Generating Avoidable Collision Scenarios for Testing Autonomous Driving Systems

Alessandro Calò
Technical University of Munich
Munich, Germany
alessandro.calo@tum.de

Paolo Arcaini
National Institute of Informatics
Tokyo, Japan
arcaini@nii.ac.jp

Shaukat Ali
Simula Research Laboratory
Oslo, Norway
shaukat@simula.no

Florian Hauer
Technical University of Munich
Munich, Germany
florian.hauer@tum.de

Fuyuki Ishikawa
National Institute of Informatics
Tokyo, Japan
f-ishikawa@nii.ac.jp

***Abstract*—Automated and autonomous driving systems (ADS) are a transformational technology in the mobility sector. Current practice for testing ADS uses *virtual tests* in computer simulations; search-based approaches are used to find particularly dangerous situations, possibly collisions. However, when a collision is found, it is not always easy to automatically assess whether the ADS should have been able to avoid it, without relying on offline analyses by domain experts. In this paper, we propose a definition of *avoidable collision* that does not rely on any domain knowledge, but only on the fact that it is possible to reconfigure the ADS (in our case, the path planner component provided by our industry partner) in a way that the collision is avoided. Based on this definition, we propose two search-based approaches for finding avoidable collisions. The first one (named *sequential approach*), based on current industrial practice, first searches for a collision, and then searches for an alternative configuration of the ADS which avoids it. The second one (named *combined approach*), instead, searches *at the same time* for the collision and for the alternative configuration which avoids it. Experiments show that the *combined approach* finds more avoidable collisions, even when the *sequential approach* doesn't find any; indeed, the *sequential approach*, in the first search, may find too severe collisions for which there is no alternative configuration that can avoid them.**

***Index Terms*—autonomous driving, path planner, avoidable collision, search-based testing**

I. Introduction

Automated and autonomous driving is an intensive area of research and development that promises to transform the current way of transport. Autonomous and near-autonomous vehicles are estimated to bring significant economic growth (200 billion to 1.9 trillion USD by 2025) in the future as determined by McKinsey [24]. On the societal side, this technology aims to reduce the number of accidents, saving thousands of lives per year. However, to achieve such impacts, it is vital to ensure that autonomous cars are safe, which is challenging due to the complex implementation of the algorithms making autonomous decisions and dealing with a potentially infinite number of environmental situations [22].

An autonomous driving system (ADS) consists of a set of components that, given a final destination, sense the surrounding environment, plan a path, and implement this path into concrete driving actions. Testing an ADS only using real test drives is infeasible [19], [20], [33], and so different efforts have been put in computer simulations using *virtual tests* [36], [37].

Different works have been proposed for testing Advanced Driver-Assistance Systems (ADAS) [8]–[11], [32] and ADS [1], [5], [17], [21] with search-based techniques for finding safety-critical scenarios. Some of them specifically focus on finding collisions [6], [7]. Search-based approaches are natural solutions for this kind of problem, since it is impossible to perform exhaustive search due to the large number of potential environmental conditions.

As a result of trying to find the "most dangerous" situations, severe collisions can be found, but some of them are not necessarily caused by the ADS. For instance, an occluded pedestrian could run on the road just before the ego car arrives. Clearly, even the best driving system or the best human could not brake in time. From its standpoint, they are effectively *unavoidable collisions*.

The main problem is that we do not know in advance whether a collision is avoidable or not, since there is not a suitable reference specification. Some models have been proposed for defining which should be the *correct behaviour* of the ADS, such that no accident is its responsibility: see, e.g., the "Responsibility Sensitive Safety" (RSS) [30], the "Safety Force Field" (SFF) [27], or formal models based on theorem proving [28]. However, these models do not specify what should be the behaviour of the ADS when a collision is not its responsibility.

In other words, the *oracle problem* [4] is a big challenge.

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. Shaukat Ali was supported by the Co-Evolver project (Project No: 286898) funded by the Research Council of Norway. We thank our industry partner Mazda for providing the software used as the testing target in our work and discussing principles in testing complex real-world cyber-physical 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. We also thank Nian-ze Lee for initial discussions about the path planner, and Paula Schlatter Hasenack for the images of the scenarios reported in the paper.

Therefore, when we find a collision scenario, we must show it to domain experts and ask them to judge if that collision should have been avoided by the ADS. They should consult the existing guidelines [27], [28], [30] to certify that the ADS was not the cause of the collision, but also that it implemented all the possible actions to avoid it. Checking this latter condition is not only tedious for the engineers, but in some cases it is also impossible, as the engineers themselves cannot give a definitive answer. Moreover, it could also be not scalable due to the huge number of collision scenarios.

One way to assess that the collision should have been avoided is to show that the ADS can be optimized (by changing its configuration) such that it actually avoids it. The process is particularly difficult because the search space for the optimization can be huge, and engineers do not know which parameters should be changed (and by how much) to avoid the collision. Therefore, this optimized system configuration should also be searched for automatically.

Our intuition is that, if the original ADS leads to a collision in a scenario S, but a different ADS (i.e., a different configuration of the original one) does not, then the collision is *avoidable*. Given this definition, we consider two search-based approaches that look for avoidable collisions. The *sequential approach* reproduces the current practice in industry in which first the testing team generates some test cases, and then the design and development team improves the ADS using the test results; the approach consists of the serial application of Search-based Testing and System Parameter Optimization techniques: first, it searches for a collision scenario S^*, and then it searches for a new configuration of the ADS which avoids the collision in S^*. The second approach is called the *combined approach* and is one of the contributions of this work. It searches, at the same time, for a collision scenario S^* *and* the weights $\overline{w}^*$ necessary to avoid it.

In order to evaluate the approaches, we consider a *path planner* of an ADS, provided by our industry partner. At every time step, the path planner computes the *short-term path* that the car should follow in the next few seconds and the related control commands (e.g., longitudinal and lateral acceleration). It generates a list of short-term paths, ranks them using a ranking function that considers different aspects, such as *safety*, *vehicle limitation*, *regulation compliance*, and *comfort*, and selects the first one as next short-term path. The path planner can be configured through some *weights* that are involved in the decision process of generating and selecting paths. Each value assignment for the weights can be seen as a new path planner instance. Therefore, we have access to a "family of path planners" (each one having different weight values) that may take different decisions, leading to different trajectories followed by the ego car.

Experiments on the path planner show that often the *sequential approach* is not successful in finding avoidable collisions, as the first search finds collisions that are so severe that they cannot be avoided by any alternative path planner. The *combined approach*, instead, is more effective as it directly aims at finding avoidable collisions.

To summarize, the main contributions of this paper are:

1) a definition of avoidable collision that does not rely on any domain knowledge: if a collision occurs while using the default ADS configuration, but does not occur when using an alternative one, then it is said to be avoidable;
2) a search-based approach that searches for such avoidable collisions;
3) an empirical evaluation to compare the *sequential* and the *combined approaches* using an industrial path planner;
4) practical implications of our approach.

The paper is structured as follows. Sec. II introduces the industrial context in which this work has been developed. Sec. III defines *path planner* and related terms, and Sec. IV presents our definition of avoidable collision and two search-based approaches for finding them. Then, Sec. V describes the experiments we performed to evaluate the approach. Sec. VI discusses practical implications of the approach, and Sec. VII some threats that may affect its validity. Finally, Sec. VIII reviews some related work, and Sec. IX concludes the paper.

II. Industrial Context

This work is driven by the collaboration with a company in Japan developing autonomous cars, for testing autonomous car components, in particular for finding realistic paths that can cause collisions. Naturally, this can be done when working with a simulation in which the complete system — including decision making — is present and is to be tested. In other words, the approaches that will be presented can be used for *system* testing. In the case of our industry partner, however, only the path planner component is available at this time, so it was the only possible target of our current work.

