AUTHOR, OLDDEGREE

Evaluation of a Genetic Algorithm on generating critical Scenarios in a Traffic Simulation

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Telematik

submitted to

Graz University of Technology

Supervisor

Dr. Some Body

Institute for Softwaretechnology
Head: Univ.-Prof. Dipl-Ing. Dr.techn. Some One

Graz, November 2013

This document is set in Palatino, compiled with pdfLaTeX2e and Biber.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Affidavit

Date

Signature

Abstract

This is a placeholder for the abstract. It summarizes the whole thesis to give a very short overview. Usually, this the abstract is written when the whole thesis text is finished.

Contents

List of Figures

1 Introduction

This Thesis will use a Genetic Algorithm in order to generate critical Driving Scenarios for testing ADAS/AD Functionality in vehicles. While generating these scenarios is the objective, the main task of the thesis will evolve around the implementation of the Genetic Algorithm as well as the Optimization of its Hyperparameter.

1.1 Research Questions

1.1.1 Research Question 1

Is a Genetic Algorithm suitable for generating critical driving scenarios compared to a random generation?

In this thesis,

1.1.2 Research Question 2

Can hypertuning improve the performance of a Genetic Algorithm?

1.1.3 Research Question 3

Can a hypertuned Genetic Algorithm generalize on different start scenarios?

1.1.4 Research Question 4

Is the usage of a Behavior Tree on the Ego vehicle improving the criticaality of resulting scenarios?

1.1.5 Research Question 5

Can rules help to improve the performance of a genetic algorithm?

1.2 Shortcomings

This Master Thesis started with the developement of the Traffic Manger and thus progress was closely linked. Without a working simulations, no genetic alogirthmis could be tested. Due to time and performance constraints, it is not possible to test a full driving stack like autoware, as well as other professional ADAS/AD functions. In this Thesis, internal functions like Time-To-Collision and Emergency Braking will be optimized. The learned information on e.g. optimal hyperparameter settings can then be applied in further steps to test these functions. This will however not be tackled by this thesis.

Performance is also a problem and will lead to many shortcuts that need to be taken. There is a hughe number of possible compations of hyperparamter, so only a handful can be tested. In further chapers, these shortcuts will be explained and their relevancy will be dicussed.

2 Foundations

2.1 Genetic Algorithm

Genetic Algorithms are a popular search algorithm that utilizes the principle of Darwin. They have been used successfully in various areas. Some of their strengths are .... However we will also look at shortcomings, which mainly evolve around performance. We will have a look at its History and then discussing the most important parameters.

Define a vocabulary

The task of the Genetic Algorithm is to search for sequences of actions that will result in the most interesting Scenarios according to its cost function.

Usage of GA

2.1.1 History

The GA was invented by....

2.1.2 Different Hyperparameter

Hyperparamter have a huge influence on the performance of a Genetic Algorihm. They have an impact on the "convergin" ... It has been shown, that there is no universal hyperparamter set and that it needs to be optimized on a per "problem" basis.

Num of generations

The Number of Generation defines the duration of a GA. As long as the algorihtm has not converged, ...? For my testing, using a generation size of 40 was almost always sufficient, and will thus mostly be used.

Pop Size

Pop size will set the number of Individuals of a GA per Generation. The higher the pop size, the bigger the less change of premature converging. It will however also lead to a longer convergin time.

Selection

Selection defines how which individuals are allowed to mate and move into the next generation.

pros and cons of roulette vs Tournament

tournament was chosen to be used for this works because of this paper (and also because of pros and cons list)

Other ideas are evolve around having a flexible selection system debending on fitness

cite paper

Crossover

Crossover is the mating process.

Discuss all used crossover methods

Mutation

Mutation is responsible for introducing new information into the gene pool.

Discuss individual mutation

Discuss all used mutation methods

Other

More to come....

2.2 Behavior Tree

A behavior tree is a decision tree.

insert a good introduction to BT

2.2.1 Usage for GA

Due to the fact, , that there is no full stack available for the EGO vehicle, a solution had to be found. In order to have the Genetic Algorithm controll only NPCs and not the EGO vehicle itselve, a behaviour tree is used. The behaviour tree is used to controll the EGO vehicle over the action interface provided by the Traffic Manager. This is the same as the Genetic Algorithm is doing.

insert ref to discussion

The behaviour tree will define which direction the EGO should take at junctions and it will realistically dodge obstacles intoduced by the Genetic Algorithm. The main goal of the BT is to make the EGO vehicle behave in a realistic way.

In a further chapter it will be dicussed if a GA with controll of the EGO (i.e. no BT will be used) lead to better cost.

While the aim of the GA is to find the most optimal solution, considering the vastness of the hyperspace, this is unlikely. Rather, we want to find the "best" local minimas. Considering the contex of Automotive testing, it is not so much of importance to find "the best fail of the ADAS/AD System", rather its important to find "all" fails.

2.3 Traffic Manager

The Genetic Algorithm will control the simulation of a custom developed Traffic Manager. This Traffic Manager was developed closely to fit the needs of the Genetic Algorithm. It, however is not part of this Thesis and will thus will only get a brief introduction. In general, it will simulate traffic starting from a predefined scenarios where the positions and types of Vehicles and Pedestrians are given (i.e. actors). It also allows for an Interface for aplying actions on all actors in the simulation, which will be discussed in section 2.3.1.

A simulation consists of multiple NPCs and exactly one EGO vehicle. While the NPCs are only controlled by the Traffic Manager (and dadurch also by its action interface), the ego vehicle can be either partly or even completly controlled by an ADAS/AD Function. This function can then be tested inside the simulation on errors.

2.3.1 Action Interface

To interface with the Traffic Manager, actions have to be used. An action will request a certain behaviour from an actor. If no action is set, the actor will behave in a normal manner inside the simulation. An action can be set to at any timestep (for this thesis, the simulation is running with 100 Hz) for any actor. Pedestrians and vehicles however have different actions.

