Constraint Sampling Reinforcement Learning: Incorporating Expertise For Faster Learning

Tong Mu^1, Georgios Theocharous ^{2}, David Arbour^2, Emma Brunskill ^{1}

^{1} Stanford University, {tongm, ebrun} @ cs.stanford.edu
^{2} Adobe Research, {theochar, arbour} @ adobe.com

Abstract

Online reinforcement learning (RL) algorithms are often difficult to deploy in complex human-facing applications as they may learn slowly and have poor early performance. To address this, we introduce a practical algorithm for incorporating human insight to speed learning. Our algorithm, Constraint Sampling Reinforcement Learning (CSRL), incorporates prior domain knowledge as constraints/restrictions on the RL policy. It takes in multiple potential policy constraints to maintain robustness to misspecification of individual constraints while leveraging helpful ones to learn quickly. Given a base RL learning algorithm (ex. UCRL, DQN, Rainbow) we propose an upper confidence with elimination scheme that leverages the relationship between the constraints, and their observed performance, to adaptively switch among them. We instantiate our algorithm with DQN-type algorithms and UCRL as base algorithms, and evaluate our algorithm in four environments, including three simulators based on real data: recommendations, educational activity sequencing, and HIV treatment sequencing. In all cases, CSRL learns a good policy faster than baselines.

1 Introduction

Online reinforcement Learning (RL) algorithms have the large potential for improving real world systems with sequential decisions such as recommendation systems(Theocharous et al. 2020) or intelligent tutoring systems (Bassen et al. 2020). Such domains often have large or infinite state spaces, and existing RL methods that can scale to these settings frequently require a prohibitively large amount of interaction data to learn a good policy. Incorporating human expert knowledge can accelerate learning, such as through expert demonstrations (Wu et al. 2019; Arora and Doshi 2021; Hussein et al. 2017; Taylor, Suay, and Chernova 2011) or having them provide properties the optimal policy is guaranteed to satisfy, such as a specific function class constraint (Ijspeert, Nakanishi, and Schaal 2002; Tamosiunaite et al. 2011; Buchli et al. 2011; Kober, Bagnell, and Peters 2013). However, such approaches also run the risk of reducing the system performance when the provided information is misleading or suboptimal. For example, in recommendation systems, prior Human Computer Interaction (HCI) literature has found that diversity of recommendations across time is important (Nilashi et al. 2016; Bradley and Smyth 2001; Komiak and Benbasat 2006; Chen et al. 2019). But confidently concluding this finding will generalize to a new system, and translating this high level knowledge into a concrete constraint for the policy class is subtle (does 'diversity' mean 3 or 4 different item categories? Within the last 5 or the last 10 recommendations?). Choosing the wrong formulation can significantly impact the resulting reward.

An alternate approach is to allow a human domain expert to provide many different formulations, and use model or algorithm selection to automatically learn to choose the most effective (e.g. (Lee et al. 2021; Laroche and Feraud 2018). However, much of the work on selection focuses on theoretical results and cannot be used with popular deep RL algorithms that may be of more practical use (Lee et al. 2021). Perhaps most closely related to our work is the work of Laroche and Feraud (2018), which uses an upper confidence bound bandit approach to learn to select among a set of reinforcement learners. This work was constructed for optimizing across learning hyperparameters (such as learning rate or model size) and, like many upper confidence bound approaches, relies on tuning the optimism parameter used, which is often hard to do in advance of deployment.

In this work we instead focus on leveraging human insight over the domain to speed RL through weak labels on the policy. We propose an algorithm, Constraint Sampling Reinforcement Learning (CSRL) which performs adaptive selection and elimination over a set of different policy constraints. Our selection algorithm optionally use these constraints to learn quickly, and to distinguish this from the safety constraints used in safe RL, we will also refer to them as *policy restrictions*. A policy constraint or restriction limits the available actions in a state and can speed learning by potentially reducing the exploration of suboptimal actions. Our method performs algorithm selection over a set of RL learners, each using a different policy restriction. For example, in a recommendation system, one restriction might specify that 3 items should be from unique categories in the past 5 items shown, and another could require at least 2 unique items in the past 10, but only for users who have used the system for more than 6 months. A third RL learner could use the unrestricted policy class, which does not limit the available actions in any state. Note that other areas of machine learning have significant reductions in cost by allowing people to provide weak supervision through labeling functions that may be imperfect(e.g. (Ratner et al. 2017)). At a high level, we apply a similar idea here for reinforcement learning, allowing people to provide weak, potentially imperfect policy restrictions to be used by different RL learners. Our algorithm then performs adaptive selection over the set using optimism under uncertainty over the potential performance of the RL learners, each of which is operating with a different set of restrictions. A second technical innovation comes from noting that tuning the optimism parameter of the selection mechanism in advance can be infeasible, and a parameter either too high or too low can slow learning. Instead, we introduce a simple heuristic which uses the amount of model change to estimate the convergence of each RL learner and use this to eliminate RL learners with low performance. This allows us to achieve good performance much faster than through optimistic adaptive selection alone.

These simple ideas lead to substantial empirical improvements in the diverse range of settings we consider, which include simulators created with real data in movie recommendations, tutoring system problem selection, and HIV treatment, as well as the Atari lunar lander simulator. We conduct a careful experimental analysis to illustrate the benefits of our additional change-based learner elimination and the robustness to the inclusion of poor constraints. Our approach is simple and can be used with a wide variety of base reinforcement learners, and may make RL more feasible for a wider set of important domains.

