GABezier at the SBST 2021 Tool Competition

Florian Klück, Lorenz Klampfl, Franz Wotawa
Christian Doppler Laboratory for Quality Assurance Methodologies for Autonomous Cyber-Physical Systems
Institute for Software Technology, Graz University of Technology, Inffeldgasse 16b/2, A-8010 Graz, Austria
{fklueck,lklampfl,wotawa}@ist.tugraz.at

***Abstract*—GABezier is a search-based tool for the automatic generation of challenging road networks for virtual testing of an automated lane keep system (ALKS). This paper provides a brief overview on the tool and summarizes the results of GABezier's participation at the first edition of the Cyber-Physical Systems Testing Tool Competition. We submitted our tool in two configurations, namely GABExplore and GABExploit. Especially the latter configuration has efficiently generated valid test cases and triggered many faults.**

I. Introduction

The ambitious development of Advanced Driver Assistant Systems (ADAS) introduces new challenges to the well-established verification and validation methods in the automotive industry as described in [1] and [2] . In the first edition of the Cyber-Physical Systems Testing Tool Competition different research groups developed automated test generation tools for virtual testing of self-driving vehicle software. With GABezier, we submitted a tool for the automatic generation of challenging road networks, combining Bezier Curves and Search [3]. We submitted two configurations of our tool, an exploratory configuration (GABExplore) that aims to maximize the number of unique failing test cases and an exploitative configuration (GABExploit) that aims to maximize the overall number of failing test cases. Both configurations were able to generate failure inducing inputs and in particular the latter configuration was able to expose the highest number of failing test cases [4] during the competition. In this paper we provide an overview on the GABezier method, summarize the results achieved in the competition and finally outline present limitations that we plan to consider as part of our future work.

II. Overview GABezier

GABezier aims to generate challenging road networks for virtual testing of an ALKS. For our method, road networks are represented as parametric curves, which are constructed based on control point sets. The road points described by the parametric curve form the input for the underlying code pipeline to automatically generate virtual roads, which are executable in a simulation environment. Furthermore, the obtained virtual road defines the dynamic driving task the ALKS has to perform from start to end. Changing the location of one control point has global influence on the resulting road geometry. During testing, we search for control point arrangements that result in challenging road networks. We consider road networks challenging, when the ALKS equipped vehicle does not perform the dynamic driving task correctly and leaves the intended road path. For search, as in our previous work [5], we use a genetic algorithm, where one set of control points represents one individual in the seed population. For evaluation of one control point individual, we construct the corresponding parametric curve and hand over the obtained road points to an underlying code-pipeline that was provided by the organizers of this competition [6]. Here, the final road is constructed, validated and executed in a virtual environment namely BeamNG.research [7]. To determine each individual's fitness, we measure the distance between the ego vehicle and the center line and outer line of the road. For optimization we rely on the standard genetic operators of crossing and mutation and aim to maximize the fitness function. For the competition we submitted two configurations of our tool, GABExploit and GABExplore.

a) GABExploit: In the exploitative configuration we randomly generate a seed population and continue the search until the given time budget is consumed. We aim to maximize the total number of failures exposed.

b) GABExplore: In the exploratory configuration, once a failure inducing input is found we restart the search with a new randomly generate seed population until the given time budget is consumed. Here, we aim to find more diverse failing test cases.

III. Experimental Set-Up and Procedure

The goal of the competition is to generate as many diverse failing test inputs as possible until a given time budget is consumed. A test input is considered failed, when the resulting virtual road forces the ALKS equipped vehicle to leave its designated lane by more than a certain pre-defined tolerance value. The system under test is an omniscient driving agent called BeamNG.AI that aims to follow the given road path while staying close to a pre-defined speed limit. During the competition each tool was evaluated in two different experimental set-ups: Default and SBST21. For both configurations, test inputs had to be generated within a given map of size 200 m x 200 m. The Default set-up was executed 5 times with a time budget of 5 hours per execution, a tolerance value of 0.95 and no speed limit. The SBST21 set-up was carried out 10 times with a shorter time budget of 2 hours per run, a tolerance value of 0.85 and a speed limit of 70 km/h. Each test-run is evaluated regarding the total number of triggered failures, but also the diversity between failure-inducing inputs. Furthermore, the total number of generated test inputs as well as the share of valid and invalid test inputs are considered to evaluate the tool performance regarding test generation efficiency and effectiveness.

IV. Benchmark Results

Table I shows the results obtained for both GAB configurations in the default experimental set-up and Table II summarizes the results obtained in the SBST21 experiment set-up. The following abbreviations are used: TR: test run, N-TC: test cases generated, V: valid test cases, I: invalid test cases, P: passed test cases, F: failing test cases, E: errors, SP: sparseness, N-SH: number of to sharp test cases, N-I: number of intersecting test cases.

TABLE I
Benchmark results for the default experiment set-up.

