High Confidence Generalization for Reinforcement Learning

James E. Kostas [1] **Yash Chandak** [1] **Scott M. Jordan** [1] **Georgios Theocharous** [2] **Philip S. Thomas** [1]

Abstract

We present several classes of reinforcement learning algorithms that safely generalize to *Markov decision processes* (MDPs) not seen during training. Specifically, we study the setting in which some set of MDPs is accessible for training. For various definitions of safety, our algorithms give probabilistic guarantees that agents can safely generalize to MDPs that are sampled from the same distribution but are not necessarily in the training set. These algorithms are a type of *Seldonian* algorithm (Thomas et al., 2019), which is a class of machine learning algorithms that return models with probabilistic safety guarantees for user-specified definitions of safety.

1. Introduction

In *reinforcement learning* (RL), it is often desirable for trained agents to be robust to changes in their environments and tasks. For example, users of RL algorithms may wish to train an agent using an imperfect simulation and then deploy the agent in the real world. Unfortunately, trained RL agents are often sensitive to changes in their environment: even slight modifications may catastrophically upset an agent's ability to perform a task (Witty et al., 2018). In this work, we present a class of RL algorithms, called *high confidence generalization algorithms* (HCGAs), that provide probabilistic safety guarantees for agents' performances for environments not necessarily seen during training. These guarantees guard against catastrophic outcomes and ensure that agents can successfully generalize to entire distributions of tasks.

This work focuses on the setting where an RL task is represented by some distribution of *Markov decision processes* (MDPs). We assume that the agent is trained on some set of MDPs (that is, a training set) drawn *independently and identically distributed* (i.i.d.) from this distribution. Our algorithms then provide guarantees regarding an agent's performance on MDPs drawn from the distribution, including MDPs not in the training set.

HCGAs first train using a standard RL algorithm and then perform a *safety test* on the resulting policy. The safety test provides guarantees on the trained agent's performance. While some sophistication can be added to the learning process to account for the nature of the task, in their most basic form, the algorithms presented in this work are agnostic to the policy structure, the RL algorithm used for training, and the hyperparameters of the training algorithm. Also, no assumptions are required about the MDPs or the distribution from which they are drawn. These properties make HCGAs versatile and robust; they can be employed in any setting matching the above description, regardless of the training algorithm and policy representation used for the task.

The contributions of this paper are: **1)** a presentation and analysis of HCGAs, and a proof that they provide the probabilistic guarantees that we claim; **2)** a presentation and analysis of a class of HCGAs which provides guarantees regarding the expected performance on the MDP distribution representing the task; **3)** the proposal and analysis of an extension to the class of HCGAs described above; this extension uses control variates designed for the HCGA setting to improve these algorithms without violating the safety guarantees; **4)** a presentation and analysis of classes of HCGAs with risk-sensitive performance guarantees; and **5)** empirical results from two environment distributions that demonstrate that the safety guarantees hold in practice and that the control variate extension may improve results without violating the safety guarantees.

2. Related Work

Safe policy improvement with baseline bootstrapping (SPIBB) (Laroche et al., 2019) is a class of safe RL algorithms with some similarities to HCGAs. Both classes of algorithms generalize with high confidence to some target MDP or MDPs that may be inaccessible for training. However, the problem setting differs in several ways. Unlike HCGAs, SPIBB algorithms do not have direct access to any environment. Instead SPIBB algorithms have a *baseline* policy which they aim to improve and data (state, action, reward, state tuples) gathered from the target MDP using this baseline policy (some SPIBB algorithms need not have

[1]College of Information and Computer Sciences, University of Massachusetts, Amherst, MA, USA [2]Adobe Research. Correspondence to: James E. Kostas <jekostas@umass.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

direct access to the baseline policy (Simão et al., 2020)). These algorithms use this data to return a policy that is probabilistically guaranteed to match or exceed the performance of the baseline policy in the target MDP. Unlike SPIBB algorithms, HCGAs have direct access to a set of MDPs, may not have any data from any target or "test" MDPs, and create a policy from scratch rather than improving upon a baseline.

In RL, *transfer learning* (TL) is the study of how a policy or other knowledge may be transferred between similar but distinct tasks. The HCGA problem setting and approach falls under the broad category of TL. Taylor & Stone (2009) provide a comprehensive survey of TL techniques for RL.

Much of the RL TL literature proposes RL algorithms designed specifically to leverage the TL setting. Konidaris & Barto (2006) investigate how an RL agent may, over a series of similar tasks, learn a reward-shaping function that speeds up the learning of each individual task. Taylor et al. (2007) investigate how a policy may be transferred between two tasks with known intertask mappings between states and actions. Doshi-Velez & Konidaris (2016) and Killian et al. (2017) develop the formulation of *hidden parameter Markov decision processes* (HiP-MDPs). The HiP-MDP framework provides specialized model-based algorithms and is designed for TL between MDPs that are similar but differ slightly in dynamics. These are just a few of the many papers that propose learning algorithms designed to leverage specific properties of the TL setting.

The class of algorithms introduced in this work is *agnostic* to the specific learning algorithm used for initially training the agent: the RL algorithm could be a classical Q-learning algorithm (Watkins, 1989), or it could be a sophisticated algorithm designed to exploit some other aspect of the TL setting (for example, one of the TL algorithms above).

In TL, the *zero-shot* setting requires agents to perform well on unseen tasks without any training time on these tasks. Several papers discussed above fall partly or entirely within this category. Irpan & Song (2019) analyze this RL generalization setting and propose the *principle of unchanged optimality*, which states that "when designing a generalization benchmark, there should exist a [policy] which is optimal for all MDPs." Oh et al. (2017) discuss zero-shot TL for RL and propose a novel approach involving hierarchical skills.

HCGAs are motivated by, and most intuitively applicable to, the zero-shot setting where the principle of unchanged optimality holds, but they can be leveraged in other settings. For example, HCGAs may be used to provide guarantees of safe-but-suboptimal performance for some MDP distribution in which this principle does not hold. The resulting safe-but-suboptimal policy may then be fine-tuned in some specific application environment, as in the meta-learning setting (Finn et al., 2017).

