(\pi) \;\; \text{s.t.} \;\; G_i^\beta(\pi) \leq \alpha_i. \tag{2}$$

The key problem in (2) unlike (1) is that the RL agent needs to find a randomized policy π^* which not only maximizes V^β but also results in each of the safety functions G_i^β to be lower than a threshold value α_i. A policy which respects the threshold on the constraint functions is said to be *feasible*. In order to make the optimization tractable, we restrict our attention to the randomized policy space which is parameterized by $\theta \in \mathbb{R}^p$. This space is denoted as $\Pi_\Theta = \{\pi^\theta(a|s) : s \in S, a \in A, \theta \in \mathbb{R}^p\}$.

B. Off Policy Learning

Even though the constrained formulation (2) incorporates safety aspect in the policy learnt by RL algorithm, it does not restrict the kind of actions the RL agent can explore during learning. This can potentially jeopardize the system. Therefore, to impose restriction on the exploration of actions, we consider off-policy learning as opposed to on-policy learning. In off-policy learning, the agent is provided with an existing (historic) dataset or a behaviour policy from which the agent is allowed to choose actions. The goal of the agent is to learn an optimal policy meeting the safety constraints based on experience samples from the behaviour policy. Note this is a harder learning problem as the agent gets only indirect feedback when using the behaviour policy as opposed to executing and evaluating the current policy in the on policy setting. In the off-policy learning, the amount of exploration the agent can perform is limited and with that limited exploration, agent is expected to learn feasible policies that maximize the objective.

In the off-policy learning setup, the RL agent is endowed with an input data set $\mathcal{D}$, a finite collection of *experience* tuples, i.e., tuples of the form $e_n = (s(n), a(n), r(n), c_1(n), \ldots, c_K(n))$ to compute a good feasible policy. Note that the data set $\mathcal{D} = \{e_0, e_1, \ldots, e_T : 1 < T < \infty\}$ is generated from a behaviour policy μ (a randomized policy). The agent has no model information concerning P, R and c_i and is provided with the threshold values α_i, $1 \leq i \leq K$.

C. Constrained RL optimization: Cross Entropy Method (CEM) vs Lagrangian

Prior algorithms for constrained RL setting utilize the Lagrangian theory to derive actor critic type algorithms [19]. For large complex problems, recent works [16], [20] utilize neural networks for function approximation in this setting. These works compute the gradient of the unconstrained objective by including the constraint into the objective. This computation typically involves on-policy learning or parallel simulations, both of which pose obvious limitations. Additionally, these algorithms do not satisfy the constraints during learning [16]. These limitations motivate us to adapt other optimization solution techniques for solving (2) and we consider the cross entropy method as it is simple and appropriate in the constrained off-policy setting.

Cross entropy method (CEM) [21] is a zero-order optimization method. Hence, it can be used in scenarios where

the objective function does not exhibit smoothness properties or obtaining precise gradients or higher order derivatives is challenging. Note that obtaining gradients in the constrained setting is difficult. The basic idea underlying CEM is simple: in each iteration, we generate samples of the objective function according to some parameterized distribution. A set of *elite* samples is chosen corresponding to points where objective function value is high among the samples obtained. Based on these elite samples, the parameters of the distribution from which the points are sampled is updated.

CEM has been utilized in RL [17], [18], [22] as a policy iteration algorithm. In this setting, each sample of the objective function V corresponds to evaluation of a policy sampled from a distribution. Based on the elite samples of V, the distribution from which the policies are obtained are updated. [18] attempted at improving policy over multiple iterations for a finite-horizon finite MDP setting. In later works [17], [22] that utilize CEM, parameterization of policy space has been considered. [22] proposed a stochastic approximation variant of CEM with policy and value function approximation in a simple MDP setting and [17] proposed CEM solution for constrained on-policy setting. In [17] episodic constrained RL problems were considered while [22] dealt with discounted unconstrained RL problems. None of the earlier algorithms considered off-policy learning.

D. Off-Policy Constrained Safe RL Algorithm

The method we propose (shown in Algorithm 1) generalizes CEM for off-policy setting. This alleviates the problems associated with prior approaches described above. The procedure in Algorithm 1 is described for $K = 1$ but the same algorithm works for any $K > 1$.

