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

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

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?

1.1.2 Research Question 2

Is it possible to improve the performance of a Genetic Algorithm using Taguchi Orthogonal Array Testing?

Taguchi Orthogonal Arrays allow for minimal experiment runs when testing across different settings Roy, 1990). It will be analyzed if this method can be used to

1.1.3 Research Question 3

Can a hypertuned Genetic Algorithm generalize on different start scenarios?

Due to performance considerations, only one start scenario was defined. When evaluating the optimzied Genetic Algorithm, its performance will be compared across different start scenarios.

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.

Genes are the building blocks of a GA

Usage of GA

" From a general perspective, GA application requires, besides transforming the test problem into a search problem (representation), the definition of a method to distinguish better from worse solutions (fitness function) and to choose appropriate parameter settings. These parameters are often referred to as strategy or control parameters, where one has to specify, for instance, the population size, selection mechanism (roulette wheel, tournament, rank-based, etc.), crossover-type and probability, and mutation-type and probability. The choice of all these parameters might significantly impact the performance of a search algorithm. In the worst case, an “unfortunate” parameter setting might make it impossible to solve the problem at hand.23 While there is empirical data available to support the choice of appropriate default parameters for a specific problem to be solved,24 it has been formally proven in the “No Free Lunch” (NFL) theorem that there is not one optimal parameter setting for all possible search problems.25 "Klampfl, Klück, and Wotawa, 2023

dejong talks about dynamic param and why its not good

" For instance, in a GA, one may want to decrease the probability of crossover and mutation over time to avoid too high reproductive variation hindering convergence to local optima. However, in that case, one would need to set new parameters that define how much and how often each probability is decreased. Research has not yet concluded if one technique has a clear advantage over the other. "Klampfl, Klück, and Wotawa, 2023

" As an intelligent search optimization technique, genetic algorithm (GA) is an important approach for non-deterministic polynomial (NP-hard) and complex nature optimization problems.

The most attractive feature of GA is that it has the ability to explore the search space with the help of the entire population of individuals [2]. "Hussain and Muhammad, 2020

"

A very common issue about GA is premature convergence to find the optimal solution of a problem. This is strongly linked to the loss of population diversity. If it is very low then a very quick convergence will be observed by GA; otherwise, time-consuming and may cause wastage of computational resources. Hence, there is essential to find a trade-off between exploration (i.e., exploring the new areas of search space) and exploitation (i.e., using already detected points to search the optimum). Therefore, the performance of the GA highly depends on its genetic operators, in general.

The balance between exploration and exploitation can be adjusted either by selection pressure in a selection approach or by the recombination operators with adjustment of their probabilities. "Hussain and Muhammad, 2020

" Genetic algorithms are global optimization techniques that avoid many of the shortcomings exhibited by local search techniques on difficult search spaces.

A GA is an iterative procedure which maintains a con- stant-size population P(t) of candidate solutions. During each iteration step, called a generation, the structures in the current population are evaluated, and, on the basis of those evaluations, a new population of candidate solutions is formed (see Fig. 3.)

The power of GA's derives largely from their ability to exploit efficiently this vast amount of accumulating knowledge by means of relatively simple selection mechanisms [17]. "Grefenstette, 1986

" Termination ofthe GA may be triggered by finding an acceptable approximate solution, vel adaptive system model. by fixing the total number of structure evaluations, or some other application dependent criterion. "Grefenstette, 1986

" GA's consistently outperform both gradient techniques and various forms of random search on more difficult (and more common) problems, such as optimizations involving discontinuous, noisy, high-dimensional, and multimodal objective functions. "Grefenstette, 1986

" GAs can find good solutions within a large, ill-defined search space, and can be readily adapted to a wide variety of search problems (Mitchell, 1998). "Mills, Filliben, and Haines, 2015

" Genetic algorithm search methods are rooted in the mechanisms of evolution and natural genetics. The interest in heuristic search algorithms with underpinnings in natural and physical processes began as early as the 1970s,when Holland' first proposed genetic algorithms "Srinivas and Patnaik, 1994

" In nature. individuals best suited to competition for scanty resources survive. Adapting to a changing environment is essential for the survival of individualsof each species. While the various features that uniquely characterize an individual determine its survival capacity. the features in turn are determined by the individual's genetic content. Specifically, each feature is controlled by a basic unit called a gene. The sets of genes controlling features form the chromosomes, the "keys" to the survival of the individual in a competitive environment. "Srinivas and Patnaik, 1994

" Genetic algorithms manipulate a population of potential solutions to an optimization (or search) problem. Specifically, they operate on encoded representations of the solutions, equivalent to the genetic material of individuals in nature, and not directly on the solutions themselves "Srinivas and Patnaik, 1994

" Each solution is associated with a fitness vulue that reflects how good it is, compared with other solutions in the population. The higher the fitness value of an individual, the higher its chances of survival and reproduction and the larger its representation in the subsequent generation. Recombination of genetic material in genetic algorithms is simulated through a crossover mechanism that exchanges portions between strings. Another operation, called mutation, causes sporadic and random alteration of the bits of strings. Mutation too has a direct analogy from nature and plays the role of regenerating lost genetic material. "Srinivas and Patnaik, 1994

" Simulated annealing, genetic algorithms, and evolutionary strategies are similar in their use of a probabilistic search mechanism directed toward decreasing cost or increasing payoff. These three methods have a high probability of locating the global solution optimally in a multimodal search landscape. (A multimodal cost function has several locally optimal solutions as well.) "Srinivas and Patnaik, 1994

" The principal difference between genetic algorithms and evolutionary strategies is that genetic algorithms rely on crossover, a mechanism of probabilistic and useful exchange of information among solutions, to locate better solutions, while evolutionary strategies use mutation as the primary search mechanism. "Srinivas and Patnaik, 1994

" Genetic algorithm (GA) is one of the optimization methods that has received increasing attention in recent decades. Its popularity is attributed to the independence of functional derivatives, and the capability to solve both discrete and continuous optimization problems and to avoid getting trapped in the local optima that inevitably appear in many practical optimization problems. "Xia et al., 2019

" Prof. Holland (1974) from the University of Michigan developed the ideas and concepts GA based on the principles of Genetics and Natural Selection “survival of the fittest” and many authors have refined his initial approach. Genetic algorithms imitate the evolutionary process of species and natural selection by a computer program. A very important point to note that GA searches the solution space by maintaining a population of potential solutions and is less likely to get trapped at a local optimum. Each individual in the population is referred as a chromosome. The genetic information of the chromosome is encoded using an appropriate method

(Eg. Binary Encoding, Hexadecimal Encoding, Tree, etc.), representing a solution to the given problem. These chromosomes then evolve through successiv iterations, called generations and subsequently follows certain steps. The details are explained through the flow chart shown in Figure 1 "Majumdar and Ghosh, 2015

" GAs are a subclass of evolutionary algorithms (EAs). EAs comprise a collection of heuristic methods that use techniques, such as mutation, recombination, and selection (inspired by genetics and biological evolution) to search for optimal solutions to difficult problems. "Mills, Filliben, and Haines, 2015

" Single-solution based metaheuristic algorithms utilize single candidate solution and improve this solution by using local search. However, the solution obtained from single-solution based metaheuristics may stuckinlocaloptima[112]. The well-known single-solution based metaheuristics are simulated annealing, tabu search (TS), microcanonical annealing (MA), and guided local search (GLS). Population-based metaheuristics utilizes multiple candidate solutions during the search process. These metaheuristics maintain the diversity in population and avoid the solutions are being stuck in local optima. Some of well-known population-based metaheuristic algorithms are genetic algorithm (GA) [135], particle swarm optimization (PSO) [101], ant colony optimization (ACO) [47], spotted hyena optimizer (SHO) [41], emperor penguin optimizer (EPO) [42], and seagull optimization (SOA) [43]. "Katoch, Chauhan, and Kumar, 2021 " GA mimics the Darwinian theory of survival of fittest in nature. GA was proposed by J.H. Holland in 1992. "Katoch, Chauhan, and Kumar, 2021

" The basic elements of GA are chromosome representation, fitness selection, and biological-inspired operators. Holland also introduced a novel element namely, Inversion that is generally used in implementations of GA [77]. Typically, the chromosomes take the binary string format. In chromosomes, each locus (specific position on chromosome) has two possible alleles (variant forms of genes) - 0 and 1. Chromosomes are considered as points in the solution space. These are processed using genetic operators by iteratively replacing its population. The fitness function is used to assign a value for all the chromosomes in the population [136]. The biological-inspired operators are selection, mutation, and crossover. In selection, the chromosomes are

selected on the basis of its fitness value for further processing. In crossover operator, a random locus is chosen and it changes the subsequences between chromosomes to create off-springs. In mutation, some bits of the chromosomes will be randomly flipped on the basis of probability [77,135,136]. T "Katoch, Chauhan, and Kumar, 2021

" Genetic algorithm (GA) is an optimization algorithm that is inspired from the natural selection. It is a population based search algorithm, which utilizes the concept of survival of fittest [135]. "Katoch, Chauhan, and Kumar, 2021

" The procedure of GA is as follows. A population (Y)ofn chromosomes are initialized randomly. The fitness of each chromosome in Y is computed. Two chromosomes say C1 and C2 are selected from the population Y according to the fitness value. The single-point crossover operator with crossover probability (Cp) is applied on C1 and C2 to produce an offspring say O. Thereafter, uniform mutation operator is applied on produced offspring (O) with mutation probability (Mp) to generate O'. The new offspring O' is placed in new population. The selection, crossover, and mutation operations will be repeated on current population until the new population is complete. "Katoch, Chauhan, and Kumar, 2021

" Real-coded GAs (RGAs) have been widely used in various real-life applications. The representation of chromosomes is closely associated with real-life problems. The main advantages of RGAs are robust, efficient, and accurate. However, RGAs suffer from premature convergence. Researchers are working on RGAs to improve their performance. Most of RGAs are developed by modifying the crossover, mutation and selection operators. "Katoch, Chauhan, and Kumar, 2021

" Genetic Algorithms !GAs) represent a highly efcient search procedure for the determination of global extrema of multivariable functions, imitating the patterns of genetic reproduction in living organisms. GAs work by successive modi®cations of a collection !often referred to as population) of parameter combinations in the search space, using a binary coded representation termed chromosome. The initial set of chromosomes evolves through a number of operators !selection, crossover, mutation) so that subsequent generations produce more and more chromosomes in the neighborhood of the optimum, de®ned through a given ugure of merit, or tness. Genetic

Algorithms were ®rst introduced by Goldberg !1989) and Holland !1992) and differ from conventional search algorithms in the following aspects: "Marsili Libelli and Alba, 2000

" - GAs work with a coding of the parameters, not the parameters themselves; - GAs search using a population of points, not a single one, thus thoroughly cover the search region and avoid being trapped into local extrema; - GAs use probabilistic, not deterministic, transition rules to alter the initial population through subsequent generations; - GAs have a high implicit parallelism, making them numerically very efficient; - Like all derivative-free search methods, GAs use only point-wise evaluation, thus are very insensitive to discontinuities and irregularities in the fitness function. "Marsili Libelli and Alba, 2000