Wang et al. (2019) study generalization in RL in the setting in which the environment transitions can be viewed as deterministic given some random variables that represent the stochasticity. They derive generalization bounds and guarantees for this setting.

Cobbe et al. (2018), Witty et al. (2018), Zhang et al. (2018), and Song et al. (2020) study the phenomena of generalization and overfitting in RL. While our work does not directly study overfitting, our empirical studies make it evident that when our algorithms cannot produce safe solutions, it is primarily because the agent is overfit to the training set; in some sense, the agent "memorizes" the training task(s) in a way that is not generalizable to other similar tasks. Our work provides algorithm designers and end-users with a principled and safe method of ensuring that their RL algorithms and agents do not overfit and fail to generalize in performance-critical applications.

3. Background and Notation

Consider an MDP, $m = (\mathcal{S}, \mathcal{A}, \mathcal{R}, P, R, d_0, \gamma)$. $\mathcal{S}$ is the set of possible states of the MDP, $\mathcal{A}$ is the set of possible actions, and $\mathcal{R}$ is the set of possible rewards. We assume that $\mathcal{S}$ and $\mathcal{A}$ are finite to simplify notation, but the methods in this paper extend to settings where these sets are infinite and uncountable. An *episode* is a sequence of states, actions, and rewards from time $t = 0$ to an indefinite value of t. The random variables S_t, A_t, and R_t are the state, action, and reward at time t. The distribution of the initial state, S_0, is given by $d_0 : \mathcal{S} \to [0, 1]$, and $\gamma \in [0, 1]$ is a parameter called the reward discount factor. A policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ defines the probability of taking each action in each state. Let $\pi(s, a) \coloneqq \Pr(A_t{=}a | S_t{=}s)$. We define a policy to be parameterized by some θ in some feasible set Θ, such that different values of θ result in different policies. $P{:}\mathcal{S}{\times}\mathcal{A}{\times}\mathcal{S}{\to}[0,1]$ is the *transition function*, defined as $P(s, a, s') \coloneqq \Pr(S_{t+1}{=}s' | S_t{=}s, A_t{=}a)$. $R{:}\mathcal{S}{\times}\mathcal{A}{\to}[0,1]$ is the *reward function*, defined as $R(s, a) \coloneqq \mathbf{E}[R_t | S_t{=}s, A_t{=}a]$.

In the typical RL setting, an agent's goal is find a θ that maximizes the *objective function* $J(\theta) \coloneqq \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t | \theta\right]$, where conditioning on θ denotes the use of a policy parameterized by θ. In this work, rather than a single MDP, we consider a distribution of MDPs. For all MDPs, we define the objective for MDP m as $J_m(\theta) \coloneqq \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t | \theta, m\right]$, where "given θ, m" indicates that the environment is MDP m and that the agent is running the policy parameterized by θ. Without loss of generality and to simplify later derivations, we assume that $J_m(\theta) \in [0, 1]$ for all $\theta \in \Theta$.

Below, we typically denote an individual MDP as M_k,

where k is some integer (for example, M_1), a distribution over some set of MDPs as μ, a set of k MDPs as $M_{1:k}$, and the set of all possible sets of MDPs as $\mathcal{M}$. We use $M_1 \sim \mu$ to denote that a single MDP M_1 is sampled from μ, and $M_{1:k} \sim \mu$ to denote that a set of k MDPs $M_{1:k}$ is sampled i.i.d. from μ. When a set of MDPs has a particular name, we denote it as M_{name} below. For example, we denote a training set of MDPs as M_{train}. We refer to the MDPs in such a set as the [name] MDPs (for example, the "training MDPs").

We define the performance (the objective) on a probability distribution of MDPs, μ, as $J_\mu(\theta) \coloneqq \mathbf{E}[J_{M_1}(\theta)|M_1 \sim \mu]$, and the performance given a finite set of MDPs of size k, $M_{1:k}$, as $J_{M_{1:k}}(\theta) \coloneqq \frac{1}{k}\sum_{m \in M_{1:k}} J_m(\theta)$. We define the random variable representing an *episodic return* for an MDP m as $G_m(\theta) \coloneqq \sum_{t=0}^{\infty} \gamma^t R_t$, where the policy used is parameterized by θ.

Let the MDPs $M_1, \ldots, M_k$ denote the individual MDPs in some finite set of MDPs $M_{1:k}$. For some policy parameterized by θ, we define the sample standard deviation of the expected returns, $\hat{\sigma}_J(\theta, M_{1:k})$, to be the sample standard deviation of the set $\{J_{M_1}(\theta), J_{M_2}(\theta), \ldots, J_{M_k}(\theta)\}$.

4. Problem Statement

The primary goal of high confidence generalization algorithms (HCGAs) is the same as in the standard RL setting: to maximize some objective. Specifically, these algorithms maximize $J_\mu(\theta)$, where μ is some arbitrary distribution of MDPs of interest; however, they do so *safely*. That is, they maximize the objective while guaranteeing that some user-defined safety constraint based on the objective or episodic returns holds. Let $f : \Theta \cup \{\text{NSF}\} \to [0, 1]$ be some *safety function* that measures the performance or episodic returns for some solution in $\Theta \cup \{\text{NSF}\}$ (NSF is defined below). All algorithms in this paper guarantee that

$$\Pr\left(f(\text{output}) \geq j\right) \geq 1 - \delta, \tag{1}$$

where $\delta \in (0, 1)$ is a user-specified probability, $j \in [0, 1]$ is a user-defined safety threshold, and where in this one equation only, "output" $\in \Theta \cup \{\text{NSF}\}$ is a random variable that represents the policy parameters output by our algorithm. This output is defined more formally below.

Any algorithm that provides this probabilistic guarantee is a *Seldonian algorithm* as proposed in Thomas et al. (2019). Seldonian algorithms output models (policies in the RL setting) with probabilistic guarantees that the models are safe for a user-defined safety metric, with any desired probability.

