Simultaneously Searching and Solving Multiple Avoidable Collisions for Testing Autonomous Driving Systems – Extended abstract

Alessandro Calò, Paolo Arcaini, Shaukat Ali, Florian Hauer, Fuyuki Ishikawa

The oracle problem is a key issue in testing Autonomous Driving Systems (ADS): when a collision is found, it is not always clear whether the ADS is responsible for it. Our recent search-based testing approach offers a solution to this problem by defining a collision as *avoidable* if a differently configured ADS would have avoided it. This approach searches for both collision scenarios and the ADS configurations capable of avoiding them. However, its main problem is that the ADS configurations generated for avoiding some collisions are not suitable for preventing other ones. Therefore, it does not provide any guidance to automotive engineers for improving the safety of the ADS. To this end, we propose a new search-based approach to generate configurations of the ADS that can avoid as many different types of collisions as possible. We present two versions of the approach, which differ in the way of searching for collisions and alternative configurations. The approaches have been experimented on the path planner component of an ADS provided by our industry partner. This is the extended abstract of [8].

This is the extended abstract of paper [8].

1 Introduction

An Autonomous Driving System (ADS) in an integral part of autonomous cars, responsible for planning and executing a safe and comfortable path to a destination. Naturally, ensuring that an ADS always plans and executes a safe path avoiding a collision is critical. Search-based testing (SBT) is a popular approach to test an ADS under diverse environmental conditions, to ensure that during autonomous driving the ADS will not generate and execute paths that will lead to a collision [4] [5] [6] [2] [13] [9].

Irrespective of the type of testing approach, testing of an ADS faces the *oracle problem* [1] [3]. For example, if testing reveals a system execution in which the *ego car* (i.e., the car running the ADS) collides, it is not always apparent whether the ego car should be held responsible for it, and, even if not, whether it could have behaved differently in order to avoid it. Although some guidelines [12] [10] [11] have been proposed to define whether or not the ego car is responsible for a collision, they cannot be used to assess whether the ego car could avoid the collision with a different behaviour.

We recently proposed a more practical approach in [7]. It exploits the fact that ADSs are usually *configurable* and different ADS configurations lead to slightly different behaviours. In [7], a collision is defined to be *avoidable* if the ADS configured in a different way does not collide in the exact same scenario. The alternative configuration is said to *solve*

Alessandro Calò, Technical University of Munich, Germany.
Paolo Arcaini, National Institute of Informatics, Japan.
Shaukat Ali, Simula Research Laboratory, Norway.
Florian Hauer, Technical University of Munich, Germany.
Fuyuki Ishikawa, National Institute of Informatics, Japan.

the collision, and it is referred to as its *solution*. In [7], we also proposed a search-based approach (called the *combined approach*) in order to detect avoidable collisions.

The main problem of the combined approach is that the solution (i.e., the alternative configuration) found for a particular collision may be too specific, i.e., it may not solve other collisions that have been proven to be avoidable with different configurations. In this paper, in order to assess this, we have performed an experiment using the results we obtained in [7]: we run the path planner over all the found collisions using all the found alternative configurations. We discovered that the solutions found for one kind of collision can solve from 46.07% to 68.00% of other types of collisions. Therefore, the solutions found by the combined approach tend to overfit a particular collision type and are not too useful for re-engineering the path planner.

2 Proposed approaches

Starting from the needs of our industry partner, we understood that it would be desirable to find a *single configuration* that solves *multiple collisions* to better help the developers to improve the system. Following this research direction, we propose two approaches that aim at finding configurations which avoid as many collisions as possible. Both approaches continue to be based on evolutionary algorithms, as was done for the *combined approach* in [7].

2. 1 Single-stage approach

The first approach, named *single-stage approach* (SSA), is the natural generalization of the *combined approach* and aims at finding, at the same time, different collisions that can be avoided by the same alternative configuration. The first objective of SSA aims at finding alternative configurations that are not too different from the original one. The rationale is that the current system configuration is already very well-thought-out by the experts, and so it should not be changed too much, in order to avoid to obtain weird behaviors of the car.