" The optimization procedure starts from a population of randomly chosen chromosomes and generates successive populations applying the operators of reproduction, mutation and crossover !Goldberg, 1989; Davis, 1990; Holland, 1992). "Marsili Libelli and Alba, 2000

" John Holland pioneered genetic algorithms by simulating evolution of the nature. During the past forty years, genetic algorithms have been applied in many fields such as pattern recognition, robotics, artificial life, experts system, electronic and electrical field, cellular automata, etc [1]-[7]. By using the genetic algorithms to solve a problem, we first present the candidate solutions as a sequence of values, and define an evaluation function to evaluate the candidate solutions. An artificial genetic system uses concepts like population and generation to simulate the natural genetic system. One population consists of a certain number of individuals which serve as candidate solutions. New generation of population is created by genetic operations such as selection, crossover and mutation in iteration. According to the Darwinian principles of survival, which is called “the survival of the fittest”, the excellent individuals have far more chances to adapt themselves to the environment and survive, while the inferior ones die out [8]. The survivals reproduce new individuals with better genes which make the new generation more endurable to the nature. Similarly, the GAs(Genetic Algorithms) use this process of reproduction as a basic genetic operation for the algorithms. Currently there are several selection strategies, such as

roulette wheel, tournament selection, rank-based selection and deterministic sampling. "Jinghui Zhong et al., 2005

" The SGA uses the steps as below: Step1. Encode the given problems in gene strings. Step2. Create an initial population consists of a certain number of individuals. Step3. Perform sub-steps as follows until the termination condition is satisfied: (a)Evaluate the fitness value of each individual in the population. (b)Create a new population by three genetic operations as follows: ¾ According to the fitness value, individuals are chosen with a probability. Replicate the selected ones to form a new population. ¾ Create two new individuals by two parents who are selected probabilistically from the population and recombine them at the crossover point. ¾ Create a new individual by mutating an existing individual with the probabilistically selected. Step4. When the process ceases, we can find a solution from the result of the genetic algorithm. "Jinghui Zhong et al., 2005

" Genetic algorithm (GA) is one of the optimization methods that has received increasing attention in recent decades (Mitra et al.;Kalayci et al.). Its popularity is attributed to the independence of functional derivatives, and the capability to solve both discrete and continuous optimization problems and to avoid getting trapped in the local optima that inevitably appear in many practical optimization problems (Javadi et al). "

```
simple_genetic_algorithm()
{
    initialize_population();
    evaluate_population();
    for(int i = 0; i < num_of_generations; i++)
    {
        select_individuals_for_next_population();
        perform_crossover();
        perform_mutation();
        evaluate_population();
    }
}
```

Srinivas and Patnaik, 1994

" Each iteration of the simple GA creates an entirely new population from an existing population. GASthat replace the entire population are called

generational GAS.GAS that replace only a small fraction of strings at a time are called steady-state GAS "Srinivas and Patnaik, 1994

" Genetic algorithmsare particularly attractive because instead of a naive “search and select” mechanism they use crossover to exchange information among existing solutions to locate better solutions. "Srinivas and Patnaik, 1994

" GAs are search algorithms based on the mechanics of natural selection and natural genetics. In natural genetics, the presence/absence of genes and their order in the chromosome decide the characteristic features of individuals of a population. The different traits are passed on from one generation to the next through different biological processes, which operate on the genetic structure. By this process of genetic change and survival of the fittest, a population well adapted to the environment results. Similarly, in GA, a finite-length string coding is used to describe the parameter values of each solution for the search problem under consideration. Each string corresponds to an individual, and every individual acquires its power in the survival process in terms of its fitness value. Higher the fitness values, better the individuals performance in the evolution process. A fixed number of individuals correspond to a generation. GA is an iterative algorithm such that in every generation, first parents are selected depending on their fitness values, and then by some genetic operators the strings of children are produced. With their calculated fitness values, the new generation is obtained. And this procedure is repeated until some stopping criterion is met. "Boyabatli and Sabuncuoglu, 2004

" The power and simplicity of GA make it popular for even largescale optimization problems. The main advantage of GA is that it does not require neither mathematical expression of response surfaces nor any derivative or gradient information "Boyabatli and Sabuncuoglu, 2004

2.1.1 Encoding

Binary, Hex, ...

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

" Traditionally, binary encodings have been used because they are easy to implement and maximize the number of schemata processed. The crossover and mutation operators described in the previous sections are specific only to binary encodings. When alphabets other than [OJ] are used, the crossover and mutation operators must be tailored appropriately. "Srinivas and Patnaik, 1994

" Binary encoding is the commonly used encoding scheme. Each gene or chromosome is represented as a string of 1 or 0 [187]. "Katoch, Chauhan, and Kumar, 2021

" Binary encoding scheme is not appropriate for some engineering design problems due to epistasis and natural representation. In octal encoding scheme, the gene or chromosome is represented in the form of octal numbers (0–7). In hexadecimal encoding scheme, the gene or chromosome is represented in the form of hexadecimal numbers (0–9, A-F) [111,125,187]. T "Katoch, Chauhan, and Kumar, 2021

" In value encoding scheme, the gene or chromosome is represented using string of some values. These values can be real, integer number, or character [57]. This encoding scheme can be helpful in solving the problems in which more complicated values are used. "Katoch, Chauhan, and Kumar, 2021

" In tree encoding, the gene or chromosome is represented by a tree of functions or commands. These functions and commands can be related to any programming language. This is very much similar to the representation of repression in tree format [88]. This type of encoding is generally used in evolving programs or expressions. "Katoch, Chauhan, and Kumar, 2021

" A large number of optimization problems have real-valued continuous variables. A common method of encoding them uses their integer representation. Each variable is first linearly mapped to an integer defined in a specified range, and the integer is encoded using a fixed number of binary bits. "Srinivas and Patnaik, 1994

Use Controll Parameter instead of Hyperparams???

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.

Population

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.

" As Srinivas and Patnaik explained in their survey on GAs,24 increasing the population size also increases the diversity of individuals, which reduces the chances that the search converges to a local optimum. However, if the population size is too large, converging to optimal regions takes longer than required. Considering a fixed time budget for testing, as is the case for this study, with increasing population size also, the impact of genetic search is increasingly fading as fewer optimization steps can be performed between generations. "Klampfl, Klück, and Wotawa, 2023

" The initial population P(O) can be chosen heuristically or at random. "Grefenstette, 1986

" GA s generally do poorly with very small populations [22], because the population provides an insufficient sample size for most hyperplanes. A large population is more likely to contain representatives from a large number of hyperplanes. Hence, the GA's can perform a more informed search. As a result, a large population discourages premature convergence

to suboptimal solutions. On the other hand, a large population requires more evaluations per generation, possibly resulting in an unacceptably slow rate of convergence. "Grefenstette, 1986

" Population: Population is a collection of individuals. Population specifies the options for the population size in GA. The two important aspects are the initial population generation and the population size. "Majumdar and Ghosh, 2015

"Initial population is always considered as an important factor for the performance of genetic algorithms. The size of population also affects the quality of solution [160]. The researchers argue that if a large population is considered, then the algorithm takes more computation time. However, the small population may lead to poor solution [155]. "Katoch, Chauhan, and Kumar, 2021

" Premature convergence is a common issue for GA. It can lead to the loss of alleles that makes it difficult to identify a gene [15]. Premature convergence states that the result will be suboptimal if the optimization problem coincides too early. "Katoch, Chauhan, and Kumar, 2021

" higher selection pressure can decrease the population diversity that may lead to premature convergence [71]. "Katoch, Chauhan, and Kumar, 2021

" The GA begins by generating a random population of individuals, where each individual consists of an appropriate length bit string representing values for every variable of a problem to be solved. The population size is a control parameter of the GA. The GA evaluates the fitness of many populations of individuals over time, where each population is called a generation. The population of individuals for generation n+1 is created through some transformation of individuals composing generation n. "Mills, Filliben, and Haines, 2015

" DeJong also observes that larger population size increases parallelism, which aids in solving complex problems, but that there is diminishing return to increasing population size. He reports that choosing a selection method is difficult due to interactions with population size. "Mills, Filliben, and Haines, 2015

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

" GA is one of those algorithms whose performance is highly affected by the choice of selection operator. Without this mechanism, GA is only simple random sampling giving different results in each generation. Hence, we can say that the selection operator is the backbone of the GA process. " Hussain and Muhammad, 2020

" The tournament selection (TS) is also widely used as an alternative to FPS. In TS, first, randomly select the t (where t is the predefined tournament size) individuals from the population and then they compete against each other based on their fitness. An individual with higher fitness value is declared as a winner and selected for mating process. The selection pressure can be adjusted with change the tournament size [7]. Usually, the most used tournament size is 2 (binary tournament selection (BTS)), which is the simplest form of TS [21]. However, the larger tournament size can be used to enhance the competition among individuals, but it leads to loss of population diversity [22,23]. "Hussain and Muhammad, 2020

" The first selection mechanism for GA was fitness proportional selection (FPS), which was introduced by Holland [1]. Now, it has become the most prevalent selection approach which used the concept of proportionality. It works as the fitness value of each individual in a population corresponds to the area of roulette wheel proportions. Then, an individual is marked by the roulette wheel pointer after it has spun. This operator gives individuals, a probability pi of being selected Eq. (1) that is directly proportionate to their fitness: However, the difficulty is encountered when a significant difference appears in the fitness values [14,17,18]. The scaling problem which is the major drawback of this scheme was first pointed out by Grefenstette [19]. It has happened when population evolves, the ratio between the variance

and the fitness average becomes increasingly small. The selection pressure, therefore, drops as the population converges [7]. On the other hand, high selection pressure may lead to premature convergence to a sub-optimal solution.

The most popular technique is the linear rank selection (LRS) scheme proposed by Baker [20]. It sorts the individuals in the sequence as worst to best according to the fitness and allocates them a survival probability proportional to their rank order. After this task, a sampling procedure (i.e., roulette wheel sampling) is used to select the individuals for mating process. In this way, the LRS can maintain a constant selection pressure throughout in the sampling process, because it introduces a uniform scaling across the population. The weakness of this scheme is that it can lead to slower convergence, because there is no significant difference between the best and other individuals. The selection probability of two consecutive chromosomes by the same amount is regardless of whether the gap between their fitness is larger or smaller [7].