This work considers three definitions of the safety function f: one definition based on the expected objective J_μ (Section 6), one risk-sensitive definition based on the "worst-case" MDPs in μ (Section 8.2), and one risk-sensitive definition based on the "worst-case" episodes (Section 8.3). These definitions of f are formally defined in the sections below. For settings where other definitions of safety might be more appropriate, new HCGAs can be created as needed using the approach outlined in this paper.

Consider the case where a user defines an unreasonable value of δ or an unreasonable safety constraint. For example, if the safety constraint requires $J_\mu(\theta) > 0.9$, where θ parameterizes the policy returned by the algorithm, but $J_\mu(\theta') < 0.9$ for all $\theta' \in \Theta$, then this safety constraint cannot be satisfied. For the algorithm to handle such a case safely, we must give it a means to say "I cannot do that." *No Solution Found* (NSF) is this means.

NSF is the output produced by the algorithm when it does not have sufficient confidence that its "best-guess" candidate solution in Θ (defined formally below) will be safe to return; we always define NSF to be safe. Formally, we define $f(\text{NSF}) \coloneqq j$ for all definitions of f. Notice that these algorithms do not give any guarantees concerning the probability of returning a solution that is not NSF: a (useless) algorithm that always returns NSF would technically satisfy the guarantee above. Note that meeting the safety constraint is *not* the only goal of the algorithm; rather, the algorithm's goal is to maximize the expected performance while meeting the safety constraint. Although the naive always-return-NSF algorithm would satisfy the safety constraint, its performance would be poor in terms of the primary objective.

This formulation means that our approach is limited to applications where it is acceptable for the algorithm to not return a solution. The advantage of our approach is that these algorithms will never violate the probabilistic safety guarantees, even if the user-defined values are impossible to satisfy.

In this paper, we study the problem of finding an HCGA, `alg`, that produces a policy that maximizes the objective function, J_μ, while guaranteeing that (1) holds, for various definitions of f. Let $\mathcal{M}$ be the set of possible sets of MDPs that an HCGA could take as input. Formally, we define an HCGA, `alg`, as the function $\texttt{alg} : \mathcal{M} \to \Theta \cup \{\text{NSF}\}$. We can now use this definition to rewrite (1) more formally: $\Pr[f(\texttt{alg}(M_{\text{acc}})) \geq j] \geq 1 - \delta$, where $M_{\text{acc}} \subset \mathcal{M}$ (this set is defined below and is sampled from μ) is the random input to the algorithm.

5. Algorithm Template

The class of algorithms given in this section leverages the Seldonian framework to tackle a difficult and important problem: how to correctly give high-confidence guarantees of generalization. A summary of these algorithms follows. Let M_{acc} be a set of MDPs accessible to the algorithm, sampled i.i.d. from μ. An HCGA partitions M_{acc} into M_{train} and

M_{safety}; M_{train} is used for training, and M_{safety} is used for a safety test. The ratio of the sizes of these two sets can be viewed as a hyperparameter. The algorithm will satisfy the safety guarantees regardless of the setting of this hyperparameter, but might return NSF less frequently for certain values of this parameter. As a simple heuristic, we partition the data into two sets of equal size in all experiments. Additionally, we assume each set consists of at least two MDPs (to satisfy the requirements of all algorithms below). Next, the HCGA uses an RL algorithm and M_{train} to obtain a trained *candidate* policy, θ_c.

Finally, the algorithm performs a *safety test*: it uses M_{safety} to determine whether or not this policy meets some definition of safety for μ. Specifically, the HCGA uses some *high-confidence bounding function* $b : (\Theta \cup \{\text{NSF}\}) \times \mathcal{M} \times (0,1) \rightarrow [0,1]$. For all definitions of f below, we give one or more definitions of b. **Each of these bounding function definitions, combined with the template below, forms a complete algorithm.** The HCGA uses the bounding function to, with the specified confidence, establish a high-confidence lower bound on the value of $f(\theta_c)$. If the candidate policy is safe with the specified confidence, that policy is returned. Otherwise, the algorithm returns NSF. This general form of HCGAs is given in Algorithm 1.

Algorithm 1 HCGA Template

Input : Feasible set Θ, a set of MDPs M_{acc}, user-defined threshold j, probability $1-\delta$, and high-confidence bounding function b.
Output : $\theta \in \Theta \cup \{\text{NSF}\}$

```
Partition M_acc into two data sets, M_train and M_safety;
Compute a θ_c ∈ argmax_{θ∈Θ} J_{M_train}(θ);
if b(θ_c, M_safety, δ) ≥ j then return θ_c;
else return NSF;
```

For all $\theta \in \Theta \cup \{\text{NSF}\}$ and $\delta \in (0,1)$, if Algorithm 1 takes a bounding function b such that $\Pr(b(\theta, M_{\text{safety}}, \delta) \leq f(\theta)) \geq 1-\delta$, then the algorithm will return a safe result with at least probability $1-\delta$. Formally:

Theorem 1. *If* $\Pr(b(\theta, M_{safety}, \delta) \leq f(\theta)) \geq 1-\delta$, *then*

$$\Pr[f(\texttt{alg}(M_{acc})) \geq j] \geq 1-\delta.$$

Proof. See supplementary material Section A. □

Notice that in practice, `alg` can include stochasticity in the optimization process (for example, stochasticity due to the transition function, policy, etc.). One way to capture this stochasticity is to have the algorithm take a random seed as input. For example, in the case where the seed is an integer, `alg` would be the function $\texttt{alg} : \mathcal{M} \times \mathbb{Z} \rightarrow \Theta \cup \{\text{NSF}\}$. For brevity, we make this random seed input implicit.

In Figure 2 in supplementary material Section D, we provide a concise summary of the four HCGAs that we present and study in this paper.

