Discovery and Analysis of Rare High-Impact Failure Modes Using Adversarial RL-Informed Sampling

Rory Lipkis
Intelligent Systems Division
NASA Ames Research Center
Moffett Field, CA
rory.lipkis@nasa.gov

Adrian Agogino
Intelligent Systems Division
NASA Ames Research Center
Moffett Field, CA
adrian.k.agogino@nasa.gov

ABSTRACT

Adaptive learning agents have tremendous potential to handle critical tasks currently performed by humans. Unfortunately, due to their complexity, it can be difficult to verify that these learning agents do not have critical failure modes. Standard verification and validation methods often do not apply directly to learning agents and Monte Carlo methods have difficulty covering even a small fraction of the state space, especially in multiagent systems or over long time horizons. To overcome this difficulty, we demonstrate an adaptive stress-testing method based on reinforcement learning of correlations that raise the probability of failure. This approach has three key properties: (1) it is able to find rare failure modes with far greater sample efficiency than Monte Carlo methods, (2) it can estimate the true probability of a failure mode despite the inherent bias in the learning method, and (3) it is capable of learning and resampling compact representations of multimodal failure spaces. These properties are important in practice as we need to find disparate failure modes while accounting for their actual relevance. This is a significant advantage over traditional adaptive stress testing methods that give abstract likelihoods of particular failure instances, but cannot estimate the probability of a broader failure mode. We test our algorithm on a simple problem from the aviation domain where an autonomous aircraft lands in gusty wind conditions. The results suggest that we can find failure modes with far fewer samples than the Monte Carlo approach and simultaneously estimate the probability of failure.

KEYWORDS

Reinforcement learning, validation, statistical failure analysis

1 INTRODUCTION

Validation of complex stochastic systems is challenging. Methods such as Monte Carlo sampling often fail, since for large complex systems, a very small fraction of possible outcomes can be realistically sampled. Indeed, even learning problems involving simple sequences of actions can have enormous state spaces that compound with each additional step. The problem becomes even worse in the case of multiagent systems.

To address these issues, we propose to use a stochastic validation process based on reinforcement learning. Validation of a system under test (SUT) entails determining through testing and analysis whether specified requirements are met. Traditionally, statistical validation (falsification) is performed by randomly sampling system inputs and transitions, producing an estimate of the failure likelihood. This Monte Carlo approach becomes computationally infeasible when validating the performance of complex systems over longer horizons. For example, in the case of collision avoidance systems, a common target of verification efforts, system failure might only result from the unlikely confluence of reckless operation and bad luck over an extended period of time.

Detecting such complicated failure modes is a difficult task. Since direct sampling explores regions in proportion to their likelihood of occurrence, the majority of computational effort is expended evaluating near-nominal system behavior. Any amount of model error – neglecting a small interstep correlation, for instance – can render a valid failure mode undiscoverable. In the event that a failure event is detected, it may not recur sufficiently to characterize the larger mode.

To mitigate this problem, it is common to manually bias the search in a manner that takes advantage of domain knowledge to expedite failure discovery. Though successful in eliciting higher numbers of failures, this technique arguably jeopardizes the notion of *independence* in verification and validation. If a prejudice towards expected failure modes is built into the testing methodology, the tester potentially forecloses on discovering unknown failure modes. For safety-critical systems, these may represent the failures of most concern.

Adversarial testing provides a compelling solution to this dilemma. As AI systems become more widespread, much research has focused on generating adversarial attacks against deep neural networks in decision and perception subsystems. For instance, it has long been observed that classifier accuracy can be vulnerable to small perturbations in the input, particularly when a model has not been trained for robustness [3].

In recent years, many efforts have applied the idea of adversarial testing to autonomous control systems in simulation. Intelligent test case generation has been used to find static environment parameters that challenge self-driving vehicle algorithms [18, 20]. In such cases, different combinations of weather conditions, sensor faults, and pedestrians are explored until critical requirements are violated. A more fine-grained testing strategy involves step-wise perturbations. In particular, a controller that makes decisions based on its surroundings can be pitted against an adversarial agent in the same environment whose goal is to force violations [2, 19]. In such experiments, the opponent's disruptive ability must be limited if the goal is to produce realistic failure cases, often achieved with a handpicked heuristic.

Proc. of the Adaptive and Learning Agents Workshop (ALA 2023), Cruz, Hayes, Wang, Yates (eds.), May 9-10, 2023, Online, https://ala2023.github.io/. 2023.

In adaptive stress testing (AST), the adversarial agent is the environment itself, which is taken to be a probabilistic model of perturbations [12]. This induces a natural measure of likelihood that is factored into the agent's reward function to limit its adversarial capacity. AST thus provides a framework for learning the *likeliest* failures of an autonomous SUT with reinforcement learning. Information about unsafe regions of the state space can be automatically gathered and exploited, allowing failure modes to be discovered more efficiently while maintaining a degree of objectivity. This approach has proven useful for risk characterization and system development, and has seen use in a variety of applications, including aircraft collision avoidance [12, 13], autonomous driving [1, 7], trajectory planning for small unmanned aircraft [11], autonomous aircraft taxiing [5], and flight management [15].

In this paper, we present a framework that combines the likelihood awareness of AST with explicit policy learning and importance resampling, providing several advantages: (1) rare and disparate failure modes can be efficiently discovered, (2) failure mode probabilities can be estimated with low bias, and (3) learned failure modes can be resampled, generating random failure cases with minimal additional computation. These benefits allow for a much more comprehensive analysis of failure events in complex systems.