Another rank-based selection scheme is exponential ranking selection (ERS). It works similar as to LRS, except for the non-linear assignment of probabilities to the individuals. "Hussain and Muhammad, 2020

" An ideal situation may exist, if the selection pressure is low at the early stage of the search to gives a free hand to an exploration of the solution space and enhance at the ending stage to help the algorithm for convergence [26]. Hence, to trade-off between these two competing criteria, an adjustable selection pressure must desired [7]. "Hussain and Muhammad, 2020

" The main contribution of this article is in the development of the proposed selection approach which reduces the weakness associated with FPS and LRS in the GA procedure. The proposed approach is based on the ranking scheme which splits the individuals after ranking and then assign them probabilities for selection. This will increase the competition among individuals to be selected for mating process to regulate the selection pressure. "Hussain and Muhammad, 2020

" The LRS introduces slow convergence speed and sometimes converges to a sub-optimal solution as less fit individuals may be preserved from one generation to another. In GA, the FPS has the essence of exploitation, while LRS

is influenced by exploration. The information about the relative evaluation of individuals is ignored, all cases are treated uniformly regardless of the magnitude of the problem and, finally, the schema theorem is violated. LRS prevents too quick convergence and differs from FPS in terms of selection pressure. This discussion suggests that, whenever a selection procedure is used, some kind of adaptation of the selection pressure is highly desirable. "Hussain and Muhammad, 2020

" In this research, we propose an alternative selection scheme [split rank selection (SRS)] that maintains a fine balance between exploration and exploitation.

In the proposed procedure, the individuals are ranked according to their fitness scores from worst to best, thus overcoming the fitness scaling issue. After this, split the whole population into two portions and assigning them probabilities for selection based on their ranks. "Hussain and Muhammad, 2020

" selection operator is a crucial strategy in GA, because it has a vital role in exploring the new areas of the search space and converges the algorithm "Hussain and Muhammad, 2020

" Selection (Reproduction): Selection is the process of choosing two parents from the population for crossing. Some of the various selection methods are stochastic uniform, remainder, roulette wheel selection, random selection, rank selection, Figure 1 : Flowchart of Genetic Algorithm(GA) tournament selection, elitism etc. "Majumdar and Ghosh, 2015

" Roulette wheel selection maps all the possible strings onto a wheel with a portion of the wheel allocated to them according to their fitness value. This wheel is then rotated randomly to select specific solutions that will participate in formation of the next generation [88]. However, it suffers from many problems such as errors introduced by its stochastic nature. "Katoch, Chauhan, and Kumar, 2021

" De Jong and Brindle modified the roulette wheel selection method to remove errors by introducing the concept of determinism in selection procedure. Rank selection is the modified form of Roulette wheel selection. It utilizes the ranks instead of fitness value. Ranks are given to them according to their fitness value so that each individual gets a chance of getting selected

according to their ranks. Rank selection method reduces the chances of prematurely converging the solution to a local minima [88]. "Katoch, Chauhan, and Kumar, 2021

" Tournament selection technique was first proposed by Brindle in 1983. The individuals are selected according to their fitness values from a stochastic roulette wheel in pairs. After selection, the individuals with higher fitness value are added to the pool of next generation [88]. "Katoch, Chauhan, and Kumar, 2021

" Stochastic universal sampling (SUS) is an extension to the existing roulette wheel selection method. It uses a random starting point in the list of individuals from a generation and selects the new individual at evenly spaced intervals [3]. It gives equal chance to all the individuals in getting selected for participating in crossover for the next generation. Although in case of Travelling Salesman Problem, SUS performs well but as the problem size increases, the traditional Roulette wheel selection performs relatively well [180]. "Katoch, Chauhan, and Kumar, 2021

" Boltzmann selection is based on entropy and sampling methods, which are used in Monte Carlo Simulation. It helps in solving the problem of premature convergence [118]. "Katoch, Chauhan, and Kumar, 2021

" However, there is a possibility of information loss. It can be managed through elitism [175]. Elitism selection was proposed by K. D. Jong (1975) for improving the performance of Roulette wheel selection. It ensures the elitist individual in a generation is always propagated to the next generation. If the individual having the highest fitness value is not present in the next generation after normal selection procedure, then the elitist one is also included in the next generation automatically [88]. "Katoch, Chauhan, and Kumar, 2021

"Qualitative analysis of the selection strategies is depicted, and the numerical experiments show that SGA with tournament selection strategy converges much faster than roulette wheel selection." Jinghui Zhong et al., 2005

" In the operation of selection, the selection of an individual is based on its fitness value. The better the fitness of the individual is, the larger the probability it is to be chosen. Without any changes, the selected one is replicated into the next generation of the population. In addition, individuals

with poor fitness value may be selected because the genetic selection is probabilistic. To implement the operation of selection, strategies such as roulette wheel and tournament selection can be used. Different selection strategies affect the performance of the algorithm differently. Crossover is performed on two selected individuals at a time. With a probability, two individuals are chosen, and then the crossover point is selected randomly, and the pair of selected individuals undergoes the process of crossover. "Jinghui Zhong et al., 2005

"

Roulette wheel selection is the most frequently used selection strategy. "Jinghui Zhong et al., 2005

" Tournament selection is also a selection strategy which selects individuals based on their fitness value. The basic idea of this strategy is to select the individual with the highest fitness value from a certain number of individuals in the population into the next generation. In the tournament selection, there is no arithmetical computation based on the fitness value, but only comparison between individuals by fitness value. The number of the individuals taking part in the tournament is called tournament size. "Jinghui Zhong et al., 2005

" 1) Randomly select several individuals from the population to take part in the tournament. Choose the individual that has the highest fitness value from the individuals selected above by comparing the fitness value of each individual. Then the chosen one is copied into the next generation of the population. 2) Repeat step1 n times where n is the number of individuals of the population. "Jinghui Zhong et al., 2005

" From the results shown above, SGA using tournament selection always obtains the satisfied solutions with more times at earlier generations than roulette wheel selection. This indicates SGA based on tournament selection converges more quickly than roulette wheel selection. "Jinghui Zhong et al., 2005

" Selection.Selection models nature's survival-of-the-fittest mechanism. Fitter solutions survive while weaker ones perish. In the SGA, a fitter string receives a higher number of offspring and thus has a higher chance of surviving in the subsequent generation. "Srinivas and Patnaik, 1994

" Selection provides the favorable bias toward building blocks with higher fitness values and ensures that they increase in representation from generation to generation. "Srinivas and Patnaik, 1994

" In the initial generations of the CA,the population typically has a low average fitness value. The presence of a few strings with relatively high fitness values causes the proportionate selection scheme to allocate a large number of offspring to these “superstrings,” and they take over the population, causing premature convergence. A different problem arises in the later stages of the CAwhen the population has converged and the variance in string fitness values becomes small. The proportionate selection scheme allocates approximately equal numbers of offspring to all strings, thereby depleting the driving force that promotes better strings. Scaling mechanisms and rankbased selection schemes overcome these two problems. "Srinivas and Patnaik, 1994

" An alternate way to avoid the twin problems that plague proportional selection is rank-based selection, which uses a fitness value-based rank of strings to allocate offspring. The scaled fitness values typicallyvary linearly with the rank of the string. The absolute fitness value of the stringdoes not directlycontrol the number of its offspring. To associate each string with a uniquerank,this approach sortsthe stringsaccordingto their fitnessvalues,introducing the drawback of additional overhead in the GA computation. Another mechanism is tournament selection. For selection,a stringmust win a competition with a randomly selected set of strings. In a k-arytournament, the best of k stringsis selectedfor the next generation. "Srinivas and Patnaik, 1994

Crossover

Crossover is the mating process. " The higher the crossover rate, the more quickly new structures are introduced into the population. If the crossover rate is too high, high-performance structures are discarded faster than selection can produce improvements. If the crossover rate is too low, the search may stagnate due to the lower exploration rate. "Grefenstette, 1986

" On the one hand, increasing the crossover probability also increases the recombination of gene strings, supporting exploration and exploitation. On the other hand, it also increases the disruption of individuals with good gene sequences, which can be a disadvantage. "Klampfl, Klück, and Wotawa, 2023

" Crossover serves two complementary search functions. First, it provides new points for further testing within the hyperplanes already represented in the population.

Second, crossover intro- duces representatives of new hyperplanes into the popula- tion. "Grefenstette, 1986

" Crossover (Recombination): Crossover combines two individuals, or parents, to form a new individual, or child, for the next generation. Some of its types are scattered, single point crossover, two point crossover, intermediate, heuristic, arithmetic etc. "Majumdar and Ghosh, 2015

" Crossover. After selection comes crossover,SGA's crucial operation. Pairs of strings are picked at random from the population to be subjected to crossover. The SGA uses the simplest approach single-point crossover. Assuming that 1 is the string length, it randomly chooses a crossover point that can assume values in the range 1to 1- 1. The portions of the two strings beyond this crossover point are exchanged to form two new strings. The crossover point may assume any of the 1- 1possible values with equal probability. Further, crossover is not always effected. After choosing a pair of strings, the algorithm invokes crossover only if a randomly generated number in the range 0 to 1 is greater than pc, the crossover rate. (In GA literature, the term crossover rate is also usedto denote the probability of crossover.) Otherwise the strings remain unaltered. The value ofp,. lies in the range from 0 to 1. In a large population,p, gives the fraction of strings actually crossed. "Srinivas and Patnaik, 1994

" Crossover tends to conserve the genetic information present in the strings to be crossed. Thus, when the strings to be crossed are similar, its capacity to generate new building blocks diminishes. "Srinivas and Patnaik, 1994

" Crossover operators are used to generate the offspring by combining the genetic information of two or more parents. "Katoch, Chauhan, and Kumar, 2021

" In a single point crossover, a random crossover point is selected. The genetic information of two parents which is beyond that point will be swapped with each other [190]. Figure 3shows the genetic information after swapping. It replaced the tail array bits of both the parents to get the new offspring. "Katoch, Chauhan, and Kumar, 2021

" In a two point and k-point crossover, two or more random crossover points are selected and the genetic information of parents will be swapped as per the segments that have been created [190]. "Katoch, Chauhan, and Kumar, 2021

" In a uniform crossover, parent cannot be decomposed into segments. The parent can be treated as each gene separately. We randomly decide whether we need to swap the gene with the same location of another chromosome [190]. "Katoch, Chauhan, and Kumar, 2021

" Partially matched crossover (PMX) is the most frequently used crossover operator. It is an operator that performs better than most of the other crossover operators. The partially matched (mapped) crossover was proposed by D. Goldberg and R. Lingle [66]. Two parents are choose for mating. One parent donates some part of genetic material and the corresponding part of other parent participates in the child. Once this process is completed, the left out alleles are copied from the second parent [83]. "Katoch, Chauhan, and Kumar, 2021

" Order crossover (OX) was proposed by Davis in 1985. OX copies one (or more) parts of parent to the offspring from the selected cut-points and fills the remaining space with values other than the ones included in the copied section. "Katoch, Chauhan, and Kumar, 2021

" If crossover is not considered during evolution, then the algorithm can result in local optima. "Katoch, Chauhan, and Kumar, 2021

" Crossover consists of splitting a chromosome pair at a random location with reciprocal exchange of the two halves, thus causing a mutual w of information between pairs. "Marsili Libelli and Alba, 2000

" In the two-point crossover scheme, two crossover points are randomly chosen and segments of the stringsbetween them are exchanged. Two-point

crossover eliminates the single-point crossover bias toward bits at the ends of strings. "Srinivas and Patnaik, 1994

" An extension of the two-point scheme, the multipoint crossover, treats each string as a ring of bits divided by k crossover points into k segments. One set of alternate segments is exchanged between the pair of strings to be crossed. "Srinivas and Patnaik, 1994

