UNIVERSITÄT PASSAU

PASSAU UNIVERSITY

BACHELOR THESIS

Comparing novelty search and multi-objective search for testing self-driving car software

Author:
Simon FIERBECK

Supervisor:
Prof. Dr.-Ing. Gordon FRASER

Advisor:
Alessio GAMBI, Ph.D.

Monday 8th July, 2019

Contents

Zusammenfassung

Eines der größten Probleme für den Einsatz von autonomen Fahrsystemen ist die Gewährleistung der Sicherheit. Aufgrund der hohen Anzahl an möglichen Test-Scenarien, welche aus der großen Vielfalt an möglichen Interaktionsmöglichkeiten mit der Umgebung resultieren, ist das ausschließliche Testen durch kostspielige Feldtests nicht möglich und deshalb ist der Einsatz von Simulationstests weit verbreitet. Als Konsequenz der benötigten Zeit und Rechenleistung um die Testfälle zu generieren und auszuführen, ist es erstrebenswert, dass wenige Test-Scenarien einen großen Bereich der möglichen Testparameter abdecken. Das ist der Grund, warum die Unterschiedlichkeit der Test-Scenarien ein wichtiger Punkt ist um die Anzahl an auszuführenden Testfällen zu verringern. Deshalb vergleiche ich in dieser Arbeit zwei Ansätze, die darauf abzielen, diese Unterschiedlichkeit zu maximieren: Novelty Search und Multi-Objective Search. Das Ergebnis dieser Arbeit zeigt, dass Multi-Objective Search effektivere Testfälle generiert, obwohl dass die beiden Ansätze eine ähnlichen Verteilung bezüglichen dem Auftreten von Fehlern des Test-Subjekts haben.

Abstract

Evaluating the safety of autonomous driving systems is one of the biggest obstacles for the deployment of these systems. Because of the high number of test scenarios, which arouses through the huge variety of possible interactions with the environment, relying only on expensive real-life testing is not practical and therefor, simulation testing is been widely used. As a consequence of the needed time and computation power to generate and run a test case, it is desirable, that few test scenarios cover a wide range of the possible testing parameters. This is why diversity of the test scenarios is an important element to consider to decrease the number of test case executions. I compare in this thesis two approaches which aim to maximize diversity: novelty search and multi-objective search. The result of this thesis shows that multi-objective search generates more effective test cases, despite both have a similar distribution regarding where the test subject fails.

1 Introduction and Motivation

"Can you imagine, you're sitting in the back seat and all of a sudden this car is zig-zagging around the corner and you can't stop the fucking thing?" [14]. This quote is from the American President Donald Trump, the leader of one of the most advanced countries in the world in the field of self driving cars. While his statement aligns with 71 percent of America's population, who doesn't fully trust self driving systems [13], more and more companies, ranging from well known automakers to tech-companies and young startups, are developing, testing and shipping their own autonomous driving systems [3]. There is a huge wariness toward these systems, mainly because they have to operate independently and safely for the passengers, despite the unknown environment. For example, self-driving cars must account for natural elements, such as lighting and weather conditions, unpredictable humans, and other traffic participants. To ensure the safeness of self-driving cars, they have to be tested in as many different scenarios as possible. In situations, where the autonomous driving system hasn't been tested, unforeseen errors can happen, which could lead to incidents and the loss of human life. Because of the time and execution power needed to run a test case, it is not practical to test the autonomous driving system in every possible scenario and therefore, the challenge is to generate an adequate test suite. In this thesis, I investigate the generation of test cases for self-driving car software which aim to achieve diversity. In particular, I implement a test generation technique using novelty search based on the novelty search of Lehman, which was intended to solve a labyrinth [8]. Also I re-implement the existing multi-objective search technique of Loiacono et al., which was designed to increase the enjoyment of roads in a video game [10], for generating test cases. The reason I choose these two test generation techniques is because both of these approaches focus on finding diversity in their own way. There are other test case-generation approaches, which focus on maximizing other aspects of the test case. AsFault focuses on maximizing the distance between vehicle and the centre of the lane [7]. That means that AsFault promotes those test cases, which expose possible problems in self driving car software. Ben Abdessalem et al. propose a minimum time to collision fitness function, in which the time to hit an obstacle is measured [2]. Here, the safety aspect is prioritized. But instead of focusing directly on safety-critical properties, other aspects, which stress the system in different ways, should be considered to tackle the above mentioned problems with autonomous driving systems. Therefore, through maximizing diversity of the test cases for the system, the robustness of autonomous driving systems could be evaluated in more diverse scenarios. The code of my tool, the data and instructions for replicating this study are available at 'https://github.com/Fierl/ComparingNoveltyMulti-Objective'.

1.1 Problem statement

The overall goal of maximizing diversity is to test the system in as many ways as possible. As the autonomous driving system controls the steering, acceleration and throttling of the car, the aim is to stress these aspects. One way to achieve this is by maximizing the variety of road segments inside the test case itself. Naturally, a long straight is different compared to a sharp right curve and as a consequence, the action taken by the autonomous driving system is different as well. Therefore, maximizing the diversity inside the test case will test the system in many different ways regarding the composition of road segments. Another way to maximize diversity for testing autonomous cars can be done by maximizing the difference inside the test suite, i.e. making tests, which are different from one another to stress the system under different conditions. Hence, maximizing the variety inside a test suite will test the system in many different ways regarding the overall shape of the road. Both approaches maximize the diversity of tests for testing autonomous driving systems in their own way and because of that, should be able to evaluate the robustness of these systems. Therefore, in this thesis I study the problem of generating test suites, which aim to maximize stressing the test subject in different ways and because of that produce different actions of the test subject.

2 Background

2.1 Genetic algorithm

The novelty search approach and the multi-objective search, which are the two approaches for testing self-driving cars that I consider in this thesis, are both based upon genetic algorithms. Genetic algorithm is a method for solving optimization problems, which uses techniques of the natural evolution to compute the results [16]. These techniques contain the ability to initialize a population, generate offspring, evaluate the individuals with a given function and select the best individuals out of the population. This four evolutionary tools in combination with iterative steps can propagate a certain feature and produce results, which are maximized in specific aspects. The novelty search focuses not on maximizing the property of the individual itself, rather it maximizes the distance between the property values. The multi-objective search is a special form of generic algorithm, because instead of maximizing one property of the individual, it maximizes multiple properties at once. In this thesis the optimization problem is to find diversity in roads. I use the Distributed Evolutionary Algorithms in Python (DEAP) framework for the implementation of these methods. The DEAP framework's advantages are the ability to control every step of the evolutionary aspect [12] and to produce prototypes fast [6].

2.2 Procedural content generation

The random creation of the roads is based on procedural content generation. Procedural content generation has been used in video games since the early 1980 and was at the beginning mainly used to overcome the lack of memory space by creating the content instead of saving it and nowadays is more often used to generate personalized gameplay content [10]. The procedural content generation algorithm usually produces certain content based on defined parameters in combination with randomness [15]. In addition to this, the field of search-based procedural content generation has developed. The search-based procedural content generation algorithm is shifting it's focus from just creating the content to evaluating the generated content based on specific aspects [15]. This generate-and-evaluate approach is able to automatically assess the generated content, which I use in my thesis to determine the diversity of a road.