2 ADAPTIVE STRESS TESTING

In the AST formulation, the SUT interacts with a stochastic environment within a simulation. The simulation is summarized by a state $s \in \mathcal{S}$; failure corresponds to some subset $\mathcal{F} \subset \mathcal{S}$. The environment consists of a set of external disturbances collected into the random variable $X \in \mathcal{X}$, where $X \sim p(x)$. It is helpful to specify some sort of distance-to-failure metric $d(s)$ to guide the learning. Recent research has demonstrated success in the absence of such a heuristic, although this requires a significantly more specialized solution technique [8].

As an example, in the aircraft collision avoidance setting, the state might contain the positions and velocities of several aircraft while the environment describes externalities (from the perspective of the SUT) such as pilot controls, wind gusts, or sensor noise. A sensible distance metric would be the minimum pairwise distance between aircraft.

Rather than sampling values from the environment as in traditional sample-based testing, AST explicitly optimizes over the disturbances to find the most likely failure events in the SUT. For an arbitrary T-step trajectory, the joint likelihood is given by

$$\begin{aligned} p(s_0, \ldots, s_T) &= p(s_0) \prod_{t=1}^{T} p(s_t \mid s_0, \ldots, s_{t-1}) \\ &= p(s_0) \prod_{t=1}^{T} p(s_t \mid s_{t-1}) \\ &= p(s_0) \prod_{t=0}^{T-1} p(x_t \mid s_t)\,, \end{aligned} \tag{1}$$

where simplifications are due to the Markov property and the assumption that all randomness is captured in the specification of the environment (i.e., the SUT is either deterministic or derandomizable). Thus, the high-level goal of AST is to solve the optimization problem

$$\begin{aligned} &\max_{x_0, \ldots, x_{T-1}} \prod_{t=0}^{T-1} p(x_t \mid s_t) \\ &\text{subject to } s_T \in \mathcal{F} \end{aligned} \tag{2}$$

for a fixed initialization s_0. By considering failure states to be absorbing, one may account for failures that occur at any $t \leq T$. Note that this problem amounts to the maximization of a relatively simple objective function subject to an arbitrarily complex constraint. The likely non-convexity (and even disconnectedness) of the feasible region suggests that classical optimization methods may be insufficient.

In AST, the problem is formulated quite naturally as a Markov decision process (MDP), enabling the use of reinforcement learning techniques. Figure 1 illustrates the typical AST architecture. At each time step, the agent observes the state of the simulation, selects an action (an environment instance), and receives a reward

$$r(s, x) = \begin{cases} r_f & \text{if } s \in \mathcal{F} \\ -d_{\min} & \text{if } s \notin \mathcal{F},\ t = T \\ \log p(x \mid s) & \text{if } s \notin \mathcal{F},\ t < T\,, \end{cases} \tag{3}$$

where $d_{\min}$ is the closest distance to failure achieved across the entire sequence of states in the reinforcement learning episode. This formulation encourages solutions to prioritize likelier failures, and amounts to a softened version of the original problem, since the constraint is replaced by a penalty.

In the original description of AST, the MDP is solved with the Monte Carlo tree search (MCTS) algorithm, which builds a tree of actions and recursively updates its value estimates as it explores different paths [6]. This algorithm has the advantage of anonymizing states: if an explicit representation of the environment is inaccessible but the random number generator of the simulation can be seeded, learning can still be performed over the space of seed sequences; the state s_t is effectively the history of seeds used up to step t.

2.1 Limitations

AST has historically achieved excellent results, as is clear from its use by system designers and researchers across governmental, industrial, and academic domains. However, the original framework has several important limitations, in part due to its output consisting of a set of failure traces (sequences of environment values that lead to SUT failure).

2.1.1 Modality. AST can find failures much more quickly than a basic Monte Carlo search. However, these failures are typically very similar and ultimately converge on the mode (or a local maximum) of the conditional probability distribution

$$p_{\mathcal{F}}(x) = p(x_0, \ldots, x_{T-1} \mid s_T \in \mathcal{F})\,. \tag{4}$$

The diversity of the output depends on the extent to which the particular learning algorithm explores while seeking the global optimum, as well as its capacity for storing suboptimal solutions; it is not a deliberate feature of the search. This is a result of the formulation of the MDP and is asymptotically independent of the

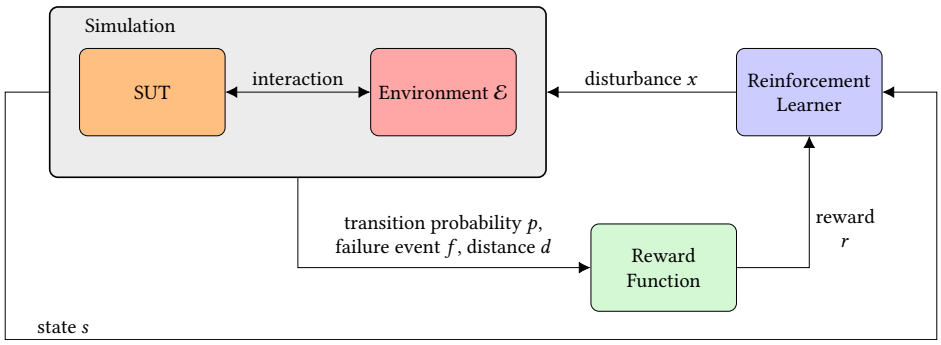

Figure 1: Adaptive stress testing architecture. A generic reinforcement learning agent chooses instances of a stochastic environment to elicit the likeliest possible failure in the system under test. The top-scoring failure traces are returned.

specific solution method: with enough training, the top k solutions will all be minor variations of the optimum.

