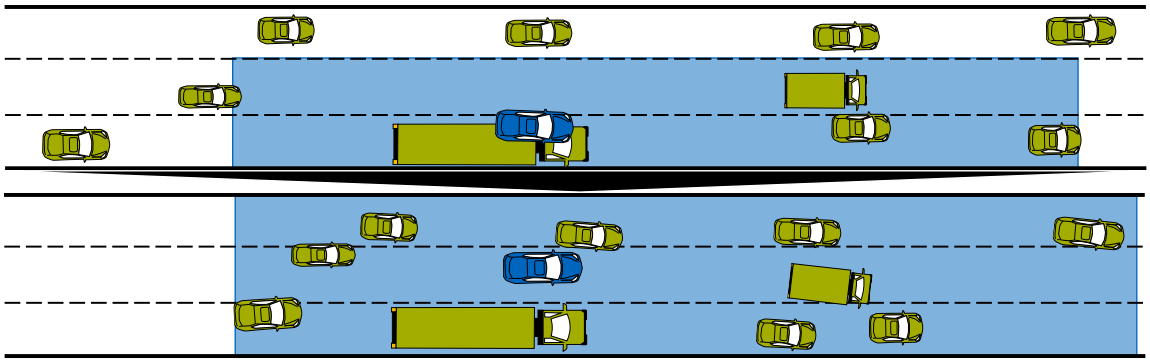
Creation of Complex Test Scenarios for Automated Vehicles by Means of Evolutionary Algorithm



Semester Thesis

at the Department of Mechanical Engineering of Technical University of Munich

Supervised by	Prof. Dr.-Ing. Markus Lienkamp Thomas Ponn, M.Sc. Chair of Automotive Technology
Submitted by	Yuanfei Lin Heiglhofstraße 66 81377 München
Submitted on	May 02, 2020

Chair of Automotive Technology
Department of Mechanical Engineering
Technical University of Munich

Project description

Creation of Complex Test Scenarios for Automated Vehicles by Means of Evolutionary Algorithm

Nowadays, the automotive industry is going through a period of continuous changes, which are mainly related to the need for electrification of the traditional means of transport and to the search for an unsupervised driving mode: autonomous Driving. In order to launch automated driving vehicles onto the market, it is necessary that they behave safely in a constant way. However, because theoretically infinitely many different situations can occur in real traffic, it is not possible to check the entire parameter space. Therefore, an attempt is made to filter out the most relevant and complex scenarios from the large number of possible scenarios.

In particular, in this work, the trajectories of the surrounding traffic are to be determined so that the complexity of the traffic situation becomes maximum. To solve this problem, different approaches can be followed as shown in [1], [2]. One powerful possibility to generate the most complex scenarios is the use of global optimization methods, specifically evolutionary algorithm.

The following points are to be investigated by Mr. Yuanfei Lin:

- Literature research and showing the state-of-the-art
- Evaluation of various optimization methods, use of a suitable optimization process, validation and discussion of the method
- Introduction of final objective function and generating results from real data
- Addition of constraints for acceleration/deceleration
- Documentation of the procedure and the results achieved

The elaboration should document the individual work steps in a clear form. The candidate undertakes to write the master thesis independently and to state the scientific sources used by him.

The submitted work remains the property of the chair as an examination document and may only be made accessible to third parties with the consent of the chair holder.

Announcement date: 02.11.2019

Submission date: 02.05.2020

Prof. Dr.-Ing. Markus Lienkamp

Thomas Ponn, M.Sc.

Chair of Automotive Technology
Department of Mechanical Engineering
Technical University of Munich

Geheimhaltungsverpflichtung

Herr: **Lin, Yuanfei**

Gegenstand der Geheimhaltungsverpflichtung sind alle mündlichen, schriftlichen und digitalen Informationen und Materialien die der Unterzeichner vom Lehrstuhl oder von Dritten im Rahmen seiner Tätigkeit am Lehrstuhl erhält. Dazu zählen vor allem Daten, Simulationswerkzeuge und Programmcode sowie Informationen zu Projekten, Prototypen und Produkten.

Der Unterzeichner verpflichtet sich, alle derartigen Informationen und Unterlagen, die ihm während seiner Tätigkeit am Lehrstuhl für Fahrzeugtechnik zugänglich werden, strikt vertraulich zu behandeln.

Er verpflichtet sich insbesondere:

- derartige Informationen betriebsintern zum Zwecke der Diskussion nur dann zu verwenden, wenn ein ihm erteilter Auftrag dies erfordert,
- keine derartigen Informationen ohne die vorherige schriftliche Zustimmung des Betreuers an Dritte weiterzuleiten,
- ohne Zustimmung eines Mitarbeiters keine Fotografien, Zeichnungen oder sonstige Darstellungen von Prototypen oder technischen Unterlagen hierzu anzufertigen,
- auf Anforderung des Lehrstuhls für Fahrzeugtechnik oder unaufgefordert spätestens bei seinem Ausscheiden aus dem Lehrstuhl für Fahrzeugtechnik alle Dokumente und Datenträger, die derartige Informationen enthalten, an den Lehrstuhl für Fahrzeugtechnik zurückzugeben.

Eine besondere Sorgfalt gilt im Umgang mit digitalen Daten:

- Für den Dateiaustausch dürfen keine Dienste verwendet werden, bei denen die Daten über einen Server im Ausland geleitet oder gespeichert werden (Es dürfen nur Dienste des LRZ genutzt werden (Lehrstuhllaufwerke, Sync&Share, GigaMove).
- Vertrauliche Informationen dürfen nur in verschlüsselter Form per E-Mail versendet werden.
- Nachrichten des geschäftlichen E-Mail Kontos, die vertrauliche Informationen enthalten, dürfen nicht an einen externen E-Mail Anbieter weitergeleitet werden.
- Die Kommunikation sollte nach Möglichkeit über die (my)TUM-Mailadresse erfolgen.

Die Verpflichtung zur Geheimhaltung endet nicht mit dem Ausscheiden aus dem Lehrstuhl für Fahrzeugtechnik, sondern bleibt 5 Jahre nach dem Zeitpunkt des Ausscheidens in vollem Umfang bestehen. Die eingereichte schriftliche Ausarbeitung darf der Unterzeichner nach Bekanntgabe der Note frei veröffentlichen.

Der Unterzeichner willigt ein, dass die Inhalte seiner Studienarbeit in darauf aufbauenden Studienarbeiten und Dissertationen mit der nötigen Kennzeichnung verwendet werden dürfen.

Datum: 02. Mai 2020

Unterschrift: _____

Chair of Automotive Technology
Department of Mechanical Engineering
Technical University of Munich

Erklärung

Ich versichere hiermit, dass ich die von mir eingereichte Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Garching, den 02. Mai 2020

Yuanfei Lin

Chair of Automotive Technology
Department of Mechanical Engineering
Technical University of Munich

Declaration of Consent, Open Source

Hereby I, Lin, Yuanfei, born on November 05, 1995, make the software I developed during my Semester Thesis available to the Institute of Automotive Technology under the terms of the license below.

Garching, May 02, 2020

Yuanfei Lin

Contents

List of Abbreviations

ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance Systems
AEB	Auto Emergency Braking
AV	Automated Vehicles
CAS	Collision Avoidance System
CDF	Cumulative Distribution Function
DE	Differential Evolution
EA	Evolutionary Algorithm
GA	Genetic Algorithm
GPS	Generalized Pattern Search
GSS	Generating Set Search
HighD	Dataset from German Highways
IDM	Intelligent Driver Model
LKA	Lane Keep Assist
MADS	Mesh Adaptive Direct Search
MOBIL	Minimizing Overall Braking Induced by Lane Changes
PS	Pattern Search
PSO	Particle Swarm Optimization
ROI	Region of Interest
SA	Simulated Annealing
SAE	Society of Automotive Engineers

Formula Symbols

Formula Symbols	Unit	Description
$x,\ x'$	m	x-coordinate
$y,\ y'$	m	y-coordinate
Pos_x	m	Longitudinal position of vehicle
Pos_x	m	Lateral position of vehicle
x_t	m	Position of particle
l_{tot}	m	Total length of lane change process
w_{cw}	m	Lane change width
O	m	Lateral deviation
$a,\ b,\ e$	m	Ellipse parameters
T_{tot}	s	Time horizon of one scenario
T_g	s	Time-gap
T_s	s	Time-step
T_{op}	s	Optimization duration
T_{sim}	s	Running time of one simulation
s_0	m	Linear jam distance
s_1	m	Non-linear jam distance
p_1	m	Individual optimal position
p_2	m	Global optimal position
d_{safe}	m	Safety distance
C_s	m	Safety coefficient
S_{ROI}	m^2	Area of ROI
v_x	m/s	Longitudinal velocity
v_y	m/s	Lateral velocity
v_v	m/s	Velocity of vehicle
v_t	m/s	Velocity of particle

v_0	m/s	Desired Velocity
a_x	m/s^2	Longitudinal acceleration
a_y	m/s^2	Lateral acceleration
j_x	m/s^3	Longitudinal jerk
j_y	m/s^3	Lateral jerk
$b_{x,com}$	m/s^2	Comfortable deceleration
ψ_c	°	Course angle
ϕ	°	Orientation
p	-	Politeness factor
δ	-	Acceleration exponent
δ_{bias}	-	Constant bias
δ_{th}	-	Switching threshold
$\boldsymbol{f}$	-	Normalized values of influence factors
$\boldsymbol{w}$	-	Weights of influence factors
w_0	-	Inertia weight
c_1	-	Cognitive learning factor
c_2	-	Social learning factor
PI	-	Performance indicator
nop	-	Number of parameters
nov	-	Number of variables
P	-	Penalty function
C	-	Complexity evaluation function
J	-	Objective function
η	-	Algorithm efficiency
N_{veh}	-	Number of vehicles
N_f	-	Number of frames
N_c	-	Number of complexity attribute
N_{GA}	-	Population size
N_{PSO}	-	Swarm size
$T_{g,GA}$, $T_{g,PSO}$	-	Number of generations
C_f	-	Crossover fraction
C_t	-	Crossover type
S_t	-	Selection type

E_r	-	Elite Ratio
Ini_{ss}	-	Initial swarm span
$\boldsymbol{A}_0$	-	Initial population/swarm matrix

1 Introduction

At present, the automotive industry is undergoing a significant change, mainly due to the large-scale introduction of electric and hybrid drives, shared mobility, individual mobility, connected vehicles, and autonomous driving [3]. In order to launch automated driving vehicles onto the market, it is necessary that they behave safely in a constant way. However, because theoretically infinitely many different situations can occur in real traffic, it is not possible to check the entire configuration space [4]. Therefore, an attempt is made to filter out the most relevant and complex scenarios from a large number of possibilities. In particular, in this thesis, the trajectories of the surrounding traffic are to be determined so that the complexity of the traffic situation around the vehicle under test (ego-vehicle) becomes maximum. One powerful possibility to generate the most complex scenarios is the use of global optimization methods, specifically Evolutionary Algorithm (EA).

1.1 Autonomous Driving

Autonomous driving has great potential nowadays, and it can change the transportation system by improving safety, comfort, and intelligence on the road. Society of Automotive Engineers (SAE) announced a visual chart for use with its J3016TM “Levels of Driving Automation” standard that defines six levels of driving automation, as shown in Figure 1.1, from no automation to full automation.

The safety benefits of Automated Vehicles (AV) from level 3 to 5 are paramount. To ensure safe driving for higher automation level, a great deal of technologies is developed to avoid collision and enhance driving comfort with advanced hardware and software. According to Weast [5] 2020 will be the year that Advanced Driver Assistance Systems (ADAS)-equipped vehicles are becoming the new normal. Customers will get more familiar with automated driving technology, which will hopefully lessen fear around AV of level 3-5. It is predicted that the percentage of AV on the road will reach 75 percent by 2040, with an estimated marketing value of $7 trillion by 2050 [6].

1.2 Goal and Structure of the Work

Only a few techniques were developed for the automatic test case generation of hybrid dynamics for the requirements of autonomous systems. The goal of this thesis is to develop a sufficient optimizer to generate highway scenarios for AV with high complexity, which also obeys the traffic rules and meets physical possibilities. Based on the previous work, final functions of complexity evaluation from Yu [8][9] and the clusters generated from Dataset from German Highways (HighD) by Breitfuss [10] will be included as the underlying inputs for the optimizer,

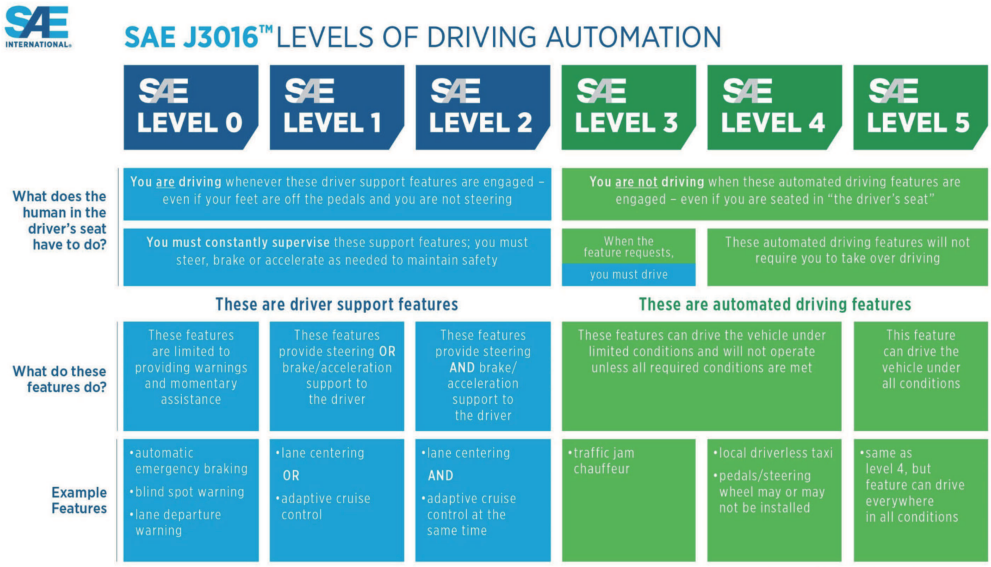


Figure 1.1: The levels of driving automation for on-road vehicles from SAE [7]

the simulation model of vehicles and optimization structure in [2] will be partially preserved and improved. The development of complex test scenarios is implemented in Matlab with its global optimization toolbox (Version 2019b).

The remainder of this thesis is organized as follows: After briefly introducing the complexity analysis method, optimization algorithms and HighD dataset in Chapter 2, Chapter 3 firstly explains the model used in simulator and constructs an interface with clusters generated from HighD, then the complexity evaluation and penalty function are integrated to an objective function, which is the last function used for optimization. Following it, a tuner for the optimizer is built up to achieve better optimization performance. Next, the results using different EA are listed and discussed in Chapter 4. In the last chapter, the shortcomings of the optimization method and possible improvements are summarized. The overview of how the present thesis approaches the topic is visualized in Figure 1.2.

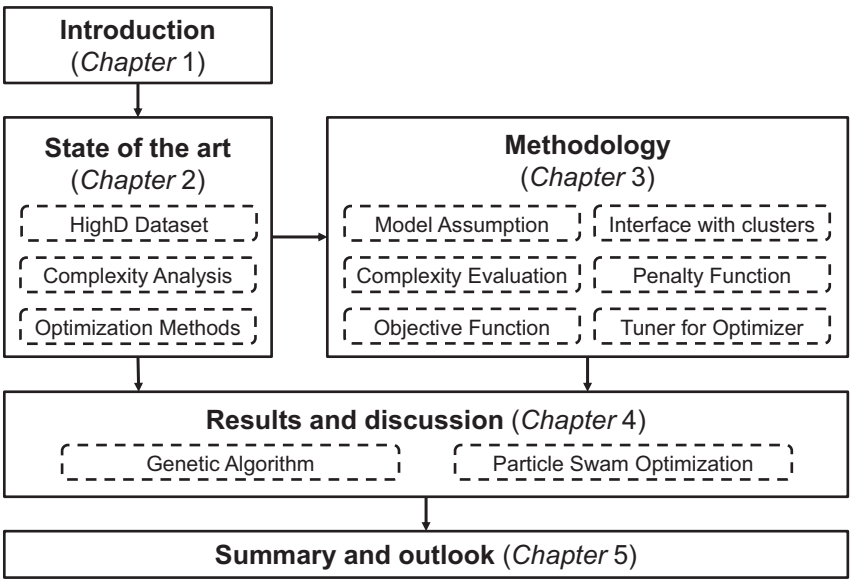


Figure 1.2: Work structure of this thesis

2 State of the art

In this section, the essential fundamental part of this thesis is presented. Firstly, the underlying data-set used in this thesis, named HighD, will be explained. Then the complexity evaluation of test scenarios in HighD for AV will be introduced. Finally, to enhance the complexity, a short introduction of optimization methods and their implementation with Matlab will be presented.

2.1 HighD Dataset

In our previous works, Villalobos [1] and Mayr [2] generate complex scenarios under simple configuration of vehicles and limited complexity metrics, since the information from background research is not enough. Thus it is necessary to look into the available dataset with more traffic information.

A large-scale naturalistic vehicle trajectory data-set from German highways called HighD [11] consists of measurements from six locations with 110000 vehicles and a total driven distance of 45000 km. The drone recorded the behaviour of vehicles passing a highway section of 420 m as shown in Figure 2.1. This information is sufficient to analyze highway scenarios and can be used as input to the optimizer.

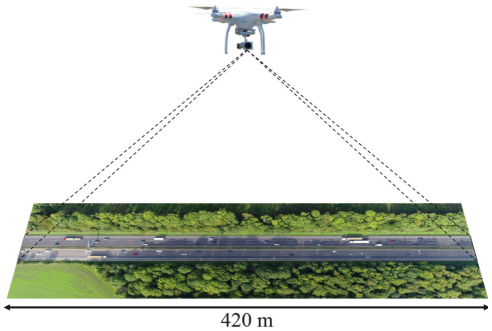


Figure 2.1: The recording setup of HighD [11]

Figure 2.2 depicts the global coordinate system of HighD with the upper left corner as origin, the horizontal axis to the right as the x-axis and the downward vertical axis as the y-axis.

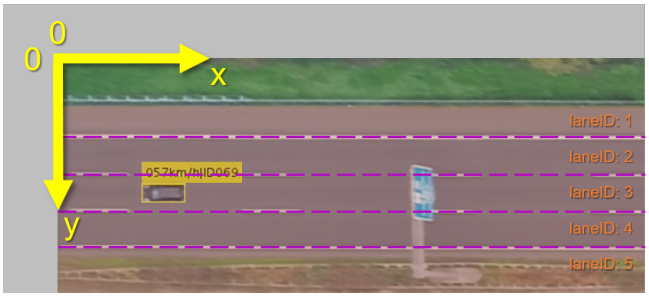


Figure 2.2: The global coordinate system of HighD[12]

Furthermore, some important information of coordinate and data structure transformation from [12] is listed in Table 2.1, which is the basis for the interface in next chapter.

Table 2.1: Important information of coordinate system in HighD

Name	Description
Frame Rate in Hz	The frame rate which was used to record the video discretely (25 Hz).
Vehicle Type	Vehicles tracked including cars and trucks.
Lane Markings	The y positions of the lane markings that separated by ";".
Width in m	The width of the post-processed bounding box (corresponding to the vehicle).
Length in m	The length of the post-processed bounding box (corresponding to the vehicle).
x in m	The x position of the upper left corner of the vehicle's bounding box.
y in m	The y position of the upper left corner of the vehicle's bounding box.
xVelocity in m/s	The longitudinal velocity in the image coordinate system.
yVelocity in m/s	The lateral velocity in the image coordinate system.
xAcceleration in m/s^2	The longitudinal acceleration in the image coordinate system.
yAcceleration in m/s^2	The lateral acceleration in the image coordinate system.

2.2 Complexity Analysis

In the work of Breitfuss [10], the real data from HighD is extracted to various clusters. Based on the processed data, Yu [8][9] develops a method to evaluate the traffic scenarios of HighD, which can be adopted as adequate metrics of the complexity in highway.

Complexity is what characterizes all evolved, open systems, where the structure and organization have emerged over time through processes of self-organization [13]. Focusing on the problem of road complexity, the adversity of the situation usually depends on the environment and the behaviours of traffic participants. To organize all information in one scenario, we propose to use an adapted layered model for scene representation from Bagschik et al. [14] based on the work of Schuldt [15] as shown in Figure 2.3.

The fourth layer (L4) lists all stationary and moving objects that do not belong to the infrastructure (L2). Besides, the manoeuvres, as well as the interactions of the objects in the fourth layer, are defined. Because of the diversity of sensors and infrastructure in real traffic, it is difficult to unify their impact on the complexity analysis. Therefore, the method for quantitative evaluation of traffic complexity proposed in [8][9] only focus on the fourth layer.

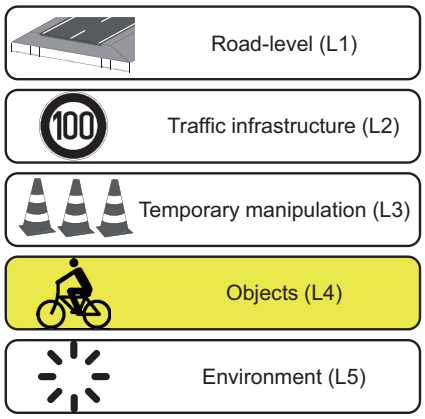


Figure 2.3: Layer model for the representation of driving scenes adapted from [16]

2.2.1 Definition of ROI

In one scenario, there are dozens of vehicles appearing, and it is impossible to involve all of them in the complexity analysis. In addition, the influence caused by the vehicles beyond a certain distance is negligible, thus a concept called Region of Interest (ROI) is introduced.

Since a safety distance d_{safe} of one vehicle is often defined as half of the current velocity (km/h) and the behaviours of vehicles ahead impact more on the decision-making of the ego-vehicle, the ROI in the longitudinal direction is defined as the area of two safety distance forwards and one backwards. In the lateral direction, it is intuitive to consider the currently located and adjacent lanes, as shown in Figure 2.4. In the following analysis of complexity, only the vehicles within ROI will be considered.

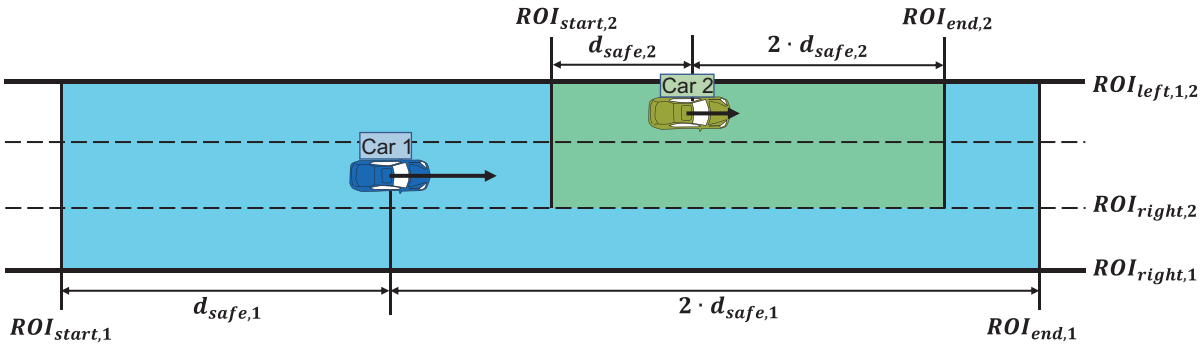


Figure 2.4: Demonstration of ROI adapted from [8] (Blue part: ROI of Car 1 driving in the middle lane; green part: ROI of Car 2 in the edge lane, only area of two lanes are considered in the lateral direction)

2.2.2 Classification of Surrounding Traffic

The surrounding traffic should be classified according to its position or behaviour before complexity evaluation, to make the observation more convenient and distinguish the influence degree of different positions in the ROI of ego-vehicle.

Since the area in front of ego-vehicle plays a more influential role than the area behind, the 8-nearest-neighbour model that introduced by Wang et al. [17] is extended to 11-nearest-neighbour according to the work of Antona-makoshi et al. [18]. Correspondingly the ROI can be divided into

4 longitudinal sections according their hierarchy of effects, which is presented by their relative position to ego-vehicle in ROI. Five dividing lines are defined in Eq. (2.1). To illustrate the hierarchy of effects of each section, 1_{st} section(x_{s_1} x_{s_0}) is called "leading", 2_{nd} (x_{s_2} x_{s_1}) "pre", 3_{rd} (x_{s_3} x_{s_2}) "along", 4_{th} (x_{s_4} x_{s_3}) "follow".

$$\begin{cases} x_{s_0} = ROI_{end};\ x_{s_4} = ROI_{start}; \\ x_{s_3} = x_{ego} - \frac{1}{2} \cdot width_{ego} - 5m; \\ x_{s_2} = x_{ego} + \frac{1}{2} \cdot width_{ego} + 5m; \\ x_{s_1} = \frac{x_{s_0} + x_{s_2}}{2}; \end{cases} \tag{2.1}$$

Combining the assignment in lateral direction according to their relative lane position ("left", "middle", "right") 12 sectors is determined and a 4x3 matrix is defined indicating the occupation state of surrounding vehicles with labels. The finial label assignment of surrounding traffic is shown in Figure 2.5.

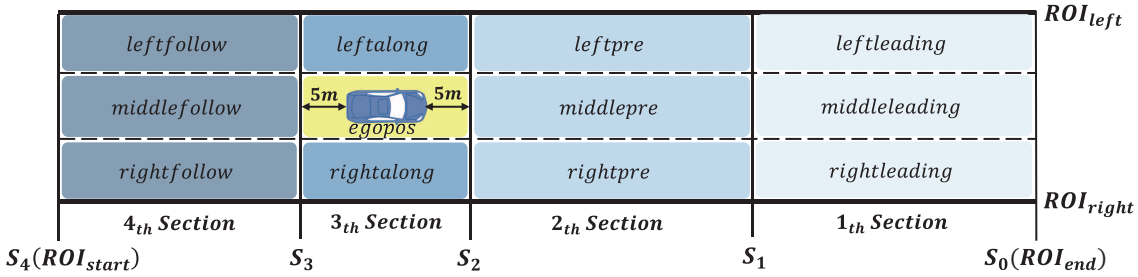


Figure 2.5: Label assignment of surrounding traffic within ROI modified from [8]

2.2.3 Influence Factors

There exist aggregately 13 influence factors for complexity evaluation in [9]. During the simulation, the complexity of each frame will be calculated, and all the influence factors have equal weights, which sums to 1. For the final complexity evaluation of the whole scenario, the complexity of all frames will be averaged as Eq. (2.2):

$$C = \frac{T_s}{T_{tot}} \sum_{j=1}^{T_{tot}/T_s} \boldsymbol{f}_j^T \cdot \boldsymbol{w} \tag{2.2}$$

with

$$\begin{cases} \boldsymbol{w} = [w_1,\ w_2,\ ...,\ w_{N_c}]; \\ w_1 = w_2 = ... = w_{N_c};\ \sum_{i}^{N_c} w_i = 1 \end{cases} \tag{2.3}$$

where T_{tot} presents the duration of the whole scenario, T_s the discrete time-step, N_c the number of the complexity attributes, vector $\boldsymbol{f}$ the normalized values of influence factors and $\boldsymbol{w}$ the corresponding weight vector.

Within the ROI of the ego-vehicle, we can summarize the influence factors as shown in Table 2.2 to evaluate the complexity of the highway traffic according to the definition in [9].