2 Setting

A Markov Decision Process (MDP) (Bellman 1957) is described as a tuple $(\mathcal{S}, \mathcal{A}, P, R)$ where $\mathcal{S}$ is the set of states and $\mathcal{A}$ is the action set. The transition dynamics, $P(s'|s,a)$ defines the probability distribution of next states s' after taking action a in state s, and $R(r|s,a)$ defines the distribution of rewards r. We assume the action space is discrete (however the state space can be either discrete or continuous) and rewards are bounded $|r| < R_{max}$. We consider the episodic, finite horizon setting where the length of each episode is less than or equal to the maximum horizon length H. A policy, denoted as π is a potentially stochastic mapping from states to actions, where $\pi(a|s)$ defines the probability of taking action a in state s. A trajectory, $\tau := (s_0, a_0, r_0, s_1, a_1, r_1, ...)$, is defined as the sequence of states, actions, and rewards in an episode. The state-action value of a policy, $Q^{\pi}_{\gamma}(s,a)$, is the expected discounted sum of rewards of starting in a state s, taking action a and then following the policy: $Q^{\pi}_{\gamma}(s,a) := \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{h} \gamma^t r_t | s_0 = s, a_0 = a])$, where $\gamma \in [0,1]$ is the discount factor. The value function is the expected discounted sum of rewards obtained starting in state s and following policy π: $V^{\pi}_{\gamma}(s) := \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{h} \gamma^t r_t | s_0 = s]$ The optimal policy, denoted π^*, is the policy that maximizes V: $\pi^* = \arg\max_{\pi} V^{\pi}_{\gamma}$.

3 Algorithm

We first briefly present the aim and overview of our algorithm, Constraint Sampling Reinforcement Learning (CSRL), before going into detail. The goal of our work is to provide a simple method for leveraging (potentially weak) human domain knowledge for faster learning without sacrificing final performance. CSRL takes as input a set of different candidate policy *constraints* (we also refer to as *restrictions* to distinguish our goal from that of the Safe RL work which uses constraints), each of which is used to define a RL learner that must follow the restriction while learning. Some or all of the provided restrictions may disallow the (unknown) optimal policy and the unrestricted option may be included as a candidate. CSRL then uses an upper confidence bandit (UCB) (Auer 2002) method that considers the prior performance of the learners to adaptively select the next learner to use to make decisions for the episode. In practice such optimism based approaches often require careful tuning to achieve good performance which can be infeasible in new domains lacking prior data. To increase robustness to this hyperparameter and speed learning, our CSRL method introduces a simple heuristic that tracks the model parameter changes in the RL learner for hypothesizing when a learner has converged, and eliminates those that may be misspecified or have low performance.

We now introduce some notation. In this paper we use the following definition of constraint/restriction:

Definition 3.1 (Constraint/Restriction) *A constraint (or restriction) C is a function that maps each state to a set of allowable actions, $C(s) = \{a_i, a_k, \dots\}$.*[1]

Given a set of K restrictions $\mathcal{C}$, let C_k denote the k^{th} restriction. We say a policy π *satisfies* a restriction C_k if for every state, π only takes actions allowed by C_k: $\forall(s,a)$ $\pi(a|s) > 0$ only if $a \in C_k(s)$.

Definition 3.2 (Restricted Policy Set) *We denote the policy set of restriction C_k as Π_k and define it as the set of all policies π that satisfy C_k:*

$$\Pi_k = \{\pi : \forall(s,a)\pi(a|s) > 0 \to a \in C_k(s)\}$$

Definition 3.3 (Restricted RL Learner) *Given a restriction C_k, and a base RL learning algorithm that can learn under constraints (such as DQN) we instantiate a **restricted reinforcement learner**, denoted l_k. l_k is restricted to executing and optimizing over policies in Π_k*

We assume each restriction in the set is unique and we define the subset property between restrictions:

Definition 3.4 (Subset/More Restricted) *Restriction C_k is a subset of restriction C_j if every action allowed in C_k is also allowed in C_j: $\forall s$ $C_k(s) \in C_j(s)$. In this case, we will also refer to C_k as more restricted than C_j and define the $<$ operator: $C_k < C_j$. We also apply this notation to describe the corresponding policy sets and RL learners.*

Note on specifying restrictions: Note that while policy restrictions are defined by the state-action pairs allowed, they often do not need to be specified by humans at that level of granularity. For example, a human expert might specify that students should only be given practice problems involving at most one new skill. As long as the state and action

[1]Note this in is contrast to the Constrained MDP (CMDP) framework (Altman 1999), which has a different focus on costs and budgets.

space has features representing the skills involved, it is easy to programmatically translate this high level specification to the desired constraint without requiring the human expert to enumerate all state-action pairs.

Our algorithm, CSRL (Algorithm 1), takes as input a base RL algorithm, $\mathcal{A}lg$, a set of potential restrictions $\mathcal{C}$ and a confidence bound function $B(h, n)$. CSRL starts by initializing $|\mathcal{C}|$ RL learners, each which use the input base algorithm $\mathcal{A}lg$, along with one of the restrictions $C_i \in \mathcal{C}$. Learner l_i will only chose actions and learn over policies that satisfy C_i. Let $\mathcal{L}$ denote the set of active learners and initially set $\mathcal{L} = \{1, \ldots, |\mathcal{C}|\}$. Each episode proceeds in 3 steps. First CSRL chooses a RL learner l_i in the active set $\mathcal{L}$ to select actions in this episode. Second, CSRL gathers a trajectory of states, actions, and rewards using l_i. This data is used both to update the learner l_i as well as all the relevant other learners: sharing data speeds learning across all the RL learners. Third, a heuristic is used to potentially eliminate l_i from $\mathcal{L}$. The next episode then starts.