The path planner of our industry partner is configurable with weights for the cost function that considers various aspects (e.g., safety and comfort) to generate paths. One issue is that the interval in which these weights can be considered has not been provided, so it's not clear which values are acceptable. In any case, the selection of these weights may potentially prevent *avoidable* collisions. Hence, by finding different weights that could potentially prevent them, we can better understand the behaviour of path planners under various situations and consequently improve the safety of the cars.

Such an approach would be helpful for testing engineers, who could immediately understand if an observed collision constitutes a failure or not, depending on whether it is avoidable. Moreover, it would bring new and useful insights to system designers: it would present them with witnesses of alternative configurations, which would help debugging the system as well as give suggestions about how it could be (re-)configured in order to improve the behaviour of the system.

Testing automated and autonomous driving systems and drawing accurate conclusions from obtained results is difficult: test engineers typically have limited knowledge about the detailed design of the system, including its configuration parameters. Similarly, it is difficult for system designers and developers to understand how the system configuration might be changed to avoid collisions identified during testing. To this

end, we aim to develop a search-based approach that can help test engineers in finding useful test cases as well as the system design and development team in improving the system with the help of the test results in relation to *avoidable* collisions. Such automated approaches are aimed at ensuring the safe operation of the autonomous cars.

III. Definitions

In the following, we provide some definitions related to our simulation environment.

A. Path and Scenario

Definition 1 (State of a car)**.** The *state* of a car c at time t is denoted as $s_t^c = \langle p, \vec{v}, \vec{a} \rangle$, where: $p = (x, y)$ is the position of car's geometric center, $\vec{v}$ is its (vectorial) velocity, with magnitude $|\vec{v}|$ and direction θ, and $\vec{a}$ is its acceleration. We use the dot notation to access tuple fields (e.g., $s_t^c.p$).

The *ego car* will be denoted by e, and its states by s_t^e. The simulator permits to instantiate other cars with predefined behaviour, called *scenario cars*. They will be denoted by indexes $i = 1, \ldots, m$ and their states by $s_t^1, \ldots, s_t^m$.

Definition 2 (Path)**.** Given a car c, its *path* p is a sequence of car states $[s_{t_0}^c, \ldots, s_{t_n}^c]$ such that the time difference between two consecutive states is fixed, i.e., $t_{i+1} - t_i = \Delta t_{sim}$ (for $i = 0, \ldots, n-1$), where Δt_{sim} is a parameter of the simulator[1].

Definition 3 (Scenario)**.** A *scenario* S describes the environment in which the ego car is operating. It is constituted of:

- a map describing the road structure;
- s_0^e: initial state of the ego car;
- a target destination for the ego car;
- the initial states of m scenario cars ($\{s_0^1, \ldots, s_0^m\}$) and their dimensions (width and length);
- a timeout t_{max} until which the simulation is run.

We will use the dot notation (e.g., $S.t_{max}$) to access a particular field of a scenario.

There is no particular driver model for the scenario cars: given only an initial position, velocity, and acceleration, a scenario car always follows exactly the same path. As future work, in order to test the system under more realistic situations, driver models could be included in the simulator: the better the driver models of other cars and the more realistic the physical model of the simulation are, the more likely the scenario instances are to be real.

The path of the ego car, instead, is governed by an ADS equipped with a path planner, as described in the next section.

B. Path planner

In an automated or autonomous [29] driving system, the *path planner* is an internal component that is repeatedly executed in order to determine the trajectory to be followed by the ego car. A generic path planner component works as follows. At each time step, the position of the surrounding elements (within a given horizon) is analyzed, and a large number of *short-term paths* are calculated. The *best* one is chosen, and followed by the ego car only for Δt_{sim}[2] seconds, after which the process restarts. Therefore, the final path p followed by the ego car is obtained by the concatenation of the first element of all the chosen short-term paths.

Different path planner systems implement different ways of selecting appropriate paths in order to identify a safe trajectory. Usually, a number of aspects are considered:

- *Safety*: no collision with moving or static objects must happen and safety distances must be respected;
- *Vehicle Dynamics Limitation*: actions that cannot be achieved by the car must be avoided (e.g., the planned path must not require impossible steering angles);
- *Compliance*: the car should respect road regulations;
- *Comfort*: the path should be as comfortable as possible for the passenger (e.g., not too much lateral acceleration).

The path planner provided by our industry partner (also considered in [23]) uses a *weighted* cost function to rank short-term paths and selects the least costly one. The following weights are used to address some of the above aspects:

- w_1: a factor that is multiplied with the maximum lateral acceleration of the short-term path;
- w_2: a constant that is added to the total cost if the maximum lateral acceleration is over a given threshold;
- w_3: a constant that is added to the total cost if the speed is over a given speed limit;
- w_4: a constant that is added to total cost if the maximum acceleration is over a certain threshold;
- w_5: a constant that is added to the total cost if the maximum deceleration is over a given threshold;
- w_6: a constant that is added to the total cost if the curvature of the short-term path is over a given threshold;
- w_7: a constant that is added to the total cost if the path is considered *extremely* dangerous.
- w_8: a constant that is added to the total cost if the path is considered *very* dangerous, but not *extremely* dangerous.

The cost function itself is simply the sum of all the described velocity and acceleration values, weighted by their respective weights. It cannot be reported here due to IP protection.

Definition 4 (Automated/Autonomous Driving System)**.** We identify with $\texttt{ADS}_{\overline{w}}$ the automated or autonomous driving system (ADS) equipped with a path planner configured with given weight values $\overline{w} = [w_1, \ldots, w_8]$. It can be seen as a function that, given a scenario S, produces a path p for the ego car up to simulation time $S.t_{max}$. Formally, $\texttt{ADS}_{\overline{w}}(S) = p$.

From here on, we identify with $\texttt{ADS}_{\overline{w}'}$ the same system ADS whose path planner component has been configured with different weight values $\overline{w}'$.

IV. Proposed approach

In this work, we consider the approach of utilizing testing and optimization techniques in order to address safety concerns. In particular, we are interested in checking if dangerous

[1] In our experiments, $\Delta t_{sim} = 0.1s$.

[2] Note that the time difference $\Delta t_{planner}$ between path planner executions could be different from Δt_{sim}; however, in our case, they are the same.

collisions can occur, and particularly on whether they can be somehow avoided. As discussed in Sect. II, we only have a path planner component available at this point, but our approach is applicable to a complete ADS.

In the following, we define in Sec. IV-A a measure of *danger* assessing how close the ego car is to a collision, and, in case the collision happens, how severe it is. Then, in Sec. IV-B, we introduce a notion of collision avoidability, and finally, in Sec. IV-C, we introduce two approaches for finding avoidable collisions: the first one is based on the current practice in industry, while the second one is one of the contributions of this paper that tries to overcome the limitations of the first approach.

A. Danger

Definition 5 (Danger)**.** Given the states $s^e_{t_i}$ and $s^j_{t_i}$ of the ego car and scenario car j at time t_i, we define the *danger* of their interaction as:

$$danger(s^e_{t_i}, s^j_{t_i}) = \begin{cases} \vec{v}_{e|j}^{\,t_i} + K & \text{if } collision(s^e_{t_i}, s^j_{t_i}), \\ \frac{\vec{v}_{e|j}^{\,t_i}}{\|s^e_{t_i}.p, s^j_{t_i}.p\|^2} & \text{otherwise.} \end{cases}$$

where $\vec{v}_{e|j}^{\,t_i}$ is the relative speed between the ego car and the scenario car j, $\|\cdot\|$ the Euclidean distance, and $collision(\cdot,\cdot)$ a predicate telling if there is a collision between the two cars[3].

