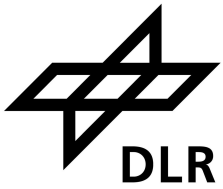
Carl von Ossietzky University Oldenburg

Department of Computer Science

Deutsches Zentrum für Luft- und Raumfahrt

Institute of Systems Engineering for Future Mobility



Application of Evolutionary Algorithms to Analyze Criticality in Urban Traffic Scenarios

M.Sc. Computer Science

Master's Thesis

First Examiner: Prof. Dr. Martin Fränzle
Second Examiner: Dr. Christian Neurohr

submitted by: Anselm Fehnker
Billhorner Deich 88
20539 Hamburg
+4915228676352
anselm.fehnker@uni-oldenburg.de

date of submission: 22.05.2022

Fehnker, Anselm:
Application of Evolutionary Algorithms to Analyze Criticality in Urban Traffic Scenarios
Master's Thesis, Carl von Ossietzky University Oldenburg, 2021.

Abstract

As automated driving system equipped vehicles (AVs) join traditional vehicular traffic, a safety analysis of their programmed behavior is essential. This holds for a variety of domains including highways, rural roads, and particularly, urban traffic. In contrast to other traffic domains, AVs in urban traffic settings must interact with vulnerable road users (VRUs) such as bicyclists and pedestrians. In order to ensure that AVs do not endanger VRUs, it is crucial to identify all potential traffic situations in which criticality between AVs and VRUs could arise.

This master's thesis examines how the application of evolutionary algorithms (EAs) can identify critical situations in urban traffic scenarios. To this end, three separate urban traffic scenarios are designed. Each scenario models different criticality phenomena and has a different complexity. In order to enable high-frequency repetitions of the scenarios without endangering real humans, the scenarios are implemented in the urban traffic simulator CARLA.

The aim of the EAs is to learn how to derive *concrete scenarios* that yield a high criticality from the three designed scenarios. Hence, as a fitness function of the EAs, a criticality metric that quantifies the occurring criticality in a *concrete scenario* is used. Specifically, a new criticality metric called the *Predictive Conflict Index (PCI)* is proposed and validated in this thesis.

With the use of this fitness function, six different variants of EAs are implemented in PYTHON. These are the basic $(\mu + \lambda)$ and $(\mu - \lambda)$ algorithms, $(\mu + \lambda)$ in combination with Rechenberg, $(\mu - \lambda)$ in combination with Rechnberg, $(\mu + \lambda)$ in combination with self adaptation and $(\mu - \lambda)$ in combination with self adaptation.

The two basic variants and $(\mu+\lambda)$ in combination with self adaptation are able to generate a large amount of critical *concrete scenarios*. Moreover, the learning process of the algorithms identify different properties that are linked to high criticality, such as high target speed of AVs or close distance of parked vehicles to an intersection. Furthermore, utilizing methods from data analysis such as clustering, different types of critical situations in urban traffic are revealed. Ultimately, the examined approach is highly promising as a method to improve the development of AVs.

Contents

List of Figures

List of Tables

1 Introduction

Urban mobility is undergoing rapid change due to the influence of digitalization as well as due to efforts toward sustainable mobility transformation.
On the one hand, digitalization presents the opportunity to create smart infrastructures and intelligent traffic systems which make the introduction of automated driving system equipped vehicles (AVs) in public traffic more realistic.
On the other hand, the threats of the climate crisis and the hope for a higher quality of life in many cities, including massive European metropolises such as Paris, Barcelona, London, and Berlin, are incentivizing a redesigning of traffic infrastructure. In particular, cities are changing their street structures away from streets designed mainly for cars toward streets where cars share the traffic space with vulnerable road users (VRUs) such as bicyclists, pedestrians, and e-scooter drivers. This development requires AVs to master increasingly complex traffic environments [3].

Given this reality, it is even more important for the verification and validation of the behavior of AVs to investigate the interactions of AVs and VRUs. In current traffic statistics, VRUs make up over one-third of fatally injured road users [4]. In order to prevent further VRU deaths, and to introduce AVs in a responsible manner, it is necessary to identify all kinds of situations in which AVs can endanger VRUs.
However, the presence of VRUs also complicates the analysis of criticality in urban traffic. For one, the increased severity of an accident involving VRUs makes it cost-intensive and unethical to analyze criticality in real-life traffic. Furthermore, the more flexible behavior of VRUs increases the number of potential situations in urban traffic.

In order to nevertheless identify critical situations in the urban traffic domain, evolutionary algorithms (EAs) will be applied. These EAs aim to sample *concrete scenarios* from pre-defined scenario spaces and thereby learn which scenario constructions lead to high levels of criticality.
For this application, the open-source simulator CARLA will be used, which has been developed to support the development, training, and validation of automated urban driving systems [5]. Through this approach, a scenario space can be explored by executing large amounts of *concrete scenarios* while ensuring that a crash between an AV and a VRU has no real-life negative consequences.

For the evaluation of criticality, a new criticality metric called *Predictive Conflict Index (PCI)* is proposed in this thesis. This criticality metric is particularly adapted for urban traffic. The outcome of *PCI* is used as fitness function of the EAs. Hence, the EAs learn

how to sample from the designed scenario spaces to obtain a high output of the criticality metric.
As a result of the EAs, a set of *concrete scenarios* with a high criticality is obtained. This set is then analyzed with the aim of identifying causes of criticality in urban traffic. Thus, situations that challenge AVs in urban traffic can be identified without endangering VRUs.

Next, in chapter 2, a summary of the current state of the research is given. This includes the current state of AV development and an overview of what is particularly challenging for AVs in urban traffic. Moreover, approaches to how criticality in traffic can be analyzed are presented. At the end of the chapter, an introduction to EAs and how they are used in traffic research is given.
In chapter 3, the four research questions that are investigated in this master's thesis are presented.
Following this, in chapter 4 the methodology behind the evaluation of these four research questions is explained. This includes the usage of the CARLA simulator, the design of the three scenario spaces, the proposal of the criticality metric *PCI* and the implementation of EAs in PYTHON.
The evaluation of the research questions is presented in chapter 5. In a first step, the newly proposed criticality metric *PCI* is validated. Thereafter, the generated *concrete scenarios* are analyzed in each of the three designed scenarios.
In the last chapter, chapter 6, the results of this evaluation are summarized. Finally, an outlook about possible future work is given.

2 Background

This chapter gives an overview of the current state of the development of AVs and the challenges that need to be resolved before AVs can participate in urban traffic. Further with the description of the criticality analysis and the approach to optimize criticality with the help of evolutionary algorithms (EAs), an approach is introduced to generate critical traffic scenarios and thereby identify critical situations between AVs and VRUs in urban traffic.

2.1 Automated Driving Systems and Urban Traffic

The vision to develop vehicles that are able to drive without human control has kept scientist busy for a long time. The first steps towards humanless driving are considered to be made already 100 years ago, in the 1920 [6, 7]. At this time a radio controlled car called Linriccan Wonder was developed and demonstrated. Although its control system was still operated by human, the Linriccan Wonder was the first car that drove without a human inside, and therefore counts as the first step towards humanless driving.
Since then many new technologies were invented, ethical and legal discussions were held, and predictions were made when vehicles completely operated by computers finally will be on the roads.
In the subsection 2.1.1, an overview of what computer-controlled vehicles include is given, the term automated driving system equipped vehicles (AVs) is defined, the benefits that the introduction of AVs promises for our lives are explained, and an overview of how far the current state of development is given. In subsection 2.1.2 an overview of the challenges AVs face in urban traffic is given.

2.1.1 Development of Automated Driving System Equipped Vehicles

When this topic is addressed, several different terms such as autonomous vehicles, automated vehicles or self-driving cars are used. The exact differences between these terms and by what level of technology a vehicle stops to be a conventional vehicle and starts to be a autonomous -, automated - or self-driving vehicle is often not clearly defined.

The reality is that there are many gradutions between vehicles which are fully human-driven and vehicles which partially or fully drive themselves [8]. In order to structure these graduations, the global association SAE International, which consists of more than 128,000

engineers and related technical experts in the aerospace, automotive and commercial-vehicle industries, introduced the *SAE J3016* taxonomy of driving automation.
According to this taxonomy, vehicles are divided into six discrete and mutually exclusive levels of automation [9]. A detailed division of the six levels can be seen in figure 2.1.

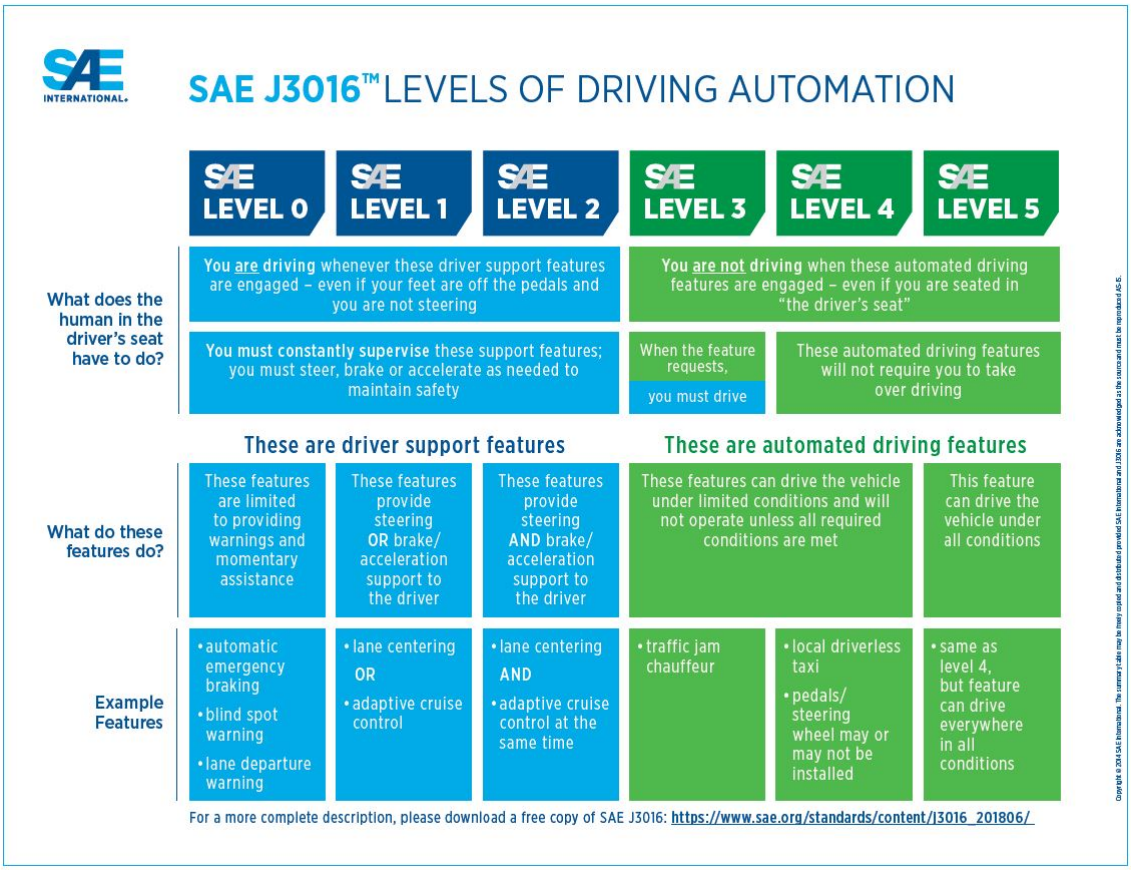


Figure 2.1: Overview of the six levels of the *SAE J3016* taxonomy of driving automation published by SEA international [1]

Based on this taxonomy the term *Automated Driving System equipped vehicles (AVs)* is used in this thesis, which is defined as follows:

Definition 1 ***Automated Driving System equipped Vehicles (AVs)*** *are motorized road vehicles, which are controlled by an automated driving system (ADS) with Level 3, Level 4 or Level 5 automation of the SAE J3016 taxonomy.*

The introduction of AVs that match this definition to public traffic has the potential to change life in many different ways. On the one side it can increase safety in traffic, have a positive influence on social and structural aspects of the current transport system and a reduce pollution in traffic [8, 10]. On the other side the introduction of AVs can also cause technological, ethical or legal difficulties.

When it comes to traffic safety, the biggest issue is the amount of people getting injured or killed in accidents. According to the latest report of the World Health Organization (WHO) on global road safety, about 1.35 million people die in traffic every year [11]. This number is classified by the WHO as unacceptably high [11]. When these traffic deaths are examined more closely, it is observed that the majority of them are caused by human error.
In the United States, for example, according to the National Highway Traffic Safety Administration (NHTSA), human error is involved in 94% of traffic accidents [12]. Also in the EU, accidents involving human error have with 95% a similiar high share, according to the European Parliament [13]. With the introduction of AVs to public traffic, there is the hope that these errors can be avoided and thereby the number of death can decrease significantly [8, 14, 15].
Therefore, it is necessary to ensure that the introduction of AVs does not lead to deaths caused by errors made by the ADS. Hence, the validation and verification of the driving behavior of AVs is extremely important.

Additionally to the major oppertunity of decreasing the number of deaths in traffic by leaving the task of controlling the vehicles to automation, the introduction of AVs would also release the burden of the driver and thereby makes the time spent in a vehicle less stressful. The gained time could, for example, be used for more productive tasks [8, 15].

Besides the safety and driver comfort aspects, the introduction of AVs can also improve the way how society has access to mobility. For example, AVs could increase the mobility of persons who are unable to drive a car, because they are too young for a driver's license, too old to control a car safely or cannot drive because of medical aspects [16]. In particular on the countryside, where due to the lack of alternatives people are more dependent on cars, AVs could enable many people to become more mobile and participate in public life more easily.

Also the mobility in cities can be positively influenced by the presence of AVs. While the huge amount of private cars negatively affects quality of life by creating unbearable traffic, occuping a large amount of public space for parking, and emitting noxious emissions, the introduction of AVs could reduce these problems [17].
On the one hand, as AVs can cooperate with smart infrastructures and intelligent traffic systems to generate a more efficient traffic flow and thereby reduce traffic jams [18, 16].
On the other hand, as the introduction of AVs might alter models of vehicle ownership and patterns of land use. For example, could concepts such as ride-sharing and automated shuttle services become more attractive and in some cases substitute private car ownership. Thereby, both the number of vehicles on the road and the number of parking cars on the side could decrease [7, 12, 17].
However, these changes do not just come from the technological introduction of AVs but additionally require the will of society.

Another aspect that is particularly important when considering the role of traffic emissions in public health and climate change is that AVs can reduce the emissions in traffic. For example, with the help of intelligent acceleration and deceleration profiles, AVs can be more efficient than human-driven vehicles and reduce wasted fuel or in case of electric vehicles the energy consumption [8, 10]. Further, a smarter handling of power consumption and charging times could accelerate the transition to electromobility.

Overall, a thoughtful introduction of AVs can be very promising to benefit public life. However, this requires a lot of work to ensure that AVs do not pose an additional risk to road safety or are not introduced in a manner in which they harm society. Thus, the question about the current state of development remains.

In the last years several technological advances have pushed the state of the art of AVs. This includes both, advances in computation technologies such as better sensors, better computer vision, promising machine learning approaches or hardware acceleration, and advances in the communication technologies such as the dedicated short-range communication (DSCR), cellular vehicle-to-everything (C-V2X) or 5G [7, 19]. As a result of these developments, vehicles equipped with ADS at SAE Level 1 or SAE Level 2 are already common on public streets and vehicles equipped with ADS on SAE Level 3 are beginning to be introduced to the market [20].

The predictions when vehicles equipped with ADS that fulfill SAE Level 4 or SAE Level 5 will occur in peoples daily life are also optimistic. Many predictions assume that cars can operate completely independent of humans by 2035 [6]. For example, Anderson et al. predict in their *Guide for Policymakers* from 2014 [8] that fully driverless cars will be on roads by at latests 2034. Most of the big automotive companies have already made enormous investments in the development of AVs [7, 19] and market research companies such as IHS Markit predict that by 2040, the annual sales of AVs will exceed 33 million vehicles [15, 21].

However, while the market is already preparing for the introduction of AVs, there are still gaps in the scientific research, which cause current automated driving technologies to be still immature and in development [12, 14, 19].
Before AVs can occur in public traffic, both the development of new technologies, such as even better sensors and steering systems, and many advances in scientific research, such as better methods for the validation and verification of AVs, are necessary.

One major concern about AVs is their safety and security. During the driving process, it is an important requirement that AVs are always reliable and are protected against all kinds of technical errors and threats. This includes several different challenges.
One important challenge is, for example, sensor safety, which deals with the risk that the physical components of AVs could be deceived with false signals. Another concern

is operating system security, where the operating system needs to be protected against unauthorized access. Further V2X communication security is also important, which deals with the threat that the communication between AVs with other vehicles or traffic management systems (TMS) could be attacked [14, 15].
Before AVs can be mass produced more research is necessary to satisfy all requirements on safety and security.

Another problem is that the real traffic environment is still too complicated for the currently developed AVs. Even though prototypes of AVs have already covered plenty of miles on test tracks, the varying conditions in the real world, such as not ideal road conditions, intemperate weather or interferences from different kinds of actors can challenge AVs in ways that they are not able to handle yet [7, 12, 19]. According to Yurtsever et al. [12] one of the biggest challenges since the earliest attempts of developing AVs is the urban traffic environment. The following subsection provides an overview of the specific challenges that have to be overcome, if AVs are to participate in the urban traffic domain.

2.1.2 Challenges in Urban Traffic

Different traffic domains have different characteristics and therefore also pose heterogeneous requirements on AVs. The highway domain is, for example, a relatively closed domain, in which roads have several, clearly delimited lanes. The vehicles on highways are mainly cars or trucks and sometimes motorcycles, while road users like pedestrians and bicycles are not allowed. Also the speed limits on highways are rather high and in Germany even unlimited. Therefore, important requirements on AVs in the highway domain are, for example, good lane keeping, detecting other vehicles on the same and adjacent lanes, longitudinal distance keeping, safe lane changing if necessary, and stable movement while driving with high speed.

Compared to the highway domain, the characteristics of the urban traffic domain are quite the opposite. The roads are more narrow and have less lanes. If one vehicle wants to overtake another vehicle in front, it is not uncommon that it has to use the lane of the opposite traffic. Also the speed limits in cities are comparatively low. But the biggest difference in urban traffic is that road users such as pedestrians and bicyclists, who have completely different behavior patterns than four-wheeled, motorized vehicles, use the same traffic space. This group of road users is referred to as vulnerable road users (VRUs).

Definition 2 ***Vulnerable Road Users (VRUs)*** *are individuals on or near roadways, which have relatively low mass and protection, and are therefore exposed to a higher risk of injury and fatality in road crashes. Examples for VRUs are pedestrians, bicyclists or e-scooter drivers. [22, 23, 24]*

Remark 2.1 *Per definition, motorcycles also count as VRUs. But as their behavior patterns are quite similar to the behavior of four-wheeled, motorized road users, this thesis mainly focuses on non-motorized VRUs.*

The presence of VRUs in urban traffic leads to additional requirements for AVs, as the interaction of VRUs and AVs introduces several challenges.

Already in today's traffic without AVs, VRUs are exposed to higher risks in traffic situations. It is shown in several studies that VRUs suffer from a disproportionately high share of serious injuries and deaths in traffic accidents [24, 25, 26].
According to the latest WHO report on global road safety, VRUs make up more than half of the global traffic deaths [11]. In the European Union more than 138.000 pedestrians and bicyclists where killed in traffic accidents between 2001 and 2013, which made up 29% of traffic deaths during this time [24]. According to the WHO this number even increased in the last years to 32% in 2018. [11].
Studies further show that collisions with motorized vehicles are the main cause of these deaths [23]. The introduction of AVs could lower the risk for VRUs of getting killed in traffic. Therefore, safety requirements for AVs, such as minimum distance and braking times, need to be even more stringent in urban traffic than in other traffic domains.

Not only are higher safety requirements necessary, but the interaction with VRUs also causes additional technological challenges for the development of AVs. Both detecting VRUs and predicting their behavior is more difficult than detecting other motorized four-wheeled vehicles and predicting their behavior.

Cars and trucks, for instance, have a relatively high mass and drive on pre-defined lanes. It is therefore well possible to detect and classify them with the help of sensors and image recognition networks [27, 28].
VRUs, on the other hand, have a much smaller mass and appear in different forms. For example, different body sizes, clothing styles or luggage items make image recognition networks more prone to errors.
In addition, the traffic space of VRUs is often not as structured as the road. For example, bicyclists can either ride on the road, on bike lanes, or in some cases on the sidewalk. Further, bike lanes and sidewalks are often occluded by trees or parking cars, which makes it more difficult for sensors to detect VRUs [29, 30].
Moreover, lighting conditions are also often worse for VRUs. While most urban streets are well illuminated and cars have front and rear lights, lighting on sidewalks is often unsatisfactory. This is one of the reasons why most accidents involving VRUs occur in the morning or evening in low illumination conditions or darkness [22, 29].
All these circumstances cause that the detection rate for cars has continually outstripped that for VRUs [31]. As requirements for AVs in urban traffic, it is necessary that they are able to detect and classify VRUs correctly, no matter where they are, which shape they have and what time of the day it is.

Even if AVs are able to detect all VRUs in their environment correctly, they are facing the next complex task - the prediction of their behavior over time.

VRUs, like pedestrians and bicyclists, have their own goals, utilities, and decision making systems [32]. Their movement is more flexible than the movement of four-wheeled road vehicles, as they have smaller turning circles and their traffic space is not regulated by fixed lanes. For example, pedestrians in urban traffic are not always obligted to use designated areas such as pedestrian crossings but can also cross roads at several other points using various paths and angles. Further, in contrast to the highway domain where every traffic participant needs a driving license, VRUs can also be children who might be unfamiliar with traffic rules [26].
Due to these factors, the behavior of VRUs is the least predictable of all road users. Differences in their behavior patterns can be influenced by features such as walking speed, age, sex, knowledge of the environment, individual or group transit, and time of day [4]. For the prediction of the movement, AVs need to consider various pieces of information, such as demographics, traffic dynamics, environmental conditions, road conditions and social factors [33].
Understanding and predicting the intention of VRUs is an essential requirement to enable AVs in the urban domain [34]. If it is not given, AVs are not able to take required action in order to guarantee the safety of all road users [35].

All together, VRUs do not only have a high chance to get fatally injured in an accident, but accidents could also become more likely, as an AV could either misbehave because it did not detect a VRU properly or because it did not predict the correct behavior of a VRU. This can increase both, the severity and the probability of an accident.
Therefore, to drive the development of AVs, it is necessary to identify all possible circumstances under which an AV fails to detect a VRU or predicts the VRU's behavior incorrectly. In this way the underlying causalities of a possible accidents can be identified and the corresponding behavior of AVs can be improved.

2.2 Criticality in Traffic

Identifying all possible circumstances in which AVs have problems to detect or predict VRUs is not an easy task. The urban traffic domain is an arbitrary domain which is due to the many different road users and street properties even more complex than, for example, the highway domain. AVs and VRUs can occur in different ways with an unlimited number of different characteristics. Already at a single specific urban intersection, infinite different compositions of parking cars, pedestrians and bicyclists can be arranged. Added to this are different weather, time of day, and lighting conditions. The space that has to be searched for potential problems is therefore infeasible large.

In order to nevertheless identify circumstances under which AVs could endanger VRUs, it is necessary to map this infeasible large domain to a finite set of artifacts. For this purpose, Neurohr et al. [2] developed a method called crititcality analysis. Before this method will be explained in detail, some further definitions need to be introduced that

are necessary to structure the traffic domain.

Thus, the terms *scene*, *situation* and *scenario* as well as the terms *criticality* and *criticality phenomenon* are defined in subsection 2.2.1. After this, an overview of how criticality in traffic can be evaluated is given in subsection 2.2.2. At the end of this section, in subsection 2.2.3, the work-flow and application of the criticality analysis will be explained.

2.2.1 Important Definitions for Traffic Research

In order to give the infeasible large domain of urban traffic some structure, it is necessary to define clear terms that describe the structure and characteristics of traffic compositions. For this purpose the terms *scene*, *situation* and *scenario* will be defined in the following and used in this thesis afterwards.

2.2.1.1 Scenes, Situations and Scenarios

The terms *scene*, *situation* and *scenario* are regulary used in literature but are often not clearly defined and in some cases even used contradictorily. To enable a unified use of these terms, Ulbrich et al. reviewed in [36] several papers and came up with consistent definitions of these three terms. These definitions will also be used in this thesis. Therefore in the following an overview is given, how these terms are defined, what the difference between these terms is and how they can be used for the validation and verification of AVs in urban traffic.

The first term *(traffic) scene* describes the composition of a traffic environment at a certain point of time. Ulbrich et al. define the term *scene* as follows:

Definition 3 ***(Traffic) Scene*** *"A (traffic) scene describes a snapshot of the environment including the scenery and dynamic elements, as well as all actors' and observers' self-representations, and the relationships among those entities. Only a scene representation in a simulated world can be all-encompassing (objective scene, ground truth). In the real world it is incomplete, incorrect, uncertain, and from one or several observers' points of view (subjective scene)" [36, p.983].*

According to this definition, a scene therefore consists of all information about the stationary scenery, dynamic elements, and actors and observers in the environment. The information about the stationary scenery includes the lane network, possible conflict areas as well as properties of static elements like obstacles. Information about dynamic elements can, for example, be the states of traffic lights. For the actors and observers, information about state, skills and abilities such as their heading angle or the current field of view are part of the scene.

An illustration of an example scene in urban traffic is shown in figure 2.2. In this example, the stationary scene is a single-track road that has a sidewalk on each side. The actors,

which are in this scene also the observers, are an AV, a bicyclist and a pedestrian. Note that only the current state of the actors and not their intentional behavior is part of the scene.
For the development of AVs a scene like this can be used as interface between the perception of the environment and the planning of the future behavior [36]. It can, for example, be examined how much of the scene is perceived by the AV and what future behavior the AV is planning with this perception.

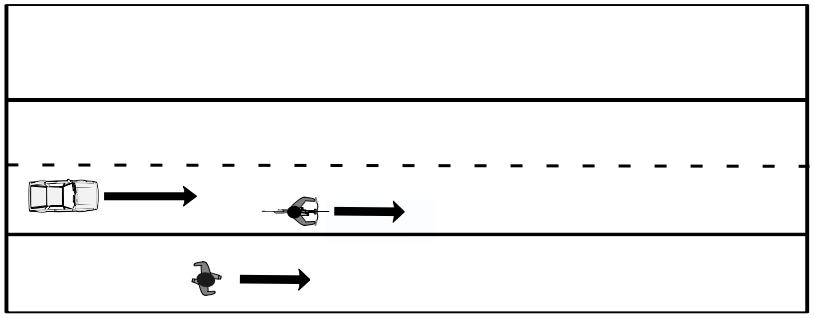


Figure 2.2: An example *traffic scene* in urban traffic, consisting of a single-track road, two sidewalks, an AV, a bicyclist, and a pedestrian. The arrows represent the speed and the heading angle of the actors.

However, a scene itself is independent of the future behavior of the actors. When the intentional behavior of actors shall be investigated, the term *(traffic) situation* becomes important. Ulbrich et al. define a *situation* as follows:

Definition 4 ***(Traffic) Situation*** *"A (traffic) situation is the entirety of circumstances, which are to be considered for the selection of an appropriate behavior pattern at a particular point of time. It entails all relevant conditions, options and determinants for behavior. A situation is derived from the scene by an information selection and augmentation process based on transient (e.g. mission-specific) as well as permanent goals and values. Hence, a situation is always subjective by representing an element's point of view" [36, p.985].*

In contrast to a scene, a situation therefore contains all relevant aspects that are necessary to plan a particular behavior.
To continue with the previous introduced example in urban traffic, this behavior could be that the AV decides based on the information in the scene that it wants to overtake the bicyclists. The relevant aspects that are needed for this behavior are the information about the AV, the bicyclists and the traffic space that is needed for this maneuver. Information about the pedestrian and the sidewalk are not important for this particular behavior. An illustration of this situation is shown in figure 2.3.
For the development of AVs a situation can be used as a data container which includes all necessary information that are needed for the planned behavior.

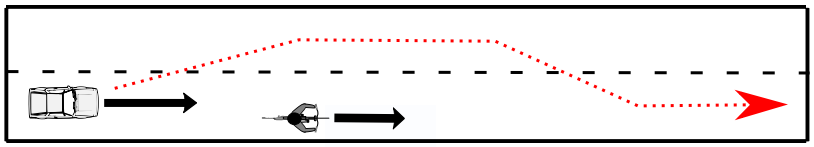


Figure 2.3: An example *traffic situation* in which an AV decides to overtake a bicyclists. In contrast to a *scene*, information are more specific to this behavior.

However, a situation only represents a snapshot in time. It can be useful for the planning and decision making for the next steps of the actor's behavior, but it does not describe a complete maneuver. For this purpose the term *(traffic) scenario* is used, which is defined by Ulbrich et al. as follows:

Definition 5 ***Traffic Scenario*** *"A (traffic) scenario describes the temporal development between several scenes in a sequence of scenes. Every scenario starts with an initial scene. Actions and events as well as goals and values may be specified to characterize this temporal development in a scenario. Other than a scene, a scenario spans a certain amount of time" [36, p.986].*

Given an initial scene and goals of the actors, a scenario can model a complete maneuver. As a scenario consists of a sequence of scenes, not only the entities involved in the maneuver but all entities in the environment are part of it.
In the example of the AV overtaking the bicyclist in urban traffic, the scenario starts with a scene before the maneuver and specifies the goal that the AV wants get ahead of the bicyclists. An illustration of this scenario can be seen in figure 2.4.

The representation of a scenario is often divided into four different levels of abstraction, which can be seen in figure 2.5.
The most abstract representation is the *functional scenario*. It consists of a human readable description and specifies in an informal way what is happening in the scenario. This abstraction level can also be supplemented with a visualization like the one in figure 2.4.
The second level of abstraction is the *abstract scenario*. It includes a formal description of the scenario focusing on the causal relations. It is machine readable and closely tied to an ontology.
The third level of abstraction is the *logical scenario*. It consists of a parameterized representation of the start situation and of the goals of the scenario. Here, the parameters are given in ranges in order to enable parameter variation.
The last level of abstraction is the *concrete scenario*. It describes a concrete instance of the *logical scenario* with all parameters set to one specific value. Hence, a *concrete scenario* is the executable level of abstraction from a scenario.

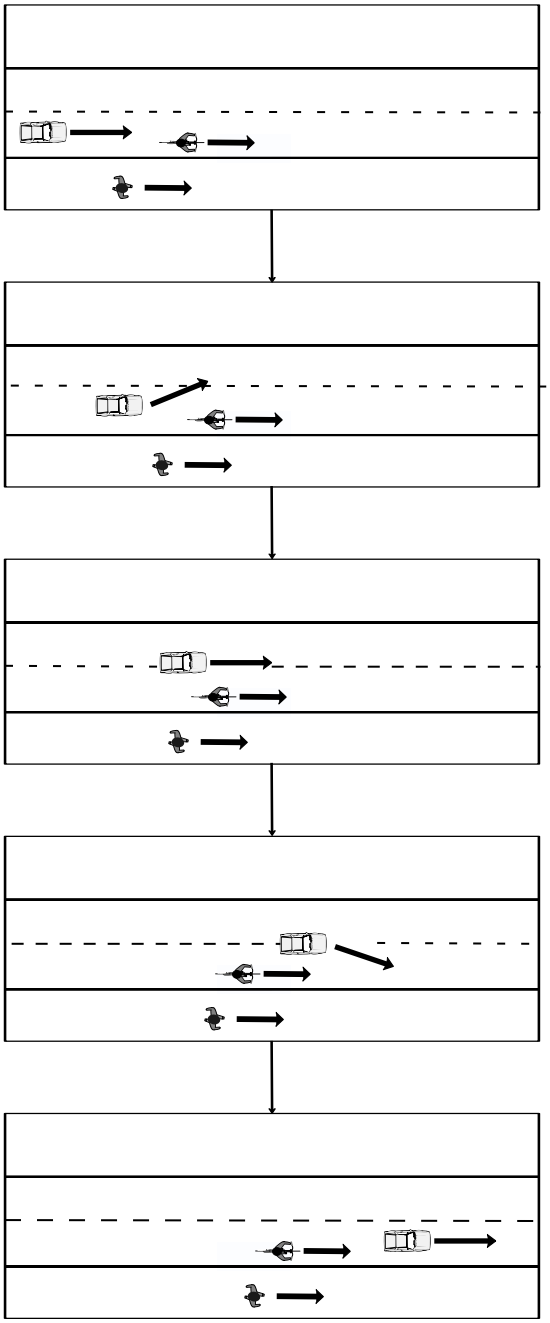


Figure 2.4: An example *traffic scenario* in which the AV overtakes the bicyclists. The initial scene is the *scene* from figure 2.2.

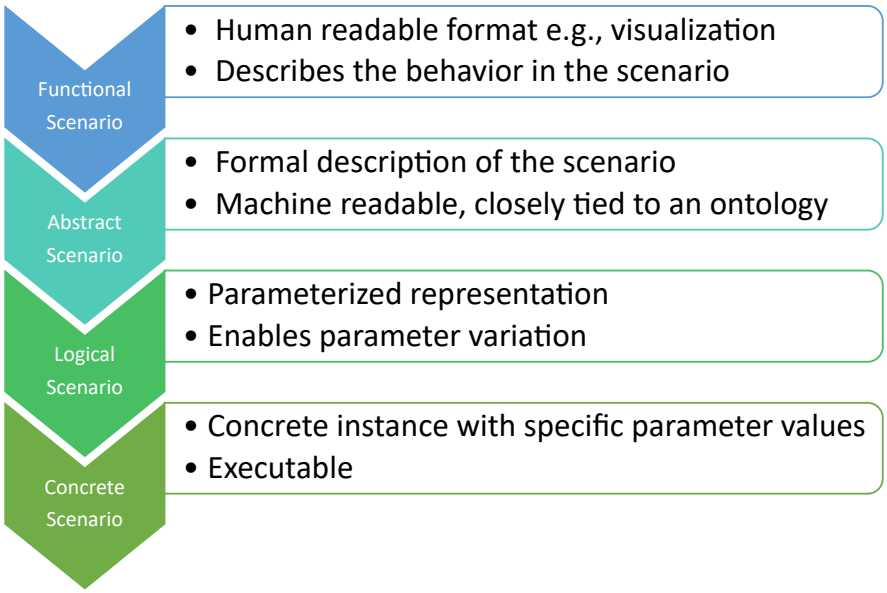


Figure 2.5: The four levels of abstraction of a scenario description used in the criticality analysis by Neurohr et al. [2].