" Uniform crossover exchanges bits of a string rather than segments. At each string position, the bits are probabilistically exchanged with some fixed probability. The exchange of bits at one string position is independent of the exchange at other positions. "Srinivas and Patnaik, 1994

" To classify techniques,we can use the notions of positional and distributional biases. A crossover operator has positional bias if the probability that a bit is swapped depends on its position in the string. Distributional bias is related to the number of bits exchanged by the crossover operator. If the distribution of the number is nonuniform,the crossover operator has a distributional bias. Among the various crossover operators, single-point crossover exhibitsthe maximum positional bias and the least distributional bias. Uniform crossover, at the other end of the spectrum, has maximal distributional bias and minimal positional bias. "Srinivas and Patnaik, 1994

" At one end, uniform crossover swaps bits irrespectiveof their position, but its higher disruptive nature often becomes a drawback. Two-point and single-point crossover preserve schematabecause of their low disruption rates, but they become less exploratory when the population becomes homogeneous. "Srinivas and Patnaik, 1994

" A related issue is theinterplay between the population size and the type of crossover. Empirical evidence suggests that uniform crossover is more suitable for small populations, while for larger populations, the less disruptive two-point crossover is better. Uniform crossover's disruptivenesshelps sustain a highly explorative search in small populations. The inherent diversity in larger populations reduces the need for exploration and makes two-point crossover more suitable. "Srinivas and Patnaik, 1994

" Increasing the crossover probability increases recombination of building blocks, but it also increases the disruption of good strings. "Srinivas and Patnaik, 1994

Mutation

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

" Setting the mutation probability too high may transform the search procedure into random testing, but it also helps to introduce new gene material, which promotes the exploration of new input space regions. "Klampfl, Klück, and Wotawa, 2023

" Mutation is a secondary search operator which increases the variability of the population. "Grefenstette, 1986

" A low level of mutation serves to prevent any given bit position from remaining forever converged to a single value in- the entire population. A high level of mutation yields an essentially random search. "Grefenstette, 1986

" With a larger population and higher mutation rate, the population will tend to contain more variety, thus increasing the random aspects of the GA. "Grefenstette, 1986

" The absence of mutation is also associated with poorer performance, which suggests that mutation performs an important service in refreshing lost values. "Grefenstette, 1986

" After crossover. strings are subjected to mutation. Mutation of a bit involves flipping it: changing a 0 to 1 or vice versa. Just asp, controls the probability of a crossover, another parameter. P,, (the mutation rate), gives the probability that a bit will be flipped.The bits of a string are independently mutated that is, the mutation of a bit does not affect the probability of mutation of other bits. The SGA treats mutation only as a secondary operator with the role of restoring lost genetic material. "Srinivas and Patnaik, 1994

" Increasing the mutation probability tends to transform the geneticsearch into a random search, but it also helps reintroduce lost genetic material. "Srinivas and Patnaik, 1994

" Mutation is not a conservative operator and can generate radically new buildingblocks. "Srinivas and Patnaik, 1994

" Mutation: After crossover, the springs are subjected to mutation. Mutation functions make small random changes in the individuals in the population, which provide genetic diversity and enable the genetic algorithm to search a broader space. The different forms of mutation are constraint dependent, uniform, adaptive feasible etc. Mutation of a bit involves flipping it, changing between 0 to 1 and vice versa with a small mutation probability. "Majumdar and Ghosh, 2015

" Mutation is an operator that maintains the genetic diversity from one population to the next population. "Katoch, Chauhan, and Kumar, 2021

" If the mutation is not considered during evolution, then there will be no new information available for evolution. "Katoch, Chauhan, and Kumar, 2021

"In Genetic Algorithms mutation probability is usually assigned a constant value, therefore all chromosome have the same likelihood of mutation irrespective of their ®tness."Marsili Libelli and Alba, 2000

" Mutation introduces random binary changes in a chromosome. Usually the mutation likelihood is kept constant at a low value for all bits. In this paper, a new mechanism of mutation will be introduced and the consequences of this modi®cation assessed. "Marsili Libelli and Alba, 2000

Adaptive Mutation "It is shown in this paper that making mutation a function of ®tness produces a more ef®cient search. This function is such that the least signi®cant bits are more likely to be mutated in high-®tness chromosomes, thus improving their accuracy, whereas low-®tness chromosomes have an increased probability of mutation, enhancing their role in the search. In this way, the chance of disrupting a high-®tness chromosome is decreased and the exploratory role of low-®tness chromosomes is best exploited. " Marsili Libelli and Alba, 2000

" The weak point of "classical" GAs is the total randomness of mutation, which is applied equally to all chromosomes, irrespective of their fitness. Thus a very good chromosome is equally likely to be disrupted by mutation as a bad one. On the other hand, bad chromosomes are less likely to produce good ones through crossover, because of their lack of building blocks, until

they remain unchanged. They would benefit the most from mutation and could be used to spread throughout the parameter space to increase the search thoroughness. So there are two conflicting needs in determining the best probability of mutation. Usually, a reasonable compromise in the case of a constant mutation is to keep the probability low to avoid disruption of good chromosomes, but this would prevent a high mutation rate of low-fitness chromosomes. Thus a constant probability of mutation would probably miss both goals and result in a slow improvement of the population. "Marsili Libelli and Alba, 2000

" This paper has presented an improved search algorithm to reduce the chance that high-®tness chromosomes are muted during the search, thus losing their favourable schemata. In GAs, mutation is usually assigned a constant probability and thus all chromosome have the same likelihood of mutation irrespective of their ®tness. Conversely, making mutation a function of ®tness produces a more ef®cient search. "Marsili Libelli and Alba, 2000

" signi®cant bits are likely to be muted in high-®tness chromosomes, thus improving their accuracy, whereas low-®tness chromosomes are much more likely to change, enhancing their exploratory role in the search. "Marsili Libelli and Alba, 2000

" In all cases the new algorithm showed a faster improvement of the maximum ®tness, which was always greater than the one produced by the classical GA at the same generation. "Marsili Libelli and Alba, 2000

" In 2007, DeJong took a broader view, considering what was known with respect to the wider community of evolutionary algorithms (EAs). He observed that while it appears that adapting mutation rate on-line during execution provides advantages, most EAs are deployed with a default set of static parameter values that have been found quite robust in practice. "Mills, Filliben, and Haines, 2015

" Parameter settings optimal in the earlier stages of the search typically become inefficient during the later stages. Similarly, encodings become too coarse as the search progresses, and the fraction of the search space that the GA focuses its search on becomes progressivelysmaller. To overcome these drawbacks, several dynamic and adaptive strategiesfor varying the control

parameters and encodings have been proposed. One strategy exponentially decreases mutation rates with increasing numbers of generations, to gradually decrease the search rate and disruption of strings as the population converges in the search space. Another approach considers dynamically modifying the rates at whichthe various genetic operators are used, based on their performance. Each operator is evaluated for the fitness values of strings it generates in subsequent generations. Very often, after a large fraction of the population has converged (the strings have become homogeneous), crossover becomes ineffective in searching for better strings. Typically, low mutation rates (0.001 to 0.01) are inadequate for continuing exploration. In such a situation, a dynamic approach for varying mutation rates based on the Hamming distance between strings to be crossed can be useful. The mutation rate increasesas the Hamming distancebetween strings decreases. As the strings to be crossed resemble each other to a greater extent,the capacity of crossover to generate new strings decreases, but the increased mutation rate sustainsthe search. "Srinivas and Patnaik, 1994

Other

Diversity " In initial stage of GA, the similarity between individuals is very low. The value of R should be low to ensure that the new population will not destroy the excellent genetic schema of individuals. At the end of evolution, the similarity between individuals is very high as well as the value of R should be high. "Katoch, Chauhan, and Kumar, 2021

Fitness Function " The Fitness Function: The fitness function in Genetic Algorithm represents the objective function and the fitness value corresponds the performance of an individual chromosome. "Majumdar and Ghosh, 2015

" Multiobjective GA (MOGA) is the modified version of simple GA. MOGA differ from GA in terms of fitness function assignment. The remaining steps are similar to GA. The main motive of multiobjective GA is to generate the optimal Pareto Front in the objective space in such a way that no further

enhancement in any fitness function without disturbing the other fitness functions [123]. "Katoch, Chauhan, and Kumar, 2021

" The concept of Pareto dominance was introduced in multiobjective GAs. Fonseca and Fleming [56] developed first multiobjective GA (MOGA). The niche and decision maker concepts were proposed to tackle the multimodal problems. However, MOGA suffers from parameter tuning problem and degree of selection pressure. "Katoch, Chauhan, and Kumar, 2021

" The majority of applications of optimization tools to subsurface remediation problems have been based on single objective optimization methods. Single objective methods can accommodate multiobjective problems in several ways, such as minimizing a weighted, linear combination of the objective functions or minimizing a single objective while transforming the remaining ob- jectives into constraints. However, these methods rely on a priori knowledge of the appropriate weights or constraint values. Furhtermore they are only capable of findig individual points on the tradeoff curve(or sureace) for each problem solution.

As already mentioned, previous approaches for solving the multiobjective problem have involved re- ducing the problem dimension, either by combining all objectives into a single objective (e.g. [27]) or optimizing one while the rest are constrained (e.g., [3]). "Erickson, Mayer, and Horn, 2002

" True multiobjective methods have the potential to simultaneously generate all possible optimal combina- tions of objectives, with less effort than other ap- proaches. Multiobjective problems involve several objective functions, each of which is a function of de- cision ðdÞ and state variables ðsÞ. tiveproblemcanbestatedas: "Erickson, Mayer, and Horn, 2002

" Multiobjective approaches in this category operate on the concept of “Pareto domination”, which states that one candidate dominates another only if it is at least equal in all objectives and superior in at least one "Erickson, Mayer, and Horn, 2002

More on multiobjecte approaches can be found in Erickson, Mayer, and Horn, 2002.

Stopping Criteria " Stopping criteria: Stopping criteria determines what causes the algorithm to terminate-generations, time limit, fitness limit etc. "Majumdar and Ghosh, 2015

This is not used for this GA

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.

3 Implementation

3.1 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 which defines the positions and types of vehicles and pedestrians(i.e. actors). A simulation always consist of at least one EGO vehicle. Additionally any number of NPCs can be used.

While the NPCs are only controlled by the Traffic Manager, the ego vehicle can be either partly or even completely controlled by an ADAS/AD Function. The stated goal is to test these functions for errors.

For all simulations done by this thesis, the Traffic Manager was set to 100Hz.

3.1.1 Action Interface

To control the behaviour of the actors inside the simulation, actions can be requested over the "Action Interface" provided by the Traffic Manager. An action will request a certain behaviour from an actor. An action can be set to at any timestep for any actor[1]. Pedestrians and vehicles have a different set of actions.

Insert graph of action interface

[1]depending on the ADAS/AD function under test, the Action Interface might be disabled for the EGO vehicle

If no action is set, the actor will behave in a normal manner inside the simulation. This means that the actor will follow along its path until a new action changes its behaviour.

The following list are 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.

