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

Rory Lipkis Adrian Agogino

NASA Ames Research Center
Intelligent Systems Division

May 29, 2023

Motivation: verification

- Formal verification
  - Difficult due to system size and complexity
  - Often requires making strong assumptions about system dynamics
  - Produces mathematical guarantees
- Sample-based approach
  - One-sided "proof" (falsifiability)
  - Requires simulation
  - Contingencies are explored in proportion to their likelihood
  - Expensive at scale
  - Often requires domain knowledge to inform exploration
  - Scenario "engineering" can violate principles of IV&V

System formulation

- System under test (SUT) embedded in a simulation, where it responds to a semi-stochastic environment
  - State space: $s \in \mathcal{S}$
  - Environment: set of stochastic disturbances $\mathcal{X} \sim p(x)$
  - Failure criterion: $s \in \mathcal{F} \subset \mathcal{S}$
- For particular state trajectory $\{s_0, s_1, \ldots, s_T\}$,

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

Failure probability

- Failure indicator function:

$$f(\mathbf{x}) = \mathbf{1}_{\mathcal{F}}(s(\mathbf{x}))$$

- Let $X = [X_1, X_2, \ldots X_T] \sim p(\mathbf{x})$ be the random trace corresponding to a T-step "rollout" of the environment
- Failure probability is given by

$$\mu = P(f(X) = 1) = \mathbf{E}[f(X)]$$

Monte Carlo estimation

- Estimated failure probability:

$$\hat{\mu}_{\text{MC}} = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}^{(i)})$$

  where $\mathbf{x}^{(i)} \sim p(\mathbf{x})$.
- When failures are rare, direct Monte Carlo is
  - Inefficient
  - Inaccurate
- If process is accelerated, probabilities may be misleading

Importance sampling estimation

- Estimated failure probability:

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

  where $\mathbf{x}^{(i)} \sim q(\mathbf{x})$.
- Surrogate distribution $q(\mathbf{x})$ skews the environment towards learned failure modes

Surrogate distribution

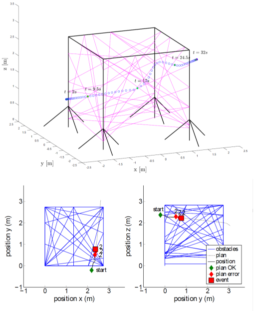
- A desirable surrogate environment
  - Reproduces learned failures
  - Preserves variance of original distribution
- Variance of the estimate is minimized when

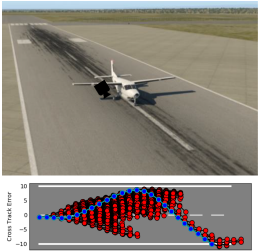
$$q(\mathbf{x}) \propto f(\mathbf{x})p(\mathbf{x})$$

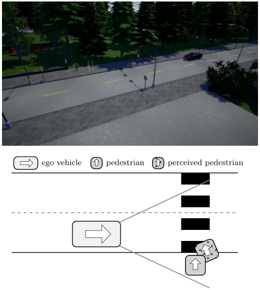
- Consider failure-conditioned distribution

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

- Accelerated validation framework
- Requires little knowledge of system under test
- Generates adversarial environments to find **likeliest failures**
- Better performance at scale
- Flexible and general

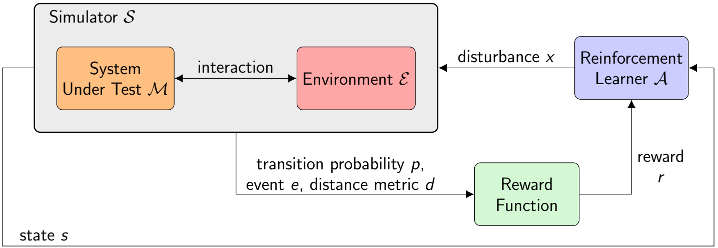






AST overview

- Simulated system under test
- Stochastic system environment (set of disturbances)
- Formulates stress-testing problem as MDP or POMDP
- Agent chooses actions to optimize overall marginal likelihood with the constraint of eventual system failure
  - Solutions correspond to the *mode* of $p_{\mathcal{F}}(\mathbf{x})$



AST formulation

- Optimization:

$$\max_{\mathbf{x}_0,\ldots,\mathbf{x}_{T-1}} \quad \sum_{t=0}^{T-1} \log p(\mathbf{x}_t)$$
$$\text{subject to} \quad s_T \in \mathcal{F}$$