When developing AVs, scenarios can be used to evaluate how well a maneuver is executed. However, this still requires a way to determine whether a maneuver was executed well. In order to identify possible circumstances in which AVs endanger VRUs in urban traffic, it is necessary to define what exactly constitutes a hazardous traffic situation.

2.2.1.2 Criticality and Criticality Phenomena

A term that is widely used in traffic research for this purpose is criticality [2, 37, 38]. Neurohr et al. define criticality as follows:

Definition 6 ***Criticality*** *"Criticality (of a traffic situation) is the combined risk of the involved actors when the traffic situation is continued" [2, p.3].*

Remark 6.1 *"In order to determine criticality, probabilities and types of harm, dynamical and behavioral models and actions restrictions of the involved actors are taken into account" [2, p.3].*

Remark 6.2 *"The time-horizon of the criticality of a situation is bound by the fulfillment of the intentions of the involved actors" [2, p.6].*

Remark 6.3 *"Criticality is inversely correlated with the amount of(sequences of) actions to avoid harm that are available to the involved actors" [2, p.6].*

According to this definition its remarks, criticality indicates not only the fact whether an accident has occurred, but moreover the probability and severity with which the actors in a traffic situation are exposed to harm. Hence, criticality refers to traffic situations, and thus relates to an certain behavior pattern at a particular point of time.

But criticality can easily be extended to a measure for traffic scenarios, by aggregating the criticality of the traffic situations included in the sequence of scenes in the scenario. The aggregation could for example be the sum or the maximum of the criticality of the situations.

Extending criticality measurements to entire traffic scenarios has the advantage of not only assessing whether a particular situation is dangerous, but also identifying classes of dangerous factors. These classes are also called *criticality phenomena* and are defined by Neurohr et al. as follows:

Definition 7 ***Criticality Phenomenon*** *"A criticality phenomenon is a concrete influencing factor in a scenario (or a combination thereof) which is associated with increased criticality" [2, p.6].*

With the term criticality phenomenon it is possible to describe concrete influencing factors, which could lead to an increased criticality in traffic. In case of the interaction of AVs with VRUs, such a criticality phenomenon is, for example, the occlusion of bicycles by parking cars that interferes with the perception of AVs.

2.2.2 Criticality Metrics

In order to analyze how criticality emerges in traffic it is further necessary to quantify criticality. Since criticality is not a quantity that can be measured directly, it is a common approach to work with measurement methods that approximate criticality.
These measurement methods are called criticality metrics and have already been used in several fields of research in the last decades, such as in traffic accident research or for the validation and verification of AVs [16]. Specifically, a criticality metric is defined as follows:

Definition 8 ***Criticality Metric*** *A criticality metric is a function $\kappa : \mathcal{S} \to \mathcal{O}$ that maps a traffic scene $S \in \mathcal{S}$ to a value on a predetermined scale of measurement $\mathcal{O} \subseteq \mathbb{R} \cup \{-\infty, +\infty\}$. This value represents the criticality measured in the scene.*

Remark 8.1 *Analogous to the definition of criticality (Definition 6), criticality metrics can also be extended to function $\kappa : \mathcal{CS} \to \mathcal{O}$ that maps a concrete scenario $CS \in \mathcal{CS}$ to a value that represents the criticality measured in the concrete scenario.*

Over time and through the wide field of application, several different criticality metrics have been introduced. A detailed overview of common criticality metrics and how they can be used for verification and validation of AV safety is given by Westhofen et al. in [16] and by Junietz et al. in [38]. Among the different metrics, a distinction can be made between deterministic and probabilistic criticality metrics [37].

Typical deterministic metrics are, for example, the *Time To X metrics*, such as *Time To Collision (TTC)*, which measures the time that one actor needs to collide into another actor, *Time To Headway (THW)*, which measures the time until one actor reaches the current position of another actor or *Time-To-Zebra (TTZ)*, which measures the time until on actor reaches a zebra crossing.
Other examples for deterministic metrics are the *Post Encroachment Time (PET)*, which measures the time gap between one actor leaving and another actor entering a designated conflict area, or the *Required Acceleration (a_{req})*, which denotes the acceleration that is required to bring the velocity of an actor to zero before a potential collision happens.
Note that not all of these metrics are always applicable, or at least only return a finite value if, for example, the predicted trajectories of the actors intersect.

Probabilistic criticality metrics on the other hand usually consider multiple possible trajectories of actors to calculate the probability and severity of any possible collision. For this purpose, multiple trajectory predictions are often combined with deterministic criticality metrics. For example, in the *Worst Time To Collision (WTTC)*, where the predictions are combined with *TTC*, or in the *Pedestrian Risk Index (PRI)*, where the predictions are combined with *TTZ*. This allows criticality metrics to cover a wider range of possible future scenes.

Which metric or combination thereof calculates a good approximation of criticality depends on the scene or scenario that is investigated [16, 37]. One metric, for example, can calculate an accurate value to represent criticality in one scenario, but fail to measure criticality in another one. Therefore, depending on the application, it must be decided which metrics are to be used. Typical benchmarks to determine if a metric is well suited for the investigated scene or scenario are the *reliability*, the *validity*, the *sensitivity* and the *specifity* of the metric.

The *reliability* of a metric is defined as the degree of closeness of repeated measurements to one another and refers to how consistent the results of a criticality metric are [16]. This means, in particular, that for two scenes with similar criticality, also the outputs of the criticality metric should have a similar size.

Further the *validity* of a metric is defined as the closeness of the metrics' measurement to representing the actual accident probability and severity [16]. This means that the output of the metric also corresponds to the emerging criticality in the scene. A metric with a high *validity* usually also has a high *reliability*, while vice versa is not necessarily given. However, to show high *validity*, a ground truth is needed about how critical a scene or scenario truly is.

If a ground truth is given, the *sensitivity* and the *specificity* of a metric can be calculated. Both values are related to the confusion matrix of a criticality metric, which can be seen

in Figure 2.6. In order to create a confusion matrix for non-binary criticality metrics, a threshold needs to be defined, at which a scene is classified as critical. The confusion matrix shows how many scenes were correctly or incorrectly classified as critical or non-critical.
Sensitivity, on the one hand, is than defined as the *true positive rate*, which is the number of the *true positive* scenes divided by the number of all scenes classified as critical. A high *sensitivity* indicates that a scene classified as critical by the criticality metric is very likely to be truly critical.
Specificity, on the other hand, is defined as the *true negative rate*, which is the number of the *true negative* scenes divided by the number of all scenes classified as non-critical. The value of the *specificity* indicates how high the probability is that a scene classified as non-critical is actually non-critical. If both *sensitivity* and *specificity* are high, also a high *validity* can be assumed.

		Criticality Metric Outcome	
		Classified as Non-critical	Classified as Critical
Ground Truth	Non-critical	*true negatives*	*false positives*
	Critical	*false negatives*	*true positives*

Figure 2.6: Confusion matrix of the outcome of a criticality metric compared to the actual criticality of a traffic scenes or scenario.

In order to analyze how criticality arises in urban traffic, one or more criticality metrics must be selected that are able to address the challenges described in subsection 2.1.2.

2.2.3 Application of Criticality Analysis

With the defined terms and the introduction to criticality mertrics, it is now possible to have a closer look at the criticality analysis, which is proposed by Neurohr et al. in [2] as part of the VVM – Verification and Validation Methods for Level 4 and 5 - project [1]. The criticality analysis is a method that maps an infinitely-dimensional traffic domain to a finite set of artifacts and thereby investigates how criticality emerges. In particular, Neurohr et al. define in their paper the following six high level goals of their method:

1. Extract criticality phenomena, i.e. observations of traffic that are associated with increased criticality.

2. Deliver explanations of the criticality phenomena by analyzing the possible underlying causalities.

[1] `www.vvm-projekt.de/en`

3. Derive a structuring of the open context according to these causalities.
4. Construct a catalog of abstract scenarios based on the classification, including representative instances.
5. Find an adequate level of abstraction for the criticality phenomena, explanations and scenarios.
6. Achieve a convergence towards a manageable set of criticality phenomena.

Hence, the aim of the criticality analysis is to derive a finite scenario catalog that covers the critical subspace of the investigated traffic domain. For this purpose, criticality phenomena and their underlying causal relations need to be identified. Further, a scenario class is to be created for each of the identified criticality phenomena. The process to achieve these six high level goals is divided in three different branches: the *method branch*, the *information branch* and the *scenario branch*. All three branches cooperate with each other. A detailed overview of the work-flow is shown in figure 2.7.

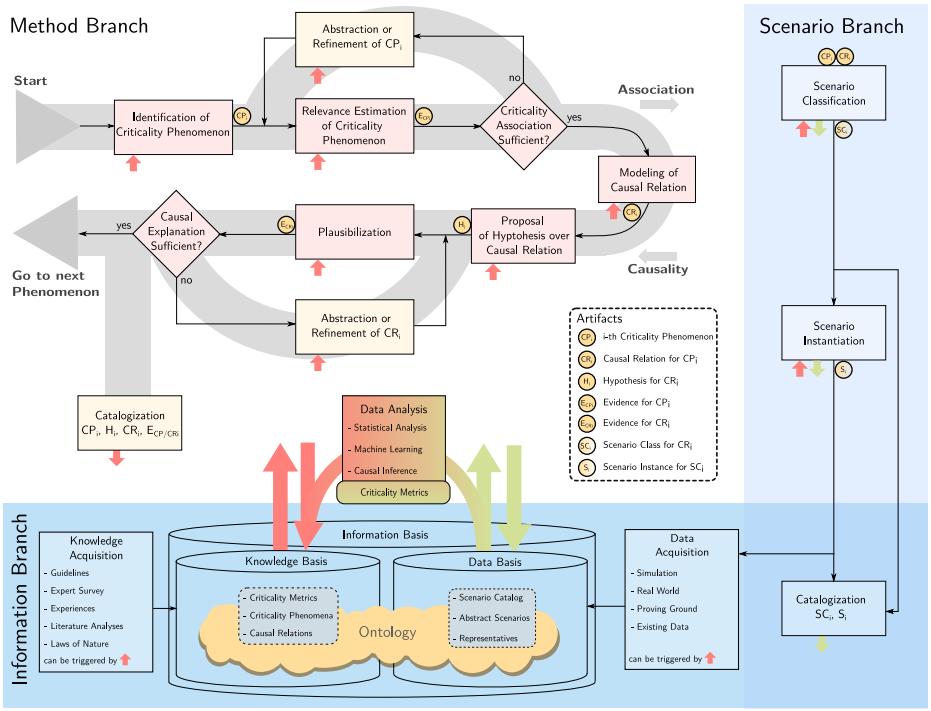


Figure 2.7: The procedure of the criticality analysis proposed by Neurohr et al. in [2].

The *method branch* involves the process of identifying criticality phenomena, investigating the underlying causal relationships, and gathering evidence for their plausibility [2]. The goal is to identify all influencing factors under in which criticality arises in the investigated traffic domain. Therefore also criticality metrics play an important role.

The relevant information that are needed in the *method branch* needs are supplied by the *information branch*. For this purpose, the information is divided into a knowledge and a

data basis.
The knowledge basis, on the one hand, includes general facts about the world. For the acquisition of the knowledge basis, traffic research analyses, expert opinions as well as guidelines and traffic laws are studied.
The data basis, on the other hand, includes concrete instances and facts. The information for the data acquisition comes from accident data bases, driving studies and sensors of manual or part automated vehicles. The collected data is then analyzed using statistical and machine learning approaches.

The last branch, the *scenario branch*, describes how a finite scenario catalog covering the identified criticality phenomena is constructed. For this purpose, in a first step, *abstract scenarios* are constructed that cover the criticality phenomena identified on the *method branch*. In a second step, *logical* and *concrete scenarios* are instantiated to derive more information for the data requisition. In a last step, a finite scenario catalog is created which adequately covers the critical subspace of the analyzed traffic domain.

Thus, by performing the criticality analysis, a finite set of scenarios is created that covers all possible circumstances under which AVs have trouble detecting or predicting VRUs. To achieve this, however, it is not sufficient to focus only on functional scenarios that model all criticality phenomena. The mere presence of a criticality phenomena in a scenario is not sufficient to make the scenario critical. For example, a *concrete scenario* in which a pedestrian is occluded for an AV by parking cars, but the pedestrian crosses the road long after the AV has passed has a low criticality. For a successful criticality analysis it is therefore necessary to identify all kinds of *concrete scearios* in which the criticality phenomena causes a high criticality.

2.3 Evolutionary Algorithms

In order to identify all kinds of *concrete scenarios* with high criticality that can be derived from a *logical scenario*, a large parameter space must be searched. In particular, all parameter ranges from all parameters defined in the *logical scenario* must be combined with each other and the criticality of the resulting *concrete scenarios* must be evaluated. This space quickly becomes infeasible large, especially as it grows exponential with the number of parameters that are necessary for the scenario description.

One approach from the field of mathematics to nevertheless approximate optimal solutions in infeasible large spaces is to apply optimization algorithms. In general, optimization algorithms deal with the organized search for solutions that yield the best value for one or more objective functions [39].
For the search of critical *concrete scenarios* in the space of all possible *concrete scenarios*, a solution could be the parameter assignment of a *concrete scenario* and the objective

function a criticality metric. In this way, an optimization algorithm searches for the *concrete scenarios* that yield the highest criticality.

One group of optimization algorithms are the evolutionary algorithms (EAs). In subsection 2.3.1 an introduction to how EAs work and an overview of different variations of EAs is given. Further, subsection 2.3.2 summarizes how EAs are already being used to identify critical scenarios in traffic.

2.3.1 Introduction to Evolutionary Algorithms

Evolutionary algorithms (EAs) are a group of blackbox optimization algorithms that are linked to the field of computational intelligence [40]. In particular, EAs are problem-independent optimization algorithms that have their roots in the group of metaheuristic algorithms [41] and are able to find good solutions to problems whose exact solution is infeasible [42]. In contrast to many other optimization algorithms, EAs are able to solve problems no matter if they are large, complex, noncontinuous, nondifferentiable, or multimodal [43]. Especially in multimodal spaces, EAs are able to find different optimal solutions in a single optimization run [40, 44], which makes them a promising approach for identifying different kinds of critical *concrete scenarios* that can be derived from a *logical scenario*.

Like many algorithms in the field of computational intelligence, EAs are inspired by nature. In particular, EAs combine the classical Darwinian evolutionary theory with the selectionism of Weismann and the genetics of Mendel to evolve solutions of a problem iteratively [42, 43]. Therefore, much of the terminology used to describe EAs is also derived from nature.
For instance, a single solution of the addressed problem is often referred to as individual and a set of different solutions is called a population. Further, EAs work iteratively and every step of the iteration is called a generation. Also the objective function is referred to as fitness function and the value of the objective function is referred to as fitness.

The work-flow of an EA is visualized in figure 2.8. EAs usually start with an initial parent population, i.e., a set of potential solutions of the addressed problem. Each of these solutions, also called individual, that are part of the parent population is either generated randomly or sampled from a set of already known solutions [45]. The initial parent population is displayed by the red dots at the top of figure 2.8.
In the next step, a set of new individuals is generated as child population, which is represented by the black dots in the figure. This generation is performed through crossover between individuals from the parent generation, which mixes the genetic material of the parents, and an additional mutation, which adds randomness to the newly created individuals [40].
In the third step, the fitness of all individuals from the child population is evaluated.

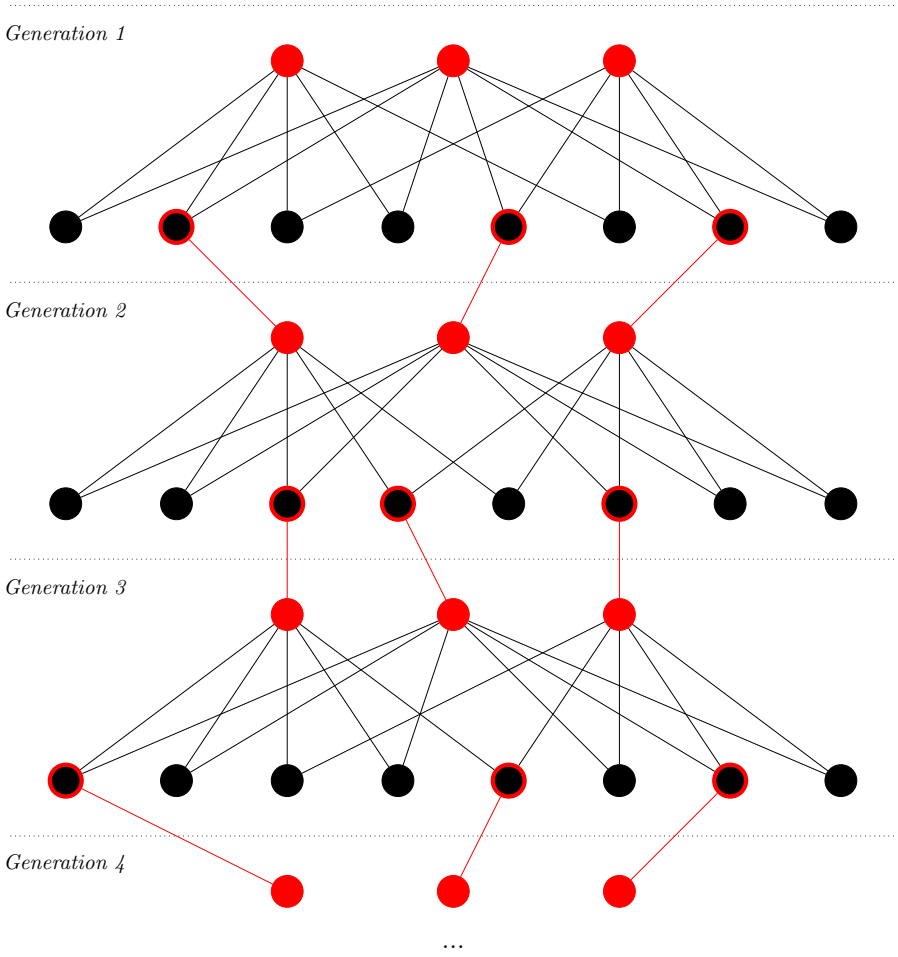


Figure 2.8: Visualization of the process of evolutionary algorithms (EAs). The red nodes represents individuals from the parent generation and the black nodes represent individuals from the generated child generation. The individuals from the child generation with the highest fitness (black node, red circle) are taken as parents for the next generation.

According to the principle *survival of the fittest*, the individuals with the highest fitness are then used as parent population for the next generation [43], which is illustrated by the black dots with the red border in figure 2.8.
After this, a new generation starts with the new parent population. This process repeats itself until a stopping criterion is fulfilled.

The size of the populations depends on the variant of the algorithm. It can either be fixed or change dynamically throughout the evolutionary process [45]. Also the way how the crossover and the mutation work can be adjusted with different hyper parameters.

In this work, the two variants $(\mu + \lambda)$ and $(\mu - \lambda)$ are investigated, where μ denotes the size of the parent population and λ denotes the size of the child population. Further they are both combined with Rechenberg's 1/5th success rule as well as to the principle of self adaptation.

2.3.1.1 $(\mu + \lambda)$ and $(\mu - \lambda)$ Algorithm

The $(\mu + \lambda)$ and $(\mu - \lambda)$ algorithms are two basic variants of EAs, which for the most part follow the same procedure. The workflow of these algorithms is illustrated in figure 2.9.

In both algorithms, the variable μ indicates the size of the parent population. It specifies as well the number of individuals that are produced for the initial parent population as the number of the fittest individuals from a child population that are used as parents for the next generation. The size of the parent population μ can be chosen as a natural number greater or equal to one and is thereafter fixed for the entire algorithm.
The variable λ further indicates the size of the child population, which is the amount of new individuals created in every generation. The size of λ can be chosen as a natural number greater or equal to μ and is also fixed for the entire run.

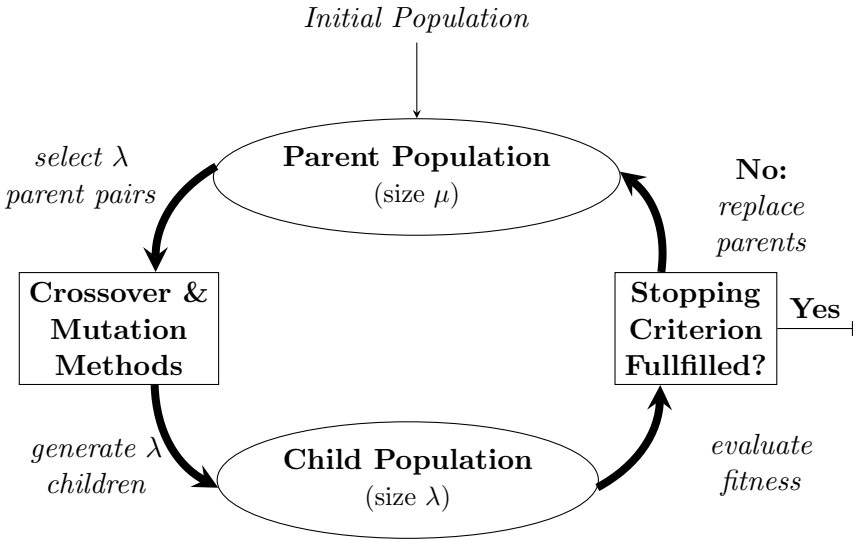


Figure 2.9: The procedure of the $(\mu + \lambda)$ and $(\mu - \lambda)$ algorithms. In a first step, λ parent pairs are selected from the parent population. These parent pairs generate λ children with crossover and mutation in the second step. In the thrid step, the newly created individuals are evaluated. At the end, the μ individuals with the highest fitness are used as parent population for the next generation.

For the crossover, a variable κ is set that determines how many individuals from the parent population are involved in the creation of a new individual. This number could be chosen as $\kappa = 2$ like in nature or as any other natural number between 1 and μ.
The way how the crossover works can vary depending on the characteristics of the individuals. For example, in the application where an individual is the parameter assignment of a *concrete scenario*, the crossover method could be that for each parameter it is decided randomly from which parent it shall be inherited. But also other methods such as taking the average of all involved parents are applicable.

After the crossover, a random mutation is applied on the newly created individuals. This

random mutation is usually the addition of a normal distributed random number to every dimension of the individual. The magnitude of this random number is determined with a previously set step size σ, which decides how much the children deviate from the parents. This step size is also fixed for the entire algorithm.
It also applies to the mutation that the type of change can be different for different characteristics of parameters, depending on whether the parameters are continuous, discrete, or binary. In addition, it must be ensured that neither the crossover nor the mutation leave the space of valid solutions to the problem.

After the new child population is generated with crossover and mutation, its fitness is evaluated. For this purpose, the fitness function is applied to each of the new generated individuals.
Before the parents for the next generation are selected, it is checked if a stopping criterion is fulfilled. This stopping criteria could, for example, be that a predefined fitness value is reached or that the algorithm has performed a specified number of generations.
If the stop criteria is fulfilled, the algorithm terminates. Otherwise, the next generation is started with the best individuals from the previous evaluated generation as parent population.

The difference between the $(\mu + \lambda)$ and $(\mu - \lambda)$ algorithms is which individuals are considered as potential parents for the next generation.
While the $(\mu + \lambda)$ algorithm chooses the μ individuals with the highest fitness from both, the previous parent population and the newly generated child population, the $(\mu - \lambda)$ only considers the individuals from the child population.
This can make the $(\mu - \lambda)$ algorithm slower but also lowers its chance of getting stuck in local optima.

2.3.1.2 Rechenberg's 1/5th success rule

While the step size σ of the parameter mutation is fixed in the $(\mu + \lambda)$ and $(\mu - \lambda)$ algorithms, it can be reasonable to adjust it dynamically in order to speed up the optimization process and make it more flexible [40].
One common approach to do so is the *1/5th success rule* proposed by the German computer scientist Ingo Rechenberg. It states that the step size σ should be increased if the rate at which the algorithms finds solutions with a higher fitness than the current best solution is larger then 1/5, while it should be decreased if the rate is lower [42].

The idea behind this is that if better solutions are frequently found this is an indication that the space currently being searched does not contain an optimum. The increased step size leads to a wider parameter variation and therefore also to a bigger space being searched.
If, in contrast, better solutions are only rarely found, it indicates that the currently best

found solution is near to an optima. The reduction of the step size leads to a closer examination of the immediate environment of the currently best solution. This increases the chance to find the optima.

However, when applying an EA with Rechenberg's rule, attention must be paid to ensure that the algorithm does not get stuck in a local optima.

2.3.1.3 Self Adaptation

Another way to adapt the step sizes dynamically is the principle of self adaptation. This principle is a mechanism that uses the mechanics of evolution in the algorithm to adapt the step sizes of the mutation [46].
Self adaptation is based on the idea that it is worthwhile to continue searching with a certain step size if a good solution has been found with it. Therefore, not only are the best individuals from one generation used as parents for the next generation, but also the step sizes used to generate them are transferred to the next generation. In this way, the algorithm learns not only where to find good solutions, but also how to mutate the solutions to find even better ones.

In contrast to Rechenberg's rule, where the step size is adjusted equally for all parameters, self-adaptation offers the possibility to increase the step size for one parameter and simultaneously decrease the step size for another. This enables a more targeted search for optima.
Both, Rechenberg's 1/5th success rule and the principle of self adaptation, can be combined with the $(\mu + \lambda)$ and $(\mu - \lambda)$ algorithms.
Which combination thereof is best suited to identify *concrete scenarios* with a high criticality that can be derived from a *logical scenario* is investigated in this master's thesis.

2.3.2 Existing Research Approaches

Although the research on evolutionary algorithms (EAs) has began already over 50 years ago [45], their application for the verification and validation of AVs has only recently become the subject of investigation. Since then, two papers have been published which primarily focus on how EAs can be used to generate traffic scenarios with a high criticality.

The first paper was published by Klischat and Althoff [47] in 2019. In their work EAs are used to generate critical scenarios that can be used for the testing of the motion planning algorithms from AVs. In particular they apply EAs to minimize the drivable area that AVs can use in a scenario without having a collision.
As a result of this approach, it is shown that EAs are well able to generate scenarios that are useful for the testing of motion planners. Thereby, EAs are able to deal with complex road layouts as well as dynamics for a high number of involved traffic participants [47].

However, in this approach EAs are only applied on traffic scenarios without VRUs, and thus an important part of the urban traffic domain is not considered. Moreover, the only indicator of criticality that is examined is whether a collision occurs. With respect to definition 6 of criticality, other aspects such as the probability and the severity of a potential accident would be reasonable to consider as well.

The second paper from Bussler et al. [48], published in 2020, takes a closer look at criticality in the urban traffic domain. In their approach EAs are also used to identify parameter combinations from *logical scenarios* that lead to *concrete scenarios* with a high criticality. Therefore, a scenario at an urban intersection with an AV, an oncoming conventional car, and a crossing pedestrian is investigated. As fitness function for the EA, the criticality metric TTC as well as the minimal distance from the AV to the other actors are compared. With this approach, Bussler et al. [48] show that EAs are able to find critical *concrete scenarios* inside *logical scenarios* and at the same time identify correlations of parameters that lead to critical situations.
However, they also note that the used criticality metrics are not sufficient to cover criticality in urban intersection scenarios adequately, especially when it comes to situations with a crossing pedestrian. Further, they only investigated one scenario with one VRU and thus only cover a small subset of the challenges that VRUs can cause AVs in urban traffic.

Overall, both papers indicate that the application of EAs is a promising approach to generate critical *concrete scenarios* in traffic and thereby identify critical situations between AVs and other traffic participants. However, in order to examine how well EAs are applicable for the generation of critical *concrete scenarios* between AVs and VRUs, further research steps are necessary.
In particular it needs to be analyzed how EAs perform on a set of traffic scenarios that cover the challenges between VRUs and AVs outlined in subsection 2.1.2. Moreover, a fitness function for EAs has to be designed and evaluated that is able to recognize all kind of critical situations in urban traffic.

In the following, four research questions are formulated to investigate how well EAs can generate critical *concrete scenarios* in urban traffic and how well this generated scenarios can be used to identify different kinds of critical situations.

3 Research Questions

For the application of EAs to analyze criticality in the urban traffic domain several aspects have to be investigated. The previous chapter introduced an approach how EAs can generate critical *concrete scenarios* from predefined *logical scenarios*. In this chapter, four research questions are formulated that aim to examine how this approach can be implemented and how the results can be used for the analysis of criticality between AVs and VRUs in urban traffic.

The first problem to be addressed is that the presence of VRUs also makes it more complicated to analyze criticality. In order to apply EAs on traffic scenarios that include VRUs, a safe environment is necessary. Therefore, the first research question investigated in this master's thesis is the following:

RQ 1 *How can the emergence of criticality be analyzed when AVs and VRUs interact in an urban intersection setting without endangering VRUs?*

Once a safe environment is found in which traffic scenarios containing AVs and VRUs can be executed, EAs can be applied to optimize criticality. For this optimization a fitness function is needed that calculates a good quantification of the emerging criticality in urban traffic scenarios. In particular, this fitness function needs to take the challenges into account that are caused by the interaction of AVs with VRUs.
Accordingly, the second research question in this master's thesis is the following:

RQ 2 *How can criticality in urban intersection scenarios with AVs and several VRUs be measured?*

When a suitable fitness function is found, EAs can be applied to generate *concrete scenarios* that yield a good output of the fitness function and thus have a high criticality. Hence, as a result of an EA not only the best found solution is returned but moreover a set of *concrete scenarios* with a high criticality that can be derived from a predefined *logical scenario* is obtained. In order to understand how criticality arises when AVs and VRUs interact in urban traffic, the critical situations that occur in the generated *concrete scenarios* need to be analyzed. In particular, the following two research questions are examined:

RQ 3 *How is it possible to identify critical situations between AVs and VRUs in large parameter spaces of urban intersection scenarios?*

RQ 4 *How can it be ensured that all kinds of critical situations in an urban intersection scenario are identified?*

While research question 3 focuses on the question whether the generated *concrete scenarios* can be used to identify critical situations between AVs and VRUs in urban traffic, research question 4 addresses the problem how it can be assured that the *concrete scenarios* cover all major classes of possible critical situations. For the validation and verification of AVs in urban traffic it is important that, on the one hand, critical situations are correctly identified and that, on the other hand, no critical situation is overseen.
Hence, when EAs are applied to generate critical *concrete scenarios*, it is necessary to examine not only whether they are able to find a good solution in the parameter space of a *logical scenario*, but moreover whether EAs search for optima in the entire parameter space.

In the following chapter the methodology used to examine these four research questions is explained.

4 Methodology

This chapter provides an overview of the methodology used to investigate the previously presented research questions.
Therefore, as an approach to answer research question 1, simulation-based testing and the Carla simulator are introduced in section 4.1. After this, in section 4.2, three urban traffic scenarios are presented which model challenges between AVs and VRUs. These scenarios are designed in a way that they can be executed in the Carla simulator. In section 4.3 a new criticality metric is proposed, which aims to calculate a good approximation of criticality in the designed scenarios and thus answers research question 2. In the last section 4.4, the implementation of different EAs is explained and approaches to investigate how well the implemented EAs are able to generate critical *concrete scenarios* are outlined. In particular, these approaches aim to investigate research question 3 and 4.

4.1 Simulation-Based Testing of AVs

As addressed in research question 1, a safe environment in which critical scenarios can be evaluated without endangering VRUs is necessary for the validation and verification of AVs in urban traffic.
As described in subsection 2.1.2, the accident severity is particularly high for VRUs. This makes it unethical to analyze criticality in real life traffic. Since test grounds with crash dummys are expensive in a economic and time-related way, and the generation of critical scenarios with EAs needs a large number of repetitions, this option is also not feasible.

One approach that is common in the field of validation and verification of AVs is to use virtual simulations. Not only because the damage caused by accidents and the associated endangered human lives are no longer a problem, but additionally as the test distance predicted for the approval of AVs is hardly achievable only by real life testing [47].
Testing the behavior of AVs in a simulation allows the repetition of a large number of scenarios and environments with different characteristics [49]. At the same time, simulation tests offer the advantage of being simple, inexpensive and easy to reproduce [50]. Especially agent-based simulations are capable of simulating complex systems, such as an ADS [51].

However, simulations only approximate the real world. When using simulations it is always important to consider how realistic the conditions are. For example, if an ADS masters a scenario in a simulation, it does not necessarily mean that the ADS will master this

scenario in real life. Thus, simulation testing should always be used supplementary to real life testing.

Overall, it can be said that agent-based simulations provide good conditions to conduct a first analysis of criticality in urban traffic scenarios. The critical situations identified in simulation based testing can be used to improve the ADS. This could prevent these critical situations from occurring in real life testing and endangering human lives. Therefore, it will be investigated whether agent-based simulations are a suitable environment to analyze the emergence of criticality between AVs and VRUs and thus are a first step to answer research question 1.
In the next subsection 4.1.1, the open-source simulator Carla, which will be used in this work, is introduced. Furthermore, subsection 4.1.2 describes the different agents used to control the behavior of AVs and VRUs in the simulation.

4.1.1 Carla Simulator

The open-source simulator Carla[1] has been developed to support the development, training, and validation of AVs. In particular, Carla is developed as simulator for urban driving and therefore includes urban layouts, street signs, traffic lights and additionally to a multitude of four-wheeled road vehicle models also models for pedestrians and bicyclists. Further, Carla uses the Unreal Engine 4[2] with the aim to offer realistic physics. This is especially important when the accident severity is evaluated [5].