We now describe step 1, RL learner selection. We use UCB to select the RL learner with the maximum estimated upper bound on the potential return in the active set. The UCB mechanism uses the average prior returns observed during past executions of RL learner l, denoted $\hat{\mu}_l$, and the input confidence bound function.

$$k = \arg\max_{l \in \mathcal{L}} (\hat{\mu}_l + B(h, n_l)) \tag{1}$$

There are two important things to note about this learner selection strategy. First, CSRL does not use any RL learners' internal estimates of their own performance or potential value function. This allows us to leverage very general base algorithms Alg without requiring that they accurately estimate their own value functions. Instead CSRL relies only on the observed returns from executing each learner, treating them as one arm in a multi-armed bandit. Second, note that the estimated upper bound for a given learner l in Equation 1 will generally not be a true upper confidence bound on the performance of the RL algorithm. This is because the UCB multi-armed bandit algorithm assumes that the stochastic returns of individual arms are unknown, but stationary. In contrast, in our setting, arms are RL learners whose policies are actively changing. Fortunately, prior related work has successfully used UCB to select across arms with non-stationary returns in an effective way: this is the basis of the impactful upper confidence trees, a Monte Carlo Tree Search method that prioritizes action expansions in the tree according to upper bounds on the non-stationary returns of downstream action decisions (Shah, Xie, and Xu 2020). It has also been used in related work on online RL (Laroche and Feraud 2018), and we will demonstrate it can both be empirically effective in our setting, and, under mild assumptions, still guarantee good asymptotic performance.

After gathering data using the selected RL learner, this data is provided to all RL learners to optionally update their algorithm[2]. Then the third step is to eliminate the potential learner associated with the chosen constraint. The elimination heuristic checks if a RL learner l_i's value or state-action value has stabilized, and if its average empirical performance is lower than another RL learner l_j, we will eliminate l_i if a less constrained learner l_k is in the active set. We now state this more formally.

When RL learner l_k is used for the n^{th} time, and generates trajectory τ_n, let δ_k^n represent the change in the value function. For example, in tabular domains, we can measure the average absolute difference in the state-action values

$$\delta_k^n = \frac{||Q_k^{n-1} - Q_k^n||_1}{|\mathcal{S}||\mathcal{A}|}, \tag{2}$$

where Q_k^n represents the state-action value function after updating using the data just gathered. In value-based deep RL we can use the loss function over the observed trajectory as an approximate estimate of the amount change[3]

$$\delta_k^n = \sum_{\tau^n} (r_t + \max_{a \in C_k(s_{t+1})} Q_k^n(s_{t+1}, a)) - Q_k^n(s_t, a_t). \tag{3}$$

For a RL learner to be potentially eliminated, the change, δ_k^n, must be below a threshold T_l for at least T_n successive steps, suggesting that the underlying learning process is likely to have stabilized. If this condition is satisfied for a RL learner l_i the meta-leaner first checks there exists a less constrained learner l_k in $\mathcal{L}$. If such a l_k exists and at least one other learner l_j in $\mathcal{L}$ has higher average performance: $\hat{\mu}_i < \hat{\mu}_j$, then l_i is removed. See Algorithms 1 and 2 for pseudocode. We give examples of instantiations of CSRL with base learners of UCRL and DDQN in section 3 below and we will release all our code on Github.

Elimination Mechanism Intuition: Recall that CSRL should try to select the most restricted learner that is compatible with the optimal policy, since learning with a reduced action space will often speed learning and performance. To increase robustness, the RL learner selection UCB strategy relies on minimal information about the internal algorithms. However, UCB often can be conservative in how quickly it primarily selects high performing arms (in our case, RL learners) rather than lower reward arms (in our case, learners with restrictions incompatible with the optimal policy). Consider if one can identify when learner l_i has converged and can correctly evaluate its performance through the running average μ_i. If there exists l_j with higher average returns μ_j, restriction C_i is likely to not include the optimal policy and l_i can be removed from the set of potential RL learners. Our method uses a proxy heuristic for convergence and sample estimates for returns which can be noisy. Therefore it may incorrectly conclude a learner l_i has converged and/or has suboptimal returns. To ensure that we preserve a learner that admits the optimal policy, we only eliminate a learner l_i, and its constraint C_i, if it is a subset of at least one other

[2]A reader may wonder if providing such off policy data is always useful or how best to incorporate it. In the particular base RL algorithms we use, it is straightforward to incorporate the observed trajectories into experience replay or in an estimate of dynamics and reward models, but more sophisticated importance sampling methods could be used for policy gradient based methods.

[3]Note one could use other measures of the change in the RL learner, including differences in the neural network parameters, or changes in predictions of the value of states.

```
Algorithm 1: CSRL
Inputs: $Alg, \mathcal{C}, \mathcal{Z}, B(h,n), T_l, T_n$
Initialize: $\mathcal{L} \leftarrow$ Create Restricted Learners from $\mathcal{C}$
$D_k = [\,]\ \forall C_k \in \mathcal{C}$    // Model Change Amounts
for Episode $h = 1, 2, ...$ do
    $l_k \leftarrow$ select RL learner [Eqn. 1]
    $\tau_h \leftarrow$ GenerateTrajectory ($l_k$)
    $R_h = \sum_{t=1}^{len(\tau_h)} r_t$
    $n_k = n_k + 1$
    $\hat{\mu}_k = \frac{(n_k - 1)\hat{\mu}_k + R_h}{n_k}$
    UpdateLearners ($\mathcal{L}, \tau_h$)
    $\delta_h \leftarrow$ Calculate Change of $l_k$ [ eq. 2 or 3]
    $D_k \leftarrow [D_k, \delta_h]$
    if Eliminate ($l_k, D_k$) then
        $\mathcal{L} \leftarrow \mathcal{L} \setminus l_k$
```

```
Algorithm 2: Eliminate (Eliminate Learner)
Inputs: $l_k, D_k$
if $\exists n \in \{n_k - T_n : n_k\}$ such that $D_k(n) > T_l$ then
    Return False (Don't Eliminate)
if Exists $l_i, l_j \in \mathcal{L} : C_i > C_k$ and $\hat{\mu}_j > \hat{\mu}_k$ then
    Return True (Eliminate)
Return False
```

active constraint C_j. Therefore the the set of policies that satisfy C_i will continue to exist in the active set, even when C_i is eliminated. We now provide some basic soundness to this proposed approach, before describing instantiations of CSRL with particular base learners, and demonstrating its performance empirically.