In the event of a collision, the danger increases with the collision speed; if there is no collision, the relative speed is still an indication of danger, but its effect diminishes with increasing distances. Note that the definition guarantees that the *danger* measure is always higher when there is a collision than when there is none; the value of K is the boundary between collision and not collision and must be determined on the base of the simulation settings.[4]

We can therefore identify the most unsafe situation in a scenario as that having the maximum danger.

Definition 6 (Scenario danger)**.** The danger of the execution of a scenario S (containing m scenario cars) with a system $\texttt{ADS}_{\overline{w}}$ is defined as:

$$danger(S, \texttt{ADS}_{\overline{w}}) = \max_{\substack{i\in\{1,\dots n\},\\ j\in\{1,\dots m\}}} danger(s^e_{t_i}, s^j_{t_i})$$

where $\texttt{ADS}_{\overline{w}}(S) = [s^e_{t_1}, \dots, s^e_{t_n}]$ is the path followed by the ego car, and $[s^j_{t_1}, \dots, s^j_{t_n}]$ is the path of the scenario car j. Note that $t_n = S.t_{max}$.

In the following, we use the predicate $collision(S, \texttt{ADS}_{\overline{w}})$ to identify if there is a collision when running scenario S with $\texttt{ADS}_{\overline{w}}$. Following Def. 5, $collision(S, \texttt{ADS}_{\overline{w}}) = danger(S, \texttt{ADS}_{\overline{w}}) \geq K$.

[3]Note that the detection of collision is based on the overlapping of the cars.

[4]In our setting, the minimum distance in which there is no collision is 1.8 m, and the maximum relative speed that can be obtained is 320 km/h. Given these values, setting K to 100 guarantees the desired order of danger values between collision and non-collision scenarios.

B. Avoidable collision

The *danger* measure can be used to drive a search algorithm to find scenarios in which the ego car collides. However, these are not necessarily evidence that the path planner component is faulty: indeed, in some scenarios the ego car cannot avoid the collision. If we run the search algorithm trying to maximize the danger, it is likely that it finds such unavoidable collisions. The main problem is that it is difficult to decide upfront whether a collision is unavoidable or not. As explained in Sec. II, the context in which these kinds of systems are developed makes it difficult for testing engineers to understand system specifications. Additionally, as external researchers, we have not been given any information on how the system was developed, nor a detailed specification document describing which behaviours are expected. In other words, the complete system, knowledge-wise, is to us a black box.

The only information we have is that the path planner can be reconfigured: by changing the weights, it will behave differently in some situations. Therefore, we can consider that each weight configuration characterizes a different path planner. This allows us to compare the behaviour of the current path planner with that of *alternative* versions which could behave better in some cases. Exploiting this, we give the definition of *avoidable collision*.

Definition 7 (Avoidable collision)**.** Let $\texttt{ADS}_{\overline{w}}$ be an autonomous driving system and S a scenario for which $collision(S, \texttt{ADS}_{\overline{w}})$ holds. We say that the collision is *avoidable* if there exist some different weights $\overline{w}'$ such that $\neg collision(S, \texttt{ADS}_{\overline{w}'})$ holds, i.e., the system configured with different weights $\overline{w}'$ for the path planner component can avoid the collision. We identify the avoidable collision as $\langle S, \overline{w}' \rangle$.

Note that, although $\texttt{ADS}_{\overline{w}'}$ avoids a collision in a certain scenario, it is not necessarily better than the original $\texttt{ADS}_{\overline{w}}$. There could exist other scenarios where $\texttt{ADS}_{\overline{w}'}$ decreases the safety of the run, or even introduces collision situations not present under $\texttt{ADS}_{\overline{w}}$. In that sense, $\texttt{ADS}_{\overline{w}'}$ is only a *witness* that the current path planner component can be improved as this particular collision could be avoided.

C. Approaches for finding avoidable collisions

In order to find avoidable collisions, we identify two possible approaches.

The first one follows the workflow commonly adopted in industry (according to the companies our groups are in contact with) in which first the testing team applies test case generation, and then the design and development team uses the test results to improve the system. We reproduce this common practice by defining the *sequential approach*, which applies search-based testing and system parameter optimization sequentially: it first executes a search to find a test scenario S^* where $\texttt{ADS}_{\overline{w}}$ leads to a collision, and then executes a second one to find some alternative weights $\overline{w}^*$ such that $\texttt{ADS}_{\overline{w}^*}$ does not lead to a collision in S^*.

The main weakness of the *sequential approach* is that it may only find unavoidable collisions. This prevents the design and

development teams from being able to improve the system. Therefore, in this paper, we propose a novel approach (named combined approach) that consists in running a search for finding, *at the same time*, a scenario S^* and weights $\overline{w}^*$ such that S^* leads to a collision with $\texttt{ADS}_{\overline{w}}$, but not with $\texttt{ADS}_{\overline{w}^*}$.

In this way, $\overline{w}^*$ represents a witness of alternative system weights which avoid the collision, and it can be handed to the design and development team. This additional information eases the process of making the system more safe overall.

In the following, all the problems are defined as minimization problems and the fitness functions are defined accordingly.

1) Sequential approach (SA): This approach is constituted of two consecutive searches, the *collision search* and the *weights search*, described in the following.

a) Collision search (C): In this search, individuals represent scenarios. Search variables are selected fields $\overline{f}$ describing the initial states of scenario cars (e.g., initial position and velocity of a given scenario car) of a baseline scenario S, denoted in this context as $S_{\overline{f}}$. Hence, $S_{\overline{f}'}$ will denote the scenario $S_{\overline{f}}$ with modified values $\overline{f}'$ for the searched fields. The collision search is a single objective search and the fitness function is defined as:

$$\mathit{fitness}^{C}_{\mathit{danger}}(\overline{f}') = -\mathit{danger}(S_{\overline{f}'}, \texttt{ADS}_{\overline{w}})$$

The result of the search are some values $\overline{f}^*$ for the searched fields. They identify a scenario $S_{\overline{f}^*}$ in which the danger is maximized (possibly describing a collision) for the original system $\texttt{ADS}_{\overline{w}}$. If $S_{\overline{f}^*}$ is not a collision scenario (because there is no collision in the search space or the search wasn't given enough time), the *sequential approach* ends, and the following weights search is not performed.

b) Weights search (W): In this search, we look for alternative weights values $\overline{w}'$ such that $\texttt{ADS}_{\overline{w}'}$ does not lead to a collision when executed on the collision scenario $S_{\overline{f}^*}$ found in the previous search. Therefore, each individual identifies new weights values. This search has two objectives. The first one, oppositely to the collision search, is to minimize the danger obtained in $S_{\overline{f}^*}$ with alternative weights, i.e., avoid the collision. The corresponding fitness function is:

$$\mathit{fitness}^{W}_{\mathit{danger}}(\overline{w}') = \mathit{danger}(S_{\overline{f}^*}, \texttt{ADS}_{\overline{w}'})$$

The second objective is not to deviate excessively from the original weights $\overline{w}$. There are different reasons for this requirement. First of all, it has been empirically observed that systems configured with weights very different than the original ones sometimes behave strangely, for example causing the ego car to stand still even when not in the presence of other vehicles. Secondly, it cannot be claimed — nor can it be easily determined — that a *very different* set of weights doesn't introduce dangerous situations in different scenarios, where there were previously none. In other words, the weight values provided by our industry partner are considered to be the only ones extensively tested. The farther away the generated weights are from these 'original' weights, the less confidence we have that they won't increase the danger in some other unknown scenario. The intuition for this claim comes from the (empirically observed) fact that difference in weights is somehow proportional to difference in behaviour. Obviously it needs to be formally verified, which will be left as future work. The corresponding fitness function is:

$$\mathit{fitness}^{W}_{\Delta w}(\overline{w}') = \sqrt{\sum_{i=1}^{8} \left(\frac{w_i - w'_i}{d_i^U - d_i^L} \right)^2} \tag{1}$$

being $[d_i^L, d_i^U]$ the search interval of weight w_i, i.e., the change of a weight is normalized with respect to the size of the domain from which the optimizer has to choose from.