2.3 NSGA-II

The selection of the best individuals is based on the non-dominated sorting genetic algorithm II (NSGA-II). Maximizing the speed and curve values of a road is a multi-objective optimization problem and possibly can have a set of optimal solutions also known as Pareto-optimal solutions [5]. Finding the solution, which dominates the other solutions, is not a trivial task. NSGA-II has two main advantages for this kind of problem: It is using elitism, which is evaluating the population through non-dominated sorting in combination with assigning a crowding-distance and can be easily applied, and it doesn't require a specified sharing parameter, which is needed in most other optimization methods, to ensure diversity of the population [5].

2.4 BeamNG

For simulating the generated roads and for testing them against the test subject BeamNG is used as the driving simulator. The game engine of BeamNG is physically-accurate, in terms of movement and trajectory, and there are extensive settings for the BeamNG-AI, which is the non-human-driver of BeamNG, regarding the acceleration and speed. BeamNG also provides a programming interface, which gives access to detailed information, i.e. speed and position, of the simulated vehicle and BeamNG is also able to run programmatic generated roads.

Manual
Risk: 1.5 In Lane: on C/O Drivability: 0
d km/h: 0 Mode: off
Map: Vehicle:
e Map: Vehicle: Disabled

(a) (b)

Figure 1: Screenshots of BeamNG

2.5 DeepDriving

For this thesis, DeepDriving, which was developed by Chen et al. [4], and it's implementation DeepDrive from Andre' Netzeband are used as the test subject. Many of the autonomous driving systems can be grouped into either 'Mediated perception approaches', where a decision is made by computing the information given through the sensors, or 'Behavior reflex approaches', where a decision is based on trained and similar knowledge [4]. DeepDriving is a combination of both of them. It learns decisions based on training input as an indicator, but uses a rule based controller to compute the action. [4] Therefore, this state of the art autonomous driving system is a good test subject.

2.6 Statistical difference & effect size

To evaluate the relevance of my results I compute the statistical difference and the effect size of the outcome. As suggested by Arcuri and Briand in their paper about comparing randomized approaches [1], I use two empirical tests, because both focus on something different than the other. The statistical difference focuses on the degree of difference between two sets of numbers. This is done by formulating a null hypothesis (H_0), i.e. 'There is no difference between the two sets', and rejecting it by computing the p-value. To reject H_0 you have to consider the two types of statistical errors, rejecting a true H_0 and accept a false H_0. The focus of statistical difference lies on rejecting a true H_0. As a result of this comparison you get the p-value, which is the probability of rejecting a true H_0. In science $\alpha = 0.05$ is used as the standard significance level, for which higher p-values are rejected and smaller p-values are accepted. The effect size of two sets of values is used to determine the degree of difference in terms of superiority. In this thesis, Vargha and Delaney's A_{12} statistic is used to compute the effect size of the outcome. The A_{12}-value gets computed by ranking the values inside the sets and looking at the distance between each value-pair, i.e. comparing each value of one set with their respective counterpart of the other set. The resulting A_{12}-value determines which one is more effective, for example a A_{12}-value of 0.7 means, that the first set would produce better results in 70% of the time. For both statistical methods 30 executions would be desirable to claim statistically significance. [1].

3 Related Work

Through the need to use simulations and the complexity of the test cases of the autonomous driving systems, many different approaches to verify the safeness of these systems in critical situations have been developed over the years.Among the many, I focus on the multi-objective search of Loiacono et al. and on the novelty search of Lehman, which I compare in this thesis, as well as on the work of Müller, on which my software is mostly based on. I choose the multi-objective search of Loiacono et al. and a novely search based on the novelty search of Lehman, because they both are specialized in finding diversity through different approaches.

Both approaches, which I compare in my thesis, are focused on maximizing the diversity of the test cases, which are in this case the generated roads. Loiacono et al. reason in their paper, that one of the main reasons for a successful game, if it's not event based, is the diversity of the tracks. While driving on one type of road-segment gets repetitive and boring for the human player, driving on a more variety in the type of road-segment and in the combination of those segments leads to a more interesting experience for the human player [10]. Because of that, they suggest to generate diverse tracks using the speed and curve profile as fitness functions [10]. This multi-objective search is done by computing the fitness-value of the curvature and speed proles the track through splitting it up into a specified number of segments, then each segment is calculated independently and after that, the overall values get computed [10]. Moreover, the tracks are recombined with each other for a certain time budget, where the better evaluated tracks are selected and the worse evaluated tracks get discharged. Therefor, the resulting tracks are various in form and gameplay-properties, like achievable speed [10]. While Loiacono et al. uses this approach to satisfy the players of a game, I use it to generate diverse test cases for an autonomous driving system. From a non gameplay-based perspective, diversity in roads can lead to unknown behaviour for the autonomous driving systems. As observed for human players, a limited number of different types of road-segments and combinations of road-segments is less challenging than a higher number [10]. Therefore, a more diverse road, i.e. has a variety of different needed velocities and types of road-segments, should lead to more actions performed by the autonomous driving system. Hence, I take their approach as an example for a multi-objective search aiming to maximize diversity in the field of track generation.

In the book 'Evolution Through The Search For Novelty ' Lehman demonstrates his approach by solving mazes and comparing it against a 'distance to objective' - function [8]. The idea behind this approach is rewarding novelty instead of an objective value [8]. Lehman argues, that a fitness function designed to maximize the objective-value can get lost into local maximas, where novelty search is designed to overcome this issue [8]. The measurement of diversity in the novelty approach is done by looking at the distance between the neighbouring elements and group them based on this measurement [8]. For the novelty search algorithm more different neighbours are more suitable to be selected [8]. The author demonstrates, that in the maze scenario, the objective fitness function does take significant more time to compute the final result as the novelty search algorithm does [8]. This is mainly due to the fact, that novelty search bypasses the possibility of local maxima [8]. For the multi-objective search local maxima are also to consider. In the field of generating roads for testing autonomous driving systems, a certain road shape with a high fitness value could get propagated, which would decrease the variety in failed road-segment types. Because of that, novelty search based on novelty search of Lehman, which maximizes the distance between the properties of the test cases, is a good counterpart to the multi-objective search of Loiacono et al., while it also maximizes diversity.

The two chosen methods to maximize diversity have different approaches to achieve this. While multi-objective search of Loiacono et al. maximizes the fitness values, in this case, the speed- and curvature-profile of a road, which resemble the diversity of the road, novelty search based on novelty search of Lehman maximizes the distance between the the profile-values of the roads, which resembles the overall shape of the road. If diversity leads to more failures of the test subject, both approaches should be able to generate difficult roads in their own way.

The master-thesis of Müller 'Evolutionary Test Generation for Autonomous Vehicles' is about generating roads respectively test cases based on a single-objective fitness function [11]. His software, AsFault, generates roads of the initial population through procedural content generation, then evaluates them by testing an autonomous driving system inside a simulation. After that, selects the best roads as parents through tournament selection and finally, uses them to create the offspring for the next iteration [11]. Because of their different approaches towards genetic algorithms, I alter the evolutionary steps for the novelty search and the multi-objective search. But beside the evolutionary aspects, AsFault in combination with BeamNG provides an excessive set of tools to create, modulate and simulate roads. My work builds on top of AsFault. It uses Müller's AsFault for producing the initial population for the novelty search and the multi-objective search and to verify the roads, which are created through search operators. But while his single-objective fitness function is focusing on maximizing the distance between the autonomous driving system and the middle of the road respectively the rate for which the car gets out of the road [11] and, through that, propagates test cases with a high number of oob's, my aim is to maximize the difference between the road-segments of the road itself, by using the multi-objective search of Loiacono et al., and to maximize the difference between each test case of a test suite, which the novelty search algorithm of Lehman is focused on. The focus of Müller's single-objective search is to maximize the failure rate of the ego-vehicle, while the multi-objective search and the novelty search maximize the diversity of the road itself.