2.1.2 Generality. The inherent unimodality of the framework motivates a paradigm of running AST multiple times in parallel from different initializations with the hope of landing in a different failure mode – in essence, conducting a Monte Carlo search of the initialization space. Since MCTS builds failures sequentially from an initial root, its output does not apply at any other starting point. This results in a large amount of computational waste, since information about the search space is not shared between processes. As a result, similar solutions cannot be learned in a manner that takes advantage of that similarity. Automated categorization techniques have shown potential in structuring a set of failure traces based on feature similarity [10]. While useful for facilitating human interpretation of AST results, this is a purely extrinsic approach that does not refer to the underlying model. Recent research has explored refining low-fidelity AST results with a backwards retraining scheme [9]. This approach represents a form of generalization, but its focus is on perfecting existing failure traces.

2.1.3 Statistics. AST cannot generate valid failure probabilities. For any given failure, the corresponding joint probability density value is calculable, but this is not a meaningful quantity in either absolute or relative terms – worse, such values can be easily misinterpreted as indicating exceptionally low risk for a system, especially when reported without context. Because a point evaluation of the probability density function does not reflect the shape of the surrounding mode and the failures do not have a strongly relational representation, the desire to account for probability mass from arbitrarily similar failures is unrealizable.

3 FAILURE TRACE RESAMPLING

Calculating failure statistics from conventional Monte Carlo sampling is straightforward, as the failure probability is estimated by the ratio of failures to the total number of samples. However, this approach cannot be used directly with AST, which generates failures in a deliberately nonuniform process. This can be rectified by using its output as the basis of an importance sampling scheme, which involves drawing from an alternative distribution and reweighting samples to correct for their missized contribution to the estimate. This approach makes it possible to calculate the likelihood of an entire failure mode along with a confidence interval.

Consider the AST trace $x^* = [x_1^*, x_2^*, \dots x_T^*] \in \mathcal{X}^T$, a high-likelihood environment sequence resulting in SUT failure[1]. Since the failure region may be arbitrary complex, it is defined implicitly by an indicator function $\mathbf{1}_{\mathcal{F}}(s) \in \{0, 1\}$. For a fixed initialization and environment sequence x, one can alternatively consider the indicator function $f(x) = \mathbf{1}_{\mathcal{F}}(s(x))$, where $s(x)$ is the state after the application of x. Let $X = [X_1, X_2, \dots X_T]$ be the random trace corresponding to a T-step "rollout" of the environment. Then, $f(X)$ is a binary random variable indicating whether or not a failure occurs during the trace. The overall failure prevalence is given by

$$\mu = P(f(X) = 1) = \mathbf{E}[f(X)] = \int_{\mathcal{X}^T} f(x)p(x)\,dx\,, \tag{5}$$

where $p(x) = \prod_{i=1}^{T} p(x_i)$ denotes the joint probability distribution of the trace and $dx = dx_1 \wedge \cdots \wedge dx_T$. In theory, this integral could be approximated via Monte Carlo integration, in which

$$\mathbf{E}[f(X)] \approx \frac{1}{n}\sum_{i=1}^{n} f(x^{(i)})\,, \tag{6}$$

where sample traces $x^{(i)}$ are drawn from X. However, when failures are sufficiently rare, i.e., $\mu \ll 1/n$, the estimate may be highly inaccurate or simply zero.

Instead, let $X^* \sim q(x)$ be a surrogate random variable that prioritizes x^* in some way and satisfies the coverage condition that $q(x) > 0$ wherever $p(x) > 0$ and $f(x) = 1$. The probability of failure can then be rewritten as

$$\mu = \int_{\mathcal{X}^T} f(x)\frac{p(x)}{q(x)}q(x)\,dx = \mathbf{E}\left[f(X^*)\frac{p(X^*)}{q(X^*)}\right]\,. \tag{7}$$

This expectation is performed over a distribution for which failures are by construction less rare; it is realized by the estimator

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} f(x^{(i)})\frac{p(x^{(i)})}{q(x^{(i)})}\,, \tag{8}$$

[1]The time index has been shifted to simplify the notation.

where sample traces $x^{(i)}$ are now drawn from X^*, i.e., rolled out from a surrogate environment. This quantity represents the probability of the failure mode in which x^* lies.

Surrogate construction

The success of the transformation depends heavily on the choice of surrogate. A desirable surrogate distribution emphasizes known failures without sacrificing the variance of the original distribution. The theoretically optimal surrogate $q^*(x)$ is exactly proportional to $f(x)p(x)$; this distribution minimizes the resulting estimation variance but is difficult to obtain in a usable form [17].

However, using Bayes' theorem, we can expand the conditional probability distribution maximized by AST as

$$\begin{aligned} p_{\mathcal{F}}(x) &= P(X = x \mid f(X) = 1) \\ &= P(f(X) = 1 \mid X = x)\frac{P(X = x)}{P(f(X) = 1)} \\ &= f(x)p(x)/\mu\,. \end{aligned} \tag{9}$$

Consequently, AST output converges to the statistical mode of a distribution that is equal to $q^*(x)$ up to a constant of proportionality. We can therefore use this output as the basis of an approximation of the optimal surrogate.