2) Combined approach (CA): In this case, we directly search for avoidable collisions. The way to achieve this is to have a single search where the individual is constituted of scenario fields as well as alternative weights, i.e., $\langle \overline{f}', \overline{w}' \rangle$. This search is called *collision-and-weights search* (CW). Similarly to the weights search, this one is multi-objective. The objective of minimizing the weights difference remains the same, so the fitness function is the same as in Eq. 1:

$$\mathit{fitness}^{CW}_{\Delta w}(\langle \overline{f}', \overline{w}' \rangle) = \mathit{fitness}^{W}_{\Delta w}(\overline{w}')$$

The second objective, instead, is to maximize the difference in danger between the original $\texttt{ADS}_{\overline{w}}$ and the modified $\texttt{ADS}_{\overline{w}'}$; the fitness function is defined as follows:

$$\mathit{fitness}^{CW}_{\mathit{danger}}(\langle \overline{f}', \overline{w}' \rangle) = (D_{\overline{w}'} - D_{\overline{w}}) + \mathit{penalty}(\langle \overline{f}', \overline{w}' \rangle)$$

where $D_{\overline{w}} = \mathit{danger}(S_{\overline{f}'}, \texttt{ADS}_{\overline{w}})$ is the maximum danger obtained by the original $\texttt{ADS}_{\overline{w}}$, and $D_{\overline{w}'} = \mathit{danger}(S_{\overline{f}'}, \texttt{ADS}_{\overline{w}'})$ is the one obtained by the modified $\texttt{ADS}_{\overline{w}'}$. The first operand of the sum specifies that we prefer cases in which the modified system is better in terms of danger (recall that we are minimizing); however, it is not sufficient to guarantee the desired ranking of solutions according to the avoidance or introduction of collisions. Therefore, the term $\mathit{penalty}(\langle \overline{f}', \overline{w}' \rangle)$ is used to express this desired order. The term is defined as follows:

$$\mathit{penalty}(\langle \overline{f}', \overline{w}' \rangle) = \begin{cases} 0 & \text{if } C_{\overline{w}} \wedge \neg C_{\overline{w}'}, \\ 1000 & \text{if } C_{\overline{w}} \wedge C_{\overline{w}'} \vee \neg C_{\overline{w}} \wedge \neg C_{\overline{w}'}, \\ 2000 & \text{if } \neg C_{\overline{w}} \wedge C_{\overline{w}'}. \end{cases}$$

where $C_{\overline{w}} = \mathit{collision}(S_{\overline{f}'}, \texttt{ADS}_{\overline{w}})$ and $C_{\overline{w}'} = \mathit{collision}(S_{\overline{f}'}, \texttt{ADS}_{\overline{w}'})$. Among all the cases, we prefer those in which the original system leads to a collision and the modified one doesn't, as this shows an avoidable collision (penalty 0). After those, we prefer the ones in which the original and modified systems have the same collision behaviour (penalty 1000). The least preferred solutions are those in which the modified system leads to a collision while the original one didn't (penalty 2000).

D. Selection of final results

Both the *sequential* and the *combined approaches* return, as final result, a Pareto front of solutions. From this, we are only interested in those that identify avoidable collisions. Let's recall from Def. 7 that an avoidable collision is defined as a

pair $\langle S, \overline{w}' \rangle$. We here show how to extract all the avoidable collisions from the Pareto fronts.

1) Sequential approach: Given the Pareto front P_{SA} returned in the *sequential approach* (i.e., by the weights search), let $\widetilde{P}_{SA}$ be the set of solutions identifying avoidable collisions:

$$\widetilde{P}_{SA} = \{\overline{w}' \in P_{SA} \mid \neg collision(S_{\overline{f}^*}, \texttt{ADS}_{\overline{w}'})\}$$

where $\overline{f}^*$ is the solution found by the collision search and $S_{\overline{f}^*}$ the corresponding collision scenario. If $\widetilde{P}_{SA}$ is empty, it means that the *sequential approach* has no solutions. Otherwise, we can build the set of avoidable collisions as follows:

$$AC_{SA} = \{\langle S_{\overline{f}^*}, \overline{w}' \rangle \mid \overline{w}' \in \widetilde{P}_{SA}\}$$

2) Combined approach: Given the Pareto front P_{CA} returned in the *combined approach* (i.e., by the collision-and-weights search), let $\widetilde{P}_{CA}$ be the set of solutions identifying avoidable collisions:

$$\widetilde{P}_{CA} = \left\{ \langle \overline{f}', \overline{w}' \rangle \in P_{CA} \mid \begin{array}{l} collision(S_{\overline{f}'}, \texttt{ADS}_{\overline{w}}) \wedge \\ \neg collision(S_{\overline{f}'}, \texttt{ADS}_{\overline{w}'}) \end{array} \right\}$$

We can map the solutions to avoidable collisions as follows:

$$AC_{CA} = \{\langle S_{\overline{f}'}, \overline{w}' \rangle \mid \langle \overline{f}', \overline{w}' \rangle \in \widetilde{P}_{CA}\}$$

V. Experiments

A. Design

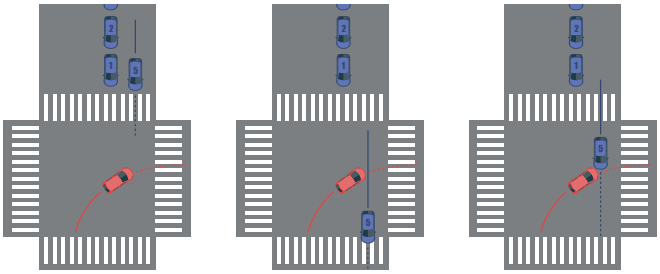
1) Search spaces: For the *sequential approach*, individuals in the collision search are scenario fields values $\overline{f}'$ of a baseline scenario $S_{\overline{f}}$ (see Sec. IV-C1a), and in the weights search they are weight values $\overline{w}'$ (see Sec. IV-C1b). For the *combined approach*, individuals in the collision-and-weights search are both scenario fields values and weight values $\langle \overline{f}', \overline{w}' \rangle$ (see Sec. IV-C2). In order to have a fair comparison between the results of the two approaches, we set the same search space for scenario fields in the collision search and in the collision-and-weights search, and the same search space for weights in the weights search and in the collision-and-weights search. The search spaces have been defined as follows.

In order to evaluate the approaches under different conditions, we identified seven search spaces for scenario fields $\overline{f}'$ as follows. We first designed the seven complete *baseline scenarios* shown in Table I, that cover different driving situations. Then, we parametrized some fields $\overline{f}$ of each baseline scenario (e.g., the speed, position, and acceleration of a scenario car). In order to do this, we first identified one scenario car with which the ego car could collide. For this car, we manually identified the extremes of the intervals for which the ego car exhibits a different behaviour w.r.t. the baseline scenario.