Brief Theoretical Discussion

We briefly provide a guarantee that at least one learner whose policy set contains the optimal policy will be taken more than all other learners under mild assumptions.

Assumption 1 (Model Parameter Convergence) *Let $M_{k,n}$ represent the model parameters of the learner l_k after the n^{th} update. In every run of CSRL, the model parameters of every learner converge:* $\lim_{n\to\infty} \mathcal{M}_{k,n} \to \mathcal{M}_k$ *for all* $C_k \in \mathcal{C}$.

Let $\pi_{\mathcal{M}_k}$ denote the policy corresponding to the converged model parameters $\mathcal{M}_k$ of RL learner l_k. Let $\mathbb{E}_{s_0}[V^{\pi_{\mathcal{M}_k}}(s_0)] = \mu_k$, $\mu^* = \max_k \mu_k$ and π^* denote the policy that achieves μ^*. Note that π^* is defined as the policy with the highest return across all learners in the set. Without loss of generality assume π^* is unique. We refer to the set of constraints compatible with π^* as the set of optimal constraints and denote this set as $\mathcal{C}^*$ with a corresponding indices set $\mathcal{K}^*$. Then require:

Assumption 2 (Convergence to Optimal) *Given a specific run of CSRL, let every learner in $\mathcal{K}^*$ converge to the μ^* of the run: $\mu_k = \mu^*$ for all $k \in \mathcal{K}^*$*

We now show that at least one RL learner in $\mathcal{C}^*$ will be chosen more than all suboptimal learners asymptotically.

Theorem 1 *Assume Assumptions 1 and 2 hold. Also assume as input a confidence bound $B(h,n)$ of the form $\frac{z(h)}{n^\eta}$ with $0 < \eta < \frac{1}{2}$ and $z(h)$ satisfying the following two conditions: (i) $z(h)$ is non-decreasing and (ii) $O(z(h)^{1/\eta}) < O(h)$. Let $T_k(h)$ be the number of times RL learner l_k has been selected at episode h. Then for at least one $k^* \in \mathcal{K}^*$, $T_{k^*}(h) > T_k(h)$ for all $k \notin \mathcal{K}^*$ as $h \to \infty$*

The proof of theorem 1 is provided in the appendix.

When the base algorithm has convergence guarantees, such as UCRL, we can additionally provide guarantees on the rate of convergence. We provide these rates and a discussion of the UCRL case in the appendix: our analysis drawns upon the analysis of convergence rates for Monte Carlo Tree Search from Shah et al. (Shah, Xie, and Xu 2020).

Algorithm Instantiations

We discuss specific instantiations of CSRL with various base RL algorithms.

CSRL-UCRL. UCRL (Auer, Jaksch, and Ortner 2009) is an influential strategic RL algorithm with good regret bounds this is based on optimism under uncertainty for tabular settings. It is simple to incorporate action constraints during the value iteration steps:

$$V_h^{t+1}(s) \leftarrow \max_{a \in C_h(s)} (\tilde{R}(s,a) + \sum_{s'} \tilde{P}(s'|s,a)\gamma V_h^t(s'))$$

All observed (s, a, r) tuples are used to update an estimated model of transitions and rewards that is shared across all RL learners. Equation 1 is used to track convergence in the estimated value function.

CSRL-DQN, CSRL-DDQN and CSRL-Rainbow Deep reinforcement learning has shown powerful results across a wide variety of complex domains. Our CSRL-DQN, CSRL-DDQN, and CSRL-Rainbow implementation uses a separate DQN (Mnih et al. 2015), DDQN (van Hasselt, Guez, and Silver 2016), or Rainbow (Hessel et al. 2018) learner for each restriction[4]. We used epsilon greedy exploration with epsilon decay. Experience is shared across learners in the form of a shared replay buffer. The `UpdateLearners` function places the tuples from the most recent trajectory in the shared replay buffer. Each learner l_k is then updated, using only samples from the buffer that satisfy the associated restriction C_k. The learner l_k is updated using a constrained Q loss (Kalweit et al. 2020) (see Equation 3).

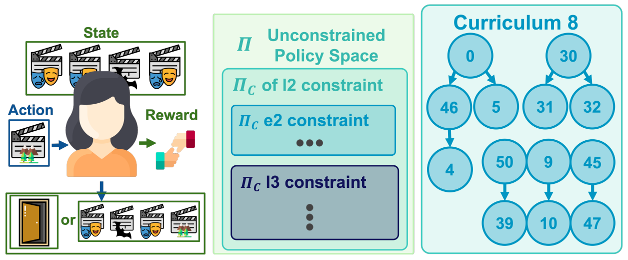
4 Experiments

We briefly introduce the evaluation environments and the constraints used and then discuss our results. Due to space constraints, we defer detailed descriptions of the environments and constraint constructions to the appendix.