All these actions are accessed by the Genetic Algorithm and the Behavior tree. The Behaviour tree sets only actions for the EGO vehicle, while the Genetic Algorithm will set all actions for the other actors in the simulation.

3.2 Genetic Algorithm

For implementing the Genetic Algorithm, DEAP was chosen. It is a popular tool for academia and allows for high customacibility. As has been stated in section 3.1.1, it has full access over setting actions for all NPCs. Using theses actions, it tries to optimize a cost function. This section aims to explain how the genetic algorithm was implemented and which different hyperparameters are variable. In chapter 4, the best hyperparameter combination will be generated, further analysis is done in 5.

cite

cite 3 examples

A few default settings for the genetic algorithm had to be chosen. It was decided that the genetic algorithm will set an action per actor every 50 steps, which translates to 0.5 seconds (simulation runs at 100hz). In other words, every 50 steps of the simulation is 1 timestep for the genetic algorithm. The time of a simulation is always set to 35 seconds. Each genetic algorithm will run for 30 generations. These to settings were chosen with the aim of reducing the amount of needed computations.

If the GA decides to not set an action for the integer, it sets "NoAction" as a placeholder.

3.2.1 Encoding

When implementing a Genetic Algorithm, it is necessary to implement an encoding that fits to the problem. Each individual basically thus needs to include all actions that the genetic algorithm wants to apply. Different encodings presented in section 2.1.1, however none directly fitted to the problem presented. A custom encoding for both chromosomes and genes needed to be generated.

Chromosome

ref

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

Time The first encoding is will be called "Time". Each gene corresponds to 1 timestep (so 1 gene per every 0.5 seconds). One gene has a list of the length of the number of all NPCs. This list is populated with actions. The index of an action in the list corresponds to the NPC id (index + 1 as the ego has id == 0, thus a start at 1 is needed). A visualization is seen in figure 3.1.

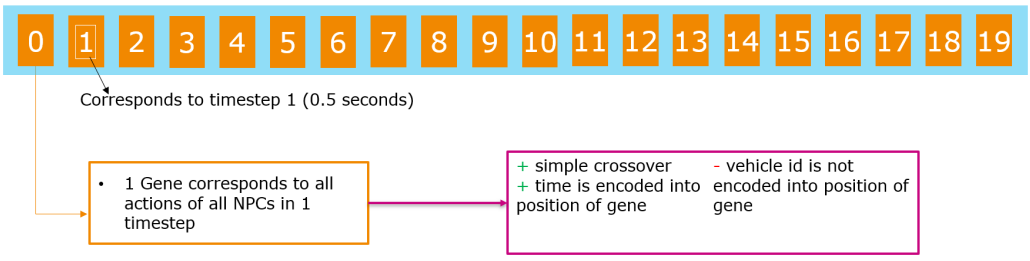

Figure 3.1: Time

Given the previously stated simulation time of 35 seconds, each chromosome has a length of $35 * 2 = 70$ genes. Each gene consits of $number\_of\_actors$ actions. Crossover can thus only move all actions of a timestep at once, modifing between actions of the same timestep can only be done using mutation. If this is desired will be seen in the next chapters.

TimeNPC The second encoding has the name "TimeNPC", and is somewhat differently structured. Now, genes only hold 1 action, encoding now not only the timestep, but also the actor id in the position of the gene inside

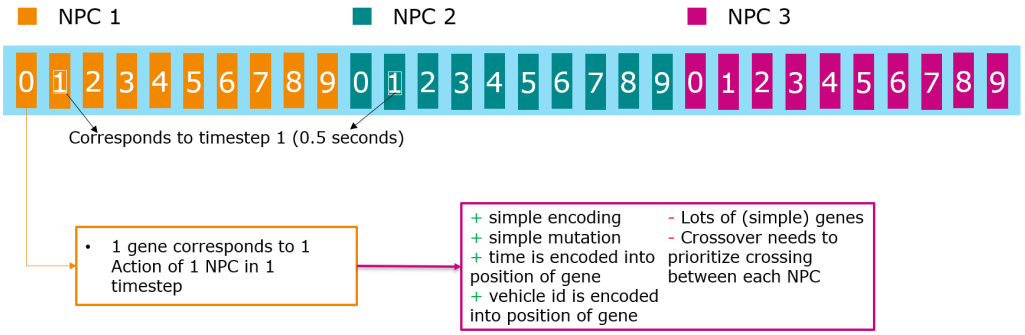

Figure 3.2: Time + NPC

the chromosome. Now, each actors actions will be listed one after another. This is visualized in figure 3.2.

Now, each gene has a length of 1 and each chromosome now has a length of $35 * 2 * number\_of\_actors$, which makes them much longer compared to the previous encoding. This now allows the crossover operation to modify only specific actions of one timesstep. Previously this was not possible.

However for this encoding to make sense, the crossover operations "One-Point" and "TwoPoint" had to be modified as follows. In an example of 10 NPCs, the operations will be executed for each NPC separately. Otherwise these two operations would have only had an effect on 1 or 2 different NPCs. For the reaiming NPCs, their actions would stay the same.

Gene

Two different encodings for genes were implemented as well. A gene always consits of a list, which depending on the chromosome type either has a length of $number\_of\_actors$ (In case ChromosomeEncoding == Time) or of length 1 (in case ChromosomeEncoding == TimeNPC). The following two encodings thus show the type of object, which is in these lists.

Integer The first encoding uses integer, which are translated into actions when the simulation is started. For each action, a range of integers is assigned, the larger the range, the more likely the action is chosen by the GA. Actions that have parameters are split into different ranges, according to which paramters make sense. For example ModifyTargetVelocity is split into five different parts, with different percentages, namely 50, 70, 100, 130, 160. The range of integers assigned to these parts is different. A percentage setting of 100 for example has the largest integer range assigned. In Appendix , the probabilty of an actions can be seen. In Appendix ... the probabitly per actions of the parameters can be viewed.

These ranges were assigned based on intuition and trial and error. The encoding is visualized in 3.3.

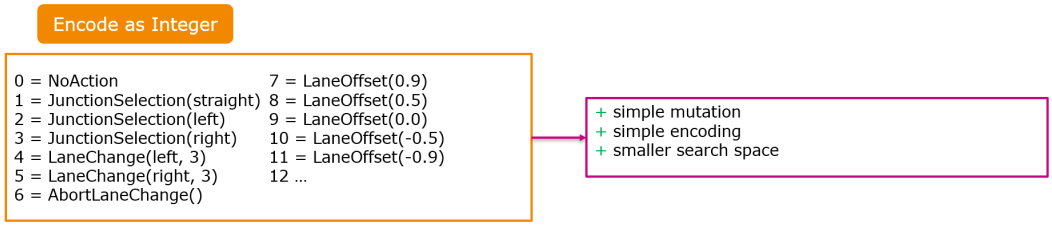

Figure 3.3: Integer

Dictionary The second encoding is much similar to the actual actions used in the simulations. Now, no translation is necessary anymore. During generation of the individuals, each action is again selected based on different probabilities assigned to actions, which again can be viewed in Appendix ... . These probablilites are the same as for the integer encoding. However the difference is, in case an action has parameters that need to be chosen. For each parameter, a range and a randomness function was chosen. For example in case of the percenetage parameter in ModifyTargetVelocity, the values are selected from a GausDistribution, with mu= 100, sigma=25 and a range limit between 0 and 300.

Again, these probability functions with settings were assigned based on intuition as well as trial and error. Detailed information can be seen in Appendix...

Figure 3.4 shows a visualization.

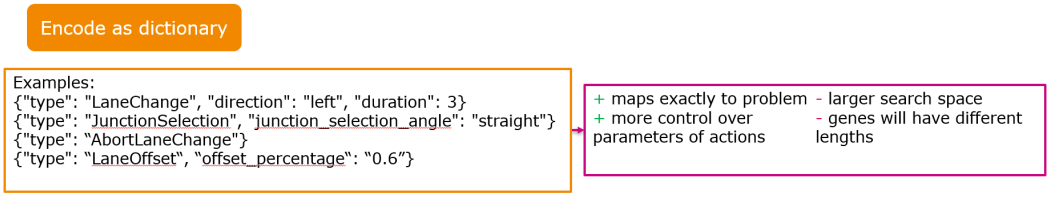

Figure 3.4: Dictionary

3.2.2 Cost Function

Cost function is a bit difficult, as we are only using internal values. No ADAS/AD system is tested and we thus have to work with what we got. This is the code of the cost function:

```
SEPS_PER_SECOND = 100
# allow emergency breaks to last only 3 seconds
MAX_DURATION = 3 * STEPS_PER_SECOND

cost = 0
duration_counter = 0
for i in range(len(result["ego_emergency_stop"])):
    if not result["ego_emergency_stop"][i]:
        # base cost for no current emergency break
        cost = cost + 1
        duration_counter = 0
    else:
        if duration_counter > MAX_DURATION:
            # increase cost if emergency break max
               duration is exceeded
            cost = cost + 10
        duration_counter += 1
return cost
```

result["ego_emergency_stop"] is a list with the length $100 * simulation_duration_seconds$ (because 100hz). It contains a boolean per step, if the EGO vehicle has initiated an emergency stop.

It would have been interesting to not only test for emergency stops (which will make the NPCs try to get the EGO to hard break often) but also improve time to collison (TTC), as was done by . However by the time of starting the testing, no working TTC functionality was implemented. Thus, only the emergency break cost function is used by the GA to be optimized.

Ref florian

3.3 Behavior Tree

Depending on the functionality under test, it is possible to let the EGO vehicle be controlled by a Behaviour Tree. This makes sense if for example a functionilty like AEB is tested, where only the breaks are controlled. In case of a full driving stack, no Behaviour Tree would be used.

The general idea is to have an EGO vehicle moving in a "relateable" manner trough the world. It will try to dodge standing or slow moving obstacles. This needs to be done in a determinisitc manner in order to no introduce randomness into the simulation.

For this, Behaviour Tree is used. While it has access to the same Action Interface (described in section 3.1.1) as the Genetic Algorithm, it is more tightly integrated with the Traffic Manger. While the Genetic Algorithm only ingests the results generated by the Simulation with the cost function, the Behaviour Tree needs access to internal functions during the simulation. The following figure shows the behaviour tree implemented.

