DSAIT 4015 2025-2026
Project assignment 3

Scenario-Based Testing of Reinforcement Learning Agents in Autonomous Driving Simulation

Instructors: Annibale Panichella and Cynthia Liem

Background

Reinforcement learning (RL) agents are increasingly used in safety-critical domains such as autonomous driving, robotics, and control systems. Unlike traditional software, RL agents learn their behavior from interaction with an environment, which makes their internal decision logic difficult to interpret and test.

In this assignment, we focus on testing an RL-based driving agent trained using **Proximal Policy Optimization (PPO)** in the highway-fast-v0 environment from the sable-baseline3[1]. The agent was trained by us in advance and is provided to you as a **pre-trained black-box policy**. You must not modify the policy itself.

Although the agent performs well on average, this does not guarantee that it behaves safely in all possible traffic situations. In particular, rare or adversarial configurations of the environment may cause unsafe behavior, such as collisions with other vehicles.

From a software testing perspective, the **code under test** is the trained RL policy together with its environment, while the **test inputs** are concrete traffic scenarios (e.g., number of vehicles, spacing, lane configuration). A test **fails** when the execution of a scenario leads to a collision.

This assignment applies **search-based software testing (SBST)** techniques to systematically generate such failing scenarios. You will design a **Hill-Climbing (HC)** search algorithm that explores the space of environment configurations with the goal of bringing the ego vehicle as close as possible to other vehicles and ideally triggering a collision.

To assess the effectiveness of your approach, you will compare it against a provided **Random Search baseline**, which samples scenarios uniformly at random without using feedback from previous executions.

Overall purpose of the assignment

In this assignment, you will:

1. **Implement a Hill-Climbing search algorithm** to generate challenging driving scenarios for a trained PPO agent operating in the highway-fast-v0 environment.
2. **Design a fitness (objective) function** that measures how close the agent is to failure (e.g., collision or minimum distance to other vehicles).

[1] https://stable-baselines3.readthedocs.io/en/master/

3. **Design mutation operators** that generate neighboring traffic scenarios by modifying environment parameters.
4. **Evaluate and compare** your Hill Climber against a Random Search baseline in terms of effectiveness and efficiency.
5. **Analyze and reflect** on the types of failures (or near-failures) discovered by your approach.

What We Provide to You

To support your work, we provide a complete Python framework that includes:

- A **pre-trained PPO agent** for the highway-fast-v0 environment.
- A **Random Search baseline** that samples traffic scenarios uniformly at random.
- Infrastructure for:
  - Generating traffic scenarios,
  - Executing simulations,
  - Collecting time-series data (positions, distances, collisions),
  - Recording videos of failures.

You are expected to **read the README.md** in the provided repository carefully before starting the assignment.

Like in Assignment 2, your entire group will work together as a single team (i.e., no sub-groups). All steps of this assignment (i.e., implementation, experimentation, and reporting) must be completed collaboratively.

1. Implement the Hill Climber

Your Hill Climber must be able to:

1. Generate or load traffic scenarios by instantiating environment configurations (e.g., number of vehicles, spacing, initial lane).
2. Execute each scenario by running the environment with the provided PPO policy.
3. Collect execution outcomes, including:
   - Whether a collision occurred,
   - The minimum distance between the ego vehicle and other vehicles over time,
   - Any additional diagnostic information needed to evaluate the scenario.

The evaluation must rely **only on observable behavior during execution** (i.e., environment states and outcomes), and **not on internal agent information**.

To support this, the shared repository already provides routines to:

- Generate random traffic scenarios,
- Execute driving scenarios using the trained policy,
- Collect time-series data (e.g., positions and states of the ego vehicle and surrounding vehicles throughout the simulation).

You are expected to **reuse this provided infrastructure**, and focus your implementation effort on the **search logic, fitness function, and mutation operators** of the Hill Climber.

b. Implement mutation operators (“neighborhood generation”)

You must implement mutation operators that generate **neighboring scenarios** from an existing one. Mutations must operate on the **scenario parameters**, not on the agent or the environment code itself. Examples of valid mutations include (but are not limited to):

- Modifying the number of surrounding vehicles,
- Changing initial spacing or headway distances,
- Altering the initial lane of the ego vehicle,
- Small perturbations to multiple parameters simultaneously.