Lehman and Loiacono et al. have demonstrated that they can produce results in their field. But in this thesis I use their methods in a field, for which neither of them was designed for, and test them against a state of the art autonomous driving system. To achieve this I implement those approaches on top of AsFault. Through this I show, weather these methods are practical in terms of time to compute results and if maximizing diversity is useful for testing autonomous driving systems.

4 Methodology

The overall structure for novelty search and multi-objective search is as follows: Both methods start by generating a specific number of roads, which serves as the initial population. After that, an offspring based on this population gets generated. Next, the roads of the population and the offspring are evaluated based on achievable speed and shape. Then the most diverse roads, which novelty search and multi-objective search define different, are selected for the next iteration. Through this, both approaches evolve their population. After a certain time budget is met, the results of both approaches are tested against the test subject.

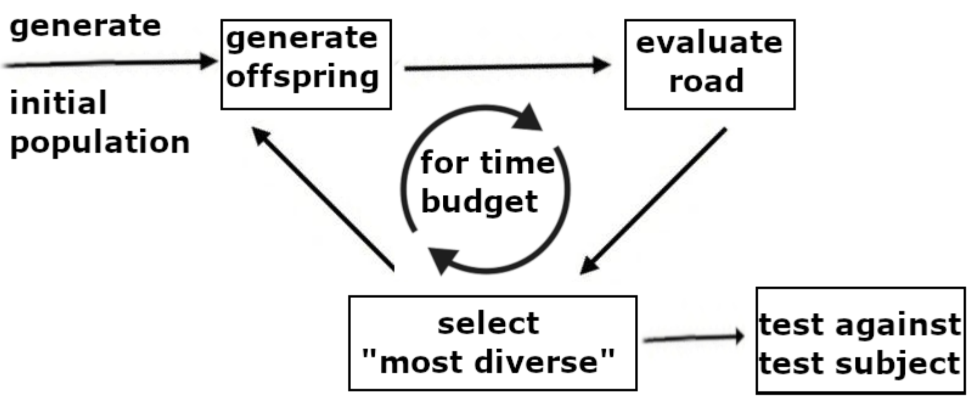


Figure 2: Overall model of both approaches

4.1 Generating and evaluating roads

4.1.1 Generating roads through AsFault

The novelty search as well as the multi-objective search require both roads to operate. For this experiment only single roads without intersections are used in the form of dictionaries, which describe the number of road-segments, their type and length. Road-segments, which represent curves, have a pivot- and an angle-property, while road-segments, which represent straights, have a length-property. I don't consider intersections and road-networks in this experiment, because my focus is on the diversity of test suites/-cases through the distribution of road-segments and the influence of intersections could distort the AI's ability to drive. These single roads are generated through AsFault. AsFault is able to verify the generated roads by examining them for intersections and if the start- and end-point are at the map boundaries. Any road which has intersections or has different start- and ending points is immediately discarded. Moreover, I can specify the overall size of the map, in which the road is generated, and the width of the road itself through AsFault. I choose a map size of 400 m^2 and a road width similar to the road width of US roads, which is used by BeamNG and DeepDriving.

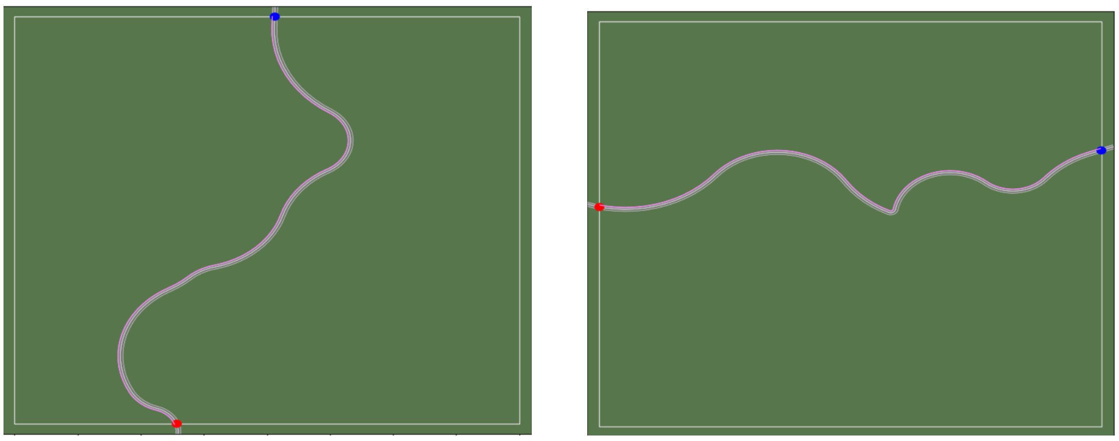

(a) (b)

Figure 3: Sample roads generated by AsFault. Roads start at red dot and finish at blue dot.

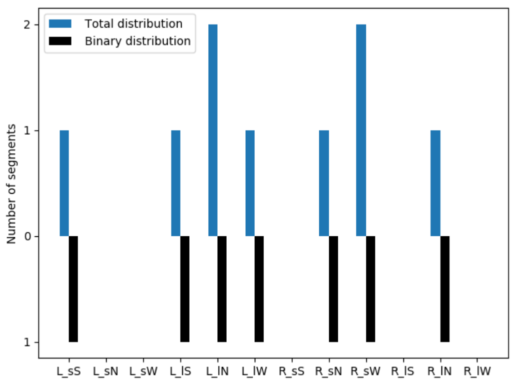
4.1.2 Evaluating the generated roads

The evaluation of the roads is needed to determine the value of the test case for further evolution. To do that, I evaluate each road-segment based on their curvature-profile, i.e. the shape of the road, respectively their speed-profile, which means the highest achievable speed on this segment. The curvature-profile is used to indicate how much the test case is stressing the steering command of the test subject, while the speed-profile indicates how much the test case is stressing the accelerate and break commands. It's important to mention, that these two profiles are not necessary correlated, which means that the same curvature-profile can result in different speed-profiles based on the order of the road-segments, but they affect each other, i.e.

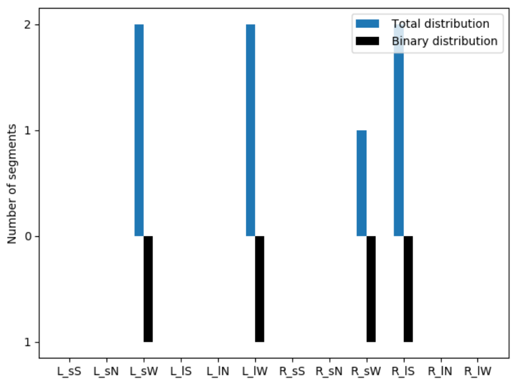
a road with a high curvature-profile has likely a lower speed-profile and vice versa [10]. The curvature- and speed-profile are computed based on the entropy of the curvature- respectively speed-distribution [10]. To generate the profile-values, the evaluated segments are distributed into twelve bins, which resemble a specific property range. Loiacono et al. use 16 bins in their work, but this was not optimal for the property-range of the road-segments generated by AsFault. Therefore, twelve bins are used as a compromise between the number of bins Loiacono et al. use and the property-range of the road-segments. While the profile-values or H-values of Loiacono et al. are maximized, when each bin values occupy the same percentage of the road [10], I compute the profile-values by using the binary-distribution of the bins. A binary distribution means that no matter how many values are in the bin, it only counts as one. As mentioned above, the order of the road-segments affect the value of the speed-profile, and therefore, the curvature-profile used in this thesis doesn’t decrease if the bin-value of one type of road-segment is greater than the bin-value of another.

