Debugging a Policy: A Framework for Automatic Action Policy Testing

Marcel Steinmetz, Timo P. Gros, Philippe Heim, Daniel Höller, Jörg Hoffmann

Saarland University, Saarland Informatics Campus
Saarbrücken, Germany
{steinmetz, timopgros, hoeller, hoffmann}@cs.uni-saarland.de, s8phheim@stud.uni-saarland.de

DNNs Are Taking Over Control!

How to gain trust in the policy's learned decisions? Here: Testing!

The Assumptions

- Deterministic action policies π mapping **states** s to **actions** $\pi(s)$.
- **Policy value function** V^π mapping every state s to the objective value achieved by π on s.
  For qualitative objectives:
  $$V^\pi(s) := \begin{cases} 0 & \text{no run of } \pi \text{ on } s \text{ satisfies the objective} \\ 0.5 & \text{some run of } \pi \text{ on } s \text{ satisfies the objective} \\ 1 & \text{all runs of } \pi \text{ on } s \text{ satisfy the objective} \end{cases}$$
- **Optimal value function** V^* mapping every state s to best value any policy can achieve on s.
- Generic **better than** notion: $V(s) \prec V(s')$ iff
  $$\begin{cases} V(s) < V(s') & \text{objective is minimization} \\ V(s) > V(s') & \text{objective is maximization} \end{cases}$$

Policy Bugs: Examples

- **Classical planning**: the cost of running the policy on s exceeds that an optimal plan by Δ.
- **Oversubscription planning** and **discounted-reward MDPs**: running the policy on s achieves Δ less reward than possible.
- **MaxProb MDPs**: running the policy on s reaches the goal with Δ less probability than possible.

In qualitative setting:

- **Classical planning**: Δ must be 1
  $\Rightarrow$ s is solvable but π does not reach the goal from s
- **Contingent planning**: goal can be reached with certainty, but
  $\Delta = 0.5$: policy only reaches the goal in some cases.
  $\Delta = 1$: policy does not reach the goal at all.

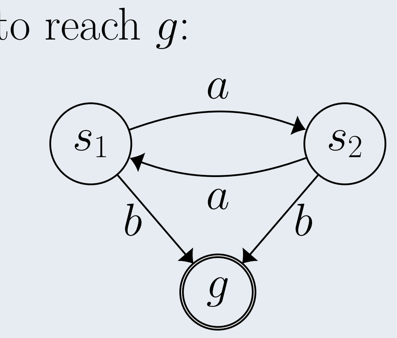
Policy Bugs: Definition

Definition (Bug)

A state s is a **bug in** π if
$$\Delta := |V^\pi(s) - V^*(s)| > 0$$

$\Rightarrow$ π is bug-free iff π is optimal

Note: π can have bugs despite each $\pi(s)$ being optimal individually. Consider $\pi(s_1) = \pi(s_2) = a$ with the (qualitative) objective to reach g:



But, how can we actually

→ **confirm** whether a given state s is a bug?
→ **generate** suitable candidates for testing?

Bug Confirmation: The Easy Cases

- V^* can be computed for all states offline, before starting testing / deploying the policy.
- Domain-specific knowledge: user provides V^* for certain states as input, can use these states for testing.
- Specifically design test-case generation methods, generating only states s with a given $V^*(s)$ value. Example: in qualitative setting, generate states via backward samples from the goal.
- $V^*(s)$ may be efficiently computable for certain states s, e.g., due to structural properties of the domain, or if a fixed-depth lookahead search is sufficient.

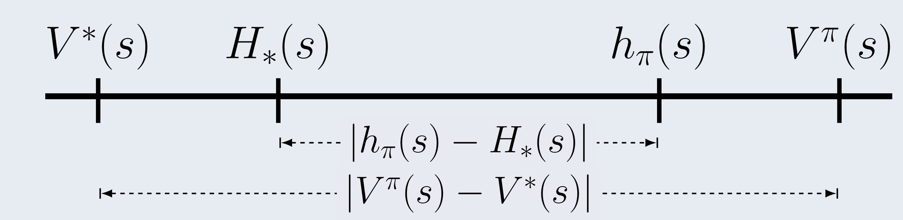
Bug Confirmation: The General Case

If V^* cannot be computed exactly, then **approximate**!

- Optimistic approximation of V^π: $h_\pi(s) \preceq V^\pi(s)$
- Pessimistic approximation of V^*: $H_*(s) \succeq V^*(s)$

Proposition 4 (Bug Confirmation):

If (i) $h_\pi(s) \succeq V^*(s)$ and (ii) $H_* \preceq V^\pi(s)$, then $|h_\pi(s) - H_*(s)| \leq |V^\pi(s) - V^*(s)|$.