Environments and Constraints

Recommendation System Environment The movie recommendations (See Fig 1a for illustration) environment

[4]We explored sharing some model weights but found that resulted in worse performance



(a) (b) (c)

Figure 1: (a) An image of the recommendation system environment (RS env.) (b) A visualization of the policy space for the RS env. For example, every policy allowed under the *exactly 2 variability* constraint ('e2') is also allowed under the *at least 2 variability* ('l2') constraint. (c) An example curriculum graph in the education domain. Each directed edge indicates the source node is a prerequisite of the sink node.

is a slighlty modified environment from prior work (Warlop, Lazaric, and Mary 2018) fit with the Movielens 100K dataset (Harper and Konstan 2015) which contains 100K user-movie ratings across 1000 users with $|\mathcal{A}| = 5$ and $|\mathcal{S}| = 625$. Each action is a movie genre and the state space encodes recently seen movies. The rewards correspond to the ratings of the recommended movies. The episode length is random and each state-action pair has some probability of leading to the terminal state.

Following prior work that suggests diversity is correlated with system usage (Nilashi et al. 2016), we design a set of 12 constraints using a *variability* factor, which we define as the number of unique actions in recent history. Our constraints require the policy to maintain a certain level of variability. A partial visualization of the structure between some constraints in the set is given in Figure 1b. Because our state space contains the history of past actions, these high level constraint specification are easily translated programmatically.

Education: Deep Knowledge Tracing Student. Our new educational activities sequencing environment uses a popular student learning model, the Deep Knowledge Tracing (DKT) model (Piech et al. 2015), to simulate students. The model is trained with the ASSISTment 2009-2010 (Feng, Heffernan, and Koedinger 2009) dataset of algebra learning, containing data from 3274 students over 407K student-problem interactions. Each action ($|\mathcal{A}| = 51$) corresponds to presenting a different problem to the student. The horizon is length $H = 100$. The state space is a continuous $\mathbb{R}^{58}$ and encodes the current proficiency (the predicted probability of mastery by the DKT model) on each problem and the binary encoded timestep. The reward corresponds to a signal of mastery and is 1 when the proficiency of a problem first exceeds a threshold $m_t = 0.85$ and 0 otherwise.

In education, curriculum or prerequisite graphs are common; however, setting the correct granularity to model knowledge can be difficult. We create a constraint set consisting of different graphs: Figure 1c shows an example using automatic curriculum generation methods that requires hand specifying a hyperparameter to adjust the number of edges (Piech et al. 2015). We construct a constraint set containing 13 different graphs. Given a graph, we only allow unmastered problems that have all prerequisites mastered (this information is encoded in the state space) to be selected.

HIV treatment Simulator The HIV treatment simulator (Adams et al. 2004; Ernst et al. 2006) simulates a patient's response to different types of treatments. The action space is discrete with size 4 and represents various treatment actions. The state space is continuous $\mathbb{R}^6$, where each dimension represents a marker of the patient's health. Each episode is 200 timesteps and the reward encourages patients to transition to and maintain a healthy state while penalizing drug related actions.

We created a simulator for multiple heterogeneous patient groups by perturbing the internal hidden parameters of the system following Killian, Konidaris, and Doshi-Velez (2017). We learn an optimal decision policy for 3 different groups, and then use the known optimal policies as constraints to learn in a new group which may or may not be similar to a group with a known policy. We create a constraint set of 7 constraints.

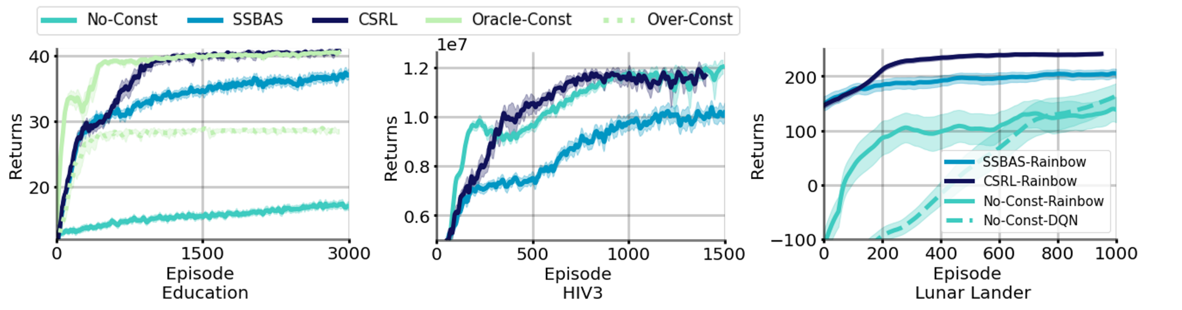
Lunar Lander The Lunar Lander environment from Open AI Gym (Brockman et al. 2016) simulates landing an aircraft. The action space is discrete with 4 actions which correspond to firing different actuators. The state space is continuous $\mathbb{R}^8$ and gives position and velocity information. Positive reward is given for safely landing and negative reward is given for firing the actuators and crashing.

We generate different policies with differing performance levels to mimic access to policy information from multiple human pilots. We create a constraint set of 10 constraints each of which limits the available action to that of the policy of one of the past policies or a mixture of them.

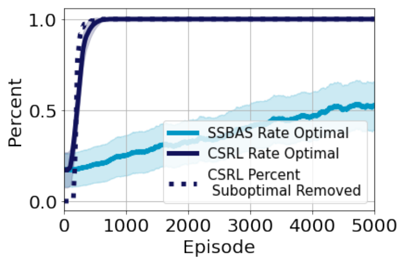
Results and Discussion

We compare CSRL against 4 baselines: unrestricted reinforcement learning, reinforcement learning with the *oracle-constraint* which is the best constraint (but unknown in real situations), reinforcement learning under a non-oracle constraint, and SSBAS (Laroche and Feraud 2018), a prior algorithm for selecting across learners with different hyperparameters or models that uses a UCB selection approach but does not eliminate learners.