The following list are now all actions provided by the traffic manager that were available for the genetic algorithm at the time of this master thesis.

- JunctionSelection
  - Parameters: Vehicle ID: int, Junction_selection_angle: float
  - Angle is set in radiant. Default value is 0. Vehicles will chose which direction to take at a junction based on this angle.
- LaneChange
  - Parameters: Vehicle ID: int, ...
  - Initiates a LaneChange based on its given parameters.

- AbortLaneChange
  - Parameters: Vehicle ID: int, ...
  - If a LaneChange is currently happening, it will get aborted.
- ModifyTargetVelocity
  - Parameters: Vehicle ID: int, ...
  - Modifies the interal Target Velocity of the Traffic Manager by a percentage. If it is for example 0, the vehicle will stop.
- TurnHeading
  - Parameters: Pedestrian ID: int, ...
  - The pedestrian will turn 180 degrees and walk in the oposite direction
- CrossRoad
  - Parameters: Pedestrian ID: int, ...
  - The pedestrian will cross the road immediately.
- CrossAtCrosswalk
  - Parameters: Pedestrian ID: int, ...
  - The pedestrian will cross the road at the next crosswalk.

2.3.2 Graphics

During the simulation, usually no graphics engine is used in order to save performance. In order to visualize the results, two options can be chosen. The more lightweight Esmini, as well as Carla, which is using Unreal Engine to render realistic graphics.

3 Implementation

This chapter will explain

All these actions are accessed by the Genetic Algorithm to maximize a given Cost Function.

3.1 Map and Starting Scenario

The map is Town10 from Carla. It was chosen, because 1. its roads are self contained, 2. its not too big, yet still complex and 3. its supported by Carla and thus visualization looks better.

The Starting Scenario defines the number and type of all actors as well as their position. It needs to be created manually. Changing the scenario will have a great impact on the Genetic Algorithms performance. For time and complexity reasons, it was thus decided to first stick with one scenario and do all hyperparameter testing there. And finally test the performance for a handfull different scenarios.

3.2 Genetic Algorithm

explain why pygad was NOT chosen

For implementing the Genetic Algorithm, DEAP was chosen. It is a popular tool for accademia .

cite

cite 3 examples

dejong talks about dynamic param and why its not good

3.2.1 Encoding

When implementing a Genetic Algorithm, it is necessary to implement a Encoding that fits to the problem.

cite what makes an encoding good: eg. simplicity,...

Gene

Genes are the building blocks of a GA.

Chromosome

Each Individual has 1 chomosome which consits of a list of genes. Starting out, 2 different encodings came to mind, in both cases, the genes position in the chromosome defined the time an action is set.

generate images

Encoding 1 has the idea that each gene stands for 1 time step. Because multiple actors exist in the simulation, a gene thus needs to be a list of actions. This list always has the length of the number of all actors. This means that crossover can only move all actions of a timestep at once, modifing between actions of the same timestep can only be done using mutation.

Encoding 2 has not only the time step encoded in the position, also the actor ID is encoded. This makes a chromosome now much longer than in the previous encoding, whith the eqatuion beeing: number of timesteps * number of actors. Now crossover has more possibilities.

In the chapter 5 these two chromosome types will be compared.

3.2.2 Rules

Often, actions are not possible if specific requirements are not met. The obvious example is that it is not possible to perform the action Abort-LaneChange if there is no current LaneChange happening. LaneChange during a LaneChange is not possible as well. Also Pedestrians can not CrossRoad shortly in a Row. The hypothesis is that implementing Rules that dont allow for these behaviours will reduce the searchspace and will thus make GA converge quicker.

be carefull, lanechange after lanechange or crossroad after crossroad might be possible if prev did not happen. Good to explain

3.2.3 Cost Function

Cost function is a bit difficult, as we are only using interal values. No ADAS/AD system is tested and we thus have to work with what we got. Currently 3 different cost functions are tested

Oracles

While not implemented here, Oracles are needed in order to get a list of good scenarios.

3.3 Behavior Tree

The Behavior Tree will controll the ego vehicle

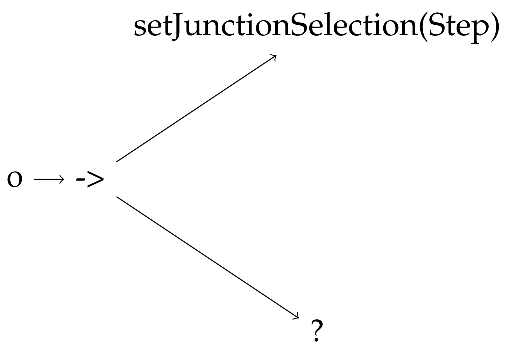

Figure 3.1: Used Behaviour Tree

4 Hyperparameter Tuning

In this chapter, we will incrementally move to an optimized Genetic Algorithm

4.1 No Free Lunch Theorem

No Free Lunch Theorem: The best hyperparameter settings of a Genetic Algorithm are very problem specific. K. De Jong, 2007, Dao, Abhary, and Marian, 2016

More ref

4.2 Start Scenario

4.3 Population

The number of Individuals is of high importance to a genetic algorithm, as has been explained in section 2.1. Especially considering the limed processing resources available, a suitable population size has to be found. On one hand, a population that is too low might result in less diverse runs of the genetic algorithm, on the other hand, if population is too high, the simulations will become too costly. Considering these points, the first step of the hyper parameter tuning was to find a suitable population size. In the next chapter 4.4, we will aim to improve the hyperparamter using a more robust approach.

In order to test for the best population size, the other hyperparameters have to be assumed using an educated guess. While reviewing the literature, trends of general settings for genetic algorithms can be found. However Mills, Filliben, and Haines, 2015 highlight the inconsistencies between findings, stating to have "uncovered conflicting opinions and evidence regarding key GA control parameters".