Since $p(x)$ may be arbitrarily complex or entirely blackboxed, there are limited options for $q(x)$ to be constructed generally and systematically. One reasonable method is to create a mixture model between the original distribution and a shifted variant. If X is continuous and unbounded, let $X^* = X - \mathbf{E}[X] + x^*$ be the random variable with elementwise probability distribution

$$q_{\text{sh}}(x_t) = p\left(x_t + \mathbf{E}[X_t] - x_t^*\right)\,. \tag{10}$$

This is a version of the original distribution shifted to center around the high-likelihood AST trace. It has the benefit of not requiring any knowledge of the functional form of $p(x)$, other than its expectation, which may be omitted if necessary (or if known to be zero). To sample X^*, one needs only to sample X and add the appropriate offset. If X is bounded in some way, the shifting approach will not work and a handpicked distribution such as a truncated normal may be required to satisfy the coverage condition. Note that $q_{\text{sh}}(x)$ cannot approximate the shape of $q^*(x)$, only match its mode.

The approach can be augmented by using a mixture model that samples from either $p(x)$ or $q_{\text{sh}}(x)$ at each step of the rollout, resulting in a distribution

$$q(x_t) = \epsilon p(x_t) + (1-\epsilon)p\left(x_t + \mathbf{E}[X_t] - x_t^*\right)\,. \tag{11}$$

This modification limits the estimation variance by bounding the ratio in the integrand. The full scheme is described in Algorithm 1, where $\mathbf{E}[X]$ is assumed to be zero.

Algorithm 1 Importance resampling of a failure trace

Input: s_0, x^*, ϵ, T, N
Output: $\hat{\mu}$

$\Sigma \leftarrow 0$ ▷ Accumulator
for $i = 1$ to N **do**
  $w \leftarrow 1$
  $s_t \leftarrow s_0$ ▷ Fixed initialization
  for $t = 1$ to T **do** ▷ Fixed horizon
    $x_t \leftarrow$ **Sample**$[p(x)]$
    $\alpha \leftarrow$ **Sample**$[\text{Unif}(0,1)]$
    if $\epsilon < \alpha$ **then** ▷ Mixture model
      $x_t \leftarrow x_t + x_t^*$
    end if
    $w \leftarrow wp(x_t)/(\epsilon p(x_t) + (1-\epsilon)p(x_t - x_t^*))$
    if not IsFailure(s_t) **then**
      $s_t \leftarrow$ **Step**(s_t, x_t)
    end if
  end for
  if IsFailure(s_t) **then**
    $\Sigma \leftarrow \Sigma + w$
  end if
end for
return Σ/N ▷ Sample mean

3.1 Error estimation

Naturally, estimation variance causes the estimate to differ from the true failure probability. This effect can be greatly exacerbated by the importance sampling scheme, which involves a potentially divergent ratio of probability densities. Nonetheless, error can be captured by probabilistic bounds.

Note that the estimator is theoretically unbiased, since its expectation is equal to the true probability. If $\hat{\mu}$ is the random variable corresponding to the estimated probability, then

$$\begin{aligned} \mathbf{E}[\hat{\mu}] &= \mathbf{E}\left[\frac{1}{n}\sum_{i=1}^{n} f(X^{(i)})\frac{p(X^{(i)})}{q(X^{(i)})}\right] \\ &= \mathbf{E}\left[f(X^*)\frac{p(X^*)}{q(X^*)}\right] = \mu\,, \end{aligned} \tag{12}$$

using the linearity of expectations and the fact that $X^{(i)} \sim X^*$ for all i. A similar analysis yields the variance of the estimation

$$\begin{aligned} \text{Var}[\hat{\mu}] &= \text{Var}\left[\frac{1}{n}\sum_{i=1}^{n} f(X^{(i)})\frac{p(X^{(i)})}{q(X^{(i)})}\right] \\ &= \frac{1}{n}\text{Var}\left[f(X^*)\frac{p(X^*)}{q(X^*)}\right]\,, \end{aligned} \tag{13}$$

using the scaling and linearity properties of variance. This value can itself be estimated as

$$\widehat{\sigma^2} = \frac{1}{n(n-1)}\sum_{i=1}^{n}\left[f(x^{(i)})\frac{p(x^{(i)})}{q(x^{(i)})} - \hat{\mu}\right]^2\,, \tag{14}$$

where the $n-1$ term is the standard bias correction. As the number of samples drawn from X^* increases, the distribution of the estimator $\hat{\mu}$ slowly approaches a normal distribution centered on the true failure probability μ with variance given by the above expression [17]. This yields the standard concentration bound

$$P(|\mu - \hat{\mu}| \geq \delta) \approx 2\left(1 - \Phi\left(\frac{\delta}{\sqrt{\widehat{\sigma^2}}}\right)\right), \tag{15}$$

equivalently expressed as the confidence interval

$$P\left(|\mu - \hat{\mu}| \geq \sqrt{\widehat{\sigma^2}}\ \Phi^{-1}\left(1 - \frac{\epsilon}{2}\right)\right) \approx \epsilon\,. \tag{16}$$

Since the importance sampling scheme may overly concentrate within a mode of failure, the estimate tends to underestimate the true probability. It may occasionally be useful to independently upper-bound the overall failure probability with a standard Monte Carlo computation, even if it fails to yield a single failure. Applying a Chernoff bound to the sampling process yields

$$P(\mu - \hat{\mu}_{\text{mc}} \geq \delta) \leq e^{-n\delta^2/2}\,. \tag{17}$$

Alternatively, an exact upper bound can be derived from Bayesian principles. If the sampling yields $\hat{\mu}_{\text{mc}} = n/k$, then

$$P(\mu - \hat{\mu}_{\text{mc}} \geq \delta) = I_{1-\hat{\mu}_{\text{mc}}-\delta}\left(n - k + \tfrac{1}{2}, k + \tfrac{1}{2}\right)\,, \tag{18}$$

where $I_x(a, b)$ is the regularized incomplete beta function. These bounds can be factored into the calculation of a refined confidence interval.

