SBFT Tool Competition 2023 - Cyber-Physical Systems Track

Matteo Biagiola
Università della Svizzera italiana
Lugano, Switzerland
matteo.biagiola@usi.ch

Stefan Klikovits
Institute for Business Informatics - Software Engineering
Johannes Kepler University Linz, Austria
stefan.klikovits@jku.at

Jarkko Peltomäki
Åbo Akademi University
Turku, Finland
jarkko.peltomaki@abo.fi

Vincenzo Riccio
University of Udine
Udine, Italy
vincenzo.riccio@uniud.it

Abstract—We report on the organization and results of the third edition of the Cyber-Physical Systems tool competition, held as part of the SBFT workshop. Six tools (i.e., CRAG, EvoMBT, RIGAA, RoadSign, Spirale, and WOGAN) competed with the aim of triggering failures of two autonomous driving agents.

We evaluated the effectiveness of the tools in exposing failures as well as the diversity of the generated failures. This report describes our methodology, the competitors, the results, and the challenges we faced while running the competition experiments.

Index Terms—Autonomous Vehicles, Search-Based Software Testing

I. Introduction

Among the huge variety of existing safety-critical Cyber-Physical Systems (CPSs), self-driving cars are steadily growing in relevance within the software engineering research community [1] and industry [2]. Tool competitions offer the opportunity to benchmark new testing approaches and improve the state of the art.

For this reason, we organized, for the third consecutive year, the CPS track within the SBFT Testing Tool Competition along with the Java Test Case Generation [3] and the Fuzzing [4] tracks. This edition received six submissions: CRAG, EvoMBT, RIGAA, RoadSign, Spirale, and WOGAN. We received the same number of submissions as the previous edition. Four teams that joined the previous edition improved their tools for this year's competition. The other two teams participated in the competition for the first time confirming the attractiveness of this competition.

To facilitate the development of new test generators, we provided the participants an open-source and extensible test framework [5] together with the documentation on how to use it (tutorials, instructions, sample driving agents, and test generators). The test framework encapsulates the definition of complex driving scenarios and their execution in a simulator.

Since the past edition [6], we updated our test framework to support the latest version of the BeamNG.tech simulator [7]. We also introduced a more straightforward and comprehensive failure diversity metric to compare the performance of the

Fig. 1: A driving task for lane keeping assist systems (left); test simulation in BeamNG.tech (right).

tools in terms of structural and behavioral features of the tests. The comparison is done by utilizing feature maps as proposed by Zohdinasab et al. [8], [9]. As test subjects, we selected (1) BeamNG.AI, the driving agent included with the BeamNG.tech simulator, and (2) a DL-based driving agent based on the Dave-2 architecture proposed by Bojarski et al. [10]. Both test subjects have been used in previous research [11]–[15].

We compared the competing tools by running them multiple times on each test subject in order to account for the stochastic nature of the tools. CRAG by Arcaini and Cetinkaya [16] achieved the highest feature map coverage for both agents. RIGAA by Humeniuk et al. [17] and WOGAN by Winsten and Porres [18] ranked second and third.

II. Experimental Comparison

A. Goal

In this competition, we challenged the participants to implement generators of virtual tests for the CPSs. For simplicity, we restricted the search space to driving scenarios taking place in fixed environmental conditions (i.e., sunny day) and road layouts (i.e., flat roads with two lanes surrounded by grass).

The autonomous agents under test (test subjects) are lane-keeping assist systems (LKASs). The driving task consists of driving from the beginning to the end of a given road without

going off the right lane. The goal of the test generators is to generate challenging virtual roads that cause the test subjects to fail at the task, that is, drive off the right lane.

As in Figure 1, the test framework allows a generator to define virtual roads as sequences of road points (○) on a two-dimensional map. The test framework interpolates the road points using cubic splines and considers the first and last road points as the starting position (△) and the target (□). The framework initializes a given road in the simulator, executes the test subject, and returns the results of the simulation to the test generator. The test framework ensures that only *valid* roads are considered. In particular a given road is valid if: (i) does not self-intersect; (ii) does not contain overly-sharp turns; and (iii) is contained in a fixed-size map.