However Grefenstette, 1986 suggests, that "while it is possible to optimize GA control parameters, very good performance can be obtained with a range of GA control parameter settings." This is also complimented by findings from K. De Jong, 2007: "The key insight from such studies is the robustness of EAs with respect to their parameter settings. Getting “in the ball park” is generally sufficient for good EA performance. Stated another way, the EA parameter “sweet spot” is reasonably large and easy to find [18]. As a consequence most EAs today come with a default set of static parameter values that have been found to be quite robust in practice."

Chosing the right selection method is complicated as well, as discuees by K. De Jong, 2007: "One source of difficulty here is that selection pressure is not as easy to “parameterize” as population size. We have a number of families of selection procedures (e.g, tournament selection, truncation selection, fitness-proportional selection, etc.) to choose from and a considerable body of literature analyzing their differences (see, for example, [19] or [15]), but deciding which family to choose or even which member of a parameterized family is still quite difficult, particularly because of the interacting effects with population size [13]."

Looking at the literature might lead to hyperparameters are used that at least sufficient enough, to get an idea which range for population size is suitable. We will now look at different concrete hyperparameter suggestions from the literature.

4.3.1 Suggested hyperparameter from the literature

Use best values also from : Using genetic algorithms for automating automated lane-keeping system testing

Talk about rules (e.g. 1/n for mut rate...) - look at: Parameter selection in genetic algorithms

In an often cited thesis by K. A. De Jong, 1975, the following parameters have been suggested: GA(50, 0.6, 0.001, 1.0, 7, E) These suggested parameters have been used successfully by various different genetic algorithms Grefenstette, 1986.

An extensive study by Mills, Filliben, and Haines, 2015 which that took over "over 60 numerical optimization problems." into consideration found that "the most effective level settings found for each factor: population size = 200, selection method = SUS, elite selection percentage = 8%, reboot proportion = 0.4, number of crossover points = 3, mutation rate = adaptive and precision scaling = 1/2 as fine as specified by the user."

Grefenstette, 1986 claim that GA(30, 0.95, 0.01, 1.0, 1, E) and GA(80, 0.45, 0.01, 0.9, 1, P) produced the best results. They also advised against, a mutation rate of over 0.05, suggesting poor performance. Using a low mutation rate is also suggested by Whitley, 1994 and Jinghui Zhong et al., 2005. On the other hand, Boyabatli and Sabuncuoglu, 2004 state, that "Controversial to existing literature on GA, our computational results reveal that in the case of a dominant set of decision variable the crossover operator does not have a significant impact on the performance measures, whereas high mutation rates are more suitable for GA applications." Other paper also find a relatively high mutation rate useful. Almanee et al., 2021 uses genetic algorithms in a similar domain as this thesis. There, a Population of 50, crossover of 0.8 and mut of 0.2 was used. These used params are the same as the default params from deap (pop = 50 CXPB, MUTPB, NGEN = 0.5, 0.2, 4).

cite https://deap.readthe

Srinivas and Patnaik, 1994 state, that for a higher population, cross : 0.6, mut: 0.001 and pop: 100 is a good starting point, while a lower population needs higher crossover and mutation rates like this cross: 0.9, mut: 0.01, pop: 30

These next three paper use ANOVA analysis to come a conclusion. Fazal et al., 2005 recommend: Migration direction: Forward Population size: 50 Fitness scaling function: Rank Selection function: Tournament Elite count: 5 Crossover fraction: 0.5 Crossover function: Scattered

Dao, Abhary, and Marian, 2016 suggests these values after anova: Migdirection: forwards pop size: 200 fitness scaling: rank selection: roulette elite count: 1 Crossover prop: 0.7 MutationFunc: Gaussian Crossover FUnc: two point hybrid function: none

Assistant Professor, Amity University, Jaipur, Rajasthan, India et al., 2019 use these values after anova: Direction: Forward Pop: 200 Fitness Scaling

Function: linar Shift selection: Roulette elite count: 10 Crossover: 0.4 Mutation: Constraint Dependent Crossover function: Heuristi Hybrid Function: None

4.3.2 results

This now leads to a difficult decision in choosing the right parameters. Based on the extensive research, we will compare population size of 32, 48, 64 and 96. We will compare the different crossover rates: 0.8 and 0.6. For mutation, 0.01 and 0.2 will be discussed. Further we will use tournament selection with 2 and 4. Each run will be executed 5 times to get rid of randomness and to make the results more robuts. We will run each simulation for 40 Generations.

Comparison of Population Size - mean(standard deviation)					
Settings	Code	32	48	64	96
C: 0.6, M: 0.01, TS: 2	A	3051 (74)	3016 (85)	2851(132)	2871 (57)
C: 0.6, M: 0.01, TS: 4	B	3111 (79)	3021(110)	3079(103)	2937(129)
C: 0.6, M: 0.2, TS: 2	C	3062(128)	3010 (55)	3002 (76)	2831(110)
C: 0.6, M: 0.2, TS: 4	D	3020 (44)	2967 (37)	2891(181)	2850 (90)
C: 0.8, M: 0.01, TS: 2	E	3063(105)	2892(222)	2971(108)	2916(158)
C: 0.8, M: 0.01, TS: 4	F	3052(109)	3049 (96)	3054 (68)	2897(117)
C: 0.8, M: 0.2, TS: 2	G	3099(127)	2940(111)	2959 (96)	2869(131)
C: 0.8, M: 0.2, TS: 4	H	3058 (49)	3005 (84)	2794(173)	2809(105)

Figure 4.1: List Settings per Population Size

In figure 4.3.2, the results per population are plotted. The line is corresponds to the mean, while the bars show the spread (min to max) of all 5 repetitions.