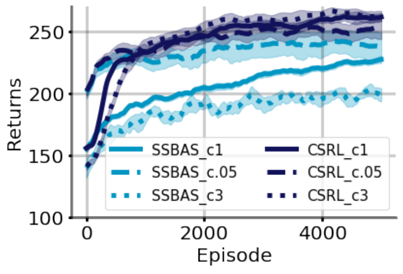
We used UCRL as a base learner for our Recommendation System experiments, DDQN for Education and HIV Treatment, and Rainbow for Lunar Lander. In each experiment, the same base learning model parameters (model architecture, learning rates, exploration rate scheduling, etc) were used for all algorithms: see appendix for details. For CSRL and SSBAS, total episode return was scaled to the range $[0, 1]$ for the adaptive learner selection and we used the confidence bound $B(h, s) = c\frac{\sqrt{\log(t)}}{s^{1/2}}$. This bound satisfies the conditions of $B(h, s)$ required for Theorem 1 to hold. We did not tune c and used $c = 1$ for both CSRL and SSBAS for all experiments. We later present a discussion of sensitivity to different values of c. For CSRL, T_n was set to 20 and T_l was set to 0.05 for all experiments. We did not tune



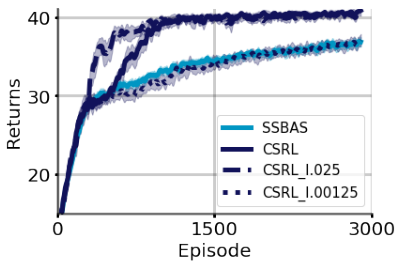
(a) Training Returns



(b) Lunar: Rate of optimal



(c) Lunar: c sweep



(d) Education: T_l sweep

Figure 2: We plot all values with 95% confidence intervals. **Top Row** We plot episode returns during training from the Education, Lunar Lander and HIV domains. We plot our algorithm CSRL, as well as the SSBAS (Laroche and Feraud 2018) and unconstrained (labelled No-Const) baselines. In the educational domain, a misspecified constraint, labelled Over-Const, is also shown. The Oracle-Const plots the performance of following the oracle best constraint in the set which is not known beforehand. In the **bottom row**, we provide further experiments and visualizations used to illustrate various properties of our algorithm for our discussion (Section 4). Additional plots including parameter sweeps in other environments and plots over longer episodes are provided in the Appendix.

	90% Of Max		97% Of Max	
	SSBAS	Unconst	SSBAS	Unconst
Edu	3.0 ± 0.3	20+	4.54 ± 0.6	20+
Mov.	1.0 ± 0.07	1.5 ± 0.1	1.7 ± 0.3	2.1 ± 0.2
HIV	27 ± 0.8	5.5 ± 2.4	20+	15.3 ± 0.7
LL	5.2 ± 0.6	2.7 ± 0.5	3.4 ± 0.7	1.6 ± 0.3

Table 1: The sample complexity speedup of CSRL over baselines in terms of the factor of episodes more required by baselines to achieve returns 90% and 97% of the maximum observed value compared to CSRL.

either parameter and we later present a discussion of robustness to different values of T_l. For each experiment, the results were averaged over multiple random seeds, with 200, 20, 20, and 40 seeds for the recommendations, education, HIV treatment, and Lunar Lander experiments respectively.

Results Across all domains CSRL learned a good policy faster than non-oracle baselines, and often significantly faster. In Table 1 we list the speedup of CSRL over the SSBAS and Unrestrained (Unconst) baselines in terms of the factor of episodes more compared to CSRL needed by the baselines to achieve mean returns 90% and 97% of the observed maximum return. In most cases, we observe CSRL can achieve a high level of performance significantly faster than baselines, often at least 1.5 times as fast and occasionally much more. In the appendix we give a table of the raw number of episodes needed to reach these performances for each environment. In Figure 2a we plot the returns through training along with 95% confidence intervals for the Education, HIV treatment and Lunar Lander domains. These plots for the other domains are presented in the appendix. We observe CSRL is able to utilize the constraints to avoid the extended initial poor performance compared to the unconstrained condition. Additionally the elimination mechanism allows the algorithm to eliminate suboptimal learners to quickly achieve near-oracle performance. We investigate adaptive selection and elimination in depth in the discussion:

Discussion

Importance of Adaptive Selection It is expected that following a single good constraint can lead to quick learning by reducing the policy space that needs to be explored. We see this is indeed true as the oracle-constraint performs best for most cases. However the best constraint is not known beforehand, and following a single constraint that is misspecified can lead to suboptimal performance. We demonstrate this in the education domain, shown in Figure 2a where we additionally plot the tightest restriction in the set, labeled *Over-Const* (which stands for Over-Constrained). Over-Const also quickly converges but to a return much lower than optimal. We see the adaptive selection mechanism of CSRL and SS-

BAS over the set can leverage the constraints to learn faster than the unrestricted baseline while avoiding this potential misspecification. Additionally the benefit of adaptive selection has over standard unrestricted RL increases with action space size as larger action spaces are harder to explore. This is illustrated by comparing the speed of learning unrestrained in our Education ($|\mathcal{A}| = 51$) and Lunar Lander ($|\mathcal{A}| = 4$) environments (Fig 2a).