4.1.2.1 Curvature profile The curve-profile is based on the number and types of segments of the road. The overall types of road-segments are left curve, right curve and straight. Left and right curves have a pivot- and an angle-property, while straights have a length-property. The number of bins for the curvature-profile arises through splitting up the possible pivot range by two (in figure 4 ’X_sX’ for short and ’X_lX’ for long) and the angle range by three (in figure 4 ’X_xS’ for sharp, ’X_xN’ for neutral and ’X_xW’ for wide). This is done for right turns and left turns (in figure 4 ’L_xX’ for left turn and ’R_xX’ for right turn). Every segment is put in their intended bin, which resembles a curvature type. By calculating the binary-distribution, roads, which have a lot of different road-segments, get a higher curve-value than roads with a less variety in the types of road-segments.

$$H(C) = 1/N * \sum_{i=1}^{N} c_i$$



(a) Road-segment distribution of figure 3 (a)



(b) Road-segment distribution of figure 3 (b)

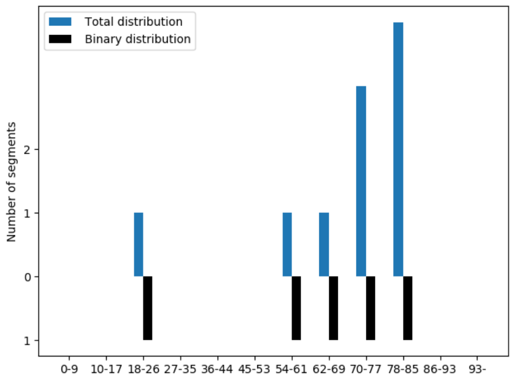
Figure 4: Distribution of road-segments

$H(C)$ is the entropy value of the curvature-profile and it’s calculated by counting the number of filled bins c_i and divide them through the overall number of bins N [10]. The curvature-value of (a) in figure 3, which is represented in figure 4 (a), is calculated by $(1/12) * 7 = 0.58$.

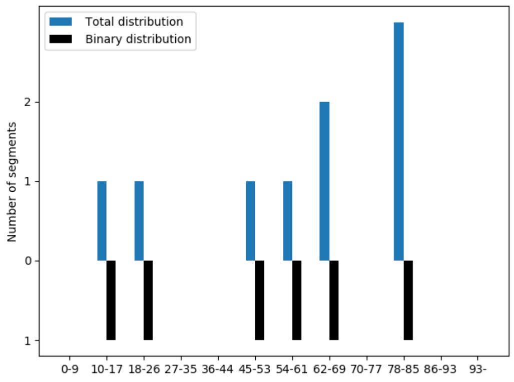
The curvature-value of (b) in figure 3, which is represented in figure 4 (b), is computed by $(1/12) * 4 = 0.33$.

4.1.2.2 Speed profile The speed-profile is based on the highest speed, for which a defined artificial intelligence is able to drive a simulated car, further referred to as 'ego-vehicle', on the road without getting out of the road boundaries. For both approaches the BeamNG-AI is used to determine the speed-profile. In the work of Loiacono et al. the ego-vehicle drives the track for three laps, where either the second lap-time or the best time is used. But because the test subject starts at zero miles per hour, BeamNG-AI also starts at zero miles per hour and drives the road only once. The ego-vehicle drives the road, which has to be evaluated, from start to finish and for every frame of the simulation the speed of the vehicle is saved. These speed values are assigned to the road-segments from which they are taken and a mean-value, which resembles the road-segment as a whole compared to the minimum- or maximum-speed-value, of the segment gets calculated. When the ego-vehicle drives too fast and gets off the road, the simulation is repeated and the speed is decreased through a fixed decrement in this specific segment until the ego-vehicle can drive the segment without getting of the road. Then, the calculated mean value of each segment is put in their intended bin. The number of bins for the speed-profile matches the number of bins of the curvature-profile to facilitate the comparison. The speed-profile is calculated as following:

$$H(S) = 1/N * \sum_{i=1}^{N} s_i$$



(a) Speed-value distribution of figure 3 (a)



(b) Speed-value distribution of figure 3 (b)

Figure 5: Distribution of speed-values

Where $H(S)$ is the entropy value of the road, N is the number of bins and s_i is the binary value of a specific bin [10]. The speed-profile of (a) in figure 3, which is represented in figure 5 (a), is computed by $(1/12) * 5 = 0.42$. The speed-profile of (b) in figure 3, which is represented in figure 5 (b), is calculated by $(1/12) * 6 = 0.5$. Unfortunately, BeamNG-AI has a bug regarding a specific shape of a curve, which resolves in the ego-vehicle cutting the road, i.e. getting of the road, even while driving at 1 miles per hour. In this case, the segment gets the default value of 12 miles per hour. I drove ten different road-segments, where this bug happened, manually and

12 miles per hour is the minimum value for which I was able to drive them without going off the road. I use the minimum value of the manually produced speed-values to ensure that the segment is drive-able with this speed.

4.2 Evolving test suite/-cases

4.2.1 Basic structure

The novelty search as well as the evolutionary algorithm by Lehman are both based on genetic algorithms, which means they both follow a certain methodology. The general procedure of both methods is to generate a initial population, i.e. roads, produce children through search operations, evaluate the initial population and the newly generated offspring and finally, select the best individuals for further evolution. This is done until a certain time budget is met.

4.2.2 Search operator

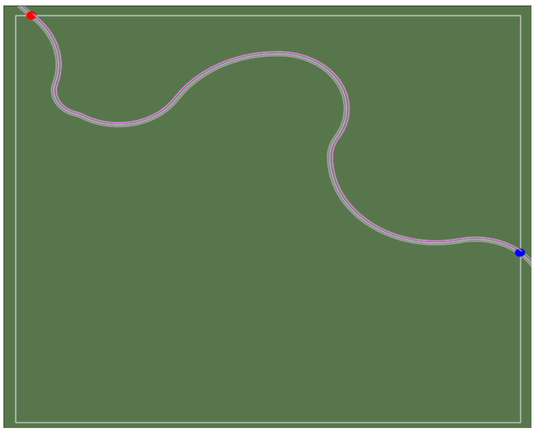
To evolve the generated roads I use two types of evolutionary operation: crossover between two roads and mutation of a single road. The crossover is applied to any individual of the population by combining two individuals randomly. It is done by splitting two roads in half, i.e. take the first respectively last half of all road-segments, and join the first half of one road with the last half of the other and vice versa. After this is done, the newly generated roads are examined by AsFault regarding intersections and drive-ability, i.e. if all segments are connected and inside the map. If this isn't the case or the start- and end point of the road are not at the map boundaries the roads get immediately discarded. Therefore, not all individuals created through crossover can be used. The mutation-operation has a chance of 0.05 to mutate a individual. In this process, a new road gets generated and one segment of it is randomly picked. This segment gets changed with a randomly chosen segment of the individual. After this, the road gets verified through AsFault as well and, if it fails, gets immediately discarded.