1) Approximation in Policy Space, Value and Constraint function: The algorithm utilizes a dataset $\mathcal{D}$ consisting of experience tuples generated by following a behaviour policy μ, which is a SRP. The RL agent has no access to any other dataset, forcing it to learn good feasible policies using samples only from $\mathcal{D}$. This scenario is very common in robotic applications where certain trajectory samples are collected using expert supervision and the RL agent needs to learn optimal behaviour strategy from only the available samples.

The policy search space is parameterized by θ and it gets sampled from a distribution $f \in \mathcal{C}$. In Algorithm 1, we consider Gaussian distributed policy parameterization and denote this by $\mathcal{F}_{\mathcal{V}} = \{f_v : f_v = \mathcal{N}(v, \sigma^2\mathcal{I})\}$ with tunable parameters v and σ. We sample θ from a distribution $f_v \in \mathcal{F}_{\mathcal{V}}$. For a given parameter vector θ, the analytical form π^θ (target policy) is defined using softmax function of fixed features $\psi(s,a) \in \mathbb{R}^l$, $\forall (s,a) \in S \times A$. For this policy, using the samples from $\mathcal{D}$, the RL agent computes the expected objective and constraint function values using off-policy estimation. The objective and constraint function values are approximated using linear function approximation given the features $\phi(s)$ for every state s.

2) Objective and safety constraint evaluation: The OFF-POLICY-PREDICT subroutine tunes the weights $x, y \in \mathbb{R}^m$, such that $\phi^\top x \approx V^\beta(\pi^\theta)$ and $\phi^\top y \approx G_1^\beta(\pi^\theta)$. For off-policy evaluation, we adapt Emphatic TD [23] (ETD) and Perturbed TD [24] (PTD) algorithms for estimating the value function and the constrained value function. Let $\eta_t = \frac{\pi^\theta(a_t|s_t)}{\mu(a_t|s_t)}$ be the importance sampling ratio at epoch t. Similarly, let x_t be the weight of V^β and y_t the weight vector of G_1^β in iteration t. The temporal difference error w.r.t reward is defined as $\delta_t^x = R(s_t, a_t) + \gamma x_t^\top \phi(s_{t+1}) - (1 + tdf) * x_t^\top \phi(s_t)$, where tdf is a scaling factor. For ETD update, $tdf = 0$, while for PTD update, $tdf > 0$.

We define the eligibility trace iterate starting at $b_{-1} = 0$ and a follow-on trace iterate $F_0 = 1$. These are updated along with the weight vector x_t. The update rule for x_t utilizes a function $h(F_t, \gamma, \eta_t, b_t, \phi(s_t))$ for scaling the temporal difference term δ_t^x. The form of update is as follows:

$$x_{t+1} = x_t + stepsize * \delta_t^x * h(F_t, \gamma, \eta_t, b_t, \phi(s_t)). \quad (3)$$

The function $h(F_t, \gamma, \eta_t, b_t, \phi(s_t))$ differs in ETD and PTD algorithms, the details of which are in [23], [24]. Weights corresponding to the first constraint function, i.e., y_t is updated similar to (3), where x_t is replaced by y_t and δ_t^x is replaced with δ_t^y. The step size chosen is usually diminishing according to stochastic approximation assumptions [19].

3) Policy Parameter Updation: The policy parameter distribution is updated using the output of OFF-POLICY-PREDICT. A number of policy parameters are sampled and their values are estimated (Lines 4-7). These samples are collected and the POLICY-UPDATE subroutine updates this distribution in a manner that the trade-off between value and constraint functions is maintained (Line 8). This takes place over a number of iterations (Lines 3-10). The number of θ samples for policy iteration j is denoted as n_j. For each such sample, the approximate values of $V^\beta(\pi^\theta)$ and $G_1^\beta(\pi^\theta)$ are computed using $\mathcal{D}$. Further, the computed pairs are sorted and the number of samples satisfying the constraint threshold α_1 are observed. Let this number be denoted as m_j. If $m_j < \lfloor \rho n_j \rfloor$, then this implies we have deficient samples. So, θ from the remaining $(n_j - m_j)$ samples whose $V^\beta(\pi^\theta)$ values are high are used for updating the policy parameter distribution. Otherwise if $m_j > \lfloor \rho n_j \rfloor$, then from the m_j samples of θ, the ones having highest values of $V^\beta(\pi^\theta)$ are selected for updating the policy parameter distribution. The selected sample indices are contained in H_j in iteration j (Line 3). The θ samples in indices H_j are used to update v_j as shown in Line 9 using a weighting factor $0 < s_p < 1$.