Importance of Elimination In our setting where we expect some constraints to be misspecified, we found eliminating suboptimal learners to be very important for robustness against performance decreases due to missspecification. This is illustrated in the HIV experiments shown in Figure 2a. In this case, all constraints are suboptimal and the unconstrained option is optimal. We notice that CSRL is able to quickly use elimination to approach the unconstrained performance compared to SSBAS. In Figure 2b we plot the rate of selecting the optimal constraint in the lunar lander experiment for CSRL and SSBAS through learning. We see that elimination allows the algorithm to achieve high rates of selecting the optimal policy much faster.

When is elimination not important? We expect elimination to not be important when the performance gap between the optimal and suboptimal constraints is large. Intuitively a large difference is easier for the UCB mechanism to distinguish the best item so CSRL and the SSBAS baseline learn quickly to choose the best item and achieve nearly identical performance. We demonstrate a case of this in the appendix.

The confidence bound parameter In Figure 2c, we plot performance for various values of c, the multiplier on the confidence bound of the UCB constraint selection mechanism in the lunar lander environment. For both algorithms, we see that a low value, $c = 0.05$, results in higher initial performance but a much slower rate of learning which is a poor outcome for both algorithms. On the other hand, a higher value, $c = 3$ leads to significantly worse performance for SSBAS while CSRL's elimination mechanism protects the performance from suffering. The uncertainty over the value of the multiplier naturally comes in when the maximum reward is unknown. It is undesirable to underestimate the maximum reward as it may lead to a slow rate of learning. In these cases CSRL along with a high estimate of maximum reward can lead to good performance.

The effect of the loss threshold T_l: In Figure 2d we plot the performance for various values of T_l, the model change threshold for the elimination mechanism, in the education environment. We demonstrate that even for T_l set to large values ($T_l = 0.25$, 5 times greater than initial used value), the performance does not decrease (in fact it increases due to faster elimination). This shows our robust elimination procedure is able to maintain performance when the algorithm incorrectly hypothesizes a constraint as suboptimal. When T_l is set to low values ($T_l = 0.00125$, 40 times less than initial), the elimination of constraints slow and we approach the performance of the SSBAS baseline.

Summary Overall when there is a good constraint in the set, we demonstrate our algorithm is (1) able to achieve a good policy quickly, often significantly faster than baselines (2) this performance improvement is due to CSRL's robustness against both misspecified constraints and hyperparameters such as the confidence bound parameter.

5 Related Work

We discuss some additional areas of related work not previously mentioned.

Constrained RL Our work is related to work in constrained RL. Most prior work considers learning under a single constraint that is always enforced. Constrained RL has been studied under various combinations of what is known and unknown about the components of the model (the constraints, rewards, and transition dynamics). It has been studied in cases where all components are known (Altman 1999), as well as cases where one or more of them need to be learned online (Efroni, Mannor, and Pirotta 2020; Zheng and Ratliff 2020; Achiam et al. 2017; Wachi and Sui 2020; Bhatnagar and Lakshmanan 2012). This work spans a variety of different algorithms including tabular (Efroni, Mannor, and Pirotta 2020), policy search (Achiam et al. 2017), and actor-critic (Bhatnagar and Lakshmanan 2012) RL algorithms. Contrary to this work that focuses on learning to satisfy a single constraint, we consider a set of weak constraints, which may or may not be compatible with the action selections of the unknown optimal policy. We additionally note our method is separate from the constraint sampling algorithms for solving unconstrained dynamic programming problems approximately using linear programming (De Farias and Van Roy 2004).

Inferring Constraints/Rules There has been prior work on inferring constraints or rules from demonstrations (Taylor, Suay, and Chernova 2011). In two papers (Noothigattu et al. 2019; Balakrishnan et al. 2019) a single constraint is inferred from demonstrations, and a 2-armed bandit learns if the inferred constraint should be followed. Our work differs in that we consider utilizing domain knowledge instead of demonstrations and we consider multiple potential constraints as opposed to a single constraint. We additionally differ in considering a method for RL learner elimination.

6 Conclusion

There often exists domain expertise for real world systems that can be leveraged in RL algorithms to speed learning. In this work we propose a method, CSRL, to incorporate this knowledge in the form of constraints. As it is often difficult to create a single constraint the algorithm designer is confident is correct, our work takes as input a set of potential constraints the algorithm designer hypothesizes, but does not have to be certain will speed learning. We provide a brief theoretical discussion on our upper confidence with elimination selection algorithm and focus on showing strong empirical results. We show this simple approach is compatible with deep RL methods and that CSRL can learn a good policy substantially faster than state-of-the-art baselines, suggesting its potential for increasing the range of applications with RL is feasible by leveraging imperfect human guidance.

7 Acknowledgements

This material is based upon work supported by the Stanford Human Centered AI Hoffman Yee grant and the Graduate Fellowships for STEM Diversity.

References

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained policy optimization. In *International Conference on Machine Learning*, 22–31. PMLR.

Adams, B. M.; Banks, H. T.; Kwon, H.-D.; and Tran, H. T. 2004. Dynamic multidrug therapies for HIV: Optimal and STI control approaches. *Mathematical Biosciences & Engineering*, 1(2): 223.

Altman, E. 1999. *Constrained Markov decision processes*, volume 7. CRC Press.

Arora, S.; and Doshi, P. 2021. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 103500.

Auer, P. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov): 397–422.

Auer, P.; Jaksch, T.; and Ortner, R. 2009. Near-optimal regret bounds for reinforcement learning. In *Advances in neural information processing systems*, 89–96.

Balakrishnan, A.; Bouneffouf, D.; Mattei, N.; and Rossi, F. 2019. Using multi-armed bandits to learn ethical priorities for online AI systems. *IBM Journal of Research and Development*, 63(4/5): 1–1.