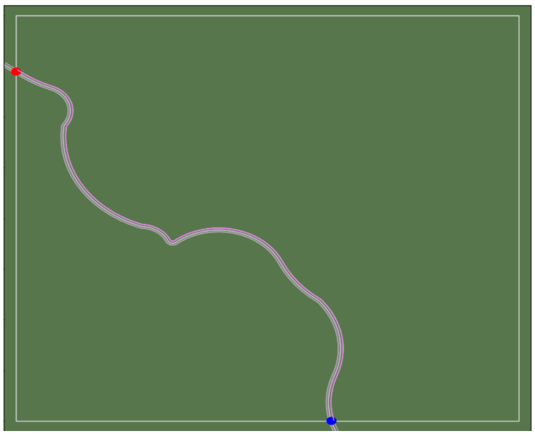
4.2.3 Novelty search

The novelty search is focused on maximizing the diversity of the speed- and curvature-profile inside a test suite. At first a population is generated and the curvature- and speed-profile of the road are calculated. This population serves as an archive as well, because every time a new individual increases the distance, the new individual will be put into the population and another individual of the population will be thrown out. After the population/archive got generated, the distance between each test case gets computed. First, the population gets sorted based on pareto-domination. This sorting is needed to ensure that the neighbour of an individual is the nearest neighbour of the population. The calculation of the distance is then done by computing the euclidean distance($p(x)$) between the individual(x) and the nearest neighbour(x_i) for each individual of the population(N) [8].

$$p(x) = \sum_{i=0}^{N} dist(x, x_i)$$

Besides generating the offspring through the evolutionary operators, a new road is generated through AsFault and evaluated to expand the ability to search for novelty. This newly generated road will also be treated as an offspring. Then, each individual of the offspring gets exchanged with one individual of the population at a time to compute the distance. After all individuals of the offspring were exchanged with all individuals of the population and the distances are calculated, the population with the highest distance between the neighbours sustains, while all other combinations are discarded. This process repeats until the time budget is met. As a result, the distance between the individuals is maximized, i.e. there are roads with lower curvature- and speed-values in the population as well as with higher curvature- and speed-values. This is demonstrated in figure 6, where the test case with the highest curvature- and speed-value of the outcome of the novelty search has a speed-value of 0.92 and a curvature-profile of 0.67, while the test case with the lowest curvature- and speed-value outcome has a speed-value of 0.42 and a curvature-profile of 0.33.


(a) Test case with lowest profile-values

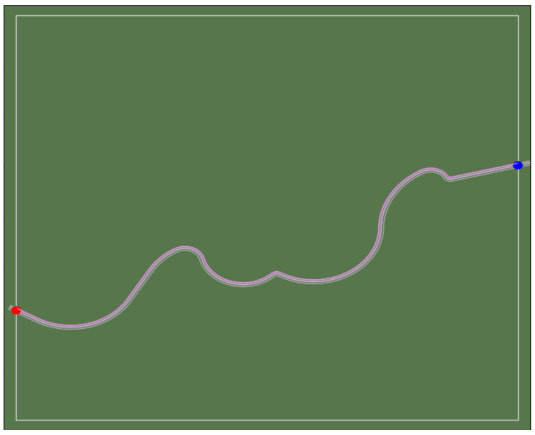

(b) Test case with highest profile values

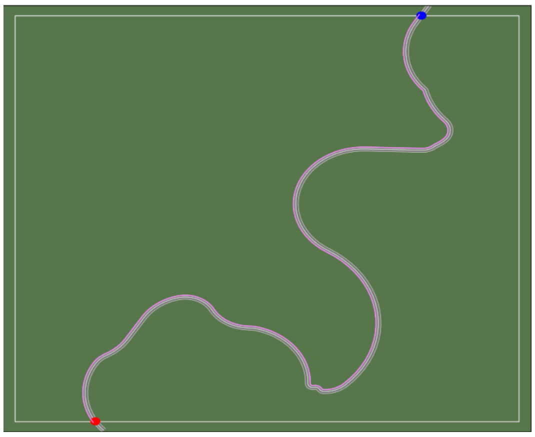
Profile Type	(a)	(b)
Speed:	0.42	0.92
Curvature	0.33	0.67

Figure 6: Result examples of novelty search

4.2.4 Evolutionary algorithm

The multi-objective search by Loiacono et al. tries to maximize the diversity of each test case by maximizing the speed- and curvature-profile of the road. Because of the character of this method, which is to produce just one best individual, and the focus on diversity of the test cases, I use an archive to sustain the best individuals of each evolution. Without an archive, the result of this approach would have maximized profile-values as well, but because of the crossover-operation all roads would be more and more similar over time and the archive counters this. The approach starts by generating a population, which also serves as the initial population of the archive. Then the evolutionary operators are applied onto the population to generate the offspring. Next the curvature- and speed-profile of the road are calculated. After this, NSGA-II is used to select the best individuals from both, the population and the offspring. At some point, the newly generated offspring won't be selected into the population. As the population evolves, the curvature- and speed-values of each individual were increased to a point, where the search operators can't produce an offspring with higher curvature- and speed-values. To verify the increase of profile-values of the population, I use the distance function from the novelty search to calculate the distance of the population and observe changes. If the distance of the test suite doesn't change for a specified number of iterations, the best individual of the population is selected to replace the worst individual of the archive. This approach runs until the time budget is met and it results in a test suite, where every test case is maximized regarding the speed- and curvature profile, i.e. every test case has high speed- and curvature profiles. This is shown in figure 7 as the worst evaluated outcome has a speed-value of 0.75 and a curvature-profile of 0.67 and the best evaluated outcome has a speed-value of 0.75 and a curvature-profile of 0.83.


(a) Test case with lowest profile-values


(b) Test case with highest profile values

Profile Type	(a)	(b)
Speed:	0.75	0.75
Curvature	0.67	0.83

Figure 7: Result examples of novelty search

4.3 Implementation

4.3.1 DEAP

The novelty search as well as the multi-objective search are implemented through the Distributed Evolutionary Algorithms in Python (DEAP) framework, which can be found at 'https://github.com/DEAP'.

DEAP is a well know library for genetic algorithms and provides the basic construction, i.e. the workflow, for both generic algorithms. Its set of tools, which I use in my thesis, consists of methods to link the individuals and their fitness function value, group population and offspring for the evolutionary operators and it provides the selection method, in this case, NSGA-II.

4.3.2 BeamNG

BeamNG and BeamNGpy, which can be found at 'https://github.com/BeamNG/BeamNGpy' (BeamNG version 1.3 is used in this thesis), is used for multiple reasons in this thesis: It provides a non-human-driver BeamNG-AI, gives detailed information of the ego-vehicle through its programming interface BeamNGpy, has an accurate physics engine regarding the trajectory of the car and it is free to use. BeamNG-AI can determine the speed-profile through driving the test cases with the highest speed possible. Especially when BeamNG-AI drove off the track, the extensive settings of it allows to fine adjust the speed in a way, which allows to achieve the highest possible speed for the segment. The information of BeamNG are used to adjust the speed, if the vehicle got off the road, and to provide the information and pictures for DeepDriving.

4.3.3 AsFault

AsFault, which can be found at 'https://github.com/Signaltonsalat/AsFault' (version e95ff62 is used in this thesis), is mainly used to provide the roads for the novelty search and the multi-objective search and validate the roads, which are produced through the search operations. Both methods need roads to evolve them, and they are supplied through AsFault's ability to generate roads in form of dictionaries. After the mutation-operation or the crossover-operation have been applied, AsFault verifies, that the roads are drive-able and there is no intersection inside the road. The connection between AsFault and BeamNG is also beneficial, because AsFault expands the road dictionaries with information about when the BeamNG-AI got off track regarding it's speed and which segment it happened in, which I use to identify the highest possible speed for this segment. Likewise, AsFault can visually plot the generated roads.