The goal of the challenge is to generate the highest number of diverse failure-inducing inputs, i.e., valid virtual roads that cause the ego-car controlled by the agent under test to drive off the lane. The test infrastructure detects a failure if the part of the ego-car that is outside of the lane is above a threshold called *OOB tolerance*. For instance, the OOB tolerance 0.5 triggers a failure when at least half of the ego-car is outside of the lane. We label failures as *Out of Bound (OOB) episodes*. We limit the time the test generators can use for test generation and simulation by using a fixed real-time budget.

B. Metrics

For this competition, we designed a metric that measures how many failure-inducing inputs with diverse features are found by each tool. Triggering similar failures is unhelpful in assessing the quality of a driving agent since this likely only exposes the same issues multiple times [13]. Therefore, we consider an OOB diversity measure based on both structural features (characteristics of the road) and behavioral features (characteristics of the output of the simulator). In particular, we consider high-level features that meaningfully characterize the tests. Specifically, we selected the following four features that have been empirically assessed by Zohdinasab et al. [9]:
Direction Coverage is a *structural* feature indicating how many directions the road covers;
Maximum Curvature is a *structural* feature that quantifies the smoothness the road as the inverse of its turns' radiuses;
Standard Deviation of the Steering Angle is a *behavioral* feature that measures the standard deviation of the ego-car's steering angles collected during simulation;
Mean Lateral Position is a *behavioral* feature that measures how close the driving agent drives to the lane margins.

We use these features to define a four-dimensional *feature map* where the failure-inducing inputs are positioned based on their feature values such that similar failure-inducing inputs occupy the same or neighboring map cells. The cells are obtained by dividing the range of each feature into 10 intervals. Such a feature map enables us to measure the OOB coverage achieved by a test generator hence quantifying how many diverse failure-inducing inputs it generated. This approach is inspired by recent work on test generation [8] and has been used for test selection [19] and as a test diversity metric [20].

Since the road inputs can be long but only a short part of them determines an OOB, we compute the considered features only for the road segment relevant to the triggered OOB, i.e., 30 meters before and after the OOB location. We measure the number of cells C_A covered by all test generators across all runs on a single test subject. We define the *Relative OOB Coverage* of a test generator on its i-th run as the ratio $|C_i|/|C_A|$ where C_i consists of the cells covered by the test generator during that run. The tests generators are ranked by the average Relative OOB Coverage values across all runs.

C. Test Subjects

We evaluated the competing tools using the BeamNG.tech driving simulator [7], kindly provided to our participants by BeamNG.GmbH. BeamNG.tech is widely-used in the software testing literature [11], as it features a *soft-body dynamics* simulation based on a spring-mass model that allows accurate reproduction of physical properties, e.g., vehicle deformation.

We chose two test subjects widely used in the software testing literature: BEAMNG.AI, the driving agent shipped with the BeamNG.tech simulator, and DAVE-2, a DL-based driving agent. The competitors had access to both test subjects, but we did not disclose our final experimental setup.

BEAMNG.AI is omniscient, i.e., it knows the geometry of the whole road before the simulation is performed and leverages this information to plan trajectories so that the ego-car is driven as close as possible to the speed limit, while keeping the vehicle as much as possible inside the lane.

DAVE-2 [10] is an end-to-end approach that uses a DL architecture consisting of a sequence of three convolutional layers and five fully-connected layers to predict steering angles from images taken by the on-board camera. DAVE-2 implements behavioral cloning, i.e., learns how to map images to steering angles from examples provided by experts. Therefore, we used only positive examples for training this agent, while we discarded the image-angle pairs in which the ego-car drove out of the lane. In particular, we trained DAVE-2 with images captured by the BeamNG.tech camera paired with steering angles of the ego-car, automatically collected by a script that forces BEAMNG.AI to drive at the center of the lane.