At the same time, the approach searches for collisions over n different *search spaces*. For each search space S, we define an objective function that assesses whether an avoidable collision has been found in S, or how close we are from finding it. We identify two ways to consider these different objective functions:

1. SSA_{MO}: they are kept separate, each one constituting an objective of the multi-objective problem (so, SSA_{MO} is an (n+1)-objective problem);
2. SSA_{AO}: we define an objective function that *aggregates* the objectives for all the different search spaces (so, SSA_{AO} is a 2-objective problem). We propose two ways to define this function:
   (a) SSA_{AO_max}: the fitness value is given by the worst objective value across all the search spaces;
   (b) SSA_{AO_sum}: the fitness value is given by the sum of all objective values of all the search spaces.

2. 2 Two-stage approach

The search space of SSA is very large, being this given by the combination of the n search spaces for the different scenarios and of the alternative configuration; therefore, SSA may suffer from scalability issues. We then propose another approach, named *two-stage approach* (TSA), which aims at achieving more scalability by decomposing the task in two stages. In the first stage, n combined searches (i.e., applications of the combined approach from [7]) are conducted for n search spaces in order to find n avoidable collisions. In the second stage, a single search is performed to find a configuration that si-

表 1 **Number of runs (out of 30) which avoid $j = 1, \ldots, 7$ types of collision**

Approach	collision types (j)						
	1	2	3	4	5	6	7
SSA_{AO_max}	0	0	0	0	0	0	0
SSA_{AO_sum}	8	4	0	0	0	0	0
SSA_{MO}	9	20	1	0	0	0	0
TSA_{min}	0	0	0	5	13	8	4
TSA_{max}	0	0	0	5	12	9	4

multaneously solves all of them.

3 Experimental results

We run the two proposed approaches over the combination of 7 different search spaces (identifying 7 types of scenarios). Experimental results are reported in Table 1. It shows in how many runs (out of 30) each proposed approach solved j collisions ($j = 1, \ldots, 7$) of different types (i.e., from different search spaces), with the same alternative configuration; note that each column represents an amount of collision types. The last column represents the best case, in which all seven avoidable collisions are solved. For example, SSA_{MO} solves two collisions (of two different scenarios) in 20 runs.

3.1 SSA_{AO}

Regarding the SSA, the first strategy consisted in aggregating the danger differences across all the scenarios, and two methods were investigated. The first was to consider only the worst objective across all the search spaces (SSA_{AO_max}). This method was not effective at generating any favorable solutions within the allocated time budget. It is likely that such a conservative approach is simply not suitable to handle such a complex problem in a timely manner. Indeed, since the fitness focuses exclusively on the worst objective value, the search does not improve any other ones.

The second method consists in aggregating all objective values of all search spaces (SSA_{AO_sum}). This way, the search is given more information about improvements in all 7 search spaces. This worked better in the short term, because it defined a more "greedy" behavior, which was capable of finding solutions for 2 scenarios in 4 cases.

3.2 SSA_{MO}

The second strategy consisted in assigning each search space to an individual search objective (SSA_{MO}). This proved to be better, being able to find the solution for at least one collision of a search space in all 30 executions.

However, these results are still not acceptable for our aim of solving multiple collisions with a single alternative configuration: there should be at least a few cases solving the *majority* of them. Looking ahead at the results of the TSA, we see that this is possible.

3.3 TSA

Being composed of two separate stages, the TSA requires the selection of results from the first stage, which will be given as input to the second one. At the first stage, the selection chooses a single solution from the Pareto front, out of potentially many. Two straightforward selection criteria were employed: taking the solution which minimizes the configuration difference and the one which maximizes it. The first one represents the collision which is avoidable with the configuration closest to the original one, therefore being potentially the easiest one to solve. It will be denoted as TSA_{min}. The second one represents the collision that requires the biggest change in weights to be avoided, therefore being potentially the most difficult one to solve. It will be denoted as TSA_{max}. From the results in Table 1, we observe that the TSA is much more effective than the SSA, as it is always able to solve at least 4 collisions. From these results, it seems that there is no significant difference between the two selection criteria.