4.3.4 Testing the subject

As the simulation and the test subject are not connected in any way, transferring information and orders is not trivial. To work properly, i.e. controlling the steering and speed of the ego-vehicle, DeepDriving, which implementation DeepDrive by Andre' Netzeband can be found at 'https://bitbucket.org/Netzeband/deepdriving/src/master/', needs a picture of the situation. This picture, which needs a specific brightness and format, gets provided by the BeamNG-API through a wrapper. This wrapper is also used to deal with providing the information to the test subject, pausing the simulation, waiting for orders and executing them. Besides this, the wrapper also takes care of counting the number of times and segments, where the test subject gets out of bounds (oob), i.e. off the road, and exchange test subject with the BeamNG-AI, if DeepDriving gone too far off the road and can't find back to it by itself.

4.3.5 Experiment

The code of my tool, the data and instructions for replicating this study are available at 'https://github.com/Fierl/ComparingNoveltyMulti-Objective'.

5 Evaluation

I address three research questions in this thesis. The main question of the thesis is 'How effective are diversity based test generation techniques regarding testing autonomous driving system?'. They use different approaches to achieve this, while multi-objective search aims to maximize the profile-values of each test case, novelty search focuses on maximizing the distance between the profile-values inside the test suite. Because of that, the second research question is 'Which diversity based test generation is better regarding testing autonomous driving systems?'. Moreover, it would not be desirable, if all oob's happen in one type of road-segment and therefore, diversity in the type of road-segment, in which the car got off the road, is also an important factor to consider in generating test cases. Because of the nature of the multi-objective search to maximize certain properties through crossover, there is a tendency to minimize the variety of road-types. But on the other hand, the focus of the multi-objective search lies towards diversity at test case level, in this case, various types of road-segments. So it can be assumed that the way, in which the test subject fails, i.e. the segments in which the ego-vehicle goes off the road, are more diverse than it is for the novelty search. On account of that, my third research question is 'Does multi-objective search produces more diverse failed test cases, in terms of road-segment variety, than novelty search?'.

5.1 Experimental setup

The experimental software (AsFault, BeamNG and DeepDriving) was executed on two PC. The first PC runs on Windows 10 and has an Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz, 8 GB of memory and a NVIDIA GeForce GTX 1050 Ti. The second one runs also on Windows 10 and is equipped with an Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 16 GB of memory and a NVIDIA GeForce GTX 1070.

5.1.1 Random baseline

I incorporate some generic peculiarities into the random baseline to facilitate the comparison of the novelty search algorithm and multi-objective search algorithm by Lehman to it. My main focus lies on counting the number of times the BeamNG-AI drives off the road and select the population with the highest number.To do that, I start by generating a population and let BeamNG-AI drive on it. I count the number of times, where the Ego-Vehicle goes off the road, for the whole population. Then, another population gets generated and the off-road number counted. Both off-road numbers are compared and the population with the higher number gets selected. This is done until the time budget is met. As a result, random generation creates the population with the highest off road count for BeamNG-AI.

5.1.2 Test subject

For this experiment DeepDriving is used as the test subject. DeepDriving gets footage of the situation from the simulation and computes the action for the ego-vehicle. To do that, the simulation has to be stopped for a short time, while DeepDriving determines the next command and then continued again. This is done every 250ms. DeepDriving drives on the roads, which have been generated through multi-objective search, novelty search and random, and I count every time it goes off the road and the corresponding road-segment.

5.1.3 Execution

Each of the three algorithms, i.e. the novelty search, multi-objective search and the random baseline, is executed fifteen times for six hours. Every 30 minutes the population gets saved, which results in 5400 test cases. Of these 5400 test cases 450 test cases are part of the final outcome. While the test subject drives on all 5400 test cases, I count the number of oob's and their corresponding road-segment. The test is finished, when the ego-vehicle reaches the end of the road, which takes two minutes on average.

5.2 RQ1: 'How effective are diversity based test generation techniques regarding testing autonomous driving systems?'

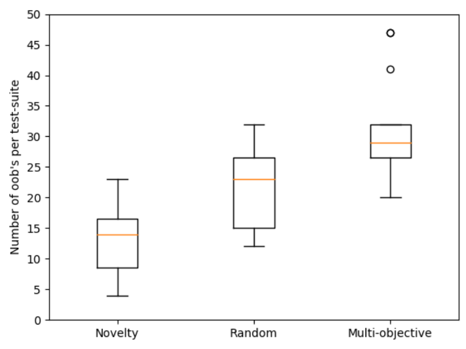
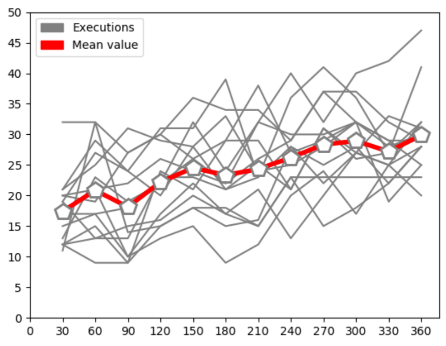
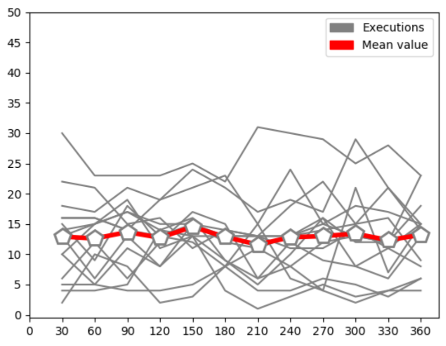


Figure 8: Comparison of the results

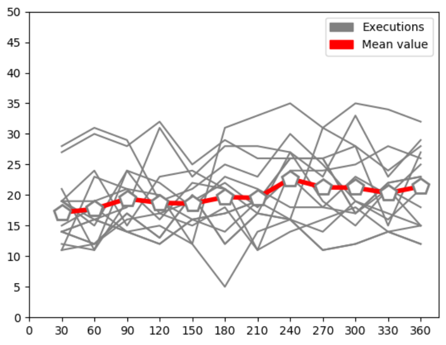
To answer the first research question, I let DeepDriving drive against the final outcome of each method and the random baseline to examine the effectiveness of the final results regarding the number of oob's of the test subject as well as against all saved test cases to verify the improvement of the outcome through evolutionary methods. In figure 8 are the numbers of oob's per test suite of the final result of novelty search, multi-objective search and the baseline displayed. The multi-objective search has a mean value of 29 oob's, while the mean value of random baseline is 24 and of novelty search is 14. The first quartile of the multi-objective search is at the same oob's value, 26, as the third quartile of the random baseline and the first quartile



(a) Multi-objective search



(b) Novelty search



(c) Random

Figure 9: Oob's over time

of the random baseline has with 17 oob's two more than the third quartile of the novelty search. Moreover, multi-objective search has two outliers at 47 and 42 oob's, while the other two haven't any outliers. Figure 9 shows, that the mean value of the multi-objective search and the random baseline increases over time, from 17 oob's to 30 respectively from 16 oob's to 20, while the novelty search stagnates at 12 oob's. The growth of the multi-objective search is 76% and of random baseline is 25%. As figure 8 demonstrates, the multi-objective search does produce 16% more oob's than the random baseline with an A_{12} of 0.81 and a p-value of 0.005, while the novelty search has the smallest number of oob's with 58% less oob's than the random baseline with a A_{12} of 0.82 and a p-value of 0.004. Figure 9 shows that multi-objective search, through maximizing the profile-values, increases the number of oob's by 68% on average over the time budget, while novelty search, by maximizing the distance between each test case, stagnates the number of oob's.