Table 2.2: List of influence factors for complexity evaluation in [9]

Nr.	Description	Domain	Symbol
1	Number of surrounding traffic	nb_num	
2	Types of surrounding traffic	nb_type	
3	Number and type of actions by surrounding traffic	noa_nb	
4	Possible actions of surrounding traffic	pa_nb	Surrounding
5	Variation of action parameters of surrounding traffic	$variation$	Traffic
6	Dynamic of surrounding traffic	$dynamic$	
7	Interactions between traffic participants	$connection$	
8	Deviation from predicted state	$devi\_eu$	
9	Number and type of actions required to be performed	noa_ego	Ego
10	Possible actions of ego-vehicle	pa_ego	Vehicle
11	Occluded area (in percentage) within ROI	$ratio$	Sense/Plan
12	Time-gap	tg	Criticality
13	Time-to-break	$t2b$	

To combine the different influence factors, normalization of these values as post-processing is necessary. Using the results from [9] the original normalization values are shown in Table 2.3, where pa denotes the possible actions of ego-vehicle or surrounding vehicles, BS the blind spot which is not visible to the driver, and S_{ROI} the area of the entire ROI. For instance, according to Figure 2.5, when each sector is occupied by exactly one vehicle, the value of nb_num is now assigned with 11. Then the nb_num_{norm} is equal to 1 after normalization , and the f_{nb_num} in this situation is determined correspondingly.

Table 2.3: Normalization values of influence factors [9]

Factor	Value	Factor	Value
nb_num	11	nb_type	2
noa_nb	10	pa_nb	8
$variation$	$\begin{cases} v_x\text{: 15 } m/s,\ v_y\text{: 1.3 } m/s \\ a_x\text{: 1.5 } m/s^2,\ a_y\text{: 0.5 } m/s^2 \end{cases}$	$dynamic$	$\begin{cases} v_x\text{: 35 } m/s,\ v_y\text{: 0.65 } m/s \\ a_x\text{: 0.65 } m/s^2,\ a_y\text{: 0.22 } m/s^2 \end{cases}$
$connection$	17	$devi\_eu$	1.4
noa_ego	20	$ratio$	$\sum BS/S_{ROI}$
pa_ego	$\begin{cases} 3.25 \cdot pa + 0.5\text{, if } pa\text{<2;} \\ 7\text{, if } 2 \le pa \le 3\text{;} \\ -3.25 \cdot pa + 16.75\text{, otherwise} \end{cases}$	$t2b$	$\begin{cases} t2b\text{, if } t2b\text{<0;} \\ 2\text{, if } 0 \le t2b \le 2\text{;} \\ \text{Inf, otherwise} \end{cases}$
tg	$e^{-0.5 \cdot tg}$		

2.3 Optimization Methods

After obtaining the initial data from HighD and the metrics of complexity evaluation, the optimization structure can be established. An optimization problem is to find the best solution from all feasible solutions in a certain time. In this section, the algorithms of basic optimization methods are described and a combination with the commands in Matlab will be made to implement the optimization algorithm.

2.3.1 Background Knowledge

First, some background knowledge of optimization algorithm and its implementation tools need to be introduced.

As an effective optimization tool, EA uses simulated evolution to explore the solutions for complex real world problems, which becomes very prevalent tool for searching, optimization and providing solutions to complex problems [19]. The most popular EA include Genetic Algorithm (GA), Particle Swarm Optimization (PSO). In the work of Klischat et al. [20], an approach of utilizing EA (Differential Evolution (DE) and PSO) to tackle the resulting highly critical test scenarios for AV is presented, which uses the drivable area as a measure for criticality and shows a sufficient result. Mayr [2] compares the performance of different optimization algorithms with Matlab and uses PSO to generate complex scenarios under simple configurations.

There are many tools to implement optimization algorithms, such as Matlab, Python, and etc. Global Optimization Toolbox in Matlab provides functions that search for global solutions to problems that contain multiple maximum or minimum [21, p.1-2], which is useful for the optimization problem to find the most complex scenario under certain circumstances. Toolbox solvers such as EA (GA, PSO included), Simulated Annealing (SA), Pattern Search (PS) can help us to generate different kinds of optimization algorithms.

2.3.2 Genetic Algorithm

Genetic Algorithm (GA) is a method for solving both constrained and unconstrained optimization problems based on a natural selection process that mimics biological evolution [21, p.5-2]. GA is a sub-class of EA, which can be viewed as a search procedure that generates potential solutions to a problem, tests each for suitability and then generates new solutions [22]. The basic procedure of GA is shown in Algorithm 1.

Algorithm 1 Genetic Algorithm

```
1: procedure GA
2:     set ending condition                          ▷ max time, max generation ect.
3:     import initial population
4:     while ending condition not satisfied do
5:         evaluate population    ▷ assign score to each individual with the objective function
6:         select parents                                   ▷ based on selection rules
7:         generate offspring     ▷ based on reproduction options (crossover, mutation)
8:     return best individual                          ▷ with minimum or maximum value
```

It can be found in Algorithm 1 that, there exist three main types of rules to obtain the new generation, that is, selection rules, crossover rules and mutation rules, which will mainly affect the optimization performance.

- **selection rules:** The individuals that contribute to the population for the next generation are selected, which can be called parents. Among them, some of the original ones will remain, and some newly created individuals will also be saved according to different rules. Different selection types [21, p.11.31-32] are explained in Table 2.4.

Table 2.4: The options of selection type

Name	Description
Stochastic uniform	A single random value is uniformly used to select the position of parents by choosing at evenly spaced intervals.
Roulette	The parents are selected by simulating roulette, and the fitness value of each individual in a population corresponds to the area of the roulette wheel proportion.
Remainder	Firstly, the size of individuals are scaled to be suitable for the selection function. Then the parents are assigned deterministically from the integer part of the scaled value, and the roulette selection is used for the remaining fractional part.
Tournament	It involves running several "tournaments" among the individuals, which is chosen at random from the population. The winner of each tournament (the one with the best fitness) is selected as parents.

- **crossover rules:** Crossover stands for the recombination operator, which combine two parents to form offspring iteratively. Table 2.5 shows the varied functions supplied by Matlab [21, p.11.35-36].

Table 2.5: The options of crossover type

Name	Description
Scattered	A random binary vector such as [1 0 0 ... 1] is used. The position with a value of 1 will be selected from the first parent, with a value of 0 from the second, the offspring is then formed.
Single point	A single point is chosen randomly, and parts to the right of that point are swapped between the two parents.
Two-point	Two crossover points are chosen randomly and the parts within the two points are swapped between the two parents.
Intermediate	Offspring is created by taking a weighted average of the parents with parameter $Ratio$ using the following formula, which can be a scalar or row vector. $$child = parent_1 + rand * Ratio * (parent_2 - parent_1) \quad (2.4)$$
Heuristic	Offspring lies on the line containing the two parents and have a smaller deviation from the parent with the better fitness value (e.g. $parent_1$) according to the Eq. (2.5). $$child = parent_2 + Ratio * (parent_1 - parent_2); \quad (2.5)$$

- **mutation rules:** Mutation applies random changes to individual parent and then derives the new offspring. The different ways that GA make small random changes in the individuals in the population to create mutation offspring are shown in Table 2.6 [21, p.11.33-35].

Besides using mutation and crossover, a certain number of individuals are guaranteed to survive to the next generation, which is called $elite$ and whose size should be less than or equal to the population size. In the reproduction option of Matlab GA toolbox, the scale for crossover, and mutation in one generation can be determined by the value of crossover fraction C_f. Based on

Table 2.6: The options of mutation function

Name	Description
Gaussian	A random number taken from a normal Gaussian distribution is added to each entry of the parent vector. The standard deviation of this distribution is determined by the parameters $scale$ and $shrink$. $Scale$ determines the standard deviation at the first generation, while $shrink$ controls how the standard deviation shrinks as generations go by. The $shrink$ is usually assigned with 1, which means the amount of mutation will decrease to 0 at the final step.
Uniform	Firstly a fraction of the vector entries of an individual for mutation is selected. Then the algorithm replaces each selected entry by a random number selected uniformly from the range for that entry.
Adaptive Feasible	The mutation chooses a direction and step length that satisfes bounds and linear constraints with respect to the last successful or unsuccessful generation.

this, the general flow chart of GA is described in Figure 2.6, which also explained the key points for improving its performance.

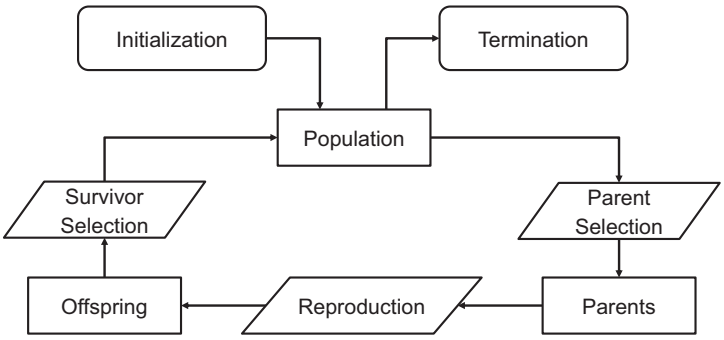


Figure 2.6: General flow chart of GA

In addition, the typical syntaxes of GA in Matlab [21, p.12-7] are:

```
1 [val,fval] = ga(fun,nvars)
2 [val,fval] = ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon,options)
```

which consists of input arguments ($@fun$: fitness function, $nvars$: the number of independent variables, $A \cdot val \leq b$: linear inequalities, $A_{eq} \cdot val = b_{eq}$: linear equalities, $lb \leq val \leq ub$: possible range between lower bound lb and upper bound ub, $nonlcon$: nonlinear constraints, $options$:optimization options) and output variables of GA (val: the final individual, $fval$: the value of the objective function for val). If the above-mentioned input argument is empty, it can be replaced by `[]`.

2.3.3 Particle Swarm Optimization

Particle swarm optimization (PSO) is also a population-based stochastic optimization algorithm, which is motivated by the intelligent collective behaviour of some animals such as flocks of birds or schools of fish [23]. Similar to the population in GA, particle (single individual) information iterates in each generation to adjust the direction of the searching process.

Algorithm 2 Particle Swarm Optimization

```
1: procedure PSO
2:     set ending condition                          ▷ max time, max iteration ect.
3:     initialize particle swarm
4:     while ending condition not satisfied do
5:         evaluate particle fitness                 ▷ assign score with the objective function
6:         calculate the individual historical optimal position (pbest)
7:         calculate the swarm historical optimal position (gbest)
8:         update particle velocity and position     ▷ according to Eq. (2.6)
9:     return best particle                          ▷ with minimum or maximum value
```

By default, PSO creates particles at random within bounds initially, which also called the swarm. To control the span of the initial swarm, the "InitiaSwarmSpan" option in Matlab can be varied. Next, the moving of particles afterwards depends on the velocity vector. For the local particle, the velocity and position update equations are derived as shown in Eq. (2.6) [23].

$$\begin{cases} v_{t+1}^i = w_0 \cdot v_t^i + c_1 \cdot rand \cdot (p_t^i - x_t^i) + c_2 \cdot rand \cdot (p_t^g - x_t^i) \\ x_{t+1}^i = x_t^i + v_{t+1}^i \end{cases} \tag{2.6}$$

where the position and velocity of the particle i at time t can be denoted by x_t^i and v_t^i, w_0 is called inertia weight, c_1 self adjustment weight, c_2 social adjustment weight, p_i the individual optimal position (pbest), p_g the optimal position in the local neighbourhood (gbest), and $rand$ returns a random number between 0 and 1.

To explain the above equations, Figure 2.7 depicts the update scheme of particle velocity and position. Each particle's new velocity v_{t+1}^i is a function of its current velocity v_t^i, the vector that points to the particle's own best location that saved in the memory $(p_t^i - x_t^i)$ and the vector that points to the best location of the particles inside the swarm $(p_t^g - x_t^i)$. Then the next position x_{t+1}^i is determined.

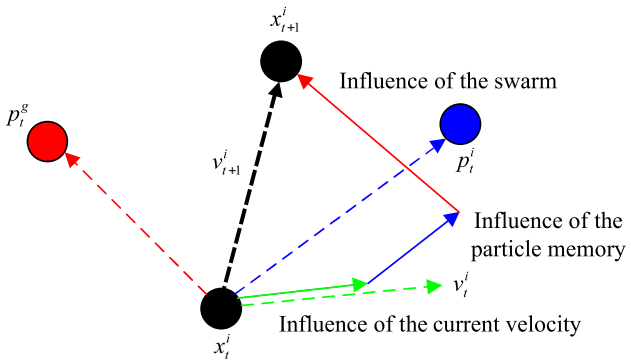


Figure 2.7: Position and velocity update of particles in PSO [23]

Thus the whole working flow of PSO can be summarized in Figure 2.8. Each particle continually adjusts its speed and trajectory in the search space based on the aforementioned information, then moves closer towards the global optimum with each iteration.

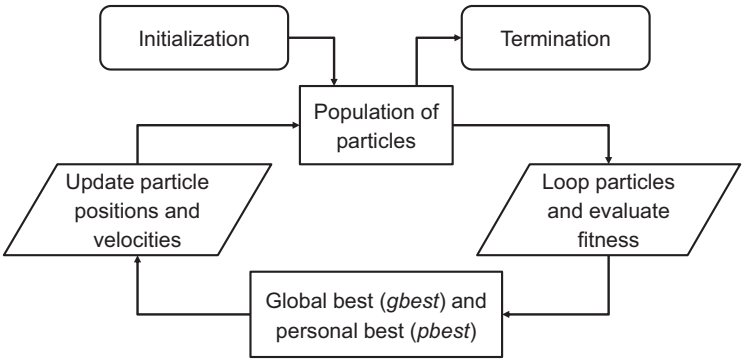


Figure 2.8: General flow chart of PSO

The following syntax of PSO in Matlab attempts to find a vector val that achieves a local minimum of $@func$ [21, p.12-116]. The input and output arguments are similar to the ones defined in GA. Notably, the PSO toolbox in Matlab does not support the variables with constraints.

```
1 [val,fval] = particleswarm(fun,nvars)
2 [val,fval] = particleswarm(fun,nvars,lb,ub,options)
```

2.3.4 Simulated Annealing

Simulated Annealing (SA) is an effective and useful algorithm in solving unconstrained and bound-constrained optimization problems [21, p.8-2]. The concept “annealing” is derived from thermodynamics, which specifies the way that metals cool and anneal [24]. SA uses the objective function of an optimization problem instead of the energy of a material.

From [25] the algorithm of SA can be derived as shown in Algorithm 3, where s defines the state space, E the energy (goal) function, P the acceptance probability function, $neighbour$ the candidate generator procedure and $temperature$ the annealing schedule temperature, which should be specified before each optimization process.

Algorithm 3 Simulated Annealing

1: **procedure** SA
2: $k = 0$ ▷ k is the number of iteration
3: $s = s_0$ ▷ initial state
4: **while** $k! = k_{max}$ **do**
5: $T = temperature((k+1)/(k_{max}))$
6: choose a random neighbour s_{new} ▷ $neighbour(s)$
7: **if** the acceptance probability function P satisfied **then**
8: $s = s_{new}$ ▷ $P(E(s), E(s_new), T) \geq rand$
9: $k = k + 1$
10: **return** the final state s

Figure 2.9 shows the convergence process of SA. The initial temperature must be large enough to generate the same probabilities for the transitions in different directions, i.e., moving uphill and downhill. Detailed introduction and Matlab command can be viewed in [21, p.12-127].

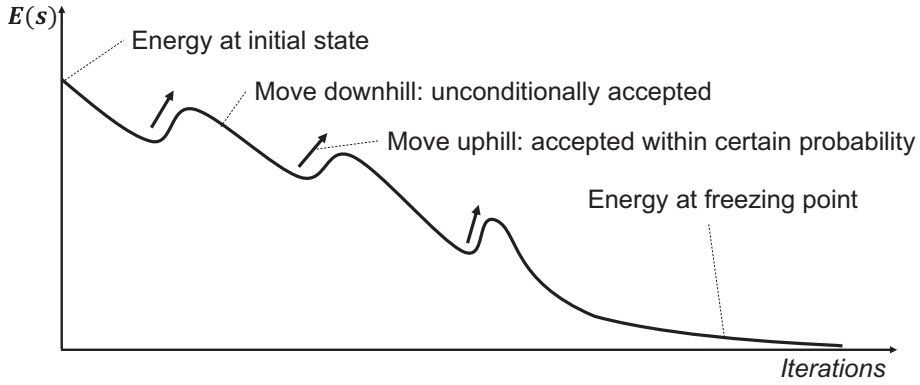


Figure 2.9: Convergence process of SA

2.3.5 Pattern Search

Pattern Search (PS) generates and maintains multi-dimensional search directions in a dynamic manner [26] without requiring any information about the gradient of the objective function, which is also known as derivative-free, direct search or black-box optimization method.

The procedure of pattern search updated from [27] is outlined in Algorithm 4, where Δ denotes the search step, P_k the generating matrix, p_k a column of P_k, and Ω the neighbourhood of the current solution.

Algorithm 4 Pattern Search

```
1:  procedure PS
2:      initialize Δ_0                          ▷ default search step
3:      initialize p_0                          ▷ initial solution
4:      Δ = Δ_0
5:      while ending condition not satisfied do
6:          Ω = {p_0 + Δ · p_k}                 ▷ for each column of P_k
7:          evaluate the nearest neighbours in Ω
8:          if improvement exists then
9:              update p to the best neighbour
10:             Δ = Δ
11:         else
12:             Δ = Δ/2
13:     return the final solution p
```

Generalized Pattern Search (GPS), Generating Set Search (GSS) and Mesh Adaptive Direct Search (MADS) are widely used in PS [28, p.1250-1251] to control how the searching process polls the mesh points at each iteration, which will highly affect the optimization performance. Specific examples and implementation in Matlab are introduced in more detail in [21, p.4-19].

3 Methodology

In this chapter, all optimizer-related interfaces, models, and functions will be introduced, which plays an important role in building the entire optimization system and improving its performance. Figure 3.1 depicts the flow chart of the whole optimization process. In section 3.1, the model assumption of vehicles in the simulator will be introduced. The following sections will explain the interface with clusters, the structure of objective function (complexity evaluation and penalty function), and the tuner of the optimizer.

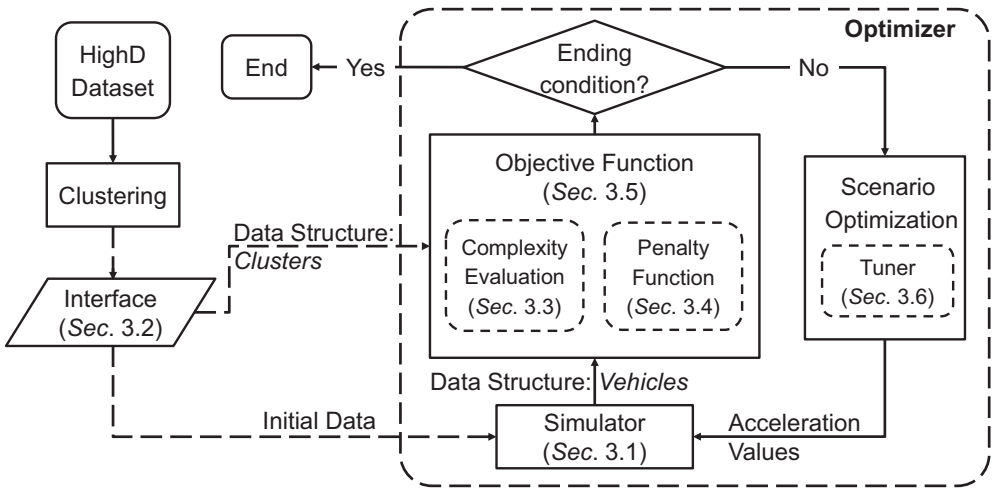


Figure 3.1: Flow chart of the whole optimization process

3.1 Model Assumption

In this section the model assumptions used in the simulator will be introduced, i.e., the dynamic models for ego-vehicle and surrounding traffic. In the general vehicle model, any brusque press on the throttle or the brake pedal would correspond to a discontinuity of the acceleration input [29]. Therefore, it is assumed that the control inputs of vehicles are piece-wise continuous functions with finite limits at the discontinuity points, which can be denoted by $\mathscr{D}^0$ as the examples in Figure 3.2. On account of the observation limitation in HighD, the input of optimizer is further considered as a combination of piece-wise constant functions, which is a special case of $\mathscr{D}^0$. Furthermore, the velocity and position of the vehicles commanded in acceleration in the simulator will be correspondingly continuous and differentiable. Thus the acceleration values as input are admissible for the simulator.

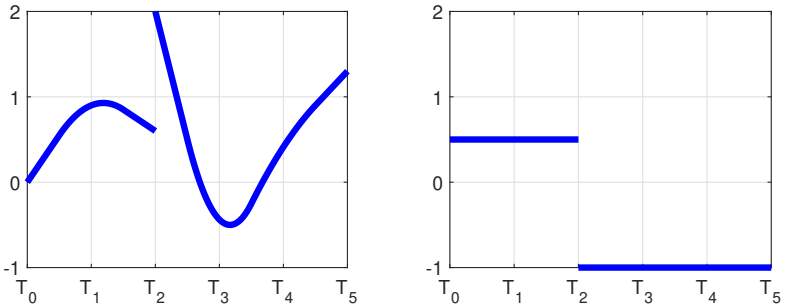


Figure 3.2: Examples of $\mathscr{D}^0$ function (left: piece-wise continuous function, right: piece-wise constant function)

3.1.1 Ego-Vehicle

The dynamics of ego-vehicle can be considered separately along the x- and y-direction, i.e., in longitudinal and lateral direction. The longitudinal dynamic of ego-vehicle is simulated by Intelligent Driver Model (IDM), which is a simple time-continuous car-following model for the simulation of highway traffic. Then a lane change model with Minimizing Overall Braking Induced by Lane Changes (MOBIL) rules is applied, and the multi-lane traffic is correspondingly simulated in combination with IDM.

Longitudinal Dynamic: IDM is used for the deterministic modeling of longitudinal motions, which is based on Adaptive Cruise Control (ACC) system and suitable to describe the longitudinal dynamics for AV approximately [30].

To reach the desired speed in a accident-free manner, the optimal acceleration or deceleration within time-step T_s for any vehicle with general IDM is described by

$$a^*_{x,\alpha}(s_\alpha, v_\alpha, \Delta v_\alpha) = a_{x,max,\alpha}[1-(\frac{v_\alpha}{v_0})^\delta - (\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha})^2] \tag{3.1}$$

with the desired gap s^*, the maintenance of the gap s_α and approach rate Δv_α:

$$\begin{cases} s^*(v_\alpha, \Delta v_\alpha) = s_{0,\alpha} + s_{1,\alpha}\sqrt{\frac{v_\alpha}{v_0}} + T_{g,\alpha}v_\alpha + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{a_{x,max,\alpha}\, b_{x,com,\alpha}}} \\ s_\alpha = x_{\alpha-1} - x_\alpha - l_\alpha \\ \Delta v_\alpha = v_{\alpha-1} - v_\alpha \end{cases} \tag{3.2}$$

where parameters with index $\alpha - 1$ show the current value of leading vehicle, α the value of ego-vehicle. The other parameters are summarized in the Table 3.1:

Table 3.1: List of the parameters used by IDM in [2]

Parameter	Symbol	value
Desired Velocity	v_0	scenario-specific(m/s)
Time-Gap Headway	T_g	1.8 s
Maximum Acceleration	$a_{x,max}$	3 m/s^2
Comfortable Deceleration	$b_{x,com}$	8 m/s^2
Acceleration Exponent	δ	4
Length of car	l	5 m
Linear Jam Distance	s_0	2 m
Non-linear Jam Distance	s_1	3 m

Lateral Dynamic: In order to determine whether the lane should be changed, the MOBIL rules proposed by Schmidt [31] are introduced. A specific lane change depends on the two following vehicles in the current and the target lanes. As shown in Figure 3.3, the successive vehicles in the target and current lanes are denoted by b' and b respectively. The acceleration a_c denotes the value on the current lane for AV, whilst $\tilde{a}_c$ refers to the situation in the target lane. Likewise, $\tilde{a}_b$ and $\tilde{a}_{b'}$ present the acceleration of the old and new followers after the lane change of ego-vehicle.

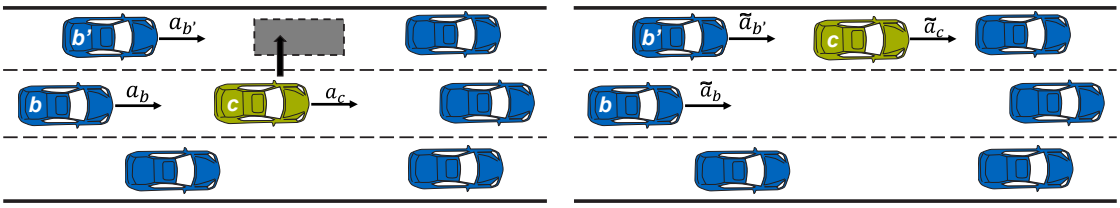


Figure 3.3: Nearest neighbours of AV considering lane change [32] (left: before lane change, right: after lane change)

Firstly, the possibility of executing a lane change will be checked with the safety criterion in Eq. (3.3), where $\tilde{a}_{b'}$ should not exceed a given safe limit $|b_{safe}|$ that is assigned with $8\ m/s^2$ according to [2, p.30]:

$$|\tilde{a}_{b'}| \leq |b_{safe}| \tag{3.3}$$

Under this prerequisite, the incentive criterion [32] in Eq. (3.4) and (3.5) is introduced to check whether a lane should be changed and whether it improves the individual local traffic situation.

Condition 1: changing lane from left to the right:

$$a_c + p(a_b + a_{b'}) > \tilde{a}_c + p(\tilde{a}_{b'} + \tilde{a}_b) + \delta_{th} - \delta_{bias} \tag{3.4}$$

Condition 2: changing lane from right to the left:

$$\tilde{a}_c + p(\tilde{a}_{b'} + \tilde{a}_b) > a_c + p(a_b + a_{b'}) + \delta_{th} + \delta_{bias} \tag{3.5}$$

The politeness factor p determines the degree of the influence of the lane-changing decision and is assigned with 0.5 to balance the egoistic and the altruistic behaviour of ego-vehicle. The keep-right directive of the lane usage rule in Europe is implemented by a constant bias δ_{bias} and the switching threshold δ_{th}, which are assigned with $0.3\ m/s^2$ and $0.1\ m/s^2$ respectively according to [32, p.10].

Moreover, for the planned lane change after MOBIL-checking, the ego-vehicle will move along an adapted cosine trajectory according to [31, p. 41-42]. The lateral deviation $O(x(t))$ of the vehicle from the road center line can be described by Eq. (3.6) for a lane change to the left $(-)$ or to the right $(+)$.

$$O(x(t)) = O(x_0) \pm \frac{w_{cw}}{2}(1 - cos(\frac{x(t) - x_0}{l_{tot}})\pi) \tag{3.6}$$

with

$$l_{tot} = v_t \sqrt{\frac{w_{cw} \cdot \pi^2}{2 \cdot a_{y,max}}} \tag{3.7}$$

where $x(t)$ indicates the abscissa where the ego-vehicle is located, x_0 the starting longitudinal location of lane change, l_{tot} the total length of the lane change process, w_{cw} the change width, and $a_{y,max}$ the maximal lateral acceleration which can be defined as $3\ m/s^2$ according to [2, p.30], with the purpose of ensuring a good dynamic performance of ego-vehicle.