For example, given the baseline scenario S_1, car #5 was selected as it is the one crossing the intersection and therefore the one which could possibly collide with the ego car, which turns right at the intersection. An illustration can be seen in Fig. 1. By setting its speed to 8 m/s, it crosses the intersection after the ego car turns right (Fig. 1(a)). By increasing its speed to 15 m/s, instead, the ego car turns after its passage

TABLE I
Baseline scenarios

ID	Description
S_1	The ego car must turn right at the intersection, and there is one car crossing the intersection from the opposite direction. This car is hidden by a queue of cars waiting at the stop (adapted from Uber collision also considered in [14])
S_2	The ego car must overtake a preceding slow car, but there is a car approaching from behind on the passing lane
S_3	The ego car is proceeding on its lane and another car is proceeding in the opposite direction in the other lane
S_4	The car is proceeding on its lane and two cars are crossing the main road from left to right
S_5	The car is proceeding on its lane and two cars are crossing the main road, one from left, and one from right
S_6	The ego car must overtake a parked car, but there is a car coming from the opposite direction on the passing lane
S_7	The ego car must turn right at the intersection, and there is one car crossing the intersection from right, and another car crossing from the opposite direction



(a) $s_0^5.|\vec{v}| = 8m/s$ (b) $s_0^5.|\vec{v}| = 15m/s$ (c) expected collision

Fig. 1. Method for devising search spaces for fields $S_{\overline{f}'}$ in scenario S_1, with the two identified extremes and the expected collision inside this interval.

(Fig. 1(b)). The intuition is that the boundary between these two behaviours could exhibit a collision (Fig. 1(c)). In a similar way, we identified intervals for all other scenario fields related to car #5 which somehow change the behaviour of the ego car. More than one scenario field was selected because collisions are more likely to happen due to a combination of changes than to a single one. We proceeded in a similar way for the design of the search spaces of all collision searches which are also used by the collision-and-weights searches. Table II reports, as an example, the search space for the baseline scenario S_1.

Deriving the search spaces for the weights is not easy, since our industry partner did not disclose insights about the system design. We have only been given the path planner configured with some weights values $\overline{w}$ that they consider satisfactory; apart from this, the path planner is black box. As mentioned previously, we believe that any feasible alternative path planner should have weights values $\overline{w}'$ not too different from $\overline{w}$; as already observed in Sec. IV-C1b, changing the weights too much could generate a system which leads to extreme behaviours, e.g., the ego car not moving at all. For this reason, we avoided constructing a too-big search space for weights by restricting the search to values having the same order of magnitude of the original ones; namely,

TABLE II
SEARCH SPACES FOR BASELINE SCENARIO S_1

Approach		Search space	
CA	*SA*	Variable	Interval
collision-and-weights search	collision search	$s_0^5.x$: Initial x of 5	[5, 7]
		$s_0^5.y$: Initial y of 5	[220, 260]
		$s_0^5.\|\vec{v}\|$: Initial speed of 5	[8, 15]
		$s_0^5.a$: Initial acc. of 5	[0, 2]
	weights search	w_1	[4, 6]
		w_2	[2000, 4000]
		w_3	[150, 350]
		w_4	[10, 30]
		w_5	[10, 30]
		w_6	[10, 30]
		w_7	[90000000, 110000000]
		w_8	[9000, 11000]

given each original value w_i, the search interval was set to $[w_i - 10^{\lfloor \log_{10}(w_i) \rfloor}, w_i + 10^{\lfloor \log_{10}(w_i) \rfloor}]$.

For each baseline scenario, we also had to define the timeout t_{max}. We approached this by first defining, for each scenario car, a 'critical' section where it could possibly collide with the ego car. By setting scenario fields for those cars so that they are as slow as possible and as far away as possible from their respective critical sections, we were able to compute the maximum amount of time it took for all cars to exit them. We took this value as t_{max} for its respective scenario.

2) Settings of the search algorithm: We used jMetal 5.7 as search framework [15], [26], namely NSGA-II [12]. As setting for NSGA-II, we used the default settings of jMetal: parents selection with Binary tournament, SBX crossover operator, polynomial mutation operator, crossover rate of 0.9, and mutation rate equal to the reciprocal of the number of variables. We set a population size of 100 individuals.

As termination condition of all searches, we set 1200 as the number of fitness evaluations. Therefore, in total, the *sequential approach* can rely on 2400 fitness evaluations (from the collision and weights searches), while the *combined approach* (i.e., the collision-and-weights search) only relies on 1200 fitness evaluations. This seems to be unfair for the *combined approach*; however, since the collision-and-weights search requires two simulations of the scenario for each fitness evaluation, while the collision and the weights search require only one, the total execution times required by the *sequential* and *combined approaches* are comparable.

Experiments have been executed on Amazon Elastic Compute Cloud, using instances with a 2.9 GHz Intel Xeon CPU and 15GB RAM. To account for the randomness in search algorithms, each experiment has been executed 30 times.

B. Comparison with Random Search (RS)

The use of a multi-objective search algorithm (i.e., NSGA-II in our context) must be justified by comparing it with a simple comparison baseline algorithm, i.e., Random Search (RS). Given that NSGA-II produces a Pareto front comprising of a set of solutions, it is recommended to use a quality indicator to compare two algorithms. To this end, we selected *hypervolume* (HV) to compare NSGA-II and RS in terms of the quality of solutions produced by them. HV was selected based on a guide that provides a step-wise process to select quality indicators for a problem at hand [34]. Followed by this, statistical tests shall be chosen to determine the statistical significance of the results. To select the suitable statistical tests, we followed another guideline [2]. We selected two tests: 1) Mann-Whitney U test to determine the statistical significance of the results, 2) The A12 value statistics to determine the strength of the significance. The results of the tests in our case are interpreted as follows: If Mann-Whitney U produces a p-value < 0.05, it means there exist significant differences between the quality of solutions provided by NSGA-II and RS. The A12 statistical test tells which algorithm has the highest chance to be better and it is interpreted as follows: 1) If A12 is equals to 0.5, it means that there is no difference between NSGA-II and RS, 2) if A12 is greater than 0.5, it means that NSGA-II has higher chance to produce better quality solutions than RS, and vice versa, if A12 is less than 0.5. Note that the results of Mann-Whitney U and A12 must be studied together to determine the significance of the results and to conclude which algorithm is better in terms of quality solutions measured with HV.

For the collision search, there is no statistical difference between NSGA-II and RS. For the weights search, instead, NSGA-II is better for scenario S_1, and there is no statistical difference for the other scenarios. For the collision-and-weights search, NSGA-II is better in all the scenarios, except for scenarios S_1 and S_2 for which there is no statistical difference. These results allow to estimate the complexity of the problems tackled by the three searches. The collision search is a relatively easy problem, as RS can find solutions of the same quality of NSGA-II; indeed, finding an aggressive behaviour of a scenario car that leads to a collision is an easy task. The weights search per se is not an easy problem; however, as we will see in RQ1, the weights search will be often asked to solve difficult collisions that are unavoidable: for these "impossible" tasks, using NSGA-II or RS does not make any difference. On the other hand, collision-and-weights is a complex problem in which we don't simply want to find collisions, but these must be avoidable: for this, NSGA-II demonstrates to be much better.

C. Results

We analyze the results using two research questions.