GABExploit									
TR	**N-TC**	**V**	**I**	**P**	**F**	**E**	**SP**	**N-SH**	**N-I**
1	306	299	7	287	12	0	4.093	7	0
2	474	393	81	308	85	0	12.785	81	0
3	461	431	30	311	120	0	10.795	30	0
4	311	309	2	299	10	0	8.608	2	0
5	415	357	58	357	0	0	-	58	0
AVG	393.4	357.8	35.6	312.4	45.4	0	9.07	35.6	0

GABExplore									
TR	**N-TC**	**V**	**I**	**P**	**F**	**E**	**SP**	**N-SH**	**N-I**
1	425	357	68	356	1	0	-	68	0
2	351	325	26	322	3	0	19.731	26	0
3	470	384	86	383	1	0	-	86	0
4	331	322	9	314	8	0	21.887	9	0
5	329	310	19	308	2	0	18.087	19	0
AVG	381.2	339.6	41.6	336.6	3	0	19.9	41.6	0

First, we see that both test generator configurations fulfill their unique purpose, i.e., GABExploit maximizes the amount of failing test cases and GABExplore maximizes the sparseness between failing test cases . In particular, with 120 failing test cases GABExploit exposed the highest number of failing tests in the default experimental set-up in the course of this competition. However, GABExploit did not show consistent performance and did not even find one failing test case in the last run. Furthermore, since this configuration does not promote searching for diverse failing test cases we see a lower sparseness of the generated failing test cases, which indicates an opportunity for improvement considering test efficiency. In contrast, GABExplore did not generate a comparable amount of failing test cases but the once we generated were unique. Furthermore, for the default set-up, GABExplore showed consistent performance over all test runs in finding at least one failure inducing test input. Similar observations can be made for the SBST21 experiment set-up, depicted in Table II. However, here we see that GABExplore is less efficient when given the smaller time budget. If we compare the results to the once obtained in the default configuration we see that GABExplore significantly improves when given more time for the calculation and search. It is worth noting that we were able to limit the number of invalid test cases and never generated intersecting test cases at all.

TABLE II
Benchmark results for the SBST21 experiment set-up.

GABExploit									
TR	**N-TC**	**V**	**I**	**P**	**F**	**E**	**SP**	**N-SH**	**N-I**
1	184	160	24	75	84	1	5.462	24	0
2	126	119	7	101	18	0	33.45	7	0
3	224	216	8	89	126	1	12.854	8	0
4	124	119	5	81	38	0	11.841	5	0
5	153	135	18	135	0	0	-	18	0
6	121	120	1	109	11	0	15.466	1	0
7	155	129	26	82	47	0	9.28	26	0
8	123	123	0	123	0	0	-	0	0
9	129	121	8	93	28	0	9.41	8	0
10	124	114	10	109	5	0	4.563	10	0
AVG	146.3	135.6	10.7	99.7	35.7	0.2	12.79	10.7	0

GABExplore									
TR	**N-TC**	**V**	**I**	**P**	**F**	**E**	**SP**	**N-SH**	**N-I**
1	137	125	12	125	0	0	-	12	0
2	132	123	9	122	1	0	-	9	0
3	120	113	7	112	1	0	-	7	0
4	127	118	9	118	0	0	-	9	0
5	141	135	6	134	1	0	-	6	0
6	143	132	11	129	3	0	2.434	11	0
7	33	30	3	8	0	22	-	3	0
8	142	139	3	138	0	0	-	3	0
9	121	118	3	115	3	0	18.075	3	0
10	151	126	25	126	0	0	-	25	0
AVG	124.7	115.9	8.8	112.7	0.9	2.2	10.25	8.8	0

V. Conclusion

This paper reports on the results of GaBezier's participation at the first edition of the Cyber-Physical Systems Testing Tool Competition. In one test run GAExploit achieved the highest number of failing test cases of all tools in the competition. However, the high similarity between failing test cases, reduces the overall test efficiency, offering opportunity for improvement.

References

[1] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, pp. 15–24, 04 2016.

[2] F. Wotawa, "Testing autonomous and highly configurable systems: Challenges and feasible solutions," in *Automated Driving: Safer and More Efficient Future Driving*, D. Watzenig and M. Horn, Eds. Cham: Springer International Publishing, 2017, pp. 519–532. [Online]. Available: "https://doi.org/10.1007/978-3-319-31895-0-22"

[3] F. Klück, L. Klampfl, and F. Wotawa, "Automatic generation of challenging road networks for alks testing based on bezier curves and search," 2021, arXiv:2103.01288 [cs.SE].

[4] S. Panichella, A. Gambi, F. Zampetti, and V. Riccio, "Sbst tool competition 2021," in *International Conference on Software Engineering, Workshops, Madrid, Spain, 2021*. ACM, 2021.

[5] F. Kluck, M. Zimmermann, F. Wotawa, and M. Nica, "Genetic algorithm-based test parameter optimization for adas system testing," in *Proceedings - 19th IEEE International Conference on Software Quality, Reliability and Security, QRS 2019*, ser. Proceedings - 19th IEEE International Conference on Software Quality, Reliability and Security, QRS 2019. United States: Institute of Electrical and Electronics Engineers, 7 2019, pp. 418–425.

[6] "https://github.com/se2p/tool-competition-av." [Online]. Available: https://github.com/se2p/tool-competition-av

[7] BeamNG GmbH, "BeamNG.research," software version 1.3.0.0, 2018-10-11, `https://www.beamng.gmbh/research`.