The Carla simulator is build in a client-server architecture. While the server handles everything related to the simulation, such as the sensor perception, the physics computation or the states of the actors, the client handles the logic of actors and the build up of scenarios. The architecture is supported by the Carla Python Api[3], an interface that mediates between server and client [52].

When it comes to the application of EAs to optimize criticality in urban traffic scenarios, the Carla Python Api brings many advantages.
Firstly, Python is a well suited programming language to implement EAs. Due to the API, the complete logic behind the EAs can be programmed in it. The parameter assignments generated by the algorithm can then directly be transferred to the Carla server, where the *concrete scenario* is simulated.
Secondly, all information needed for the evaluation of criticality metrics, such as speed, heading angle, position or mass of the actors, can be easily retrieved from the simulation via the API. This makes it possible to compute the outcome of criticality metrics in Python and thus evaluate the fitness of the executed simulation.
In this way, the whole process of generating, executing and evaluating *concrete scenarios*

[1] `https://carla.org`, assessed on 17.05.2022
[2] `https://www.unrealengine.com`, assessed on 17.05.2022
[3] `https://carla.readthedocs.io/en/latest/python_api/`, assessed on 17.05.2022

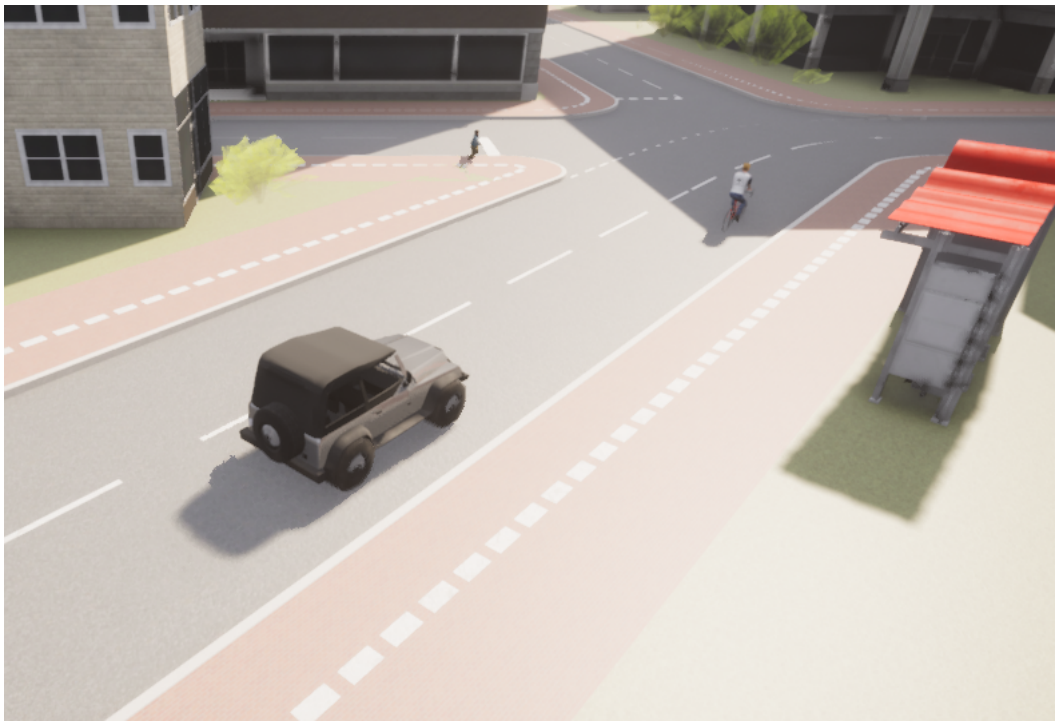


Figure 4.1: Screenshot of the CARLA simulator in which a AV interacts with a pedestrian and a bicyclist.

can be managed with PYTHON and CARLA.

4.1.2 Simulation Agents

In order to evaluate the behavior of AVs in traffic simulations, an agent acting as an ADS is needed. The tasks of this agent include sensor perception evaluation, decision making, and vehicle control.
For this purpose, different agents are available in CARLA. The most complex agent predefined in CARLA is the *Behavioral Agent* [4]. This agent is able to reach a destination in the shortest possible time, while obeying traffic rules and being considerate of other traffic participants [53].

For the agent used to control AVs in this master's thesis, the *Behavioral Agent* is extended with additional sensors and functions.
This includes sensors that are able to recognize parking vehicles and any possible occlusions that may result from them.
Further, front sensors are added to better detect VRUs in front of the vehicle. While the original agent was only able to detect VRUs when they were located in the middle of the lane, the adapted agent responds to VRUs regardless of their lane position.
With these extensions the *Behavioral Agent* can be used to investigate the research questions.

[4] `https://github.com/carla-simulator/carla/blob/master/PythonAPI/carla/agents/navigation/behavior_agent.py`, assessed on 17.05.2022

In contrast to the agent that controls the behavior of the AVs, the agents that control the behavior of the VRUs are chosen to be less complex. The idea behind this is that when AVs are to participate in real life traffic, they also have to deal with VRUs that behave careless, such as a child running on the road without looking. If the actors controlling the VRUs would be smart enough to recognize the AV in advance and change their behavior to avoid it, these kind of critical situations would no longer occur.
Therefore, the actors used for VRUs in this master's thesis strictly follow their waypoints and only react to other road users shortly before a collision would happen. In particular, a variant of the *Basic Agent*[5] is used to control bicyclists and a variant of the *WalkerAIController*[6] is used to control pedestrians.

A disadvantage of CARLA is that even though different weather conditions can be set, the adhesion of the road always stays the same. Further, the perception of the environment of the implemented *Behavioral Agent* is independent of the light conditions. Therefore, critical situations that are caused by bad light conditions or heavy rainfall can not be investigated in the simulator.

4.2 Scenario Spaces

For the evaluation of all four research questions, traffic scenarios are needed that cover a wide range of challenges in urban traffic. Specifically, three different scenarios containing several criticality phenomena are designed and implemented in CARLA. All three scenarios take place at urban intersections, as the majority of accidents in urban traffic occurs on those [4]. However, street design and arrangement of actors is different for the scenarios.

Further, it has to be noted that the three designed scenarios are not a complete scenario catalog created by a criticality analysis. They do not cover all critical phenomena occurring in the urban traffic domain, but rather represent a set of scenarios that cover typical challenges that AVs face when interacting with VRUs in urban traffic and, at the same time, investigate features with whose the performance of EAs can be evaluated.
Therefore, the three scenarios vary in their complexity, the number of participating actors and the properties of the parameters.

In the following subsections the three scenarios are described. Thereby, they are first introduced as *functional scenarios* and than modeled as *logical scenarios* that can be implemented in CARLA.

[5] `https://github.com/carla-simulator/carla/blob/master/PythonAPI/carla/agents/navigation/basic_agent.py`, assessed on 17.05.2022
[6] `https://carla.readthedocs.io/en/latest/python_api/#carla.WalkerAIController`, assessed on 17.05.2022

4.2.1 Scenario 1: Occlusion in Urban Traffic

The criticality phenomena addressed in the first scenario is the occlusion of bicycles by parking vehicles in urban traffic. For this purpose, the scenario is composed of a straight street with a side street and parking vehicles as scenery and an AV and a bicyclist as actors. An illustration of the scenario is shown in figure 4.2.
In the initial scene, the parking vehicles are placed on one side of the straight street and stay there as static objects for the whole scenario run. The AV starts on the straight street before the side street and drives pass it. The bicyclist starts riding on the side street and turns left onto the straight street.

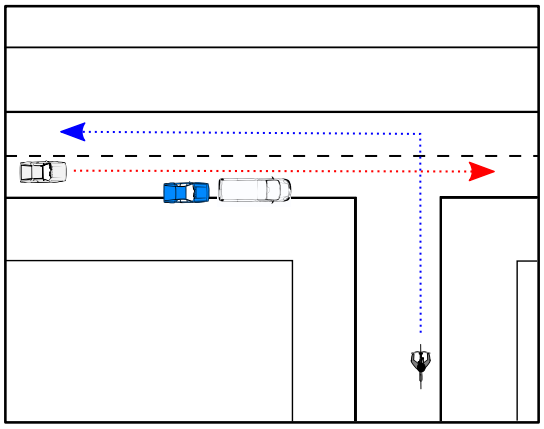
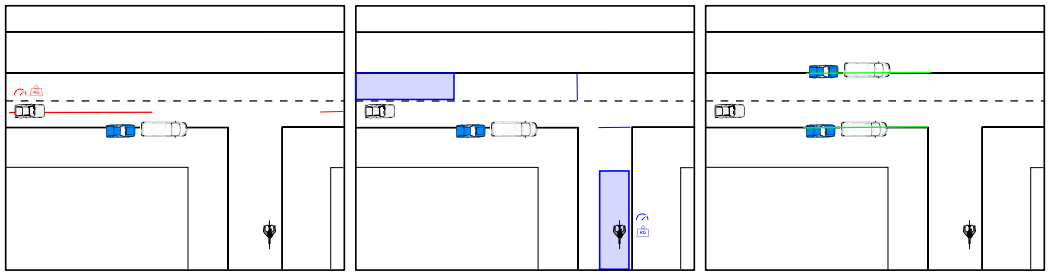


Figure 4.2: Abstract visualization of scenario 1. The AV drives straight on the street while the bicyclist makes a left turn onto the opposite lane of the AV. The parking vehicles on the side can lead to an occlusion of the bicyclist and thereby to a high criticality.

Criticality can arise in this scenario if the parking vehicles, the AV and the bicyclist are arranged in a way that the bicyclist is occluded for the AV at the time when both actors reach the intersection of the two streets. Hence, to model this *functional scenario* as a *logical scenario*, the arrangement of the AV, the bicyclist and the parking vehicles need to be parameterized.

The parameters that need to be considered to model the AV are its start position, its target position, the mass of the vehicle and the target speed of the vehicle.
As four-wheeled road vehicles usually drive in the middle of the lane, only the distance to the intersection has to be chosen for the start and the target position. The mass of the AV can be any value between 700 kg, which corresponds e.g. to the mass of a Smart, and 2000 kg, which corresponds e.g. to the mass of a Jeep.
The target speed can be any value up to 50 km/h, which is a usual speed limit in many cities. As vehicles in CARLA are placed in the world in a standing position and than accelerate to their target speed over time, the combination of target speed, distance of the start position to the intersection and the decision making of the ADS influences the speed

the AV has when it reaches the intersection.
A visualization of the parameter space $S_{1,av}$ is shown in figure 4.3 (a). For the parameters that determine the characteristics of the AV in the *logical scenario* holds $S_{1,av} \subset \mathbb{R}^4$. A detailed listing of all parameters from $S_{1,av}$ is shown in the appendix in tabular A.1.



(a) (b) (c)

Figure 4.3: A visualization of the parameters needed to model scenario 1 as *logical scenario*. (a) the starting position, target position, target speed and mass of the AV (b) the starting position, two further waypoints, the target position, the target speed and the mass of the bicyclist (c) the side, the amount and the distance to the intersection of the parking vehicles.

Similar properties as for the AV need to be modeled for the bicyclist. However, as bicycles move more flexible than four-wheeled road vehicles, the parameters that describe their movement are slightly different. The implementation of this behavior in CARLA causes some difficulties.
First, bicyclists can ride on both, the road and bicycle lanes. However, as bicycle models in CARLA are implemented similar to four-wheeled road vehicles, all bicycles have to drive on the road. As a result, scenarios in which a bicyclist drives on a bike lane can not be easily implemented in CARLA. To nevertheless model the behavior of bicyclist on the road realistically, their trajectories are not always located in the middle of the road. Thus, not only the distance to the intersection but also the position on the lane are considered as parameters for the start and the target position.
Further, as bicyclists do not always drive as straight as four-wheeled road users a variation in their trajectories would be reasonable. However, since the routing of CARLA agents plans trajectories in a straight line through given waypoints, this cannot be easily implemented either. In order to again model the behavior of the bicyclists in a more realistic way, two more waypoints, one shortly before and one shortly after the intersection, are added to the bicyclist's route planning. This additionally enables a variation in the angle in which the bicyclist turns.
As a result, the bicyclist in this scenario navigates from the starting point to the destination point via a point before the turn and a point after the turn, which makes his trajectories more flexible than those of the AV. The mass and target speed of the bicyclist are chosen similarly, but with a mass between 40kg and 100kg and a speed limit of 25 km/h, the values are smaller than those of the AV.

A visualization of the parameter space $S_{1,bic}$ is shown in figure 4.3 (b). For the parameters that determine the movement and properties of the bicyclist holds $S_{1,bic} \subset \mathbb{R}^8$. Again, a detailed overview of the parameters is shown in the appendix in tabular A.2.

In order to describe the arrangement of parking vehicles, the number of vehicles, the side on which they are placed and the distance from the first parking vehicle to the intersection are used as parameters.
The parameters for the parking vehicles are particularly interesting because, unlike the ones for the AV and bicyclist where all parameters are continuous, the number of parking vehicles is a discrete and the side of the parking vehicles is a binary parameter. This makes the parameter space more challenging and can therefore be used to investigate how well EAs are able to handle it. Especially the side of the parking vehicles is an interesting characteristics for the evaluation of EAs as, from an expert's view, the vehicles should always be placed on the right side of the road to create critical situations.
A visualization of the parameter space $S_{1,obs}$ is shown in figure 4.3 (c). For the parameter space that determines the arrangement of the parking vehicles holds $S_{1,obs} \subset \mathbb{R} \times \mathbb{N}_0^2$. A detailed overview is shown in the appendix in tabular A.3.

Overall, the parameters space of scenario 1 is composed as $S_1 = S_{1,av} \times S_{1,bic} \times S_{1,obs} \subset \mathbb{R}^{13} \times \mathbb{N}_0^2$.

4.2.2 Scenario 2: Driving Behind a Bicyclist in Urban Traffic

In the second scenario the challenges an AV faces when it drives behind a bicyclist in urban traffic are addressed. The scenery used for this scenario is an urban intersection with a street at an acute angle. The actors in the scenario are a bicyclist, an AV and a pedestrian. An illustration of this scenario is shown in figure 4.4.
The bicyclist starts on the road with the acute angle and turns left at the intersection. The AV starts behind the bicyclist and also turns left at the intersection. Since overtaking of the bicyclists is not readily possible with the *Behavioral Agent* from CARLA, the AV stays behind the bicyclist the whole time. However, driving behind a bicyclist at an intersection with an acute angle can also cause criticality in urban traffic. The pedestrian starts on the sidewalk next to the target road of the two vehicles and crosses either the road from which the two vehicles are coming or the road where the two vehicles are heading.

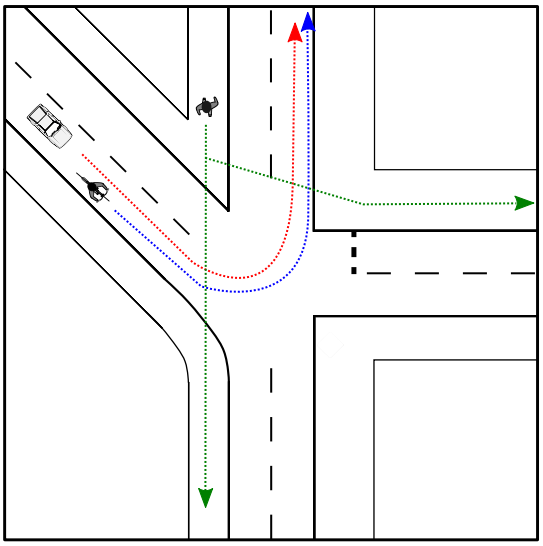
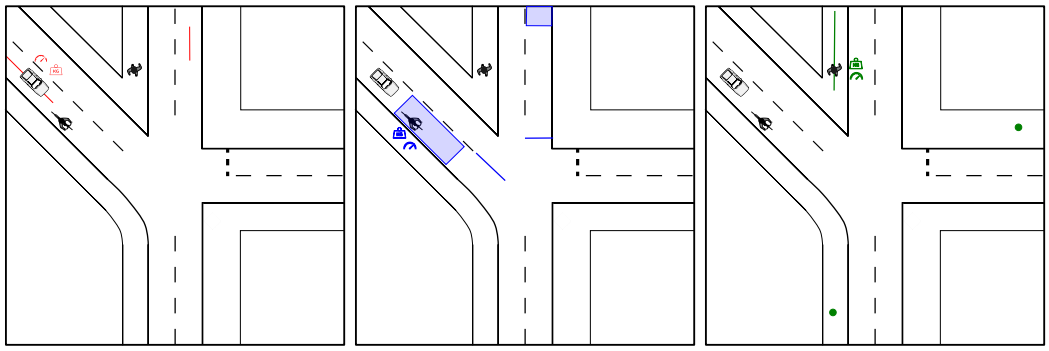


Figure 4.4: Abstract visualization of the scenario 2. The AV drives behind a bicyclist and turns left at an intersection with an acute angle. A pedestrian crosses the road before or after the left turn. Criticality can arise in different situations.

In contrast to scenario 1, in which critical situations only occurs when the bicyclist is occluded, the AV faces multiple challenges at the same time in scenario 2. These include driving behind a bicyclist without endangering the VRU, turning at an intersection in an acute angle and predicting where a pedestrian crosses the road. Depending on the composition of the actors, those task can challenge each other and lead to critical situations. Hence, with respect to research question 4, it is particularly interesting to see whether EAs are able to identify different types of critical situations or whether they only further optimize the first critical situation they have found.

Since an AV and a bicyclist have already participated in scenario 1, these actors can be modeled in the *logical scenario* in a similar way. Therefore, the parameter spaces that define the behavior of the AV and the bicyclist are similar to those in the first scenario. The parameter space $S_{2,av}$ again consists of the distance to the intersection of the start and the target point as well as the mass and the target speed of the AV and is therefore subset of the space $\mathbb{R}^4$. A visualization of these parameters is shown in figure 4.5 (a). Likewise, the challenges caused by implementation of the bicyclists behavior in CARLA are the same in this scenario. Thus, the parameter space $S_{2,bic}$ consists of the same parameters necessary to define the four waypoints that the bicyclist passes, the mass of the bicyclist and the speed of the bicyclist. Hence, the parameter space is again subset of $\mathbb{R}^8$. A visualization of these parameters is shown in figure 4.5 (b).
Detailed listings of all parameters from $S_{2,av}$ and $S_{2,bic}$ are shown in the appendix in tabular A.4 and tabular A.5.

What is new in this scenario is the behavior of the pedestrian. In order to model it, three continuous parameters are used that specify the distance of the start position on the side-



(a) (b) (c)

Figure 4.5: A visualization of the parameters needed to model scenario 2 as *logical scenario*. (a) the starting position, target position, target speed and mass of the AV (b) the starting position, two further waypoints, the target position, the target speed and the mass of the bicyclist (c) the starting position, the target speed, the mass and the two possible target positions of the pedestrian.

walk to the intersection, the mass of the pedestrian and the target speed of the pedestrian, as well as one binary parameter is used that decides where the pedestrian crosses the road. In contrast to the binary parameter in the first scenario, critical situations can occur with both value assignments. Again, it is particularly interesting how EAs are able to deal with this characteristic.

The parameter space $S_{2,ped}$ is subset of $\mathbb{R}^3 \times \mathbb{N}$. A visualization of the parameters that describe the behavior of the pedestrian is shown in figure 4.5 (c) and a detailed overview is shown in tabular A.6 .

Similar to the other actors, the pedestrian in CARLA also follows its waypoints in a straight line. However, especially for pedestrians a more versatile behavior would be reasonable. In real life, pedestrians often have different movement patterns, for example, if they carry luggage or push a baby stroller. Also the behavior of elderly or disabled people is not covered in the simulation. Therefore, attention must be paid if the results of the simulation are transferred to the real world.

Overall, the parameters space of scenario 2 is composed as $S_2 = S_{2,av} \times S_{2,bic} \times S_{2,ped} \subset \mathbb{R}^{15} \times \mathbb{N}_0$.

4.2.3 Scenario 3: Interacting with Several VRUs

The third scenario deals with the challenges an AV faces when interacting with several VRUs at the same time in urban traffic. The scenery used for this scenario is a straight street with a side street. The actors in the scenario are one AV, two bicyclists and two pedestrians. A visualization of the scenario is shown in figure 4.6.

In contrast to the first scenario, the AV starts in this scenario on the side street and turns left to the straight street. The four VRUs are each moving in opposite direction along the

straight street.

Criticality can arise in this scenario, if one or multiple VRUs reach the intersection at approximately the same time as the AV. Several different critical situations can occur in this scenario as well. These includes, for example, critical situation between the AV and only one of the VRUs or critical situations between the AV and a combination of up to all four participating VRUs, resulting in fifteen different combinations.
The way the actors are arranged in the critical situations can also be different. Critical situations in which the AV endangers VRUs with a side collision, as well as critical situations in which the AV endangers a bicyclist with a head-on or rear-end collision are possible.
Thus, it is again interesting in terms of research question 4 whether EAs are able to generate all kinds of critical situations.

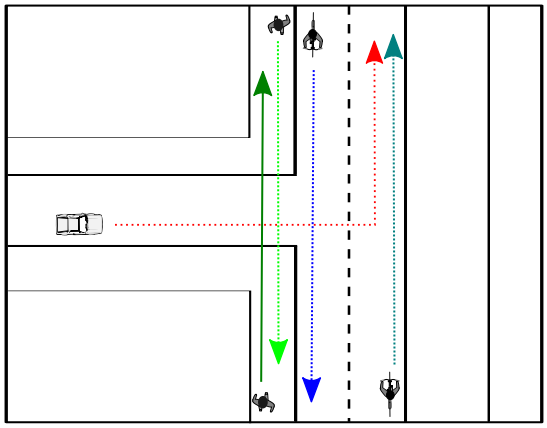
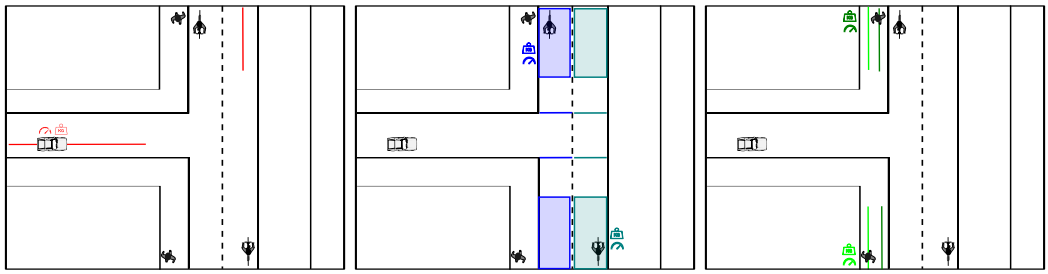


Figure 4.6: Abstract visualization of the scenario 3. The AV turns left on the straight street. Two bicyclists and two pedestrians move along the the straight street in each direction. Critical situations can occur when the AV and VRUs reach the intersection at the same time.

Similar to the parameters necessary for the implementation of the AV in scenario 1 and scenario 2, the needed parameters to implement the AV in scenario 3 are again its starting position, target position, target speed and mass. Hence, the parameter space $S_{3,av}$ is again subset of $\mathbb{R}^4$. A visualization of this parameter space is shown in figure 4.7 (a) and a detailed overview is given in the appendix in tabular A.7.



(a) (b) (c)

Figure 4.7: A visualization of the parameters needed to model scenario 3 as *logical scenario*. (a) the starting position, target position, target speed and mass of the AV (b) the starting position, two further waypoints, the target position, the target speed and the mass for each of the two bicyclists (c) the starting position, the target position, the target speed and the mass for each of the two pedestrians.

Also the parameters necessary for the implementation of a bicyclist in scenario 3 are the same as in the previous scenarios. But since two bicyclists participate in this scenario, these parameters are duplicated.
For each participating bicyclist an independent parameter space that is subset of $\mathbb{R}^8$ is added to the parameter space of the scenario. Thus, if scenarios with even more bicyclists are to be implemented, the parameter space increases by eight additional parameters per bicyclist.
A visualization of the parameters necessary to implement the bicyclists is shown in figure 4.7 (b) and a detailed overview is given in the appendix in tables A.8 and A.9.

In contrast to scenario 2, the pedestrians participating in scenario 3 do not have two possible target locations. Thus, the binary parameter that modeles this behavior is not necessary for scenario 3.
Instead, the parameters used for the implementation of the pedestrians are the starting position, the target position, the mass and the target speed. Similar to the implementation of the bicyclists, these parameters occur twice and independently of each other in the parameter space of scenario 3.
As this results in a parameter space for each pedestrian that is subset of $\mathbb{R}^4$, the parameter space of scenario 3 only consists of continuous parameters. Therefore, it is of particular interest to compare the performance of the EAs in this paramter space with their performance in the two previous mixed parameter spaces.
A visualization of the parameter space for the pedestrians is shown in figure 4.7. A detailed listing of the parameters can be seen in the appendix in tables A.10 and A.11.

Overall, the parameters space of scenario 3 is composed as $S_3 = S_{3,av} \times S_{3,bic1} \times S_{3,bic2} \times S_{3,ped1} \times S_{3,ped2} \subset \mathbb{R}^{28}$.

4.3 Criticality Metrics for Urban Traffic

In order to investigate research question 2, a criticality metric is necessary that calculates a good quantification of criticality in urban traffic scenarios with VRUs. Therefore, a new criticality metric is proposed in this section, called the *Predictive Conflict Index (PCI)*, which is based on the *Conflict Index (CI)* proposed by Alhajyaseen in [54]. This criticality metric is later also used as fitness function for the EAs that are applied for the investigation of research questions 3 and 4.

The derivation and the formula to calculate *PCI* is presented in subsection 4.3.1. An aggregation method that extends the *PCI* to a measure for *concrete scenarios* is given in subsection 4.3.2. Further, an approach to evaluate how well the *PCI* measures criticality in the three urban traffic scenarios designed in section 4.2 is presented in subsection 4.3.3.

4.3.1 Predictive Conflict Index (PCI)

A criticality metric that calculates a good quantification of criticality in urban traffic also must be able to measure criticality between AVs and VRUs. As described in subsection 2.1.2, both the severity and the likelihood of accidents are higher for VRUs in urban traffic than for other traffic participants. Hence, it is necessary to take both into account for the design of the proposed criticality metric.
Since many existing criticality metrics, such as *TTC*, *THW* or *PET*, fail to represent accident probability and severity at the same time, Alhajyaseen proposed the criticality metric *Conflict Index (CI)* in [54], a criticality metric that, according to him, considers crash occurrence probability as well as expected severity.

Originally the *CI* is proposed as a criticality metric that measures criticality between two actors a and b in a *concrete scenario* CS as follows:

$$\begin{array}{lcl} CI_{a,b} : \mathcal{CS} & \rightarrow & \mathbb{R} \\ CI_{a,b}(CS) & = & \frac{\alpha \cdot \Delta K_{a,b}(t_c)}{e^{\beta \cdot PET_{a,b}(CS)}} \end{array} \tag{4.1}$$

Where, in the nominator, $\Delta K_{a,b}(t_c)$ denotes the released kinetic energy in case of a collision between actors a and b at the time of the potential collision t_c, and the parameter $\alpha \in [0, 1]$ denotes the percentage how the released energy affects the people inside a vehicle. In the denominator, $PET_{a,b}(CS)$ denotes the output of the criticality metric *PET* in the *concrete scenario* CS and the parameter $\beta \in [0, 1]$ reflects the effect of the conflict type on crash probability [54].
This parameter β is chosen with the unit $\frac{1}{s}$ in order to make the denominator unitless. Hence, the unit of *CI* is joule and thus represents the kinetic energy released by a collison divided by the risk that the collision occurs [54].

The severity of the accident in *CI* is reflected by the released kinetic energy $\Delta K_{a,b}(t_c)$ in

case of a collision at time t_c. The released kinetic energy is a reasonable indication for the expected severity, as the mass and the velocity of the actors are the most influencial vehicle parameters on the severity of an accident [55, 56].
Further, an accident between an AV and a VRU can be considered as an inelastic collision. Hence, the released kinetic energy of a collision between two actors a and b at any time t can be calculated as follows:

$$\begin{array}{lcl} \Delta K_{a,b} : \mathbb{R} & \rightarrow & \mathbb{R} \\ \Delta K_{a,b}(t) & = & \frac{1}{2} \cdot \frac{m_a \cdot m_b}{m_a + m_b} \cdot (v_a(t) - v_b(t))^2 \end{array} \tag{4.2}$$

Where m_a and m_b denotes the masses of actors a and b and $v_a(t)$ and $v_b(t)$ denotes the velocities of the actors at time t.

The accident probability in CI is reflected by the *Post Encroachment Time* $PET_{a,b}(CS)$ in the *concrete scenario* CS, which calculates the time gap between the first actor leaving and the second actor entering a designated conflict area CA. Specifically, PET is defined as follows:

$$\begin{array}{lcl} PET_{a,b} : \mathcal{CS} & \rightarrow & \mathbb{R} \\ PET_{a,b}(CS) & = & t_{exit}(CA) - t_{entry}(CA) \end{array} \tag{4.3}$$

Where $t_{exit}(CA)$ denotes the time at which the first actor exited the designated conflict area, and $t_{entry}(CA)$ denotes the time at which the second actor reached the same area. However, for the prediction of the accident probability in urban traffic, PET has two inaccuracies.

The first inaccuracy is that PET only considers how close the actors have actually been to an accident, and does not investigate whether there was a high accident probability at any point in the *concrete scenario*. Due to its history in accident research, PET is designed as offline metric in which the two time points $t_{exit}(CA)$ and $t_{entry}(CA)$ are measured during the scenario execution. The outcome is then obtained after the *concrete scenario* is finished. In this way, only the trajectories the actors really took are considered. However, in order to analyze how criticality in urban traffic emerges it would be reasonable to investigate whether, during the *concrete scenario*, actors planned to take other trajectories that would have let to a higher criticality.

For this purpose, the *Predictive Encroachment Time (PrET)* can be used [16]. In contrast to PET, it is evaluated in every scene of the *concrete scenario*. Thereby, trajectory predictions of the two actors are used to determine when the trajectories of the actors intersect. In particular, $PrET$ calculates the criticality between two actors a and b in a scene S_t at time t as follows:

$$
\begin{aligned}
PrET_{a,b} : \mathcal{S} &\rightarrow \mathbb{R} \cup \{\infty\} \\
PrET_{a,b}(S_t) &= \begin{cases} |t_a - t_b| & \text{if } \exists t_a, t_b > 0 : Tr_a(t_a, t) = Tr_b(t_b, t), \\ \infty & \text{otherwise} \end{cases}
\end{aligned} \tag{4.4}
$$

Where $Tr_i(t_i, t)$ describes the point the actor i is predicted to be at time t_i, considering the trajectory prediction made at time point t.

The second inaccuracy of *PET* as measure for the accident probability in urban traffic is that only the time difference but not the duration of the way is considered in its calculation. However, this duration has a significant impact on the criticality in traffic scenes.
For example, a scene in which an AV and a pedestrian are expected to reach a conflict point at approximately the same time but that time is still several minutes away is not yet critical, as there still remains enough time for one of the actors to brake.
In contrast to this, a scene in which both actors reach the predicted conflict point in a few seconds would leave no time to brake and would therefore have a high criticality. Nevertheless, the outcome of *PrET* is the same for both scenes.

In order to solve this problem, another new criticality metric called *Duration-dependent Predictive Encroachment Time (DPrET)* is proposed in this work.
This criticality metric takes into account that in order to make a scene critical both the time difference between the actors reaching the predicted conflict point and the minimal time until the first actor reaches the conflict point must be low.
Therefore, in the calculation of *DPrET* the time difference between both actors reaching the predicted conflict point is multiplied with the minimal time that one actor needs to reach the conflict point. In this way, as soon as one of the factors increases also the output of *DPrET* gets larger, and thus the rating of the scene becomes less critical.
However, if one of the two factors is smaller than 1, even a large other factor would be reduced by it. In order to avoid this, the calculation of *DPrET* includes special cases for when at least one of the factors is smaller than 1.

With those special cases, *DPrET* is defined as follows:

$$
\begin{aligned}
DPrET_{a,b} : \mathcal{S} &\rightarrow \mathbb{R} \cup \{\infty\} \\
DPrET_{a,b}(S_t) &= \begin{cases}
\infty & \text{if } \neg\exists t_a, t_b > 0 : Tr_a(t_a, t) = Tr_b(t_b, t), \\
max(|t_a - t_b|, min(t_a, t_b)) \cdot 1s & \text{if } \exists t_a, t_b > 0 : Tr_a(t_a, t) = Tr_b(t_b, t) \\
& \wedge\ |t_a - t_b| < 1 \wedge min(t_a, t_b) < 1, \\
min(t_a, t_b) \cdot 1s & \text{if } \exists t_a, t_b > 0 : Tr_a(t_a, t) = Tr_b(t_b, t) \\
& \wedge\ |t_a - t_b| < 1 \wedge min(t_a, t_b) \geq 1, \\
|t_a - t_b| \cdot 1s & \text{if } \exists t_a, t_b > 0 : Tr_a(t_a, t) = Tr_b(t_b, t) \\
& \wedge\ |t_a - t_b| \geq 1 \wedge min(t_a, t_b) < 1, \\
|t_a - t_b| \cdot min(t_a, t_b) & \text{if } \exists t_a, t_b > 0 : Tr_a(t_a, t) = Tr_b(t_b, t) \\
& \wedge\ |t_a - t_b| \geq 1 \wedge min(t_a, t_b) \geq 1,
\end{cases}
\end{aligned} \tag{4.5}
$$

These special cases do not only avoid that a small factor keeps the whole product small, but at the same time ensure that the function is continuous over time as long as a predicted conflict point exists. Further, with a multiplication of $1s$ in the special cases, it is ensured that all cases have the unit s^2.