RQ1 *Which approach is better in finding avoidable collisions?*

Here we compare the two approaches introduced in Sec. IV in terms of their ability to find avoidable collisions: the classical approach that performs test case generation and system parameter optimization one after the other (*sequential approach*), and the novel approach (*combined approach*) that performs them together. Experimental results are shown in Table III. For the *sequential approach*, the table shows the number of runs in which a collision has been found by the collision search, and then how many of these collisions have

TABLE III
EXPERIMENTAL RESULTS – AVOIDABLE COLLISIONS FOUND

ID	Sequential approach		Combined approach
	Collision search	Weights search	Collision-and-weights search
S_1	30/30	21/30	13/30
S_2	30/30	3/30	10/30
S_3	30/30	10/30	30/30
S_4	30/30	0/30	30/30
S_5	30/30	2/30	22/30
S_6	30/30	7/30	29/30
S_7	30/30	30/30	30/30

been proved to be avoidable by the weights search (i.e., the search was able to find alternative weights $\overline{w}'$ avoiding the collision). For the *combined approach*, the table shows the number of runs in which an avoidable collision has been found (recall that this search directly targets avoidable collisions).

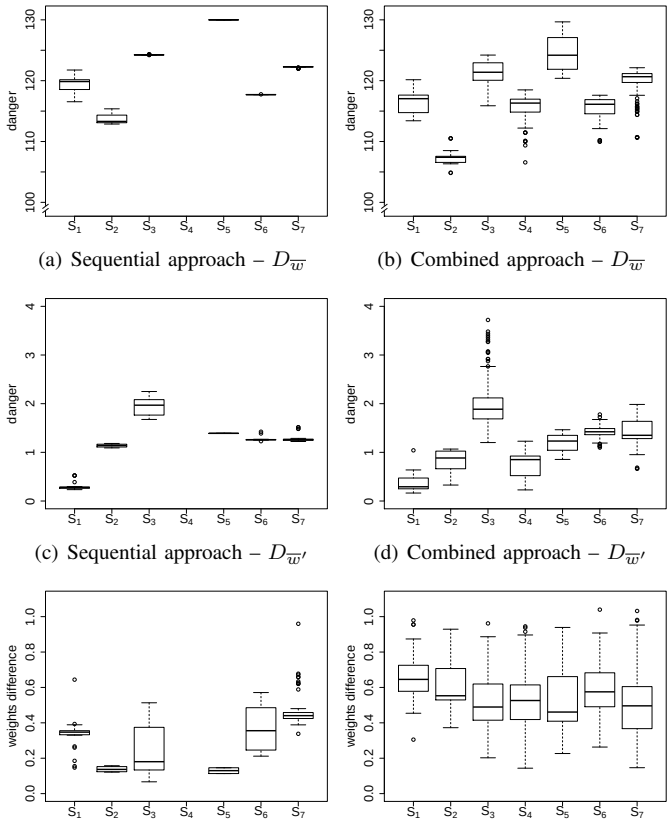
Regarding the *sequential approach*, we observe that finding collisions with the collision search is relatively easy, as it is always possible to find some aggressive behaviour of some other scenario car that leads to a collision. However, we observe that often these collisions cannot be proven to be avoidable by the weights search: this means that the collisions were so severe that there was no alternative path planner (at least in the defined search space) able to avoid them. Only for scenario S_7 all the collisions found are avoidable; for five scenarios, at most 10 out of 30 collisions are shown to be avoidable by the weights search (no solution is found for S_4).

Regarding the *combined approach*, we observe that it usually finds many more avoidable collisions than the *sequential approach* (scenarios S_2, S_3, S_4, S_5, and S_6): these are the cases in which the collision search of the *sequential approach* found too severe collisions that cannot be avoided, but there are less severe collisions that can. For scenario S_7, both the *sequential* and the *combined approaches* always find an avoidable collision; this could be due to the particular relative position of the ego car and the scenario car: at the intersection, the ego car can always stop or accelerate to avoid the collision.

Scenario S_1 is still at the intersection, but it is more complex than scenario S_7 because some stopped scenario cars occlude the sight of the ego car. For this scenario, the *sequential approach* is slightly better than the *combined approach*. We believe that, in this case, almost all the collisions that can be found in the search space are avoidable; since in the *sequential approach* the problem is decomposed in two simpler problems, its complexity is lower than that of the *combined approach* in which the two problems are solved at once. Therefore, the *sequential approach* can be advantageous only if the search space of a particular baseline scenario contains only avoidable collisions. However, as observed for the other baseline scenarios, in most of them it is likely to find unavoidable collisions.

RQ2 *Is there a difference between the avoidable collisions found by the two approaches?*

In RQ1, we observe that, in some cases, both the *sequential* and the *combined approaches* find an avoidable collision. Now, we want to compare the solutions found by the two approaches in these cases (i.e., the avoidable collisions AC_{SA} and AC_{CA} extracted from the Pareto fronts as described in Sec. IV-D).



(a) Sequential approach – $D_{\overline{w}}$ (b) Combined approach – $D_{\overline{w}}$

(c) Sequential approach – $D_{\overline{w}'}$ (d) Combined approach – $D_{\overline{w}'}$

(e) Sequential approach – Weights diff (f) Combined approach – Weights diff

Fig. 2. Comparison between AC_{SA} and AC_{CA}, i.e., avoidable collisions found by the *sequential* and the *combined approaches*

TABLE IV
COMPARISON BETWEEN AVOIDABLE COLLISIONS AC_{SA} AND AC_{CA} FOUND BY THE SEQUENTIAL AND THE COMBINED APPROACHES ($>$ (RESP. $<$): RESULT FROM SA IS STATISTICALLY SIGNIFICANTLY BIGGER (RESP. SMALLER) THAN THE RESULT FROM CA. $=$: THERE IS NO STATISTICALLY SIGNIFICANT DIFFERENCE.)

$D_{\overline{w}}$							$D_{\overline{w}'}$							weights difference						
S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_1	S_2	S_3	S_4	S_5	S_6	S_7
$>$	$>$	$>$	N/A	$>$	$>$	$>$	$=$	$>$	$=$	N/A	$=$	$<$	$<$	$<$	$<$	$<$	N/A	$<$	$<$	$<$

we want to compare the solutions found by the two approaches in these cases (i.e., the avoidable collisions AC_{SA} and AC_{CA} extracted from the Pareto fronts as described in Sec. IV-D).

Fig. 2 reports, for the two approaches and for all the scenarios, three measures of the solutions: the danger $D_{\overline{w}}$ obtained with the original system $\texttt{ADS}_{\overline{w}}$, the danger $D_{\overline{w}'}$ obtained with the modified system $\texttt{ADS}_{\overline{w}'}$, and the weights difference between the two systems. The boxplots report the distribution of the results across the runs that found at least one avoidable collision. Mann-Whitney U Test and A12 statistics are used to compare the results of the two approaches; tests results are reported in Table IV. Note that we cannot compare results for S_4, as the *sequential approach* did not find any avoidable collision. In the following, when we report that a measure is *higher/lower* in one of the two approaches or *equal*,

we refer to the result of the corresponding statistical test.

We observe that, for each scenario, the danger $D_{\overline{w}}$ obtained with the original system is always higher in the *sequential approach* than in the *combined approach*. This confirms our intuition that the collision search of the *sequential approach* tends to find more severe collisions than those found by the collision-and-weights search.

Regarding the danger $D_{\overline{w}'}$ obtained with the modified system, instead, there is no clear trend: it is lower in the *sequential approach* for S_6 and S_7, bigger in S_2, and equal in the other cases.

For the weights difference, the weights solving the collision in the *sequential approach* are closer to the original weights than those of the *combined approach*. This could be explained by the *difficulty* of the problems tackled in the two approaches. In the *sequential approach*, in which the two searches are separated, the problem complexity is much lower than in the *combined approach*, in which the two searches are performed at the same time and therefore there is a strictly larger number of search variables. As a consequence, for the *sequential approach*, in the few cases in which the collision found by the collision search is avoidable (see Table III), the weights search has enough time to find the minimum difference in weights that still avoids the collision. On the other hand, although the *combined approach* is able to find many more avoidable collisions, it usually does not find the one with a minimum weights difference, because it would need a significantly longer time to cope with the bigger search space.