In the light of these results, the answer to RQ1 is as follows: The multi-objective search is able to produce better test suites as the random baseline by evolving test cases over time through maximizing the profile-values. The focus of novelty search to maximize the distance between each test case inside the test suite over time is nonconstructive for achieving a higher number of oob's. Beside that the final results of novelty search produce less oob's than the random baseline, novelty search also doesn't improve the number of oob's through the evolutionary approach.

5.3 RQ2: 'Which diversity based test generation is better regarding testing autonomous driving system?'

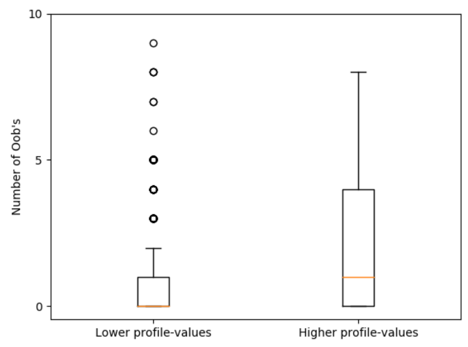


Figure 10: Comparison between low profile-values and high profile-values

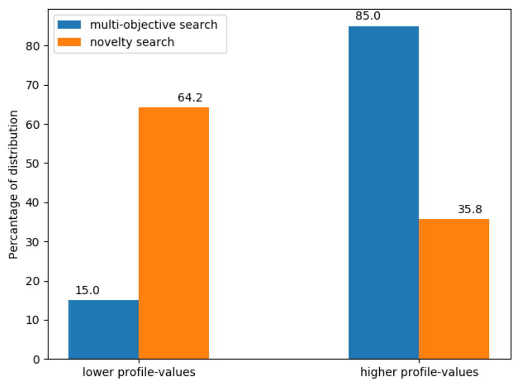


Figure 11: Distribution of low profile-values and high profile-values

To answer the second research question, DeepDriving drives against all generated roads of novelty search and multi-objective search and then the test cases are grouped based on their profile-value. To do that, I split the test cases into two sets. If the test case has a curvature- or speed-profile value below 0.7, it is considered as a test case with lower profile-values, if both profile-values are higher than 0.7, it is considered as a test with higher profile-values. This boundary value is chosen, because it groups the test cases into two sets of similar size. Figure 10 shows the number of oob's for the lower-profile-values-set and the higher-profile-set and figure 11 displays the distribution of test cases into the lower values and higher values profile-sets of novelty search and multi-objective search in percent. The mean value of test cases, where oob's happened, is zero for the lower-profile-values-set and one for the higher-profile-set. The mean value of the higher-profile-values-set has the same number of oob's than the third quartile of the lower-profile-values-set. Figure 10 also shows, that the lower-profile-values-set has outliers with an greater number of oob's than multi-objective search has. Figure 11 shows, that 64.2% of all test cases generated through novelty search are in the lower-profile-values-set, while only 15% of all test cases generated through multi-objective search are considered as test cases with lower profile values. The first research question has shown that the multi-objective search produces more oob's than the novelty search. Figure 8 states, that the multi-objective search produces 100% more oob's than the novelty search with an A_{12} of 0.99 and a p-value of 0.001. While the multi-objective search maximizes the profile-values of the test cases, the novelty search maximizes the distance inside the test suite, which results in test cases with lower profile-values, as figure 11 displays. The results of figure 8 suggests that the test subject produces a higher number of oob's on test cases with a higher profile-value. Figure 10 shows, that the higher-profile-values-set produces one more oob's on average than the lower-profile-values-set with an A_{12} of 0.66 and a p-value of 0.001.

As a result of the experiment, the answer to RQ2 is as following: The multi-objective search produces a higher number of oob's than novelty search by maximizing the profile-values of test cases, because a higher profile-value generate more oob's on average. The approach of novelty search to maximize the distance between the test cases results in test cases with a low curvature- and speed-profile. These test cases with a lower profile-value don't produce as many oob's on average as test cases with a higher profile-value do and because of that, the overall number of oob's of the test suite evolved by the novelty search is smaller than it is of the test suites evolved by the multi-objective search. Moreover, the maximizing of the distance stagnates the number of oob's over time as seen in figure 9, because low evaluated test cases are promoted instead of being replaced by better evaluated test cases.

5.4 RQ3: 'Which diversity based test generation produces more diverse failed test cases, in terms of road-segment variety?'

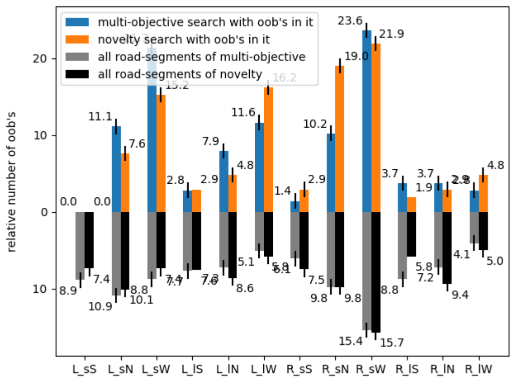


Figure 12: Distribution of oob's and overall road-segment distribution

To answer the last research question, DeepDriving drives against the final outcome of novelty search and multi-objective search. I count every time DeepDriving gets off the road and save the associated road-segment. The x-label of Figure 12 contains the name of the bins,'L_xX' for left turn and 'R_xX' for right turn, 'X_sX' for short and X_lX' for long and 'X_xS' for sharp, 'X_xN' for neutral and 'X_xW' for wide. Figure 12 represents the distribution of road-segments in which oob's happened in percent as well as the overall distribution of road-segments for novelty search and multi-objective search in percent. Figure 12 shows that the distribution of road-segments with oob's in it of novelty search and multi-objective search have similar outliers in minima and in maxima. The percantage of oob's, which happened in R_sW, is 23.6% for the multi-objective search and 21.9% for the novelty search. This is 16 respectively eight times that size of R_sS, which is 1.4% respectively 2.9%. L_sS doesn't have any oob's in it for both approaches, despite the fact, that it is not under-represented in the overall distribution of road-segments. The biggest difference between the two methods is in L_sW, where multi-objective search has a percentage of 21.3% and novelty search has a percentage of 15.2%. Despite achieving diversity through different approaches, the variety in terms of road-segment types overall and road-segments, in which oob's occure, is quite similar for both. The multi-objective search as well as the novelty search have one high outlier and one segment in which no oob has happened. Both methods balance right turns and left turns and have their most oob's in short and wide turns.

As a result of the experiment, the answer to RQ3 is as following: There is no significant difference between the distribution of road-segments in terms of oob's. Despite their different approach, both get high and low numbers of oob's in the same road-segment types. The most oob's are in short and wide turns, because the wide angle leads to an acceleration of the ego-vehicle, while a short pivot means, that the next road segment respectively a new situation for DeepDriving follows faster. On the other hand DeepDriving decelerate for small and sharp road-segements, which explains the low number of oob's for this road-segment type.