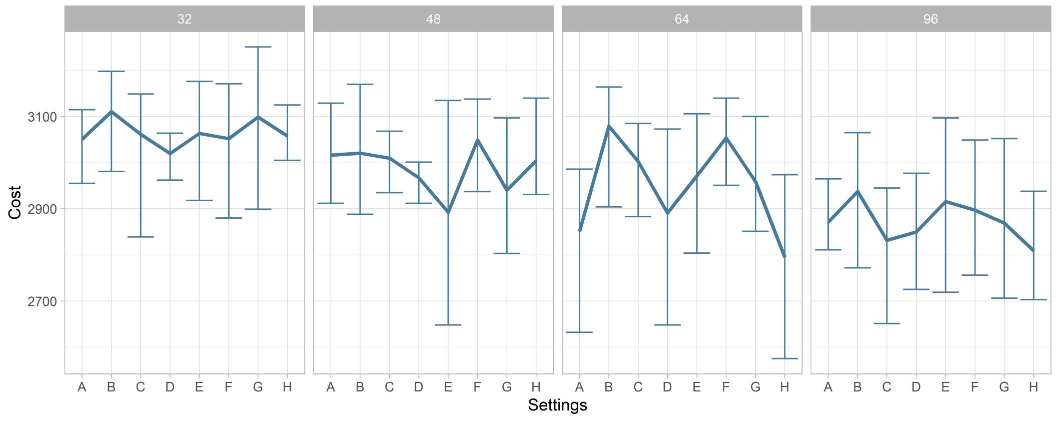

Figure 4.2: mean and error bars per population

A high spread can be seen when looking at small population sizes. Considering these findings, a population size of 96 was chosen. While such a high value will result in a performance impact, it is important to keep the variation low.

4.4 Design of Experiment

In order to tune the hyperparameter of the genetic algorithm, various different strategies can be used. Using automated hyperparamter tuning approaches like "Grid Search", "Bayesian Optimization, "Simmulated Annealing or "Hyperband" might lead to good results with minimal effort (tuning hyperparameter of these search algorithms is still needed), however they require a high number of runs, which is not feasable.

find references

Following the conclusion from the previous section 4.3, a population size of 96 will be used. Executing one run for 30 generations currently takes around 3:50 hours. Although two different workstations were available, the time required to execute the needed number of runs for these automated tests would exceed the available time budged. This is without considering a minimum required number of repetitions to remove randomness in the results.

A different approach called "design of experiment" (DOE), also known as factorial design (Roy, 1990). Each design of experiment has factors, of which each consists of at least two settings, with the actual number of settings being called "levels" (Yang and El-Haik, 2009). Design of experiment needs manual expertise to define which factors are possibly of importance and which settings each factor should have, this is a drawback compared to automatic hyperparameter tuning. Afterwards, main effects and interactions can be calculated to find the best settings per factor. Using ANOVA (Analysis of Variance) it is possible to identify the significance of each main effect and interaction. More details on these analysis tools will be provided in section 4.4.3.

In a factorial designs, each possible combination of factor levels needs to be tested. Looking at the proposed factors in table 4.4.1, we would require 1024 runs .

generated by minitab (or https://datatab.net/statistics-calculator/design-of-experiments)

"Techniques such as fractional (or partial) factorial experiments are used to simplify the experiment. Fractional factorial experiments investigate only a fraction of all possible combinations. This approach saves considerable time and money but requires rigorous mathematical treatment, both in the design of the experiment and in the analysis of the results." Roy, 1990

Is this really the case? What about combinatorial testing?

4.4.1 Taguchi Design

Various improvements to Design of experiment have been but forward by Dr. Genichi Taguchi, such as reducing the influence of uncontrollable (noise) factors on processes and products and reducing variability. Some of these methods evolve around Signal-to-noise (S/N) analysis and utilizing cost functions to "express predicted improvements from DOE results in terms of expected cost saving" (Roy, 1990). This master thesis will not discuss all of this proposed considerations, for more detail Roy, 1990 as well as Yang and El-Haik, 2009 is highly recommended.

This masters thesis will mainly utilizes Taguchis orthogonal arrays (OAs), "which represent the smallest fractional factorials and are used for most common experiment designs." (Roy, 1990). Taguchis proposed various different orthogonal arrays which need to be selected based on the individual needs. Each row of these arrays stands for 1 trial condition (i.e factor level combination of individual experiment), while the columns represent the factors.

Definition orthogonal array

Using these orthogonal arrays instead of full factorial experiments will lead to needing a much smaller amount of simulation runs (in our case only 16 compared to 1024), while the latter "might not provide appreciably more useful information" Roy, 1990.

cons of taguchi arrays: no higher level interactions

When choosing a suitable taguchi ortogonal arrays, we need to take various factors into account, which can make the process tricky. According to Yang and El-Haik, 2009, we will have to follow a three step procedure:

1. Calculate the total degree of freedom (DOF).
2. Following two rules, standard orthogonal array should be selected:
    a) Total DOF need to be smaller than the number of runs provided by the orthogonal array.
    b) All required factor level combinations need to be accommodated by the orthogonal array.
3. Factors have to be assigned using these rules:
    a) In case the factor level does not fit into the orthogonal array, methods such as column merging and dummy level can be used to modify the original array.
    b) Using the linear graph and interaction table, interactions can be defined.
    c) In case some columns are not assigned, its possible to keep these columns empty.

For this genetic algorithm, 7 factors (3 Factors of Level 4 and 4 Factors of Level 2) have been selected. Which factors to choose and with which level was done based on experience gained on section 4.3. In table 4.4.1, every factor with correspondig levels has been listed,

Factors	Code	Level 1	Level 2	Level 3	Level 4
CrossoverType	A	one point	two point	uniform 0.1	uniform 0.5
CrossoverProp	B	0.2	0.5	0.8	0.9
MutationProp	C	0.01	0.1	0.3	0.5
ChromosomeType	D	Time	Time+NPC	-	-
GeneType	E	int	dict	-	-
TournamentSize	F	2	4	-	-
IndMutationProp	G	0.1	0.5	-	-

Figure 4.3: List of Hyperparamters (Factors) matched to a Code and defined settings (Levels)

Using this table, we will now find the best standart orthogonal array in section 4.4.2. Before doing so, it is important to state, that taguchi allows to test for possible (pre determined) two-level interactions (Yang and El-Haik, 2009). Analysing interactions comes at a cost of Degrees of freedom. If we look at the table, an interaction between ChromosomeType and GeneType might be of interest. Using the power of hindsight, we know, that a second two factor interaction is possible within our chosen array, thus we will have a look at the interaction between Tournament Size and IndMutationPropability.