D. Concluding Remarks

Based on the results of the research questions, we have the following key observations.

First, it was easier for the *sequential approach* to find collisions in general, since it was only required to find *any* collision. However, it wasn't able to find a higher number of *avoidable* collisions, as it can be seen from the results of the weights search. This is mainly due to the fact that it is easier to find severe collisions which are not necessarily avoidable.

Second, the *combined approach* found more avoidable collisions than the *sequential approach*. The only exception was the first scenario, where the *sequential approach* performed better than the *combined approach*. This is due to the fact that the search space of this scenario had a higher number of avoidable scenarios. Since the *sequential approach* breaks the overall problem into two sub-problems (i.e., collision search and weights search), it managed to find more avoidable collisions given the much smaller search spaces.

Third, danger values obtained with the original system with the *sequential approach* were always higher than the *combined approach*. This was due to the fact that the collision search of the *sequential approach* finds severe collisions that are unavoidable as compared to the collision-and-weights search of the *combined approach*.

Finally, in terms of weight difference, when the *sequential approach* found avoidable collisions, it was able to find closer weights to the original ones than the *combined approach*. As we observed in RQ2, this is due to the fact that the complexity of the *sequential approach* is lower than that of the *combined approach*. We can conclude that the higher complexity of the *combined approach* allows it to find many more avoidable collisions (which is our main aim), but it pays a price in terms of quality of the solutions.

VI. Practical Implications

Reducing Manual Effort for Finding Avoidable Collisions. Finding collisions for autonomous cars under varied environmental conditions is critical to ensure their safety. Manually finding realistic collisions is labor-intensive, inefficient, and error-prone and may even lead to missing severe collisions that could be avoided. Tools exist that can help finding collisions; however, they cannot determine if such collisions were realistic, nor if they would have been avoidable by a system configured differently. Therefore, in practice such collisions have to be manually checked by test engineers. To this end, our work aims to provide an automated way for test engineers working at our industry partner to automatically find avoidable collisions, reducing manual effort in determining whether a collision is avoidable or not. Moreover, our approach provides additional information about how the system could be configured to avoid such a collision, further reducing debugging efforts.

Better Understanding the Behaviours of Path Planners. Often test engineers do not have detailed knowledge about the inner working of the path planner they work with. Moreover, they have limited knowledge about possible known environmental conditions under which the path planners have to make decisions. Thus, testing is based on the assumptions on the environmental conditions that do not necessarily hold in reality. Therefore, it is essential to understand the behaviours of such components during their development as much as possible, so that their implementation can be made safe before their actual deployment in autonomous cars. Our tool can help test engineers understand the behaviour of path planners by assessing how changing weights of different aspects in the cost function of path planners can prevent collisions under various environmental conditions. Finding such weights is a key to improve the safety of these path planners, since it can be already evaluated during development and doesn't require an extensive knowledge about the system design.

Application to Other Scenarios and Path Planners. Our approach is by no means specific to the scenarios we experimented with and the path planner provided to us by our industry partner. It can be applied to other scenarios of the provided path planner and to other path planners. However, the application to other path planners will require adapting the implementation of the approach.

Limitations. Our work has demonstrated promising results; however, it still has several limitations that we plan to deal with in the future. First, we experimented with one scenario at a time, and we need to extend the approach for multiple scenarios. Second, we experimented only with NSGA-II, and there is a possibility that other algorithms may provide even

better quality solutions [16]. Third, finding avoidable collisions aims at improving safety; however, it doesn't ensure the safety. Thus, our approach should be followed by the application of other approaches that aims at checking whether a system behaves as suggested by safe models [27], [28], [30]. Fourth, we need approaches to *explain* why a collision is actually avoidable: to this aim, we could analyse how the values computed by the cost function change in the modified system.

VII. Threats to validity

This section discusses the threats to validity of our experiments based on the classification of threats defined in [35].

Conclusion Validity. When dealing with search algorithms, their randomness must be accounted for while drawing conclusions from the results. By following the guidelines of reporting results for randomized algorithms [2], we repeated our experiments 30 times for both NSGA-II and RS to ensure that the results are not obtained by chance. Moreover, we compared the two algorithms using Mann-Whitney U Test and A12 statistics by following the same guidelines [2]. Moreover, to have a fair comparison between *sequential* and *combined approaches*, we used the same search space size for scenarios fields for collision search and collision-and-weights search, and weights in weights search and collision-and-weights search.

Construct Validity Threats. When designing the experiments involving the search algorithms, appropriate measures must be chosen to compare various algorithms. To this end, to compare the results of NSGA-II and RS, we chose the HV quality indicator by following the guidelines provided in [34]. We chose NSGA-II since it is one of the most commonly used multi-objective search algorithms. However, we admit that more experiments with additional algorithms are needed. In addition, to avoid any bias towards any specific algorithm, we use the same stopping criterion for NSGA-II and RS, i.e., the number of fitness evaluations. For the *sequential approach*, we set the number of fitness evaluations to 2400, whereas for the *combined approach* we set this to 1200. This is due to the reason that for each fitness evaluation for the *combined approach*, we invoke the path planner twice, whereas for the *sequential approach* the path planner is invoked once for each fitness evaluation for collision and weights searches. Thus, the total time for both approaches is comparable.

Internal Validity. In the context of search algorithms, the most relevant threats are related to the selection of the algorithms and their parameter settings. We chose NSGA-II since it is a widely used algorithm in SBSE. However, we plan to investigate other relevant algorithms in the future. Moreover, we chose the default parameter settings of NSGA-II from jMetal. We are fully aware that different parameter settings can lead to different quality of results; however, it has been suggested [3] that even default parameter settings of the algorithms lead to good quality results.

External Validity. Like any experiment, our experiments have external validity threats. We used seven scenarios for our experiments, and we definitely need more scenarios (e.g. discussed in [18], [38] to generalize the results to a wider context). Moreover, we also admit that we need to experiment with different path planners to further generalize the results.

VIII. Related work

There is a rich literature of related work, both from the perspective of testing and of various kinds of driving-related systems. The starting point of the research is contained in [9]–[11], where various kinds so-called Advanced Driver Assistance Systems (ADAS) have been tested. Improvements have been reported in [6]–[8], [31], [32]. While the testing procedures are comparable, the system provided by our industry partner is not an ADAS, but an ADS. As the different levels of automation [29] suggest, there are crucial differences between assistance systems of level 1&2 and automated and autonomous driving systems of level 4&5. Technically speaking, the assistance systems are oftentimes rather simple control loops, while the more complex systems contain dedicated decision, planning and control components, making design, development, and testing much more difficult. Further, while [6], [7] search for collisions, those collisions may be not avoidable and therefore not of great use, since for some collisions the ADS could not do much to avoid them. Other works [1], [5], [17], [21] provide a basis within the domain in which we conducted our research. These works provide various methodological and technical ideas, which involve search-based test scenario generation. They all yield safety-critical test scenarios. However, none of them provides a suitable system configuration to improve on failed test scenarios. Therefore, the design and development team receives no additional information about how the system can be improved. Works in [13], [25] come from different domains but are still relevant due to their concern with search-based scenario generation.