Algorithm 1 Off-Policy Constrained Safe RL (OffPol_CCE)

1: **Input:** M, $\{\rho_j,\ 1 \le j \le M\}$, $\beta \in \mathcal{P}(S)$, α_1 and $\mathcal{D} = \{e_0, e_1, \ldots, e_T\}$.
2: **Initialize:** $j = 1$, $v_1 \in \mathbb{R}$, $s_p \in (0,1)$, $\{n_l : 1 \le l \le M, n_l > 10\}$
3: **for** $j = 1$ to M **do**
4: Sample $\theta_1, \ldots, \theta_{n_j} \sim f_{v_j} \in F_{\mathcal{V}}$ i.i.d
5: **for** $i = 1$ to n_j **do**
6: $V^\beta(\pi^{\theta_i})$, $G_1^\beta(\pi^{\theta_i})$ = OFF-POLICY-PREDICT($\mathcal{D}$, θ_i, Π_Θ, Φ, Ψ)
7: **end for**
8: H_j = POLICY-UPDATE($\{(V^\beta(\pi^{\theta_i}), G_1^\beta(\pi^{\theta_i})) : 1 \le i \le n_j\}$, ρ_j, α_1)
9: $v_{j+1} = s_p \sum_{k \in H_j} \frac{V^\beta(\pi^{\theta_k})\theta_k}{\sum_{k \in H_j} V^\beta(\pi^{\theta_k})} + (1 - s_p)v_j$
10: **end for**

Algorithm 2 POLICY-UPDATE($\{(V^{\beta}(\pi^{\theta_i}), G_1^{\beta}(\pi^{\theta_i})) : 1 \leq i \leq n_j\}$, ρ_j, α_1)

Sort $\{(V^{\beta}(\pi^{\theta_i}), G_1^{\beta}(\pi^{\theta_i}))\}$ in ascending order of G_1^{β}. Let sorted order be denoted Υ.
Let m_j be the number of (V^{β}, G_1^{β}) tuples such that $G_1^{\beta} \leq \alpha_1$. Denote $\Lambda = \{\theta_i : G_1^{\beta}(\pi^{\theta_i}) \leq \alpha_1\}$.
$B =$ Null Set
if $m_j < \lfloor \rho_j n_j \rfloor$ **then**
    Sort $\{\theta_i : G_1^{\beta}(\pi^{\theta_i}) \leq \alpha_1\} \subset \Upsilon$ in descending order of V^{β}. Denote this ordered set as Ξ.
    Pick the first $(\lfloor \rho_j n_j \rfloor - m_j)$ samples from Ξ.
    $B = \Lambda \cup \Xi$
else
    Sort Λ in descending order of V^{β}. Pick the first $(\lfloor \rho_j n_j \rfloor)$ elements of Λ and Ξ be the set of θ corresponding to this selection.
    $B = \Xi$
end if
return B

IV. Experimental Results

The performance of our problem is evaluated on different benchmark problems. We further compare our algorithm with the constrained actor-critic algorithm (CAC) [19].

A. Benchmarks

Following benchmark applications are carefully designed to highlight critical safety aspects in autonomous systems. For all benchmarks, the randomized policy space Π_θ consists of softmax policies with temperature parameter.

1) Chain Walk MDP: Chain Walk has 450 states and 2 actions. Agent can either move left or right at each state (see Fig. 1). Transition probabilities are shown in Table I where s is current state and a is action taken in state s. Reward function is $R(150, \cdot) = -10$, $R(300, \cdot) = 100$ and -1 for all other transitions. The constraint values are randomly assigned for each state-action pair by Python MDPtoolbox. This benchmark is a simplified scenario of robot which operates under safety restrictions, wherein some of its movements may cause physical damage.