4.4.2 Selection of a suitable standart orthogonal array

The total degree of freedom can be quickly calculated using the rules provided by Yang and El-Haik, 2009:

1. 1 DOF is always used for the overall mean.
2. Each factor has a DOF of NumberOfLevels - 1.
3. Two-factor interactions use this equation to calculate DOF: $(n_{factor1} - 1)(n_{factor2} - 1)$ where n = number of levels.

This leads to the following calculation for the needed 3 Factors of Level 4 and 4 Factors of Level 2 as well as the two interactions between ChromosomeType-GeneType and TournamentSize-IndMutationProp:

$$\begin{aligned} DOF &= 1 + 3 * (3-1) + 4 * (2-1) + 2 * (2-1) * (2-1) \\ &= 13 \end{aligned} \tag{4.1}$$

A L_{16} array seems suitable to accommodate the required 13 DOF, which can be seen in 4.4.2.

	$L_{16}(2^{15})$														
NO.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
3	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2
4	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1
5	1	2	1	1	1	2	2	1	1	2	2	1	1	2	2
6	1	2	2	1	1	2	2	2	2	1	1	2	2	1	1
7	1	2	2	2	2	1	1	1	1	2	2	2	2	1	1
8	1	2	2	2	2	1	1	2	2	1	1	1	1	2	2
9	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
10	2	1	2	1	2	1	2	2	1	2	1	2	1	2	1
11	2	1	2	2	1	2	1	1	2	1	2	2	1	2	1
12	2	1	2	2	1	2	1	2	1	2	1	1	2	1	2
13	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1
14	2	2	1	1	2	2	1	2	1	1	2	2	1	1	2
15	2	2	1	2	1	1	2	1	2	2	1	2	1	1	2
16	2	2	1	2	1	1	2	2	1	1	2	1	2	2	1

Figure 4.4: $L_{16}(2^{15})$ Taguchi ortohogonal array taken from Roy, 1990

This graph now needs to be fitted to the needed factors. 4 Level Factors need more space which will be generated using column merging, while interactions need to be assigned as well. For this, either an interaction table or linear graphs of this L_{16} array can be used (Roy, 1990, **nazandanacioglu_taguchi_2005**). The linear graph approach is straight forward and will be selected. While there are multiple linear graphs for L_{16} array, the following graph has the best fit for the requirements from table 4.4.1.

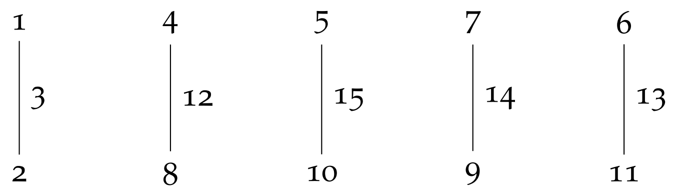

Figure 4.5: Linear Graph of $L_{16}(2^{15})$ taken from Yang and El-Haik, 2009

In a taguchi linear graph, the nodes as well as the ?Lines? both represent collumns in the orthogonal array. An interaction between to columns that are represented as nodes "comes out to" to the connecting line column **taguchi_taguchis_2005**. This is usefull for both analyzing interactions between columns as well as combining (mergin) interacting columns in case a higher factor is needed.

Go over notes from paper and explain this

Column Merging A, B and C are both 4 level factors. The currently selected orthogonal only fits 2 level factors.

OLD COLUMN			NEW COLUMN
1	1	->	1
1	2	->	2
2	1	->	3
2	2	->	4

Figure 4.6: Rules taken from Roy, 1990

Go over notes from paper and explain this

Assigning Interactions An interaction between DE is might be possible. As we still have some unused space in the graph, we will also look at the interaction of FG.

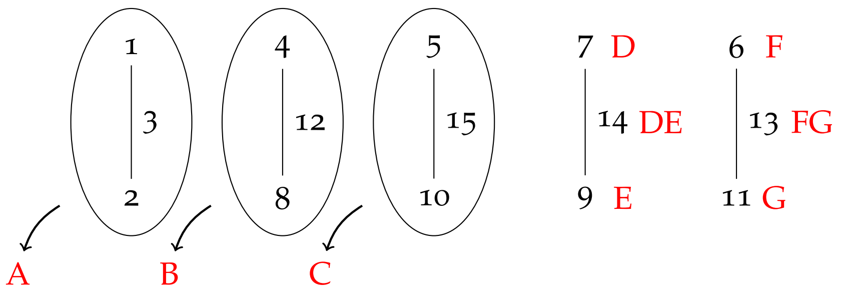

Figure 4.7: Modified Linear Graph to fit our needs

Combining columns 1 2 3 to A, 4 8 12 to B and 5 10 15 to C using rules defined by table 4.6 is done in 4.8.