- State and environment spaces are continuous and high-dimensional
- Environment selection is parameterized as a policy
    - Probabilistic decision tree
    - Q-table
    - (Deep) neural network
- Failure constraint is enforced via penalty
- Reward is given by

$$r(s_t, x_t) = \log p(x_t) - \Delta d_F + R_F \cdot \{s_t \in \mathcal{F}\}$$

Policy sampling

- AST produces optimal failure policy $\pi^*(s)$
  - Multiple modes are latently represented in policy
- Candidate surrogate is given by

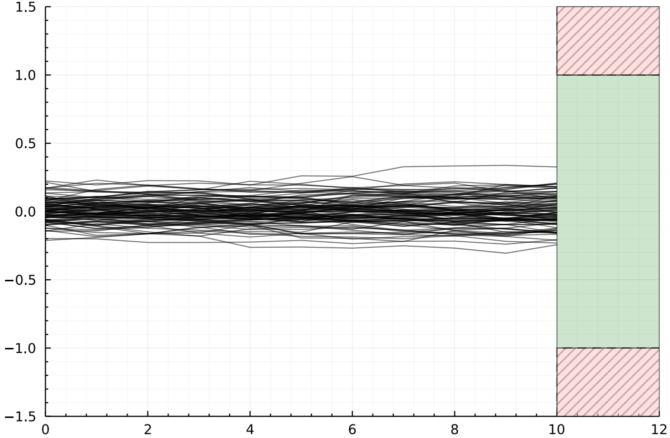
$$q(x_t) = \epsilon p(x_t) + (1-\epsilon) p\left(x_t + \mathbf{E}[X_t] - \pi^*(s)\right)$$

- Failure probability:

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

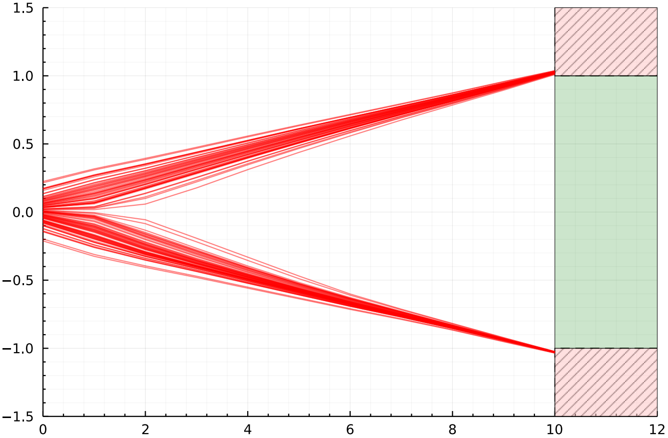
where $\mathbf{x}^{(i)} \sim q(\mathbf{x}_t)$.

Example: runway (random rollout)



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

Example: runway (failure policy)



$$N_{\text{train}} = 2.5 \times 10^4 \qquad N_{\text{sample}} = 10^7$$

$$\hat{\mu} \approx 3.253 \times 10^{-13} \qquad \hat{\sigma}_\mu \approx 1.378 \times 10^{-13}$$

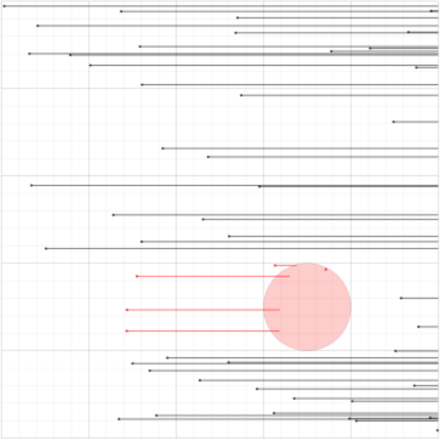
Later work

- For any initial state, rolling out the policy produces trajectories around the mode of the *conditional probability distribution*

$$p_{\mathcal{F}}(\mathbf{x}) = p(x_0, \ldots, x_{T-1} \mid s_T \in \mathcal{F}) \propto f(\mathbf{x})p(\mathbf{x})$$