State and Action	Probability = 0.1	Probability = 0.9
$1 < s < \|S\|$, $a = L$	$s + 1$	$s - 1$
$1 < s < \|S\|$, $a = R$	$s - 1$	$s + 1$
$s = 1$, $a = L$	2	1
$s = 1$, $a = R$	1	2
$s = \|S\|$, $a = L$	$\|S\|$	$\|S\| - 1$
$s = \|S\|$, $a = R$	$\|S\| - 1$	$\|S\|$

TABLE I: Transition probability information for Chain Walk MDP.

2) Threshold Type MDP: The threshold type MDP has 400 states and 4 actions. Transition probabilities, reward and constraint function values are shown in Table II. This structure allows for clear policy behaviour as these policies allow the agent to operate in a subset of the states depending on what is being optimized. This benchmark is a simplified scenario where exploration robots move in a small geographical area which has some unsafe regions. To limit the movement of these robots within a safe area, safety constraints are added.

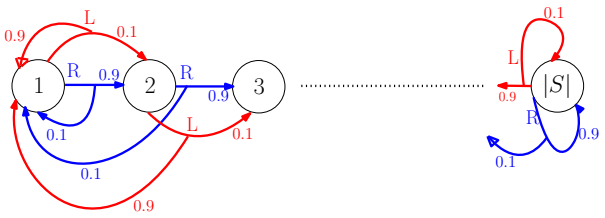


Fig. 1: Chain Walk MDP: Shows transition probabilities for each state and for both actions $\{L, R\}$.

State and Action	Next State	Reward	Constraint
$0 \leq s < 100$, $a \in A$	$100a + s$	10	5
$100 \leq s < 200$, $a \in A$	$100a + (s - 100)$	3	-1
$200 \leq s < 300$, $a \in A$	$100a + (s - 200)$	0	-4
$300 \leq s < 400$, $a \in A$	$100a + (s - 300)$	-3	-8
$s = 1$, $a = R$	1	2	
$s = \|S\|$, $a = L$	$\|S\|$	$\|S\| - 1$	
$s = \|S\|$, $a = R$	$\|S\| - 1$	$\|S\|$	

TABLE II: Transition probability, reward and constraint value information for Threshold Type MDP. Here $A = \{0, 1, 2, 3\}$.

3) Random MDP: In order to show that our algorithm is not dependent on any innate structure present in the MDP, this benchmark is designed to investigate learning behavior of our algorithm. In this benchmark, number of states $|S| = 200$ and number of actions $|A| = 5$. P, R and constraints generated using Python MDPtoolbox.

4) Straight Path Navigation: A race car needs to be controlled by the agent in a manner that it moves on a straight path with minimum deviations on a slippery road. Safety restriction arising out of road slippery conditions require the car to be navigating close to $y = 0$ line. State depends on (x, y) position coordinates, inertial heading(yaw) of the centre of mass ψ and corresponding velocities. Action a is a tuple: $a = \{v, \delta\}$, where v is the agent velocity, δ is steering angle. Reward function is $R(s, a) = -|y| - \lambda_1(v - v_f)^2$ (distance penalty - velocity penalty). The policy space in this benchmark is provided using deep neural network architectures. The dynamics of the vehicle and the tyre models are adapted from [25]. The high level path planner requires the car to move on the line $y = 0$.

B. Analysis of Results

Off policy constrained cross-entropy (OffPol_CCE) method is tested on the Chain Walk MDP and value function learnt by our function is shown in Fig. 2. This plot shows normalized value function (normalizing constant is number of states) when safety constraints also need to be satisfied. The threshold was fixed at 8 for constraint value function. The objective and constraint functions for Threshold Type MDP are as shown in Fig. 3, where threshold value for constraint function is fixed at 0. ETD [23] is used to evaluate the policies in every update. Fig. 3 plots the norm of the objective and constraint values as the agent training proceeds. The variance of the norm is also observed. From these plots it is observed that even though initial policies may or may not be feasible while learning, OffPol_CCE method quickly moves towards policies which are feasible while learning. For the Chain Walk MDP, Fig. 4 plots the proportion of

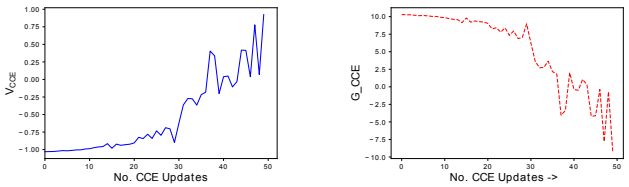


Fig. 2: Chain Walk: (a) Objective and (b) Constraint value function during training of agent using off-policy constrained cross entropy method.