6. Expected Return HCGAs

In this section, we present a class of HCGAs with safety constraints specifying that the expected performance should be above some threshold. Specifically, the algorithms in this class define the safety function f to be J_μ, and therefore give the following probabilistic guarantee: $\Pr\left(J_\mu(\texttt{alg}(M_{\text{acc}})) \geq j\right) \geq 1-\delta$.

Two examples of high-confidence bounding functions for this definition of safety are below, based on Hoeffding's inequality (Hoeffding, 1994) and Student's t-test (Student, 1908), respectively:

$$b(\theta, M_{\text{safety}}, \delta) := J_{M_{\text{safety}}}(\theta) - \sqrt{\ln(1/\delta)/(2|M_{\text{safety}}|)}, \quad (2)$$

$$b(\theta, M_{\text{safety}}, \delta) := J_{M_{\text{safety}}}(\theta) - \frac{\hat{\sigma}_J(\theta, M_{\text{safety}})\texttt{t}_{1-\delta,|M_{\text{safety}}|-1}}{\sqrt{|M_{\text{safety}}|}}, \quad (3)$$

where the sample standard deviation function $\hat{\sigma}_J$ used in (3) is defined in Section 3, and the $\texttt{t}_{*,*}$ used in (3) represents the inverse cumulative distribution function of the Student's t distribution. The t-test bound represented by (3) will often be tighter than that represented by (2), but the t-test bound requires the assumption that the performances of θ_c for the MDPs in M_{safety} are normally distributed. This assumption may not be reasonable, especially for small values of $|M_{\text{safety}}|$. However, by the central limit theorem, it is often a reasonable assumption for large values of $|M_{\text{safety}}|$.

In Algorithm 3 in supplementary material Section L, we give the algorithm represented by the bounding function defined in (2). This serves as an example of how to apply bounding functions to Algorithm 1 to form a complete HCGA. For all other variants, such as that represented by (3), we provide only the bounding functions.

7. Expected Return HCGAs with Control Variates

In this section, we consider a slightly modified problem setting: one in which **1)** each MDP is parameterized by a *known* set of parameters, p_i, in an arbitrary space, $\mathcal{P}$ (for example, the space of possible friction coefficients), and **2)** *parameters* of MDPs can be sampled from the entire distribution of MDPs without needing to (or necessarily having the capability to) construct or run episodes of these MDPs. This setting is of interest when training in simulation using a distribution of MDPs, since the parameters of these MDPs and the MDP distribution, μ, are usually known.

One way to exploit this additional information is to learn a control variate for the candidate policy's expected return given MDP parameters $p_i \in \mathcal{P}$, and to use this control variate to derive unbiased estimates of J_μ that have lower variance than the estimates used in the previous section. These lower variance estimates can then be used in the bounding functions to reduce the probability of returning NSF without compromising the safety guarantees. This method uses a constant which can take any real value; we propose two theoretically grounded methods for choosing optimal values for this constant.

In this section, we write $p_i \in \mathcal{P}$ to denote the parameters of the i^{th} MDP, M_i. Note that $\mathbf{E}[(\text{some expression involving } p_i)|M_i \sim \mu]$ means that p_i are the parameters of MDP M_i.

We introduce a control variate that takes MDP parameters $p_i \in \mathcal{P}$ as input, and estimates the objective value of the corresponding MDP M_i for the policy parameterized by θ. Formally, we write this control variate as the function $\bar{v}_\theta : \mathcal{P} \to [0, 1]$ (recall that we assume the objective and returns are normalized to $[0, 1]$). For example, given some $\theta \in \Theta$ and MDP M_i, $\bar{v}_\theta(p_i)$ estimates $J_{M_i}(\theta)$. The control variate can be an arbitrary function trained with an arbitrary supervised learning algorithm as described below.

Define $Z_i(\theta, c, \bar{v}_\theta, \mu) := J_{M_i}(\theta) - c(\bar{v}_\theta(p_i) - \mathbf{E}[\bar{v}_\theta(p_j)|M_j \sim \mu])$, for some constant $c \in \mathbb{R}$. For brevity, we abbreviate $Z_i(\theta, c, \bar{v}_\theta, \mu)$ as Z_i. Let $M_1, M_2, \ldots, M_k$ be the safety MDPs. While computing the safety test, instead of using $J_{M_1}(\theta), J_{M_2}(\theta), \ldots, J_{M_k}(\theta)$ to estimate the mean, we can instead use the unbiased and potentially lower variance estimates of the mean $Z_1, Z_2, \ldots, Z_k$.

Property 1. *For all $\theta \in \Theta$, for all $c \in \mathbb{R}$, Z_i is an unbiased estimator of $J_\mu(\theta)$.*

The proofs of Properties 1 and 2 and Corollary 1 are given in supplementary material Section B.

Next, we address the question of how to choose a value of c to minimize variance. To estimate an optimal value, we derive an expression for the variance of Z_i and then minimize this expression with respect to c. For the purposes of Property 2 and Corollary 1 below, we assume that the learned control variate is not a constant (that is, it varies with its input, the MDP parameters). This assumption is given more formally in supplementary material Section B.

Property 2.

$$\begin{aligned}
&\operatorname*{argmin}_{c \in \mathbb{R}} \operatorname{Var}(Z_i|M_i \sim \mu) \\
=&\mathbf{E}\Big[(J_{M_i}(\theta) - \mathbf{E}[J_{M_k}(\theta)|M_k \sim \mu]) \\
&\times (\bar{v}_\theta(p_i) - \mathbf{E}[\bar{v}_\theta(p_j)|M_j \sim \mu])\Big|M_i \sim \mu\Big] \\
&/\ \mathbf{E}\Big[(\bar{v}_\theta(p_{i'}) - \mathbf{E}[\bar{v}_\theta(p_{j'})|M_{j'} \sim \mu])^2\Big|M_{i'} \sim \mu\Big],
\end{aligned}$$

where $\times$ and $/$ denote scalar multiplication and division respectively, split across multiple lines.