Explain BT

Starting out,

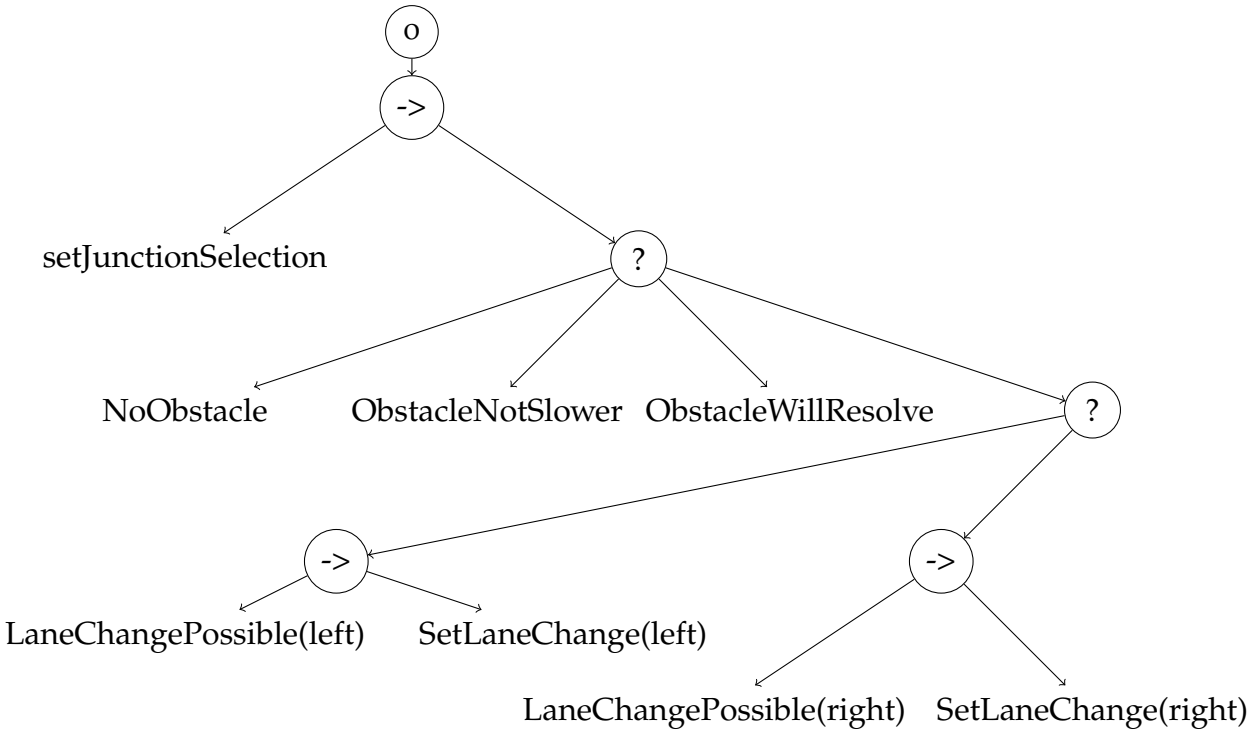

Figure 3.5: Used Behaviour Tree

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 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.

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.readthec

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

Fazal et al., 2005 recommends a population size of 50, a scattered crossover function with a crossover probability of 0.5. The used selection function was tournament selection. Elite count was set to 5.

Dao, Abhary, and Marian, 2016 suggests a population size of 200, two point crossover with a crossover probability of 0.7. A Gausian Mutation Function as well as roulette selection and elite count set to 1.

A population size of 200 and roulette selection is used by Assistant Professor, Amity University, Jaipur, Rajasthan, India et al., 2019. Further, the elite count is set to10. A heuristic crossover function with a crossover probability of 0.4 is also used.

TODO: Jinghui Zhong et al., 2005 set the range as "We set the value of POPSIZE as 50, 100, 150, 200, 250, the value of PXOVER as 0.1, 0.3, 0.5, 0.7, 0.9, and the value of PMUTATION as 0.05, 0.1, 0.15, 0.2, 0.25."

" DeJong believes that EAs pre-tuned with default parameter values for particular problem classes will continue providing better performance than EAs that attempt to dynamically adapt too many control parameters for specific problems. So, even after 30+ years of research, the question of best settings for EA and GA control parameters has no widely agreed answer. "Mills, Filliben, and Haines, 2015

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					
Settings	Code	32	48	64	96
C: 0.6, M: 0.01, TS: 2	A	3051	3016	2851	2871
C: 0.6, M: 0.01, TS: 4	B	3111	3021	3079	2937
C: 0.6, M: 0.2, TS: 2	C	3062	3010	3002	2831
C: 0.6, M: 0.2, TS: 4	D	3020	2967	2891	2850
C: 0.8, M: 0.01, TS: 2	E	3063	2892	2971	2916
C: 0.8, M: 0.01, TS: 4	F	3052	3049	3054	2897
C: 0.8, M: 0.2, TS: 2	G	3099	2940	2959	2869
C: 0.8, M: 0.2, TS: 4	H	3058	3005	2794	2809

Figure 4.1: List Settings per Population Size

In figure 4.2, 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.

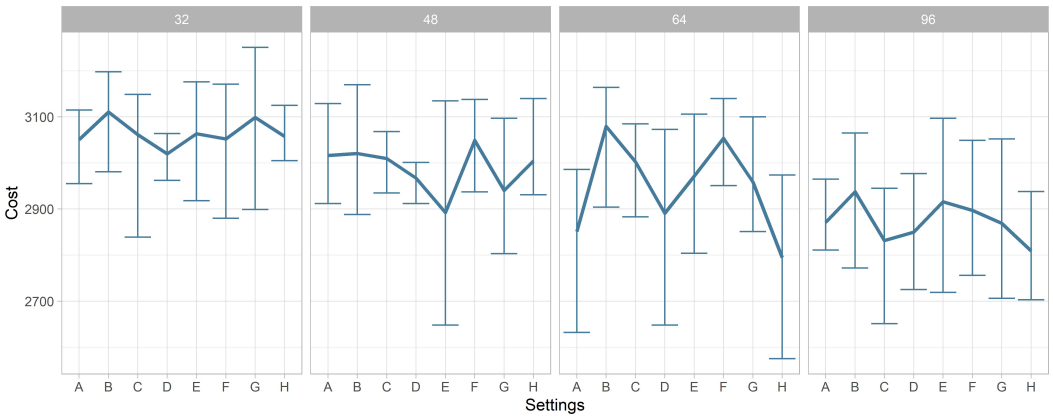

Figure 4.2: mean and error bars per population

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

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.

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. Even a higher level GA can be used

find references

A different approach called "design of experiment" (DOE), also known as statistically designed experiments. DOE tries to find the causeand-effect relationship between the factors and the output of experiments. It uses factorial design where each 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.

“If the range of variable is too small, then we may miss lots of useful information. If the range is too large, then the extreme values might give infeasible experimental runs.” (Yang and El-Haik, 2009)

Afterwards, main effects and interactions can be calculated to find the best settings per factor. It provides a graphical representation of these relationship by using interaction as well as main-effects charts. Using ANOVA (Analysis of Variance) it is possible to identify the significance of each factor and interaction, which enables the ranking of these factors. More details on these analysis tools will be provided in section 4.4.3.

A full factorial design will test over all possible combinations of the manually selected factor levels. Looking at the proposed factors in table 4.4.1, we would require 1024 runs[1], which not not feasible performance wise. A full factorial design has the drawback, that as the number if factors k gets increased, the number of needed experimental runs increases exponentially, thus resulting in lengthy experiments. Yang and El-Haik, 2009 state, that most of the results obtained by testing over all combinations are only used for estimating higher-order interactions, which are in most cases insignificant.

"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)"

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 Taguchi's proposed considerations, for more detail Roy, 1990 as well as Yang and El-Haik, 2009 is highly recommended.

Find examples of Papers that use taguchi for ga

Using a Taguchi design for evaluating the best hyperparamter has been successfully performed by Dao, Abhary, and Marian, 2016 as well as Assistant Professor, Amity University, Jaipur, Rajasthan, India et al., 2019.

" There are many similarities between "regular" experimental design and Taguchi's experimental design. However, in a Taguchi experiment, only the main effects and two-factor interactions are considered. Higher-order interactions are assumed to be nonexistent. In addition, experimenters are

[1]number of runs calculated using: `https://datatab.net/statistics-calculator/design-of-experiments`

asked to identify which interactions might be significant before conducting the experiment, through their knowledge of the subject matter." (Yang and El-Haik, 2009)

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). This means, that only a fraction of combinations needs to be tested which drastically improves performance. Each row of these matrices contains the factors of one experiment, while the columns correspond the factors Hamzaçebi, 2021.

Different orthogonal arrays have been proposed by Taguchi. The researcher has the responsibility to select an array based on the individual needs (Hamzaçebi, 2021). 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.

Definition orthogonal array

An orthogonal array has multiple properties: " OFF and OLH experiment designs sample a full factorial design space in a balanced and orthogonal fashion. Balance ensures good effect estimates by reducing an estimator's bias and variability. Orthogonality ensures good estimates of two-term interactions. Balance is achieved by ensuring that each level of every factor occurs an equal number of times in the selected sample. Orthogonality is achieved by ensuring that each pair of levels occurs an equal number of times across all experiment parameters in the selected sample. OFF and OLH designs exhibit good space-spanning properties, which aid screening, sensitivity, and comparative analyses. On the other hand, highly fractionated OFF or OLH designs can have poor space-filling properties, which are necessary for optimization analyses.

"Mills, Filliben, and Haines, 2015

As has been stated, probably the biggest drawback of using Taguchi orthogonal arrays is on the one hand to increased manual labour and on the other hand the fact, that higher order interactions ignored.

Roy, 1990 explains why this might not be a big problem: "Generally speaking, OA experiments work well when there is minimal interaction among factors; that is, the factor influences on the measured quality objectives are

independent of each other and are linear. In other words, when the outcome is directly proportional to the linear combination of individual factor main effects, OA design identifies the optimum condition and estimates performance at this condition accurately. If, however, the factors interact with each other and influence the outcome, there is still a good chance that the optimum condition will be identified accurately, but the estimate of performance at the optimum can be significantly off. The degree of inaccuracy in performance estimates will depend on the degree of complexity of interactions among all the factors."

This is complimented by Yang and El-Haik, 2009, who states, that: "During many years of applications of factorial design, people have found that higher-order interaction effects (i.e., interaction effects involving three or more factors) are very seldom significant. In most experimental case studies, only some main effects and two-factor interactions are significant."

Selection of orthogonal array When choosing a suitable Taguchi orthogonal array, 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. When selecting levels, it is important to have them "as far away from either side of the current working condition as possible."(Roy, 1990) In table 4.4.1, every factor with corresponding 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 standard 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 as well.

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 and modified to accommodate the needed factors. 4 Level Factors need additional space which will be generated using

column merging, while interactions will need to be assigned as well. For this, either an interaction table or linear graphs of the L_{16} array can be used (NazanDanacıoğlu, 2005). The linear graph approach is straight forward and will be selected. While there are multiple linear graphs for L_{16} array, 4.4.2 describes the graph which best fits the requirements from table 4.4.1. If no graph with the perfect fit is found, theses graphs can be modified as well, using rules described by NazanDanacıoğlu, 2005.

"In each of Taguchi's orthogonal arrays, there are one or more accompanying linear graphs. A linear graph is used to illustrate the interaction relationships in the orthogonal array."Yang and El-Haik, 2009

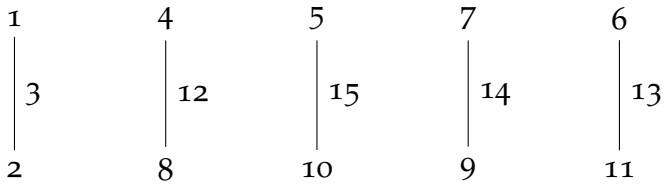

Figure 4.5: Linear Graph of $L_{16}(2^{15})$ taken from Yang and El-Haik, 2009

In a Taguchi linear graph, the nodes as well as the connections both represent columns in the orthogonal array. An interaction between to columns that are represented as nodes "comes out to" to the connecting line column Taguchi et al., 2005. This is useful for both analysing interactions between columns as well as combining (merging) interacting columns in case a higher factor is needed.