Acknowledgments

P. Arcaini and F. Ishikawa are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST. Funding Reference number: 10.13039/501100009024 ERATO. We thank our industry partner Mazda for providing the software used as the target in our work and discussing principles in testing and improving complex real-world automotive systems. The provided software is a prototype constructed for the purpose of evaluating new testing methods and its quality has no relation with the quality of Mazda products. Shaukat Ali is supported by the Co-evolver project (No. 286898/F20) funded by the Research Council of Norway under the FRIPRO program.

参 考 文 献

[1] Barr, E. T., Harman, M., McMinn, P., Shahbaz, M., and Yoo, S.: The Oracle Problem in Software Testing: A Survey, *IEEE Transactions on Software Engineering*, Vol. 41, No. 5(2015), pp. 507–525.

[2] Ben Abdessalem, R., Panichella, A., Nejati, S., Briand, L. C., and Stifter, T.: Testing Autonomous Cars for Feature Interaction Failures Using Many-objective Search, *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, New York, NY, USA, ACM, 2018, pp. 143–154.

[3] Briand, L., Nejati, S., Sabetzadeh, M., and Bianculli, D.: Testing the Untestable: Model Testing of Complex Software-Intensive Systems, *Proceedings of the 38th International Conference on Software Engineering Companion*, ICSE ’16, New York, NY, USA, Association for Computing Machinery, 2016, pp. 789–792.

[4] Bühler, O. and Wegener, J.: Evolutionary functional testing of an automated parking system, *Proceedings of the International Conference on Computer, Communication and Control Technologies (CCCT) and the 9th. International Conference on Information Systems Analysis and Synthesis (ISAS)*, 2003.

[5] Bühler, O. and Wegener, J.: Evolutionary functional testing of a vehicle brake assistant system, *6th Metaheuristics International Conference, Vienna, Austria*, 2005.

[6] Bühler, O. and Wegener, J.: Evolutionary functional testing, *Computers & Operations Research*, Vol. 35, No. 10(2008), pp. 3144–3160.

[7] Calò, A., Arcaini, P., Ali, S., Hauer, F., and Ishikawa, F.: Generating Avoidable Collision Scenarios for Testing Autonomous Driving Systems, *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, 2020, pp. 375–386.

[8] Calò, A., Arcaini, P., Ali, S., Hauer, F., and Ishikawa, F.: Simultaneously Searching and Solving Multiple Avoidable Collisions for Testing Autonomous Driving Systems, *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO ’20, New York, NY, USA, Association for Computing Machinery, 2020, pp. 1055–1063.

[9] Hauer, F., Pretschner, A., and Holzmüller, B.: Fitness Functions for Testing Automated and Autonomous Driving Systems, *International Conference on Computer Safety, Reliability, and Security*, Springer, 2019, pp. 69–84.

[10] Nister, D., Lee, H.-L., Ng, J., and Wang, Y.: The Safety Force Field.

[11] Rizaldi, A., Keinholz, J., Huber, M., Feldle, J., Immler, F., Althoff, M., Hilgendorf, E., and Nipkow, T.: Formalising and Monitoring Traffic Rules for Autonomous Vehicles in Isabelle/HOL, *Integrated Formal Methods*, Polikarpova, N. and Schneider, S.(eds.), Cham, Springer International Publishing, 2017, pp. 50–66.

[12] Shalev-Shwartz, S., Shammah, S., and Shashua, A.: On a Formal Model of Safe and Scalable Self-driving Cars, *CoRR*, Vol. abs/1708.06374(2017).

[13] Vos, T. E., Lindlar, F. F., Wilmes, B., Windisch, A., Baars, A. I., Kruse, P. M., Gross, H., and Wegener, J.: Evolutionary functional black-box testing in an industrial setting, *Software Quality Journal*, Vol. 21, No. 2(2013), pp. 259–288.