An alternative to this method of calculating c follows. Ideally, the control variate $\bar{v}_\theta(p_i)$ will converge to a perfect estimator of $J_{M_i}(\theta)$ given sufficient training data. Consider the value of c for the setting in which the control variate has converged to a perfect estimator: $\bar{v}_\theta(p_i) = J_{M_i}(\theta)$.

Corollary 1. *If $\bar{v}_\theta(p_i) = J_{M_i}(\theta)$, then*

$$\operatorname*{argmin}_{c \in \mathbb{R}} \operatorname{Var}(Z_i|M_i \sim \mu) = 1.$$

Again, the proofs of Properties 1 and 2 and Corollary 1 are given in supplementary material Section B.

Given these properties, it is straightforward to apply control variates to expected value HCGAs such as the Student's t-test HCGA: **1)** After training, before the safety test, evaluate $J_{M_i}(\theta)$ for all $M_i \in M_{\text{train}}$. **2)** Use a supervised learning algorithm of choice to learn a $\bar{v}_\theta(p_i)$, using the $J_{M_i}(\theta)$ values (from the training data). This is a simple regression problem. **3)** Use Property 2 to estimate the optimal c value, using the training MDPs, **or** choose c=1 (we compare and analyze these two methods in supplementary material Section M). **4)** Proceed with the safety test using MDPs $M_1, M_2, \ldots, M_k \in M_{\text{safety}}$ using the set $\{Z_1, Z_2, \ldots, Z_k\}$ instead of the set $\{J_{M_1}(\theta), J_{M_2}(\theta), \ldots, J_{M_k}(\theta)\}$ to calculate the value of the high-confidence bounding function b. These steps are given more formally in Algorithm 2 in supplementary material Section C.

The approach proposed in this section may be particularly advantageous using bounds that vary significantly with the variance of the estimates (for example, Student's t-test). However, even in the case of bounds that do not vary significantly with the variance of the estimates (for example, Hoeffding's), an unbiased but lower variance estimator of the mean can be considered a strict improvement to the HCGAs proposed in the previous section.

8. Risk-Sensitive HCGAs

The above bounds concern the *expected*, or average, return j. In other words, they guarantee that a solution will, with user-specified probability $1 - \delta$, result in an average return greater than or equal to some j. Such solutions could, however, regularly result in MDPs drawn from μ with objective function values less than j or episodes for MDPs drawn from μ with returns less than j. Even a majority of MDPs and/or episodes could result in objective function values and/or returns, respectively, of less than j; as long as the expected return is above j, the criterion above is satisfied.

For this reason, the expected value may not be a suitable measure of safety in some settings. This section proposes

two alternative definitions of safety based on the *conditional value at risk* (CVaR): bounds concerning the *distribution of expected returns* for an MDP drawn from μ, and bounds concerning the *distribution of episodic returns* for an MDP drawn from μ. Supplementary material Section E introduces and defines CVaR for readers unfamiliar with it.

Value at risk (VaR) is another popular risk measure. However, VaR has the disadvantage of being insensitive to rare catastrophic risks (and such risks are one of the primary motivations for definitions of safety not based on the expected value). For this reason, we only introduce CVaR-based HCGAs in this work. However, one could create VaR-based HCGAs for situations where VaR might be a more appropriate measure of safety (see supplementary Section E for a brief discussion and, for readers unfamiliar with it, an introduction to VaR).

8.1. CVaR Bounds

Our algorithms require high-confidence guarantees on the CVaR of a policy's performance or returns, and so we require sample-based bounds on the CVaR of a random variable. In this section, we review such bounds.

We analyze the bounds of Brown (2007) and Thomas & Learned-Miller (2019). The latter bound tended to be tighter in our experiments, so we use it for our results, but the former bound is relatively simple to write and manipulate, and *not* strictly looser, so it may be more desirable in some applications. Therefore, we present the required analyses for both bounds.

Conventions differ as to whether VaR and CVaR are with respect to the lowest or highest possible values of the distribution (Thomas & Learned-Miller, 2019). We use the convention that they are with respect to the lowest possible values, since it better matches the RL setting, where lower values are less desirable. In Section H of the supplementary material, we provide a proof that the "left tail" bound given in Property 3 below is equivalent to the "right tail" bound of Thomas & Learned-Miller (2019). In Sections F and G of the supplementary material, we provide a similar bound and proof for the bounds of Brown (2007).

Let X be a random variable such that $\text{supp}(X) \subseteq [a, b]$. Given a sample of X of size n, let $W_0 := a$, and $W_1, \dots, W_n$ be the order statistics of the sample (that is, the sample sorted into increasing order). Thomas & Learned-Miller (2019) bound CVaR with high confidence:

Property 3. *For all $\delta \in (0, .5]$:*

$$\Pr\left(\text{CVaR}_\alpha(X) \geq W_0 + \frac{1}{\alpha}\sum_{i=1}^{n}(W_{n+1-i} - W_{n-i}) \times \max\left(0, \frac{i}{n} - \sqrt{\frac{\ln(1/\delta)}{2n}} - (1-\alpha)\right)\right) \geq 1 - \delta,$$

where $\times$ denotes scalar multiplication.

8.2. High Confidence Generalization Across MDPs

Instead of a bound on the value of J_μ, we may instead wish to design an algorithm that, with specified confidence $1 - \delta$, returns a solution with guarantees regarding risk measures for *an MDP drawn from* μ. Such bounds are appropriate when the user desires safety constraints on the expected return for the "worst-case MDPs" in μ. Specifically, for all $\theta \in \Theta \cup \{\text{NSF}\}$, the algorithm in this class uses the following definition of the safety function: $f(\theta) := \text{CVaR}_\alpha(J_{M_1}(\theta)|M_1 \sim \mu)$. Therefore, the probabilistic guarantee is $\Pr(\text{CVaR}_\alpha(J_{M_1}(\texttt{alg}(M_{\text{acc}}))|M_1 \sim \mu) \geq j|M_{\text{acc}} \sim \mu) \geq 1 - \delta$.