D. Tools

We evaluated a total of six tools. Below, we briefly describe the main characteristics of each of them. More information can be found in the corresponding reports:

CRAG is an approach based on a combinatorial model whose parameters are used for determining lengths and curvatures of a road in the Frenet frame representation [16].

EvoMBT [21] is a general purpose model-based tool that generates test cases from Extended Finite State Machines (EFSMs) based on model coverage criteria.

RIGAA [17] is an approach that combines reinforcement learning and evolutionary search to generate test scenarios.

RoadSign [22] combines a seeding approach promoting diversity with a multi-objective optimization process which aims to maximize road features which may reveal OOBs.

Spirale [23] is a search-based testing tool designed to generate scenarios for testing LKASs.

WOGAN [18] is an online test generation tool based on Wasserstein generative adversarial networks.

TABLE I: Experiment setups.

Name	Map Size (m × m)	Speed Limit (km/h)	Time Budget (h)	OOB Tol.
BEAMNG.AI	200 × 200	70	3	0.85
DAVE-2	200 × 200	25	3	0.1

E. Experimental Procedure

We ran each tool 6 times for BEAMNG.AI and 5 times for DAVE-2. The experiment setups are described in Table I.

The BEAMNG.AI agent has a speed limit of 70 km/h with an OOB tolerance value of 0.85 and a time budget of 3 h of real time. We executed the DAVE-2 agent with the same time budget, a speed limit of 25 km/h, and an OOB tolerance of 0.1. Thus, the DAVE-2 agent drives more slowly but a lower tolerance is used to trigger failures.

To ensure a fair comparison, we ran each tool the same number of times in each experiment setup. We ran the BEAMNG.AI experiments on a desktop PC running Microsoft Windows 10 Enterprise and featuring an eight-core Intel i9-9900K CPU @ 3.60 GHz, 64 GB of RAM, and an Nvidia Quadro RTX 4000 GPU. We collected the data for the DAVE-2 experiment on a desktop PC running Windows 10 Education with an Intel i9-10900X CPU, a Nvidia GeForce RTX 3080 GPU, and 64 GB of RAM. For all the experiments, we used the version v.0.26.2.0 of BeamNG.tech.

III. RESULTS

A. Test Generation Effectiveness and Efficiency

Table II reports the number of generated test cases (*#TCs*), the percentage of valid roads (*%Val.*), and the number of failure-inducing inputs (*OOBs*) produced by each tool (*Tool*) in each configuration (*Config*). The reported values are averages over the runs with the same configuration.

We observe that WOGAN generated the highest number of test cases, being the most efficient test generator in this competition. Across the configurations, WOGAN generated nearly twice as many roads as the second best tool Spirale. However, WOGAN also produced the highest number of invalid roads (> 50%), which reduces its effectiveness.

In terms of road validity, EvoMBT achieved a perfect score in both configurations. RoadSign, CRAG, and RIGAA achieved a comparable validity score (i.e., > 95%).

In both configurations, CRAG produced the highest number of OOBs, followed by RIGAA. We notice that, using the same budget, it was easier for the tools to trigger failures of the BEAMNG.AI agent compared with the DAVE-2 agent.

TABLE II: Valid and invalid tests generated by the tools.