NO.	1 2 ~~3~~	4 8 ~~12~~	5 10 ~~15~~
1	~~1 1~~ > 1	~~1 1~~ > 1	~~1 1~~ > 1
2	~~1 1~~ > 1	~~1 2~~ > 2	~~1 2~~ > 2
3	~~1 1~~ > 1	~~2 1~~ > 3	~~2 1~~ > 3
4	~~1 1~~ > 1	~~2 2~~ > 4	~~2 2~~ > 4
5	~~1 2~~ > 2	~~1 1~~ > 1	~~1 2~~ > 2
6	~~1 2~~ > 2	~~1 2~~ > 2	~~1 1~~ > 1
7	~~1 2~~ > 2	~~2 1~~ > 3	~~2 2~~ > 4
8	~~1 2~~ > 2	~~2 2~~ > 4	~~2 1~~ > 3
9	~~2 1~~ > 3	~~1 1~~ > 1	~~2 1~~ > 3
10	~~2 1~~ > 3	~~1 2~~ > 2	~~2 2~~ > 4
11	~~2 1~~ > 3	~~2 1~~ > 3	~~1 1~~ > 1
12	~~2 2~~ > 3	~~2 2~~ > 4	~~1 2~~ > 2
13	~~2 2~~ > 4	~~1 1~~ > 1	~~2 2~~ > 4
14	~~2 2~~ > 4	~~1 2~~ > 2	~~2 1~~ > 3
15	~~2 2~~ > 4	~~2 1~~ > 3	~~1 2~~ > 2
16	~~2 2~~ > 4	~~2 2~~ > 4	~~1 1~~ > 1

Figure 4.8: Building 4 Level columns from 2 Level columns

Removing the old and inserting the new columns in the table and transcoding 7 to D, 9 to E, 14 to DE, 6 to F, 11 to G and 13 to FG results in the final table 4.9. This A-G combinations table will subsequently be used as settings for simulations.

NO.	A	B	C	D	E	F	G	FG	DE
1	1	1	1	1	1	1	1	1	1
2	1	2	2	1	2	1	2	2	2
3	1	3	3	2	1	2	1	2	2
4	1	4	4	2	2	2	2	1	1
5	2	1	2	2	1	2	2	1	2
6	2	2	1	2	2	2	1	2	1
7	2	3	4	1	1	1	2	2	1
8	2	4	3	1	2	1	1	1	2
9	3	1	3	2	2	1	2	2	1
10	3	2	4	2	1	1	1	1	2
11	3	3	1	1	2	2	2	1	2
12	3	4	2	1	1	2	1	2	1
13	4	1	4	1	2	2	1	2	2
14	4	2	3	1	1	2	2	1	1
15	4	3	2	2	2	1	1	1	1
16	4	4	1	2	1	1	2	2	2

Figure 4.9: Final version of used Taguchi orthogonal array

4.4.3 Analysing the results

This now can be used for running all the needed testcases (the interaction columns can be ignored until the evaluation). Simply exchange all levels in the table with the corresponding setting from table 4.4.1. We will repeat every setting 8 times. These are the results:

NO.	rep1	rep2	rep3	rep4	rep5	rep6	rep7	rep8
1	1000	1000	1000	1000	1000	1000	1000	1000
2	1000	1000	1000	1000	1000	1000	1000	1000
3	1000	1000	1000	1000	1000	1000	1000	1000
4	1000	1000	1000	1000	1000	1000	1000	1000
5	1000	1000	1000	1000	1000	1000	1000	1000
6	1000	1000	1000	1000	1000	1000	1000	1000
7	1000	1000	1000	1000	1000	1000	1000	1000
8	1000	1000	1000	1000	1000	1000	1000	1000
9	1000	1000	1000	1000	1000	1000	1000	1000
10	1000	1000	1000	1000	1000	1000	1000	1000
11	1000	1000	1000	1000	1000	1000	1000	1000
12	1000	1000	1000	1000	1000	1000	1000	1000
13	1000	1000	1000	1000	1000	1000	1000	1000
14	1000	1000	1000	1000	1000	1000	1000	1000
15	1000	1000	1000	1000	1000	1000	1000	1000
16	1000	1000	1000	1000	1000	1000	1000	1000

Figure 4.10: List of results

ANOVA

Due to our number of repetitions, the number of DOF increases according to the following equation (taken from Roy, 1990):

$$\begin{aligned} DOF &= totalNumberOfResults - 1 \\ &= numberOfTrials * numberOfRepetitions - 1 \\ &= 16 * 8 - 1 = 127 \end{aligned} \tag{4.2}$$

Calculating ANOVA can be done using R:

Cite book from Gabriel

```
# pivot the results, so that the table has 8 times the rows
taguchi.combined_pivoted <- taguchi.combined %>% pivot_longer(cols

# run anova analysis
```

```
anova <- aov(results ~ factor(A) + factor(B) + factor(C)
summary(anova)
```

After pooling, we can run anova again:

```
anova <- aov(results ~ factor(A) + factor(B) + factor(C)
summary(anova)
```

This will result in the following table:

Column	DF	Sum Sq	Mean Sq	F value	p value
1	A	3	1000	1000	1000
2	B	3	1000	1000	1000
3	C	3	1000	1000	1000
4	D	1	1000	1000	1000
5	E	1	1000	1000	1000
6	F	1	1000	1000	1000
7	G	1	1000	1000	1000
8	DxE	1	1000	1000	1000
9	FxG	1	1000	1000	1000
Residuals			1000	1000	1000

Figure 4.11: ANOVA results

Main-effects or main-effect

Main-effects and interaction chart

According to Yang and El-Haik, 2009, "The main-effects chart is a plot of average responses at different levels of a factor versus the factor levels" and the calculation" of main-effect charts and iteration charts are the same as those of classical experimental data analysis".