Bassen, J.; Balaji, B.; Schaarschmidt, M.; Thille, C.; Painter, J.; Zimmaro, D.; Games, A.; Fast, E.; and Mitchell, J. C. 2020. Reinforcement Learning for the Adaptive Scheduling of Educational Activities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–12.

Bellman, R. 1957. A Markovian decision process. *Journal of mathematics and mechanics*, 6(5): 679–684.

Bhatnagar, S.; and Lakshmanan, K. 2012. An online actor–critic algorithm with function approximation for constrained markov decision processes. *Journal of Optimization Theory and Applications*, 153(3): 688–708.

Bradley, K.; and Smyth, B. 2001. Improving recommendation diversity. In *Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland*, volume 85, 141–152. Citeseer.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Buchli, J.; Stulp, F.; Theodorou, E.; and Schaal, S. 2011. Learning variable impedance control. *The International Journal of Robotics Research*, 30(7): 820–833.

Chen, L.; Yang, Y.; Wang, N.; Yang, K.; and Yuan, Q. 2019. How serendipity improves user satisfaction with recommendations? A large-scale user evaluation. In *The World Wide Web Conference*, 240–250.

De Farias, D. P.; and Van Roy, B. 2004. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of operations research*, 29(3): 462–478.

Efroni, Y.; Mannor, S.; and Pirotta, M. 2020. Exploration-Exploitation in Constrained MDPs. *arXiv preprint arXiv:2003.02189*.

Ernst, D.; Stan, G.-B.; Goncalves, J.; and Wehenkel, L. 2006. Clinical data based optimal STI strategies for HIV: a reinforcement learning approach. In *Proceedings of the 45th IEEE Conference on Decision and Control*, 667–672. IEEE.

Feng, M.; Heffernan, N.; and Koedinger, K. 2009. Addressing the assessment challenge with an online system that tutors as it assesses. *User modeling and user-adapted interaction*, 19(3): 243–266.

Harper, F. M.; and Konstan, J. A. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4): 1–19.

Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*.

Hussein, A.; Gaber, M. M.; Elyan, E.; and Jayne, C. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2): 1–35.

Ijspeert, A. J.; Nakanishi, J.; and Schaal, S. 2002. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, 1398–1403. IEEE.

Kalweit, G.; Huegle, M.; Werling, M.; and Boedecker, J. 2020. Deep constrained q-learning. *arXiv e-prints*, arXiv–2003.

Killian, T.; Konidaris, G.; and Doshi-Velez, F. 2017. Robust and efficient transfer learning with hidden parameter markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Kober, J.; Bagnell, J. A.; and Peters, J. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11): 1238–1274.

Komiak, S. Y.; and Benbasat, I. 2006. The effects of personalization and familiarity on trust and adoption of recommendation agents. *MIS quarterly*, 941–960.

Laroche, R.; and Feraud, R. 2018. Reinforcement learning algorithm selection. *International Conference on Learning Representations*.

Lee, J.; Pacchiano, A.; Muthukumar, V.; Kong, W.; and Brunskill, E. 2021. Online model selection for reinforcement learning with function approximation. In *International Conference on Artificial Intelligence and Statistics*, 3340–3348. PMLR.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Nilashi, M.; Jannach, D.; bin Ibrahim, O.; Esfahani, M. D.; and Ahmadi, H. 2016. Recommendation quality, transparency, and website quality for trust-building in recommendation agents. *Electronic Commerce Research and Applications*, 19: 70–84.

Noothigattu, R.; Bouneffouf, D.; Mattei, N.; Chandra, R.; Madan, P.; Varshney, K. R.; Campbell, M.; Singh, M.; and Rossi, F. 2019. Teaching AI agents ethical values using reinforcement learning and policy orchestration. *IBM Journal of Research and Development*, 63(4/5): 2–1.

Piech, C.; Spencer, J.; Huang, J.; Ganguli, S.; Sahami, M.; Guibas, L.; and Sohl-Dickstein, J. 2015. Deep knowledge tracing. *arXiv preprint arXiv:1506.05908*.

Ratner, A.; Bach, S. H.; Ehrenberg, H.; Fries, J.; Wu, S.; and Ré, C. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, 269. NIH Public Access.

Shah, D.; Xie, Q.; and Xu, Z. 2020. Non-asymptotic analysis of monte carlo tree search. In *Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems*, 31–32.

Tamosiunaite, M.; Nemec, B.; Ude, A.; and Wörgötter, F. 2011. Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives. *Robotics and Autonomous Systems*, 59(11): 910–922.

Taylor, M. E.; Suay, H. B.; and Chernova, S. 2011. Integrating reinforcement learning with human demonstrations of varying ability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 617–624. Citeseer.

Theocharous, G.; Chandak, Y.; Thomas, P. S.; and de Nijs, F. 2020. Reinforcement Learning for Strategic Recommendations. *arXiv preprint arXiv:2009.07346*.

van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double Q-learning. CoRR abs/1509.06461 (2015). *Proceedings of the AAAI Conference on Artificial Intelligence*.

Wachi, A.; and Sui, Y. 2020. Safe reinforcement learning in constrained markov decision processes. In *International Conference on Machine Learning*, 9797–9806. PMLR.

Warlop, R.; Lazaric, A.; and Mary, J. 2018. Fighting boredom in recommender systems with linear reinforcement learning. In *Advances in Neural Information Processing Systems*, 1757–1768.

Wu, Y.-H.; Charoenphakdee, N.; Bao, H.; Tangkaratt, V.; and Sugiyama, M. 2019. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning*, 6818–6827. PMLR.

Zheng, L.; and Ratliff, L. J. 2020. Constrained upper confidence reinforcement learning. *arXiv preprint arXiv:2001.09377*.