Decictor: Towards Evaluating the Robustness of Decision-Making in Autonomous Driving Systems

ICSE 2025 Artifact Abstract

Mingfei Cheng[1], Xiaofei Xie[1], Yuan Zhou[2], Junjie Wang[3], Guozhu Meng[4] and Kairui Yang[5]
[1]Singapore Management University, Singapore [2]Zhejiang Sci-Tech University, China
[3]Tianjin University, China [4]Chinese Academy of Sciences, China [5]Alibaba Group, China

I. Purpose

Autonomous Driving System (ADS) testing is crucial in ADS development, with the current primary focus being on safety. However, the evaluation of non-safety-critical performance, particularly the ADS's ability to make optimal decisions and produce optimal paths for autonomous vehicles (AVs), is also vital to ensure the intelligence and reduce risks of AVs. Currently, there is little work dedicated to assessing the robustness of ADSs' path-planning decisions (PPDs), i.e., whether an ADS can maintain the optimal PPD after an insignificant change in the environment. To address this gap, we propose *Decictor*, a tool designed to evaluate the robustness of ADSs' PPDs by generating non-optimal decision scenarios (NoDSs), where the ADS fails to plan optimal paths for autonomous vehicles (AVs). For a more intuitive understanding of NoDSs, we provide video examples on our website [1]. The full paper is available at https://arxiv.org/abs/2402.18393.

The provided artifact implements *Decictor*, which is built on the open-source, industrial-level autonomous driving system (ADS), Baidu Apollo [2]. As such, using this artifact requires foundational knowledge of Python, familiarity with Baidu Apollo [2], and experience with Docker [3]. We kindly request the following badges:

- **Available:** The artifact is publicly accessible at both Github [4] and Zenodo https://zenodo.org/records/14752133. Experimental results are available on our website [5].
- **Functional:** Detailed documentation is provided to guide users in utilizing *Decictor*, ensuring it can effectively generate NoDSs. While verifying the complete results is challenging due to the computational effort required (approximately 150 GPU/CPU days), we have released our experimental results on our website [5] to support reproducibility and consistency with our paper.
- **Reusable:** *Decictor* organizes and integrates various ADS testing baselines to facilitate future use. It is designed to be easily extendable, enabling researchers to develop advanced ADS testing techniques.

II. Provenance

The preprint of our full paper on *Decictor* is available at [6] with link https://arxiv.org/abs/2402.18393. The artifact can be accessed through Zenodo https://zenodo.org/records/14752133, GitHub [4], and our website [5].

III. Setup

A. Hardware dependencies

The artifact execution requires both CPU and GPU hardware. We recommend running the artifact on a Linux server with a CPU featuring at least a 16-core processor and a minimum of 32GB of memory, along with an NVIDIA Turing GPU with at least 12GB of RAM. The artifact has been tested on an NVIDIA RTX A5000 GPU. Additionally, at least 50GB of regular storage is required.

B. Software dependencies

The artifact runs on the Linux operating system with the Bash shell. It requires the Python package manager Conda, which can be downloaded and installed by following the instructions on the official webpage: https://conda.io/projects/conda/en/latest/user-guide/install/index.html. Additionally, the ADS under test is Baidu Apollo v7.0.0, which requires Docker, the NVIDIA GPU Driver, and the NVIDIA Container Toolkit. Detailed installation instructions for these dependencies can be found in the Baidu Apollo documentation: https://github.com/ApolloAuto/apollo/blob/r7.0.0/docs/specs/prerequisite_software_installation_guide.md

C. Installation

We provide straightforward installation instructions below. For more details, please refer to the 'README.md' file on our GitHub repository [4]. In the following instructions, we assume the root directory is '/workspace'. Please replace this path with your own directory as needed.

1) Install Baidu Apollo: Following the instructions below to install Baidu Apollo:

```
cd /workspace
# download apollo.zip from: https://zenodo.org/records/14752133
# unzip apollo.zip
unzip apollo.zip
# Build Apollo
cd /workspace/apollo
bash docker/scripts/dev_start.sh
# Once finished, you will see a docker container named apollo_dev_$USER. ($USER is your username)
cd /workspace/apollo
bash docker/scripts/dev_into.sh
```

```
# You are now in the container, shown like \
    $USER@in-dev-docker:/apollo. You need to
    run the following command in the container
     to build Apollo:
./apollo.sh build
# If the build finishes successfully, you can
    see the following message: [ OK ] Done
    building apollo. Enjoy!
# You can exit the container by typing `exit`.
exit
```

2) Install Decictor: Following the instructions below to install Decictor:

```
cd /workspace
# Download Decictor.zip from https://zenodo.
    org/records/14752133
# unzip Decictor.zip
unzip Decictor.zip
# Set Python Environment
conda create -n decictor python=3.7.16 -y
conda activate decictor
pip install -r requirements.txt
# install pytorch
pip install torch==1.13.1+cu117 torchvision
    ==0.14.1+cu117 torchaudio==0.13.1 --extra-
    index-url https://download.pytorch.org/whl
    /cu117
# edit root directories
vim /workspace/Decictor/config/common/project.
    yaml
# replace the following parameters with your
    paths
# apollo_root: like /workspace/apollo
# project_root: like /workspace/Decictor
# output_root: like /workspace/Outputs
```

IV. Usage

A. Basic Usage Example

After completing the setup successfully, you can use the following command to test the installation:

```
cd /workspace/Decictor
python main_fuzzer.py fuzzer=decictor fuzzer.
    run_hour=4 seed_name=scenario_1 map_name=
    sunnyvale_loop run_name=run_1
# After executing the command, you should see
    some logs appear without any errors.
```

B. Experiments

To reproduce the major results presented in our paper, we provide bash scripts located in the 'Decictor/scripts' folder. Note that you need to update the 'project_root' variable in each script before execution.

```
cd /workspace/Decictor
# To run experiments for RQ1:
bash scripts/rq1.sh
# To run experiments for RQ2:
bash scripts/rq2.sh
# All results will be saved at the output_root
    defined before.
```

For each run, you can check the 'violation_ids.csv' file to obtain the detected NoDSs.

C. Results

Considering that verifying the complete results is challenging due to the significant computational effort required (approximately 150 GPU/CPU days), we also provide the results along with corresponding scripts to reproduce the major results and figures. Please download by https://drive.google.com/file/d/1f9ca1Ggvwb5K3DA6VRjlHgrORNSKwP8e/view?usp=drive_link

References

[1] M. Cheng, X. Xie, Y. Zhou, J. Wang, G. Meng, and K. Yang, "Decictor video examples," https://sites.google.com/view/adsdecictor/video-cases, 2025.

[2] Baidu, "Apollo: Open source autonomous driving," 2019. [Online]. Available: https://github.com/ApolloAuto/apollo

[3] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[4] M. Cheng, X. Xie, Y. Zhou, J. Wang, G. Meng, and K. Yang, "Decictor," https://github.com/MingfeiCheng/Decictor, 2025.

[5] ——, "Decictor," https://sites.google.com/view/adsdecictor, 2025.

[6] M. Cheng, Y. Zhou, X. Xie, J. Wang, G. Meng, and K. Yang, "Decictor: Towards evaluating the robustness of decision-making in autonomous driving systems," *arXiv preprint arXiv:2402.18393*, 2024.