Recall that $M_1 \sim \mu$ denotes a single MDP M_1 sampled from μ, and $M_{\text{acc}} \sim \mu$ denotes a set of MDPs M_{acc} sampled i.i.d. from μ. In the inequality above, note that M_1 and M_{acc} are sampled independently of each other.

Given a sample of size n consisting of $J_{M_1}(\theta), \dots, J_{M_n}(\theta)$, where $M_1, \dots, M_n$ are the safety MDPs, let $J_1, \dots, J_n$ be the n order statistics of that sample (that is, the objective values of the n MDPs sorted in increasing order). Define $J_0 := 0$. Applying Property 3, the bounding function is: $b(\texttt{alg}(M_{\text{acc}}), M_{\text{safety}}, \delta) := \frac{1}{\alpha}\sum_{i=1}^{n}(J_{n+1-i} - J_{n-i})\max(0, \frac{i}{n} - \sqrt{\frac{\ln(1/\delta)}{2n}} - (1-\alpha))$. Recall that, combined with Algorithm 1, this bounding function represents a complete algorithm. We refer to this algorithm as the *CVaR MDP HCGA*.

8.3. High Confidence Generalization for All Episodes

Alternatively, we may instead wish to design an algorithm that, with specified confidence $1 - \delta$, returns a solution with guarantees regarding risk measures for *episodic returns* for episodes drawn from μ. Such bounds are useful when the distribution of *episodic returns* is more relevant to safety than *expected returns*. Specifically, for all $\theta \in \Theta \cup \{\text{NSF}\}$, the algorithm in this class uses the following definition of safety: $f(\theta) := \text{CVaR}_\alpha(G_{M_1}(\theta)|M_1 \sim \mu)$. In other words, users may wish to use this class of algorithms when risk measures on "worst-case episodes" are the best measure of safety. For example, when applying RL to diabetes management (Bastani, 2014), a risk-sensitive measure of episodic returns such as CVaR may be a better safety constraint than the guarantees above: a single bad "episode" could result in

the death of a patient, regardless of the value of the objective function for μ (Section 6) or the objective function for an MDP drawn from μ (Section 8.2).

This is different from the algorithm described in Section 8.2 in that there is no expectation inside the CVaR function: the algorithm considers the returns of individual episodes rather than the expected returns of those episodes (that is, rather than the objective function). The probabilistic guarantee is $\Pr(\text{CVaR}_\alpha(G_{M_1}(\texttt{alg}(M_{\text{acc}}))|M_1 \sim \mu) \geq j|M_{\text{acc}} \sim \mu) \geq 1-\delta$.

Given a sample of size n consisting of $G_{M_1}(\theta)$, $\ldots, G_{M_n}(\theta)$, where $M_1, \ldots, M_n$ are the safety MDPs, let $G_1, \ldots, G_n$ be the n order statistics of that sample. Define $G_0 \coloneqq 0$. The bounding function is $b(\texttt{alg}(M_{\text{acc}}), M_{\text{safety}}, \delta) := \frac{1}{\alpha}\sum_{i=1}^{n}(G_{n+1-i} - G_{n-i}) \max\left(0, \frac{i}{n} - \sqrt{\frac{\ln(1/\delta)}{2n}} - (1-\alpha)\right)$. In the results, we refer to this algorithm as the *CVaR Episodic HCGA*.

9. Experiments and Results

In this section, we run the four algorithms defined by the bounds above on two sets of MDPs: generalization gridworld and *dynamic arm simulator one* (DAS1) (Blana et al., 2009). In both cases, we define μ to be a uniform distribution over the sets of MDPs. Generalization gridworld is a set of gridworlds in which randomly placed "cliffs" send the agent back to the start state, but a fixed path from the start state to the goal is always clear of cliffs. As a result, while individual MDPs may have many optimal policies, there is only one optimal policy for the entire set.

DAS1 is a detailed and biomechanically plausible human arm simulator with two joints and six muscles. This environment simulates *functional electrical stimulation* (FES) for paralyzed arm muscles; it has been used in biomedical research studying patients suffering from paralysis due to brain or spinal cord injuries (Blana et al., 2009). When patients suffer paralysis due to these types of injuries, the arms and other areas of the body still have the potential to move; the muscles and the local nerves are intact, but the connection to the brain is severed. Advances in the science of FES could someday allow patients paralyzed below the neck to be able to move their arms and other parts of their body again. One challenge of FES is that controllers based on models or trained in simulation often do not perform well when applied to real physiological systems, in part because of the physiological variations between individual subjects (Jagodnik (2014), see Sections 1.4 and 1.5). In fact, individual physiological variations have caused agents trained with DAS1 to not work well on a real-world FES setting with a paralyzed subject (K. M. Jagodnik, personal communication, June 4, 2020). Therefore, the study of the DAS1 domain (and how to ensure that agents trained in it generalize successfully) is an important and impactful application motivating HCGAs. Both environments are described in more detail in supplementary material Section J.

In each experiment, we randomly choose an accessible set of MDPs, M_{acc}, and a *test set*, M_{test}. The latter is a large set of 10,000 MDPs not accessible to the algorithm. We use the test set as the ground truth: it can be used to determine whether an HCGA's returned policy is actually safe, what that policy's true performance is for μ, and what the difference is between its performance for the training set and for μ.

All hyperparameters and experimental details are given in supplementary material Section K.1.

For the plots in this section, we define $J(\text{NSF}) \coloneqq j$ (the same definition as $f(\text{NSF})$). This plotting methodology has the advantage of showing all trials, but has the disadvantage of sometimes causing the HCGA to appear to perform significantly better or worse than the average $J_\mu(\theta_c)$ value, depending on the definition of j and on the frequency with which the algorithm returns NSF. Results excluding trials in which the algorithm returns NSF (and which therefore do not require this definition of $J(\text{NSF})$) are discussed below and are available in supplementary material Section K.3. The HCGAs in this section do not use control variates unless otherwise specified.