3.2 Multimodal failure trace resampling

Multiple failure traces generated by AST can be combined to form a composite surrogate distribution. For a set of failure trajectories $\{x_1^*, x_2^*, \ldots x_k^*\}$, we can define

$$q(x) = \frac{1}{k}\sum_{i=1}^{k} q_i(x)\,, \tag{19}$$

where the $q_i(x)$ are the corresponding surrogate distributions. An issue arises when some failure traces are shorter than others. For the interpretation to remain valid, shorter traces can be padded randomly at sample time.

The distribution $q(x)$ should not be sampled directly, since this would nullify the correlated nature of each failure mode. For example, distributions centered about failure trajectories $x_+^* = \mathbf{1}$ and $x_-^* = -\mathbf{1}$ must be sampled separately to produce trajectories that reach either failure mode; mixing them produces a zero-centered distribution.

To rectify this issue, $q(x)$ must be sampled as a mixture model: a distribution q_i is selected uniformly at random, then sampled. The full probability $q(x)$ is still used in the expectation estimate, as

$$\hat{\mu} = \frac{k}{n}\sum_{i=1}^{n} f(x^{(i)})\frac{p(x^{(i)})}{\sum_{j=1}^{k} q_j(x^{(i)})}\,. \tag{20}$$

4 FAILURE POLICIES

In the previous sections, it was assumed that all rollouts begin from the same initial state. This restriction allows a tree-based reinforcement learning algorithm such as MCTS to efficiently find paths to failure. However, if the system is sufficiently transparent, it is possible to formulate a much more powerful approach to the AST problem. Instead of finding a set of paths to failure, we learn an optimally adversarial policy π^* that maps a state to the likeliest environment value that induces a path to failure. Instead of an initial state s_0, we specify an initial distribution $p_0(s)$ with support $\mathcal{S}_0 \subseteq \mathcal{S}$, which is sampled at the start of each training episode.

To accommodate a more continuous setting, the AST reward function is modified as

$$r(s, x, s') = \log p(x \mid s) + \Delta(s, s') + r_f \cdot \mathbf{1}_{\mathcal{F}}(s)\,, \tag{21}$$

where r_f is a bonus for reaching failure and

$$\Delta(s, s') \propto d(s) - d(s') \tag{22}$$

is a reward shaping term to guide the learning agent more efficiently towards failure. Since the term represents a conservative potential, i.e., the gradient of a scalar function of the MDP state, its addition to the reward function is policy-invariant [16]. A wide variety of reinforcement learning algorithms can be used to solve this MDP. Deep reinforcement learning offers an attractive option when state and environment spaces are high-dimensional. Due to the ability of neural networks to interpolate and generalize, this approach allows failure paths to be approximated between samples.

Proposition 1. *With sufficient training, the policy π^* is weakly guaranteed to capture all failure modes $\mathcal{F}_k$ for which $\mathcal{F}_k \cap \mathcal{S}_0$ is non-empty.*

Proof. Since the AST framework is only concerned with the likeliest failures of a system, this bias is reflected in the failure policy. As long as a failure region $\mathcal{F}_k$ intersects with $\mathcal{S}_0$, there must exist a region of initialization $\mathcal{F}_k \subseteq \mathcal{S}_k \subseteq \mathcal{S}_0$ from which it is the likeliest failure. Since all such regions will be sampled as $n \to \infty$ and the optimal policy π^* by construction produces the most likely path to failure, then π^* must represent all F_k, provided its underlying representation has sufficient expressive capacity[2]. The universal approximation theorem establishes that any degree of expressiveness can be achieved by a neural network of sufficient width and depth; similar guarantees can be formulated for a table-based policy. Furthermore, asymptotic convergence to the optimal policy is provable for certain well-known algorithms [4, 21]. □

In practice, the policy representation and the training set are of finite size, so optimality arguments are largely theoretical. However, they underscore the fact that multiple independent failure modes can be discovered and latently represented in a failure policy. This opens the door to analyzing the learned policy, which can be used for its generative properties: randomly sampling the initialization space and rolling out π^* efficiently produces a stream of unique failure traces.

5 FAILURE POLICY RESAMPLING

The importance sampling scheme developed for traces can be extended naturally to failure policies. At each step, instead of considering perturbations around an optimal environment x_t^*, we now consider perturbations around $\pi^*(s_t)$, the output of the optimal policy. To formalize this difference, we consider a surrogate random policy Π^*. This may be accomplished in same manner as before, through shifting or manual selection. In the case of a shifted surrogate, the policy evaluation $\Pi^*(s)$ represents a distribution over environments that behaves according to the mixture model

$$q_s(x_t) = \epsilon p(x_t) + (1 - \epsilon)p\left(x_t + \mathbf{E}[X_t] - \pi^*(s)\right)\,. \tag{23}$$

[2]It should be noted that even an optimal policy cannot necessarily capture all possible failures. Each initialization is associated with a single failure region: if from point s_0 the system admits failures $\mathcal{F}_1$ and $\mathcal{F}_2$, the policy will capture only the likelier outcome.