Note that in the case that one actor is already at the predicted conflict point, the calculation of *DPrET* is, apart of the unit, the same as the calculation of *THW*. Especially when *DPrET* is evaluated in a scene where two actors follow each other, the predicted conflict point would always be the position of the leading vehicle. Hence, the outcome of *DPrET* between the AV and the bicyclist in scenario 2 has the same magnitude as the outcome of *THW*.

This better to the urban traffic domain adapted version of the *PET* also makes it possible to improve the *CI*. Therefore, the *Predictive Conflict Index (PCI)* that considers *DPrET* instead of *PET* is proposed in this thesis. For two actors a and b in a scene S_t it can be calculated as follows:

$$
\begin{aligned}
PCI_{a,b} : \mathcal{S} &\rightarrow \mathbb{R} \\
PCI_{a,b}(S_t) &= \frac{\alpha \cdot \Delta K_{a,b}(t)}{e^{\beta \cdot DPrET_{a,b}(S_t)}}
\end{aligned} \tag{4.6}
$$

The unit of b is chosen $\frac{1}{s^2}$ in order to also have joule as unit of *PCI*. For the evaluation of criticality between the AVs and VRUs in the three proposed scenarios from section 4.2, the parameter α is set to $\alpha = 1$, as due to the lack of a protective outer shell the released kinetic energy of a collision is completely transferred to a VRU. Further, parameter β is also set to $\beta = 1$ for all scenarios, as the crash probability in urban traffic scenarios with VRUs is high.

4.3.2 PCI Aggregation for Concrete Scenarios

In order to evaluate criticality in urban traffic scenarios, the criticality metric *PCI* has to be extended from a metric that measures criticality between two actors in a traffic scene to a metric that measures criticality between all involved actors in a *concrete scenario*. Therefore, the criticality metric must first be aggregated over different actors at the same time and thereafter over the time span of the *concrete scenario*.

Since the focus of this work is on evaluating the criticality that arises when AVs interact with VRUs, and the *PCI* considers a scene involving two actors, each combination of the AV and one of the participating VRUs is evaluated separately. As a high output of *PCI* indicates high criticality, these separate measurements can be aggregated by sum. In this way an output is calculated that quantifies the criticality between all interesting actor combinations at a time point, and thus approximates the criticality between the AV and all VRUs in a scene.

In a next step, the criticality of all scenes in the *concrete scenario* is aggregated. For this purpose, several methods are possible.
One approach is to also use the sum to aggregate the scenes. However, this has the disadvantage that longer *concrete scenarios* also tend to have a higher outcome. Thus, when comparing the calculated criticality of *concrete scenarios*, it is difficult to distinguish whether a scenario is really more critical or just longer than others.
Another approach would be to take the average criticality of the scenes. But this approach has the disadvantage that several uncritical scenes would be more weighted than one critical scene. Especially for *concrete scenarios* that are non-critical for the most time it is difficult to determine whether they have been critical in at least one scene.

To avoid these disadvantages, the maximum of the criticality from the scenes is used as aggregation method. While this method has the disadvantage of not being able to distinguish whether a concrete scenario was critical in only one scene or at several points in time, it has the least effects compared to the other aggregation methods.
Especially, when comparing several *concrete scenarios*, the maximum provides a reliable information about which *concrete scenario* includes the most critical scene.

Hence, the calculation of *PCI* for a *concrete scenario* $CS \in \mathcal{CS}$ is defined as follows:

$$\begin{aligned} PCI : \mathcal{CS} &\rightarrow \mathbb{R} \\ PCI(CS) &= \max_{t \in \mathcal{T}_{\mathcal{CS}}} \Big(\sum_{v \in \mathcal{V}_{\mathcal{CS}}} PCI_{av,v}(S_t) \Big) \end{aligned} \tag{4.7}$$

Where $\mathcal{T}_{\mathcal{CS}}$ denotes the time span of the *concrete scenario*, $\mathcal{V}_{\mathcal{CS}}$ denotes a list of all participating VRUs and av denotes the AV.

4.3.3 Validation Approach for PCI

For the investigation of research question 2, it is necessary to validate whether the output of the *PCI* yields a good approximation of criticality in urban traffic scenarios. As described in subsection 2.2.2, good benchmarks for the validation of criticality metrics are their sensitivity and specificity. Hence, the sensitivity and specificity of the *PCI* is evaluated and compared to the sensitivity and specificity of other common criticality metrics, in particular, to a_{req}, $PrET$ and TTC.
Note that this validation approach has to be conducted independently for all three scenarios designed in section 4.2, as *PCI* could be well able to quantify criticality in one scenario but fail to do so in another.

The first step necessary to calculate the sensitivity and specificity of criticality metrics is to generate a ground truth of critical and non-critical scenarios.
Therefore, for each of the scenarios, 200 *concrete scenarios* are reviewed and classified from an expert's point of view. The reviewed *concrete scenarios* are either generated randomly or are sampled from a set of already known *concrete scenarios* with a high *PCI*. In this way, it is ensured that both, potential critical and potential non-critical scenarios, are among those reviewed.

In a second step, the criticality metrics are applied on the classified *concrete scenarios*. Here it must be considered that while for *PCI* and a_{req} a high output indicates a high criticality, the opposite is the case for $PrET$ and TTC. Hence, when $PrET$ and TTC are aggregated as metrics for *concrete scenarios*, the aggregation method has to be different than the one presented in 4.3.2.
In particular, the aggregation method that aggregates the criticality between different actors at the same time needs to be changed, as the sum is not a good aggregation method for metrics in which a minimal output indicates high criticality. An alternative aggregation method would be to calculate the exponential function of the additive inverse of the criticality measured between every actor pair. In this way, a high outcome again yields a high criticality and the criticality of the actors can be aggregated by sum.

Specifically, the three metrics are aggregated over a *concrete scenario* CS as follows:

$$\begin{aligned} a_{req} : \mathcal{CS} &\rightarrow \mathbb{R} \\ a_{req}(CS) &= \max_{t \in \mathcal{T}_{\mathcal{CI}}} \Big(\sum_{v \in \mathcal{V}_{\mathcal{CI}}} a_{req,av,v}(S_t) \Big) \end{aligned} \tag{4.8}$$

$$\begin{aligned} PrET : \mathcal{CS} &\rightarrow \mathbb{R} \\ PrET(CS) &= \max_{t \in \mathcal{T}_{\mathcal{CI}}} \Big(\sum_{v \in \mathcal{V}_{\mathcal{CI}}} e^{-PrET_{av,v}(S_t)} \Big) \end{aligned} \tag{4.9}$$

$$
\begin{aligned}
TTC : \mathcal{CS} &\rightarrow \mathbb{R} \\
TTC(CS) &= \max_{t \in \mathcal{T}_{\mathcal{CI}}} \Big(\sum_{v \in \mathcal{V}_{\mathcal{CI}}} e^{-TTC_{av,v}(S_t)} \Big)
\end{aligned} \tag{4.10}
$$

For every *concrete scenario* a continuous value is obtained as outcome of the metrics. In order to calculate the sensitivity and specificity, a threshold is necessary to determine which outcomes are classified as critical and which as non-critical. Thus, to evaluate the sensitivity and specificity of the metrics, the third step is to generate the *Receiver Operating Characteristic (ROC) curve* for each of the four metrics.

The ROC curve is a graphical representation that displays the relationship between sensitivity and specificity [57]. For calculating the ROC curve, the continuous output scale of a metric is divided into a discrete set. Each element of this set is then used as a threshold for calculating sensitivity and specificity. The results are plotted with sensitivity on the y-axis and 1 - specificity on the x-axis.
The generated ROC curve is a good basis to derive how well a measuring method is able to classify data [58], and thus how well a criticality metric is able to distinguish between critical and non-critical *concrete scenarios*. On the one hand, a ROC curve that rises steeply and is close to the top-left corner indicates that the metric is well able to discriminate between critical and non-critical scenarios. [59]. On the other hand, a ROC curve that is close to the bisector, in this context also called baseline, indicates that the metric is not better than random guessing.

A method to evaluate the ROC curve is to calculate the *area under the curve (AUC)*. As a result of AUC, a value between 0 and 1 is obtained, where 1 indicates that the metric classifies everything right and 0 indicates that the metric classifies everything wrong. If the ROC curve is equal to the baseline, the result of AUC is 0.5.
The optimal threshold from which on a *concrete scenario* counts as critical can also be derived from the ROC curve. Considering sensitivity and specificity with equal weight, the optimal threshold value is the point nearest to the top-left corner of the ROC curve, also known as best Youden's J, or Youden's index [59, 60]. This point can be calculated as follows:

$$
c^* = argmax_{c \in \mathcal{C}} \big(sens(c) + spec(c) - 1 \big) \tag{4.11}
$$

Where $\mathcal{C}$ denotes the discrete set of thresholds and $sens(c)$ and $spec(c)$ denote the sensitivity and specificity calculated with threshold c.

Overall, the ROC curve and AUC are good methods to evaluate the sensitivity and specificity of a criticality metric. Further, as a high sensitivity and a high specificity also indicates a high validity and reliability, this approach can be used to validate the output

of a criticality metric for traffic scenarios. Additionally, with Youden's index an optimal threshold c^* can be calculated to determine whether a *concrete scenario* is critical. This threshold can later be useful to evaluate the performance of EAs.

4.4 Implementation of the Evolutionary Algorithms

For the investigation of research questions 3 and 4, EAs are implemented. Those EAs are used to derive critical *concrete scenarios* from the *logical scenarios* designed in section 4.2. Therefore, the EAs use the criticality metric *PCI* proposed in section 4.3 as fitness function. While EAs optimize this fitness function, a set of *concrete scenarios* with a high criticality is generated. This set can then be analyzed to identify critical situations in urban traffic.

In particular, six different EAs are implemented and compared to each other. In subsection 4.4.1 the implementation of the two basic algorithm variants $(\mu + \lambda)$ and $(\mu - \lambda)$ is described. Subsection 4.4.2 describes how these two algorithms can be combined with Rechenberg's rule. Further, in subsection 4.4.3 it is described how the two basic variants can be extended with the principle of self adaptation. All six algorithms are implemented in PYTHON.

4.4.1 Basic $(\mu + \lambda)$ and $(\mu - \lambda)$Algorithms

Both, the $(\mu + \lambda)$ and the $(\mu - \lambda)$ algorithm, are implemented with a parent population size of $\mu = 10$ and a child population size of $\lambda = 50$. Each individual in the algorithms is a specific assignment of the parameters from the investigated *logical scenario*, and thus each individual represents a *concrete scenario*. In order to calculate the fitness of an individual, the *PCI* is calculated in this *concrete scenario*.

At the beginning of the algorithm, the parameter range of each parameter from the *logical scenario* is divided into 10 equal distributed values. In this process it is ensured that all of those values are valid parameter assignments for a *concrete scenario*. Hence, the discrete parameters are rounded to an integer and the binary parameters are rounded to 0 or 1.
In a next step, the equal distributed parameter assignments are randomly composed to 10 *concrete scenarios*, which are used as parents for the first generation. This approach ensures that the complete parameter space is covered in the first generation. A pseudocode of this process is shown in the appendix in figure B.1.

After the initial parent population is generated, a new child population is created from these 10 parents. As described in section 2.3.1, this happens with crossover between parents and additional random mutation.
For the crossover of a new individual, two parents are randomly selected. Hence, for these

algorithms holds $\kappa = 2$. For each of the parameters of the *logical scenario*, it is randomly decided from which parent the child inherits the value. Thus, after the crossover the child is a mixture of its two parents.

The way of mutation depends on the type of the parameters. For continuous parameters, a standard normal distributed number is drawn after the crossover. This random number is multiplied with the step size σ and added to the inherited value as mutation. Here, the step size can be any real number other than 0. The mutation for discrete parameters works similar, but the values are additionally rounded to an integer after the mutation.
For binary parameters, the step size σ is a value between 0 and 1. After a newly generated child has inherited a binary value of a parent, the mutation flips this parameter assignment with a probability of σ.
In addition, after the mutation all parameter values are checked whether they are valid parameter assignments for a *concrete scenario* and are adjusted to the allowed limits if not. A pseudocode of the crossover and mutation is shown in the appendix in figure B.2. An overview of the step sizes σ of the parameters is given in the appendix in the tables in section A.

In a last step, the *PCI* is evaluated for all created individuals and the individuals with the highest fitness are taken as parent population for the next generation. For the $(\mu + \lambda)$ algorithm, the parents as well as the children are evaluated and considered as future parents. A pseudocode of this algorithm is given in figure 4.8.
In contrast to this, the $(\mu - \lambda)$ algorithm only considers the newly generated child population as potential parents for the next generation. A pseudocode of this algorithm is given in figure 4.9.

Both algorithms are repeated for 20 generations. During this process, each generated *concrete scenario* and the corresponding evaluated *PCI* is saved. Thus, at the end of the algorithm a set of *concrete scenarios* and the occurring criticality is obtained.

```python
mu = 10
lam = 50
kappa = 2
generations = 20

# create equal distributed parent population
parent_population = generate_initial_parent_population(mu)

for i in range(generations):

    # create new child population
    child_population = crossover_and_mutate_child_population(
            parent_population, lam, kappa)

    # calculate the PCI of every individual of the child population
    # and sort the population according to the result
    evaluated_population = sorted(child_population + parent_population,
            key=lambda individual: PCI(individual), reverse=True)

    # use the best children as new parent population
    parent_population = evaluated_population[:mu]
```

Figure 4.8: PYTHON code of the $(\mu + \lambda)$ algorithm.

```python
mu = 10
lam = 50
kappa = 2
generations = 20

# create equal distributed parent population
parent_population = generate_initial_parent_population(mu)

for i in range(generations):

    # create new child population
    child_population = crossover_and_mutate_child_population(
            parent_population, lam, kappa)

    # calculate the PCI of every individual of the child population
    # and sort the population according to the result
    evaluated_child_population = sorted(child_population,
            key=lambda individual: PCI(individual) , reverse=True)

    # use the best children as new parent population
    parent_population = evaluated_child_population[:mu]
```

Figure 4.9: PYTHON code of the $(\mu - \lambda)$ algorithm.

4.4.2 $(\mu + \lambda)$ and $(\mu - \lambda)$ Algorithms with Rechenberg's Rule

Similar to the implementation of the two basic variants, the $(\mu+\lambda)$ algorithm in combination with Rechenberg's rule and the $(\mu - \lambda)$ algorithm in combination with Rechenberg's rule are implemented with a parent population size of $\mu = 10$ and child population size of $\lambda = 50$.
Also the way how the initial parent population is generated and the crossover and mutation works is the same as for the basic variants.

The difference of the two algorithms combined with Rechenberg is that after all individuals of a generation are evaluated, it is assessed whether the generation has produced better results than the previous one. This is done by adding the fitness of the individuals of the new parent population together. If the summed up fitness is higher than the previous one, the algorithm improves and the step size σ of each parameter is increased as follows:

$$\sigma \mapsto \sigma \cdot \exp(4/5)^{1/d} \tag{4.12}$$

Where $d = \sqrt{N+1}$ and N is the number of parameters in the *logical scenario*.
On the other side, if the summed up fitness is lower or equal than the previous one, the algorithm did not improve and the step size σ is decreased as follows:

$$\sigma \mapsto \sigma \cdot \exp(-1/5)^{1/d} \tag{4.13}$$

With these two formulas, the step sizes increases if the success rate of the algorithm is in the long run higher than 1/5 and decreases if it is lower than 1/5. The equations for the mutation are taken from the lecture cited in [61].

The adjusted step sizes are used for the mutation in the next generation. The pseudocode of the $(\mu + \lambda)$ algorithm with Rechenberg is shown in figure 4.10. The pseudocode of the $(\mu - \lambda)$ algorithm with Rechenberg is shown in figure 4.11.
Again, the algorithms are repeated for 20 generations and as a result a set of *concrete scenario* together with the corresponding *PCI* is obtained.

```python
mu = 10
lam = 50
kappa = 2
generations = 20
d = math.sqrt(len(logical_scenario_parameters) + 1)

# create equal distributed parent population
parent_population = generate_initial_parent_population(mu)

# calculate summed up fitness for first parent population
summed_up_fitness = sum([PCI(parent) for parent in parent_population])

for i in range(generations):

    # create new child population
    child_population = crossover_and_mutate_child_population(
            parent_population, lam, kappa)

    # calculate the PCI of every individual of the child population
    # and sort the population according to the result
    evaluated_population = sorted(child_population + parent_population,
            key=lambda individual: PCI(individual), reverse=True)

    # use the best children as new parent population
    parent_population = evaluated_population[:mu]

    # evaluate new summed up fitness
    new_summed_up_fitness = sum([PCI(parent) for parent in parent_population
        ])

    # if algorithm has improved, increase step size
    if new_summed_up_fitness > summed_up_fitness:

        for parameter in logical_scenario_parameters:
            sigma[parameter] *= math.exp(4/5) ** (1/d)

            if key in binary_parameters:
                sigma[parameter] = min(1, max(0, sigma[key]))

    # if algorithm has not improved, decrease step size
    else:

        for parameter in logical_scenario_parameters:
            sigma[parameter] *= math.exp(-1/5) ** (1/d)

            if parameter in binary_parameters:
                sigma[parameter] = min(1, max(0, sigma[parameter]))

    # save new summed up fitness
    summed_up_fitness = new_summed_up_fitness
```

Figure 4.10: PYTHON code of the $(\mu + \lambda)$ algorithm with Rechenberg.

```python
mu = 10
lam = 50
kappa = 2
generations = 20
d = math.sqrt(len(logical_scenario_parameters) + 1)

# create equal distributed parent population
parent_population = generate_initial_parent_population(mu)

# calculate summed up fitness for first parent population
summed_up_fitness = sum([PCI(parent) for parent in parent_population])

for i in range(generations):

    # create new child population
    child_population = crossover_and_mutate_child_population(
            parent_population, lam, kappa)

    # calculate the PCI of every individual of the child population
    # and sort the population according to the result
    evaluated_population = sorted(child_population,
            key=lambda individual: PCI(individual), reverse=True)

    # use the best children as new parent population
    parent_population = evaluated_population[:mu]

    # evaluate new summed up fitness
    new_summed_up_fitness = sum([PCI(parent) for parent in parent_population
        ])

    # if algorithm has improved, increase step size
    if new_summed_up_fitness > summed_up_fitness:

        for parameter in logical_scenario_parameters:
            sigma[parameter] *= math.exp(4/5) ** (1/d)

            if key in binary_parameters:
                sigma[parameter] = min(1, max(0, sigma[key]))

    # if algorithm has not improved, decrease step size
    else:

        for parameter in logical_scenario_parameters:
            sigma[parameter] *= math.exp(-1/5) ** (1/d)

            if parameter in binary_parameters:
                sigma[parameter] = min(1, max(0, sigma[parameter]))

    # save new summed up fitness
    summed_up_fitness = new_summed_up_fitness
```

Figure 4.11: PYTHON code of the $(\mu - \lambda)$ algorithm with Rechenberg.

4.4.3 $(\mu + \lambda)$ and $(\mu - \lambda)$ Algorithms with Self Adaptation

The two basic algorithm variants in combination with self adaptation are also implemented with a parent population size of $\mu = 10$ and a child population size of $\lambda = 50$. Further, the way how the first parent population is generated and the crossover between parents works in the same as in the previous introduced algorithms.

However, for these two algorithms the mutation of new individuals works different. While the basic EA variants and the variants in combination with Rechenberg use the same step sizes for the creation of the entire child population, the algorithm variants with self adaptation use a different step size σ_k for every individual. Therefore, before every mutation a new σ_k is calculated as follows:

$$\sigma_k = \sigma \cdot \exp(\tau \cdot \mathcal{N}(0, 1)) \tag{4.14}$$

Where σ is the parent step size of the generation, $\tau = \frac{1}{\sqrt{N}}$ and $\mathcal{N}(0, 1)$ is a standard normal distributed random number. Again, the hyper parameters for this mutation are taken from the lecture cited in [61].
The pseudocode for the crossover and mutation in the self adaptation algorithms is shown in the appendix in figure B.3.

After a generation, not only the best individuals are taken as parents for the next generation but additionally the step sizes σ_k that were used to mutate the best individuals are used to calculate a new parent step size σ. This is done by calculating the average of all σ_k from the new parents.

Again, these algorithms are repeated for 20 generations. The pseudocode of the $(\mu + \lambda)$ variant with self adaptation is shown in figure 4.12. The pseudocode of the $(\mu - \lambda)$ variant with self adaptation is shown in figure 4.13.

```python
mu = 10
lam = 50
kappa = 2
generations = 20

# create equal distributed parent population
parent_population = generate_initial_parent_population(mu)

for i in range(generations):

    # create new child population
    child_population = crossover_and_mutate_child_population(
            parent_population[0], sigma, lam, kappa)

    # calculate the PCI of every individual of the child population
    # and sort the population according to the result
    evaluated_population = sorted(child_population + parent_population,
            key=lambda individual: PCI(individual[0]), reverse=True)

    # use the best children as new parent population
    parent_population = evaluated_population[:mu]

    # calculate new mean sigma
    sigma = calculate_mean_sigma(parent_population)
```

Figure 4.12: PYTHON code of the $(\mu + \lambda)$ algorithm with self adaptation.

```python
mu = 10
lam = 50
kappa = 2
generations = 20

# create equal distributed parent population
parent_population = generate_initial_parent_population(mu)

for i in range(generations):

    # create new child population
    child_population = crossover_and_mutate_child_population(
            parent_population[0], sigma, lam, kappa)

    # calculate the PCI of every individual of the child population
    # and sort the population according to the result
    evaluated_population = sorted(child_population,
            key=lambda individual: PCI(individual[0]), reverse=True)

    # use the best children as new parent population
    parent_population = evaluated_population[:mu]

    # calculate new mean sigma
    sigma = calculate_mean_sigma(parent_population)
```

Figure 4.13: PYTHON code of the $(\mu - \lambda)$ algorithm with self adaptation.

5 Evaluation

This chapter presents the results of the investigation of the research questions. In section 5.1 the validation of the proposed criticality metric *PCI* is described. Afterwards, in the three sections 5.2, 5.3 and 5.4, the performance of EAs in each of the three designed urban traffic scenarios is evaluated. On the one hand, in doing so, it is examined whether EAs are able to generate critical *concrete scenarios*. On the other hand, it is investigated whether EAs are also able to identify critical situations in the urban traffic domain.

5.1 Evaluation of Criticality Metrics

A new criticality metric called *PCI* is proposed in section 4.3. For the evaluation of research question 2 and in order to investigate whether *PCI* is a suitable fitness function for EAs, it is necessary to validate how well *PCI* calculates an approximation of criticality in urban traffic scenarios. In order to do so, the sensitivity and specificity of *PCI* is evaluated with the help of the ROC curve. Further, the results are compared to the sensitivity and specificity of a_{req}, $PrET$ and TTC.

This evaluation is conducted independently for each of the three in section 4.2 presented urban traffic scenarios. The results of the validation of *PCI* in scenario 1 are presented in subsection 5.1.1, the results of its validation for scenario 2 are presented in subsection 5.1.2 and the results of the validation for scenario 3 are presented in subsection 5.1.3.

5.1.1 Validation of PCI in Scenario 1

In scenario 1 an AV drives towards an intersection where a bicyclist is turning onto the road. Thereby, the bicyclist can be occluded by parking vehicles, causing that the AV detects it too late for a regular braking maneuver. A more detailed description of the scenario is given in the methodology in subsection 4.2.1.
In order to compute a good approximation of criticality, it is important that a criticality metric identifies scenes as critical in which the AV has a high speed and a low distance to the bicyclist. For the validation of *PCI*, the metric is evaluated in 200 *concrete scenarios* that were classified in advance as critical or non-critical. With the obtained data a ROC curve is created.

A visualization of this ROC is shown in figure 5.1. For the comparison, the ROC curves calculated with a_{req}, $PrET$ and TTC are shown in figure 5.2. A listing of the AUC scores and the best threshold c^* calculated by Youden's index together with the corresponding

sensitivity and specificity of the four metrics is given in the appendix in table C.1.

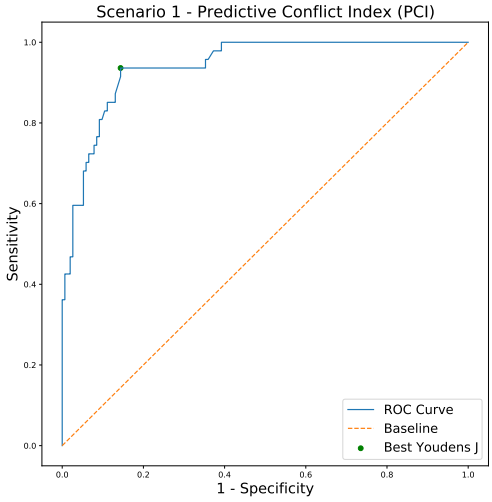
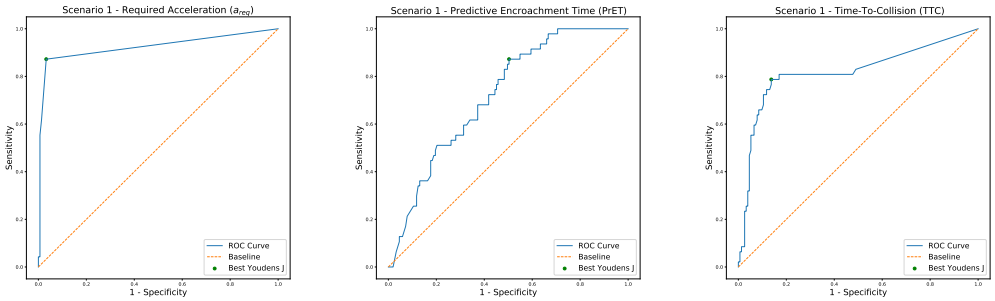


Figure 5.1: ROC curve of the *PCI* in scenario 1. The best Youdens J is found at a threshold of 247.79 joule with a sensitivity of 0.94 and a specificity of 0.86.



(a) (b) (c)

Figure 5.2: ROC curves of other criticality metrics in scenario 1. (a) a_{req} with a sensitivity of 0.87 and a specificity of 0.97 (b) *PrET* with a sensitivity of 0.87 and a specificity of 0.49 (c) *TTC* with a sensitivity of 0.78 and a specificity of 0.86.

The ROC curve of the *PCI* has a steep ascent and the AUC score of 0.94 indicates that *PCI* correctly classifies *concrete scenarios* in most cases. Furthermore, this AUC score is the highest score among the evaluated metrics. Accordingly, the best identified threshold of 247.79 joule has a high sensitivity of 0.94 and a high specificity of 0.86.

Good results are also obtained with the criticality metric a_{req}. Since criticality arises in this scenario when the AV does not detect the bicyclist in time and has to perform an emergency braking maneuver, the good performance of a_{req} is not surprising. Further, the fact that the results of *PCI* are similar to those of a_{req} indicates that the newly proposed

metric is well able to identify those kinds of scenes and correctly classifies them as critical.

In contrast to this, the worst obtained results are those of *PrET*. A reason for this could be that, as described in subsection 4.3.1, *PrET* rates scenes as critical in which the AV and the bicyclist are predicted to reach the intersection at approximately the same time, no matter how far this time is in the future. Since the calculation of *PrET* does not consider whether there is still enough time for a safe braking maneuver, some *concrete scenarios* might be falsely classified as critical. Hence, the bad performance of *PrET* indicates that it is reasonable to use *DPrET* in the calculation of *PCI*.

Overall, the results of the ROC curves indicate that *PCI* is well able to measure criticality in scenario 1. From this follows that *PCI* is also a well suited fitness function for the EAs used to investigate research question 3 and 4.

5.1.2 Validation of PCI in Scenario 2

In the second scenario, which is described in detail in the methodology in subsection 4.2.2, an AV makes a left turn in an acute angle while following a bicyclist. At the same time a pedestrian crosses the road at different possible locations.
Criticality can arise in this scenario either when the AV drives to close to the bicyclist or when the AV wrongly predicts the behavior of the pedestrian. Hence, it is important that the *PCI* correctly identifies both kinds of scenes as critical. Again, the four metrics are evaluated in 200 previously classified *concrete scenarios* and a ROC curve is created for each of them.

A visualization of the ROC curve calculated with *PCI* is shown in figure 5.3. The ROC curves calculated with a_{req}, *PrET* and *TTC* are shown in figure 5.4. An overview of the AUC score and the best threshold c^* calculated by Youden's index together with the corresponding sensitivity and specificity of the four metrics is given in the appendix in table C.2.

The ROC curve of the *PCI* has an AUC score of 0.84, which is slightly worse than the value in scenario 1, but still indicates that the metric is well able to distinguish between critical and noncritical *concrete scenarios*. Further, with a best threshold c^* of 872.24 joule it has a sensitivity of 0.73 and the specificity of 0.89.
Compared to the first scenario, the best threshold c^* is much higher in scenario 2. A reason for this could be the different number of involved actors as well as the fact that with rear-end collisions a different accident type occurs in this scenario.

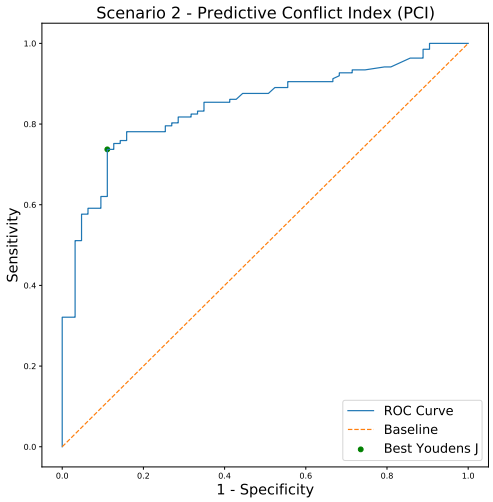
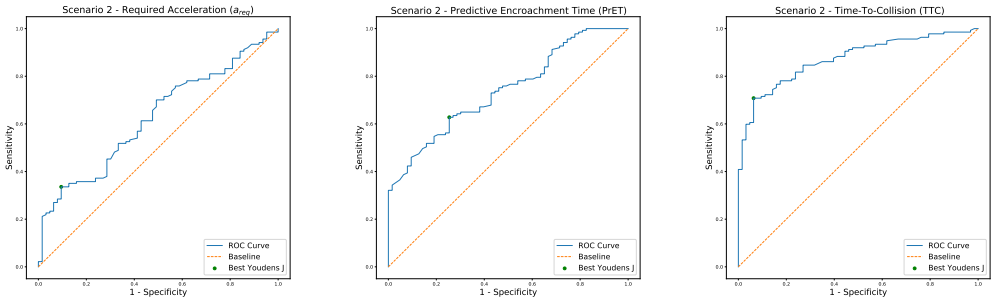


Figure 5.3: ROC curve of the *PCI* in scenario 2. The best Youdens J is found at a threshold of 872.24 joule with a sensitivity of 0.74 and a specificity of 0.89.



(a) (b) (c)

Figure 5.4: ROC curves of other criticality metrics in scenario 1. (a) a_{req} with a sensitivity of 0.34 and a specificity of 0.90 (b) *PrET* with a sensitivity of 0.63 and a specificity of 0.75 (c) *TTC* with a sensitivity of 0.71 and a specificity of 0.94.

Similar to the first scenario, the AUC score of *PCI* is among the best of the compared criticality metrics. But this time the AUC score of *TTC* is with 0.87 even slightly better. Since *TTC* is a well suited metric to measure criticality in scenarios where actors follow each other, this good result is again plausible. Further, the similarity of the results of *TTC* and *PCI* indicates that the newly proposed metric is well able to identify critical situations in which an AV drives behind a bicyclist.

However, these results do not give information about how well the *PCI* is able to measure the criticality between the AV and the crossing pedestrian. In order to investigate this more closely, the *concrete scenarios* generated with EAs that use *PCI* as fitness function are analyzed in subsection 5.3.2.

5.1.3 Validation of PCI in Scenario 3

In scenario 3 an AV drives towards a street with two bicyclists and two pedestrians. When turning on this street, the AV must keep enough distance to all four VRUs. A more detailed description of the scenario is given in the methodology in subsection 4.2.3.
Criticality can arise in this scenario in many different scenes and with different combinations of involved actors. It is therefore particularly challenging for the metric to recognize criticality in versatile ways.

Figure 5.5 shows a visualization of the ROC curve created with the data that is obtained when *PCI* is applied on the 200 classified *concrete scenarios*. This ROC curve is compared to the ROC curves calculated with a_{req}, $PrET$ and TTC, which are shown in figure 5.6.

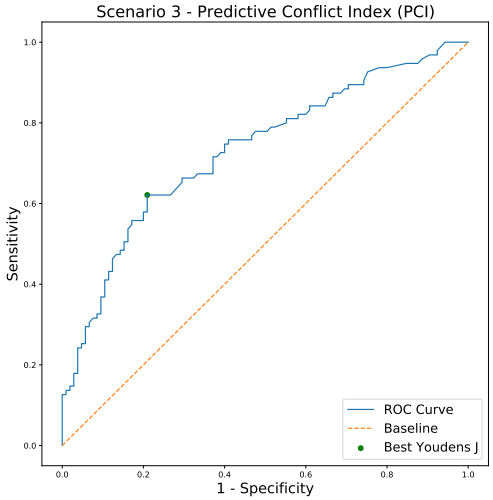
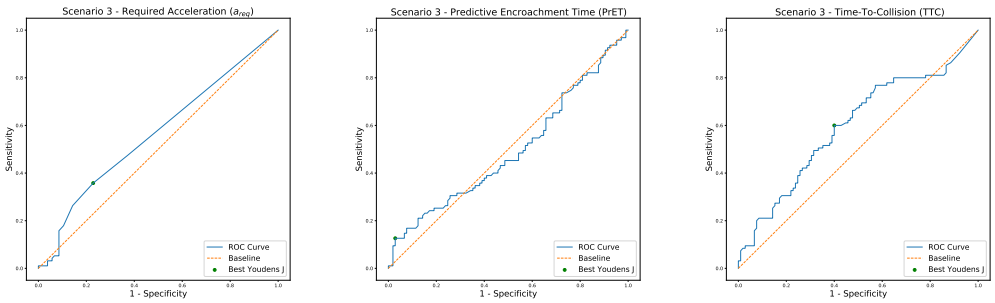


Figure 5.5: ROC curve of the *PCI* in scenario 1. The best Youdens J is found at a threshold of 703 joule with a sensitivity of 0.62 and a specificity of 0.79.