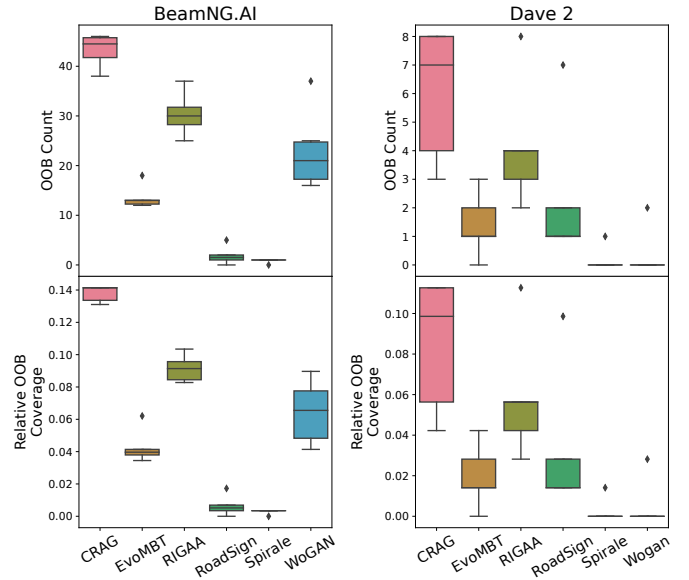| Config.	Tool	#TCs	%Val.	OOBs
BEAMNG.AI	CRAG	278.2	95.7	**43.3**
	EvoMBT	294.8	**100**	13.5
	RIGAA	322.7	94.6	30.3
	RoadSign	224.3	97.9	1.8
	Spirale	428.5	79.5	0.8
	WOGAN	**735.5**	54.2	22.8
DAVE-2	CRAG	134.2	96.1	**6.0**
	EvoMBT	160.8	**100**	1.4
	RIGAA	184.2	96.8	4.2
	RoadSign	103.6	98.5	2.4
	Spirale	232.0	80.4	0.2
	WOGAN	**529.8**	52.5	0.4

Fig. 2: Benchmark results. The box plots report the number of detected failures (top) and the achieved Relative OOB Coverage (bottom) in each configuration.

B. Final Scores

Table III reports the final ranking of the tools, computed as the sum of the Relative Map Coverage for each configuration.

CRAG won this edition of the SBFT CPS Tool Competition, since it is the tool that reached the highest score in both configurations. RIGAA ranked second by achieving the second best score for both configurations. WOGAN reached the third place mainly because of its good performance for DAVE-2, where it achieved the third best score.

As shown in Figure 2, the Relative OOB Coverage achieved by each tool (bottom) follows a similar trend as the distribution of triggered OOBs (top) in both configurations suggesting that the tools found quite diverse failures.

TABLE III: Final ranking. The Relative OOB Coverage values for each tool and agent. Bold indicates the best values.

Tool	BEAMNG.AI	DAVE-2	Final Score
CRAG	**.085**	**.138**	**.222**
RIGAA	.059	.091	.151
WOGAN	.006	.064	.070
EvoMBT	.020	.043	.062
RoadSign	.034	.006	.040
Spirale	.003	.003	.006

IV. CONCLUSIONS AND FINAL REMARKS

The SBFT CPS testing tool competition focused on the challenge of evaluating and comparing test generators for autonomous driving. In this third edition, six tools competed by testing two test subjects (i.e.,BEAMNG.AI and DAVE-2) and generated inputs that triggered failures of both systems. CRAG triggered the highest number of diverse failures thus winning the competition. RIGAA and WOGAN finished second and third, respectively. Compared to the previous edition, we designed an evaluation metric that takes into account multiple structural and behavioral features of the tests. For the next editions, we plan to renew the challenges faced by our competitors, e.g., by introducing different test subjects such as Reinforcement Learning based CPS and evaluating their accuracy and plasticity [24].

REFERENCES

[1] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, "Testing machine learning based systems: A systematic mapping," *Empir. Softw. Eng.*, vol. 25, no. 6, pp. 5193–5254, 2020.

[2] M. Borg, "The AIQ meta-testbed: Pragmatically bridging academic AI testing and industrial Q needs," in *Software Quality: Future Perspectives on Software Engineering Quality*. Springer, 2021, pp. 66–77.

[3] G. Jahangirova and V. Terragni, "SBFT tool competition 2023 - Java test case generation track," in *16th IEEE/ACM International Workshop on Search-Based And Fuzz Testing, SBFT 2023, Melbourne, Australia, May 14, 2023*.

[4] D. Liu, J. Metzman, M. Böhme, O. Chang, and A. Arya, "SBFT tool competition 2023 - Fuzzing track," in *16th IEEE/ACM International Workshop on Search-Based And Fuzz Testing, SBFT 2023, Melbourne, Australia, May 14, 2023*.