3.1.2 Surrounding Vehicles

The surrounding vehicle models are not so complicated and intelligent by comparison, and only need to allow acceleration values as input and obtain the corresponding configuration. A system dynamic model for vehicles in [33] is thus suitable. It consists of the state vector $\boldsymbol{x} = [x\ y\ v_v\ \psi_c]$ and input vector $\boldsymbol{u} = [a_x\ a_y]$, where v_v indicates the velocity in natural coordinate and ψ_c the course angle, which can be approximated by the orientation ϕ in degree. A corresponding simplified non-linear vehicle model is described by Eq. (3.8).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v_v} \\ \dot{\psi_c} \end{bmatrix} = \begin{bmatrix} v_v \cdot cos(\psi_c) \\ v_v \cdot sin(\psi_c) \\ a_x \\ \frac{a_y}{v_v} \end{bmatrix} \tag{3.8}$$

It can also be interpreted from time $t-1$ to t in a discrete time-step T_s as:

$$\begin{bmatrix} v_{v,t} \\ \psi_{c,t} \\ x_t \\ y_t \end{bmatrix} = \begin{bmatrix} v_{v,t-1} \\ \psi_{c,t-1} \\ x_{t-1} \\ y_{t-1} \end{bmatrix} + \begin{bmatrix} a_{x,t} \\ \frac{a_{y,t}}{v_{v,t}} \\ cos(\psi_{c,t}) \cdot v_{v,t} \\ sin(\psi_{c,t}) \cdot v_{v,t} \end{bmatrix} \cdot T_s \tag{3.9}$$

3.1.3 Time-Step Adaption

Before getting into the details of interface with HighD, we must choose a appropriate time-step T_s to enhance the efficiency and maximize the accuracy of the optimizer. Normally, the performance of a search algorithm can be evaluated using the following criteria adapted from [34, p.13]. Due to the use of optimization algorithm as described in Section 2.3, the second item optimality is not considered here.

- **Accuracy:** Accuracy can be understood as the degree of fit to the original data. By comparing original data from HighD and the results of the simulator, the accuracy of the model is intuitively observed before optimization. Since the integral of acceleration is velocity and integrating velocity can derive position, the accumulated position deviation in the final frame ΔPos_f of N_{veh} vehicles as shown in Eq. (3.10) can well indicate the inaccuracy of the simulator.

$$\Delta Pos_f = \sum_{i=1}^{N_{veh}} \sqrt{\Delta Pos_{x,i,f}^2 + \Delta Pos_{y,i,f}^2} \tag{3.10}$$

- **Space complexity:** Space complexity describes how much memory is needed to perform the search algorithm, which can be estimated by the number of parameters. For a given time interval T_{tot}, the number of optimization parameters nov, i.e., the lateral and longitudinal accelerations, and the time-step T_s have the following relationship:

$$nov = 2 \cdot (N_{veh} - 1) \cdot \left\lfloor \frac{T_{tot}}{T_s} \right\rfloor \tag{3.11}$$

  where coefficient 2 denotes the division of longitudinal and lateral acceleration, N_{veh} the total number of vehicles in the scenario and $\lfloor \cdot \rfloor$ the largest integer no larger than T_{tot}/T_s.

- **Time complexity:** Time complexity can be measured with the running time of one simulation in Matlab, which is represented by T_{sim}. It is not completely decoupled from space complexity, since an increase of nov will also lead to longer calculation time.

The original time-step is 0.04 s, since the frame rate is 25 Hz according to Table 2.1. For the purpose of preventing additional deviations caused by interpolation, the selected observation values are all multiples of 0.04: [0.04; 0.08; 0.12; 0.16; 0.20]. In addition, to get all results approximately on the same scale, a min-max normalization method as Eq. (3.12) is used, where vector $\boldsymbol{value}$ denotes the combination of aforementioned variables in all time-steps.

$$value_{i,norm} = \frac{value_i - min(\textbf{value})}{max(\textbf{value}) - min\ (\textbf{value})} \tag{3.12}$$

Since the HighD itself has precision errors, and the scenario configuration must have deviated from the original one during the optimization process, the weight of ΔPos_f here is smaller than the other components. Therefore, the performance indicator of different time-steps PI_{ts} can be regarded as the weighted summation of the aforementioned criteria after normalization:

$$PI_{ts} = \Delta pos_{f,norm} + 2 \cdot T_{sim,norm} + 2 \cdot nop_{norm} \tag{3.13}$$

After collecting multiple samples from HighD and calculating the value of PI_{ts}, Figure 3.4 shows the box and whisker plot for different time-steps. In general, a high time-step can reduce the time and space complexity but introduce larger deviation from original data. Although the variation in time-step 0.16 s is slightly more dramatic than others, it obtains the lowest value of PI_{ts} on average, i.e., it has overall better search performance in qualitative judgment. As a result, the time-step of the simulator is assigned with 0.16 s for the latter optimization.

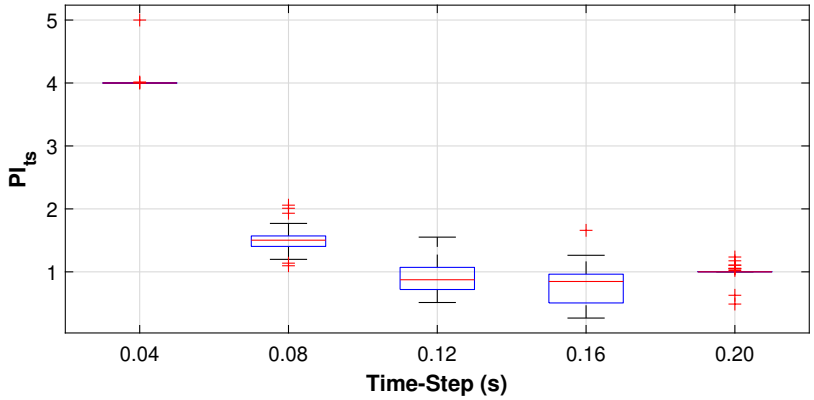


Figure 3.4: Box and whisker plots for time-step and performance indicator (based on cluster "data_12_tracks_ew" and "data_25_tracks_ew" from [10])

3.1.4 Time-Gap Adaption

The above-mentioned changes to the vehicle model will also cause corresponding altering in the scenarios. In the Eq. (3.2), the desired gap s^* highly depends on the value of time-gap headway T_g. The larger the time-gap, the farther the ego-vehicle must be away from the leading vehicle. However, in the original HighD dataset, the vehicle assumed to be AV did not drive according to the IDM and MOBIL models but moved forward in a collision-free manner. Therefore, when IDM is used in the simulator, it is likely that the ego-vehicle will maintain an excessive distance from the vehicle in front, even may collide with the rear vehicles . The former will result in a high penalty function, while the latter will affect the optimization performance, e.g., domain tg and $t2b$ are hard to increase.

According to [30, p.11], the realistic bounds of safe time headway, i.e., time-gap can be chosen in the range of [1, 3]. The German law recommends the time-gap of a relatively large value of 1.8 s [35, p.244], because it is difficult for the driver to estimate it through a safe distance in actual highway scenarios. For the purpose of better estimating the behaviour of AV in complex scenarios, the time-gap should not be set as the same as recommanded [36]. Furthermore, the time-gap is usually set between 1.0 and 1.8 s during the simulation, which is a commonly observed headway gap reported in [37]. Figure 3.5 depicts that drivers leave longer time-gap at lower speeds than at higher speeds, and the median is about 1.2 s at a speed interval of [72, 108[km/h. As a result, the time-gap used in this thesis is assigned to 1.2 s.

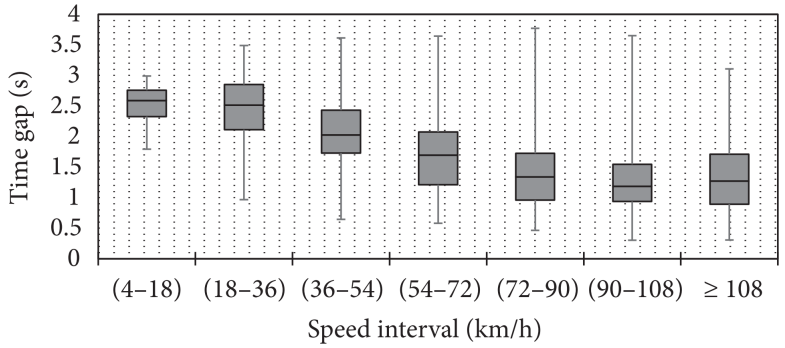


Figure 3.5: Box and whisker plots for speed and time-gap [37]

It should also be emphasized that although the time-gap here and the one called tg in the complexity evaluation refer to the same quantity, the latter is only used to describe the criticality, which does not affect the behavior of the vehicles directly.

3.2 Interface with Clusters

After introducing the model assumptions, the next step is to establish a connection between the simulator and the HighD dataset. The following introduces several important parts of building interfaces with clusters generated from [10].

3.2.1 Scenario Selection

There are thousands of different scenarios in HighD, and after clustering they are divided into different categories. Based on the clusters in [10], the information of the scenarios such as the total number of vehicles N_{veh} can be obtained directly. By observing and combining 11-vehicles-model that introduced in Section 2.2.2, a fact can be derived that the number of vehicles in the entire scenario is less than 6 is of little significance, since vehicles are generally not too close to each other on the highway and their restrictions on ego-vehicle are limited. For this reason, before selecting a scenario each time, a visualization tool is needed to observe whether the original one is representative and suitable, which is thus sometimes more subjective.

3.2.2 Coordinate Transformation

As introduced by [10], the coordinate system of clusters from HighD is similar to the one that shown in Figure 2.2. The main difference is that the origin of the lane coordinate in the cluster is on the upper bound of the northernmost lane and the vehicles are all driving from left to right. In addition, the ID of the lane is reversed as described in Figure 3.6.

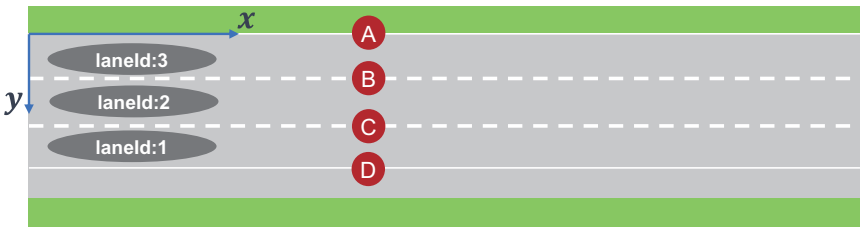


Figure 3.6: Coordinate system of clusters [10]

In the simulator, the global coordinate system of lane marking is also fixed and there exist two types of lane system in HighD: two lanes and three lanes as shown in Figure 3.7.

It is assumed that the vehicles in the simulator can be simplified with a bounding box to simulate its motion. Unlike the definition in clusters, the origin of vehicle's local coordinate falls at the center of the bounding box. Therefore, the original vehicle coordinate $(x,\ y)$ in clusters should be transformed into a new coordinate $(x',\ y')$ as follows, and the values of speed and acceleration

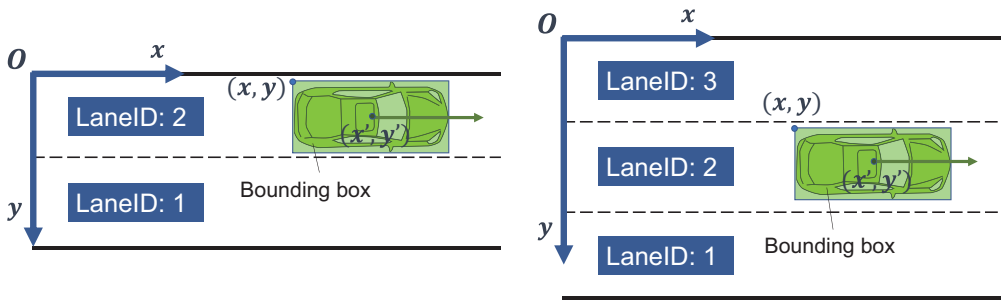


Figure 3.7: Lane coordinate system in simulator (left: 2 lanes, right: 3 lanes)

remain the same.

$$x' = x + \frac{length}{2}$$
$$y' = y + \frac{width}{2} \tag{3.14}$$

3.2.3 Missing Frames Complement

Since the observation data of HighD is based on a fixed lane range, the vehicles entering later or leaving earlier lost part of the information such as position, velocity, and etc. In order to maintain the consistency of each vehicle and the continuity of the entire scenario, we assume that the aforementioned vehicles keep driving at a constant speed in the lost frames, i.e., their longitudinal and lateral accelerations are zero. In this way, the integrity of the scenarios is guaranteed.

3.2.4 Input of the initial Data

What is important for the simulator is the initial data of each vehicle, such as the lane information, the initial position of vehicles and etc., which is already processed by the coordinate transformation and the missing frames complement. Here we distinguish between the configuration of ego-vehicle and surrounding vehicles in order to correspond to their respective models. The summary of initial data is shown in Table 3.2.

After obtaining the above basic information, the initial orientation $\phi[°]$ of all the vehicles can be calculated by the following formula:

$$\phi = \arctan \frac{v_y}{v_x} \tag{3.15}$$

3.2.5 Data Structure Transformation

The clusters generated from HighD and the complexity evaluation functions share the identical data structure named "Clusters", which contains the vehicle information and differs from the one called "Vehicles" in simulator. As shown in Figure 3.1, the form of the vehicle data from the simulator needs to be integrated to be consistent with "Clusters" before evaluating the complexity. Table 3.3 lists the correspondence between the two data structures, where N_f denotes the number of frames in one scenario.

Table 3.2: Summary of initial data

Aspect	Details
Lane information	number of lanes
	ordinate of lane markings in m
Scenario information	cluster name
	ID
	number of vehicles (ego-vehicle included)
	total time T_{tot} in s
	initial complexity C_0
	initial penalty function P_0
Ego-vehicle	initial position in m $(x',\ y')$
	initial velocity in m/s $(v_x,\ v_y)$
	size of bounding box in m $(length,\ width)$
Surrounding vehicles	name and type of vehicles
	initial positions in m $(x',\ y')$
	initial velocities in m/s $(v_x,\ v_y)$
	size of bounding boxes in m $(length,\ width)$
	longitudinal and lateral accelerations in m/s^2

Table 3.3: Correspondence between the data structures in clusters and simulator

"Clusters"	"Vehicles"
id	The order of the vehicle (ego=1)
frames	[1; 2; ... ; N_f]
bbox	[pos_x pos_y $length$ $width$]
xVelocity	
yVelocity	ValueMatrix
xAcceleration	
yAcceleration	
lane	lane
initialFrame	1
finalFrame	N_f
drivingDirection	1
class	Name
ROI	ROI

3.3 Complexity Evaluation

As stated by the complexity analysis method in Section 2.2, a total of 13 influence factors are considered in the complexity evaluation as shown in Table 2.2. However, the above analysis is based on the original HighD, and the original values for normalization in Table 2.3 are obtained by observing most normal traffic conditions, which have relatively low complexity in general. Consequently, before starting optimization, the complexity evaluation should be slightly modified to adapt to more complex scenarios and be consistent with the model assumptions defined in Section 3.1. To prevent redundant statements, we only list the difference between the optimizer and the work of Yu [8][9] as follows:

- **Time-step in traffic prediction:** The original time-step in HighD was 0.04 s. Consequently, it was not used as a variable in [8, p.34-35] and all relevant formulas in traffic prediction used it as fixed by default. After the time-step adaption in Section 3.1.3, we should change it to 0.16 s in related calculations.
- **Normalization value for factor** $variation$**:** Typical indicators of the dynamical changes are the variation of velocity and acceleration, and it was obtained from the following formulas:

$$\begin{cases} variation_{vx/vy} = max(\boldsymbol{v_x}/\boldsymbol{v_y}) - min(\boldsymbol{v_x}/\boldsymbol{v_y}) \\ variation_{ax/ay} = max(\boldsymbol{a_x}/\boldsymbol{a_y}) - min(\boldsymbol{a_x}/\boldsymbol{a_y}) \end{cases} \tag{3.16}$$

  The average longitudinal acceleration value of middle-class cars in the velocity range of 0-60, 0-80, 0-100 and 0-120 km/h is 2.9 m/s^2, while the average longitudinal deceleration value of middle-class cars comes to 8.9 m/s^2 [38, p. 466, 472]. Knowing from [2, p.30], the lateral acceleration of cars ranges from -3 to 3 m/s^2. The range of v_y will become larger at the same time, but it should still be smaller than v_x in the straight road. Due to its heavy weight and the traffic regulations, the allowable ranges of truck will be smaller than that of car. Still, for convenience, an overestimated calculation is used here, i.e., the normalized value of the car is used uniformly for all vehicles. Thus, the normalized value of aspects v_y, a_x and a_y are changed from 1.3 m/s, 1.5 m/s^2, 0.5 m/s^2 to 12 m/s, 12 m/s^2, 6 m/s^2 respectively. Moreover, the value of v_x remains unchanged, since the original value can already normalize almost all possibilities.

Besides, the average complexity of the whole scenario is more representative than the maximum, since the scenario is continuous and can be regarded as a whole. Thus, we use C to denote the average complexity of one scenario.

3.4 Penalty Function

To ensure that the vehicles do not exceed certain boundary conditions (traffic rules included) and the physical feasibility of its motion when increasing the complexity of scenarios, a penalty function is introduced. According to the existing highway driving requirements, three aspects of vehicles, i.e., location, velocity, and jerk will be considered during optimization process as shown in Table 3.4. Correspondingly, Eq. (3.17) calculates the total penalty function P of the whole scenario with N_f frames.

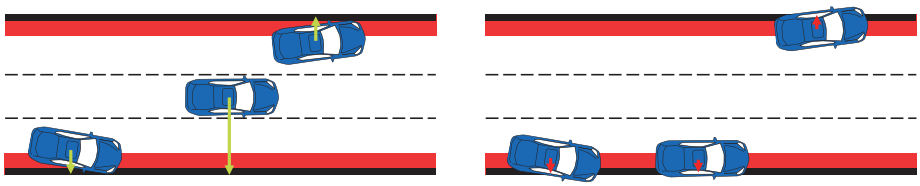
$$P = \sum_{i=1}^{N_f} (\sum_{j=1}^{3} P_{loc,j,i} + P_{vel,i} + P_{acc,i}) \tag{3.17}$$

3.4.1 Penalty of Location

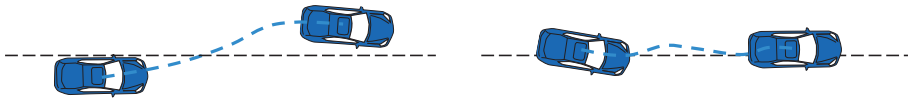
Figure 3.8 explains the vehicle configurations of allowed and penalized situations respectively when checking the location item. Next, we will discuss them in detail from three aspects as described in Table 3.4.

Table 3.4: Penalty function category

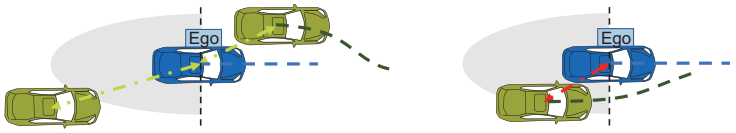
Aspect	Penalty Item	Description
Location	$pos_y + width/2 > Y_{max}$; $pos_y - width/2 < Y_{min}$	(1) Prevent vehicles from leaving the highway
	Drive over the lane marking too long (> 3 s)	(2) Prevent vehicles in critical situations
	Ellipse value $e' < 2$	(3) Prevent collisions between vehicles
Velocity	$v_v < 0m/s$	Prevent stopping on the highway and driving in the reverse direction
Jerk	$j_x > 20m/s^3$; $j_y > 10m/s^3$	Prevent physical infeasibility (jerk limitation)



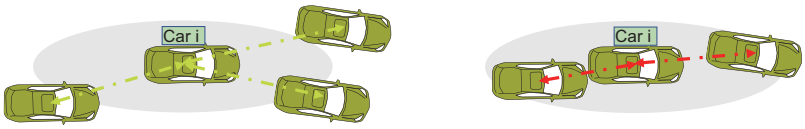
(a) Check the lateral location



(b) Check the driving time over the lane marking



(c) Check the collisions between ego- and surrounding vehicles



(d) Check the collisions between surrounding vehicles

Figure 3.8: Different examples of checking the locations in penalty function (left: allowed situation, right: penalized situation)

(1) Prevent vehicles from leaving the highway Since the only to be optimized variable is the acceleration set of the vehicles, it is necessary to consider that the vehicles should not drive off the boundary of highway, which is the first requirement that the locations need to meet. In addition, because the orientation of the vehicles in straight highway is generally small, it may happen that part of the vehicle exceeds the boundary in the actual situation, the main check here is whether the y-coordinate meets the requirements as shown in Figure 3.8a. Then adapting from [2, p.37] the penalty function $P_{loc,1}$ of leaving the highway in ith frame is calculated as follows:

$$P_{loc,1,i} = \begin{cases} 5 \cdot i \cdot |pos_{y,i} + width/2|, & if\, pos_{y,i} + width/2 > Y_{max} \\ 5 \cdot i \cdot |pos_{y,i} - width/2|, & if\, pos_{y,i} - width/2 < Y_{min} \end{cases} \tag{3.18}$$

(2) Prevent vehicles in critical situations After observing several optimization results, it was found that the vehicles may ignore the lane markings and drive over the lane for a long time as shown on the right of Figure 3.8b. Normally, on the highway, vehicles drive over the lane marking for a while only when they change lanes. With the purpose of simplifying the calculation process, it is approximately assumed that the lateral distance of the vehicle passing the lane marking when changing lanes is equal to its width. According to the data in [39] and HighD, the average width of a mid-size vehicle is about 6 to 6.5 feet (1.83-1.98 m) long. Therefore, as claimed by the process of changing lanes in Figure 3.9, the time for the vehicle itself (about 2 m) to cross the lane marking is about 3 s. The corresponding maximal number of frames can be derived by Eq. (3.19).

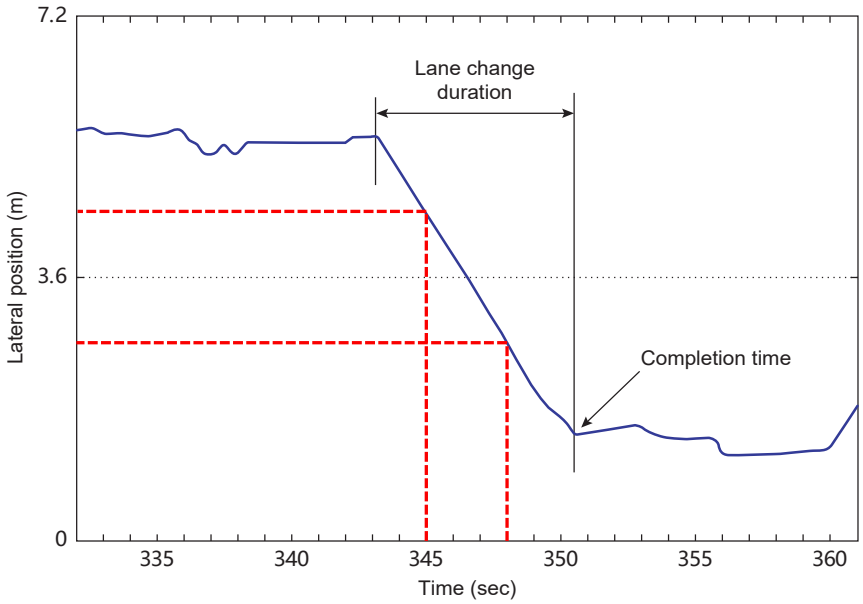


Figure 3.9: Definition of lane-change initiation and completion time points modified from [40]

$$frames_{lane,max} = \left\lfloor \frac{3\ s}{T_s} \right\rfloor = \left\lfloor \frac{3\ s}{0.16\ s} \right\rfloor = 18 \tag{3.19}$$

Considering the whole scenario, the corresponding penalty function $P_{loc,2}$ can be measured by Eq. (3.20), where $frames_{lane}$ denotes the number of frames that the vehicle drives over the lane marking.

$$P_{loc,2} = frames_{lane,max} \cdot frames_{lane},\ if\ frames_{lane} > frames_{lane,max} \tag{3.20}$$

(3) Prevent collisions between vehicles The collision prevention check between vehicles is carried out using an ellipse equation with ellipse value e as shown in Figure 3.10. In order to ensure a proper safety distance and maintain the rough side clearance, value a, b of ellipse are defined by Eq. (3.22) and e is assigned with 2. Adapting from [2, p.38], when the surrounding

vehicles are within the ellipse of the vehicle to be observed as shown on the right of Figure 3.8d, such behaviour will be penalized. The ellipse equation is:

$$\frac{(x_{oth}-x_{obs})^2}{a^2}+\frac{(y_{oth}-y_{obs})^2}{b^2}=e \tag{3.21}$$

with

$$\begin{cases} a=\frac{1}{2}\cdot(length_{obs}+length_{sur}) \\ b=\frac{1}{2}\cdot(width_{obs}+width_{sur}) \end{cases} \tag{3.22}$$

where variable with the subscript obs denotes the observed vehicle and with oth the other vehicles, which is also equivalent to:

$$\frac{(x_{oth}-x_{obs})^2}{a'^2}+\frac{(y_{oth}-y_{obs})^2}{b'^2}=1 \tag{3.23}$$

with $a'=\sqrt{e}\cdot a$ and $b'=\sqrt{e}\cdot b$.

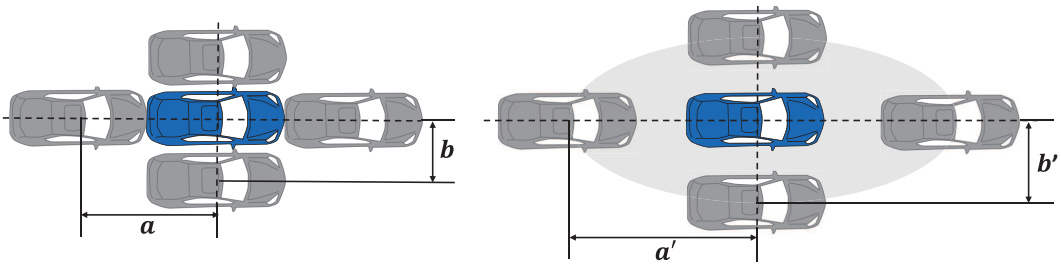


Figure 3.10: Definition of ellipse with $e=2$ (left: value a, b, right: value a', b')

It should be emphasized here, the situation of ego-vehicle will be detected separately from surrounding vehicles. For ego-vehicle as shown in Figure 3.8c, the surrounding vehicles are only not allowed to approach from the rear because the collision with leading vehicles can be avoided by the IDM. In order to prevent double calculations and improve efficiency, surrounding vehicles only check each other once to meet the above requirements for collision prevention. To this, the penalty function of preventing collisions between vehicles in ith frame is derived as follows:

$$P_{vel,3,i}=A_{ell}\cdot e^{B_{ell}\cdot e'},\ if\ e'=\frac{(x_{oth,i}-x_{obs,i})^2}{a^2}+\frac{(y_{oth,i}-y_{obs,i})^2}{b^2}<e=2 \tag{3.24}$$

where coefficients A_{ell} and B_{ell} are assigned with 24000 and -10.08 according to the definition in [2].

3.4.2 Penalty of Velocity

In the previous definition of Mayr [2, p.37], the lower limit of the speed was 5 m/s, which was to take into account the congested road conditions without dropping to a low velocity. However, it has been found in practice that low-speed driving may also produce high complexity and we can set it to a lower value. Therefore, the speed limit here is changed to 0 m/s, which means that the vehicle can not stop on the road or drive in the reverse direction. If the velocity in ith frame

does not meet the requirements, it can be penalized by Eq. (3.25).

$$P_{vel,i} = 50 \cdot i \cdot (v_{v,i})^2, \ if \ v_v < 0 \ m/s \tag{3.25}$$

3.4.3 Penalty of Jerk

To ensure the physical feasibility, the upper and lower limits of the acceleration of surrounding traffic are considered as the search boundary. In addition, the experience is that, the driving is more uncomfortable due to jerk than acceleration and the drastic change of vehicle acceleration within a limited time is also not physically possible.

According to the description in [41, p.56], the moving average over half a second of the lateral jerk shall not exceed 5 m/s^3, which is a conservative value for normal situations. Known from [42], vehicles can now initiate emergency braking with a jerk of up to 20 m/s^3. For the purpose of meeting the requirements of complex scenarios and prevent the conservatism, a high limited value 20 m/s^3 is allowed to the longitudinal jerk for the optimization.