Test-Case Generation: Fuzzing

Idea:

- Start from some state s.
- Mutate this state while enlarging the optimality gap.

Definition (Fuzzing-Bug):

A state s' is a **fuzzing-bug** relative to s if
$$\Delta := |V^\pi(s') - V^*(s')| - |V^\pi(s) - V^*(s)| > 0$$

⇛ If s' is a fuzzing-bug relative to some s, then s' is a bug.
⇛ Every bug s' with non-minimal optimality gap is a fuzzing-bug relative to some s.

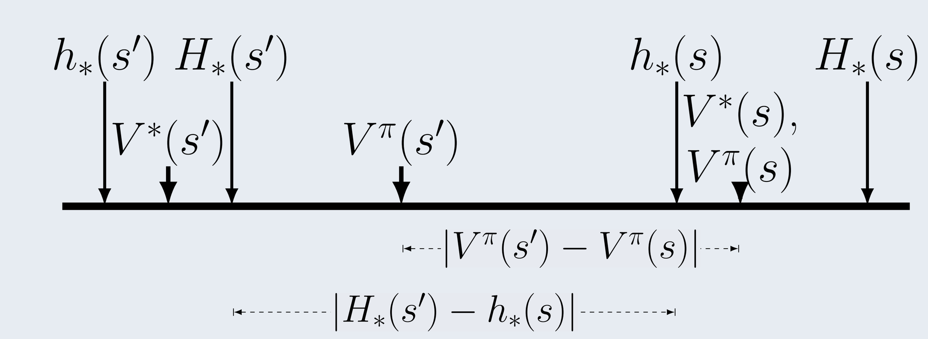
Fuzzing-Bug Confirmation

How to decide whether s' is a fuzzing-bug relative to s, without knowing the exact gap values?

- Optimistic approximation of V^*: $h_*(s) \preceq V^*(s)$
- Pessimistic approximation of V^*: $H_*(s) \succeq V^*(s)$

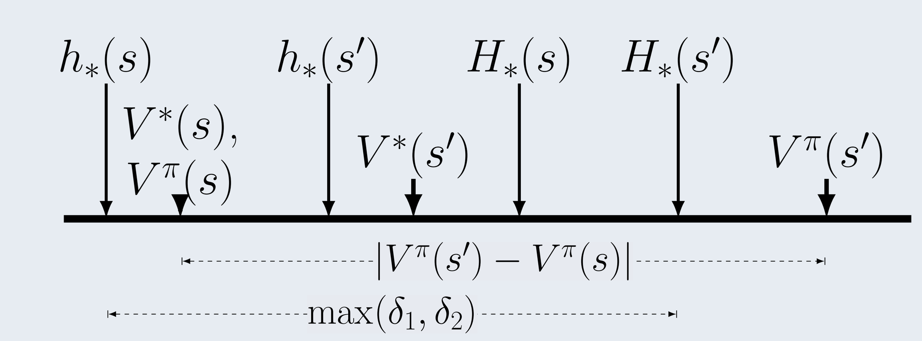
Proposition 9 (Fuzzing-Bug Confirm. I):

s' is a fuzzing-bug relative to s if $H_*(s') \prec h_*(s)$ and either $V^\pi(s') \succeq V^\pi(s)$ or $|V^\pi(s') - V^\pi(s)| < |H_*(s') - h_*(s)|$.



Proposition 10 (Fuzzing-Bug Confirm. II):

Let $\delta_1 := |H_*(s') - h_*(s)|$, $\delta_2 := |h_*(s') - H_*(s)|$. s' is a fuzzing-bug relative to s if $V^\pi(s') \succeq V^\pi(s)$ and $|V^\pi(s') - V^\pi(s)| > \max(\delta_1, \delta_2)$.



Bug Confirmation: Comparison & Takeaway

Theorem

Let s and s' be arbitrary states. If either Proposition 9 or Proposition 10 confirms s' to be a fuzzing-bug relative to s, then Proposition 4 confirms s' to be a bug.

⇝ All boil down to:

1. Approximate $V^\pi(s)$ up to a high degree of precision.
2. Then try to find a better policy for s.

Conclusion & Outlook

Bug confirmation:

- Many special cases where $V^*(s)$ feasible to compute.
- If not, fall back to well-known policy-improvement algorithms.

Let's debug your policies!

- Develop fuzzing methods (start states, fuzzing operators, biases, termination, . . .)
- Adapt other techniques from software testing (e.g., metamorphic testing).
- See what it does in your favorite planning & learning scenarios.