(a) (b) (c)

Figure 5.6: Roc curves of other criticality metrics in scenario 1. (a) a_{req} with a sensitivity of 0.35 and a specificity of 0.77 (b) $PrET$ with a sensitivity of 0.13 and a specificity of 0.97 (c) TTC with a sensitivity of 0.6 and a specificity of 0.6.

An overview of the AUC and the best threshold c^* calculated by Youden's index together with the corresponding sensitivity and specificity of the four metrics is given in the appendix in table C.3.

Already during the classification it became noticeable that many *concrete scenarios* were difficult to classify as critical or non-critical. Compared to scenario 1 and 2, much more *concrete scenarios* were somewhere between critical and non-critical.
Hence, when evaluating the results it is not surprising that they are worse than those of the previously investigated scenarios.

However, the results of the newly proposed criticality metrics *PCI* are better than those of the three common criticality metrics.
While a_{req}, *PrET* and *TTC* are not performing significantly better than random guessing, the ROC curve of *PCI* is clearly above the baseline. The AUC score of 0.73 indicates that *PCI* classifies the *concrete scenarios* correctly in almost three quarters of the cases. Although this still leaves space for improvement, with a best sensitivity of 0.62 and a specificity of 0.79 at the threshold of 703 joule, the metric *PCI* is much better than all of the other metrics.

Therefore, it is again reasonable to use *PCI* as fitness function for the EAs. Nevertheless, it should be investigated in future work how criticality can be better measured in scenarios that have a similar design as scenario 3.

5.2 Analysis of EAs in Scenario 1

As shown in the previous section, *PCI* is well able to quantify criticality in the first scenario. Hence, when *PCI* is used as fitness function for EAs, its output is maximized, and thus *concrete scenarios* with a high criticality are generated.
In this section it is analyzed how the generated *concrete scenarios* can be used to identify critical situations in urban traffic. Furthermore, it is examined whether all kinds of critical situations that can occur in scenario 1 also occur in the *concrete scenarios* generated by the EAs.

Therefore, in a first step, the performance of the different implemented EAs is evaluated in subsection 5.2.1. In a second step, the evolution of the parameters over the generations is analyzed in subsection 5.2.2. This analysis provides information about which parameter assignments lead to critical *concrete scenarios* and at the same time investigates whether the complete parameter space was searched during the optimization process. In a last step, the generated critical *concrete scenarios* are analyzed in subsection 5.2.3 in order to identify different kinds of critical situations that can occur in scenario 1.

5.2.1 Algorithm Performance

The six different EAs described in section 4.4 are applied on scenario 1. In order to evaluate how well these algorithms perform four different characteristics are investigated.

The first characteristic is the number of critical *concrete scenarios* generated in each of the 20 generations. The 247.79 joule identified by Youden's Index in subsection 5.1.1 is used as threshold from which *PCI* output counts a *concrete scenario* as critical.
The results of this comparision are shown in figure 5.7. On the one hand, the number of critical *concrete scenarios* generated by the basic variants of $(\mu + \lambda)$ and $(\mu - \lambda)$ as well as these two variants combined with self adaptation improves of the 20 generations. The steepest increase happens in the first seven generations. Thereafter, the four algorithms generate in every generation approximately 20-30 critical *concrete scenarios*.
On the other hand, the algorithm variants in combination with Rechenberg are only hardly able to generate any critical *concrete scenarios* and even become worse after the first seven generations.

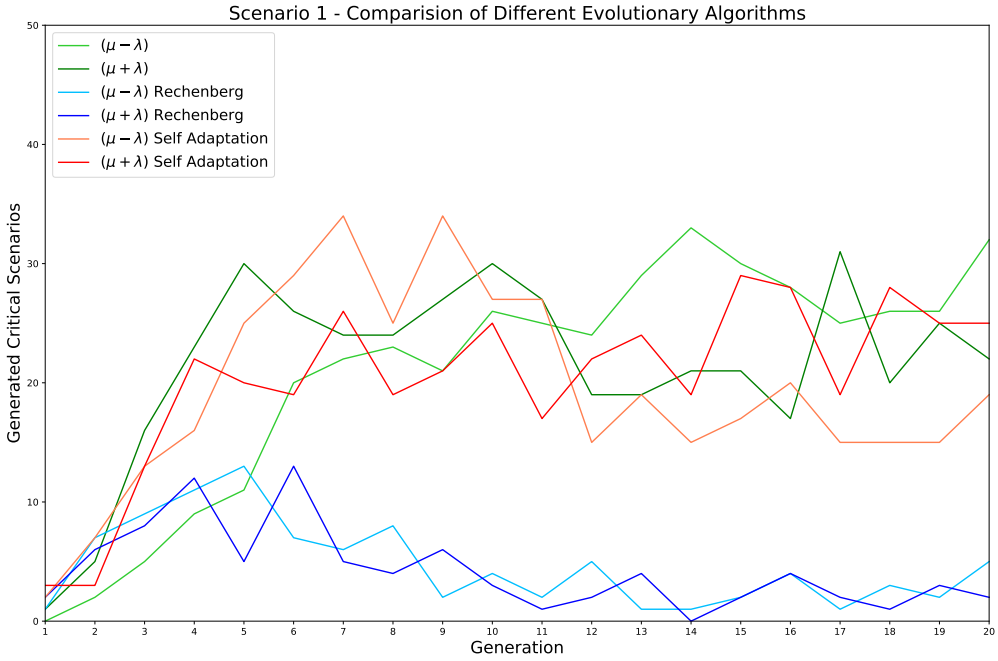


Figure 5.7: Comparison of the number of critical scenarios generated by the different algorithms over 20 generations in scenario 1. For all algorithms holds $\mu = 10$ and $\lambda = 50$.

The second investigated characteristic is the average fitness of the 10 individuals in the each generation's parent population. A visualization is shown in the appendix in figure D.1.
In this comparison it is seen that the average parent fitness of the $(\mu + \lambda)$ version is higher than the one of the $(\mu - \lambda)$ version for all three algorithm variants. Due to the fact that the $(\mu + \lambda)$ variants keep good parents over generations while the $(\mu - \lambda)$ variants only

consider new generated children as possible next parents, this observation is not surprising. However, while the steepest increase of the $(\mu + \lambda)$ versions happens in the first generations, the basic $(\mu - \lambda)$ variant still increases until the very end. Here it could be possible that if the algorithm runs for more generations the basic $(\mu - \lambda)$ variant could even be better than the $(\mu + \lambda)$ versions.
Further, similar to the comparison of the number of generated critical *concrete scenarios*, it can be noticed that the Rechenberg variant performs worse than the other algorithms.

The third compared characteristic is the average fitness of the children generated in every of the 20 generations. Again, a visualization is shown in the appendix in figure D.2.
Similar to the previous comparisons, the Rechenberg variants perform also in this characteristics worse than the other algorithms. But also the average child fitness of the $(\mu - \lambda)$ algorithm combined with self adaptation decreases after some generations.
The other three variants achieve similar good results over the 20 generations. However, this time the basic $(\mu - \lambda)$ variant has, despite a slower start, the highest average children fitness in the last generation.

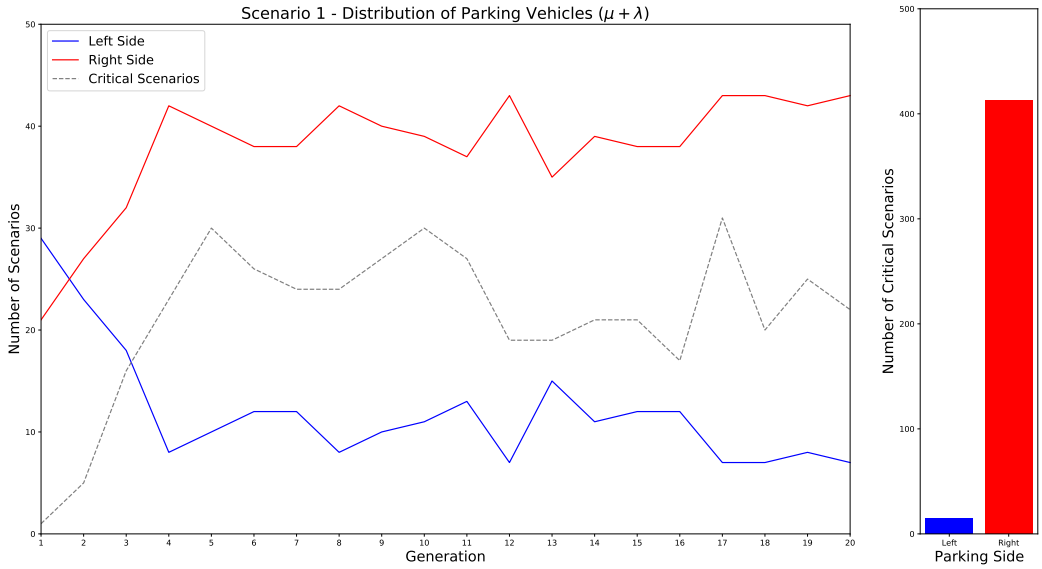
The last investigated characteristic is the highest criticality of a *concrete scenario* generated in every generation. A visualization of this characteristic is shown in the appendix in figure D.3.
Similar to the previous compared characteristics, the two basic variants of $(\mu + \lambda)$ and $(\mu - \lambda)$ as well as $(\mu + \lambda)$ combined with self adaptation perform the best.
Since they perform good for all four compared characteristics, the focus in the following two sections will be to analyze these three algorithms more closely.

5.2.2 Parameter Analysis

The first parameter that is analyzed more closely is the binary parameter that determines on which side of the road the parking vehicles are placed. An exemplary evaluation of this parameter assignment in the basic $(\mu + \lambda)$ algorithm is shown in figure 5.8.
While this binary parameter is chosen randomly in the first generation, all three of the good performing algorithms learn really fast that the parking vehicles should rather be placed on the right side. This learning process leads to good results, as in almost all found critical *concrete scenarios* the parking vehicles are on the right side.

In contrast to this, a closer look at the binary parameter also gives an answer why the Rechenberg variants fail to generate critical *concrete scenarios*. Despite that the learning process starts to learn in the opposite direction, after a few generations the algorithm places the parking cars on the left side in the majority of the generated *concrete scenarios*, as shown in figure 5.9.
A reason for this could be that, given the fact that all step sizes increase after the algo-

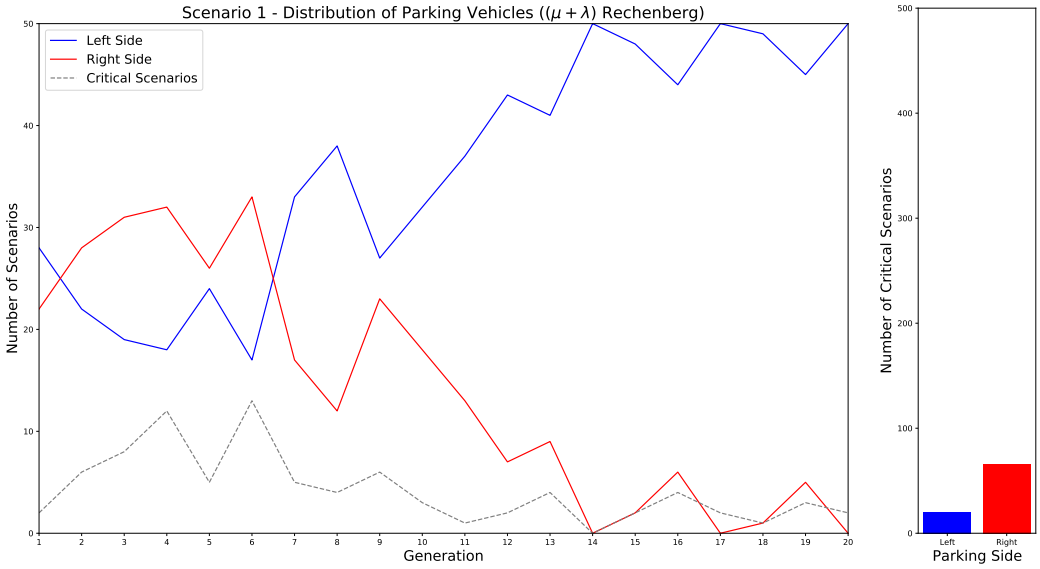
rithm improves, also the step size increases that decides with which probability the binary parameter is flipped. In this way, even though all parents have the parking cars on the right side, the parking cars of the children are all placed on left side. Despite the fact that more *concrete scenarios* with parking cars on the left side are evaluated, more critical *concrete scenarios* with parking cars on the right side are found.



(a) (b)

Figure 5.8: (a) Evolution of the side of the parking vehicles over the generations of the $(\mu + \lambda)$ algorithm (b) Distribution of the side of the parking vehicles in the generated critical scenarios.

In a next step, the evolution of the non-binary parameters is investigated. In order to understand how the algorithms learn to assign those parameters, box plots are created.
Box plots are a common graphical representation that expresses statistical characteristics of data and is well able to summarize the distribution of a dataset [62]. For every step on the x-axis, a graphical representation of the minimum, the first quartile, the median, the third quartile and the maximum of the data set is given. In particular, the minimum and the maximum are represented by two antennas, the first and the third quartile are represented by a box and the median is represented by a line in the box. In addition, outliers can be represented by dots in the box plot.
By displaying the assignment of a parameter over the 20 generations, box plots can visualize the learning process of EAs.



(a) (b)

Figure 5.9: (a) Evolution of the side of the parking vehicles over the generations of the $(\mu + \lambda)$ algorithm with Rechenberg (b) Distribution of the side of the parking vehicles in the generated critical scenarios.

The first two analyzed non-binary parameters are the distance of the first parking vehicle to the intersection and the number of parking vehicles.
The evolution of the distance to the intersection is shown in figure 5.10, again exemplary for the $(\mu + \lambda)$ algorithm as all three investigated algorithms behave similar. While the complete parameter range is searched in the first generation, the algorithms learn quite fast to set the distance as low as possible to generate critical *concrete scenarios*. Thus, the algorithms identify a small distance of parking vehicles to an intersection as criticality phenomena in urban traffic.
In contrast to that, the number of parking vehicles does not seem to be relevant. As shown in figure D.4 in the appendix, the algorithms only learn that the number should be at least one but do not further decide for an exact number.

The next two parameters that are particularly interesting to investigate in more detail are the target speed and mass of the AV.
The evolution of the target speed of the AV is shown in figure 5.11. Similar to the distance of the first parking vehicle, the algorithm starts to search the complete parameter range in the first generation and thereafter learns to maximize the target speed. For both of the parameters, this learning process happens within the first seven generations, which are also the generations with the steepest increase of the compared characteristics in chapter 5.2.1.
In contrast to the target speed, the algorithms do not learn to maximize the mass of the AV. As shown in figure D.5 and D.6 in the appendix, the algorithms learn to choose a parameter assignment in a certain range, however this range is different for different in-

vestigated algorithms. This learning behavior indicates that the mass of the AV has no big influence on the criticality in scenario 1.

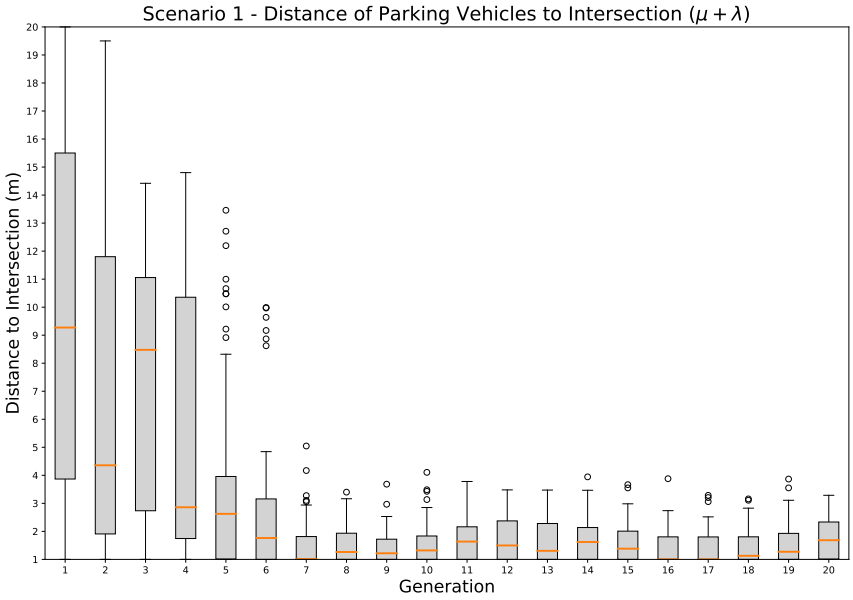


Figure 5.10: Box plot that visualizes the evolution of the distance from the first parking vehicle to the intersection over 20 generations of the $(\mu + \lambda)$ algorithm in scenario 1.

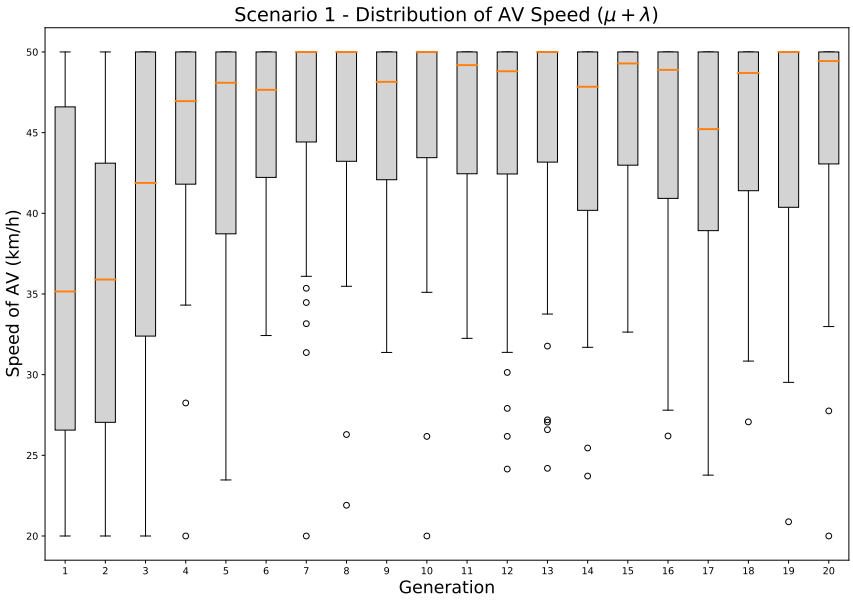


Figure 5.11: Box plot that visualizes the evolution of the speed of the AV over 20 generations of the $(\mu + \lambda)$ algorithm in scenario 1.

In figure 5.12, the correlation between the distance of the first parking vehicles to the intersection, the target speed of the AV, and the criticality measured in the generated *concrete scenarios* is shown. Here it can be seen that critical *concrete scenarios* are found when the obstacle distance is minimal and the target speed is maximal.
Again, this visualization is exemplary for the basic $(\mu + \lambda)$ algorithm as the observed correlation is similar for all of the three investigated algorithms. Hence, the algorithms learn the right properties to generate critical *concrete scenarios* in scenario 1 and thus are all well able to answer research question 3.

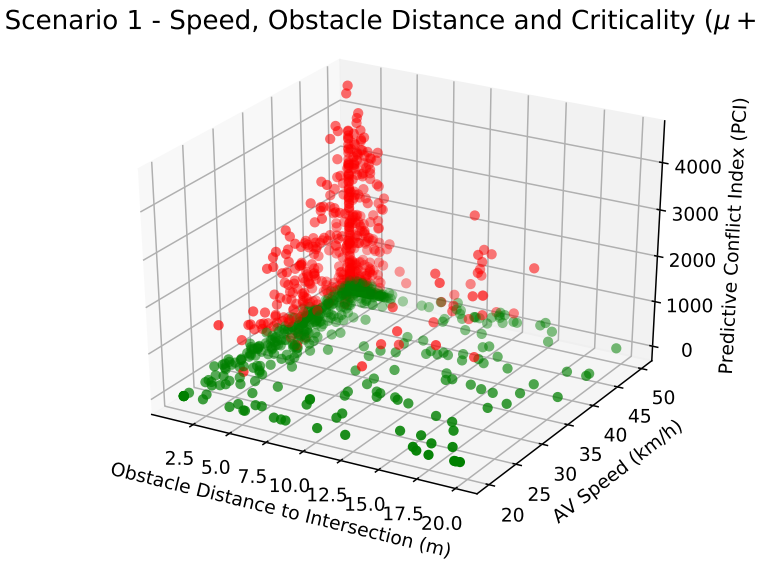


Figure 5.12: The correlation of the AV's speed, the distance from the first parking vehicle, and the measured criticality in the generated *concrete scenarios* with the $(\mu + \lambda)$ algorithm in scenario 1.

5.2.3 Analysis of Critical Situations

In a next step it is examined whether the learning approach of the EAs also finds the whole range of critical situations that can occur in scenario 1, and thus also answers research question 4. In order to investigate this, the generated *concrete scenarios* are analyzed more closely to identify different kinds of critical situations.

To this end, from each generated critical *concrete scenario* the scene with the highest criticality is investigated. In particular, the positions and speed of the AV and the bicyclist in this scene are clustered with DBSCAN with the aim to identify critical situations.
DBSCAN is a density-based clustering approach that groups points with similar characteristics [63]. For the implementation of DBSCAN, the version of the PYTHON package

`scikit-learn`[1] is used.

The results of this clustering for the basic $(\mu + \lambda)$ algorithm are visualized in figure 5.13. The visualization of the results for the $(\mu - \lambda)$ algorithm and $(\mu + \lambda)$ with self adaptation can be seen in the appendix in figures D.7 and D.8.

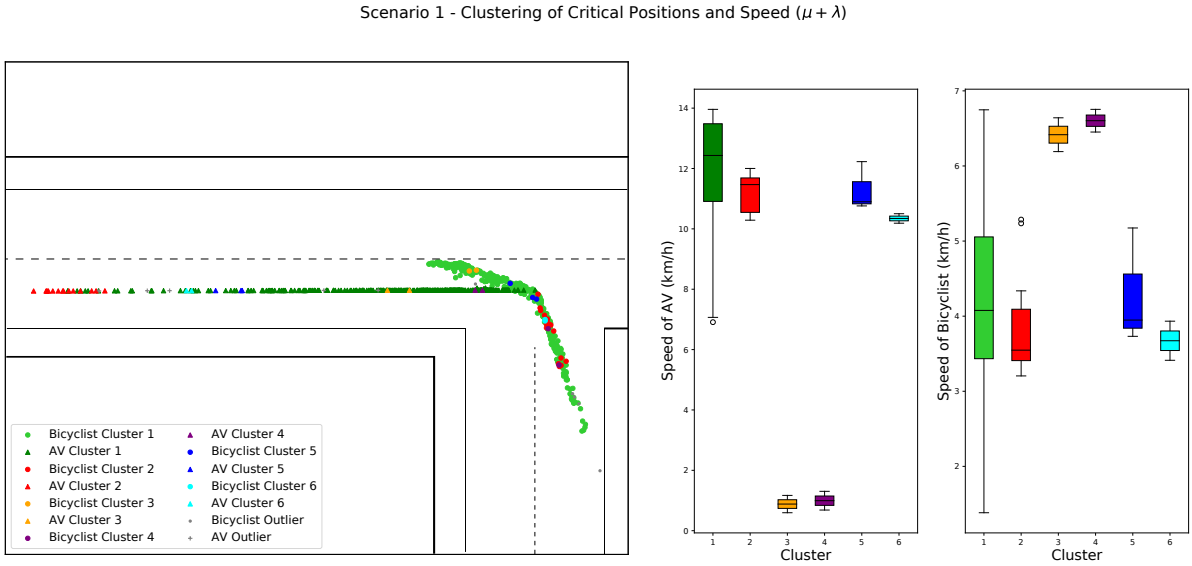


Figure 5.13: Clustering of the positions and speed of the AV and the bicyclist in critical situations generated with the $(\mu + \lambda)$ algorithm in scenario 1.

Before having a closer look at what distinguishes the clusters, it can be noted that in all clusters the speed of the AV is below 15 km/h. Hence, in all critical *concrete scenarios* the highest criticality is measured when the AV is already in the braking maneuver.

When examining the different clusters it holds for all three algorithms that the biggest cluster consists of critical situations in which the AV has a relatively high speed and a close distance to the bicyclist (cluster 1 for all algorithms). However, in all three algorithms clusters are also found in which this does not apply.
For example, for each of the three algorithms a cluster is found in which the AV has a relatively high speed but is still further away of the intersection (cluster 2 for $(\mu + \lambda)$). Additionally, in the data of all three algorithms at least one cluster is present in which the speed of the AV is low and the speed of the bicyclist is relatively high.

Hence, it can be said that EAs are able to find different kinds of critical situations in scenario 1. Furthermore, this analysis revealed neither an indication that a specific parameter assignment is never tried, nor has any of the three closer investigated algorithms

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html`, accessed on 18.05.2022

found a kind of critical situation that the others have not found.

5.3 Analysis of EAs in Scenario 2

In this section the results of the six implemented EAs in scenario 2 are analyzed. Thereby, it is investigated which algorithm performs the best and how the algorithms learn to choose parameters. Further, it is examined whether EAs can identify all kinds of critical situations that can be derived from scenario 2.
The evaluation of *PCI* showed that the metric is well able to measure criticality between the AV and the bicyclist riding in front. However, this evaluation left the question open how well the *PCI* is able to measure criticality in situations where the pedestrian is involved. Hence, the analysis also has a more detailed look on those critical situations.

In a first step, the performance of the six EAs is compared. The results of this comparison are presented in subsection 5.3.1. In a second step, it is analyzed in subsection 5.3.2 how the EAs learn to choose the parameters. At the end of the section, in subsection 5.3.3, the *concrete scenarios* generated by the EAs are clustered and analyzed in order to identify critical situations.

5.3.1 Algorithm Performance

Similar to the evaluation of the algorithm performances in scenario 1, the performances of the algorithms in scenario 2 is evaluated by comparing the number of generated critical *concrete scenarios*, the average fitness in the parent populations, the average fitness in the child populations as well as the highest criticality measured in every generation.

The comparison of the number of generated critical *concrete scenarios* is shown in figure 5.14. As threshold that determines above which *PCI* outcome a *concrete scenario* is considered critical is 872.24 joule, calculated using Youden's index in subsection 5.1.2.
The amount of generated critical *concrete scenarios* is for the two basic algorithms and the two algorithms in combination with self adaptation even higher than in scenario 1. In the most generations around 40 of the 50 generated *concrete scenarios* are critical. Similar to the first scenario, the steepest increase takes place in the first four generations.
The two algorithms combined with Rechenberg are also able to generate critical *concrete scenarios* in scenario 2. However, compared to the other algorithms they perform much worse.

The comparison of the average fitness in the parent population, the average fitness in the child population, and of the maximal achieved fitness in every generation are visualized in the appendix in figures D.9, D.10 and D.11. In these three comparisons similar properties as in scenario 1 can be observed.

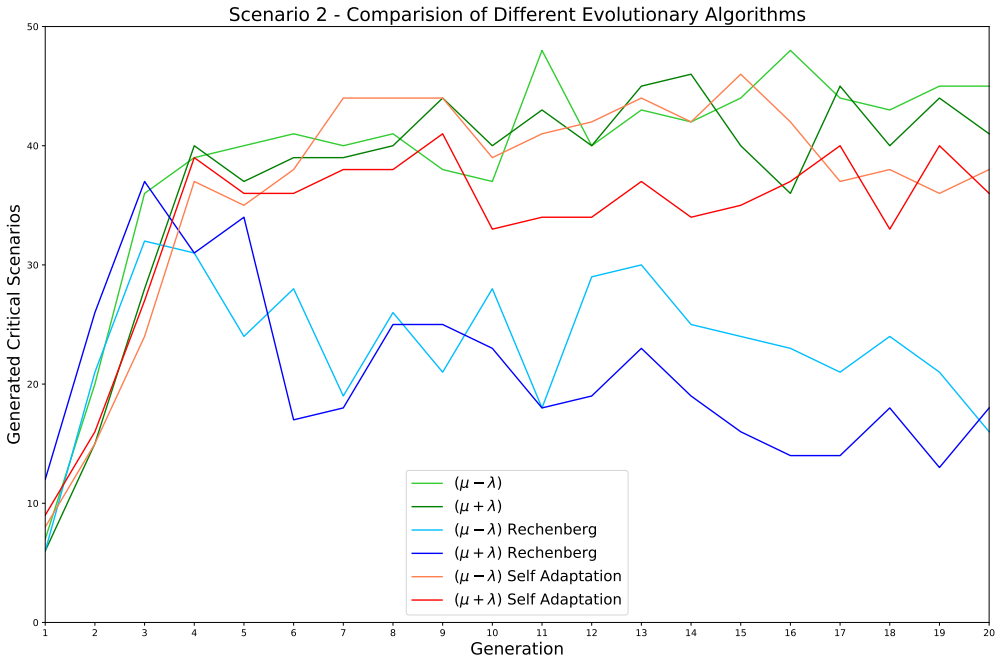


Figure 5.14: Comparison of the number of critical scenarios generated by the different algorithms over 20 generations in scenario 2. For all algorithms holds $\mu = 10$ and $\lambda = 50$.

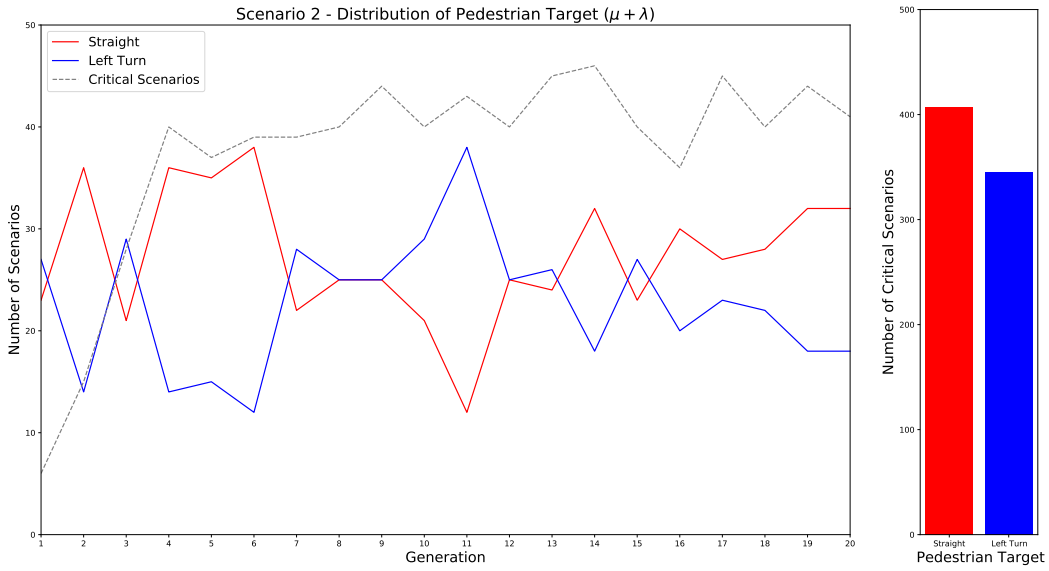
For example, the average fitness in the parent population is again higher for the three $(\mu+\lambda)$ algorithm variants. Further, while the $(\mu+\lambda)$ algorithm variants have the steepest increase in the first generations, the basic $(\mu-\lambda)$ algorithm keeps improving until the very end, resulting in the fact that is has the highest average children fitness and the highest measured criticality in the last generation.

Overall, the two basic variants of the $(\mu + \lambda)$ and $(\mu - \lambda)$ algorithms and the two algorithms in combination with self adaptation have good results in the four investigated characteristics. Hence, in the following two sections the *concrete scenarios* generated with these four algorithms are investigated more closely.

5.3.2 Parameter Analysis

Similar to the previous scenario, the first analyzed parameter assignment is the one of the binary parameter. However, this time critical situations can occur as well when the pedestrian crosses the road straight as when the pedestrian turns left before crossing the road. Thus, critical *concrete scenarios* can be generated with both parameter assignments.
Figure 5.15 shows how the distribution of the parameter assignment evolves over the 20 generations and how the parameter assignment is distributed in the generated critical *concrete scenarios*. Again, this figure is exemplary for the $(\mu + \lambda)$ algorithm, as all four closer investigated algorithms lead to similar results.
In contrast to the binary parameter in the first scenario, the algorithms do not learn to

decide for one of the two options. Instead, the algorithms are able to generate critical *concrete scenarios* with both parameter assignments. Hence, it can be said that the EAs are also able to handle binary parameters that can cause criticality with both assignments.



(a) (b)

Figure 5.15: (a) Evolution of the side of the pedestrian target over the generations of the $(\mu + \lambda)$ algorithm. (b) Distribution of the pedestrian target in the generated critical scenarios.

The next investigated parameter assignments are the target speed and mass of the AV. Because the results are similar for all four investigated algorithms, the results are exemplary visualized for the basic $(\mu + \lambda)$ algorithm.
As shown in the box plot in figure 5.16, the algorithms search the whole parameter range in the first generation and quickly learn to maximize the target speed thereafter.
In contrast to the first scenario, a similar learning process can be noticed for the mass of the AV. As shown in the box plot in figure 5.17, the algorithms learn quite fast that the best results are achieved when a mass close to the upper limit is chosen.

When investigating the correlation of the speed and the mass with the measured criticality it can be seen that the learning process of the EAs is purposeful. As shown in figure 5.18, a high criticality is achieved particularly when the target speed and mass are chosen with a high value. Thus, it can be said that a high target speed and a high mass of the AV are criticality phenomena in scenario 2.