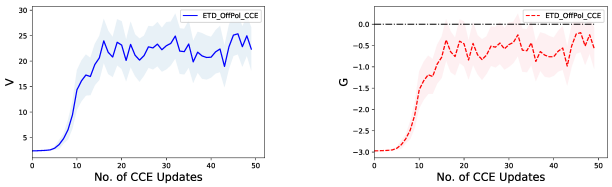


Fig. 3: The (a) Objective and (b) Safety constraint of the policy learnt by the off-policy constrained cross entropy method for the Threshold Type MDP.

visits to all states by our OffPol_CCE policy in comparison to Q-Learning [26] (QL) policy. In Fig. 4, we see that QL policy favours that agent acts in states 0-99 owing to high reward in the states. However, OffPol_CCE policy favours states 100-199 owing to the fact that these states obey the safety constraints.QL learns a policy that maximizes the discounted return ignoring the constraint values. However, with OffPol_CCE, it is observed that the agent learns to operate in states 100-199 for the maximum time, as, in these states the constraint values are low and rewards are also better when compared to states 200-400.

In Fig. 5, the objective function and the constraint values for OffPol_CCE policy with ETD and PTD off-policy algorithms is shown (labelled ETD_OffPol_CCE and PTD_OffPol_CCE resp.). The behaviour policy sample path is obtained with initial state as 0 always. Number of cross-entropy iterations is 100. The threshold for constraint values is 0. We also compare our algorithm with CAC algorithm [19] as shown in Fig. 6 for Random MDP benchmark. The plots emphasize the performance of agent after the policy is learnt in contrast to the plots shown earlier which are

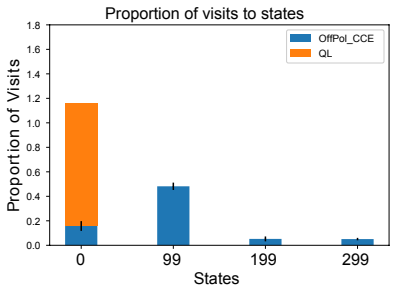


Fig. 4: QL vs. Off policy Constrained cross entropy method: Proportion of visits to states. Computed by evaluating the learnt policy for 4000 steps.

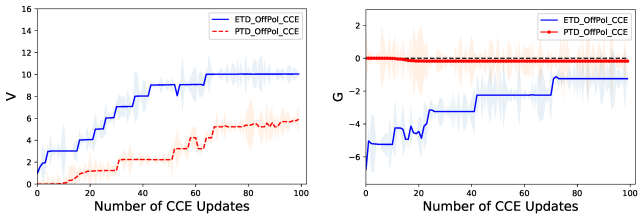


Fig. 5: The objective function and the constraint of the policy learnt by the OffPol_CCE for the Random MDP.