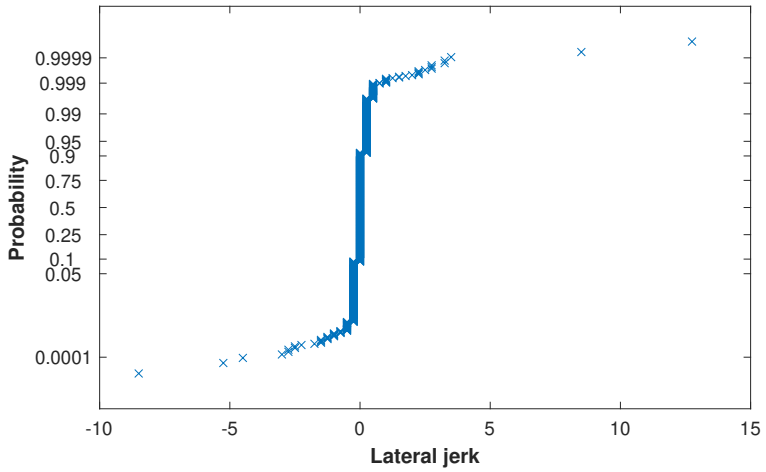


Figure 3.11: Distribution of lateral jerk in HighD (based on cluster "data_25_tracks_ew" from [10])

However, the current road is straight, thus the lateral acceleration will not change greatly in a limited time theoretically. And from the distribution in Figure 3.11 generated from HighD, it can be seen that lateral jerk is concentrated between -5 and 5 m/s^3. As a result, the lateral jerk limitation is assigned to 10 m/s^3 in the optimizer. Next, the corresponding penalty function can be calculated by Eq. (3.26), when the longitudinal and lateral jerk j_x, j_y in ith frame exceed the limit value.

$$P_{acc,i} = \begin{cases} 20, & if \ j_{x,i} > 20 \ m/s^3 \\ 10, & if \ j_{y,i} > 10 \ m/s^3 \end{cases} \tag{3.26}$$

3.5 Objective Function

Given the state $\boldsymbol{x}(t)$ subject to the model introduced in Section 3.1, the initial configuration from HighD in time t_0 and the input $\boldsymbol{a}(\cdot)$, i.e. accelerations of the surrounding vehicles, the objective function of the optimizer can be calculated in each iteration of optimization process, which is

denoted by $J(x(t), u(t), t_0)$. It has the following relationship with the previous defined complexity evaluation in Section 3.3 and penalty function in Section 3.4 according to Algorithm 5:

$$J = P - C \tag{3.27}$$

Algorithm 5 Calculation of objective function

procedure
Import initial data from HighD ▷ through interface
Import acceleration values for all frames $\boldsymbol{a_0}$ ▷ as input
Run the simulator and get data structure "Vehicles"
Interface with complexity evaluation and calculate the average complexity C
Calculate the penalty function P
Get the objective function J
return J

As a result, the purpose of the optimization algorithm is to find the optimal acceleration input $\boldsymbol{a}^*(\cdot)$ as shown in Eq. (3.28) to minimize the value of the objective function, which means that the penalty function is made as small as possible to maximize the average complexity in the whole scenario. The dimension of $\boldsymbol{a}$ is equal to the number of variables nov in Eq. (3.11).

$$\boldsymbol{a}^*(\cdot) = \underset{\boldsymbol{a}(\cdot)}{\operatorname{argmin}}\, J(\boldsymbol{x}(t), \boldsymbol{a}(t), t_0) = \underset{\boldsymbol{a}(\cdot)}{\operatorname{argmin}}\, P - \underset{\boldsymbol{a}(\cdot)}{\operatorname{argmax}}\, C \tag{3.28}$$

3.6 Tuner for Optimizer

After all the above preparations are completed, here launch into the generation of high complex scenario through optimization methods. As depicted in Figure 3.1, there exist many factors to be considered in the optimization process to enhance its performance, and it is beneficial to divide the optimizer into different parts at the beginning. Therefore, Figure 3.12 illustrates the metrics of tuning optimizer in a graphical form. It shows that the tuner of our optimizer depends on the problem setting (scenarios), the optimization algorithm, the way of configuring and analyzing the tuner, and the feedback from post-verification. In the next subsections, the evaluation methods of optimization result will be determined firstly and then the details of the tuner will be discussed in order to enhance the optimization performance. It is worth noting that configuring and analyzing the tuner is integrated in the other three parts, which are not discussed separately.

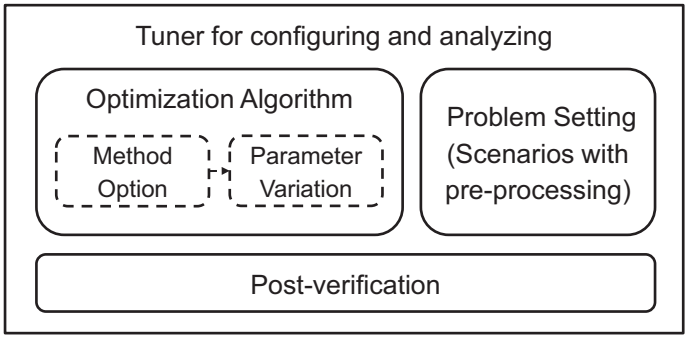


Figure 3.12: Tuner for optimizer

3.6.1 Definition of Evaluation Methods

Before going deeper to the tuner, the evaluation methods of final result should be defined. [43] provides two atomic performance measures for EA: one regarding solution quality and one regarding optimization speed, which is similar to the metrics called accuracy and complexity described in Section 3.1.3. Therefore, solution quality can be represented by the final value of objective function at certain time node or the optimization speed. Moreover, the performance metrics that are usually used in EA are the best fitness and the mean fitness. But using the mean fitness is not meaningful in our problem, whereas EA can lead to a large variance in the search period. Thus the best fitness is preferable here. As a result, in this thesis the following performance evaluation methods are mainly used:

- At maximal run-time or generation algorithm performance is defined as the final best fitness, which can be denoted by the increment of average complexity ΔC in % as shown in Eq. (3.29), since our main concern in the objective function is the change of complexity.

$$\Delta C = \frac{C_f - C_0}{C_0} \cdot 100\% \tag{3.29}$$

  where C_0 means the initial average complexity and C_f the final average complexity.

- Algorithm performance is measured by the trend and speed of optimization process, i.e., the change of objective function, within a given number of run-time or generations, when there is little difference between the final best fitness.

3.6.2 Problem Setting with Pre-processing

By reason of the IDM and MOBIL model introduced in Section 3.1, the ego-vehicle will maintain a greater distance from the leading vehicle than the original one in HighD or change lanes accidentally, although the time-gap is already adapted in Section 3.1.4. This may cause surrounding vehicles to collide with or drive too close to ego-vehicle, such as the situation in Figure 3.14, which is undesirable and manageable before optimization. Therefore, for the purpose of improving the efficiency of optimization and preventing excessive attention on reducing the penalty function at the beginning, pre-processing is used here to avoid collision caused by surrounding vehicles. According to Eq. (3.30) the initial positions $x_{follow/behind,0}$ of the vehicles behind ego-vehicle should be reduced if the safety condition is satisfied, where C_s denotes the safety coefficient that is assigned to 10 m to keep a safety distance between the centers of vehicles initially. Notably, the collision caused by the lane change of ego-vehicle is not pre-processed here, since it can be solved by the initialization of the solution space later.

$$\begin{aligned} \Delta x_{follow/behind,i,0} = x_{follow/behind,i,0} - (C_s - min(\mathbf{x}_{ego} - \mathbf{x}_{follow/behind})), \\ if\ \ min(\mathbf{x}_{ego} - \mathbf{x}_{follow/behind}) < C_s \end{aligned} \tag{3.30}$$

As introduced in Section 3.2, the initial configuration of vehicles can be obtained from the clusters. We first chose clusters with three lanes that obviously have more vehicles, and sort them by their complexity. Next, visualization tools are used to view the representativeness of the scenario and subjectively select its time interval. For instance, we should cover the situations that ego-vehicle drives in different lanes, as well as with a high-speed or medium-low speed. And the vehicle type should include both car and truck. In the end, three typical scenarios are sorted out in total with different complexities. Scenario 1 and 2 are mainly used to explore the best optimization

parameters in EA, while Scenario 3 has a high complexity originally and is suitable for detecting the performance of the optimizer. The vehicle information of all scenarios are listed in Appendix A.1.

Scenario 1 After coordination transformation and missing frames complement, Scenario 1 was intercepted for 10 seconds and the initial configuration is depicted in Figure 3.13. Table 3.5 - 3.6 briefly introduce the initial information corresponding to the description in Table 3.2.

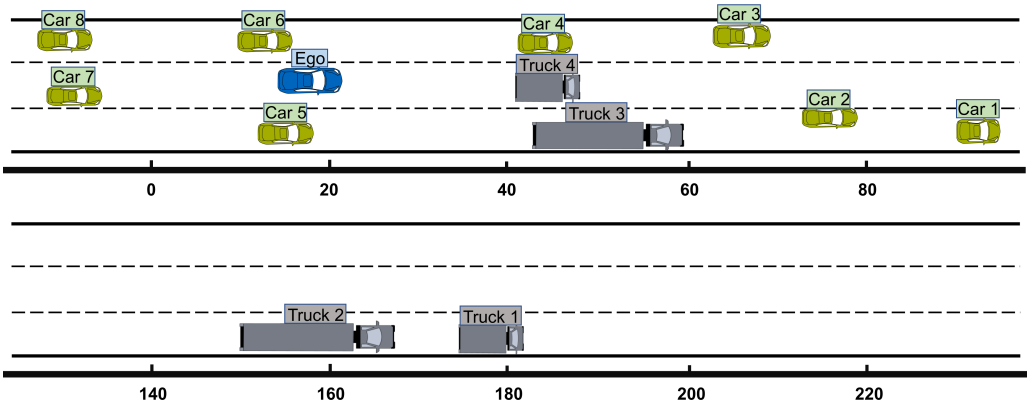


Figure 3.13: Initial configuration of scenario 1

Table 3.5: Lane information of Scenario 1 and 2

Item	Details
Number of lanes	3
Ordinate of lane markings in m	[0; 3.74; 7.38; 11.29]

Table 3.6: Scenario information of Scenario 1

Item	Details
Cluster name	data_25_tracks_ew
ID	cluster_veh13_nr02
Number of vehicles	13
Total time in s	10 (original: 29.72)
Initial complexity	0.4111
Initial penalty function	4020.2

In this scenario, the vehicles behind will not collide with ego-vehicle originally. However, after using the new model and adapting the time-gap, the ego-vehicle will change lane by MOBIL rules, although there is still a vehicle called "Truck 3" in lane 1, resulting in a penalty function greater than 0. But this situation can actually be avoided, e.g., by pre-accelerating and pre-decelerating the vehicles in the initial solution space, thus the pre-processing is not required here.

In addition, the number of variables nov_1 to be optimized can be directly calculated using Eq. (3.31):

$$\begin{aligned} nov_1 &= 2 \cdot (N_{veh,1} - 1) \cdot \left\lfloor \frac{T_{tot,1}}{T_s} \right\rfloor \\ &= 2 \cdot 12 \cdot \left\lfloor \frac{10}{0.16} \right\rfloor = 1512 \end{aligned} \tag{3.31}$$

Scenario 2 With the same lane situation defined in Table 3.5, the ego-vehicle of Scenario 2 is driving on the third lane and surrounded by 7 vehicles. Due to the IDM model of ego-vehicle, Figure 3.14 shows that the vehicle named "Car6" will get too close to the ego-vehicle, since $min(\mathbf{x}_{ego} - \mathbf{x}_{follow})$ is equal to 4.87 m, which is less than C_s. For this reason, the initial position of "Car6" will reduce $C_s - min(\mathbf{x}_{ego} - \mathbf{x}_{follow}) = 5.23\ m$ correspondingly. The pre-processed configuration of scenario 2 is depicted in Figure 3.15. Table 3.7 show the remaining initial information.

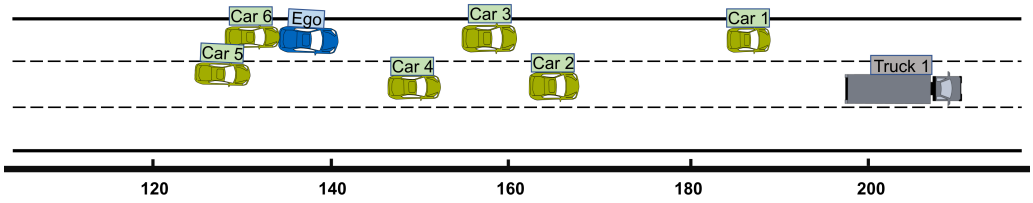


Figure 3.14: One fragment of original Scenario 2 with potential collision

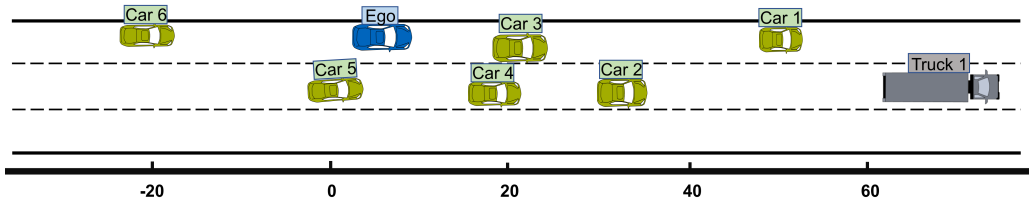


Figure 3.15: Initial configuration of Scenario 2 after pre-processing

Table 3.7: Scenario information of Scenario 2

Item	Details
Cluster name	data_25_tracks_ew
ID	cluster_veh08_nr105
Number of vehicles	8
Total time in s	16 (original: 27.68)
Initial complexity	0.3070
Initial penalty function	0

Similar to Scenario 1, Eq. (3.32) calculates the number of variables nov_2 to be optimized in Scenario 2.

$$\begin{aligned} nov_2 &= 2 \cdot (N_{veh,2} - 1) \cdot \left\lfloor \frac{T_{tot,2}}{T_s} \right\rfloor \\ &= 2 \cdot 7 \cdot \left\lfloor \frac{16}{0.16} \right\rfloor = 1400 \end{aligned} \tag{3.32}$$

Scenario 3 For the purpose of detecting whether the optimizer after parameter variation has the ability to handle scenarios with high complexity, scenario 3 is excerpted from HighD and depicted in Figure 3.16.

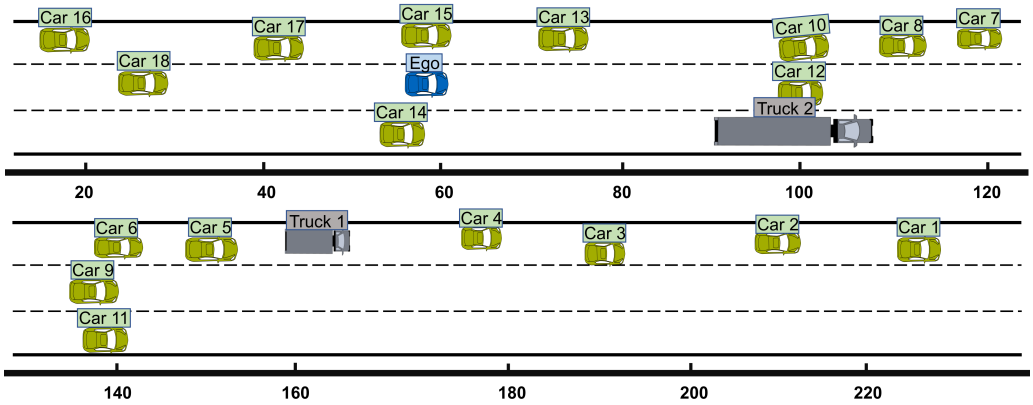


Figure 3.16: Initial configuration of Scenario 3

Checking the penalty function, no collision is found initially, thus the pre-processing is not required here. The corresponding information of Scenario 3 is shown in Table 3.8-3.9.

Table 3.8: Lane information of Scenario 3

Item	Details
Number of lanes	3
Ordinate of lane markings in m	[0; 3.89; 7.69; 11.66]

Table 3.9: Scenario information of Scenario 3

Item	Details
Cluster name	data_12_tracks_ew
ID	cluster_veh21_nr02
Number of vehicles	21
Total time in s	8 (original: 17.2)
Initial complexity	0.5311
Initial penalty function	20

The number of variables nov_3 to be optimized in this scenario is derived as follows:

$$\begin{aligned} nov_3 &= 2 \cdot (num_{veh,3} - 1) \cdot \left\lfloor \frac{T_{tot,3}}{T_s} \right\rfloor \\ &= 2 \cdot 20 \cdot \left\lfloor \frac{8}{0.16} \right\rfloor = 2000 \end{aligned} \tag{3.33}$$

3.6.3 Optimization Algorithm

Since the optimization algorithm can only find the quasi-optimal solution according to a certain rule, it requires tedious setting and adjusting of the hyper-parameters. According to the algorithm design in [43], the performance of optimization process is mainly determined by design details, i.e. parameter values. This subsection will introduce the general settings of the optimizer, the choice of optimization algorithm and corresponding parameter variation.

General Settings The input variables of the optimizer are the longitudinal and lateral accelerations of the surrounding vehicles, whose size is determined by nov from Eq. (3.11). Each vehicle has two acceleration values per time-step and the corresponding order is:

$$\boldsymbol{a} = \left[\begin{array}{c} a_{x,1,t_0}; a_{y,1,t_0}; \cdots; a_{x,(N_{veh}-1),t_0}; a_{y,(N_{veh}-1),t_0}; \cdots\cdots; a_{x,1,T_{tot}}; a_{y,1,T_{tot}}; \\ \cdots; a_{x,(N_{veh}-1),T_{tot}}; a_{y,(N_{veh}-1),T_{tot}} \end{array} \right] \tag{3.34}$$

Furthermore, realistic upper and lower limits of the longitudinal and lateral accelerations should be specified before optimizing. According to the documentation in [2, p.39], the acceleration limits of different vehicle types in HighD are declared in Table 3.10.

Table 3.10: Limits of longitudinal and lateral acceleration adapted from [2]

Type	Item	Lower Limit in m/s^2	Upper Limit in m/s^2
Track	longitudinal acceleration	-7.0	1.0
	lateral acceleration	-1.0	1.0
Car	longitudinal acceleration	-9.0	3.0
	lateral acceleration	-3.0	3.0

Method Options In Section 2.3 the definition and process of widely used optimization algorithms are introduced. Mayr [2, p.40-44] describes a efficiency test of optimization algorithms for exploring maximal complexity, which starts with identical value and scenario. The efficiency η is measured by the ratio of the reduction of the objective function ΔJ to the optimization time T_{opt} with the same maximum calculation time 1500 s, which can be described by Eq. (3.35). The results in Figure 3.17 shows the ΔJ in three situations, which suggests that PSO and GA possesses higher efficiency and better performance than PS and SA.

$$\eta = \frac{\Delta J}{T_{opt}} \cdot 100\% = \frac{1 - J_{end}/J_{start}}{T_{opt}} \cdot 100\% \tag{3.35}$$

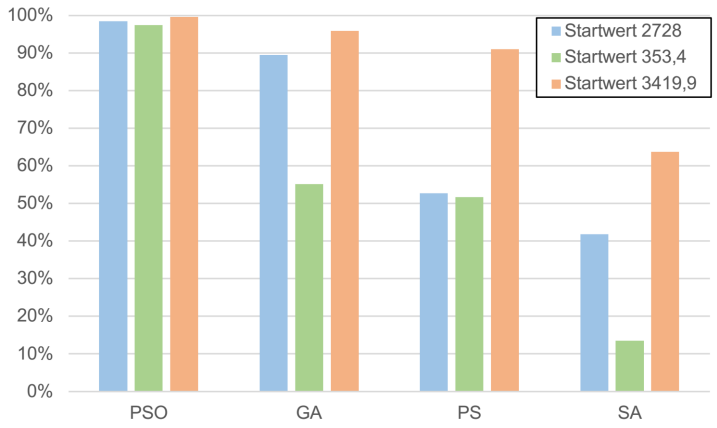


Figure 3.17: The comparison of different starting scenarios and starting values [2]

In addition, PSO has been shown to work better when its empirical parameters are properly selected with higher exploration opportunity, while SA and PS works worse, which can not guarantee an optimal solution within the same iterations [44]. GA offers similar structures as

PSO under the same framework of EA, and it can better deal with problems with constraints. Hence, the optimization methods are mainly focused on PSO and GA in this work.

Parameter Variation It is important to note that the value of parameters in EAs is not specified in general. Depending on particular design choices one might obtain different values [43]. As mentioned in Section 2.3, finding a good set of parameter values is also a complex optimization task with a nonlinear objective function, interacting variables, noise, and a lack of analytic solvers. Traditional in the area of optimization calculation distinguishes two approaches for the selection of parameter values: parameter tuning(offline) and parameter control(online) [45]. Therefore, in order to reduce the computational complexity, we chose to adjust parameters at the design layer instead of during problem solving. The main objective here is to analyze and investigate the different performance of EA with parameter variation to provide better parameter combination for effective optimization.

- **GA**: The general framework of GA is shown in Figure 2.6. Each arrow in the figure and the aforementioned rules in Section 2.3 will affect the performance of GA. Nevertheless, the optimal choices of these rules vary from different optimization problems, and it is impossible to know their corresponding relationship before optimization. Therefore, the mainly investigated parameters are the (1) population size, (2) number of generations, (3) crossover fraction, (4) elite ratio, (5) crossover type, (6) mutation option, (7) selection type, and (8) initial population matrix.

  In each set, one of the aforementioned parameters is varied according to the options in Table 3.11, while the remained are maintaining constant. Using variable-controlling approach, the undetected values will be selected from Table 3.12 through "rule of thumb" rules.

Table 3.11: Summary of values for investigated parameters in GA

Parameter	**Values**
Population size(N_{GA})	50, 100, 150, 200, 250, 300
Number of generations ($T_{g,GA}$)	50, 100, 300, 500, 1000
Crossover Fraction (C_f)	10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%
Elite Ratio (E_r)	2%, 10%, 18%, 26%, 34%, 42%, 50%
Crossover Type (C_t)	scattered, single point, two-point, intermediate, heuristic
Mutation Option ($scale, shrink$)	$scale$: 1, 5, 10, 20, 50; $shrink$: 1
Selection Type (S_t)	stochastic uniform, remainder, roulette, tournament
Initial Population Matrix ($\boldsymbol{A}_0$)	(1) Initialize only one row with original data from HighD, the rest take random values within the upper and lower bounds
	(2) Initialize only half of the rows same as the original data from HighD, the rest take random values within the upper and lower bounds
	(3) Initialize all rows same as the original data from HighD
	(4) Initialize all rows using the original data plus a random deviation within a predefined range
	(5) Initialize all rows with a new set of acceleration values using pre-acceleration and deceleration approach in the longitudinal direction, which is generated from the original data and based on the relative position of surrounding vehicles to the ROI of ego-vehicle

Table 3.12: The default values of the uninvestigated parameters in GA

Parameter	Value
Population size (N_{GA})	100
Number of generations ($T_{g,GA}$)	100
*Max Time ($T_{t,GA}/s$)	7200
Crossover Fraction (C_f)	80%
Elite Ratio (E_r)	10%
Crossover Type (C_t)	Scattered
Mutation Option ($scale, shrink$)	20, 1
Selection Type (S_t)	Stochastic uniform
Initial Population Matrix ($\boldsymbol{A}_0$)	Option (5)

Notably is that the max time $T_{t,GA}$ is only used as the termination condition for population size variation, since the larger the population size under the same generation, the longer the search time, which will affect the parallelism of the comparison.

In addition, the option (5) of generating initial population matrix ($\boldsymbol{A}_0$) utilize a pre-acceleration and deceleration method in longitudinal direction according to the relative position of surrounding vehicles to the ego-vehicle's ROI in original data. Figure 3.18 depicts the corresponding partition of highway based on ROI, which differs from the way of label assignment in Figure 2.5. In sections - and --, the vehicles in front of ego-vehicle decelerate in the longitudinal direction, whilst the vehicles in the rear accelerate in sections + and ++, so that the initial acceleration vector $\boldsymbol{a}_0$ allows the surrounding vehicles to stay in the ROI as long as possible, and the behaviour of ego-vehicle is also more constricted. In order to reduce collisions caused by acceleration and deceleration, the variations of a_x in the sections within the ROI (+ and -) are half of the outside the ROI (++ and --). Moreover, it can be also derived from the empirical Cumulative Distribution Function (CDF) of a_x in Figure 3.19, that in most cases the vehicles are accelerating, so the value of pre-deceleration is greater than that of pre-acceleration.

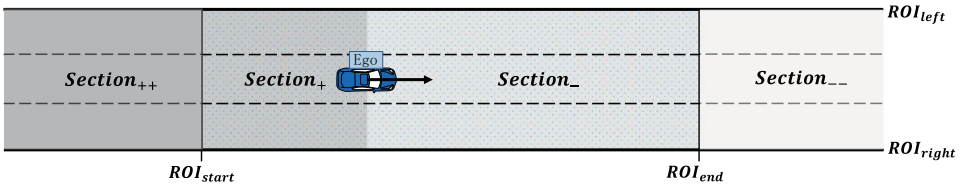


Figure 3.18: ROI-based partition of highway

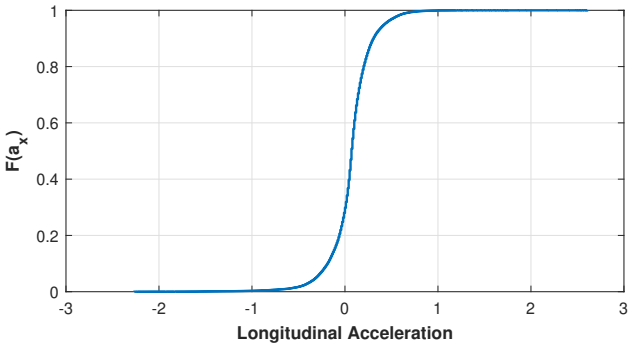


Figure 3.19: Empirical CDF of a_x in HighD (based on cluster "data_25_tracks_ew" from [10])

After adjusting and testing the range of the values adopted in this thesis are as follows:

Section--: [-0.4, 0]; **Section-:** [-0.2, 0]

Section++: [0, 0.06]; **Section+:** [0, 0.03]

The specific values within the range is determined according to the population size with the requirement to fill the entire matrix $\boldsymbol{A_0}$ as much as possible with a uniform distribution. For instance, when the population size is 100, section-- takes 10 values equidistant from -0.4 to 0, section ++ selects 10 equidistant values from 0 to 0.06, and so on. It should be noted that, in fact, the aforementioned range and the way to choose the specific values are not crucial for the optimization performance, since the GA itself has mutation option, and PSO is also based on global search with exploration.

- **PSO**: Similar to GA, the framework in Figure 2.8 shows the significant steps of PSO. From this, combining the iteration scheme in Figure 2.7 and reusing parts of the settings in GA, the following parameter variations are investigated: (1) swarm size, (2) number of iterations, (3) inertia range, (4) self adjustment weight, (5) social adjustment weight, (6) initial swarm span, and (7) initial swarm matrix. The initial swarm matrix here is equivalent to the initial population matrix above, which are used to initialize the search space at the beginning and denoted by $\boldsymbol{A}_0$.

  With the same principle as used in GA, the values of investigated and uninvestigated parameters are shown in Table 3.13 and 3.14 respectively.