5.5 Summary

This experiment shows that the multi-objective search is capable of evolving difficult test cases. The number of oob's increase throughout the evolution of the test suite, because maximizing the profile-values of the test cases produces a higher number of oob's. The novelty search on the other hand fails to evolve difficult test cases by maximizing the distance between their profile-values. The number of oob's stagnates because of the focus of novelty search, which propagates test cases with a low profile-value as well. Despite their different efficiency regarding generating failed test cases, they have a similar distribution of road-segment types, in which the ego-vehicle goes off track.

5.6 Personal experience

The work with three different types of software, AsFault, BeamNG and DeepDriving, has it's peculiarities. Since AsFault was designed to work with BeamNG, generating roads and simulating them inside AsFault was not a problem. As I started to modify the roads outside of AsFault, BeamNG didn't render any of the models of the road, which resulted in falling into water every time I ran the simulation. This was fixed by changing the path variables of every model, which AsFault uses in their generated roads. Also, starting BeamNG through the BeamNGpy doesn't use the same path as starting it through AsFault. As a workaround the generated road are copied from the game folder into my documents folder. DeepDriving on the other hand was really hard to get running on one of the execution PC's, because of the complicated installation process.

6 Threads to validity

6.1 Internal validity

I tested the output manually to ensure the credibility of the results, i.e. I computed the curvature- and speed-profile for ten samples and the distance for two populations and compared it. Also, I ensured the drive-ability of BeamNG-AI and DeepDriving on the generated road by observing them. Likewise, I used the Mann-Whitney-U test to evaluate the statistical difference of the outcome and computed the effect-size through the Vargha-Delaney-A test. Beside that, using a simulation, especially a game-engine, to test autonomous driving system regarding real world functionality comes with certain limitations, also called the simulation gap. These limitations are for example a lack of real world environmental details like dirt on the road or weather conditions. This simulation gap could lead to a drop of the effectiveness [9].

6.2 External validity

In my this thesis, one test subject, DeepDriving, is used to evaluate the results. This is because of the lack of available autonomous driving systems. Additional, DeepDriving needs time to compute it's next command and in this time, the simulation needs to be stopped. Because of that, the results of this thesis could be unable to be translated into the real world.

7 Conclusion

In this thesis I compared two approaches, which aim to maximize diversity in their own way. While maximizing certain aspects of the test cases through the multi-objective search of Loiacono et al. resulted in more difficult test cases for the test subject, maximizing the distance between test cases inside the test suite through the novelty search by Lehman did not. Beside focusing on two different aspects of diversity, the distribution of road-segment types and of road-segments, in which the ego-vehicle drove off the track, where similar for both methods. In the field of testing self-driving cars the autonomous driving system has to compute the right decision for each situation. Because of that, the diversity of the overall shape of the road, on which the autonomous driving system is operating on, doesn't affect the system in the same way as the diversity inside the road itself does. Therefor, the approach of the novelty search is not as suitable as the multi-objective search, in which this aspects get maximized, to be used in this field. In contrast to this, the multi-objective search of Loiacono et al., which was original used to create interesting roads for a video game, is capable of producing difficult test cases for an autonomous driving system, and this suggests that roads, which are more challenging to drive for humans, might be also more challenging to drive for autonomous driving systems. The code of my tool, the data and instructions for replicating this study are available at 'https://github.com/Fierl/ComparingNoveltyMulti-Objective'.

8 Future Work

Diversity in test cases is quite crucial to prove the robustness of the autonomous driving system. To expand the capability to achieve this, extensions to this work can be made.

8.1 Increase number of profiles

More parameters for the road evaluation like gradient, weather or dirt on the road, to just name a few, could be applied to widen the characteristics of a road. These new aspects could then be represented as new profiles. Thereby can be explored, if increasing the number of different profiles lead to a higher number of oob's.

8.2 Combination of novelty search and multi-objective search

To extend the ability to generate diverse test cases for autonomous driving systems both approaches could be combined. A possible combination could be done through filling the population and offspring of the novelty search approach with results gathered by the multi-objective search approach. This would decrease the profile-value range, in which the novelty search can choose from, and through the lack of test cases with lower profile-values, produce more failed test cases. The outcome of the combined approach should propagate diversity on test case level as well as on test suite level. It would be interesting to see, how the combination achieves diversity on the test case and test suite level compared to the methods by themselves in terms of testing autonomous driving systems.

8.3 Increase number of test subjects

In terms of expanding the evaluation of this work, other autonomous driving systems than DeepDriving could be tested through this approach, to see if mediated-perception-approach-driving systems and behavior-reflex-approach-driving systems correspond to the results gathered in this work. Especially comparing these two approaches regarding their ability to operate safely in situations, which are maximized in diversity, would be interesting.

8.4 Further evaluation

For the evaluation of diversity in failed test cases, i.e. the type of road-segments where the ego-vehicle goes off the track, saving the previous and next road-segment of the road-segment, where the oob occurred, would facilitate to understand why the test subject goes off the road.

References

[1] Andrea Arcuri and Lionel Briand. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Software Testing, Verification and Reliability*, 24(3):219–250, 2014.

[2] Raja Ben Abdessalem, Shiva Nejati, Lionel C Briand, and Thomas 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*, pages 63–74. ACM, 2016.

[3] Keshav Bimbraw. Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology. In *Informatics in Control, Automation and Robotics (ICINCO), 2015 12th International Conference on*, volume 1, pages 191–198. IEEE, 2015.

[4] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

[5] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[6] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. Deap: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13(Jul):2171–2175, 2012.

[7] Alessio Gambi, Marc Müller, and Gordon Fraser. Asfault: testing self-driving car software using search-based procedural content generation. In *Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings*, pages 27–30. IEEE Press, 2019.

[8] Joel Lehman. Evolution through the search for novelty. 2012.

[9] Antoine Ligot and Mauro Birattari. On mimicking the effects of the reality gap with simulation-only experiments. In *International Conference on Swarm Intelligence*, pages 109–122. Springer, 2018.

[10] Daniele Loiacono, Luigi Cardamone, and Pier Luca Lanzi. Automatic track generation for high-end racing games using evolutionary computation. *IEEE Transactions on computational intelligence and AI in games*, 3(3):245–259, 2011.

[11] Marc Müller. Evolutionary test generation for autonomous vehicles, 2018.

[12] De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau, Christian Gagné, et al. Deap: A python framework for evolutionary algorithms. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 85–92. ACM, 2012.

[13] Dan Robitzski. Just 19 percent of americans trust self-driving cars with kids, 2019.

[14] Vlad Savov. Donald trump is reportedly no fan of self-driving cars, 2019.

[15] Julian Togelius, Georgios N Yannakakis, Kenneth O Stanley, and Cameron Browne. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186, 2011.

[16] Xinjie Yu and Mitsuo Gen. *Introduction to evolutionary algorithms*. Springer Science & Business Media, 2010.

Eidesstattliche Erklärung:
Hiermit versichere ich an Eides statt, dass ich diese Bachelorarbeit
selbstständig und ohne Benutzung anderer als der angegebenen Quellen und
Hilfsmittel angefertigt habe und dass alle Ausführungen, die wörtlich
oder sinngemäß übernommen wurden, als solche gekennzeichnet sind,
sowie dass ich die Bachelorarbeit in gleicher oder ähnlicher Form noch
keiner anderen Prüfungsbehörde vorgelegt habe.

Passau, 08.07.2019

Fierbeck Simon