performance during learning. In CAC, none of the weights tuned correspond to CMDP, rather, they are weights for some relaxed MDP showing that it does not track safety during learning. Random MDP has cost objective hence, its objective function needs to be minimized. However same sorting logic in Algorithm 2 holds good. The cost-to-go is averaged over 50 sample paths for 100 decision steps with initial state distribution as β. One important point not emphasized in performance graphs is that the number of data samples used in OffPol_CCE are much lower than CAC although their performance is similar. Fig. 6(a) shows the cost-to-go (i.e. V^{β}) of policies learned by CAC, OffPol_CCE and value iteration. Value iteration has access to all model information unlike CAC and OffPol_CCE. The cost-to-go is averaged over 50 sample paths for 100 decision steps with the initial state distribution β. As we can observe, the policies learnt by CAC and OffPol_CCE are comparable in performance - even though OffPol_CCE utilizes much less information compared to CAC. The value iteration learns a policy which finds the optimal cost-to-go for given transition probability and cost functions. Fig. 6(b) shows the constrained return of policies learnt by CAC, OffPol_CCE and value iteration. The threshold is fixed at 100, i.e., $\alpha_1 = 100$ and the initial state distribution is concentrated on state 0. In this plot, we show two policies learnt in the full information case - one that minimizes the constrained return (i.e., G^{β}) and the other which maximizes G^{β}. These are labelled as MDPT_MinConstr and MDPT_MaxConstr respectively in Fig. 6(b). We can see that these two policies are at the opposite ends of the spectrum and hence shows us the spread of the various G^{β} values that can be attained by policies for the CMDP model concerned. Here too we observe that the performance of CAC and OffPol_CCE are comparable and the G^{β} value of this policy is midway between the performance of the policies learnt in the full information case.

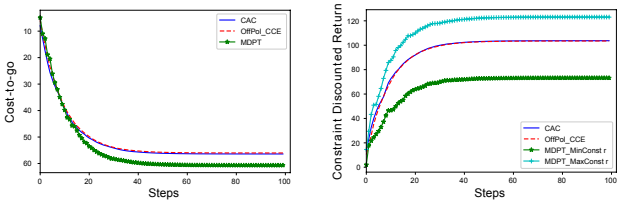
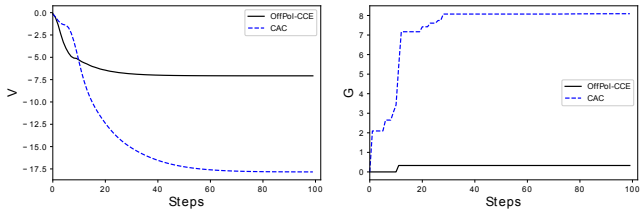


Fig. 6: Comparison of (a) Cost-to-go and (b) Long-term constraint of Constrained Actor Critic (CAC) and Algorithm 1 on Random MDP benchmark.

Manuscript 374 submitted to 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Received April 28, 2022.



(a) (b)

Fig. 7: Comparison of (a) Cost-to-go and (b) Long-term constraint of Constrained Actor Critic (CAC) and Algorithm 1 on vehicle straight-line following benchmark.

For the straight path navigation benchmark, we compare CAC and Algorithm 1. Fig. 7 plots V^β and G^β with β concentrated on one initial state of the vehicle. The vehicle specifications that we model are those of 'RC Car', as described in [25]. The policies learnt by both algorithms are evaluated for 400 steps, however only the plot for the first 100 steps is shown in Fig. 7. The threshold α_1 is fixed at 1 during learning. The results are averaged over 10 Monte Carlo runs. The number of learning iterations for both algorithms is 1×10^6 and the discount factor used is 0.9. The structure of R for this benchmark is such that it penalizes the drift from $y = 0$ line, while g_1 penalizes whenever the steering angle and y-coordinate distance from $y = 0$ are not favourable. It is observed that for the same number of learning iterations, Algorithm 1 learns better safe policies compared to CAC. Such policies attain high V^β and low G^β. The above experiments suggest that our proposed algorithm Off-Policy Constrained Safe RL (OffPol_CCE), adapted from cross-entropy algorithm is well suited for safety-constrained systems. Additionally, by utilizing neural network architectures as described here, the algorithm can be scaled to large and complex autonomous systems.

V. Conclusions

In this work we proposed a safe RL algorithm utilizing Cross Entropy Method. Our algorithm handles constraints effectively and enables off-policy learning. Unlike previous RL algorithms our method is sample efficient owing to the off-policy value prediction. Further, experimental results demonstrate the practical applicability of our algorithm. It would be interesting to develop off-policy RL algorithms for consrained risk sensitive MDPs setting for enhancing safety while deploying RL algorithms.

References

[1] K. J. Prabuchandran, H. Kumar, and S. Bhatnagar, "Multi-agent Reinforcement Learning for Traffic Signal Control," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014.

[2] L. Waymo, "On the road to fully self-driving," *Waymo Safety Report*, pp. 1–43, 2017.