For every factor, sum up the mean of the results per level, then divide by the number of runs per level. This is the calculation for D:

Insert calculation for d

The resulting main-effect charts can be seen here:

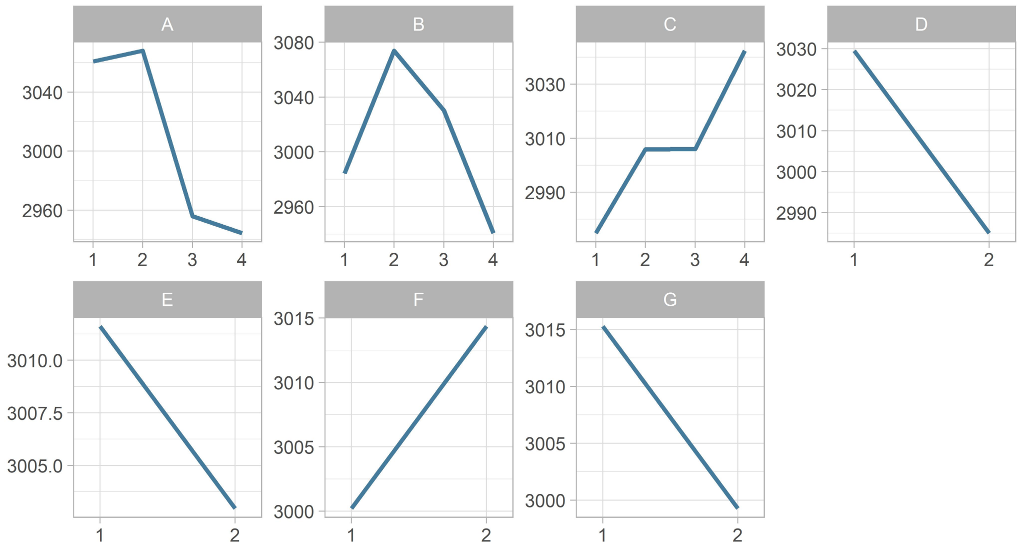

Figure 4.12: Main Effects

Similarly, the interactions are calculated like this:

Insert example calculation of interaction

The resulting interaction charts can be seen here:

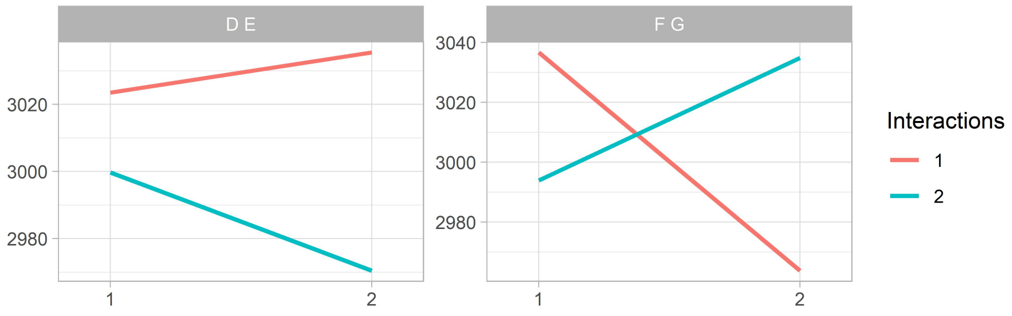

Figure 4.13: Interaction Effects

Best factor level selection and optimal performance level prediction (Take from yang_design_2009, so reformulate)

5 Evaluation

in this chapter, we evaluate and compare various different settings

5.1 Comparison with random and default ga Values

Compare with Boxplot...

Different Algorithms: 1. Optimized GA 2. Best GA setting from Population 3. Random

5.2 Generalization on different start scenarios

6 Conclusion

6.1 Test

Altough lots of shortcommings Results look very vielversprechend. This thesis hopes to have emphasised that this approach has lots of advanatages

Appendix

Appendix A.