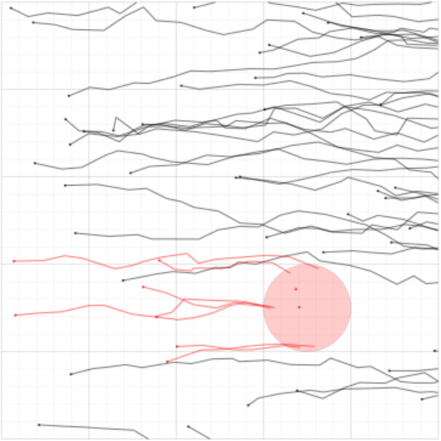
- We can use this policy as the “kernel” of a MCMC process
  - Distribution is resampled in its region of highest probability mass
  - Strongly accelerates convergence / mitigates “burn-in”
  - Allows immediate generation of independent failure traces
- Added benefits:
  - Estimates are formed directly in log-space
  - Importance sampling instability is circumvented

Example: sampling without disturbances



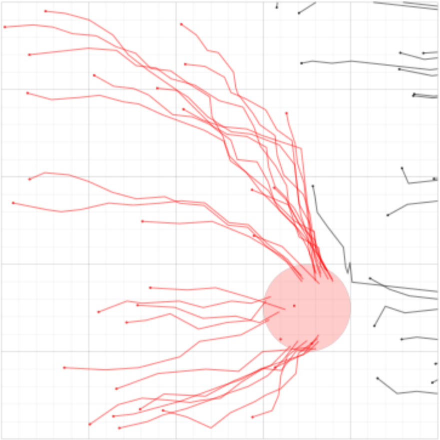
Natural system behavior moves state to right at constant rate

Example: sampling with disturbances (Monte Carlo)



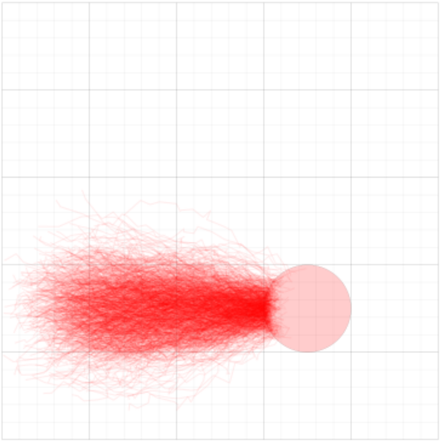
State experiences stochastic perturbations

Example: sampling failure policy



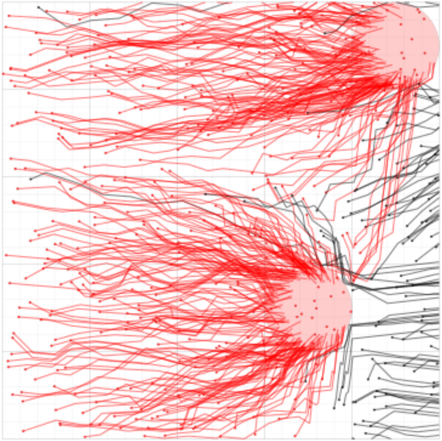
AST finds failures corresponding to a set likelihood threshold

Example: MCMC sampling failure policy



Learned policy becomes basis of statistical model

Example: multimodal capture



Information about failure modes is stored implicitly in policy

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).

Case study: validation of ACAS sXu

- Airborne Collision Avoidance System (ACAS X)
  - Replaces Traffic Alert and Collision Avoidance System (TCAS)
  - Models aircraft encounters as POMDPs
  - Stores precomputed solution as large lookup table
  - Detects potential collisions between individual aircraft
  - Issues directive guidance in the form of resolution advisories (RAs)
- Variants:
  - Xa (commercial aircraft)
  - Xo (specialized missions)
  - Xu (unmanned aircraft)
  - **sXu** (small unmanned aircraft)

Research goal

Provide ACAS sXu development team with stress-testing tools and infrastructure to inform their ongoing work

Testing setup

- **System under test:** ACAS sXu binary
- **Disturbances:** pilot commands
  - Turning rate
  - Vertical rate
  - Forward acceleration
- **Aircraft dynamics:** simple multirotor with limited acceleration ($g/2$)
- **Response to CAS:** pilot compliance with 1-second delay
  - Turning rate of $3^\circ/s$ complying with advised horizontal maneuvers
  - Acceleration of $g/4$ to $g/3$ complying with advised vertical maneuvers
- **Failure criterion:** small near mid-air collision (sNMAC)
  - 50 ft. horizontal separation
  - 15 ft. vertical separation

Result

Failure policy achieved 97% failure elicitation rate

Example: freeze failure