Table 3.13: Summary of values for investigated parameters in PSO

Parameter	Values
Swarm size (N_{PSO})	50, 100, 150, 200, 250, 300
Number of Iterations ($T_{g,PSO}$)	50, 100, 300, 500, 1000
Inertia Range (w_0)	[0.01 1], [0.01 2], [0.01 5], [0.01 10], [0.01 50]
Self Adjustment Weight (c_1)	0.5, 1.0, 1.5, 2.0, 2.5, 3.0
Social Adjustment Weight (c_2)	0.5, 1.0, 1.5, 2.0, 2.5, 3.0
Initial Swarm Span (Ini_{ss})	500, 1000, 1500, 2000, 2500, 3000
Initial Swarm Matrix ($\boldsymbol{A}_0$)	(1) Initialize only one row with original data from HighD, the rest take random values within the upper and lower bounds
	(2) Initialize only half of the rows same as the original data from HighD, the rest take random values within the upper and lower bounds
	(3) Initialize all rows same as the original data from HighD
	(4) Initialize all rows using the original data plus a random deviation within a predefined range
	(5) Initialize all rows with a new set of acceleration values using pre-acceleration and deceleration approach in the longitudinal direction, which is generated from the original data and based on the relative position of surrounding vehicles to the ROI of ego-vehicle

The max time $T_{t,PSO}$ is also only used as the termination condition for swarm size variation. Notably, the inertia range is considered more in smaller range, because larger one leads to the new velocity being more in the same direction as the old, and to stabilize the swarm too early. Furthermore, when c_2 = 0, there is no shared

Table 3.14: The default values of the uninvestigated parameters in PSO

Parameter	Value
Swarm size (N_{PSO})	100
Number of Iterations ($T_{g,PSO}$)	100
*Max Time ($T_{t,PSO}/s$)	7200
Inertia Range (w_0)	[0.1 2]
Self Adjustment Weight (c_1)	2
Social Adjustment Weight (c_2)	2
Initial Swarm Span (Ini_{ss})	2000
Initial Swarm Matrix (Ini_m)	Option (5)

information between particles, which is cognition-only, the probability to get the optima is very small because of the loss of interaction between individuals. Similarly, when c_1 is equal to 0, the particle has no cognitive ability, and the algorithm only has the ability to get to the new searching area by particles' cooperation with each other [46]. Thus they all start with a non-zero value here.

3.6.4 Post-Verification

After the procedures above are finished, we need to analyze and verify the optimization results qualitatively and quantitatively, to judge whether the given requirements are satisfied and observe which algorithm has better optimization performance.

There exist two qualitative critical issues in search strategies when using EA: exploiting the best solution and exploring the search space [47, p.158]. Exploration means that the search in the solution space is more abundantly. The algorithm will try its best to "forget" its ancestors through exploration, in particular, the influence of the good ancestors at local optima should be reduced, which leads to a development in another direction to find different solutions. In contrast, exploitation develops in a particular direction using the good ancestors, which converges the algorithm faster, but will probably let the searching area fall into the local minimum. Therefore, one way to compare optimization performance is to see if the exploration and exploitation are well balanced through the hyper-parameters used in EA.

On the other hand, we can also quantitatively analyze the results of optimization. The 13 influence factors in complexity evaluation listed in Table 2.2 can be classified according to their intuitive degree. For instance, nb_type and nb_num respectively represent the type and number of surrounding vehicles, which can be directly observed from the configuration, while the value of $devi\_eu$ is challenging to judge from the scenarios directly since it presents the behaviour of surrounding vehicles abstractly. In this way, these factors are divided into three categories as shown in Table 3.15 and can be investigated to check the optimization quality respectively.

Table 3.15: The categories of influence factors in complexity evaluation

Category	Item
Intuitive	nb_type, nb_num, $variation$, $dynamic$, $ratio$
Medium	tg, $t2b$, $connection$
Complicated	noa_nb, noa_ego, pa_ego, pa_nb, $devi\_eu$

4 Results and discussion

The following sections will present and analyze the results of the tuner for the optimizer. Section 4.1 will inspect the optimization results using GA. Firstly the final hyper-parameters based on the results of parameter variation will be selected, and then they will be used to optimize the three example scenarios. In the same way, the results of PSO will be listed in Section 4.2. Finally, in Section 4.3, the above results using EA will be discussed and compared though post-verification from quantitative and qualitative aspects.

To achieve the above sets of problems, the simulator and solver for optimization were coded in Matlab 2019b with the global optimization toolbox. The optimization processes were run on a computer with a 3.6-GHz Intel Xeon processor and a memory of 64 GB with four cores.

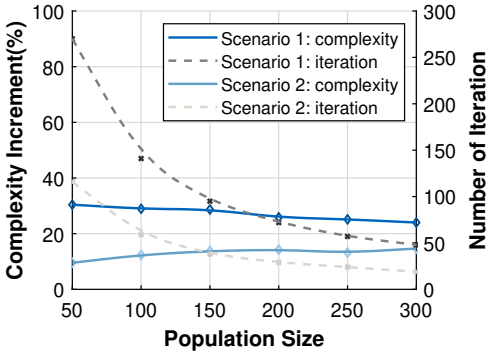
4.1 Results using Genetic Algorithm

In this section, the results using GA will be listed in detail, i.e., parameter variation and scenario tests with the derived optimal parameters. Owing to lack of time, only two scenarios (Scenario 1 and 2) are used for the investigation of parameter variation, and we finally test three scenarios that are depicted in Section 3.6.3.

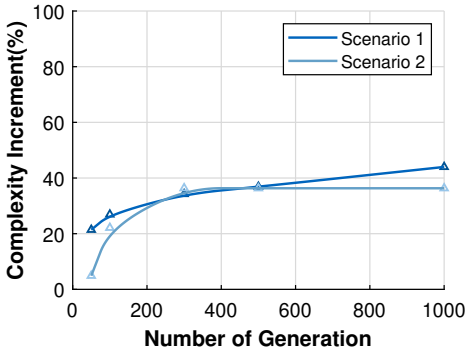
4.1.1 Parameter Variation

According to the description in Section 3.6.3, the parameter variation of GA is investigated with the setting in Table 3.11. The remaining parameters during the investigation are maintaining constant according to the default values in Table 3.12. As mentioned in Section 3.6.1, the complexity increment ΔC in % in Eq. (3.29) and the trend of the optimization process (change of objective function) are mainly used as performance evaluation methods here. The results are shown in Figure 4.1.

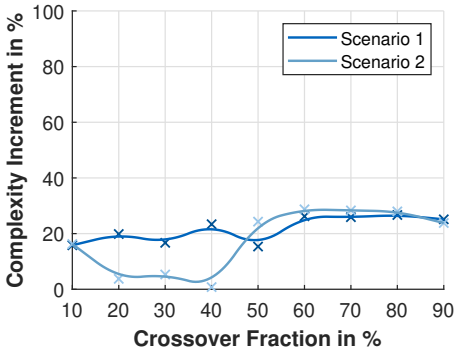
- **Population Size Variation:** To measure the impact of population size, the complexity increments (%) of different values are compared in Figure 4.1a. The termination condition is that the optimization time reaches 7200 s. It can be observed that there is no significant change between different population sizes. Besides, a "small" population size could guide the algorithm to poor solutions, while a "large" population size could make the algorithm expend more computation time in finding a solution [48]. Under this prerequisite, the result suggests to set N_{GA} to be located in interval [100, 200]. Besides, in order to enhance the computational efficiency, i.e, more iterations in a certain time, the population size N_{GA} is finally set to 100 in the optimizer.



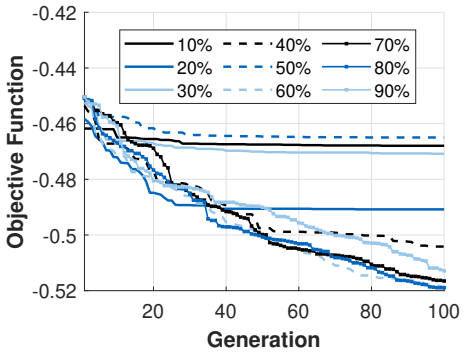
(a) Comparison of population size variation



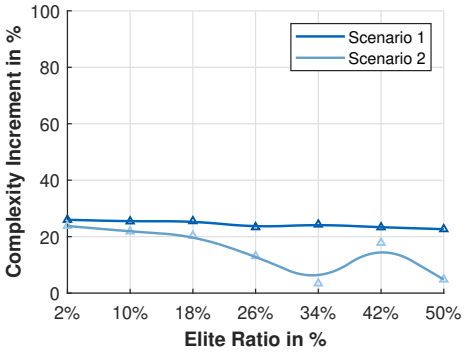
(b) Comparison of generation variation



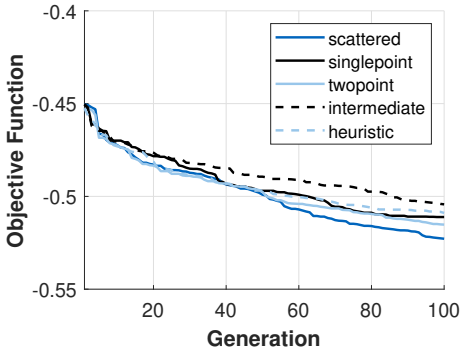
(c) Comparison of crossover fraction variation



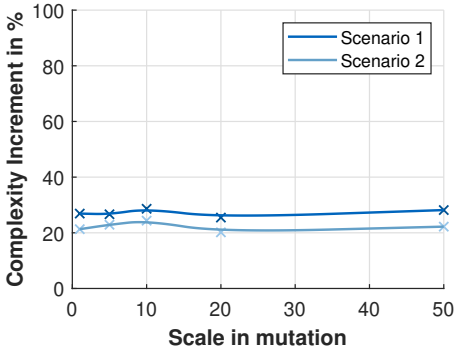
(d) Comparison of optimization process of crossover fraction variation (Scenario 1)



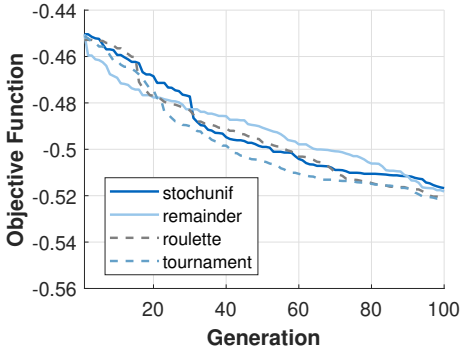
(e) Comparison of elite ratio variation



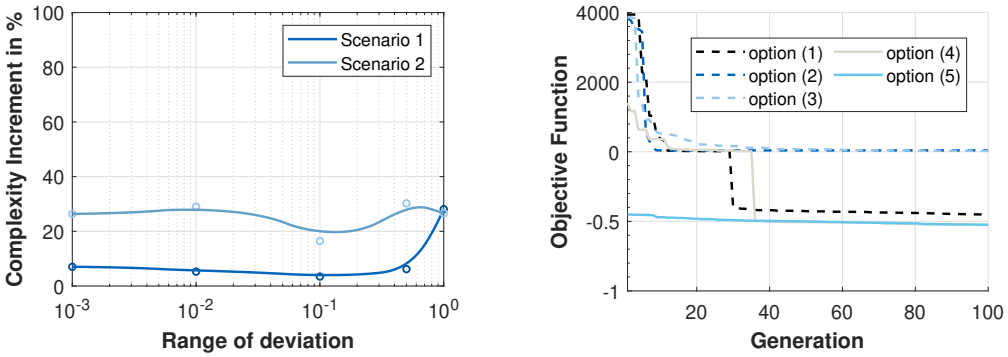
(f) Comparison of crossover type variation



(g) Comparison of mutation option variation



(h) Comparison of selection type variation



(i) Comparison of the range of deviation in the option (4) of initialization

(j) Comparison of initialization method variation (Scenario 1)

Figure 4.1: Results of parameter variation using GA under the setting in Table 3.11 and Table 3.12

- **Generation Variation:** The results that are shown in Figure 4.1b indicate that when the generation is less than 500, increasing its number can significantly improve the optimization results. Although the curve of Scenario 1 shows a slight improvement when using the generation of 1000, its time complexity is considerable in turn, which is also undesirable. Therefore, the number of generation $T_{g,GA}$ is assigned with 500.
- **Crossover Fraction Variation:** Combining the results that in Figure 4.1c and 4.1d, it can be clearly found that when crossover fraction is less than 60%, the optimization process converges more quickly with a bigger value, which shows a bad performance. The change between 60% and 80% is not particularly noticeable. Additionally, observing the curves in Figure 4.1d of Scenario 1, crossover fraction C_f can be set to 80%, which shows a better optimization trend.
- **Elite Ratio Variation:** As depicted in Figure 4.1e, when the proportion of elite is larger, the final complexity increment is generally smaller, especially in Scenario 2. Therefore, here 2% is taken as the final assignment of elite ratio E_r.
- **Crossover Type Variation:** With the trend of the optimization curves under different crossover types in Figure 4.1f, it can be seen that the decline rate of the curve "scattered" is faster than others, especially after 50 generations, thus "scattered" is chosen as the final crossover type.
- **Mutation Option Variation:** The results in Figure 4.1g show that the curves of $scale$ variation in the mutation rules that is described in Table 2.6. For both scenarios they have a local optimal solution at 10. Consequently, here $scale$ is finally taken as 10.
- **Selection Type Variation:** Although the shape of the optimization curves in Figure 4.1h are slightly different for each selection type, there is almost no difference in the final fitness of objective function. For this reason, we choose the default selection function directly in Matlab GA toolbox, that it, stochastic uniform.
- **Initialization Method Variation:** The most crucial step in the functioning of GA is the generation of an initial population, as shown in Algorithm 1. It has been recognized that if the initial population is appropriate, then the optimizer has a better possibility of finding a good solution [49].

The assumption is that the initial population can be seeded with the original solution from HighD (option (1), (2), (3)) or adjusted solutions in a pseudo-random way (option (4), (5)), as shown in Table 3.11 and Figure 4.1j. Regarding option (4), the randomness is varied by the symmetrical range of deviation. The corresponding results are shown in Figure 4.1i and recommend the optimal range of 1 (± 0.5), which obtains a higher complexity with zero penalty function after optimization, especially in Scenario 1.

Moreover, comparing the five different initialization methods in Figure 4.1j, the best way to generate the initial population is option (5), i.e., initializing the rows with a new set of acceleration values using pre-acceleration and deceleration method in the longitudinal direction based on the original data. It can reduce the value of penalty function to zero and increase the complexity before optimization starts, especially when a collision is caused by MOBIL rules in Scenario 1. Then the efficiency of optimization will be improved and the search process can find better solutions easier.

As a result, Table 4.1 summarized the optimal hyper-parameters in the optimizer using GA.

Table 4.1: The optimal parameters of GA

Parameter	**Value**
Population size (N_{GA})	100
Number of generations ($T_{g,GA}$)	500
Crossover Fraction (C_f)	80%
Elite Ratio (E_r)	2%
Crossover Type (C_t)	Scattered
Mutation Option ($scale, shrink$)	10, 1
Selection Type (S_t)	Stochastic uniform
Initial Population Matrix ($\boldsymbol{A}_0$)	Option (5)

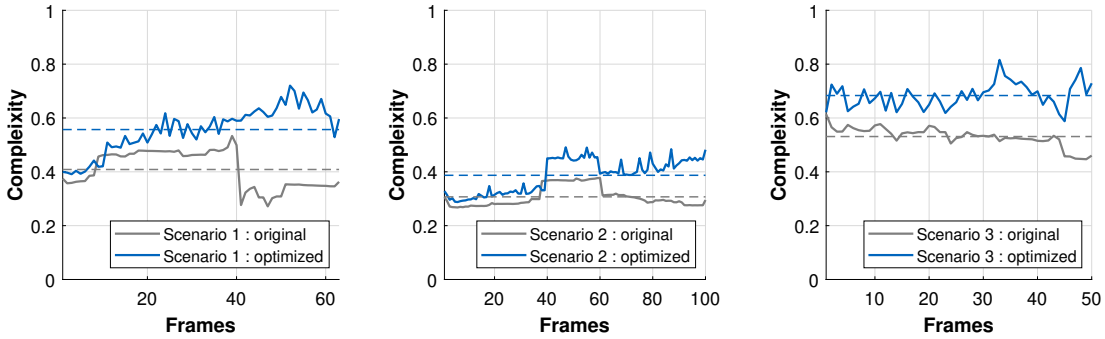
4.1.2 Scenario Test

After obtaining the optimal parameters of GA, we can use them to test three defined scenarios. Table 4.2 shows the final optimization results of complexity evaluation, penalty function, and the corresponding running time.

Table 4.2: The optimization results of three scenarios using GA

	Scenario 1	**Scenario 2**	**Scenario 3**
Initial Complexity	0.4111	0.3070	0.5311
Final Complexity	0.5570	0.3871	0.6834
Initial Penalty Function	4020.2	0	20.0
Details	Collision between vehicles	-	Beyond jerk limitation
Final Penalty Function	0.00007	0.00007	0.0017
Details	Collision between vehicles	Collision between vehicles	Collision between vehicles
Running Time in s	30061.71	28599.65	33018.71

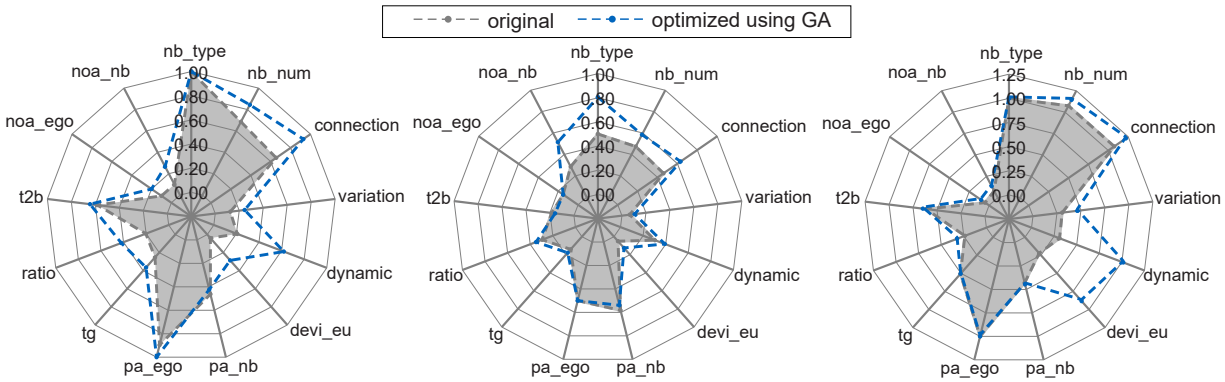
Next, the distribution of complexity in the whole scenario, i.e., complexity of each frame, is depicted in Figure 4.2. A dotted line indicates the average complexity. It can be seen that through GA, basically the complexity of each frame has been improved. And the three scenarios have increased by 35.50%, 26.1% and 28.6% respectively.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 4.2: Complexity comparison of three scenarios

In another way, the complexity evaluation of the optimized scenarios can be conducted with the help of a Kiviat-diagram, as shown in Figure 4.3, where each axis represents one of the 13 influence factors with its normalized complexity value.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 4.3: Kiviat-diagram to visualize the complexity of three scenarios

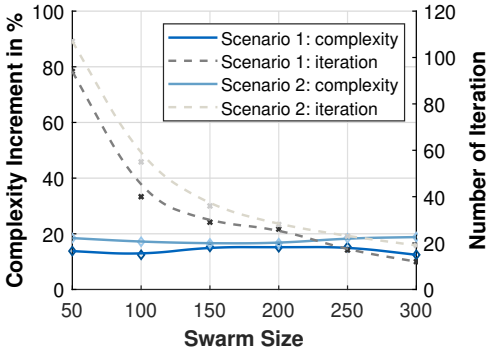
4.2 Results using Particle Swarm Optimization

Similar to Section 4.1, the results of parameter variation and scenario tests using PSO are listed in this section.

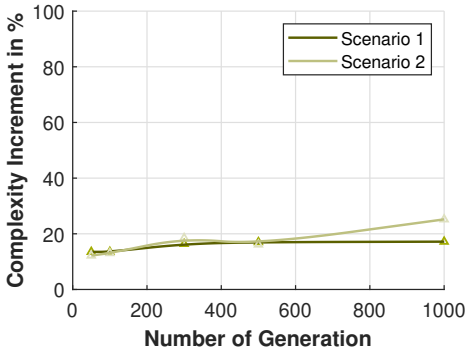
4.2.1 Parameter Variation

According to Table 3.13, seven hyper-parameters are investigated in PSO in total. As depicted in Figure 4.4, the choice of the final parameters can be derived from the results of parameter variation.

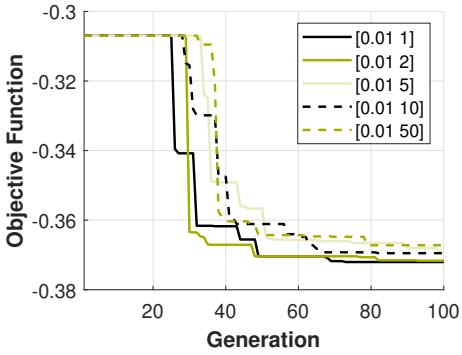
- **Swarm Size Variation:** From the results in Figure 4.4a, the complexity increment by PSO does not change much with the swarm size variation, especially for



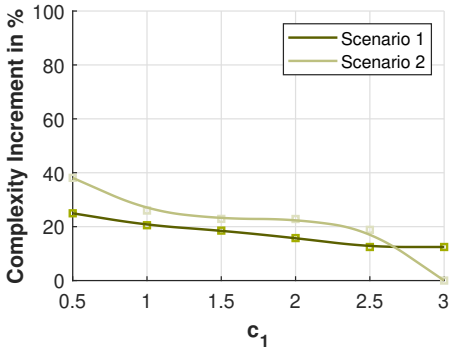
(a) Comparison of swarm size variation



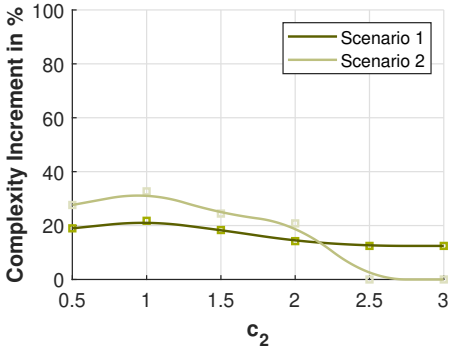
(b) Comparison of iteration variation



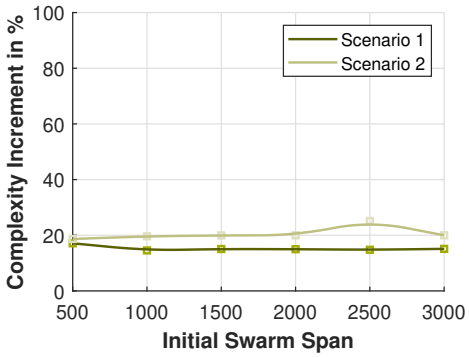
(c) Comparison of inertia range variation (Scenario 1)



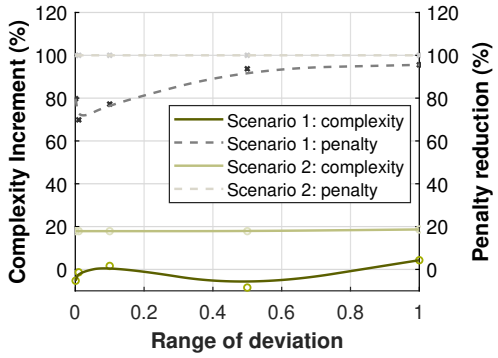
(d) Comparison of self adjustment weight variation



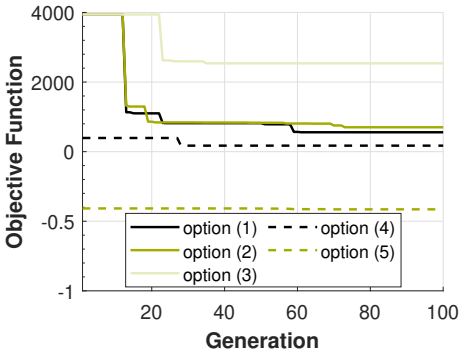
(e) Comparison of social adjustment weight variation



(f) Comparison of initial swarm span variation



(g) Comparison of the range of deviation in the option (4) of initialization



(h) Comparison of initialization method variation (Scenario 1)

Figure 4.4: Results of parameter variation using PSO under the setting in Table 3.13 and 3.14

Scenario 2. That is, PSO is not sensitive to the swarm size. Since larger swarm sizes require more function evaluations and increase the computing efforts for convergence, the swarm size N_{PSO} is finally assigned with 50 in the optimizer.

- **Iteration Variation:** Similar to GA, to balance the accuracy and complexity of the search algorithm, the number of iteration $T_{g,PSO}$ is recommended to 500 according to Figure 4.4b.
- **Inertia Range Variation:** The curves in Figure 4.4c show the optimization process under different inertia ranges. When the inertia range is too large, particles might easily pass over but without catching good solutions, since the range [0.01 5], [0.01 10] and [0.01 50] decrease slower than others. While with a smaller range the particles may not have sufficient opportunity to explore more space beyond local regions. As a result, the final inertia range w_0 is located in [0.01 2].
- **Self Adjustment Weight Variation:** As depicted in Figure 4.4d, it appears that the setting of 0.5 for self adjustment coefficient c_1 is appropriate for both scenarios, which behaves better than other values clearly.
- **Social Adjustment Weight Variation:** As shown in Figure 4.4e, the value of 1 for social adjustment weight c_2 is a relative better choice among the whole range.
- **Initial Swarm Span Variation:** The Figure 4.4f implies that the setting of the initial swarm span does not affect the optimization performance generally. Therefore, the default value 2000 in Matlab is used in the optimizer.
- **Initialization Method Variation:** Similar to that described in GA, five methods are also selected here to initialize the search space. Regarding option (4) in Figure 4.4g, it can be seen that this initialization method has no effect on increasing complexity, especially in Scenario 1 with a high initial penalty function. According to Figure 4.4h, PSO is not very sufficient at reducing the penalty function with the option (1)-(4) in Scenario 1. Option (5) can avoid the high penalty function at the beginning, which is regarded as the final choice for the initialization method.

In view of the above observations, the corresponding optimal hyper-parameters for PSO are outlined in Table 4.3:

Table 4.3: The optimal parameters of PSO

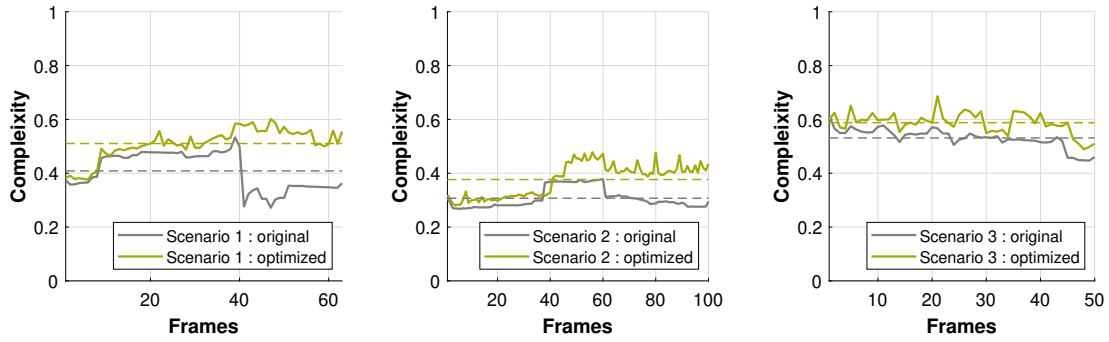
Parameter	Value
Swarm size (N_{PSO})	50
Number of Iterations ($T_{g,PSO}$)	500
Inertia Range (w_0)	[0.01 2]
Self Adjustment Weight (c_1)	0.5
Social Adjustment Weight (c_2)	1
Initial Swarm Span (Ini_{ss})	2000
Initial Swarm Matrix (Ini_m)	Option (5)

4.2.2 Scenario Test

Much the same as GA, the three pre-defined scenarios are tested using PSO with the optimal parameter that listed in Table 4.3. The optimization results of three scenarios, the distribution of complexity in each scenario, and the Kiviat-diagram are depicted in Table 4.4, Figure 4.5 and 4.6, respectively. It can be concluded that, the complexity increment of three scenarios are 24%, 22.8% and 10.7% respectively.