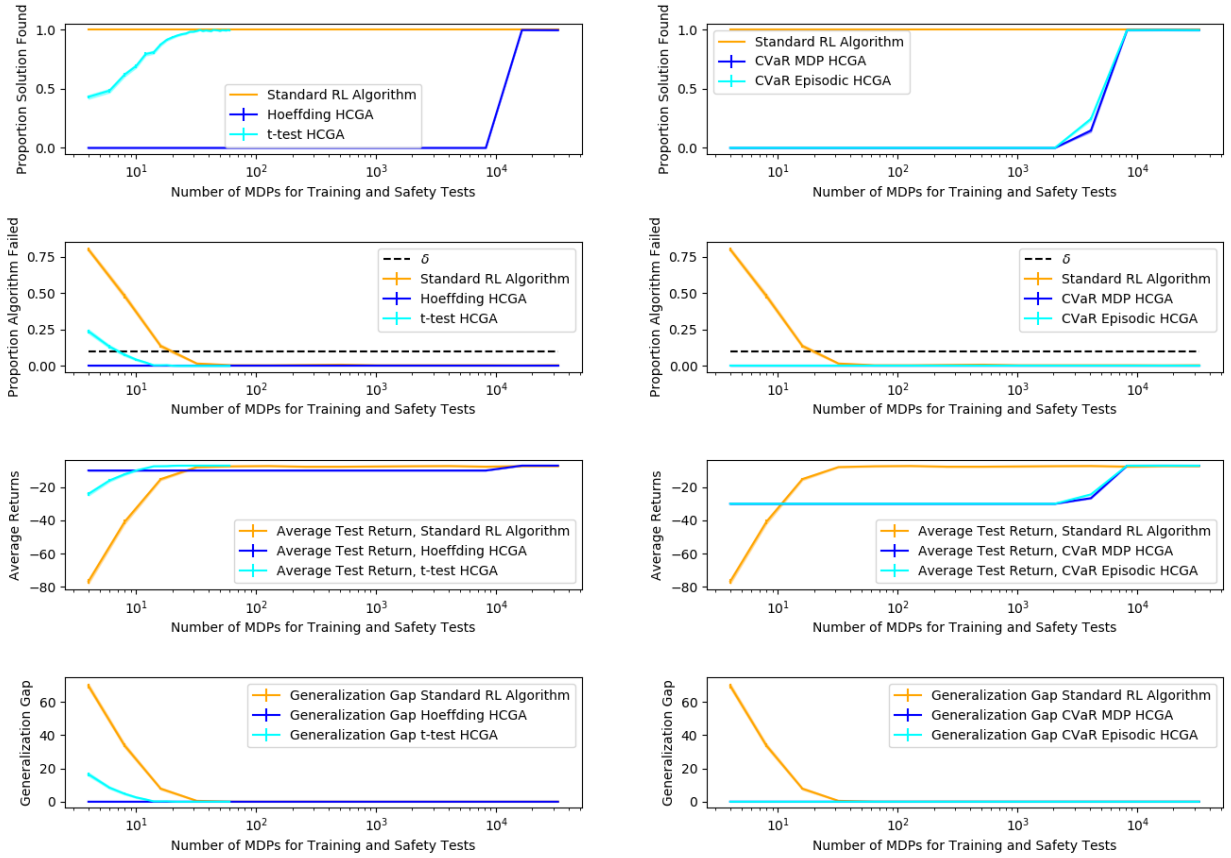
Let the *generalization gap* be defined as the difference between the training and test performances for the algorithm's output $\theta \in \Theta \cup \{\text{NSF}\}$: $J_{M_{\text{train}}}(\theta) - J_{M_{\text{test}}}(\theta)$.

9.1. Generalization Gridworld Results

The results of the generalization gridworld experiments are presented in Figure 1; they demonstrate that the HCGAs' guarantees hold. Notice that the proportion of trials in which the HCGAs failed is below δ in all cases, except in the case of the Student's t-test algorithm for low values of $|M_{\text{acc}}|$. This is expected: at low values of $|M_{\text{acc}}|$, the t-test HCGA's assumption of normality is not reasonable, so the algorithm may return an unsafe solution more than $\delta(100\%)$ of the time. These results demonstrate empirically that the probabilistic guarantees given by HCGAs hold in practice.

The fourth plot in each figure, which plots the generalization gap, makes it evident that HCGAs are preventing overfitting and ensuring generalization: the safety tests detect when a candidate policy is overfit to its training set, and reject that policy as unsafe, resulting in a significantly smaller generalization gap for HCGAs than for standard RL algorithms.

For the Hoeffding and the CVaR HCGAs, notice that the number of MDPs required to reach the best plotted candidate solution (that is, one with a generalization gap near zero) is



(a) Hoeffding and t-test Gridworld Results

(b) CVaR Gridworld Results

Figure 1. Generalization gridworld results. In all plots, the horizontal axis is the number of MDPs accessible for training and safety tests (that is, $|M_{\text{acc}}|$). All error bars represent standard error. In all plots, the phrase "Standard RL Algorithm" represents an algorithm which does not run a safety test, and instead naively maximizes the objective. The first (top) plots show the proportion of trials in which a solution is found (that is, trials in which the algorithm did not return NSF). The second plots in these figures show the proportion of trials in which the algorithm fails; that is, the proportion of trials in which an unsafe solution is returned. In all experiments, $\delta = 0.1$. The third plots in each figure show the average returns. The fourth plots show the generalization gap. These plots were generated using 1000 trials per data point (that is, 1000 trials for each location on the horizontal axis). All other details are discussed in supplementary material Section K.1.

orders of magnitude less than the number of MDPs required to consistently return a solution. This indicates that our simple heuristic of partitioning the data into two sets of equal size is poor in these settings. By allocating fewer MDPs to the training set and more MDPs to the safety set, one could design an HCGA utilizing these bounds that requires significantly fewer MDPs in M_{acc} to return a solution.