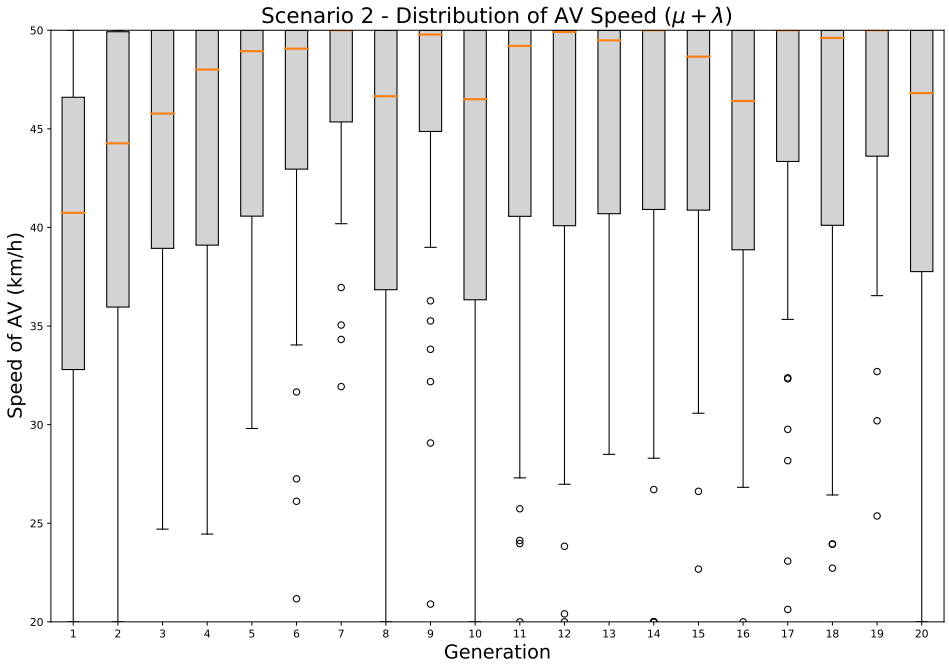


Figure 5.16: Box plot that visualizes the evolution of the speed of the AV over 20 generations of the $(\mu + \lambda)$ algorithm in scenario 2.

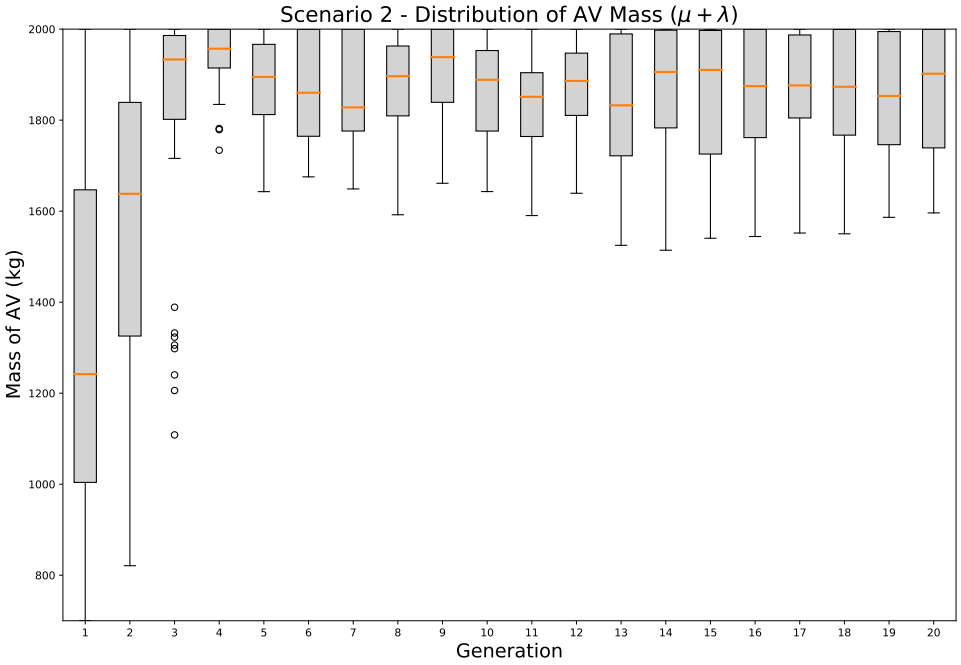


Figure 5.17: Box plot that visualizes the evolution of the mass of the AV over 20 generations of the $(\mu + \lambda)$ algorithm in scenario 2.

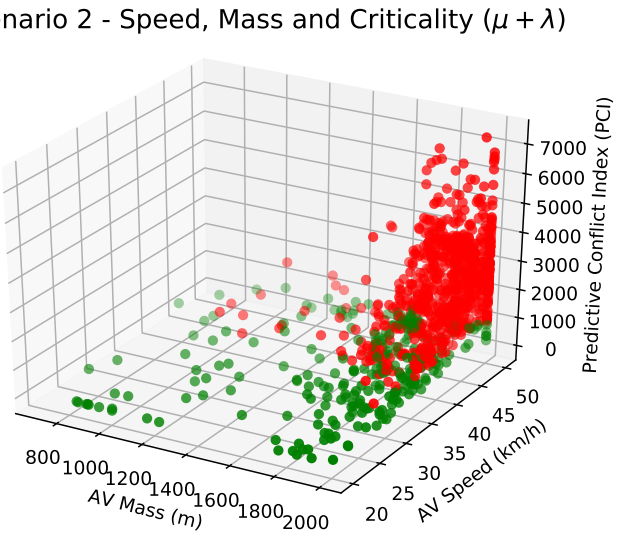


Figure 5.18: The correlation of the AV's speed, mass, and measured criticality in the generated *concrete scenarios* with the $(\mu + \lambda)$ algorithm in scenario 2.

5.3.3 Analysis of Critical Situations

In scenario 2, scenes can occur in which criticality is measured only between the AV and the bicyclist, only between the AV and the pedestrian, and between the AV and both VRUs. Hence, it is particularly interesting with respect to research question 4 to investigate whether all of these scenes occur in the *concrete scenarios* generated by the EAs. Further, in the critical scenes in which criticality was measured between the AV and the pedestrian, it is interesting to examine where the pedestrian crosses the road.
In order to analyze this, the critical *concrete scenarios* generated by each of the EAs are divided into three groups according to the actors between whom criticality was measured in the most critical scene. The different groups are then analyzed independently of each other.

First of all it can be said, that all four algorithms generate *critical scenarios* for every of the three groups. When comparing the size of the groups, it is seen that in the majority criticality is only measured in the most critical scene between the AV and the bicyclist. A visualization of this for the $(\mu + \lambda)$ algorithm is shown in figure D.12 in the appendix.
In figure 5.19 the distribution of the involved VRUs over the 20 generations of the $(\mu + \lambda)$ algorithm is shown. Here it can additionally be seen that in the first generations the algorithm only creates critical *concrete scenarios* in which criticality arises between the AV and the bicyclist. After a few generations the algorithm also learns to generate *concrete scenarios* in which criticality is measured with both VRUs at the same time. After this, it takes another eight generations until the first critical *concrete scenarios* are generated in which criticality is only measured between the AV and the pedestrian.

Overall, it can be said that the EA is best able in generating critical *concrete scenarios* involving only the AV and the bicyclist. However, over the course of the generations the algorithm learns to construct *concrete scenarios* with critical situations involving the pedestrian. Furthermore, the share of *concrete scenarios* including critical situations with the pedestrian increases by the number of generations.

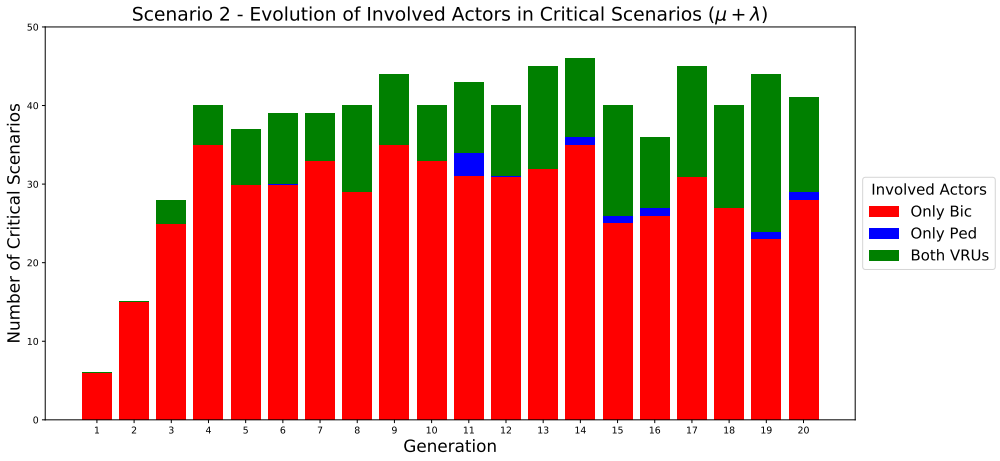


Figure 5.19: The number of critical *concrete scenarios* generated by the $(\mu + \lambda)$ algorithm in scenario 2 in every generation distinguished by the involved actors in the most critical situation.

Due to the amount of generated *concrete scenarios*, the situations involving only the AV and the bicyclist are examined in more detail first. Therefore, the positions and speed of the AV and the bicyclist in those situations are clustered using DBSCAN. The results of the clustering are shown in figure 5.20.

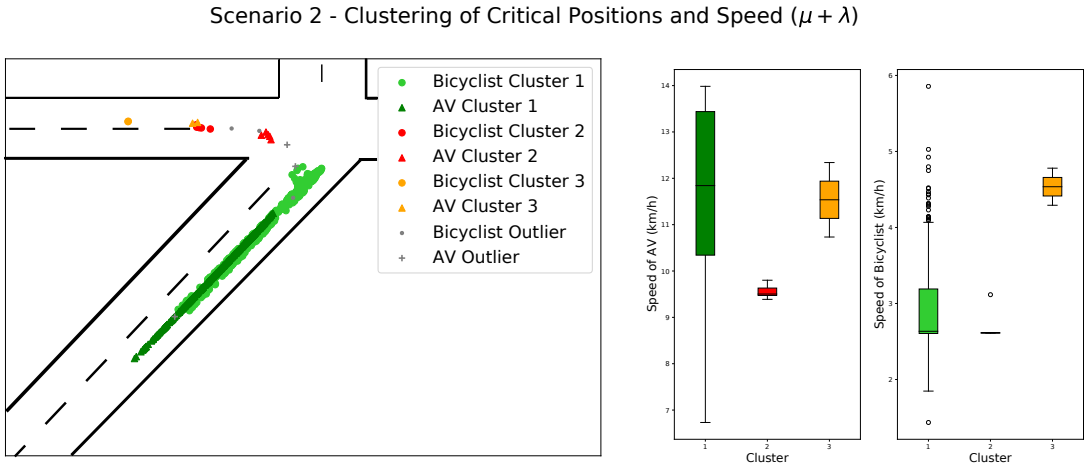
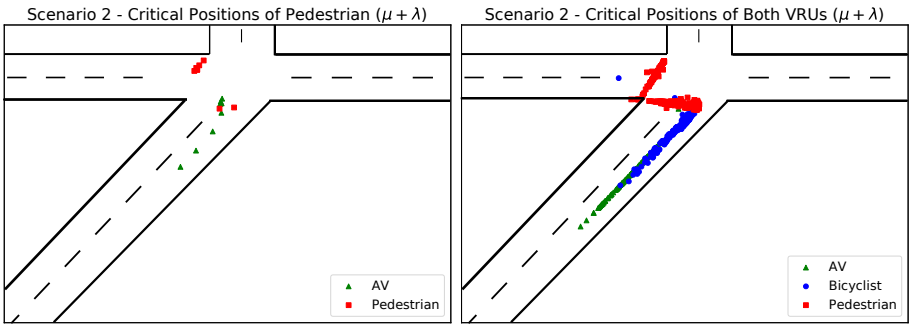


Figure 5.20: Clustering of the positions and speed from the AV and the bicyclist in critical situations involving only the AV and the bicyclist generated with the $(\mu + \lambda)$ algorithm in scenario 2.

The biggest cluster found consists of situations in which the AV drives close to the bicyclist before the left turn. In these situations, the AV mostly has a relatively high speed while the bicyclist mostly has a relatively low speed.
In order to understand what causes these critical situations, the distance of the start positions to the intersection and the target speed of the AV and the bicyclist are investigated. As shown in figure D.13 in the appendix, it holds for all critical situations that are part of cluster 1 that the AV started with a large distance to the intersection and a high target speed, while the bicyclist started with a small distance to the intersection and a low target speed. In this way, the AV has time and space to accelerate to the target speed, before it has to break because of the bicyclist. Hence, the critical situations could be avoided, if the AV earlier reacts to the bicyclist.

Additionally to the large cluster of critical situations before the left turn, also a cluster of critical situations during the left turn and a cluster of critical situations after the left turn is found. Thus, EAs are able to find different kinds of critical situations involving the AV and the bicyclist in scenario 2.

In the next step, the critical situations involving only the AV and the pedestrian and the critical situations involving both VRUs are examined. Here, it is particularly interesting to have a look at the positions of the pedestrian, in order to see whether the algorithm is able to identify critical situations with both binary parameter assignments. Hence, the positions of the involved actors in the most critical situations are shown in figure 5.21.
For both actor combinations, critical situations have been found in which the pedestrian crosses the road straight and in which the pedestrian turns left to cross the road. Hence, the EA is again able to find different kinds of critical situations.



(a) (b)

Figure 5.21: (a) Positions of the AV and pedestrian in critical situations involving only the AV and the pedestrian generated with the $(\mu + \lambda)$ algorithm. (b) Positions of the AV, the bicyclist and pedestrian in critical situations involving all three actors generated with the $(\mu + \lambda)$ algorithm .

Similar results are found for the other three investigated algorithms. Overall, the four EAs are well able to identify critical situations in scenario 2 and thus answer research question

3. The majority of found critical situations only involves the AV and the bicyclist and occurs before the left turn. However, critical situations at other locations and with other involved actors are also found. Again, there is no indication that a kind of critical situation is not identified by the algorithms. Even though it can not be definitively said that really all kinds of critical situations have been found, EAs seem to be a promising approach to also answer research question 4.

5.4 Analysis of EAs Scenario 3

In the third scenario, all tested criticality metrics struggle to calculate a good approximation of the occurring criticality. However, *PCI* is with a sensitivity of 0.62 and a specificity of 0.79 the best performing criticality metric among the evaluated. In order to evaluate whether this is good enough for a metric to be used as fitness function for an EA that identifies critical situations in the scenario, the *concrete scenarios* generated by the algorithms are analyzed.

Therefore, in subsection 5.4.1 the performance of the six implemented EAs is compared. Further, in order to understand how the algorithms learn to generate critical *concrete scenarios*, the evolution of the parameter assignments is investigated in subsection 5.4.2. In the last subsection 5.4.3, the generated *concrete scenarios* are analyzed to identify critical situations. In this way it is determined, whether EAs with *PCI* as fitness function are able to identify different kinds of critical situations that can be derived from scenario 3.

5.4.1 Algorithm Performance

In order to evaluate the performance of the six implemented EAs, the number of generated critical *concrete scenarios*, the average fitness of the parent populations, the average fitness of the child populations and the highest measured criticality are compared. For each of the four characteristics the evolution over the 20 generations is investigated. Therefore, the threshold of 703 joule from subsection 5.1.3 is used to decide whether a *concrete scenario* is critical.

The number of generated critical *concrete scenarios* is shown in figure 5.22. In this figure, it can be seen that more than half of the generated *concrete scenarios* are classified critical for the two basic algorithm variants and the two algorithms combined with self adaptation. Also in this scenario, the steepest increase takes place in the first generations.
Again, the two Rechenberg variants perform worse than the other ones. Since scenario 3 consists only of continuous parameters, the poor performance of the Rechenberg algorithms is not only due to their problems with binary parameters. Given that they perform the worst in all three scenarios, it can be said that EAs in combination with Rechenberg

are not well suited to generate critical *concrete scenarios* in urban traffic.

The figures that visualize the comparison of the other three characteristics are shown in the appendix in D.14, D.15 and D.16.
As in the two previous scenarios, the average fitness of the parent population is higher for the three $(\mu + \lambda)$ algorithm variants than for the $(\mu - \lambda)$ ones. When examining the average fitness of the child population and the highest measured criticality, the two basic $(\mu+\lambda)$ and $(\mu-\lambda)$ variants and the $(\mu+\lambda)$ algorithm combined with self adaptation have the best performance. Hence, in the following two subsections these three algorithms are investigated in detail.

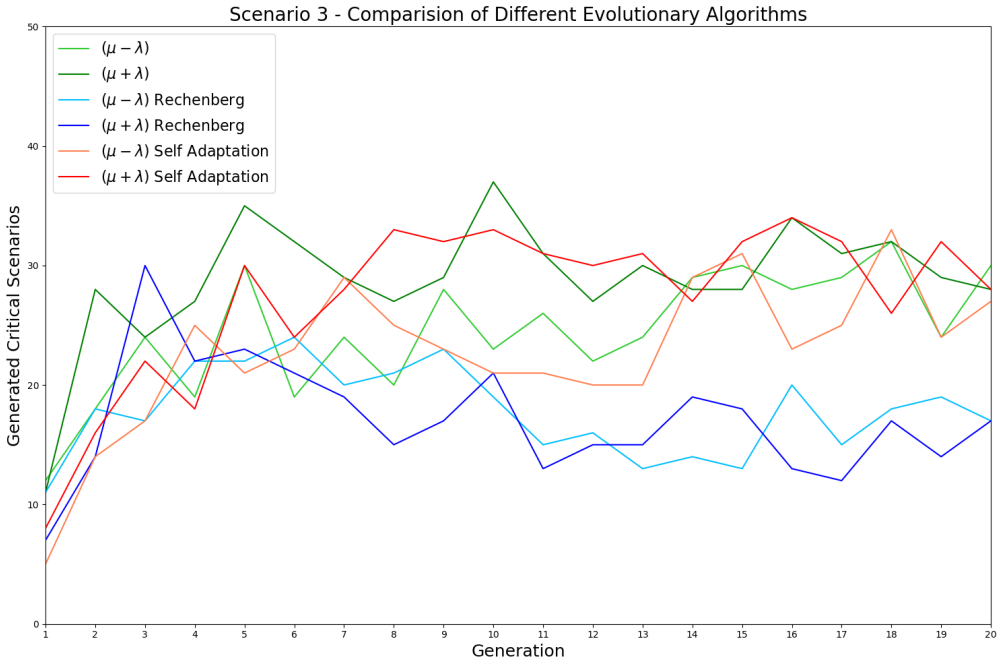


Figure 5.22: Comparison of the number of critical scenarios generated by the different algorithms over 20 generations in scenario 3. For all algorithms holds $\mu = 10$ and $\lambda = 50$.

5.4.2 Parameter Analysis

In order to understand how the basic $(\mu+\lambda)$ and $(\mu-\lambda)$ algorithms as well as the $(\mu+\lambda)$ algorithm with self adaptation generate critical *concrete scenarios*, the evolution of the parameter assignments over the generations is analyzed in this subsection. Further, this analysis gives information about potential causes of criticality in the scenario. Since the main focus of this work is to identify situations in which an AV endangers VRUs, the analysis focuses on the parameters of the AV.

In figure 5.23, the evolution of the parameter assignment for the AV's target speed is visualized for the $(\mu + \lambda)$ algorithm. The evaluation of the target speed for the other two

algorithms is similar. As already seen in the previous two scenarios, the algorithms start by searching the complete parameter range in the first generations and learn to maximize the target speed thereafter. Hence, in all three scenarios the algorithms learn to maximize the target speed of the AV. Thus, it can be inferred that a high speed of the AV is a cause of criticality in urban traffic.

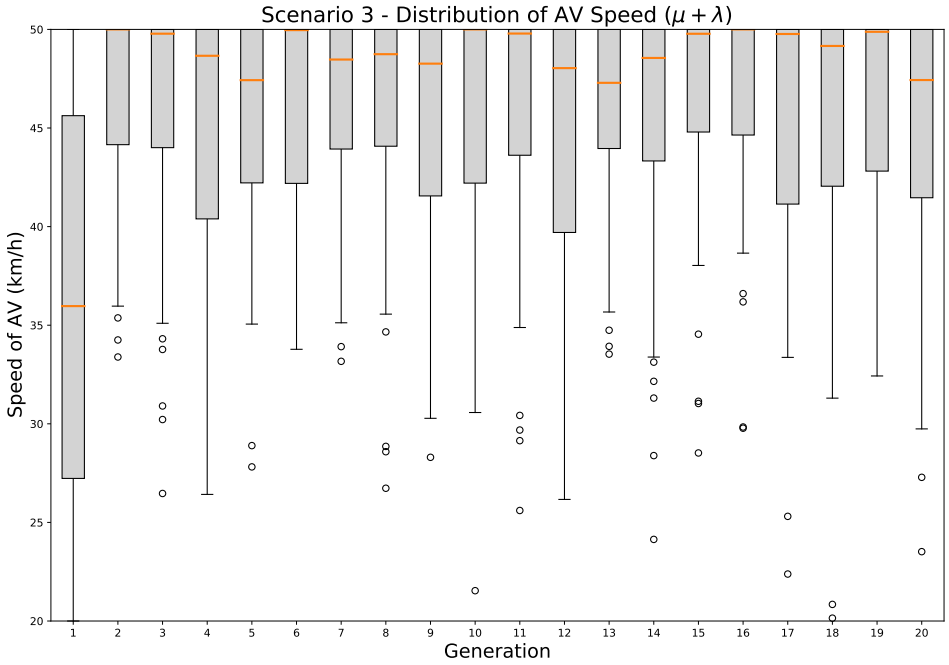


Figure 5.23: Box plot that visualizes the evolution of the speed of the AV over 20 generations of the $(\mu + \lambda)$ algorithm in scenario 3.

The next investigated parameter is the mass of the AV. In figure 5.24, the distribution of its parameter assignment in the $(\mu+\lambda)$ algorithm is visualized as a box plot. In figure 5.25, the same visualization is shown for the $(\mu - \lambda)$ algorithm. The distribution of the $(\mu + \lambda)$ algorithm combined with self adaptation is similar to the one of the $(\mu - \lambda)$ algorithm.
While the $(\mu - \lambda)$ algorithm and the $(\mu + \lambda)$algorithm combined with self adaptation quickly learn to maximize the mass of the AV, the $(\mu + \lambda)$ algorithm only started this learning process in the first few generations and goes back to searching the complete parameter space for the rest of the algorithm. A possible reason for this behavior becomes visible when comparing the correlation of the target speed, the mass of the AVs, and the measured criticality for the three algorithms.
As shown in figure 5.26, the $(\mu + \lambda)$ algorithm finds critical *concrete scenarios* when the speed and the mass are maximized, and when only the speed is maximized and the mass is chosen to be around 1000 kg. In contrast to this, the other two algorithms only find critical *concrete scenarios* when both, the speed and the mass, are maximized.
Thus, the $(\mu + \lambda)$ generates a kind of critical *concrete scenarios* that the other algorithms

missed.

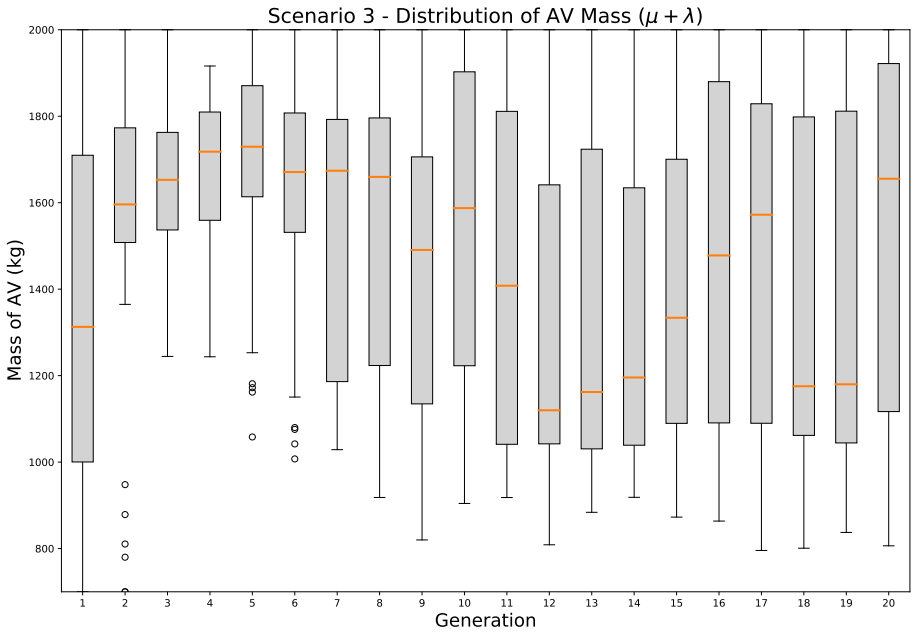


Figure 5.24: Box plot that visualizes the evolution of the mass of the AV over 20 generations of the $(\mu + \lambda)$ algorithm in scenario 3.

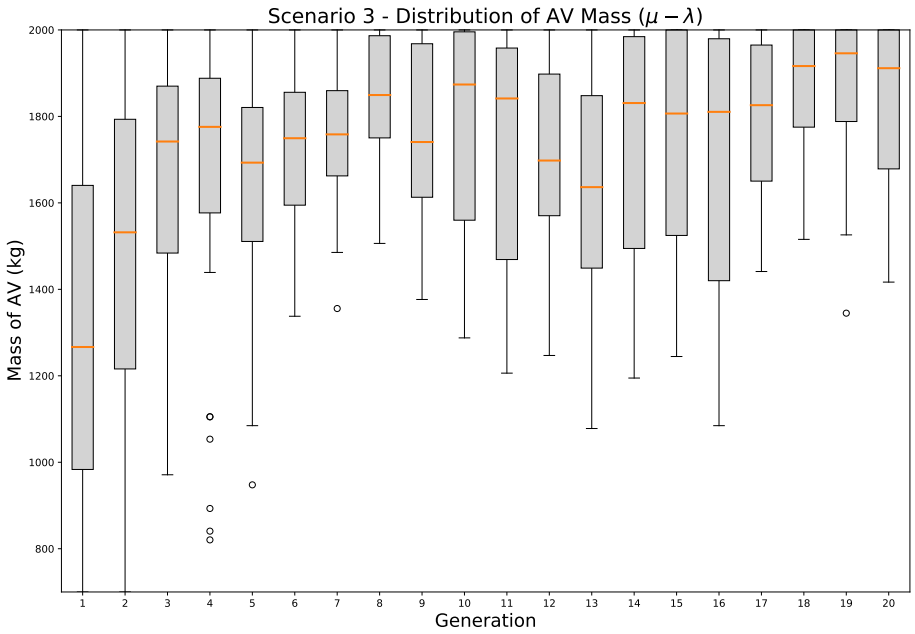
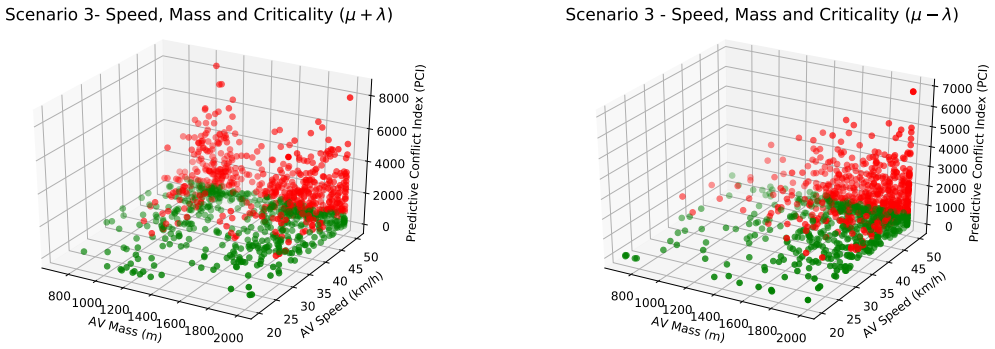


Figure 5.25: Box plot that visualizes the evolution of the mass of the AV over 20 generations of the $(\mu - \lambda)$ algorithm in scenario 3.



(a) (b)

Figure 5.26: (a) The correlation of the AV's speed, mass, and measured criticality in scenario 3 in *concrete scenarios* generate with (a) the $(\mu + \lambda)$ algorithm and (b) the $(\mu - \lambda)$ algorithm

5.4.3 Analysis of Critical Situations

In the last part of the analysis of scenario 3, the most critical scenes of the *concrete scenarios* generated by the $(\mu + \lambda)$ algorithm are analyzed. The aim of this analysis is to check whether all kinds of critical situations that can occur in scenario 3 also occur in the generated *concrete scenarios*.

Due to the fact that four VRUs participate in scenario 3, different combinations of actors can be involved in critical situations. In particular, 15 different combinations, from only the AV and one of the four VRUs to the AV and all four VRUs are possible.
Figure 5.27 shows for every combination the amount of most critical scenes from the generated *concrete scenarios* in which criticality was measured. Here it can be seen that for every of the possible combinations critical situations are found.

In figure 5.28 it is visualized how the critical situations with different combinations of actors evolve over the generations. In the first generation, the algorithm only generates critical *concrete scenarios* in which criticality arises between the AV and maximum two VRUs. From the second generation on, the algorithm also generates critical *concrete scenarios* in which criticality arises between the AV and three VRUs. Starting from the fourth generation, the algorithm is also able to generate critical *concrete scenarios* where criticality arises between the AV and all four VRUs.
Hence, the algorithm learns how to set the parameters in order to construct *concrete scenarios* in which criticality is measured for as many VRUs as possible.

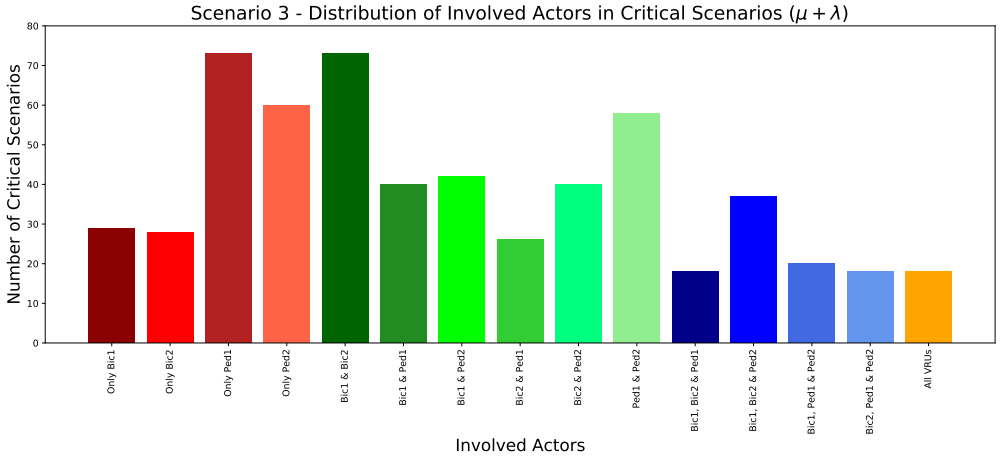


Figure 5.27: Distribution of the involved actors in the most critical situation in *concrete scenarios* generated by the $(\mu + \lambda)$ algorithm in scenario 3.

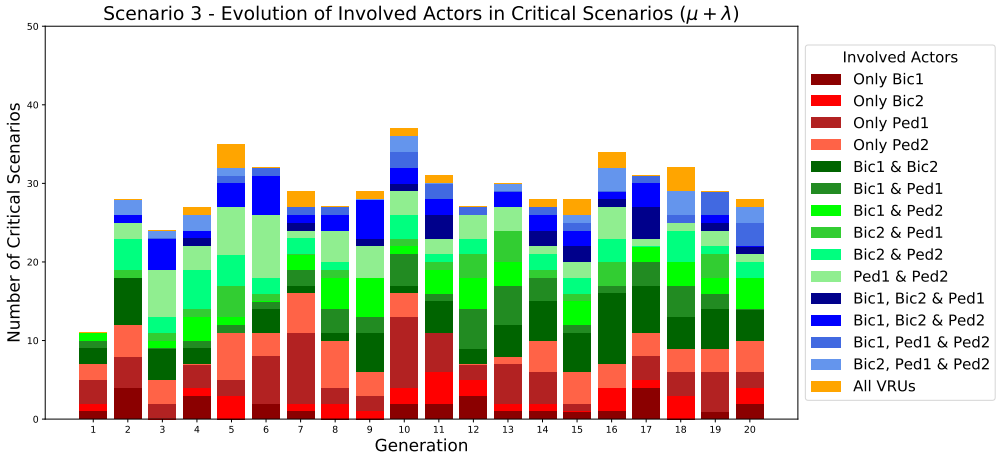


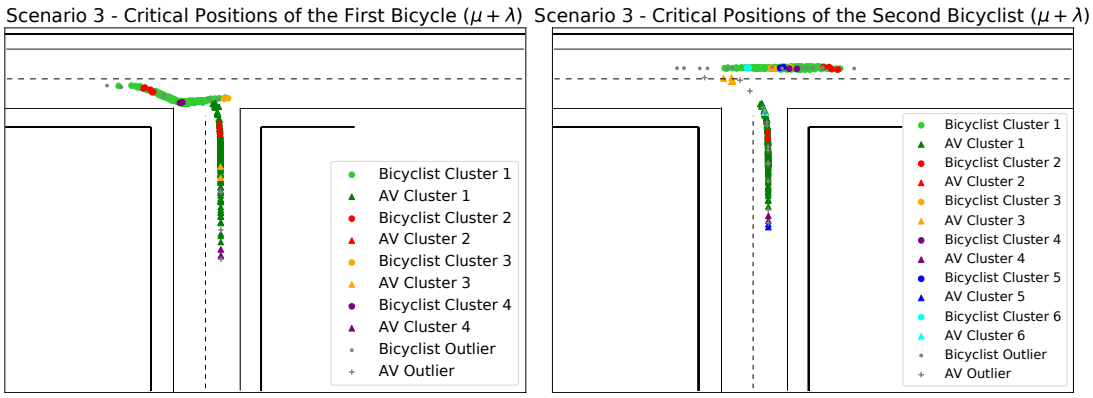
Figure 5.28: The number of critical *concrete scenarios* generated by the $(\mu + \lambda)$ algorithm in scenario 3 in every generation distinguished by the involved actors in the most critical situation.