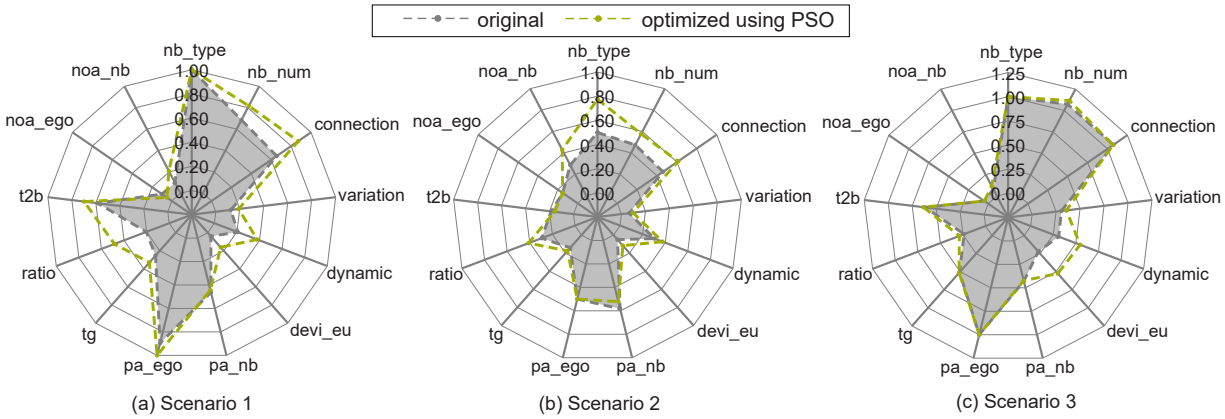
Table 4.4: The optimization results of three scenarios using PSO

	Scenario 1	Scenario 2	Scenario 3
Initial Complexity	0.4111	0.3070	0.5311
Final Complexity	0.5097	0.3765	0.5878
Initial Penalty Function	4020.2	0	20.0
Details	Collision between vehicles	-	Beyond jerk limitation
Final Penalty Function	0	0	0.0004
Details	-	-	Collision between vehicles
Running Time in s	14127.80	14568.41	16339.97



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 4.5: Complexity comparison of three scenarios



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

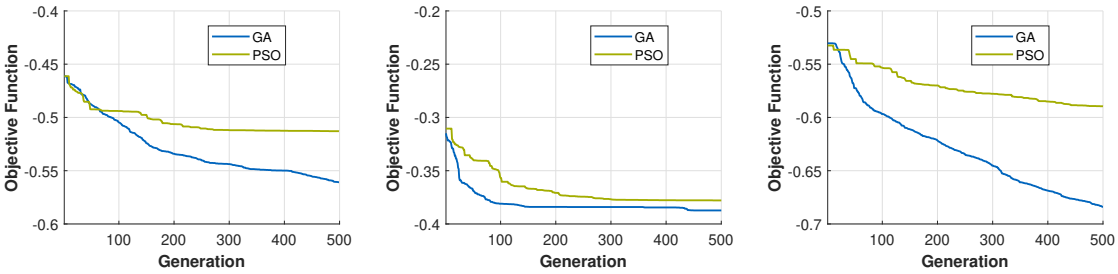
Figure 4.6: Kiviat-diagram to visualize the complexity of three scenarios

4.3 Discussion of the Results

Two performance tests for the optimizer were carried out for both EA algorithms under consideration, namely, PSO and GA. Next, this subsection will compare the results of the two optimization algorithms and discuss them quantitatively (in Section 4.3.1 and 4.3.2) and qualitatively (in Section 4.3.3 and 4.3.4).

4.3.1 Comparison of the Optimization Result

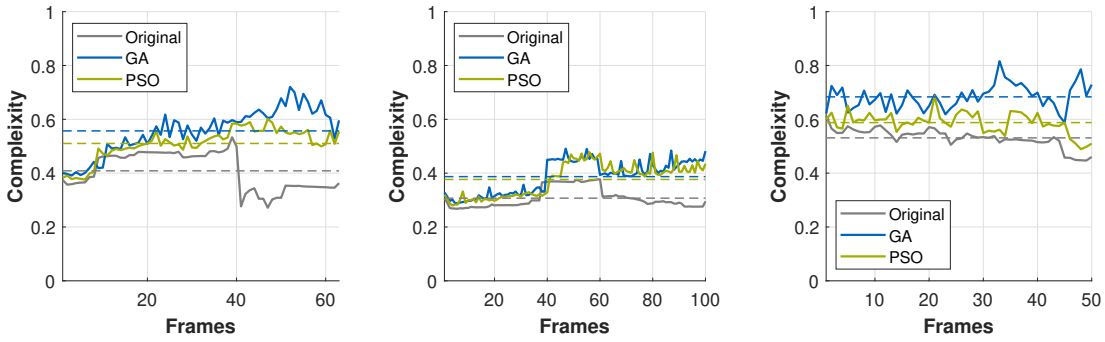
As introduced in Section 2.3, both GA and PSO can solve highly nonlinear, mixed integer optimization problems, which is proved by the aforementioned results. The curves in Figure 4.7 depict the optimization process and convergence situation of three scenario tests with two methods. It shows that GA has the highest convergence rate and yields the best value of objective function for all scenarios. However, it runs with expensive computational cost as shown in the Table 4.2 and 4.4, since it requires sorting the fitness value among the optimization process, which is not needed in PSO. Besides, in Scenario 1 and 3, GA did not fully converge at 500 generations, but according to the result of generation variation in Figure 4.1b, it changes basically only slightly when the number of generation continues to increase.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 4.7: Comparison of optimization process using GA and PSO respectively

Figure 4.8 depicts the complexity distribution in the whole scenarios using the two optimization methods. It can be seen that the difference of optimization effect of using GA and PSO in Scenario 2 is not obvious, whilst the performance of GA in Scenario 1 and 3 is better than that of PSO quantitatively.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 4.8: Complexity comparison of three scenarios

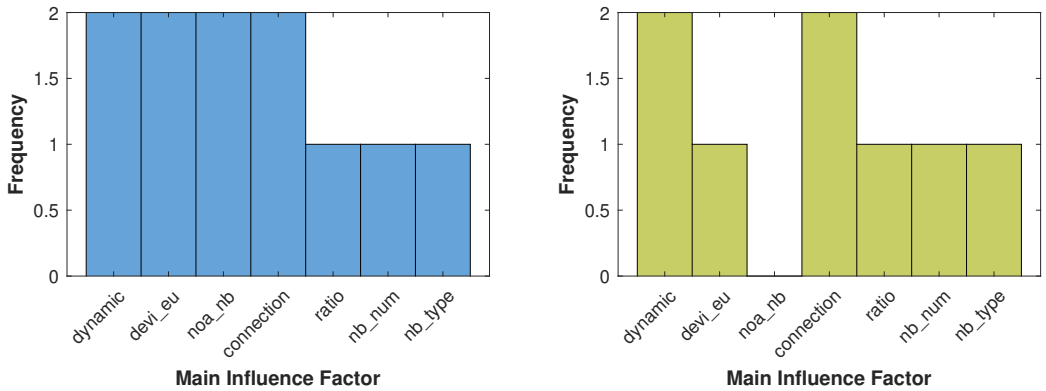
4.3.2 Quantitative Post-Verification

The performance of optimization not only lies in the final value of the objective function, but its quality also needs to be verified from the aspect of complexity. According to the categories in Table 3.15, we observe whether the complexity influence factors have really increased quantitatively as expected, which mainly focuses on the "intuitive" ones.

According to Figure 4.3 and 4.6, the optimizer has the ability to handle all given scenarios, even the one with high initial complexity (Scenario 3). It can be also observed that not every influence factor increases or changes during the optimization process, the main contributing factors can be summarized as Table 4.5 and Figure 4.9.

Table 4.5: Main contributing factors to complexity increment

	Scenario 1				Scenario 2		Scenario 3	
	Factor	Increment	Factor	Increment	Factor	Increment	Factor	Increment
GA	$dynamic$	+0.41	noa_nb	+0.17	nb_type	+0.30	$dynamic$	+0.71
	nb_num	+0.27	nb_type	+0.16	noa_nb	+0.23	$devi\_eu$	+0.66
	$devi\_eu$	+0.25			$connection$	+0.17		
	$ratio$	+0.22						
PSO	$ratio$	+0.28	$dynamic$	+0.17	nb_type	+0.27	$devi\_eu$	+0.30
	$connection$	+0.22	nb_num	+0.13	$connection$	+0.15	$dynamik$	+0.23



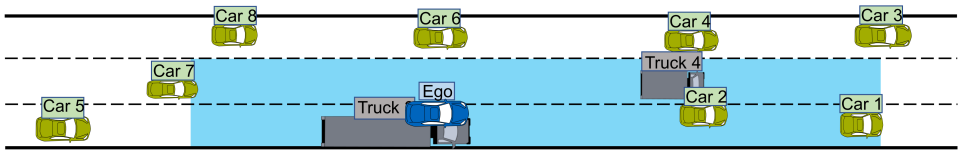
(a) GA (b) PSO

Figure 4.9: Histogram of main influence factors using GA and PSO respectively

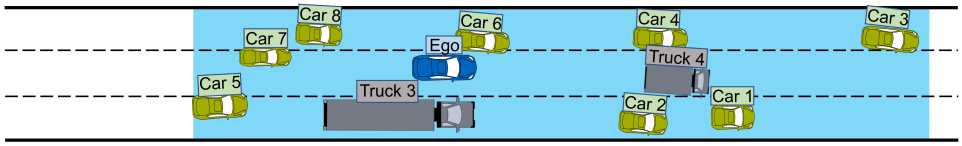
Obviously, the main contributing factors of the two optimization methods are similar, and basically the ones in PSO is a subset of GA according to Figure 4.9. And the large amount of change is almost caused by "intuitive" factors, such as $dynamic$, nb_num and etc. From this perspective, the optimization directions of the two methods are comparable, but GA generate a more sufficient result.

Next, we further analyze the "intuitive" influence factors to see if they really meet the requirements. For convenience, the following selects Scenario 1 with a more obvious change in complexity to compare and analyze the optimization quality of GA and PSO. It can be seen from Figure 4.2 that the maximum complexity of Scenario 1 after using GA optimization is at the 52nd frame, which has increased from 0.3528 to 0.7202. The specific configuration is shown in Figure 4.10.

Similarly, Figure 4.11 presents the 47th frame of Scenario 1 with the greatest complexity after using PSO. From Figure 4.5, the complexity at this frame increases from 0.2716 to 0.6013.

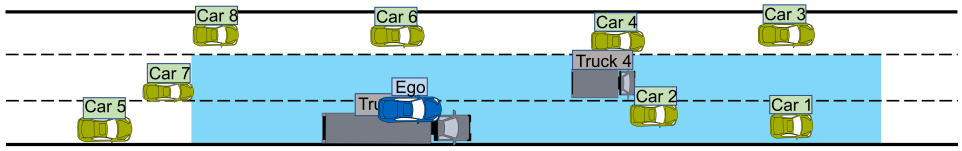


(a) Original with complexity 0.3528

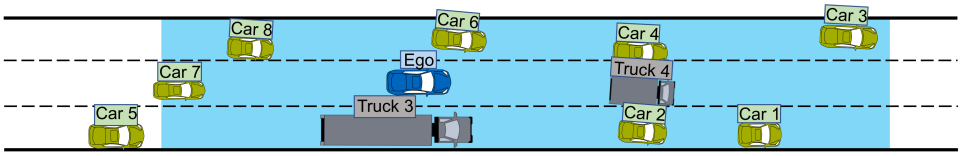


(b) Optimized using GA with complexity 0.7202

Figure 4.10: Configuration of 52nd frame of Scenario 1



(a) Original with complexity 0.2716



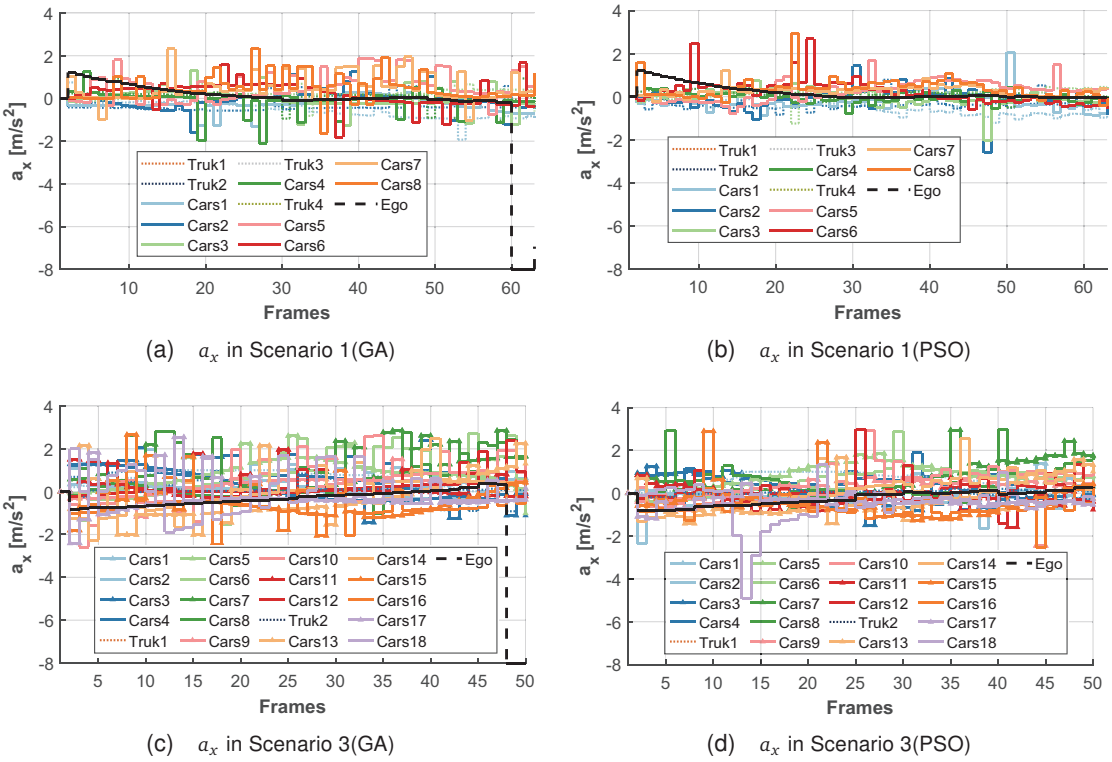
(b) Optimized using PSO with complexity 0.6013

Figure 4.11: Configuration of 47nd frame of Scenario 1

Intuitively, the differences of nb_type, nb_num and $ratio$ can be observed from Figure 4.10 and 4.11 directly. In original and optimized configuration, there are both two types of vehicles in ROI and the number of surrounding vehicles in the ROI has increased. And the number increased by GA is larger. Additionally, since $ratio$ represents the area that can not be reached by the sensor set, which is related to the non-transparency, obviously the beams generated by the ego-vehicle after optimization are occupied more than in the original one.

Regarding $variation$ and $dynamic$, the vehicle physical parameters play an important role, especially speed, acceleration and jerk. Firstly, it needs to be checked whether they meet the physical requirements after optimization according to Section 3.4. For reasons of space and clarity, the analysis results are listed in the Appendix A.2 in detail. It can be concluded that all velocity, acceleration and jerk values of surrounding vehicles not only varied more dramatically than before but also meet the requirements in Table 3.4 and 3.10 respectively, reflecting physical feasibility for both algorithms. Besides, vehicles in front of ego-vehicle are basically decelerating or driving at a quasi-constant speed, while vehicles behind ego are mainly accelerating. Notably is that under the influence of MOBIL rules, at the $60th$ frame of results from GA in Figure 4.12a, a_x of ego-vehicle (marked by black dash line) drops directly from -0.18 m/s^2 to -8m/s^2, which is caused by the setting of maximal deceleration in Section 3.1 and results in a abnormal high jerk in the longitudinal direction. Similarly, when using GA to optimize Scenario 3, this situation also appears in the 48th frame as shown in Figure 4.12c, which also obtains a larger value of $dynamic$ than using PSO. From this perspective, both GA and PSO show a good optimization

result in these two factors, whilst GA is likely to make the dynamic behaviours of the vehicles more complicated.



(a) a_x in Scenario 1(GA)

(b) a_x in Scenario 1(PSO)

(c) a_x in Scenario 3(GA)

(d) a_x in Scenario 3(PSO)

Figure 4.12: Comparison of longitudinal acceleration using GA and PSO

In the next step, we will further analyze some "complicated" factors. It can be clearly seen that the curves of ego-vehicle in Figure 4.12 is smoother than others because of using IDM, thus the factors highly related to ego have not changed much during the optimization, such as noa_ego and pa_ego. Besides, regarding the factor $devi\_eu$, the changes in Scenario 3 after using GA are relatively large. Explained by [8, p.67], high speed of ego-vehicle can lead to larger deviation because of longer prediction horizon compared to scenarios with low speed, which is consistent with the velocity situations of three scenarios as shown in Appendix A.2: the speed range of ego-vehicle in Scenario 1 after using GA is in [11.29, 15.41] m/s, Scenario 2 in [11.46, 15.88] m/s, and Scenario 3 in [19.50, 22.18] m/s.

Other factors either have not changed much after optimization in both methods, or are more complicated to understand directly. Moreover, the most influential factors are those that are more intuitive, thus the other factors in "medium" and "complicated" categories will not be discussed here. In summary, the results can show that the proposed GA method can not only have better optimization performance, but also obtain higher quality solution than PSO with the same optimization direction quantitatively.

4.3.3 Analysis of Exploitation and Exploration

After comparing the difference between the optimization results of two methods, the mechanism behind can be discussed qualitatively. The following will analyze the relationship of exploitation and exploration in GA and PSO respectively, according to the introduction in Section 3.6.4.

- **GA:** As described in [50], mutation in GA is needed to explore new states and helps the algorithm to avoid local optima, while crossover option increases the average quality of the population. Therefore, from the point of view in [51, p.35:5], a mutation operator is more of an exploration operator, whilst the selection of elite and crossover can be mainly seen as an exploitation operator. But in many cases, it is difficult to conclude whether newly created individuals created by a crossover and/or mutation operator will fall into the exploration or exploitation zones.

  In our optimizer, the final hyper-parameters of GA are listed in Table 4.1. From this we can conclude that the whole algorithm is more biased to explore at the beginning, since the value of $scale$ in the mutation option is 10, which means the first generation is expanded to a larger search space. And the initialization matrix A_0 adopts a method of pre-acceleration and deceleration, which provides more search possibilities for the algorithm initially. In the latter part of the search, exploitation dominates. The crossover fraction C_f is set to 80%, which indicates that more children are generated under the scattered combination of parents than through mutation. On the other hand, the setting of $shrink$ to 1 in the mutation option helps reduce the mutation deviation in the exploration period. There exists still a certain possibility of exploration, for instance, the elite ratio is relatively low, only 2 elites with good fitness are remained to the next generation and still 20% children are generated through mutation. In this way, a remarkable balance between exploration and exploitation of the search space in GA is achieved.

- **PSO:** Similar to GA, particle-based search in PSO is applied for high diversity during the early part of the search for global exploration of the full range of the search space. In the last stage of the search, local exploitation is achieved for more accuracy in the optimum solution.

  In this aspect, the algorithm attempts to balance exploration and exploitation by combining local and global search ability, that is represented by pbest and gbest in Algorithm 2. In particular, with appropriate initial swarm at the beginning such as initial swarm span and initial swarm matrix, the algorithm tries to explore more search space. After that, the setting of inertia range, coefficient c_1 and c_2 help the algorithm to achieve the refinement of the optimal solution, which can also be understood as exploitation. Since the mechanism is based on the floating point arithmetic, it could explore any potential values within the solution space.

Based on the aforementioned observation, one general interpretation is that EA should start with exploration and then gradually change into exploitation in the optimization process for our problem.

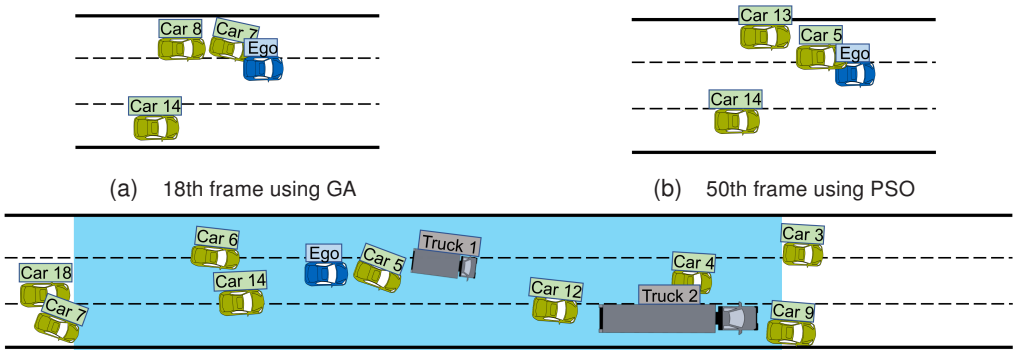
According to the optimization curves in Figure 4.7, it is clear that the positions of the starting point are similar for both algorithms, but the value of objective function in PSO converges slower than GA afterwards and the descent is sometimes step-wise. One explanation is that the possible space of the optimal solution is relatively small, since the initial configuration of the vehicles are already given, and the vehicles needs to drive within the ROI as long as possible, which should also obey the traffic rules and even maintain the physical feasibility. Thus the search algorithm must explore more at the beginning with sufficient solution possibilities, otherwise the efficiency of simply exploring is relatively low, that is the same for both algorithms. In the later exploitation phase, because initial population matrix given in the beginning is based on the original data in HighD and the change of acceleration values already meets the requirements of

jerk limitation, crossover between different fragments in parents can make the search process easier to exploit a better solution without introducing a high penalty function. But PSO does not have a crossover-like option. Although it can find a good start particle swarm initially through exploration, the swarm exploitation is then more based on random selection around the current solution, which is represented by the factor $rand$ in Eq. 2.6. Thus it is easy to stuck in a local optimal during the search process or explore more solutions that probably trigger high penalty function. As a result, GA exhibits better optimization efficiency than PSO under the exploration and exploitation framework.

4.3.4 Subjective Qualitative Evaluation

Although the above optimization results meet the quantitative requirements, through qualitative observation of the entire scenarios, it can still be found that the vehicles have many unrealistic behaviours.

In the first place, according to the final penalty function in Table 4.2 and 4.4, after optimization there are still vehicles that do not meet the location requirements in Section 3.4, i.e., collisions will occur between vehicles or the vehicles are driving too close, especially using GA. For example, Figure 4.13a and 4.13b show the above situations happened in Scenario 3 after optimization using two methods. Besides, after optimization with GA, many vehicles in the last frame have a relatively large orientation such as in the final configuration of Scenario 3 in Figure 4.13c, which is very strange and impractical. And there still exists a high probability of collision between vehicles as the scenario continues.



(a) 18th frame using GA

(b) 50th frame using PSO

(c) Last Frame using GA

Figure 4.13: Parts of penalized or unrealistic configurations of Scenario 3 after optimization

Next, the behaviours of ego-vehicle only depend mainly on the situation of heading vehicles according to IDM, which seems to be “dumb”. For instance, in Scenario 2, the distance between the ego-vehicle and "car3" at the beginning does not meet the time-gap requirements as shown in Figure 3.15. No matter before or after the optimization, the ego-vehicle always decelerates at the beginning, instead of overtaking or choosing other “intelligent” behaviours. Besides, observing the trajectories of the surrounding vehicles optimized by GA, some unrealistic behaviours can also be found. In Scenario 1 and 3 where the values of $dynamic$ are higher after optimization using GA, the lateral accelerations of the vehicles change more drastically. In this way, there will occur a impractical tremor movement of surrounding traffic as shown in Figure 4.14. And the vehicles change lanes only in a collision-free manner, such as "car7", which basically does not happen in actual situations.

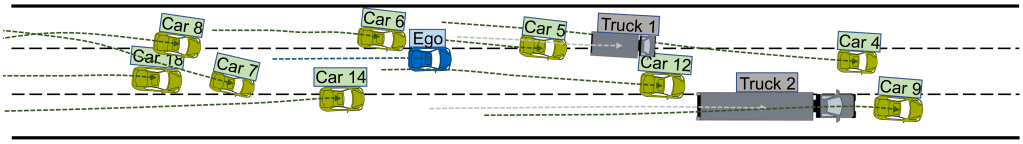


Figure 4.14: Trajectories of vehicles in Scenario 3 after optimization using GA

Overall, the optimization results using PSO are more in line with actual situations, although the complexity after optimization is not as high as GA. But these behaviours can be improved to some extent, for example, by setting a more complex and comprehensive penalty function. In order to pick out scenarios that can be used in practice, more work needs to be done in the future, such as smoothing the trajectories of surrounding traffic.

5 Summary and Outlook

In this chapter, the work of this thesis will firstly be summarized. In the second part, some possible extensions of this work will be discussed.

5.1 Summary

Theoretically, to secure and release AV, an infinite number of test cases must be checked to cover all scenarios of real traffic. The resulting test space is not manageable even with modern methods and simulation tools. One possibility is the scenario-based approach, which only takes the most interesting and relevant scenarios into account in order to limit the infinite test space to a finite number of test cases.

In this work, we present an optimization-based approach to generate more complex test scenarios for AV by means of EA. To this end, the original HighD dataset (Section 2.1) and corresponding complexity metrics (Section 2.2) are integrated to the optimizer through an interface in Section 3.2, which keeps their data structure and coordinate system highly consistent. Next, the model of ego-vehicle and surrounding vehicles are introduced in the simulator in Section 3.1 to generate the configuration of different scenarios. Moreover, with the purpose of allowing the vehicles to obey by traffic rules and maintain the physical feasibility, we introduced a penalty function (Section 3.4) and combine it with the final adapted complexity metrics (Section 3.3) to form an objective function in Section 3.5 to be optimized. Then an optimizer is constructed through the simulator of scenarios, the objective function, and its tuner. To avoid the collision between the vehicles caused by the change of the model of the AV, we also proposed the intelligent method to pre-process the scenario and initialize the search space with possible solutions, as shown in Section 3.6.2. Then we performed tuning experiments with two optimization algorithms (GA and PSO) in Section 3.6 to achieve better performance. After the analysis of the results of parameter variation, the optimal hyperparameters of the two methods are determined and utilized to test three example scenarios in Section 4.1 and 4.2. Under these prerequisites, this work then offers insight into the algorithm behavior of EA in Section 4.3, with the relationship of exploration and exploitation. In the end, after quantitatively and qualitatively discussing the results of optimization and post-verifying the physical feasibility, it can be concluded that GA offers better optimization performance and quality than PSO, which is more suitable for the optimizer, but the optimized result is still not realistic enough to be used in practice.

Therefore, unlike previous works, the optimization approach here is highly related to the actual highway situation and can ensure the physical possibilities of the motion of traffic participants through the defined objective function. It can also keep the relationship between exploration and exploitation in EA well balanced and achieve a sufficient optimization performance with the tuned parameters.

5.2 Outlook

Regarding future work, we see several promising directions. The most straightforward track is to extend the models in the optimizer. The existing AV model is relatively simple, IDM and MOBIL are only used to simulate the ACC in multiple lanes. Moreover, the implementation of MOBIL does not consider the shape and size of the vehicle, but regards the vehicle as a mass point, so that an undesirable lane change may occur sometimes. Criticality assessment requires ego-vehicle to be equipped with more functions, such as Lane Keep Assist (LKA), Auto Emergency Braking (AEB) and Collision Avoidance System (CAS). Besides, the selected scenarios are all with the straight road. In HighD and actual situations, there exist more complicated situations such as intersections. The current optimizer can then easily be projected to different road geometry, e.g., by setting more longitudinal and horizontal constraints for the surrounding traffic and ego-vehicle.