Then, for a given initial state s_0 and horizon T, the resulting rollout is represented by the length-T random vector $X^*_{s_0}$ with components

$$X^*_{s_0,t} = \Pi^*(s_t)\,, \tag{24}$$

and the corresponding joint probability is

$$q(x^*_{s_0}) = \prod_{t=1}^{T} q(x^*_{s_0,t})\,. \tag{25}$$

The expected probability of failure from an initial state is thus

$$\mu(s_0) = \mathbf{E}_x\left[f(X^*_{s_0})\frac{p(X^*_{s_0})}{q(X^*_{s_0})}\right]. \tag{26}$$

Finally, the overall failure probability can be written as

$$\mu = \mathbf{E}\left[\mu(S_0)\right] = \mathbf{E}_{s_0,x}\left[f(X^*_{S_0})\frac{p(X^*_{S_0})}{q(X^*_{S_0})}\right], \tag{27}$$

where the subscripts indicate a joint expectation over initialization and environment spaces, respectively. The expectation is realized with the estimate

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} f\left(x^{(i)}_{s_0}\right)\frac{p\left(x^{(i)}_{s_0}\right)}{q\left(x^{(i)}_{s_0}\right)}, \tag{28}$$

where each initial state s_0 is drawn from S_0 and the subsequent rollout x_{s_0} is drawn from $X^*_{s_0}$, applying the random policy step by step. The principle of deferred decisions ensures that this sequential sampling is equivalent to randomly sampling the entire trace at once [14]. As in the previous section, it is important not to terminate rollouts if an error occurs prematurely; the rollout should be padded to the horizon with actions sampled from $X^*_{s_0}$. The full algorithm is described in Algorithm 2, again assuming zero-mean environment variables.

The variance analysis is unchanged in the policy setting, so the confidence interval remains valid; however, the potential for high sample variance grows significantly. For an general Monte Carlo computation, the standard deviation of the estimate is equal to $\sigma/\sqrt{n}$, where σ is the true standard deviation of the quantity of interest. Since we are now considering a product sample space, σ is greatly increased and it is important to increase the sample size n accordingly. The estimate could also be performed with a double loop, making explicit the need to cover both spaces sufficiently. Luckily, as with most Monte Carlo schemes, the computation lends itself naturally to multiprocessing optimizations.

Algorithm 2 Importance resampling of a failure policy

Input: S_0, π^*, ϵ, T, N
Output: $\hat{\mu}$

$\Sigma \leftarrow 0$ ▷ Accumulator
for $i = 1$ to N **do**
  $w \leftarrow 1$
  $s_t \leftarrow$ **Sample**$[S_0]$ ▷ Random initialization
  for $t = 1$ to T **do** ▷ Fixed horizon
    $x_t \leftarrow$ **Sample**$[p(x)]$
    $x^*_t \leftarrow \pi^*(s_t)$ ▷ Policy query
    $\alpha \leftarrow$ **Sample**$[\mathrm{Unif}(0,1)]$
    if $\epsilon < \alpha$ **then** ▷ Mixture model
      $x_t \leftarrow x_t + x^*_t$
    end if
    $w \leftarrow wp(x_t)/(\epsilon p(x_t) + (1-\epsilon)p(x_t - x^*_t))$
    if not IsFailure(s_t) **then**
      $s_t \leftarrow$ **Step**(s_t, x_t)
    end if
  end for
  if IsFailure(s_t) **then**
    $\Sigma \leftarrow \Sigma + w$
  end if
end for
return Σ/N ▷ Sample mean

6 EXPERIMENTAL RESULTS

To demonstrate the fundamental abilities of the extended framework, we consider a simple toy problem in which a small aircraft autonomously lands on a runway in gusty conditions. Episodes are initialized with the aircraft approaching the runway and deviating slightly from the center-line: $x_1 = 0$ and $x_2 \sim \mathcal{N}(0, \sigma_i^2)$. Episodes terminate when the aircraft reaches the start of the runway ($x_1 > a$). The environment consists of stochastic cross-track transitions $\Delta x_2 \sim \mathcal{N}(0, \sigma_t^2)$, and failure occurs if the aircraft lands too far from the center-line, a region defined as

$$\mathcal{F} = \{x_1, x_2 \in \mathbf{R} \mid x_1 > a, |x_2| > b\}\,. \tag{29}$$

The system is visualized in Figure 2, where failure zones are shown in hatched red and non-failure terminating zones in green. Though the behavior of the SUT is completely trivial (it takes no actions to stabilize the trajectory), the toy problem is useful as it exhibits two distinct modes of failure with closed-form likelihoods. This allows the basic correctness of the framework to be tested against an analytical result: for the system shown in Figure 2, the overall probability of failure is

$$\mu = 2\left(1 - \Phi\left(\frac{b}{\sqrt{\sigma_i^2 + \lceil a/v \rceil \sigma_t^2}}\right)\right) \approx 4.023 \times 10^{-13}\,. \tag{30}$$

The low probability rules out a direct Monte Carlo approach, since each failure would on average require simulating over 10^{12} episodes; a meaningful estimate of the probability would require at least another order of magnitude.