[3] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 2nd ed. New York, NY, USA: John Wiley & Sons, Inc., 2005.

[4] D. Bertsekas, *Dynamic Programming and Optimal Control*, 4th ed. Belmont,MA: Athena Scientific, 2013, vol. II.

[5] K. Malialis, S. Devlin, and D. Kudenko, "Resource abstraction for reinforcement learning in multiagent congestion problems," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, ser. AAMAS '16, 2016, p. 503–511.

[6] S. Padakandla, K. J. Prabuchandran, and S. Bhatnagar, "Energy Sharing for Multiple Sensor Nodes with Finite Buffers," *IEEE Transactions on Communications*, vol. 63, no. 5, pp. 1811–1823, 2015.

[7] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, "Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 610–617, April 2019.

[8] P. Kohli and A. Chadha, "Enabling Pedestrian Safety Using Computer Vision Techniques: A Case Study of the 2018 Uber Inc. Self-driving Car Crash," in *Advances in Information and Communication*, K. Arai and R. Bhatia, Eds. Cham: Springer International Publishing, 2020, pp. 261–279.

[9] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete Problems in AI Safety," *arXiv*, 2016.

[10] E. Altman, "Constrained markov decision processes with total cost criteria: Lagrangian approach and dual linear program," *Mathematical Methods of Operations Research*, vol. 48, no. 3, pp. 387–417, 1998.

[11] Y. Chow and M. Ghavamzadeh, "Algorithms for CVaR Optimization in MDPs," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. MIT Press, 2014, p. 3509–3517.

[12] V. Borkar and R. Jain, "Risk-Constrained Markov Decision Processes," *IEEE Transactions on Automatic Control*, vol. 59, no. 9, pp. 2574–2579, Sep. 2014.

[13] A. Wachi, Y. Sui, Y. Yue, and M. Ono, "Safe exploration and optimization of constrained mdps using gaussian processes," in *AAAI*, 2018.

[14] C. E. Rasmussen, "Gaussian Processes in Machine Learning," *Lecture Notes in Computer Science*, p. 63–71, 2004.

[15] J. García and F. Fernández, "A Comprehensive Survey on Safe Reinforcement Learning," *Journal of Machine Learning Research*, vol. 16, no. 42, pp. 1437–1480, 2015.

[16] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained Policy Optimization," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17, 2017, p. 22–31.

[17] M. Wen and U. Topcu, "Constrained Cross-Entropy Method for Safe Reinforcement Learning," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 7450–7460.

[18] S. Mannor, R. Rubinstein, and Y. Gat, "The Cross Entropy Method for Fast Policy Search," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ser. ICML'03. AAAI Press, 2003, p. 512–519.

[19] S. Bhatnagar, "An actor–critic algorithm with function approximation for discounted cost constrained Markov decision processes," *Systems & Control Letters*, vol. 59, no. 12, pp. 760 – 766, 2010.

[20] R. B. Diddigi, D. S. K. Reddy, P. K.J., and S. Bhatnagar, "Actor-Critic Algorithms for Constrained Multi-Agent Reinforcement Learning," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '19. International Foundation for Autonomous Agents and Multiagent Systems, 2019, p. 1931–1933.

[21] P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A Tutorial on the Cross-Entropy Method," *Annals of Operations Research*, vol. 134, no. 1, p. 19–67, Feb 2005.

[22] A. G. Joseph and S. Bhatnagar, "An incremental off-policy search in a model-free Markov decision process using a single sample path," *Machine Learning*, vol. 107, no. 6, p. 969–1011, Feb 2018.

[23] R. S. Sutton, A. R. Mahmood, and M. White, "An Emphatic Approach to the Problem of Off-Policy Temporal-Difference Learning," *Journal of Machine Learning Research*, vol. 17, no. 1, p. 2603–2631, Jan 2016.

[24] R. B. Diddigi, C. Kamanchi, and S. Bhatnagar, "A convergent off-policy temporal difference algorithm," pp. 1103–1110, 2020.

[25] E. Ahn, "Towards safe reinforcement learning in the real world," Master's thesis, Pittsburgh, PA, July 2019.

[26] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.