Column Merging A, B and C are both 4 level factors. The currently selected orthogonal only fits 2 level factors. Using column merging, it is possible to extend columns to accomodate higher order levels.

As calculated in 4.1, a four-level column requires three degrees of freedom, thus three two-level columns need to be merged. For column merging, it is required, that the to be merged columns are part of an interaction group (Yang and El-Haik, 2009).

So, 3 interaction 2-level columns need to first be selected. One column is discarded, the remain two columns need to be merged using the rules in tabular 4.6.

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

Figure 4.6: Rules taken from Roy, 1990

The four-level factor can then be assigned to this newly generated column. Because three four-level factors are needed for the current experiment, nine two-level columns need to be merged in total.

Assigning Interactions Interactions between two-level factors can be assigned using the linear graph as well. Here, select two connected nodes. The column describing their connection will subsequently contain the interaction (Taguchi et al., 2005).

An interaction between ChromosomeType and GeneType seems possible, thus D and E will be assigned to connected nodes in the linear graph. As we still have some unused space in the graph, we will also look at the interaction of TournamentSize and IndMutationProp (F and G). The resulting graph can be seen in 4.4.2.

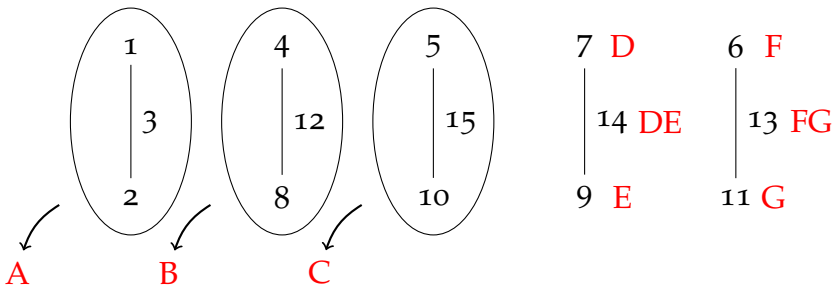

Figure 4.7: Modified Linear Graph to fit our needs

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 combinations table will subsequently be used as settings for the simulation runs.

4.4.3 Analysing the results

Table 4.9 can now can be used for running all the needed testcases (the interaction columns can be ignored until the evaluation). Transcoding all factors and levels to get the corresponding setting can be done using in the table from 4.4.1. We will repeat every setting 8 times to reduce randomness and gain information about variance. Running the Genetic Algorithm using

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

these 16 different settings each repeated 8 times took 10 days on the two previously described workstations. ref to section

The results are found in the appendix at 6.2.

Main-effects and interaction chart

Identifying the optimal conditions is done by analyzing the main effects per factor. Using them, it is possible to predict the factors, that lead to the best result Roy, 1990.

Yang and El-Haik, 2009 explains them well: “The main-effects chart is a plot of average responses at different levels of a factor versus the factor levels”

So, for every factor, sum up the mean of all results per level, then divide by the number of runs per level. Example for D

The resulting main-effect charts can be seen here:

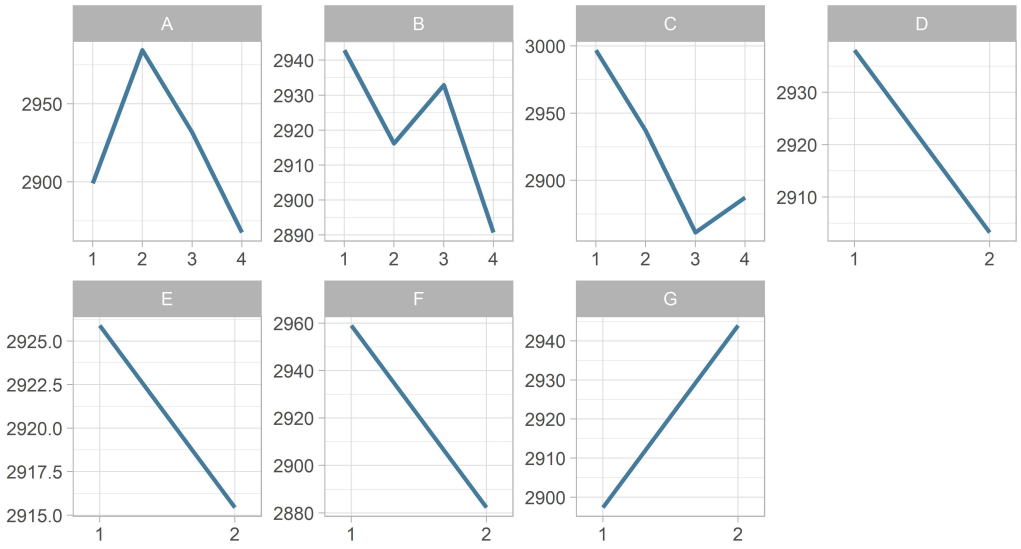

Figure 4.10: Main Effects

In case there is no interaction, the optimal setting is easily determined by using the main effects chart. Go over every factor in the chart and use the best level (in case of this experiment, the level with the lowest cost value). If interactions exist, they might have an influence on the best settings and need to be investigated (Yang and El-Haik, 2009).

Example for DE

To investigate previously defined interactions, a test of interactions can be used. Their calculation is similar to calculating main effects.

If lines cross, an interaction between the two factors exists. The more parallel the lines are, the less likely an interaction. Magnitude of the angle between the lines corresponds to the degree of interaction presence, according to Roy, 1990.

ANOVA

Before choosing the best settings, ANOVA analysis (analysis of variance) should be performend on the results. Among other things, this will provide

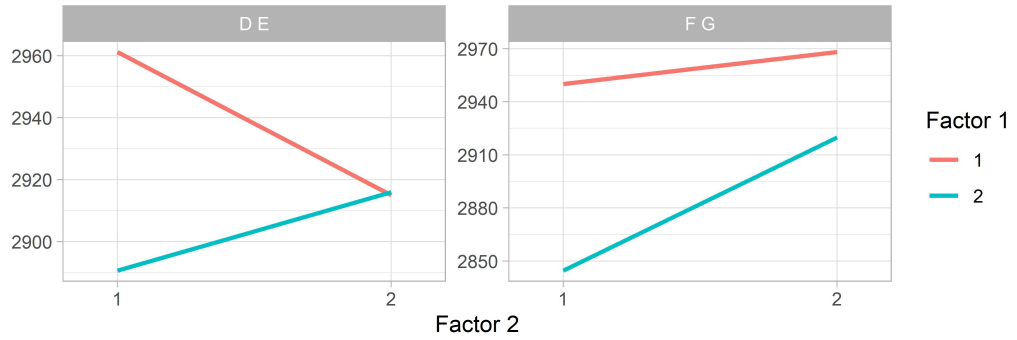

Figure 4.11: Test of interactions

information on the magnitude of contribution of each main effects and interactions. The calculation of ANOVA is the same as for a classical design of experiment, according to Yang and El-Haik, 2009.

Summary after reading "Introduction into R"

"In analysis of variance, mean squares are used in the F test to see if the corresponding effect is statistically significant."Yang and El-Haik, 2009 "F ratio is a better measure for relative performance"Yang and El-Haik, 2009

". The most commonly used criterion is to compare the p value with 0.05, or 5%, if p value is less than 0.05, then that effect is significant."Yang and El-Haik, 2009

"The variance ratio, commonly called the F statistic, is the ratio of variance due to the effect of a factor and variance due to the error term. (The F statistic is named after Sir Ronald A. Fisher.) This ratio is used to measure the significance of the factor under investigation with respect to the variance of all of the factors included in the error term. The F value obtained in the analysis is compared with a value from standard F-tables for a given statistical level of significance." Roy, 1990.

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 simply be done using R, which will result in table 4.1.

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
A	3	238901.41	79633.80	6.66	0.0004
B	3	49972.09	16657.36	1.39	0.2488
C	3	343169.03	114389.68	9.56	0.0000
D	1	38781.12	38781.12	3.24	0.0745
E	1	3507.03	3507.03	0.29	0.5893
F	1	189112.50	189112.50	15.81	0.0001
G	1	69751.13	69751.13	5.83	0.0174
D:E	1	41041.12	41041.12	3.43	0.0666
F:G	1	26277.78	26277.78	2.20	0.1411
Residuals	112	1339693.00	11961.54		

Table 4.1: ANOVA results

explain using R book

A, C, F and G have a relatively high F value, which suggests high influence on the model.

The Multiple R-squared: 0.4275, Adjusted R-squared: 0.3509 ... both are bad.

We can also look at the percentage contribution of each factor, using the formula gathered by Yang and El-Haik, 2009:

$$SS_T = SS_A + SS_B + SS_C + ... + SS_{error} \tag{4.3}$$

$$contribution_A = SS_A / SS_T * 100 \tag{4.4}$$

The percentage contribution is plotted in 4.4.3 (Sum of all factor contributions == Multiple R-squared in theory)

We can clearly see a high contribution of the residuals (error), which is concerning.

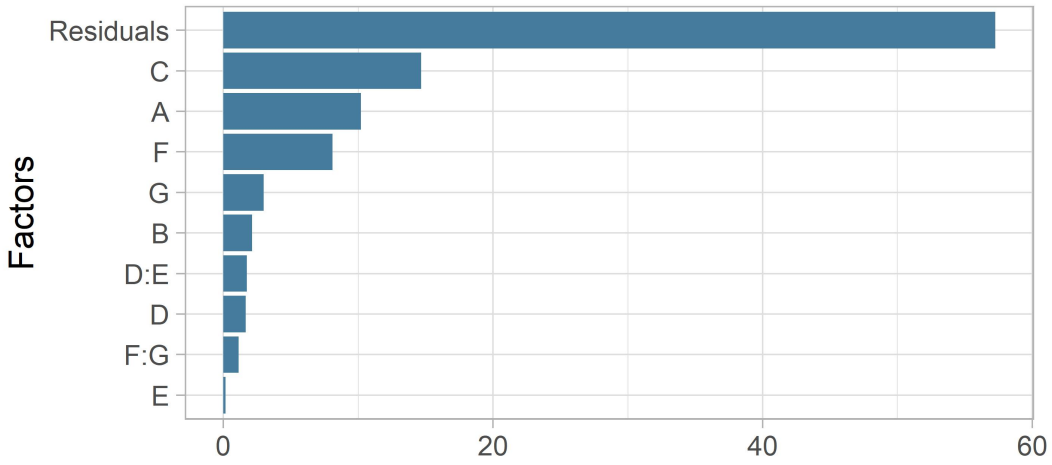

Figure 4.12: Percentage Contribution

Selection of optimal setting

When choosing the optimal setting, the first step is to look at the best main effects combination. For this experiment, the best combination would be the following: A4, B4, C3, D2, E2, F2, G1. Looking the ANOVA table, the interaction D:E has seams to have significance, especially compared to E. We will try to integrate this interaction. Compared to D:E, the second interaction (F:G) has a lower influence. This is also the case when looking at the individual factors F and G. The interaction F:G will thus not further be discussed.