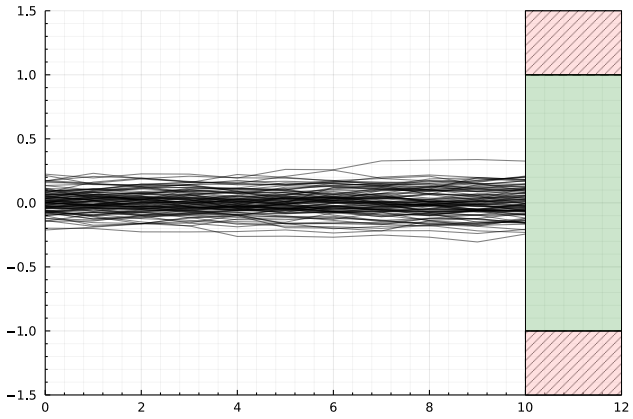

Figure 2: 100 random trajectories of the example system, with zero recorded failures. Parameters are $a = 10$, $b = 1$, $\sigma_i = 0.1$, $\sigma_t = 0.03$, and $v = 1$; the effective horizon is 10. Although the effect of the stochastic environment is visible, a sustained disturbance would be required to elicit failure.

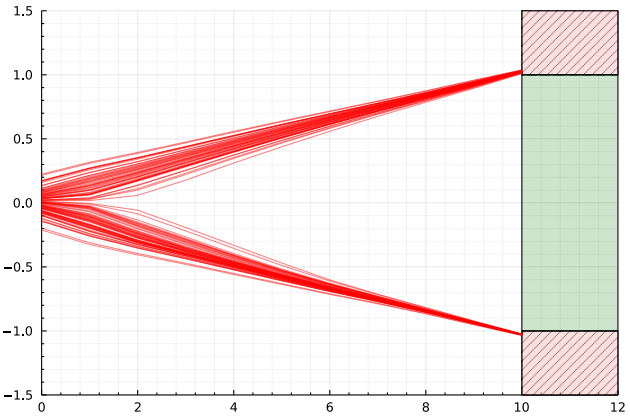

Figure 3: 100 random evaluations of a failure policy learned with SAC. Actions are instances of the environment that maximize likelihood while eliciting failure: these trajectories are approximately representative of the likeliest failures. Note that the policy encodes the location of both failure modes in its "instructions" for selecting adversarial environments. In the importance sampling scheme, this latent representation is extracted and used to calculate failure statistics.

We solve the MDP with the soft actor-critic algorithm[3], yielding the policy shown in Figure 3. The policy and value networks both contain two hidden layers of size 100. Training was performed for $2.5{\times}10^4$ episodes with a learning rate of 10^{-4}. The failure policy was then passed into the analysis stage, which resampled 10^7 additional episodes with an $\epsilon = 0.05$ mixture model. The estimated failure probability is approximately 3.253×10^{-13}, 19% less than the true probability, with an estimate standard deviation of 1.378×10^{-13}.

It should be noted that this standard deviation is quite high, given the scale of the estimate. The $\pm 3\sigma$ neighborhood encompasses all values from 0 to 7.386×10^{-13}, implying that there is a nearly 50% chance that the failure probability is arbitrarily low. This is a consequence of the fundamentally reciprocal nature of probability, which is not accounted for by this form of estimation but is crucial to how probabilities are used and understood. This issue is described further in the next section; the most immediate solution would be simply to increase the sample size.

7 DISCUSSION AND FUTURE WORK

We have derived and implemented two significant enhancements to adaptive stress testing of complex systems: (1) the ability to learn failure policies that generate the likeliest sequence of environments leading to system failure from arbitrary initializations and (2) the use of these policies in importance sampling to generate statistics that show how likely these failure modes are to occur naturally. These improvements make AST a more powerful tool for failure generation and analysis.

At a high level, these benefits are due to the fact that the model is dissociated during the learning phase (i.e., used descriptively but not generatively) and reassociated in a separate analysis phase. As a result of this separation, there are very few constraints on the solving method: the analysis can proceed from any solution that produces a valid policy. The flexibility also extends to the analysis, which can in theory admit *any* model. The importance sampling scheme can just as easily be used to retroactively analyze hypothetical changes to the SUT and environment models, but this use case is not explored in depth here.

It should be noted, however, that the approach described in this paper has a number of weaknesses. Although it has the potential to assess failure mode probability far more efficiently than by random sampling, the accuracy of the estimate is dependent on the success of the learning phase. Partially learned solutions consistently yield underestimates, which may only lower-bound the true failure probability. As such, the learning process must be monitored for convergence regardless of the subsequent sampling.

Additionally, importance sampling is, as a rule, extremely sensitive to the choice of surrogate distribution; a poor selection can yield unacceptably high variance. For failure modes spanning a higher number of time steps, the optimal surrogate $q^*(x)$ is harder to systematically approximate with the shifting method. As a result, the quality of the estimation tends to suffer in long-horizon analyses.

The range of the output can also pose challenges: though the construction of the estimate ensures its non-negativity, it is not strictly guaranteed to be a valid probability in the range $[0, 1]$. Since estimated probabilities are typically very low, the variance estimate may not accuracy represent the uncertainty due to the inherent skew of the distribution; this effect can be seen in the previous example.