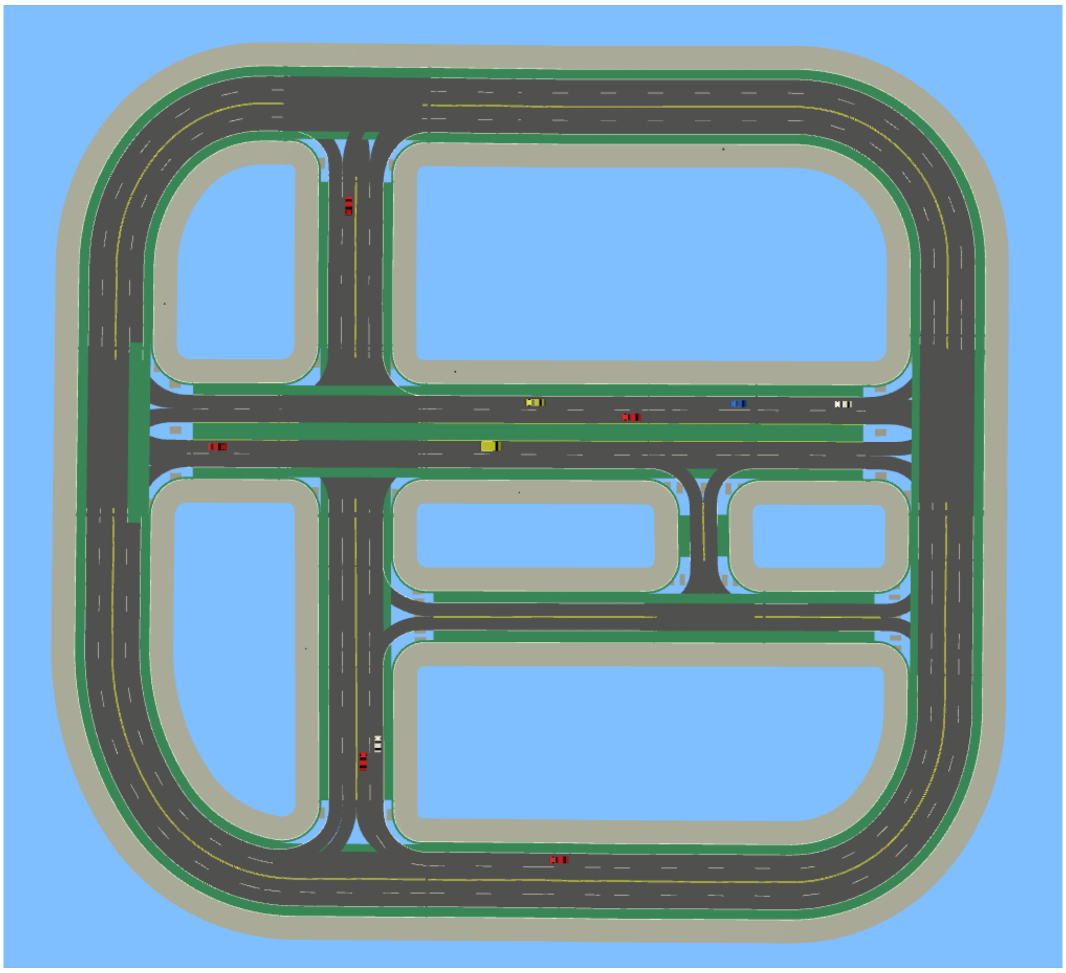

Figure 1: Start scenario 1

Bibliography

Almanee, Sumaya et al. (2021). “scenoRITA: Generating Less-Redundant, Safety-Critical and Motion Sickness-Inducing Scenarios for Autonomous Vehicles.” In: Publisher: arXiv Version Number: 1. doi: 10.48550/ARXIV.2112.09725. url: https://arxiv.org/abs/2112.09725 (visited on 10/19/2023) (cit. on p. 15).

Assistant Professor, Amity University, Jaipur, Rajasthan, India et al. (July 30, 2019). “Parameter Tuning Method for Genetic Algorithm using Taguchi Orthogonal Array for Non-linear Multimodal Optimization Problem.” In: *International Journal of Recent Technology and Engineering (IJRTE)* 8.2, pp. 2979–2986. issn: 22773878. doi: 10.35940/ijrte.B2711.078219. url: https://www.ijrte.org/portfolio-item/B2711078219/ (visited on 10/09/2023) (cit. on p. 15).

Boyabatli, Onur and Ihsan Sabuncuoglu (2004). “Parameter selection in genetic algorithms.” In: *Journal of Systemics, Cybernetics and Informatics* 4.2. Publisher: International Institute of Informatics and Cybernetics, p. 78 (cit. on p. 15).

Dao, Son, Kazem Abhary, and Romeo Marian (Feb. 2016). “Maximising Performance of Genetic Algorithm Solver in Matlab.” In: *Engineering Letters* 24 (cit. on pp. 13, 15).

De Jong, Kenneth (2007). “Parameter Setting in EAs: a 30 Year Perspective.” In: *Parameter Setting in Evolutionary Algorithms*. Ed. by Fernando G. Lobo, Cláudio F. Lima, and Zbigniew Michalewicz. Red. by Janusz Kacprzyk. Vol. 54. Series Title: Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–18. isbn: 978-3-540-69431-1 978-3-540-69432-8. doi: 10.1007/978-3-540-69432-8_1. url: http://link.springer.com/10.1007/978-3-540-69432-8_1 (visited on 10/13/2023) (cit. on pp. 13, 14).

De Jong, Kenneth Alan (1975). *An analysis of the behavior of a class of genetic adaptive systems.* University of Michigan (cit. on p. 14).

Fazal, M.A. et al. (Mar. 2005). "Estimating groundwater recharge using the SMAR conceptual model calibrated by genetic algorithm." In: *Journal of Hydrology* 303.1, pp. 56–78. ISSN: 00221694. DOI: 10.1016/j.jhydrol.2004.08.017. URL: https://linkinghub.elsevier.com/retrieve/pii/S0022169404003890 (visited on 10/23/2023) (cit. on p. 15).

Grefenstette, John (Jan. 1986). "Optimization of Control Parameters for Genetic Algorithms." In: *IEEE Transactions on Systems, Man, and Cybernetics* 16.1, pp. 122–128. ISSN: 0018-9472. DOI: 10.1109/TSMC.1986.289288. URL: http://ieeexplore.ieee.org/document/4075583/ (visited on 10/13/2023) (cit. on pp. 14, 15).

Jinghui Zhong et al. (2005). "Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms." In: *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*. International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06). Vol. 2. Vienna, Austria: IEEE, pp. 1115–1121. ISBN: 978-0-7695-2504-4. DOI: 10.1109/CIMCA.2005.1631619. URL: http://ieeexplore.ieee.org/document/1631619/ (visited on 10/23/2023) (cit. on p. 15).

Mills, K. L., J. J. Filliben, and A. L. Haines (June 2015). "Determining Relative Importance and Effective Settings for Genetic Algorithm Control Parameters." In: *Evolutionary Computation* 23.2, pp. 309–342. ISSN: 1063-6560, 1530-9304. DOI: 10.1162/EVCO_a_00137. URL: https://direct.mit.edu/evco/article/23/2/309-342/986 (visited on 10/13/2023) (cit. on pp. 14, 15).

Roy, Ranjit K. (1990). *A primer on the Taguchi method*. Competitive manufacturing series. New York: Van Nostrand Reinhold. 247 pp. ISBN: 978-0-442-23729-5 (cit. on pp. 18, 19, 21, 22, 25).

Srinivas, M. and L.M. Patnaik (June 1994). "Genetic algorithms: a survey." In: *Computer* 27.6, pp. 17–26. ISSN: 0018-9162. DOI: 10.1109/2.294849. URL: http://ieeexplore.ieee.org/document/294849/ (visited on 10/13/2023) (cit. on p. 15).

Whitley, Darrell (June 1, 1994). "A genetic algorithm tutorial." In: *Statistics and Computing* 4.2, pp. 65–85. ISSN: 1573-1375. DOI: 10.1007/BF00175354.

URL: https://doi.org/10.1007/BF00175354 (visited on 03/28/2023) (cit. on p. 15).

Yang, Kai and Basem El-Haik (2009). *Design for six sigma: a roadmap for product development*. 2nd ed. London: McGraw-Hill [distributor]. 741 pp. ISBN: 978-0-07-154767-3 (cit. on pp. 18–20, 22, 26).