9.2. DAS1 Results

DAS1 results using the same layout as Figure 1 are given in Section K.2 of the supplementary material. Individual plots for each of the eight experiments (four HCGAs run for two environments) are given in Section K.3 of the supplementary material (these individual plots use roughly four pages of space, but may be easier to read in the individual format).

The DAS1 results are similar to the gridworld results and demonstrate empirically that 1) the probabilistic guarantees given by HCGAs hold in practice and 2) HCGAs prevent overfitting and ensure generalization.

Notice that in both environments, the two HCGA CVaR algorithms return nearly identical results. Because $J(\theta) = \mathbf{E}[G(\theta)]$, this phenomenon may be common in settings for which variances in episodic returns for each MDP are low, but for which variances in episodic returns across the distribution of MDPs are high. Those conditions will cause the two CVaR definitions of f to take approximately the same value. Future work will study this phenomenon further, but the experiments make it clear that the safety guarantees hold for both CVaR algorithms.

9.3. Control Variate Results

We also study the effect of control variates on expected value HCGAs. Empirical results confirm our theoretical analysis: control variates reduce the variance of the mean estimators without violating the HCGA safety constraints. For more details, see supplementary material Section M.

9.4. Applicability to Computationally Expensive Settings

Because our plots require many trials (100 or 1000 per location on the horizontal axis in our experiments) to reasonably show the "proportion solution found" and "proportion algorithm failed" plots, we chose to perform experiments using environments that are relatively computationally inexpensive. However, when *applying* the HCGA framework to a real-world problem, one must only perform one trial (not hundreds or thousands as in our plots), which makes these algorithms scalable and practicable for computationally expensive applications. Because of our theoretical results, one can confidently apply HCGAs in these settings; the theoretical results hold whether the function approximator is a simple Q-Table, a linear approximator, or the latest and largest deep network architecture. Furthermore, since the computational bottleneck tends to be training (the safety test requires only evaluation of the candidate policies and is thus relatively inexpensive), HCGAs are typically not significantly more computationally expensive than running a standard RL algorithm without the HCGA framework.

10. Conclusion

In this paper, we introduce high confidence generalization algorithms, prove that the probabilistic guarantees given by these algorithms hold, extend one class of these algorithms with control variates, and show empirically that these guarantees hold in practice. Future work will study new types of HCGAs as well as HCGAs in the extrapolation setting, in which M_{acc} is not drawn from the same distribution as M_{test}.

Acknowledgements

Research reported in this paper was sponsored in part by a gift from Adobe, NSF award #2018372, and the DEVCOM Army Research Laboratory under Cooperative Agreement W911NF-17-2-0196 (ARL IoBT CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

Bastani, M. Model-free intelligent diabetes management using machine learning. Master's thesis, University of Alberta, 2014.

Blana, D., Kirsch, R. F., and Chadwick, E. K. Combined feedforward and feedback control of a redundant, nonlinear, dynamic musculoskeletal system. *Medical & Biological Engineering & Computing*, 47(5):533–542, 2009.

Brown, D. B. Large deviations bounds for estimating conditional value-at-risk. *Operations Research Letters*, 35(6): 722–730, 2007.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018.

Doshi-Velez, F. and Konidaris, G. Hidden parameter Markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *IJCAI'16: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pp. 1432–1440, 2016.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135, 2017.

Hoeffding, W. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pp. 409–426. Springer, 1994.

Irpan, A. and Song, X. The principle of unchanged optimality in reinforcement learning generalization. In *ICML 2019 Workshop on Understanding and Improving Generalization in Deep Learning*, 2019.

Jagodnik, K. M. *Reinforcement learning and feedback control for high-level upper-extremity neuroprostheses*. PhD thesis, Case Western Reserve University, 2014.

Killian, T. W., Daulton, S., Konidaris, G., and Doshi-Velez, F. Robust and efficient transfer learning with hidden parameter Markov decision processes. In *Advances in Neural Information Processing Systems*, pp. 6250–6261, 2017.

Konidaris, G. and Barto, A. Autonomous shaping: Knowledge transfer in reinforcement learning. In *International Conference on Machine Learning*, pp. 489–496, 2006.

Konidaris, G., Osentoski, S., and Thomas, P. Value function approximation in reinforcement learning using the Fourier basis. In *Twenty-fifth AAAI Conference on Artificial Intelligence*, 2011.

Laroche, R., Trichelair, P., and Des Combes, R. T. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning*, pp. 3652–3661, 2019.

Oh, J., Singh, S., Lee, H., and Kohli, P. Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2661–2670, 2017.

Simão, T. D., Laroche, R., and Combes, R. T. d. Safe policy improvement with an estimated baseline policy. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, pp. 1269–1277, 2020.

Song, X., Jiang, Y., Tu, S., Du, Y., and Neyshabur, B. Observational overfitting in reinforcement learning. In *International Conference on Learning Representations*, 2020.

Student. The probable error of a mean. *Biometrika*, 6(1): 1–25, 1908.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

Taylor, M. E. and Stone, P. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

Taylor, M. E., Stone, P., and Liu, Y. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(1):2125–2167, 2007.

Thomas, P. and Learned-Miller, E. Concentration inequalities for conditional value at risk. In *International Conference on Machine Learning*, pp. 6225–6233, 2019.

Thomas, P. S., da Silva, B. C., Barto, A. G., Giguere, S., Brun, Y., and Brunskill, E. Preventing undesirable behavior of intelligent machines. *Science*, 366(6468):999–1004, 2019.

Wang, H., Zheng, S., Xiong, C., and Socher, R. On the generalization gap in reparameterizable reinforcement learning. In *International Conference on Machine Learning*, pp. 6648–6658, 2019.

Watkins, C. *Learning From Delayed Rewards*. PhD thesis, University of Cambridge, England, 1989.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.

Witty, S., Lee, J. K., Tosch, E., Atrey, A., Littman, M., and Jensen, D. Measuring and characterizing generalization in deep reinforcement learning. *arXiv preprint arXiv:1812.02868*, 2018.

Zhang, A., Ballas, N., and Pineau, J. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.