IX. Conclusions

When testing automated and autonomous driving systems in virtual environments, collisions between vehicles cannot be claimed to be failures without access to a complete specification or extensive domain knowledge. In that regard, this paper introduces a definition of *avoidable collision* that depends on neither. The subject under analysis is a path planner component, provided by our industry partner, which can be re-parametrized by changing the so-called 'weights'. By first identifying a dangerous collision using search-based techniques and subsequently searching for alternative weights which are able to avoid it, it can be shown that there is at least one situation where the original ones are not optimal.

Another issue is that approaches for finding such avoidable collisions, as the one described above, are not trivially correct or efficient. We have shown examples where the search for dangerous collisions is only able to find unavoidable ones, which offer no opportunity to identify a (potentially) better set of weights, or any useful insight into how the system behaves. For this reason, this paper also introduces a search-based methodology, referred to as the *combined approach*.

In it, scenario variables and system parameters are changed simultaneously, searching directly for collisions which are avoidable — for a given scenario, they occur under the original weights, but not under alternative ones.

These results are valuable for testing engineers: every collision can automatically be determined to be an actual failure or not. Moreover, they are valuable to system designers: every collision comes with an example of an alternative parametrization which avoids it, which at the very least gives them useful insight into how the system could optimally behave.

References

[1] M. Althoff and S. Lutz. Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1326–1333. IEEE, 2018.

[2] A. Arcuri and L. Briand. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 1–10, New York, NY, USA, 2011. ACM.

[3] A. Arcuri and G. Fraser. On parameter tuning in search based software engineering. In *Proceedings of the Third International Conference on Search Based Software Engineering*, SSBSE'11, pages 33–47, Berlin, Heidelberg, 2011. Springer-Verlag.

[4] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo. The oracle problem in software testing: A survey. *IEEE Transactions on Software Engineering*, 41(5):507–525, May 2015.

[5] H. Beglerovic, M. Stolz, and M. Horn. Testing of autonomous vehicles using surrogate models and stochastic optimization. In *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*, pages 1–6. IEEE, 2017.

[6] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter. Testing advanced driver assistance systems using multi-objective search and neural networks. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, ASE 2016, pages 63–74, New York, NY, USA, 2016. ACM.

[7] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter. Testing vision-based control systems using learnable evolutionary algorithms. In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, pages 1016–1026, New York, NY, USA, 2018. ACM.

[8] R. Ben Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter. Testing autonomous cars for feature interaction failures using many-objective search. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, pages 143–154, New York, NY, USA, 2018. ACM.

[9] O. Bühler and J. Wegener. Evolutionary functional testing of an automated parking system. In *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.

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

[11] O. Bühler and J. Wegener. Evolutionary functional testing. *Computers & Operations Research*, 35(10):3144–3160, 2008.

[12] K. Deb, A. Pratap, S. Agarwal, and T. A. M. T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[13] J. Deshmukh, M. Horvat, X. Jin, R. Majumdar, and V. S. Prabhu. Testing cyber-physical systems through bayesian optimization. *ACM Trans. Embed. Comput. Syst.*, 16(5s):170:1–170:18, Sept. 2017.

[14] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia. VerifAI: A toolkit for the formal design and analysis of artificial intelligence-based systems. In I. Dillig and S. Tasiran, editors, *Computer Aided Verification*, pages 432–442, Cham, 2019. Springer International Publishing.

[15] J. J. Durillo and A. J. Nebro. jMetal: A Java framework for multi-objective optimization. *Adv. Eng. Softw.*, 42(10):760–771, Oct. 2011.

[16] R. Feldt and S. Poulding. Broadening the search in search-based software testing: It need not be evolutionary. In *Proceedings of the Eighth International Workshop on Search-Based Software Testing*, SBST '15, pages 1–7, Piscataway, NJ, USA, 2015. IEEE Press.

[17] F. Hauer, A. Pretschner, and B. Holzmüller. Fitness functions for testing automated and autonomous driving systems. In *International Conference on Computer Safety, Reliability, and Security*, pages 69–84. Springer, 2019.

[18] F. Hauer, T. Schmidt, B. Holzmüller, and A. Pretschner. Did we test all scenarios for automated and autonomous driving systems? In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2950–2955, Oct 2019.

[19] T. Helmer, L. Wang, K. Kompass, and R. Kates. Safety performance assessment of assisted and automated driving by virtual experiments: Stochastic microscopic traffic simulation as knowledge synthesis. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, pages 2019–2023. IEEE, 2015.

[20] N. Kalra and S. M. Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 2016.

[21] S. Khastgir, G. Dhadyalla, S. Birrell, S. Redmond, R. Addinall, and P. Jennings. Test scenario generation for driving simulators using constrained randomization technique. Technical report, SAE Technical Paper, 2017.

[22] P. Koopman and M. Wagner. Challenges in autonomous vehicle testing and validation. *SAE Int. J. Trans. Safety*, 4:15–24, 04 2016.

[23] T. Laurent, P. Arcaini, F. Ishikawa, and A. Ventresque. A mutation-based approach for assessing weight coverage of a path planner. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, pages 94–101, Dec 2019.

[24] J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, and A. Marrs. Disruptive technologies: Advances that will transform life, business, and the global economy. https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/disruptive-technologies, Last accessed: October 1, 2019.

[25] G. E. Mullins, P. G. Stankiewicz, and S. K. Gupta. Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles. In *IEE International Conference on Robotics and Automation (ICRA)*, pages 1443–1450, 2017.

[26] A. J. Nebro, J. J. Durillo, and M. Vergne. Redesigning the jMetal multi-objective optimization framework. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pages 1093–1100, New York, NY, USA, 2015. ACM.

[27] D. Nister, H.-L. Lee, J. Ng, and Y. Wang. The safety force field. online at https://www.nvidia.com/content/dam/en-zz/Solutions/self-driving-cars/safety-force-field/the-safety-force-field.pdf, retrieved 9th September 2019.

[28] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, E. Hilgendorf, and T. Nipkow. Formalising and monitoring traffic rules for autonomous vehicles in Isabelle/HOL. In N. Polikarpova and S. Schneider, editors, *Integrated Formal Methods*, pages 50–66, Cham, 2017. Springer International Publishing.

[29] SAE. Definitions for terms related to on-road motor vehicle automated driving systems. *J3016, SAE International Standard*, 2014.

[30] S. Shalev-Shwartz, S. Shammah, and A. Shashua. On a formal model of safe and scalable self-driving cars. *arXiv:1708.06374*, retrieved 9th September 2019.

[31] M. Tatar. Test and validation of advanced driver assistance systems automated search for critical scenarios. *ATZelektronik worldwide*, 11(1):54–57, 2016.

[32] T. E. Vos, F. F. Lindlar, B. Wilmes, A. Windisch, A. I. Baars, P. M. Kruse, H. Gross, and J. Wegener. Evolutionary functional black-box testing in an industrial setting. *Software Quality Journal*, 21(2):259–288, 2013.

[33] W. Wachenfeld and H. Winner. The release of autonomous vehicles. In *Autonomous Driving*, pages 425–449. Springer, 2016.

[34] S. Wang, S. Ali, T. Yue, Y. Li, and M. Liaaen. A practical guide to select quality indicators for assessing pareto-based search algorithms in search-based software engineering. In *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, pages 631–642, New York, NY, USA, 2016. ACM.

[35] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

[36] C. Zhang, Y. Liu, D. Zhao, and Y. Su. Roadview: A traffic scene simulator for autonomous vehicle simulation testing. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1160–1165, Oct 2014.

[37] D. Zhao and H. Peng. From the lab to the street: Solving the challenge of accelerating automated vehicle testing. *CoRR*, abs/1707.04792, 2017.

[38] J. Zhou and L. del Re. Reduced complexity safety testing for ADAS & ADF. *IFAC-PapersOnLine*, 50(1):5985–5990, 2017.