There is still a lot of work to be done on improving the performance of optimization in the future. Due to computational and time limitations, the number of scenarios used by the tuning of parameters is limited and is not represented to a certain extent, which can be explored in more depth later. Furthermore, the parameter values during the tuning of the optimizer is now selected to be constant at the design layer, which can also be realized by parameter control method, i.e., they undergo changes while the EA is running in the algorithm layer. For instance, reinforcement learning is a class of unsupervised machine learning approach, through which the trade-off problem between exploration and exploitation can be better solved while searching for the optimal behavior policy. This may be a way to improve the performance of PSO, which has excellent computing efficiency. Additionally, in this thesis, the influencing factors of complexity are equal-weighted, and there is actually a coupling relationship between different influence factors, e.g., nb_num and $connection$. Thus there still exists a potential possibility to improve the optimization performance by decoupling. Last but not least, there are still many unrealistic points from subjective evaluation introduced in section 4.3.4, which can be solved by adding more constraints in the optimization process or introducing new vehicle models.

List of Figures

List of Tables

Bibliography

[1] Eduardo Villalobos, „Generation of Complex Test Scenarios for Automated Vehicles by Means of Machine Learning," Master Thesis, Mechanical engineering, Technical University of Munich, Munich.

[2] Sebastian Mayr, „Generation of Complex Test Scenarios for Automated Vehicles by Means of Particle-Swarm-Optimization," Master Thesis, mechanical engineering, Technical University of Munich, Munich.

[3] B. Kaltenhäuser, K. Werdich, F. Dandl and K. Bogenberger, „Market development of autonomous driving in Germany," *Transportation Research Part A: Policy and Practice*, vol. 132, pp. 882–910, 2020, doi: 10.1016/j.tra.2020.01.001.

[4] B. Kim, Y. Kashiba, S. Dai and S. Shiraishi, „Testing Autonomous Vehicle Software in the Virtual Prototyping Environment," *IEEE Embedded Systems Letters*, vol. 9, no. 1, pp. 5–8, 2017, doi: 10.1109/LES.2016.2644619.

[5] „. *After 2019's reality check, what's ahead for driverless cars in 2020? – TechCrunch 2020.4.16*," 2020-4-16. Available: https://techcrunch.com/2020/01/02/after-2019s-reality-check-whats-ahead-for-driverless-cars-in-2020/.

[6] Y. Wang, Z. Su, K. Zhang and A. Benslimane, „Challenges and Solutions in Autonomous Driving: A Blockchain Approach," *IEEE Network*, pp. 1–9, 2020, doi: 10.1109/MNET.001.1900504.

[7] S. Jannifer. „*SAE J3016 automated-driving graphic: SAE Standards News: J3016 automated-driving graphic update*," 2020-4-16. Available: https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic.

[8] X. Yu, „Method for quantitative evaluation of traffic complexity on the highway," Semester Thesis, mechanical engineering, Technical University of Munich, Munich, 2019.

[9] X. Yu, „Improvements and analysis of traffic complexity evaluation method in highway scenario for automated vehicles," Master Thesis, mechanical engineering, Technical University of Munich, Munich, 2020.

[10] M. Breitfuß, „Extraktion komplexer Verkehrsszenarien aus Realdaten mittels Clusteranalyseverfahren und deren Optimierung," Semester Thesis, mechanical engineering, Technical University of Munich, Munich, 2020.

[11] R. Krajewski, J. Bock, L. Kloeker and L. Eckstein, eds. „*The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems*," (Maui, HI). IEEE, 11/4/2018 - 11/7/2018. isbn: 978-1-7281-0321-1. doi: 10.1109/ITSC.2018.8569552.

[12] „. *The highD Dataset*," 2020-3-26. Available: https://www.highd-dataset.com/format.

[13] Carlos Gershenson, *Complexity: 5 Questions*, Automatic Press/VIP, 2008.

[14] G. Bagschik, T. Menzel and M. Maurer, „Ontology based Scene Creation for the Development of Automated Vehicles," Available: http://arxiv.org/pdf/1704.01006v5.

[15] F. Schuldt, „Ein Beitrag fur den methodischen Test von automatisierten Fahrfunktionen mit Hilfe von virtuellen Umgebungen - English title: Towards testing of automated driving functions in virtual driving environments Towards testing of automated driving functions in virtual driving environments," Ph.D.dissertation, Technische Universitat Braunschweig, Braunschweig, 2017.

[16] G. Bagschik, T. Menzel and M. Maurer, „Ontology based Scene Creation for the Development of Automated Vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV 2018): Changshu, Suzhou, China, 26-30 June 2018*, Changshu, 2018, pp. 1813–1820, isbn: 978-1-5386-4452-2. doi: 10.1109/IVS.2018.8500632.

[17] J. Wang, C. Zhang, Y. Liu and Q. Zhang, „Traffic Sensory Data Classification by Quantifying Scenario Complexity," in *2018 IEEE Intelligent Vehicles Symposium (IV 2018): Changshu, Suzhou, China, 26-30 June 2018*, Changshu, 2018, pp. 1543–1548, isbn: 978-1-5386-4452-2. doi: 10.1109/IVS.2018.8500669.

[18] A.-M. Jacobo, O. Koichiro, K. Eiichi and T. Satoshi, „Development of a safety assurance process for autonomous vehicles in japan," 2019. Available: https://www-esv.nhtsa.dot.gov/Proceedings/26/26ESV-000286.pdf.

[19] P. A. Vikhar, „Evolutionary algorithms: A critical review and its future prospects," in *ICGTSPICC 2016: Proceedings : International Conference on Global Trends in Signal Processing, Information Computing and Communication : 22-24 December 2016, Jalgaon, India*, Jalgaon, India, 2016, pp. 261–265, isbn: 978-1-5090-0467-6. doi: 10.1109/ICGTSPICC.2016.7955308.

[20] M. Klischat and M. Althoff, „Generating Critical Test Scenarios for Automated Vehicles with Evolutionary Algorithms," in *2019 IEEE Intelligent Vehicles Symposium (IV 2019): Paris, France 9-12 June 2019*, Paris, France, 2019, pp. 2352–2358, isbn: 978-1-7281-0560-4. doi: 10.1109/IVS.2019.8814230.

[21] Matlab. „Global Optimization Toolbox User's Guide: R2020a," [Online]. Available: https://www.mathworks.com/help/gads/.

[22] D. B. Fogel, D. Liu and J. M. Keller, *Fundamentals of Computational Intelligence*, Hoboken, NJ, USA, John Wiley & Sons, Inc, 2016, isbn: 9781119214403. doi: 10.1002/9781119214403.

[23] D. Wang, D. Tan and L. Liu, „Particle swarm optimization algorithm: an overview," *Soft Computing*, vol. 22, no. 2, pp. 387–408, 2018, doi: 10.1007/s00500-016-2474-6.

[24] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, „Optimization by simulated annealing," *Science (New York, N.Y.)*, vol. 220, no. 4598, pp. 671–680, 1983, doi: 10.1126/science.220.4598.671.

[25] „. *Simulated annealing*," 2020-4-21. Available: https://en.wikipedia.org/w/index.php?title=Simulated_annealing&oldid=952273439 [visited on].

[26] N. V. Findler, C. Lo and R. Lo, „Pattern search for optimization," *Mathematics and Computers in Simulation*, vol. 29, no. 1, pp. 41–50, 1987, doi: 10.1016/0378-4754(87)90065-6.

[27] Z. Hu, Y. Bao and T. Xiong, „Electricity load forecasting using support vector regression with memetic algorithms," *TheScientificWorldJournal*, vol. 2013, p. 292575, 2013, doi: 10.1155/2013/292575.

[28] L. M. Rios and N. V. Sahinidis, „Derivative-free optimization: a review of algorithms and comparison of software implementations," *Journal of Global Optimization*, vol. 56, no. 3, pp. 1247–1293, 2013, doi: 10.1007/s10898-012-9951-y.

[29] Hugh Durrant-Whyte, Nicholas Roy and Pieter Abbeel, „Fast Trajectory Correction for Nonholonomic Mobile Robots Using Affine Transformations," in *Robotics: Science and systems VII*, H. F. Durrant-Whyte, N. Roy and P. Abbeel, ed. Cambridge, Mass.: MIT Press, op. 2012, pp. 265–272, isbn: 9780262305969. Available: http://ieeexplore.ieee.org/document/6301042.

[30] R. Malinauskas, „The Intelligent Driver Model: Analysis and Application to Adaptive Cruise Control," Thesis, Mathematics, Clemson University, 2014. Available: https://tigerprints.clemson.edu/all_theses/1934 [visited on 05/2014].

[31] C. Schmidt, *Hardware-in-the-Loop-gestützte Entwicklungsplattform für Fahrerassistenzsysteme – Modellierung und Visualisierung des Fahrzeugumfeldes*, 1st ed., Göttingen, Cuvillier Verlag, 2011, isbn: 9783869557274. Available: https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=5020338.

[32] A. Kesting, M. Treiber and D. Helbing, „General Lane-Changing Model MOBIL for Car-Following Models," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1999, no. 1, pp. 86–94, 2007, doi: 10.3141/1999-10.

[33] P. Junietz, F. Bonakdar, B. Klamann and H. Winner, „Criticality Metric for the Safety Validation of Automated Driving using Model Predictive Trajectory Optimization," pp. 60–65, doi: 10.1109/ITSC.2018.8569326.

[34] S. J. Russell and P. Norvig, *Artificial intelligence: A modern approach / Stuart J. Russell and Peter Norvig ; contributing writers, Ernest Davis [and seven others]*, (Prentice Hall series in artificial intelligence), Third edition, global edition, Boston, Pearson, 2016, isbn: 1292153962.

[35] G. Underwood, *Traffic and Transport Psychology: Theory and Application*, 1. Aufl., s.l., Elsevier professional, 2005, isbn: 0080443796. Available: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10190084.

[36] A. Loulizi, Y. Bichiou and H. Rakha, „Steady-State Car-Following Time Gaps: An Empirical Study Using Naturalistic Driving Data," *Journal of Advanced Transportation*, vol. 2019, no. 3, pp. 1–9, 2019, doi: 10.1155/2019/7659496.

[37] G. M. Arnaout and S. Bowling, „A Progressive Deployment Strategy for Cooperative Adaptive Cruise Control to Improve Traffic Dynamics," *International Journal of Automation and Computing*, vol. 11, no. 1, pp. 10–18, 2014, doi: 10.1007/s11633-014-0760-2.

[38] H. Burg and A. Moser, *Handbuch Verkehrsunfallrekonstruktion: Unfallaufnahme, Fahrdynamik, Simulation*, (ATZ / MTZ-Fachbuch), Wiesbaden, Springer Fachmedien Wiesbaden, 2017, isbn: 9783658161422. Available: https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=4858696.

[39] Reference. „What Is the Width and Length of the Average Car?," [Online]. Available: https://www.reference.com/world-view/width-length-average-car-9eb7b00283fb1bd8.

[40] T. Toledo and D. Zohar, „Modeling Duration of Lane Changes," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1999, no. 1, pp. 71–78, 2007, doi: 10.3141/1999-08.

[41] UNITED NATIONS, „Uniform provisions concerning the approval of vehicles with regard to steering equipment: Addendum 78: UN Regulation No. 79," p. 21, 2018.

[42] F. Greis. „*Wie die Uber-Software den tödlichen Unfall begünstigte*," 6.11.2019. Available: https://www.golem.de/news/ermittlungsberichte-wie-die-uber-software-den-toedlichen-unfall-beguenstigte-1911-144832.html.

[43] A. E. Eiben and S. K. Smit, „Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 19–31, 2011, doi: 10.1016/j.swevo.2011.02.001.

[44] F. Jia and D. Lichti, „a Comparison of Simulated Annealing, Genetic Algorithm and Particle Swarm Optimization in Optimal First-Order Design of Indoor Tls Networks," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-2/W4, pp. 75–82, 2017, doi: 10.5194/isprs-annals-IV-2-W4-75-2017.

[45] D. P. Chau, M. Thonnat, F. Brémond and E. Corvée, „Online parameter tuning for object tracking algorithms," *Image and Vision Computing*, vol. 32, no. 4, pp. 287–302, 2014, doi: 10.1016/j.imavis.2014.02.003.

[46] Y. He, W. J. Ma and J. P. Zhang, „The Parameters Selection of PSO Algorithm influencing On performance of Fault Diagnosis," *MATEC Web of Conferences*, vol. 63, no. 3, p. 02019, 2016, doi: 10.1051/matecconf/20166302019.

[47] L. Lin and M. Gen, „Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation," *Soft Computing*, vol. 13, no. 2, pp. 157–168, 2009, doi: 10.1007/s00500-008-0303-2.

[48] T. Chen, K. Tang, G. Chen and X. Yao, „A large population size can be unhelpful in evolutionary algorithms," *Theoretical Computer Science*, vol. 436, pp. 54–70, 2012, doi: 10.1016/j.tcs.2011.02.016.

[49] E. K. Burke, S. Gustafson and G. Kendall, „Diversity in Genetic Programming: An Analysis of Measures and Correlation With Fitness," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 47–62, 2004, doi: 10.1109/TEVC.2003.819263.

[50] G. Zhao, W. Luo, H. Nie and C. Li, „A Genetic Algorithm Balancing Exploration and Exploitation for the Travelling Salesman Problem," in *Fourth International Conference on Natural Computation, 2008: ICNC '08 : 18 - 20 Oct. 2008, Jinan, China ; held jointly with the 5th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2008)*, Jinan, Shandong, China, 2008, pp. 505–509, isbn: 978-0-7695-3304-9. doi: 10.1109/ICNC.2008.421.

[51] M. Črepinšek, S.-H. Liu and M. Mernik, „Exploration and exploitation in evolutionary algorithms," *ACM Computing Surveys*, vol. 45, no. 3, pp. 1–33, 2013, doi: 10.1145/2480741.2480752.

Appendix

A Appendix

In this chapter, The supplementary information required for the analysis and discussion in the third and fourth chapters is enumerated, including the analysis of initial configuration of vehicles and their physical parameters before and after optimization.

A.1 Initial Configuration of vehicles

In this section, the initial configuration of vehicles of different scenarios used in this thesis is introduced.

A.1.1 Scenario 1

The initial configurations of vehicles in Scenario 1 are shown in Table A.1 and A.2.

Table A.1: Initial configuration of ego-vehicle in Scenario 1

Item	Details
Position in m	(3.78, 5.26)
Velocity in m/s	(13.41, 0)
Orientation in °	0
Size of bounding boxes in m	(4.85, 2.02)

Table A.2: Initial configuration of surrounding vehicles in Scenario 1

Name	Position in m	Velocity in m/s	Orientation in °	Size of bounding box in m
Truck 1	(189.21 ,9.76)	(10.93, 0.03)	0.16	(8.19, 2.50)
Truck 2	(153.10, 9.67)	(10.86, 0.01)	0.05	(16.88, 2.50)
Truck 3	(42.90, 9.78)	(9.05, -0.05)	-0.31	(16.17, 2.50)
Truck 4	(30.37, 5.64)	(14.20, 0.04)	0.16	(7.18, 2.50)
Car 1	(82.61, 9.58)	(10.51, -0.05)	-0.27	(4.24, 1.72)
Car 2	(65.44, 8.58)	(10.64, -0.05)	-0.27	(4.14, 1.72)
Car 3	(50.73, 1.23)	(14.86, 0.02)	0.07	(4.55, 1.72)
Car 4	(28.56, 0.04)	(15.01, 0.01)	0.03	(4.85, 1.77)
Car 5	(7.30, 9.71)	(9.95, -0.08)	-0.46	(4.45, 1.82)
Car 6	(-2.02, 1.72)	(14.56, 0.00)	0.00	(4.75, 1.92)
Car 7	(-21.98, 6.36)	(13.97, 0.00)	0.00	(4.75, 1.92)
Car 8	(-24.61, 1.86)	(14.83, -0.09)	-0.35	(4.45, 1.82)

A.1.2 Scenario 2

Table A.3 and A.4 introduce the initial configuration of ego- and surrounding vehicles in Scenario 2 respectively.

Table A.3: Initial configuration of ego-vehicle in Scenario 2

Item	Details
Initial position in m	(2.81, 1.29)
Initial velocity in m/s	(12.48, 0.09)
Initial orientation in °	0.41
Size of bounding boxes in m	(4.35, 1.82)

Table A.4: Initial configuration of surrounding vehicles in Scenario 2

Name	Position in m	Velocity in m/s	Orientation in °	Size of bounding box in m
Truck 1	(65.71, 5.88)	(11.31, -0.02)	-0.10	(12.94, 2.50)
Car 1	(47.06, 1.69)	(13.03, -0.06)	-0.26	(3.89, 1.72)
Car 2	(29.86, 6.02)	(11.19, 0.04)	0.20	(4.65, 1.92)
Car 3	(18.48, 2.03)	(12.66, 0.04)	0.18	(5.26, 2.22)
Car 4	(15.54, 6.20)	(11.36, 0.04)	0.20	(4.14, 1.82)
Car 5	(-2.33, 5.89)	(9.95, -0.08)	-0.46	(4.04, 1.71)
Car 6	(-23.35, 0.99)	(12.45, 0.09)	0.41	(4.75, 1.82)

A.1.3 Scenario 3

According to Table A.5 and A.6, the initial configuration of vehicles in Scenario 3 can be found.

Table A.5: Initial configuration of ego-vehicle in Scenario 3

Item	Details
Initial position in m	(55.57, 5.54)
Initial velocity in m/s	(22.18, -0.11)
Initial orientation in °	-0.28
Size of bounding boxes in m	(4.14, 1.92)

A.2 Analysis of physical Parameters

In this section, the changing processes of horizontal and vertical speeds, accelerations, and jerks of all vehicles in three scenarios originally and using GA, PSO will be depicted sequentially, which supports the quantitative analysis in Section 4.3.2.

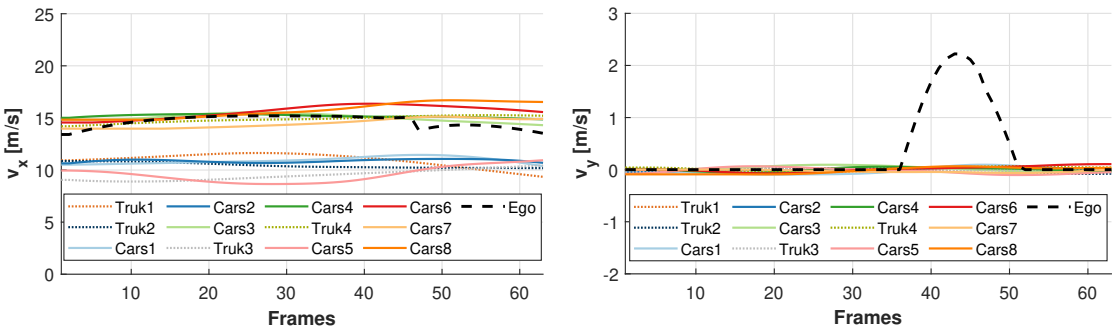
A.2.1 Scenario 1

Figure A.1-A.3 depict the physical parameters of vehicles in Scenario 1 respectively.

Table A.6: Initial configuration of surrounding vehicles in Scenario 3

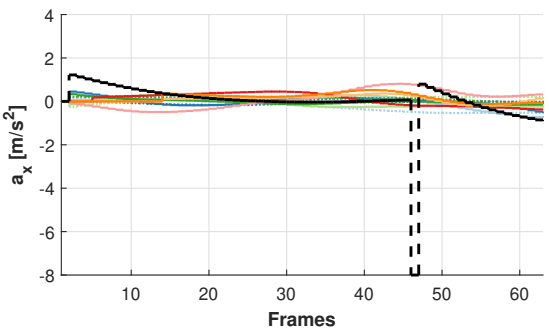
Name	Position in m	Velocity in m/s	Orientation in °	Size of bounding box in m
Truck 1	(156.20, 1.63)	(6.93, 0.05)	0.41	(7.07, 2.22)
Truck 2	(92.16, 9.70)	(20.41, 0.02)	0.05	(16.88, 2.50)
Car 1	(221.93, 2.47)	(9.83, -0.07)	-0.41	(4.65, 2.02)
Car 2	(205.79, 1.75)	(10.43, -0.05)	-0.27	(4.55, 1.82)
Car 3	(187.38, 2.66)	(8.84, -0.10)	-0.65	(4.45, 1.92)
Car 4	(174.02, 1.26)	(7.23, -0.02)	-0.16	(4.14, 1.92)
Car 5	(144.55, 2.32)	(6.85, 0.01)	0.08	(4.55, 1.82)
Car 6	(134.08, 1.89)	(7.21, 0.05)	0.40	(5.05, 1.92)
Car 7	(117.18, 1.64)	(6.48, -0.01)	-0.09	(4.04, 1.72)
Car 8	(108.78, 2.07)	(5.67, -0.01)	-0.10	(4.24, 1.82)
Car 9	(126.10, 6.00)	(18.17, -0.08)	-0.25	(4.14, 1.82)
Car 10	(97.70, 2.25)	(6.25, -0.17)	-1.56	(4.95, 2.02)
Car 11	(126.52, 10.55)	(23.31, 0.11)	0.27	(4.14, 1.82)
Car 12	(93.29, 6.24)	(19.00, 0.12)	0.36	(4.85, 2.02)
Car 13	(69.74, 1.46)	(10.18, 0.04)	0.22	(4.65, 1.92)
Car 14	(47.89, 9.92)	(21.31, 0.13)	0.34	(3.94, 1.92)
Car 15	(54.66, 1.14)	(9.81, -0.08)	-0.47	(5.05, 2.02)
Car 16	(13.66, 1.46)	(11.64, 0.00)	0.00	(4.85, 1.82)
Car 17	(38.18, 2.31)	(9.76, -0.02)	-0.11	(5.05, 2.02)
Car 18	(19.42, 5.33)	(20.44, 0.03)	0.08	(4.95, 1.92)

Original:

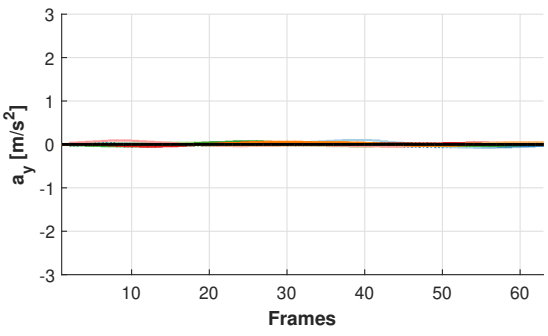


(a) v_x

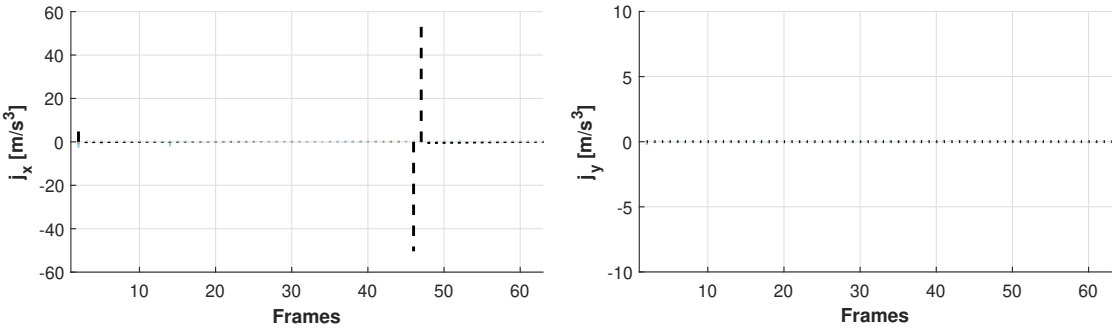
(b) v_y



(c) a_x



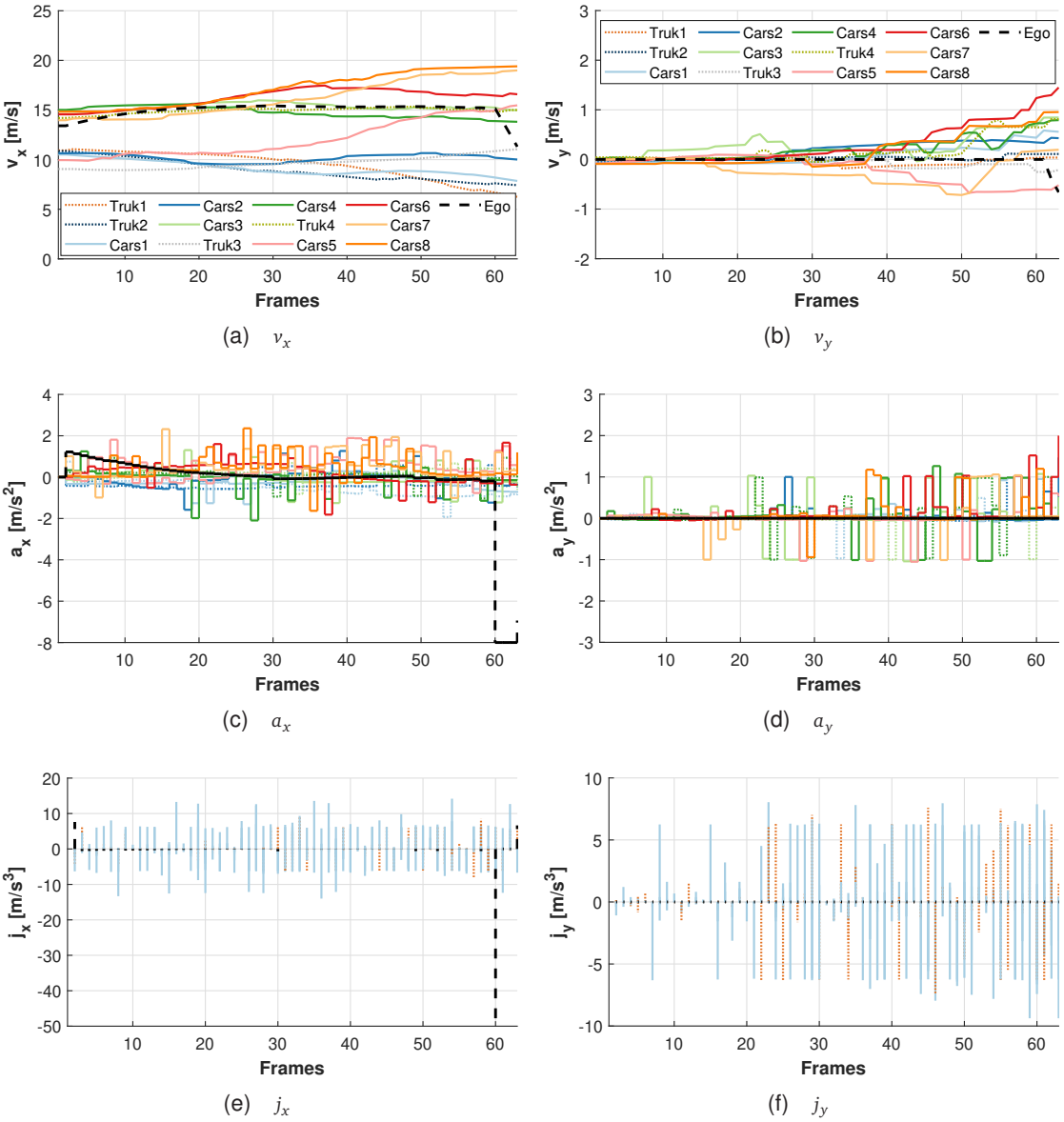
(d) a_y



(e) j_x (f) j_y

Figure A.1: Analysis of physical parameters in original Scenario 1

Using GA:



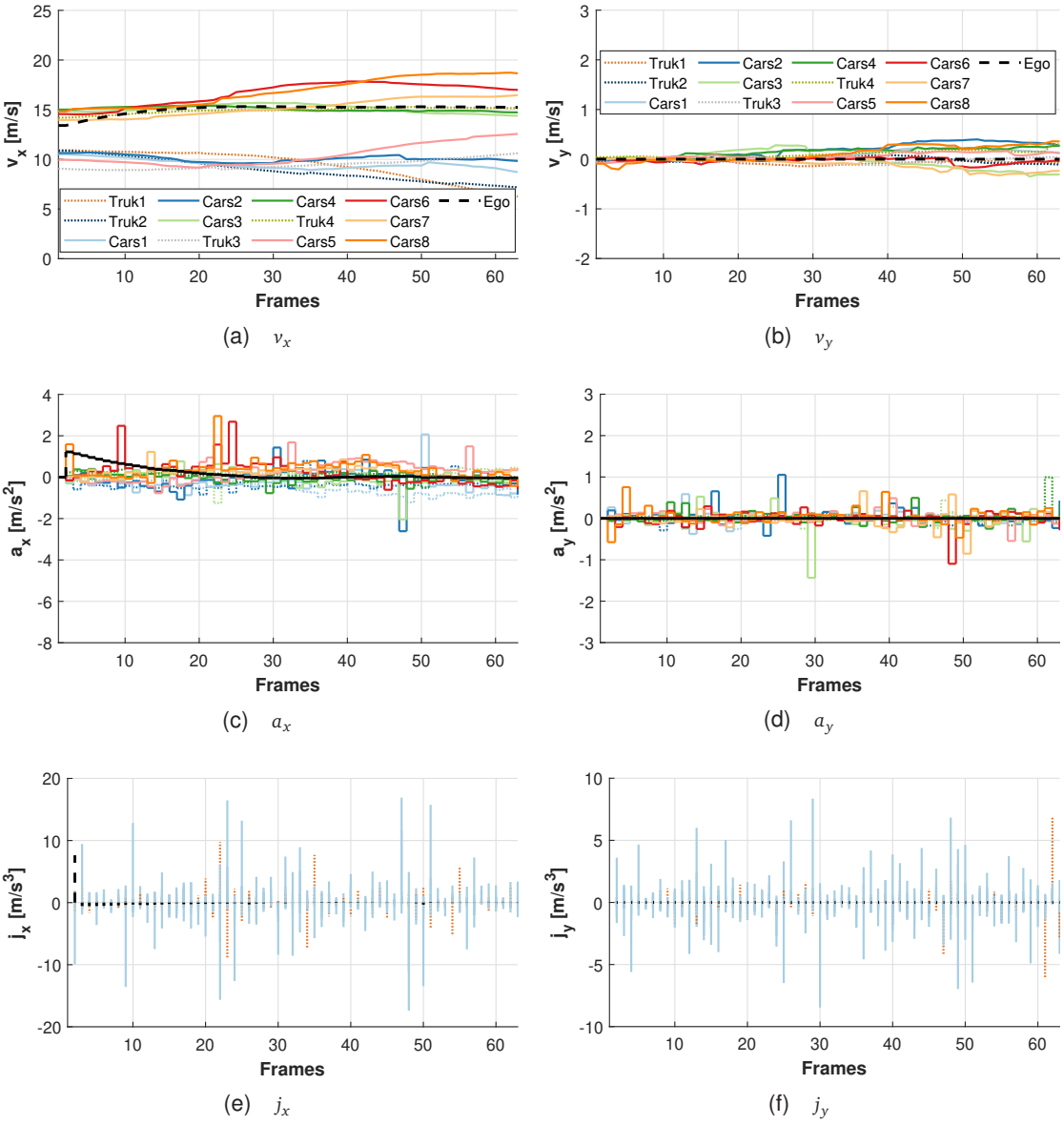
(a) v_x (b) v_y

(c) a_x (d) a_y

(e) j_x (f) j_y

Figure A.2: Analysis of physical parameters in Scenario 1 after using GA

Using PSO:



(a) v_x (b) v_y

(c) a_x (d) a_y

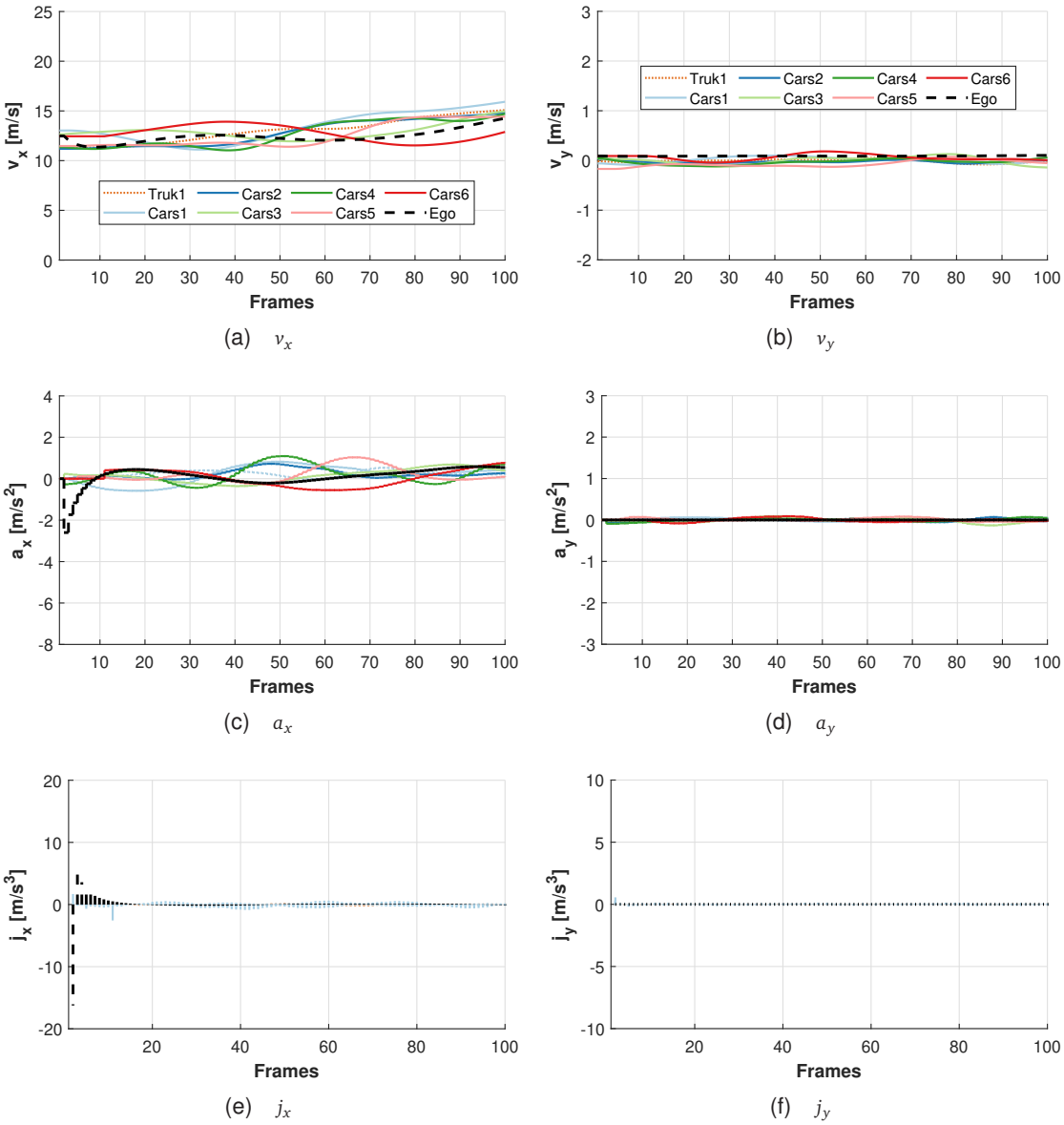
(e) j_x (f) j_y

Figure A.3: Analysis of physical parameters in Scenario 1 after using PSO

A.2.2 Scenario 2

In the following figures A.4-A.6 the physical parameters in Scenario 2 will be illustrated.

Original:



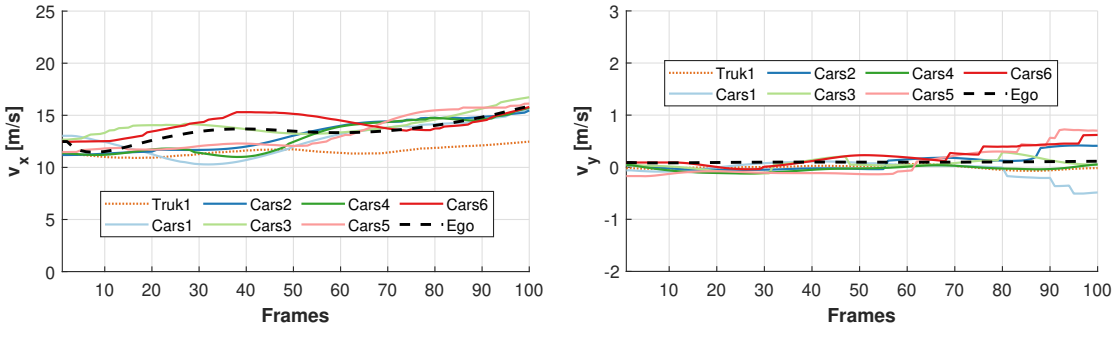
(a) v_x (b) v_y

(c) a_x (d) a_y

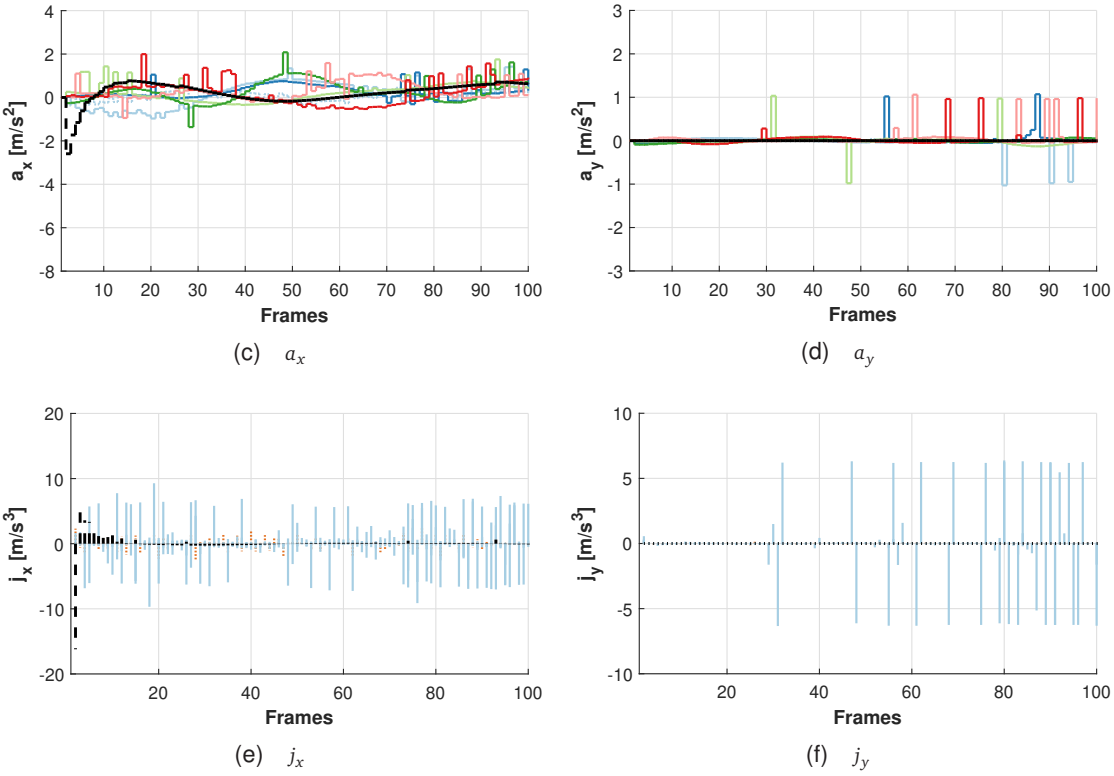
(e) j_x (f) j_y

Figure A.4: Analysis of physical parameters in original Scenario 2

Using GA:



(a) v_x (b) v_y

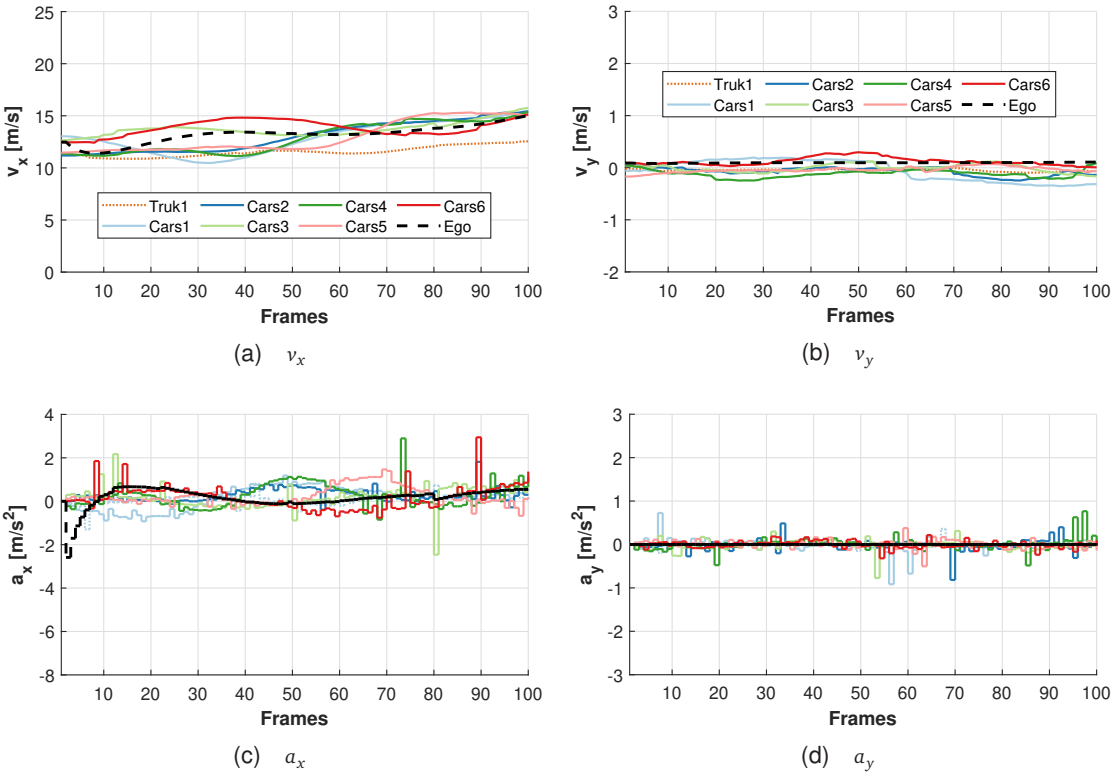


(c) a_x (d) a_y

(e) j_x (f) j_y

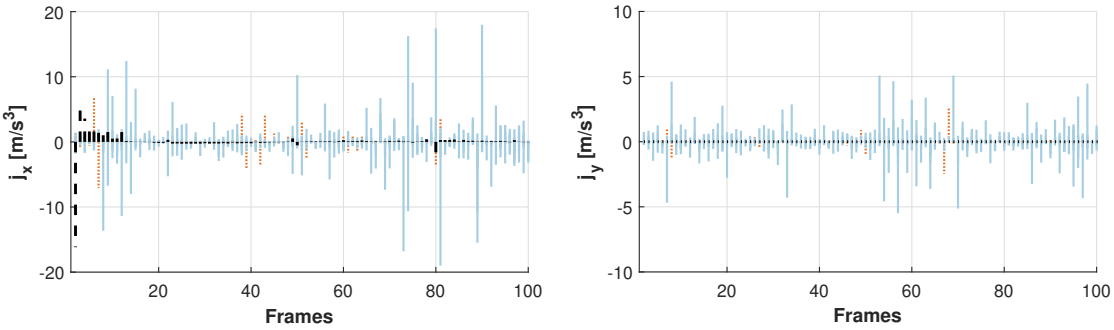
Figure A.5: Analysis of physical parameters in Scenario 2 after using GA

Using PSO:



(a) v_x (b) v_y

(c) a_x (d) a_y



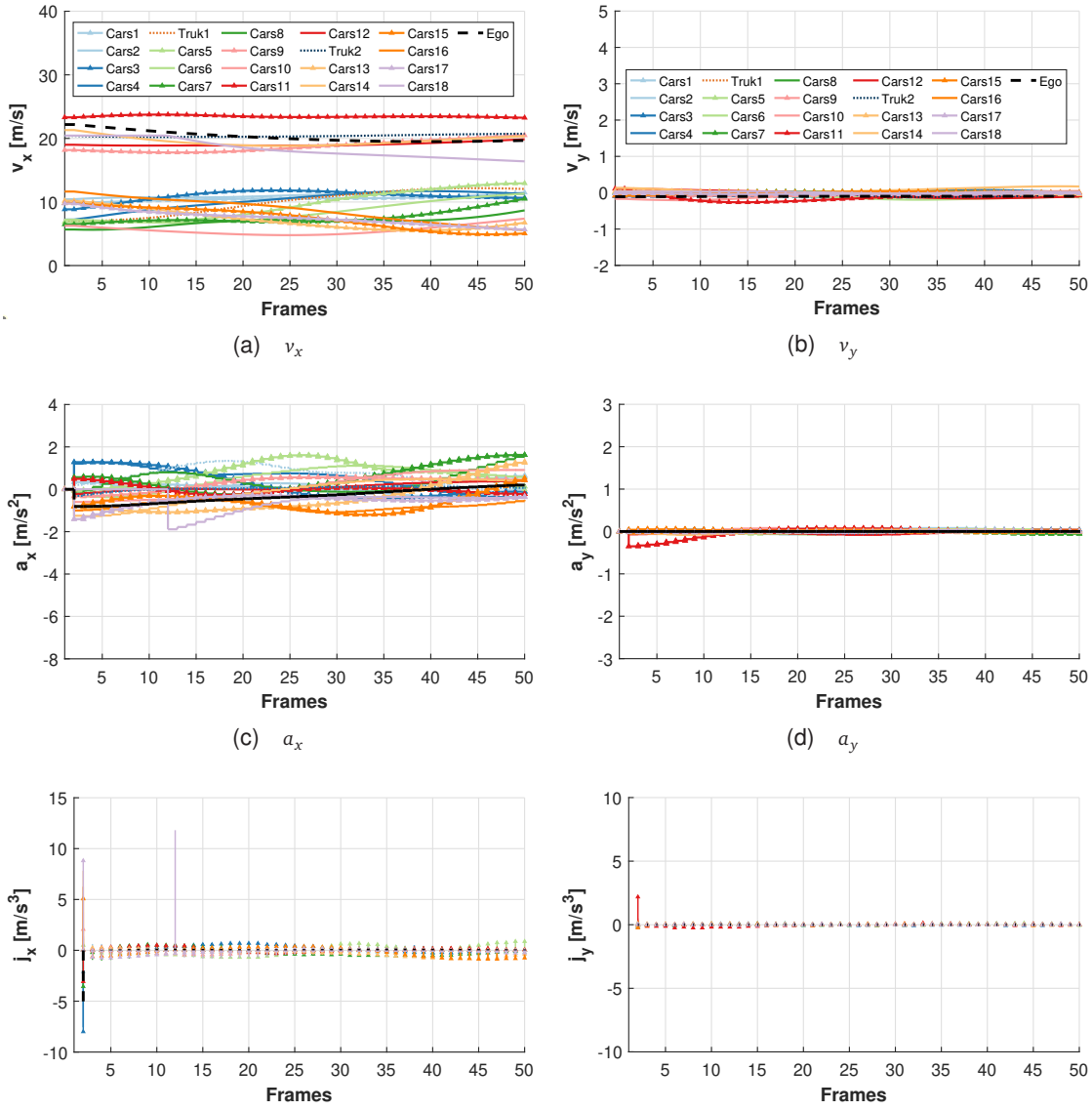
(e) j_x (f) j_y

Figure A.6: Analysis of physical parameters in Scenario 2 after using PSO

A.2.3 Scenario 3

Similarly, Figure A.7-A.9 show the physical parameters of Scenario 3.

Original:



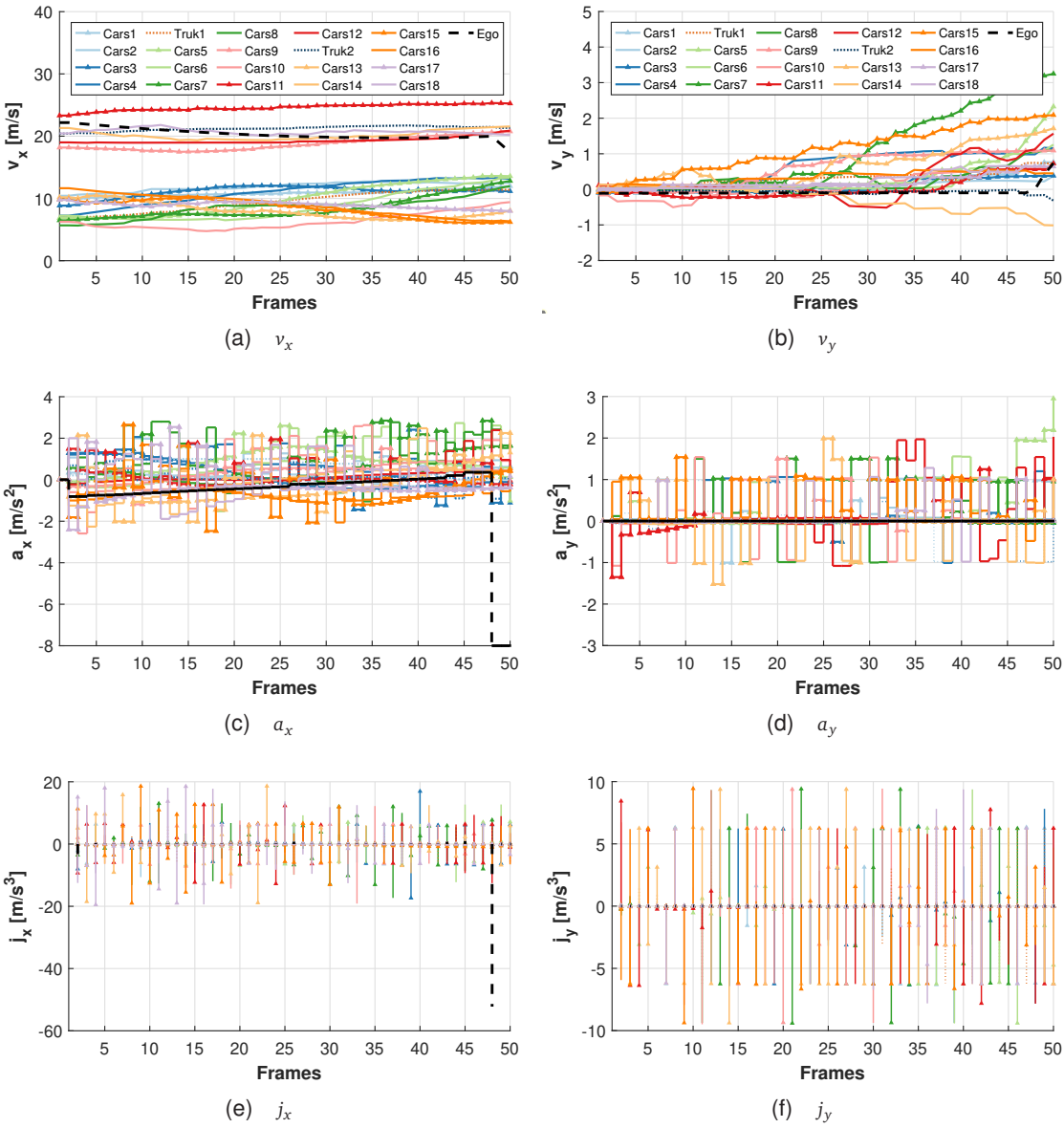
(a) v_x (b) v_y

(c) a_x (d) a_y

(e) j_x (f) j_y

Figure A.7: Analysis of physical parameters in original Scenario 3

Using GA



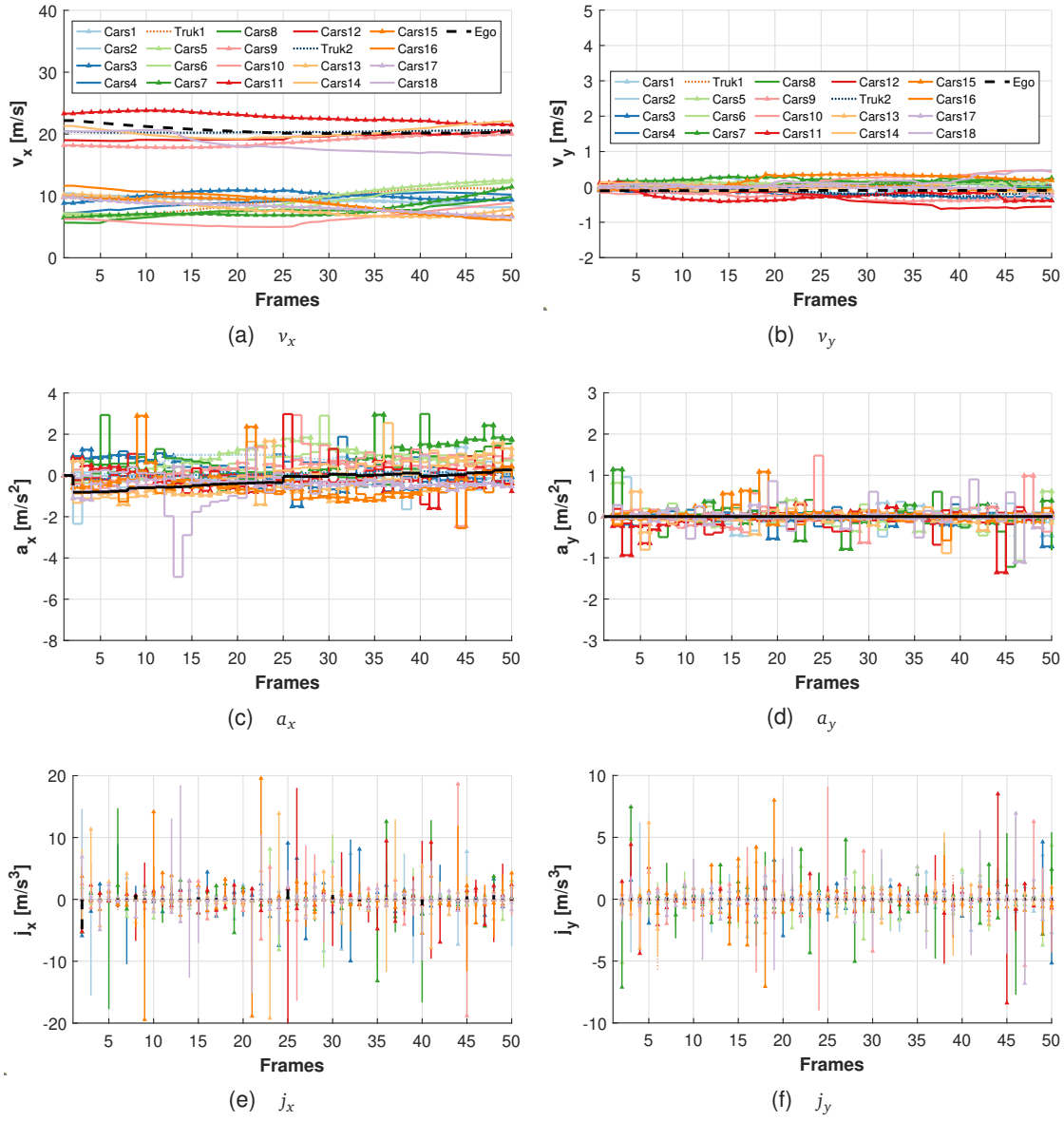
(a) v_x (b) v_y

(c) a_x (d) a_y

(e) j_x (f) j_y

Figure A.8: Analysis of physical parameters in Scenario 3 after using GA

Using PSO:



(a) v_x (b) v_y

(c) a_x (d) a_y

(e) j_x (f) j_y

Figure A.9: Analysis of physical parameters in Scenario 3 after using PSO