These various issues can compound to produce a situation where the upper confidence limit is inaccurate because of underestimation caused by poor learning or surrogate selection, while the lower

[3]Experiments with the simpler Q-learning algorithm yielded a nearly identical solution; these results are omitted for brevity.

confidence limit is not particularly useful because the spread encompasses too many orders of magnitude between the estimate mean and zero. In such cases, the lower limit can be improved with more sampling while the upper limit can be taken from the alternative bounds described earlier.

Many of these issues may be resolved by forming the estimate directly in log-probability space and sampling the optimal surrogate exactly via Monte Carlo Markov chain (MCMC) methods. This is the subject of ongoing research.

8 CONCLUSION

Adaptive stress testing can be an useful component in the validation of complex stochastic systems. We have addressed several limitations of the standard adaptive stress testing formulation. By generalizing AST results and learning failure policies, we gain the ability to query the likeliest path to failure from any initial state. This enables procedural generation of failures without additional training, which may be useful for diagnostics or runtime assurance tools.

We also show how the learned failure policy can form the basis of an importance sampling scheme to calculate the probabilities of entire failure modes along with probabilistic bounds. We demonstrate that for rare failures, AST with resampling vastly outperforms the standard Monte Carlo method and offers a degree of validative independence that an expert-guided search might lack.

ACKNOWLEDGMENTS

This work is supported by the Systems-Wide Safety (SWS) Project under the NASA Aeronautics Research Mission Directorate (ARMD) Airspace Operations and Safety Program (AOSP).

REFERENCES

[1] Anthony Corso, Peter Du, Katherine Driggs-Campbell, and Mykel J Kochenderfer. 2019. Adaptive stress testing with reward augmentation for autonomous vehicle validation. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, Auckland, 163–168.

[2] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. 2019. Adversarial policies: attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615* (2019).

[3] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[4] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*. PMLR, Stockholm, 1861–1870.

[5] Kyle D. Julian, Ritchie Lee, and Mykel J. Kochenderfer. 2020. Validation of Image-Based Neural Network Controllers Through Adaptive Stress Testing. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Rhodes, 1–7.

[6] Mykel J. Kochenderfer. 2015. *Decision Making Under Uncertainty: Theory and Application*. MIT Press, Cambridge.

[7] Mark Koren, Saud Alsaif, Ritchie Lee, and Mykel J. Kochenderfer. 2018. Adaptive Stress Testing for Autonomous Vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, Changshu, 1–7.

[8] Mark Koren and Mykel J Kochenderfer. 2020. Adaptive stress testing without domain heuristics using go-explore. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Rhodes, 1–6.

[9] Mark Koren, Ahmed Nassar, and Mykel J Kochenderfer. 2021. Finding failures in high-fidelity simulation using adaptive stress testing and the backward algorithm. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Prague, 5944–5949.

[10] Ritchie Lee, Mykel J. Kochenderfer, Ole J. Mengshoel, and Joshua Silbermann. 2018. Interpretable Categorization of Heterogeneous Time Series Data. In *International Conference on Data Mining (SDM)*. SIAM, San Diego, 216–224.

[11] Ritchie Lee, Ole J. Mengshoel, Adrian K. Agogino, Dimitra Giannakopoulou, and Mykel J. Kochenderfer. 2019. Adaptive Stress Testing of Trajectory Planning Systems. In *AIAA SciTech, Intelligent Systems Conference (IS)*. AIAA, San Diego, 1454.

[12] Ritchie Lee, Ole J. Mengshoel, Anshu Saksena, Ryan Gardner, Daniel Genin, Joshua Silbermann, Michael Owen, and Mykel J. Kochenderfer. 2020. Adaptive Stress Testing: Finding Likely Failure Events with Reinforcement Learning. *Journal of Artificial Intelligence Research* 69 (2020), 1165–1201.

[13] Rory Lipkis, Ritchie Lee, Joshua Silbermann, and Tyler Young. 2022. Adaptive Stress Testing of Collision Avoidance Systems for Small UASs with Deep Reinforcement Learning. In *AIAA SciTech 2022 Forum*. AIAA, San Diego, 1854.

[14] Michael Mitzenmacher and Eli Upfal. 2017. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge University Press, Cambridge.

[15] Robert J. Moss, Ritchie Lee, and Mykel J. Kochenderfer. 2020. Adaptive Stress Testing of Trajectory Predictions in Flight Management Systems. In *IEEE/AIAA Digital Avionics Systems Conference (DASC)*. IEEE, San Antonio, 1–10.

[16] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, Vol. 99. ICML, Bled, 278–287.

[17] Art B. Owen. 2013. *Monte Carlo theory, methods and examples*. Preprint, online.

[18] Shreyas Ramakrishna, Baiting Luo, Christopher B Kuhn, Gabor Karsai, and Abhishek Dubey. 2022. ANTI-CARLA: An Adversarial Testing Framework for Autonomous Vehicles in CARLA. In *25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Macau, 2620–2627.

[19] Aizaz Sharif and Dusica Marijan. 2021. Adversarial Deep Reinforcement Learning for Trustworthy Autonomous Driving Policies. *arXiv preprint arXiv:2112.11937* (2021).

[20] Cumhur Erkan Tuncali, Georgios Fainekos, Hisahiro Ito, and James Kapinski. 2018. Simulation-based adversarial test generation for autonomous vehicles with machine learning components. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, Changshu, 1555–1562.

[21] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3 (1992), 279–292.