[5] V. Riccio, M. Biagiola, A. Gambi, S. Klikovits, and J. Peltomäki, "SBST CPS tool competition infrastructure," https://github.com/sbft-cps-tool-competition/cps-tool-competition, 2022.

[6] A. Gambi, G. Jahangirova, V. Riccio, and F. Zampetti, "SBST tool competition 2022," in *2022 IEEE/ACM 15th International Workshop on Search-Based Software Testing (SBST)*. IEEE, 2022, pp. 25–32.

[7] BeamNG GmbH, "BeamNG.tech," 2023. [Online]. Available: https://www.beamng.gmbh/research

[8] T. Zohdinasab, V. Riccio, A. Gambi, and P. Tonella, "Deephyperion: Exploring the feature space of deep learning-based systems through illumination search," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2021, pp. 79–90.

[9] ——, "Efficient and effective feature space exploration for testing deep learning systems," *ACM Trans. Softw. Eng. Methodol.*, 2022.

[10] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, pp. 1–9, 2016.

[11] S. Tang, Z. Zhang, Y. Zhang, J. Zhou, Y. Guo, S. Liu, S. Guo, Y.-F. Li, L. Ma, Y. Xue, and Y. Liu, "A survey on automated driving system testing: Landscapes and trends," *ACM Trans. Softw. Eng. Methodol.*, 2023.

[12] A. Gambi, M. Müller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA*. ACM, 2019, pp. 318–328.

[13] V. Riccio and P. Tonella, "Model-based exploration of the frontier of behaviours for deep learning system testing," in *Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020.

[14] G. Jahangirova, A. Stocco, and P. Tonella, "Quality metrics and oracles for autonomous vehicles testing," in *Proceedings of 14th IEEE International Conference on Software Testing, Verification and Validation*, 2021.

[15] A. Stocco, M. Weiss, M. Calzana, and P. Tonella, "Misbehaviour prediction for autonomous driving systems," in *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, 2020, pp. 359–371.

[16] P. Arcaini and A. Cetinkaya, "CRAG at the SBFT 2023 tool competition – Cyber-physical systems track," in *16th IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT 2023, Melbourne, Australia, May 14, 2023*.

[17] D. Humeniuk, F. Khomh, and G. Antoniol, "RIGAA at the SBFT 2023 tool competition - Cyber-physical systems track," in *16th IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT 2023, Melbourne, Australia, May 14, 2023*. IEEE, 2023.

[18] J. Winsten and I. Porres, "WOGAN at the SBFT 2023 CPS tool competition - Cyber-physical systems track," in *16th IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT 2023, Melbourne, Australia, May 14, 2023*, 2023.

[19] V. Nguyen, S. Huber, and A. Gambi, "SALVO: Automated generation of diversified tests for self-driving cars from existing maps," in *2021 IEEE International Conference on Artificial Intelligence Testing, AITest*, 2021.

[20] S. Klikovits, V. Riccio, E. Castellano, A. Cetinkaya, A. Gambi, and P. Arcaini, "Does road diversity really matter in testing automated driving systems? – A registered report," *arXiv preprint arXiv:2209.05947*, 2022.

[21] R. Ferdous, C. Hung, F. M. Kifetew, D. Prandi, and A. Susi, "EvoMBT at the SBFT 2023 tool competition," in *16th IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT 2023, Melbourne, Australia, May 14, 2023*, 2023.

[22] J. Ayerdi, A. Arrieta, and M. Illarramendi, "RoadSign at the SBFT 2023 tool competition - Cyber-physical systems track," in *16th IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT 2023, Melbourne, Australia, May 14, 2023*, 2023.

[23] D. D. Vivo and A. R. Fasolino, "Spirale at the SBFT 2023 tool competition - Cyber-physical systems track," in *16th IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT 2023, Melbourne, Australia, May 14, 2023*, 2023.

[24] M. Biagiola and P. Tonella, "Testing the plasticity of reinforcement learning-based systems," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 4, pp. 1–46, 2022.