In a next step, it is investigated whether the algorithm is also able to identify different accident types. Therefore, for each of the two bicyclists, the position of the AV and the bicyclist in critical situations are clustered using DBSCAN.

On the one hand, criticality can arise between the AV and the first bicyclist driving from left to right either when the AV drives with a high speed sideways towards the bicyclist or when the AV drives head-on towards the bicyclist during the turn.

On the other hand, criticality can arise between the AV and the second bicyclist driving from right to left either also when the AV drives with a high speed sideways towards the bicyclist or when the AV drives into the back of the bicyclist after the left turn.

The target of the clustering is to investigate whether all of these critical situations are identified by the EA. A visualization of the found clusters is shown in figure 5.29.



(a) (b)

Figure 5.29: (a) Clustered positions of the AV and the first bicyclist in critical situations generated with the $(\mu + \lambda)$ algorithm. (b) Clustered positions of the AV and the second bicyclist in critical situations generated with the $(\mu+\lambda)$ algorithm.

In figure 5.29 (a) the found clusters of the positions of the AV and the first bicyclist driving from left to right are shown. Based on the position of the AV, it can been seen that the most critical situations are potential side collisions. Despite the fact that different clusters are found, it is not clear whether any of the critical situations are also possible head-on collisions. Hence, it is unclear if both kinds of accidents occur among the identified critical situations.
In figure 5.29 (b) the found clusters of the positions of the AV and the second bicyclist driving from right to left can be seen. Again, the majority of the critical situations seem to be potential side collisions. However, the outliers indicate that possible rear end collisions are also among the identified situations. Furthermore, with cluster 3 even another kind of critical situations is identified in which the AV cuts off the bicyclists way. Thus, the algorithm is well able to identify different kinds of critical situations between these two actors.

Overall, it can be said that the $(\mu + \lambda)$ algorithm is able to identify many different and versatile kinds of critical situations. Especially in terms of involved actors, the algorithm generates *concrete scenarios* that cover all possible combinations. Hence, the performance of the algorithm indicates that the approach is well able to answer research question 3 and is also promising to answer research question 4.
However, due to the low sensitivity of the fitness function, many situations that have been identified as critical by the algorithm might not be truly critical. Therefore, it would be reasonable to search for a better suited fitness function for scenario 3.

6 Conclusion and Future Work

In this master's thesis an approach is proposed how EAs can be applied to identify critical situations between AVs and VRUs in urban traffic. For the evaluation of this approach four research questions are investigated.

Research question 1 addresses the problem how to analyze criticality in urban traffic without endangering VRUs. In order to answer research question 1, the urban traffic simulator Carla is used.
Overall, Carla enables the execution of a high number of *concrete scenarios* at low cost and without endangering human lives. Both the execution of the concrete scenarios generated by the EAs and the evaluation of the fitness function are efficiently possible. Moreover, the obtained results seem plausible and can be well transferred into the real traffic world. However, when dealing with VRUs, there are still several points in which the simulator needs to improve.
First, the implemented Carla agents controlling AVs seem to be mainly designed for the interaction with other four-wheeled road users and have inaccuracies when dealing with VRUs. The original variant of the *behavioral agent*, for example, only recognizes other actors if they drive in the middle of the lane. Further, it is only designed to tailgate other vehicles. For a realistic modeling of urban traffic, the agent should also be able to overtake bicyclists driving on the road.
Second, the agents available to model the VRUs behavior need to become more versatile. Bicyclists in Carla behave similar to four-wheeled road users as they only drive on the road and follow waypoints in a straight line. For a realistic behavior, a much more flexible behavior of bicyclists needs to be implemented. Also pedestrians have all the same straightforward movement pattern. Again, this movement has to become more multifaceted and movement patterns that correspond, for example, with heavily loaded pedestrians or disabled humans need to be implemented.
If the simulation does not improve with respect to this aspects, it is not possible to identify all circumstances in which AVs endanger VRUs.

Research question 2 investigates how to measure criticality in urban traffic. Therefore, the new criticality metric PCI is proposed and validated in three urban traffic scenarios. The validation in scenario 1 and the similarity to a_{req} in this scenario indicate that PCI is well able to quantify criticality when an AV drives towards an intersection with a crossing VRU. Moreover, the validation in scenario 2 and the similarity to TTC in this scenario indicate that PCI is also well able to quantify criticality when an AV drives behind a VRU. From all compared criticality metrics, PCI is the only one that is able to quantify

criticality in both circumstances. Hence, *PCI* is a criticality metric that is versatile applicable and therefore well suited to measure criticality in simple urban traffic scenarios. However, as shown by the validation in scenario 3, *PCI* reaches its limits when too many VRUs are involved. Despite the fact that *PCI* performs better in this scenario than all other investigated criticality metrics, there is still further research necessary to model an even better suited criticality metric for those kinds of scenarios. In addition, the way how the ground truth is generated should also be improved to get a more valid validation of the criticality metric.

Research question 3 examines whether the set of critical *concrete scenarios* generated by an EA can be used to identify critical situations between AVs and VRUs in urban traffic. To this end, six different EAs are implemented and the generated *concrete scenarios* are analyzed.
As result of this analysis it can be said that the basic $(\mu + \lambda)$ algorithm, the basic $(\mu - \lambda)$ algorithm and the $(\mu + \lambda)$ algorithm in combination with self adaptation are well able to generate a large set of critical *concrete scenarios* for all three scenarios. With the help of this set many critical situations in urban traffic can be identified. Moreover, the learning process of the algorithms helps to understand why criticality occurs in these situations. Thus, different criticality phenomena that cause criticality during the interaction of AVs and VRUs are also identified.

The last research question, research question 4, addresses the problem how it can be assured that the by the EAs identified critical situations cover the whole range of critical situations occurring in an urban traffic scenario. In order to investigate this, different kinds of critical situations that can theoretically occur in each of the three investigated scenarios are specified in advance. The outcomes of the algorithms are then examined to see whether they have found all of the specified situations.
With regard to the number of involved actors in a critical situation, the EAs have identified critical situations with all possible combinations. Also in terms of the location and type of accidents almost all of the previous specified can be identified. Only in one case it is not totally clear, whether a kind of accident is among the identified.
Overall, the application of EAs is a promising approach to identify a multifaceted range of critical situations. Nevertheless, it can not be said conclusively that really all kinds of critical situations are found.

Ultimately, the proposed approach is well able to identify critical situations between AVs and VRUs in urban traffic and is therefore highly promising to improve the development of AVs. In a next step it would be interesting to conduct a criticality analysis on the complete urban traffic domain and generate a scenario catalog which covers all criticality phenomena that occur in urban traffic. When applying EAs on this scenario catalog, an overview is gained about how criticality emerges in this domain.

Bibliography

[1] J. Shuttleworth, "SAE Standards News: J3016 automated-driving graphic update," https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic (accessed on 10.02.2022), Jan. 2019.

[2] C. Neurohr, L. Westhofen, M. Butz, M. H. Bollmann, U. Eberle, and R. Galbas, "Criticality Analysis for the Verification and Validation of Automated Vehicles," *IEEE Access*, vol. 9, pp. 18 016–18 041, 2021.

[3] P. Schneider, M. Butz, C. Heinzemann, J. Oehlerking, and M. Woehrle, "Towards threat metric evaluation in complex urban scenarios," 2021.

[4] A. P. Morris, N. Haworth, A. Filtness, D.-P. A. Nguatem, L. Brown, A. Rakotonirainy, and S. Glaser, "Autonomous Vehicles and Vulnerable Road-Users - Important Considerations and Requirements Based on Crash Data from Two Countries," *Behavioral Sciences*, vol. 11, no. 7, p. 101, Jul. 2021.

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, Nov. 2017, pp. 1–16.

[6] K. 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 *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, vol. 01, 2015, pp. 191–198.

[7] R. Hussain and S. Zeadally, "Autonomous Cars: Research Results, Issues, and Future Challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1275–1313, 2019.

[8] James M. Anderson, Nidhi Kalra, Karlyn D. Stanley, Paul Sorensen, Constantine Samaras, and Oluwatobi A. Oluwatola, *Autonomous Vehicle Technology: A Guide for Policymakers*. RAND Corporation, 2014.

[9] E. Shi, T. M. Gasser, A. Seeck, and R. Auerswald, "The principles of operation framework: A comprehensive classification concept for automated driving functions," *SAE International Journal of Connected and Automated Vehicles*, vol. 3, no. 12-03-01-0003, pp. 27–37, 2020.

[10] C. Urmson and W. Whittaker, "Self-Driving Cars and the Urban Challenge," *IEEE Intelligent Systems*, vol. 23, no. 2, pp. 66–68, Mar. 2008.

[11] World Health Organization, *Global Status Report on Road Safety 2018*. Geneva: World Health Organization, 2018.

[12] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[13] E. Parliament, "Road fatality statistics in the EU (infographic)," https://www.europarl.europa.eu/news/en/headlines/society/20190410STO36615/road-fatality-statistics-in-the-eu-infographic, Oct. 2021.

[14] C. Gao, G. Wang, W. Shi, Z. Wang, and Y. Chen, "Autonomous Driving Security: State of the Art and Challenges," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[15] K. Ren, Q. Wang, C. Wang, Z. Qin, and X. Lin, "The Security of Autonomous Driving: Threats, Defenses, and Future Directions," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 357–372, Feb. 2020.

[16] L. Westhofen, C. Neurohr, T. Koopmann, M. Butz, B. Schütt, F. Utesch, B. Kramer, C. Gutenkunst, and E. Böde, "Criticality metrics for automated driving: A review and suitability analysis of the state of the art," *Under Review for the Archives of Computational Methods in Engineering*, Aug. 2021.

[17] A. Alessandrini, A. Campagna, P. D. Site, F. Filippi, and L. Persia, "Automated Vehicles and the Rethinking of Mobility and Cities," *Transportation Research Procedia*, vol. 5, pp. 145–160, 2015.

[18] M. Maurer, J. C. Gerdes, B. Lenz, and H. Winner, Eds., *Autonomes Fahren*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.

[19] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, and W. Shi, "Computing Systems for Autonomous Driving: State of the Art and Challenges," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6469–6486, Apr. 2021.

[20] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for Development, Test and Validation of Automated Vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. Changshu: IEEE, Jun. 2018, pp. 1821–1827.

[21] I. Markit, "Autonomous vehicle sales to surpass 33 million annually in 2040, enabling new autonomous mobility in more than 26% of new car sales," https://ihsmarkit.com/research-analysis/autonomous-vehicle-sales-to-surpass-33-million-annually-in-2040-enabling-new-autonomous-mobility-in-more-than-26-percent-of-new-car-sales.html, Jan. 2018.

[22] J. Barnett, N. Gizinski, E. Mondragon-Parra, J. Siegel, D. Morris, T. Gates, E. Kassens-Noor, and P. Savolainen, "Automated Vehicles Sharing the Road: Surveying Detection and Localization of Pedalcyclists," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 4, pp. 649–664, Dec. 2021.

[23] J. P. Nuñez Velasco, Y. Mun Lee, J. Uttley, A. Solernou, H. Farah, B. van Arem, M. Hagenzieker, and N. Merat, "Will pedestrians cross the road before an automated vehicle? The effect of drivers' attentiveness and presence on pedestrians' road crossing behavior," *Transportation Research Interdisciplinary Perspectives*, vol. 12, p. 100466, Dec. 2021.

[24] M. Vilaça, N. Silva, and M. C. Coelho, "Statistical Analysis of the Occurrence and Severity of Crashes Involving Vulnerable Road Users," *Transportation Research Procedia*, vol. 27, pp. 1113–1120, 2017.

[25] M. Dietrich, "Addressing inequal risk exposure in the development of automated vehicles," *Ethics and Information Technology*, vol. 23, no. 4, pp. 727–738, Dec. 2021.

[26] European Conference of Ministers of Transport, *Safety in Road Traffic for Vulnerable Users*. OECD, Apr. 2000.

[27] H. Huttunen, F. S. Yancheshmeh, and K. Chen, "Car Type Recognition with Deep Neural Networks," *2016 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1115–1120, Jun. 2016.

[28] Y. O. Adu-Gyamfi, S. K. Asare, A. Sharma, and T. Titus, "Automated Vehicle Recognition with Deep Convolutional Neural Networks," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2645, no. 1, pp. 113–122, Jan. 2017.

[29] P. Hurney, P. Waldron, F. Morgan, E. Jones, and M. Glavin, "Review of pedestrian detection techniques in automotive far-infrared video," *IET Intelligent Transport Systems*, vol. 9, no. 8, pp. 824–832, Oct. 2015.

[30] C. Shen, X. Zhao, X. Fan, X. Lian, F. Zhang, A. R. Kreidieh, and Z. Liu, "Multi-receptive field graph convolutional neural networks for pedestrian detection," *IET Intelligent Transport Systems*, vol. 13, no. 9, pp. 1319–1328, Sep. 2019.

[31] P. Fairley, "Self-driving cars have a bicycle problem," *IEEE Spectrum*, vol. 54, no. 3, pp. 12–13, Mar. 2017.

[32] F. Camara, O. Giles, R. Madigan, M. Rothmuller, P. H. Rasmussen, S. A. Vendelbo-Larsen, G. Markkula, Y. M. Lee, L. Garach, N. Merat, and C. W. Fox, "Predicting pedestrian road-crossing assertiveness for autonomous vehicle control," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI: IEEE, Nov. 2018, pp. 2098–2103.

[33] A. Rasouli and J. K. Tsotsos, "Autonomous Vehicles That Interact With Pedestrians: A Survey of Theory and Practice," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 900–918, Mar. 2020.

[34] S. Eiffert, K. Li, M. Shan, S. Worrall, S. Sukkarieh, and E. Nebot, "Probabilistic Crowd GAN: Multimodal Pedestrian Trajectory Prediction Using a Graph Vehicle-Pedestrian Attention Network," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5026–5033, Oct. 2020.

[35] S. Ahmed, M. N. Huda, S. Rajbhandari, C. Saha, M. Elshaw, and S. Kanarachos, "Pedestrian and Cyclist Detection and Intent Estimation for Autonomous Vehicles: A Survey," *Applied Sciences*, vol. 9, no. 11, p. 2335, Jun. 2019.

[36] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. Gran Canaria, Spain: IEEE, Sep. 2015, pp. 982–988.

[37] P. Junietz, J. Schneider, and H. Winner, "Metrik zur Bewertung der Kritikalität von Verkehrssituationen und-szenarien," in *11. Workshop Fahrerassistenzsysteme*, 2017.

[38] P. Junietz, F. Bonakdar, and B. Klamann, "PEGASUS Bericht: Kritikalitätsmetriken," *FZD-Bericht*, 2018.

[39] O. Bozorg-Haddad, M. Solgi, and H. A. Loáiciga, *Meta-Heuristic and Evolutionary Algorithms for Engineering Optimization*, ser. Wiley Series in Operations Research and Management Science. Hoboken: John Wiley & sons, 2017.

[40] O. Kramer, *Machine Learning for Evolution Strategies*. New York, NY: Springer Berlin Heidelberg, 2016.

[41] A. Kumar, S. Pant, M. Ram, and O. P. Yadav, Eds., *Meta-Heuristic Optimization Techniques: Applications in Engineering*, 1st ed., ser. De Gruyter Series on the Applications of Mathematics in Engineering and Information Sciences. Boston: DE GRUYTER, 2022, no. 10.

[42] A. Petrowski, *Evolutionary Algorithms*. Hoboken, NJ: ISTE Ltd/John Wiley and Sons Inc, 2017.

[43] K.-L. Du and M. N. S. Swamy, *Search and Optimization by Metaheuristics*. Cham: Springer International Publishing, 2016.

[44] G. Singh and K. Deb, "Comparison of multi-modal optimization algorithms based on evolutionary algorithms," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 2006, pp. 1305–1312.

[45] A. Sloss and S. Gustafson, "Evolutionary algorithms review. arXiv 2019," *arXiv preprint arXiv:1906.08870*, 2019.

[46] T. Bäck, "An Overview of Parameter Control Methods by Self-Adaptation in Evolutionary Algorithms," *Fundamenta Informaticae*, vol. vol. 35, pp. 51–66, 1998.

[47] M. Klischat and M. Althoff, "Generating Critical Test Scenarios for Automated Vehicles with Evolutionary Algorithms," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. Paris, France: IEEE, Jun. 2019, pp. 2352–2358.

[48] A. Bussler, L. Hartjen, R. Philipp, and F. Schuldt, "Application of Evolutionary Algorithms and Criticality Metrics for the Verification and Validation of Automated Driving Systems at Urban Intersections," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 128–135.

[49] H.-P. Schöner, "Simulation in development and testing of autonomous vehicles," in *18. Internationales Stuttgarter Symposium*, M. Bargende, H.-C. Reuss, and J. Wiedemann, Eds. Wiesbaden: Springer Fachmedien Wiesbaden, 2018, pp. 1083–1095.

[50] W. Huang, Kunfeng Wang, Yisheng Lv, and FengHua Zhu, "Autonomous vehicles testing methods review," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. Rio de Janeiro, Brazil: IEEE, Nov. 2016, pp. 163–168.

[51] P. Jing, H. Hu, F. Zhan, Y. Chen, and Y. Shi, "Agent-Based Simulation of Autonomous Vehicles: A Systematic Literature Review," *IEEE Access*, vol. 8, pp. 79 089–79 103, 2020.

[52] CARLA Documentation, "CARLA Introduction," https://carla.readthedocs.io/en/latest/start_introduction/ (accessed on 23.03.2022), 2022.

[53] ——, "CARLA Agents," https://carla.readthedocs.io/en/0.9.13/adv_agents/ (accessed on 23.03.2022), 2022.

[54] W. K. M. Alhajyaseen, "The Integration of Conflict Probability and Severity for the Safety Assessment of Intersections," *Arabian Journal for Science and Engineering*, vol. 40, no. 2, pp. 421–430, Feb. 2015.

[55] E. Hauer, "Speed and Safety," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2103, no. 1, pp. 10–17, Jan. 2009.

[56] J. Padmanaban, "Influences of vehicle size and mass and selected driver factors on odds of driver fatality," in *Annual Proceedings/Association for the Advancement of Automotive Medicine*, vol. 47. Association for the Advancement of Automotive Medicine, 2003, p. 507.

[57] L. Gonçalves, A. Subtil, M. R. Oliveira, and P. de Zea Bermudez, "ROC Curve Estimation: An Overview," *REVSTAT-Statistical Journal*, vol. 12, no. 1, pp. 1–20, 2014.

[58] M. D. Ruopp, N. J. Perkins, B. W. Whitcomb, and E. F. Schisterman, "Youden Index and Optimal Cut-Point Estimatedfrom Observations Affected by a Lower Limit of Detection," *Biometrical Journal*, vol. 50, no. 3, pp. 419–430, 2008.

[59] Z. H. Hoo, J. Candlish, and D. Teare, "What is an ROC curve?" *Emergency Medicine Journal*, vol. 34, no. 6, pp. 357–359, Jun. 2017.

[60] R. Fluss, D. Faraggi, and B. Reiser, "Estimation of the Youden Index and its Associated Cutoff Point," *Biometrical Journal*, vol. 47, no. 4, pp. 458–472, Aug. 2005.

[61] O. Kramer, "Computational Intelligence I: Evolution Strategies," WS 2020/21.

[62] K. Potter, "Methods for presenting statistical information: The box plot," Jan. 2006.

[63] R. Xu and D. C. Wunsch, *Clustering*, ser. IEEE Press Series on Computational Intelligence. Oxford: Wiley, 2009.

A Parameter Spaces of Scenarios

A.1 Scenario 1 ($S_1 = S_{1,av} \times S_{1,bic} \times S1_{obs} \subset \mathbb{R}^1 3 \times \mathbb{N}^2$)

Table A.1: Parameter Space of the AV ($S_{1,av} \subset \mathbb{R}^4$)

Name	Type	Subset of	Minimum	Maximum	σ	Description
start_distance	continuous	$\mathbb{R}$	20	70	10	Distance from the AV's spawn point to the intersection (m)
target_distance	continuous	$\mathbb{R}$	20	30	10	Distance from the AV's target point to the intersection (m)
speed	continuous	$\mathbb{R}$	20	50	10	Target speed of the AV (km/h)
mass	continuous	$\mathbb{R}$	700	2000	100	Mass of the AV (kg)

Table A.2: Parameter Space of the Bicyclist ($S_{1,bic} \subset \mathbb{R}^8$)

Name	Type	Subset of	Minimum	Maximum	σ	Description
start_distance	continuous	$\mathbb{R}$	15	50	10	Distance from the bicyclist's spawn point to the intersection (m)
start_lane_position	continuous	$\mathbb{R}$	-1	1	1	Position on lane of bicyclist's spawn point
position_before_turn	continuous	$\mathbb{R}$	-1	1	1	Bicyclist's position on lane before turn
position_after_turn	continuous	$\mathbb{R}$	-1	1	1	Bicyclist's position on lane after turn
target_distance	continuous	$\mathbb{R}$	30	30	10	Distance from the bicyclist's target point to the intersection (m)
target_lane_position	continuous	$\mathbb{R}$	-1	1	1	Position on lane of bicyclist's target point
speed	continuous	$\mathbb{R}$	10	25	10	Target speed of the bicyclist (km/h)
mass	continuous	$\mathbb{R}$	40	100	10	Mass of the bicyclist (kg)

Table A.3: Parameter Space of the Obstacles ($S_{1,obs} \subset \mathbb{R} \times \mathbb{N}^2$)

Name	Type	Subset of	Minimum	Maximum	σ	Description
number_obstacles	discrete	$\mathbb{N}$	0	7	1	Number of parking vehicles
first_obstacle_distance	continuous	$\mathbb{R}$	1	20	1	Distance of the first obstacle to the intersection (m)
obstacle_side	binary	$\mathbb{N}$	0	1	0.2	Side of the parking vehicles

A.2 Scenario 2 ($S2 = S_{2,av} \times S_{2,bic} \times S_{2,ped} \subset \mathbb{R}^{15} \times \mathbb{N}$)

Table A.4: Parameter Space of the AV ($S_{2,av} \subset \mathbb{R}^4$)

Name	Type	Subset of	Minimum	Maximum	σ	Description
start_distance	continuous	$\mathbb{R}$	45	65	10	Distance from the AV's spawn point to the intersection (m)
target_distance	continuous	$\mathbb{R}$	20	30	10	Distance from the AV's target point to the intersection (m)
speed	continuous	$\mathbb{R}$	20	50	10	Target speed of the AV (km/h)
mass	continuous	$\mathbb{R}$	700	2000	100	Mass of the AV (kg)

Table A.5: Parameter Space of the Bicyclist ($S_{2,bic} \subset \mathbb{R}^8$)

Name	Type	Subset of	Minimum	Maximum	σ	Description
start_distance	continuous	$\mathbb{R}$	15	35	10	Distance from the bicyclist's spawn point to the intersection (m)
start_lane_position	continuous	$\mathbb{R}$	-1	1	1	Position on lane of bicyclist's spawn point
position_before_turn	continuous	$\mathbb{R}$	-1	1	1	Bicyclist's position on lane before turn
position_after_turn	continuous	$\mathbb{R}$	-1	1	1	Bicyclist's position on lane after turn
target_distance	continuous	$\mathbb{R}$	30	30	10	Distance from the bicyclist's target point to the intersection (m)
target_lane_position	continuous	$\mathbb{R}$	-1	1	1	Position on lane of bicyclist's target point
speed	continuous	$\mathbb{R}$	10	25	10	Target speed of the bicyclist (km/h)
mass	continuous	$\mathbb{R}$	40	100	10	Mass of the bicyclist (kg)

Table A.6: Parameter Space of the Pedestrian ($S_{2,ped} \subset \mathbb{R}^3 \times \mathbb{N}$)

Name	Type	Subset of	Minimum	Maximum	σ	Description
start_distance	continuous	$\mathbb{R}$	5	30	10	Distance from the pedestrian's spawn point to the intersection (m)
speed	continuous	$\mathbb{R}$	1	7	1	Target speed of the pedestrian (km/h)
mass	continuous	$\mathbb{R}$	40	100	10	Mass of the pedestrian (kg)
target_position	binary	$\mathbb{N}$	0	1	0.2	Target location of the pedestrian

A.3 Scenario 3 ($S3 = S_{3,av} \times S_{3,bic1} \times S_{3,bic2} \times S_{3,ped1} \times S_{3,ped2} \subset \mathbb{R}^{28}$)

Table A.7: Parameter Space of the AV ($S_{3,av} \subset \mathbb{R}^4$)

Name	Type	Subset of	Minimum	Maximum	σ	Description
start_distance	continuous	$\mathbb{R}$	45	65	10	Distance from the AV's spawn point to the intersection (m)
target_distance	continuous	$\mathbb{R}$	20	30	10	Distance from the AV's target point to the intersection (m)
speed	continuous	$\mathbb{R}$	20	50	10	Target speed of the AV (km/h)
mass	continuous	$\mathbb{R}$	700	2000	100	Mass of the AV (kg)

Table A.8: Parameter Space of the First Bicyclist ($S_{3,bic1} \subset \mathbb{R}^8$)

Name	Type	Subset of	Minimum	Maximum	σ	Description
start_distance	continuous	$\mathbb{R}$	15	35	10	Distance from the bicyclist's spawn point to the intersection (m)
start_lane_position	continuous	$\mathbb{R}$	-1	1	1	Position on lane of bicyclist's spawn point
position_before_turn	continuous	$\mathbb{R}$	-1	1	1	Bicyclist's position on lane before turn
position_after_turn	continuous	$\mathbb{R}$	-1	1	1	Bicyclist's position on lane after turn
target_distance	continuous	$\mathbb{R}$	30	30	10	Distance from the bicyclist's target point to the intersection (m)
target_lane_position	continuous	$\mathbb{R}$	-1	1	1	Position on lane of bicyclist's target point
speed	continuous	$\mathbb{R}$	10	25	10	Target speed of the bicyclist (km/h)
mass	continuous	$\mathbb{R}$	40	100	10	Mass of the bicyclist (kg)

Table A.9: Parameter Space of the Second Bicyclist ($S_{3,bic2} \subset \mathbb{R}^8$)

Name	Type	Subset of	Minimum	Maximum	σ	Description
start_distance	continuous	$\mathbb{R}$	15	35	10	Distance from the bicyclist's spawn point to the intersection (m)
start_lane_position	continuous	$\mathbb{R}$	-1	1	1	Position on lane of bicyclist's spawn point
position_before_turn	continuous	$\mathbb{R}$	-1	1	1	Bicyclist's position on lane before turn
position_after_turn	continuous	$\mathbb{R}$	-1	1	1	Bicyclist's position on lane after turn
target_distance	continuous	$\mathbb{R}$	30	30	10	Distance from the bicyclist's target point to the intersection (m)
target_lane_position	continuous	$\mathbb{R}$	-1	1	1	Position on lane of bicyclist's target point
speed	continuous	$\mathbb{R}$	10	25	10	Target speed of the bicyclist (km/h)
mass	continuous	$\mathbb{R}$	40	100	10	Mass of the bicyclist (kg)

Table A.10: Parameter Space of the First Pedestrian ($S_{3,ped1} \subset \mathbb{R}^4$)

Name	Type	Subset of	Minimum	Maximum	σ	Description
start_distance	continuous	$\mathbb{R}$	5	40	10	Distance from the pedestrian's spawn point to the intersection (m)
speed	continuous	$\mathbb{R}$	1	7	1	Target speed of the pedestrian (km/h)
mass	continuous	$\mathbb{R}$	40	100	10	Mass of the pedestrian (kg)
target_distance	continuous	$\mathbb{R}$	20	40	10	Distance from the pedestrian's target point to the intersection (m)

Table A.11: Parameter Space of the Second Pedestrian ($S_{3,ped2} \subset \mathbb{R}^4$)

Name	Type	Subset of	Minimum	Maximum	σ	Description
start_distance	continuous	$\mathbb{R}$	5	40	10	Distance from the pedestrian's spawn point to the intersection (m)
speed	continuous	$\mathbb{R}$	1	7	1	Target speed of the pedestrian (km/h)
mass	continuous	$\mathbb{R}$	40	100	10	Mass of the pedestrian (kg)
target_distance	continuous	$\mathbb{R}$	20	40	10	Distance from the pedestrian's target point to the intersection (m)

B Additional Pseudocode

```
1  def generate_initial_parent_population(mu=10):
2
3      equal_distributed_parameters = dict()
4
5      # iterate over every parameter of the logical scenario
6      for parameter in logical_scenario_parameters:
7
8          # save list of equal distributed parameters in dictionary
9          equal_distributed_parameters[parameter] = split_parameter_range(
10                     parameter, mu)
11
12         initial_parent_population = list()
13
14     # generate mu parents
15     for i in range(mu):
16
17         new_individual = dict()
18
19         # iterate over every parameter of the logical scenario
20         for parameter in logical_scenario_parameters:
21
22             # select a random value from the equal distributed parameters
23             value = random.sample(equal_distributed_parameters[parameter],
24                                   1)[0]
25
26             # remove the selected value from list
27             equal_distributed_parameters[parameter].remove(value)
28
29             new_individual[parameter] = value
30
31         initial_parent_population.append(new_individual)
32
33     return initial_parent_population
```

Figure B.1: Method to create an initial parent population with equal distributed parameters. In a first step, the parameter range of every parameter is divided into μ equal distributed, valid parameters (lines 3-10). In a second step, μ parents are created. Therefore one of the equal distributed parameters is chosen randomly for every parameter assignment (lines 12-31).

```
1  def crossover_and_mutate_child_population(parent_population, lam=50, kappa
       =2):
2
3      child_population = list()
4
5      # create lambda new children
6      for i in range(lam):
7
8          new_individual = dict()
9
10         # sample kappa parents
11         parents = random.sample(parent_population, kappa)
12
13         # iterate over every parameter
14         for parameter in logical_scenario_parameters:
15
16             # get the parameter assignments of the parents
17             parent_values = [parent[parameter] for parent in parents]
18
19             # randomly choose which value is inherited
20             inherited_value = random.choice(parent_values)
21
22             # binary parameters a flipped with a chance of sigma
23             if parameter in binary_parameters:
24
25                 if random.random() < sigma[parameter]:
26
27                     mutated_value = (inherited_value + 1) % 2
28
29                 else:
30
31                     mutated_value = inherited_value
32
33             # discrete parameters are rounded after mutation
34             elif parameter in discrete_parameters:
35
36                 mutated_value = int(round(inherited_value
37                         + sigma[parameter] * random.standard_normal()))
38
39             # continious parameters
40             else:
41
42                 mutated_value = inherited_value + sigma[parameter]
43                         * random.standard_normal()
44
45             # mutated values are adjusted to the valid limits
46             mutated_value = calculate_valid_mutation(mutated_value,
47                     parameter)
48
49             new_individual[parameter] = mutated_value
50
51         child_population.append(new_individual)
52
53     return child_population
```

Figure B.2: Method to generate a new child population from given parents. First a crossover between randomly selected parents takes place (lines 16-20). After this a mutation is applied (lines 22 - 47).

```
1  tau = 1/math.sqrt(len(logical_scenario_parameters))
2
3  def crossover_and_mutate_child_population(parent_population, sigma, lam=50,
       kappa=2):
4
5      child_population = list()
6
7      for i in range(lam):
8
9          new_individual = dict()
10         new_sigma = dict()
11
12         parents = random.sample(parent_population, kappa)
13
14         for parameter in logical_scenario_parameters:
15
16             parent_values = [parent[parameter] for parent in parents]
17
18             inherited_value = random.choice(parent_values)
19
20             # mutate a new sigma for the individual
21             mutated_sigma = sigma[parameter] * math.exp(tau
22                         * random.standard_normal())
23
24             if parameter in binary_parameters:
25
26                 mutated_sigma = min(1, max(0, mutated_sigma))
27
28                 if random.random() < mutated_sigma:
29
30                     mutated_value = (inherited_value + 1) % 2
31
32                 else:
33
34                     mutated_value = inherited_value
35
36             elif parameter in discrete_parameters:
37
38                 mutated_value = int(round(inherited_value + mutated_sigma
39                             * random.standard_normal()))
40
41             else:
42
43                 mutated_value = inherited_value
44                      + mutated_sigma * random.standard_normal()
45
46             mutated_value = calculate_valid_mutation(mutated_value,
47                         parameter)
48
49             new_individual[parameter] = mutated_value
50             new_sigma[parameter] = mutated_sigma
51
52         # save new individual and corresponding sigma
53         child_population.append((new_individual, new_sigma))
54
55     return child_population
```

Figure B.3: Method to generate a new child population from given parents for self adaptation. For every individual a new sigma is mutated (line 21). This sigma is saved together with the individual (line 53).