Your mutation operators should produce valid environment configurations.

c. Implement a fitness / evaluation function

Your fitness function must evaluate **how close a scenario is to a failure**.

The fitness function may analyze only the executed scenario (input configuration), and observed execution behavior (e.g., distances, collision flags). Recall that the goal of the Hill Climber is to m**inimize the distance** between the ego vehicle and other vehicles and ideally trigger a **collision** (distance lower than the internal simulation threshold).

If a collision occurs, the scenario is considered a **failing test case**.

d. Implement the Hill-Climbing logic

Your Hill Climber must:

- Start from an initial scenario,
- Iteratively generate neighboring scenarios using your mutation operators,
- Evaluate each candidate using your fitness function,
- Select and keep improved scenarios,
- Terminate when:
    - A collision is found, or
    - A stopping condition is reached (e.g., budget exhausted).

2. Experimental Evaluation and Comparison with Baselines

In this second part, you will compare your Hill-Climbing (HC) search strategy against the provided **random search baseline** for testing the PPO-based autonomous driving agent.

Using the shared framework, run both your Hill Climber and the random search on the same environment configuration space.
Perform a structured comparison covering the following aspects:

1. **Failure discovery**: for each method, report whether a **collision (failure)** was discovered. Your discussion should include:
    - Whether random search was able to find a collision.
    - Whether your Hill Climber was able to find a collision.
    - The number of distinct failing scenarios found by each method (if any).

    If no collision is found, please report the **minimum distance to collision** achieved by each method. Summaries may include tables or aggregated statistics.
2. **Scenario Characteristics**: Compare the failing scenarios discovered by your Hill Climber and by random search. Your discussion should include at least:
    - The environment configuration of the most critical scenario(s) (e.g., number of vehicles, spacing, initial lane).
    - The minimum distance between the ego vehicle and other vehicles.
    - Whether the failure scenarios found by HC differ qualitatively from those found by random search.
3. **Efficiency**: Evaluate and compare your Hill Climber vs. random search in terms of efficiency. Include:
    - Runtime (wall-clock time in seconds).
    - Number of scenario evaluations required to reach the closest failure or collision.
    - A brief discussion of scalability as the number of search iterations increases.
4. **Qualitative observations**: Reflect on your empirical findings answering the following questions:
    - Which types of traffic scenarios were most likely to lead to failures?
    - Did you observe patterns (e.g., dense traffic, short spacing, specific initial lanes)?
    - Does Hill Climbing tend to converge toward specific kinds of risky situations compared to random search?
5. **Pros and cons of Hill Climbing vs. random search**: Discuss the relative strengths and weaknesses of the two approaches, considering:
    - Effectiveness in discovering failures,
    - Efficiency and convergence speed,
    - Generality and robustness across different scenarios.

What to deliver

As final deliverable for this assignment, submit **one ZIP file** containing:

1. Project Report: A single PDF report including:
   - **Hill-Climbing design and rationale**
     Description and justification of:
     - Fitness function,
     - Mutation operators,
     - Search strategy.
       (max 1.5 pages)
   - **Experimental evaluation and comparison**
     Comparison between Hill Climber and Random Search, including metrics and observations.
     (max 1.5 pages)
   - **Reflection and insights**
     Discussion of strengths and weaknesses of search-based testing for RL agents, limitations, and lessons learned.
     (max 1 page)
   - **Optional appendix**
     Additional plots, tables, or screenshots (no page limit).

   Total report length (excluding appendix): max 4 pages.

2. Code Submission: A ZIP archive containing:
   - A requirements.txt file listing:
     - Python version used (if different from the recommended one),
     - All additional Python dependencies with versions.
   - Your completed Hill-Climbing implementation:
     - The modified Hill Climber file (without changing provided interfaces),
     - Any additional modules you created (if applicable).
   - A README.md file describing:
     - How to run your Hill Climber,
     - How to reproduce the results reported in your paper,
     - Any special setup steps.

Final Notes

- Creativity in fitness functions and mutation operators is encouraged.
- Clear reasoning and reproducibility matter more than simply finding a collision.