C PCI Validation Results

Table C.1: Comparison of Results from the ROC Curves in Scenario 1

Metric	**AUC**	**Best Threshold (c^*)**	**Sensitivity**	**Specificity**
PCI	0.9411	247.7946 *joule*	0.9361	0.8562
a_{req}	0.9243	9.8002 m/s^2	0.8723	0.9673
PrET	0.7203	0.6776 s	0.8723	0.4967
TTC	0.8177	0.2095 s	0.7872	0.8627

Table C.2: Comparison of Results from the ROC Curves in Scenario 2

Metric	**AUC**	**Best Threshold (c^*)**	**Sensitivity**	**Specificity**
PCI	0.8440	872.2420 *joule*	0.7372	0.8889
a_{req}	0.6247	5.5575 m/s^2	0.3358	0.9048
PrET	0.7410	0.9256 s	0.6277	0.7460
TTC	0.87261	0.2418 s	0.7080	0.9365

Table C.3: Comparison of Results from the ROC Curves in Scenario 3

Metric	**AUC**	**Best Threshold (c^*)**	**Sensitivity**	**Specificity**
PCI	0.7313	703.0080 *joule*	0.6211	0.7905
a_{req}	0.5633	3.9853 m/s^2	0.3579	0.7714
PrET	0.4938	2.0383 s	0.1263	0.9714
TTC	0.5962	0.3058 s	0.6	0.6

D Further Evaluations of EAs

D.1 EAs in Scenario 1

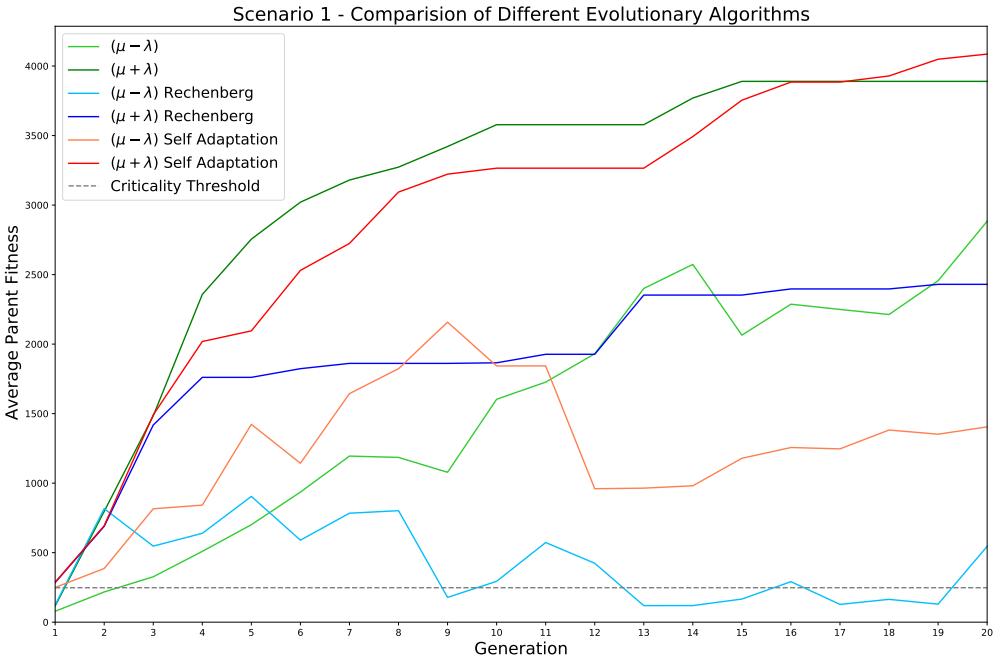


Figure D.1: Comparison of the average parent fitness of the scenarios generated by the different algorithms over 20 generations in scenario 1. For all algorithms holds $\mu = 10$ and $\lambda = 50$.

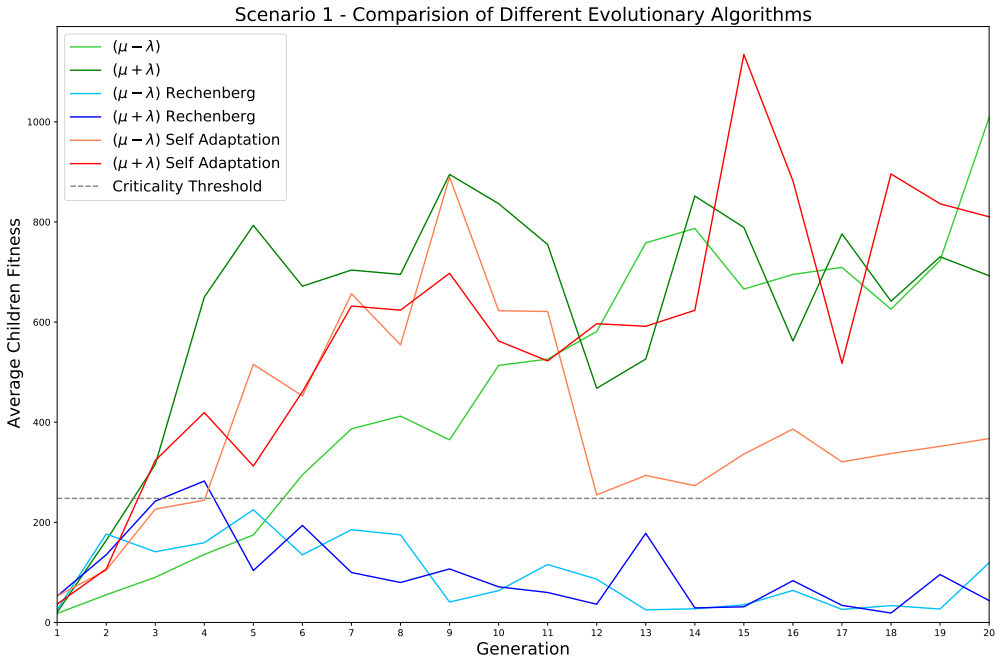


Figure D.2: Comparison of the average child fitness of the scenarios generated by the different algorithms over 20 generations in scenario 1. For all algorithms holds $\mu = 10$ and $\lambda = 50$.

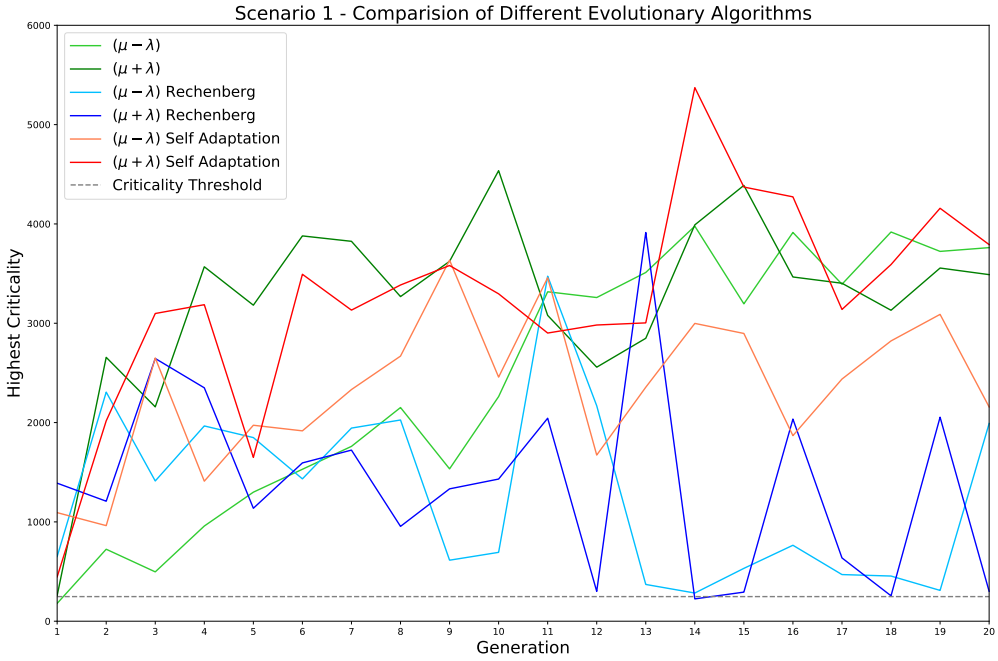


Figure D.3: Comparison of the highest criticality of a scenario generated by the different algorithms over 20 generations in scenario 1. For all algorithms holds $\mu = 10$ and $\lambda = 50$.

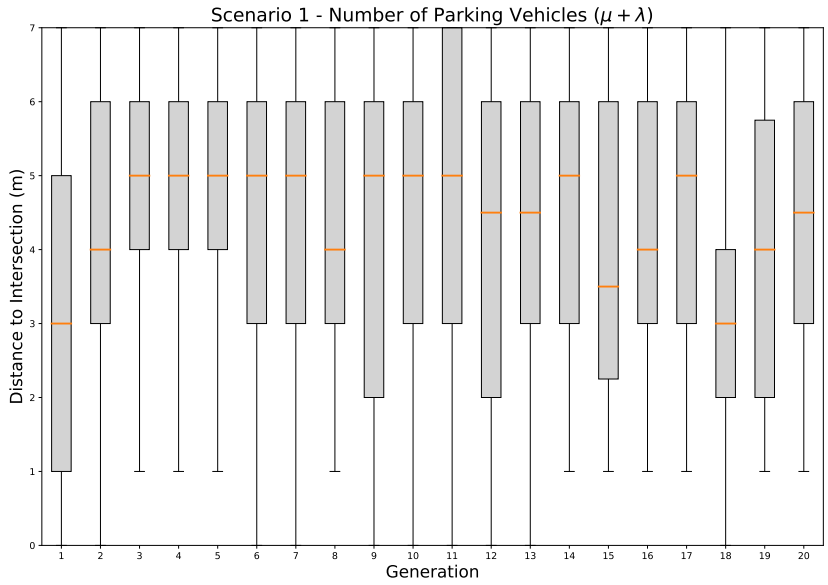


Figure D.4: Box plot that visualizes the evolution of the number of parking vehicles over 20 generations of the $(\mu + \lambda)$ algorithm in scenario 1.

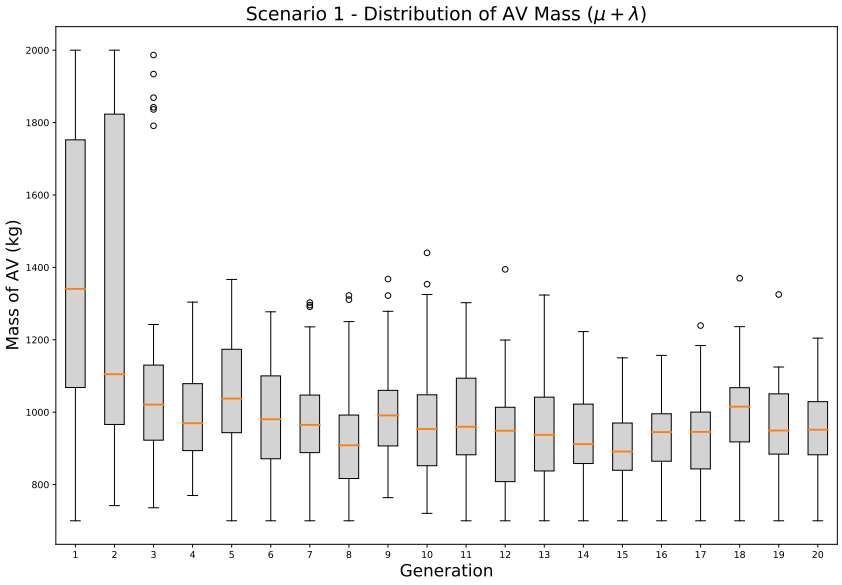


Figure D.5: Box plot that visualizes the evolution of the mass of the AV over 20 generations of the $(\mu + \lambda)$ algorithm in scenario 1.

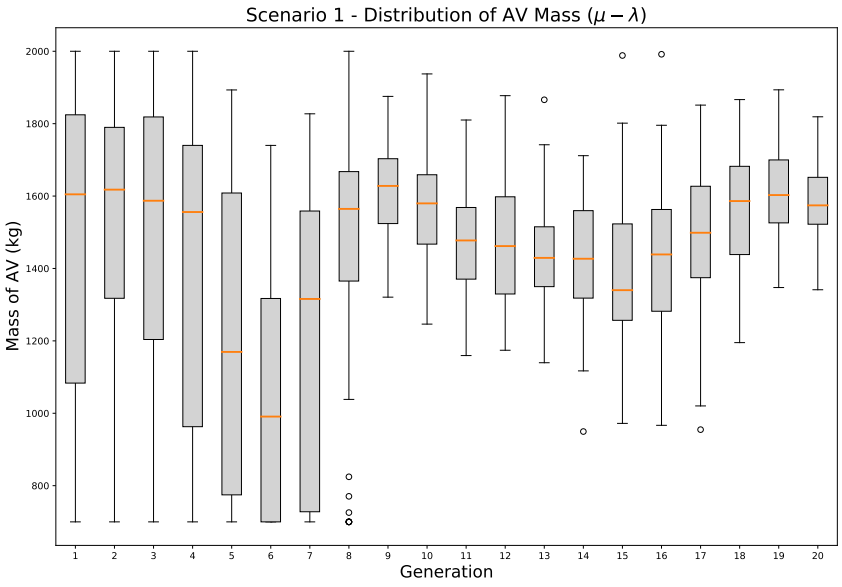


Figure D.6: Box plot that visualizes the evolution of the mass of the AV over 20 generations of the $(\mu\lambda)$ algorithm in scenario 1.

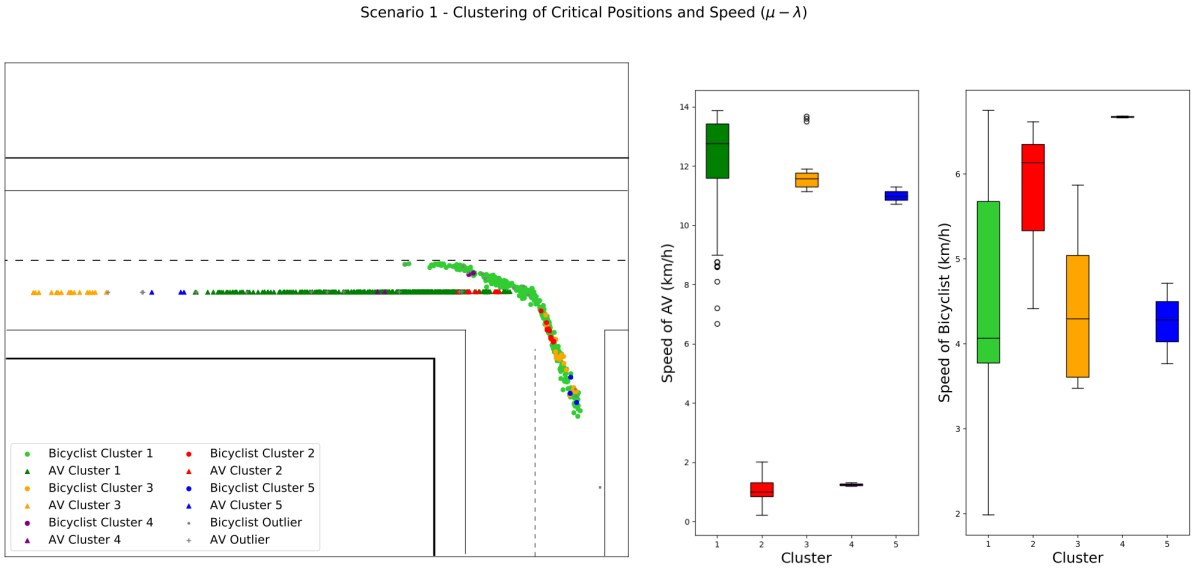


Figure D.7: Clustering of the positions and speed from the AV and the bicyclist in critical situations generated with the $(\mu - \lambda)$ algorithm in scenario 1.

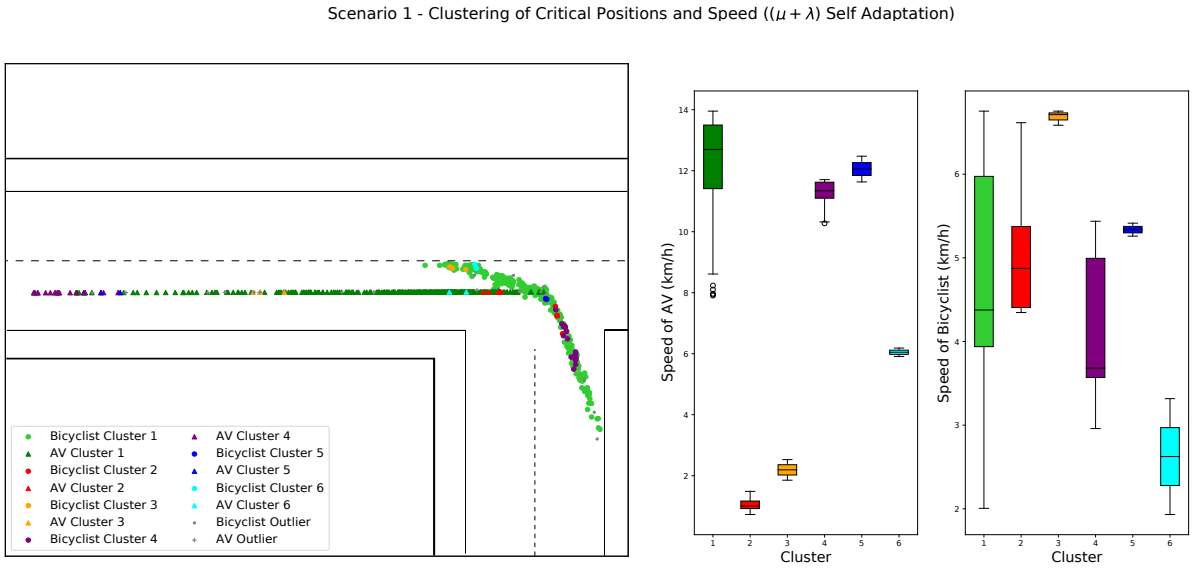


Figure D.8: Clustering of the positions and speed from the AV and the bicyclist in critical situations generated with the $(\mu+\lambda)$ algorithm with self adaptation in scenario 1.

D.2 EAs in Scenario 2

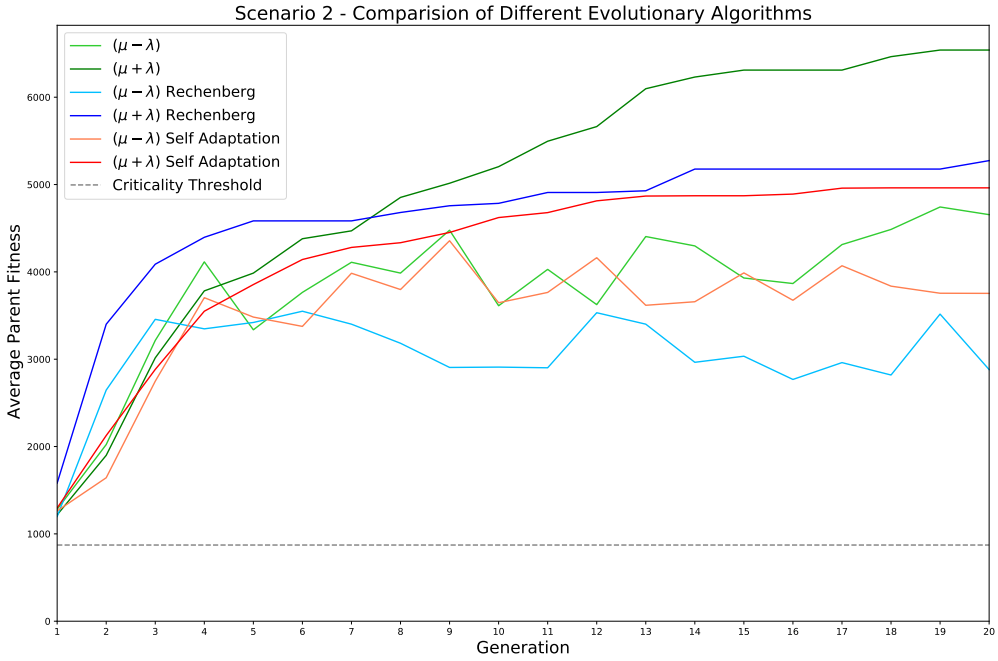


Figure D.9: Comparison of the average parent fitness of the scenarios generated by the different algorithms over 20 generations in scenario 2. For all algorithms holds $\mu = 10$ and $\lambda = 50$.

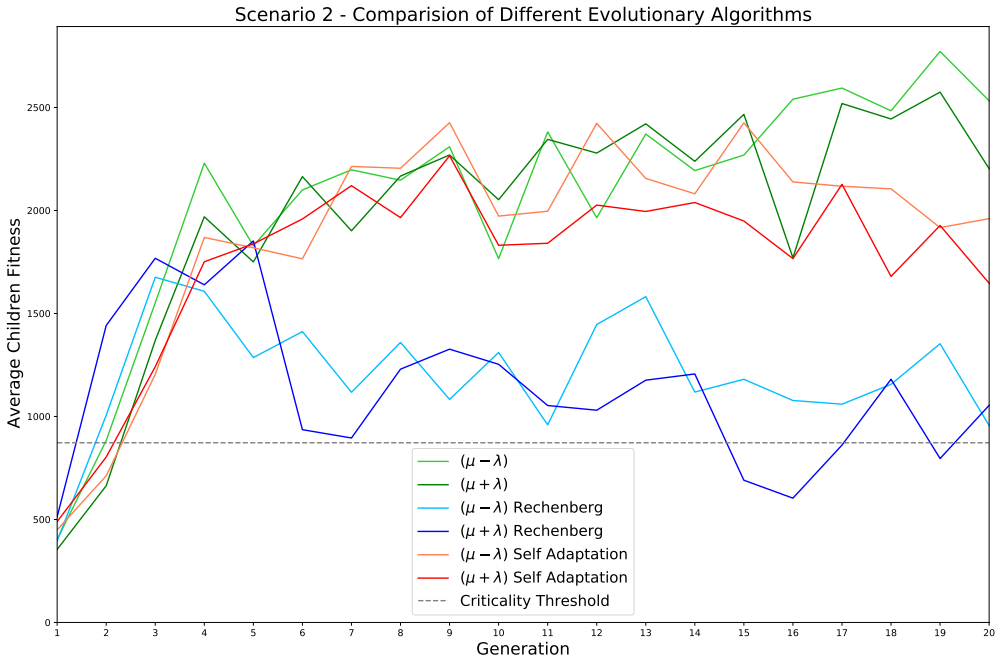


Figure D.10: Comparison of the average child fitness of the scenarios generated by the different algorithms over 20 generations in scenario 2. For all algorithms holds $\mu = 10$ and $\lambda = 50$.

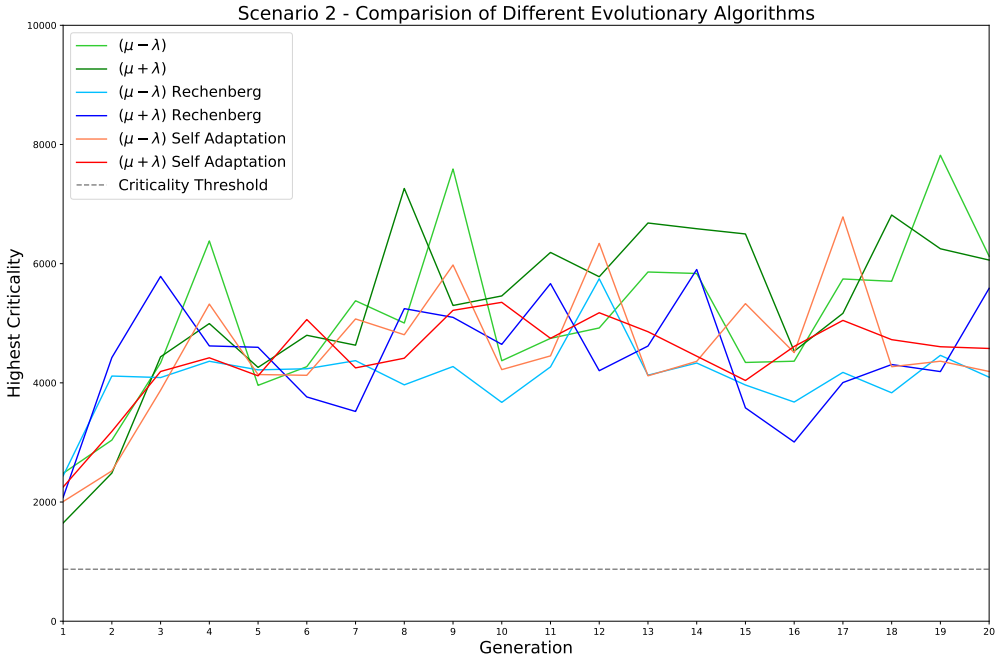


Figure D.11: Comparison of the highest criticality of a scenario generated by the different algorithms over 20 generations in scenario 2. For all algorithms holds $\mu = 10$ and $\lambda = 50$.

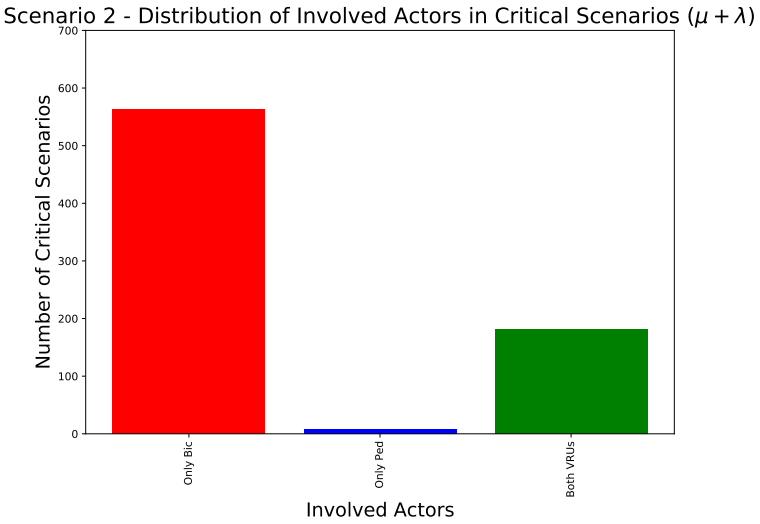


Figure D.12: Distribution of the involved actors in the most critical situation in *concrete scenarios* generated by the $(\mu + \lambda)$ algorithm in scenario 2.

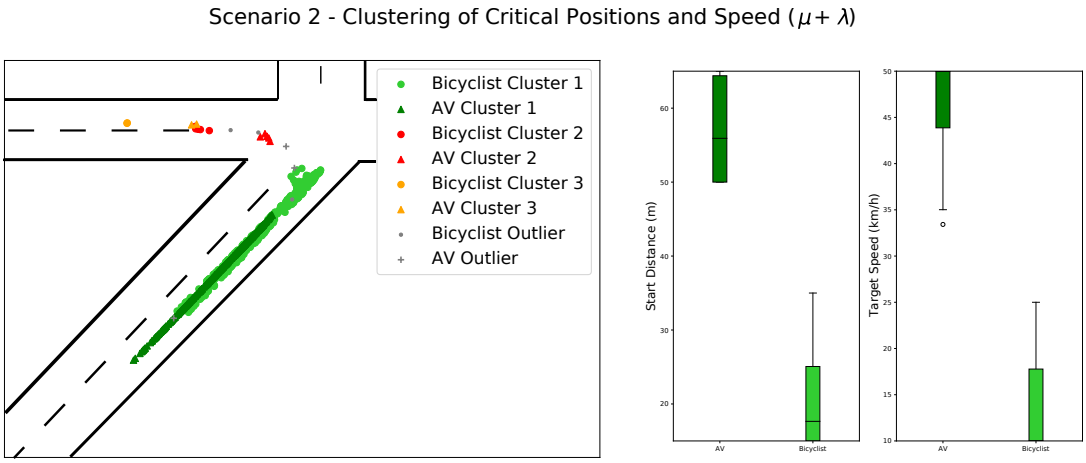


Figure D.13: Start position and target speed of the AV and the bicyclist in *concrete scenarios* that involve critical situations which are part of cluster 1.

D.3 EAs in Scenario 3

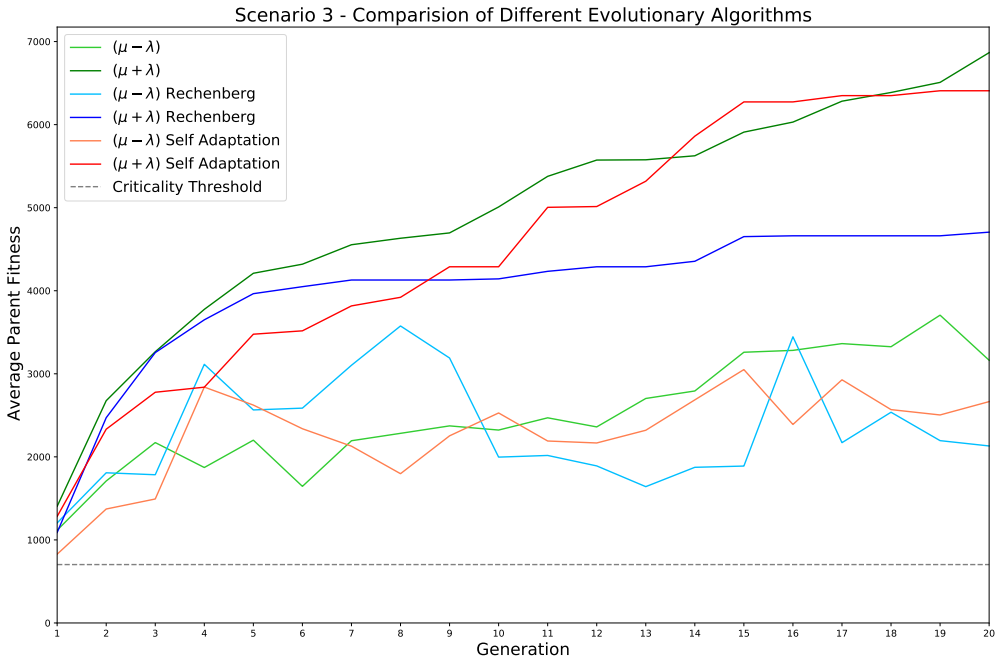


Figure D.14: Comparison of the average parent fitness of the scenarios generated by the different algorithms over 20 generations in scenario 3. For all algorithms holds $\mu = 10$ and $\lambda = 50$.

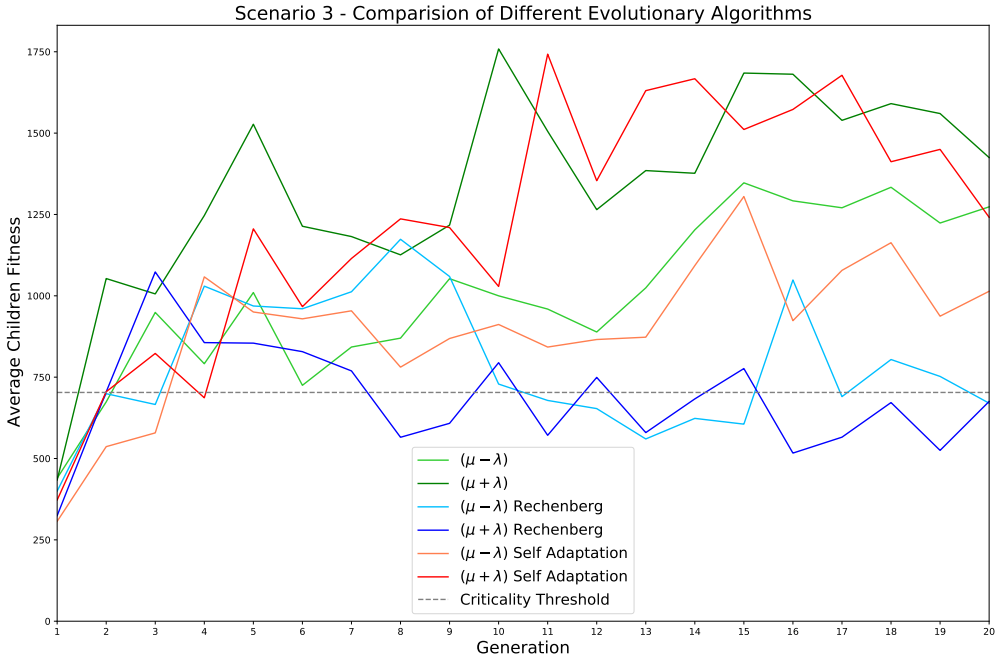


Figure D.15: Comparison of the average child fitness of the scenarios generated by the different algorithms over 20 generations in scenario 3. For all algorithms holds $\mu = 10$ and $\lambda = 50$.

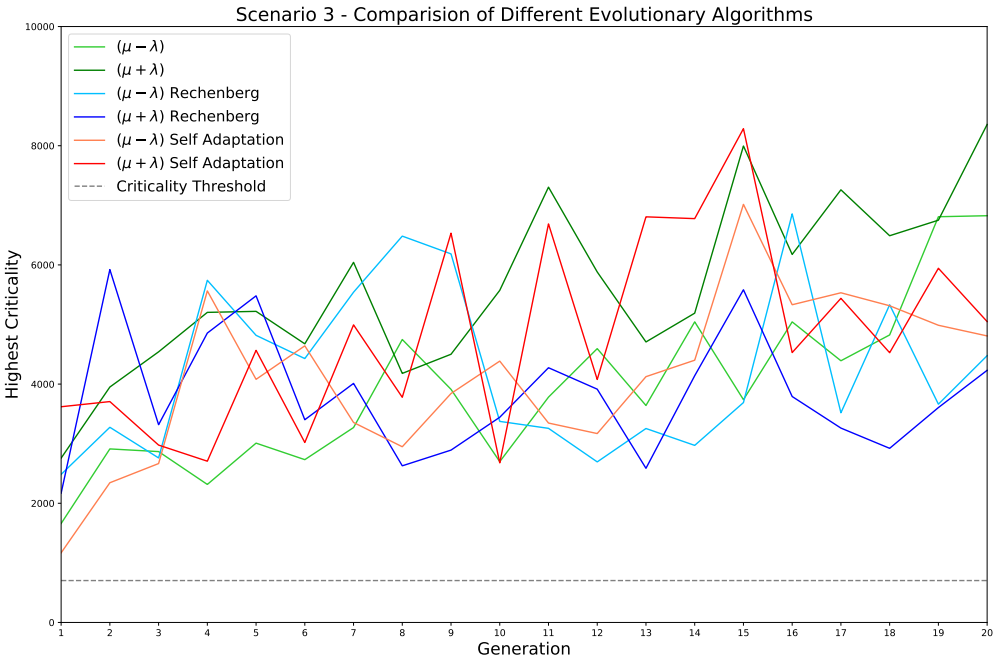


Figure D.16: Comparison of the highest criticality of a scenario generated by the different algorithms over 20 generations in scenario 3. For all algorithms holds $\mu = 10$ and $\lambda = 50$.

Eigenständigkeitserklärung

Hiermit versichere ich an Eides statt, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Oldenburg, den 20.05.2022