The test of interaction in figure 4.4.3 suggest D2 and E1 as the best combination. This is optimal, as D2 is also suggested by the main effects. E1 is different to the suggested main effects, however its low F value in the anova table suggests low significance for using E2. Concluding this line of thought, the combination A4, B4, C3, D2, E1, F2, G1 looks to be optimal.

Optimum performance calculation Using optimal performance calculation 1. using only main effects or 2. using main effects with applied interaction

Better bars

can be applied. The equation provided by Roy, 1990.

$$\begin{aligned} Y_{opt} &= \bar{T} + (\bar{A}_4 - \bar{T}) + (\bar{B}_4 - \bar{T}) + (\bar{C}_3 - \bar{T}) + (\bar{D}_2 - \bar{T}) + (\bar{E}_2 - \bar{T}) + (\bar{F}_2 - \bar{T}) + (\bar{G}_1 - \bar{T} \\ &= 2693.984 \end{aligned} \tag{4.5}$$

$$\begin{aligned} Y_{opt} &= \bar{T} + (\bar{A}_4 - \bar{T}) + (\bar{B}_4 - \bar{T}) + (\bar{C}_3 - \bar{T}) + (\bar{D}_2 - \bar{T}) + (\bar{E}_1 - \bar{T}) + ([D\bar{x}E]_2 - \bar{T}) + (\bar{F}_2 \\ &= 2686.547 \end{aligned} \tag{4.6}$$

Using the interaction D:E, the performance estimation improves from 2693.984 to 2686.547. Considering this result, interaction will be taken into account and the optimized settings are as follows: CrossoverType: Uniform 0.5, CrossoverPropability: 0.9, MutationPropability: 0.3, ChromosomeType: Time+NPC, GeneType: integer encoding, TournamentSize: 4 and Individual-MutationPropability: 0.1.

signal-to-noise (S/N) As previously discussed when looking at the anova model, the error is very high, which suggests high randomness. Taguchi recommends using signal-to-noise (S/N) ratio to reduce the variability, as using only the mean of the results does not take the variation into account (Roy, 1990). The greater the signal-to-noise ratio, the smaller the variance. Roy, 1990 further states, that the "use of the S/N ratio offers an objective way to look at the two characteristics (consistency and average value) together."

When using S/N, todo: talk about equation.

Afterwards, generating main effects and anova table is the same as using the mean. However the DOF calcualtion done in equation 4.2 is no longer valid, as repetitions get combined to 1 value per run. So, the DOF for anova changes to the following equation in 4.7 according to Roy, 1990.

$$\begin{aligned} DOF &= totalNumberOfResults - 1 \\ &= numberOfTrials * 1 - 1 \\ &= 16 - 1 = 15 \end{aligned} \tag{4.7}$$

Find out why 15 DOF is not enough

This is not enough residuals for generating the F value in anova. Thus in order to reduce the current DOF, the anova table was generated without the interaction F:G considered, which can be seen in 4.2.

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
A	3	0.26	0.09	3.01	0.3953
B	3	0.05	0.02	0.60	0.7139
C	3	0.38	0.13	4.44	0.3326
D	1	0.04	0.04	1.48	0.4378
E	1	0.00	0.00	0.12	0.7845
F	1	0.21	0.21	7.35	0.2250
G	1	0.08	0.08	2.80	0.3429
D:E	1	0.04	0.04	1.56	0.4296
Residuals	1	0.03	0.03		

Table 4.2: S/N ANOVA results

When looking at the p values of this table, it is very obvious that no factor can discard the null - hypothesis, which states that a factor has no significant effect. Considering this, it was deemed to be not necessary to perform further investigations. Somehow argument, that having less variablity is only to an extend important. It is always possible to restart a simulation, if the variability is not too large, it is important that the results have a good mean overall. Less variability is in producing products much more important.

Elite Although the optimal hyperparameter setting will be discussed in chapter 5, a problem was obvious when analyzing a run using the optimzed GA. Figure 4.4.3 shows for a few selected repetitions the best individual cost per generation.

The lines show that setbacks in the optimal cost between two generations happens frequenctly. In order to mitigate this problem, it was decided to

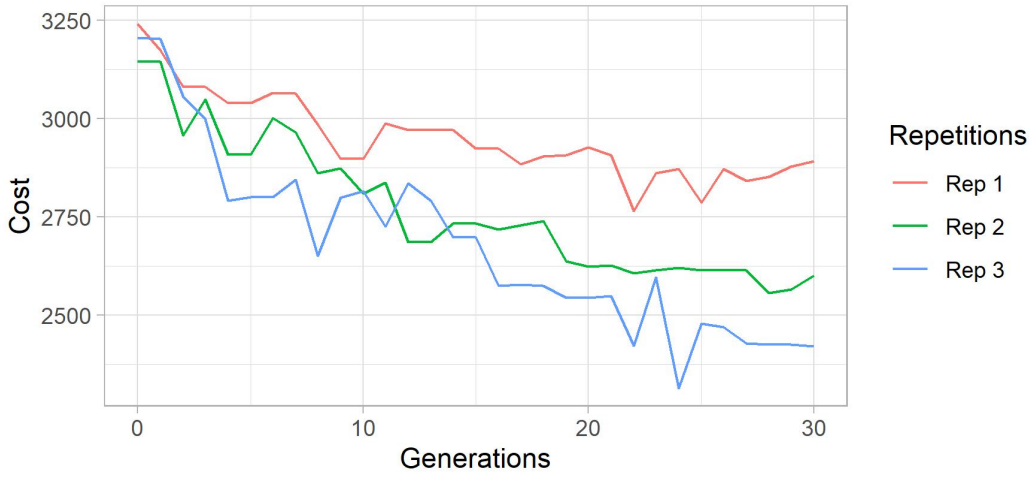

Figure 4.13: Genetic Algorithm without Elite

implement elite selection with a size of 2. This means, that per generation, the two best individuals are copied into the next generation without modifications, which makes worse performance between generations not possible. It is important to note, that the two best individuals can still be selected by tournament selection for modification, its just that a copy of them is saved. Figure **??** shows the effect of these changes.

Comparing the 10 repetitions also provides a clear picture in figure 4.4.3. It is thus concluded that the slightly modified version of the optimized Ga now using Elite of 2 will be used for chapter 5.

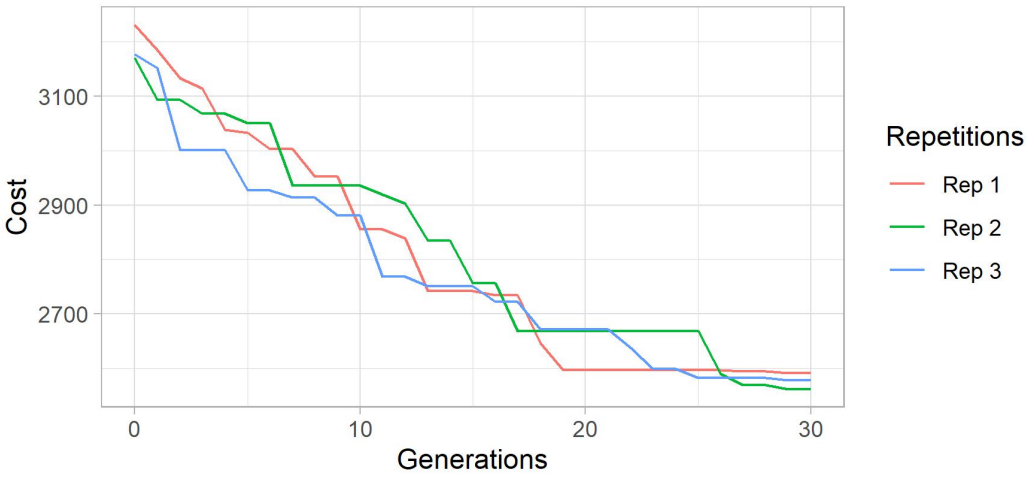

Figure 4.14: Genetic Algorithm with Elite

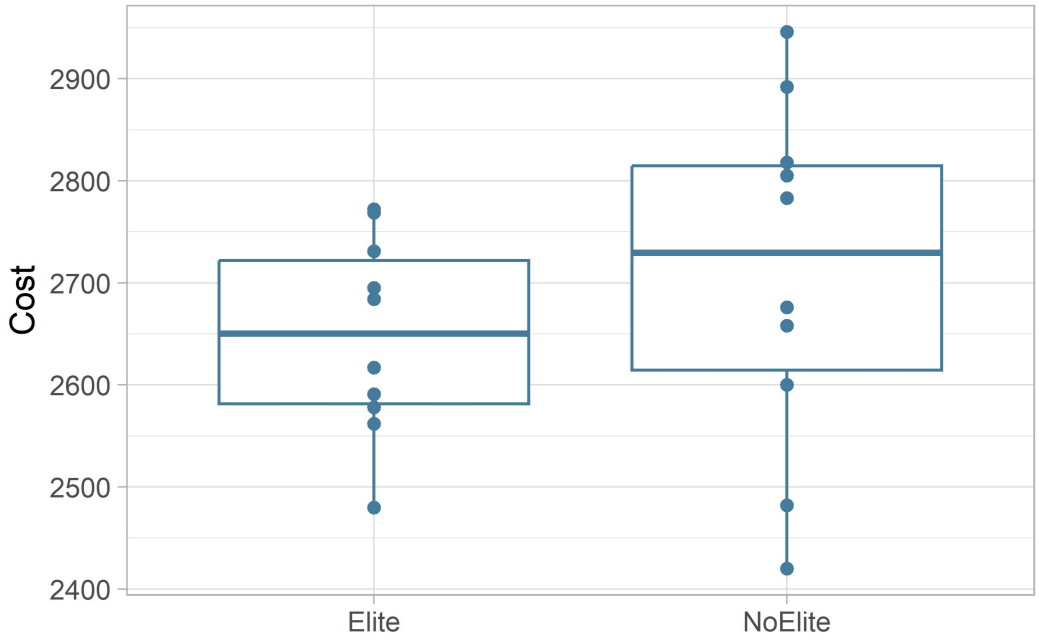

Figure 4.15: Comparison Elite vs No Elite

5 Evaluation

in this chapter, we evaluate and compare various different settings

"We found crossover and mutation most influential in GA success."Mills, Filliben, and Haines, 2015

"The cross over operator is found to be the most influential parameter in both the case studies, followed by mutation rate, population size for case study-1 and population size and selection process for case study-2. It is evident that the robust GA parameter settings are sensitive"Majumdar and Ghosh, 2015

Boyabatli and Sabuncuoglu, 2004 also suggest a high mutation rate.

5.1 Comparison with random and default ga Values

scenario 1: default : 9v 5p

5.2 Generalization on different start scenarios

Scenario 2

scenario 2: 9v 5p

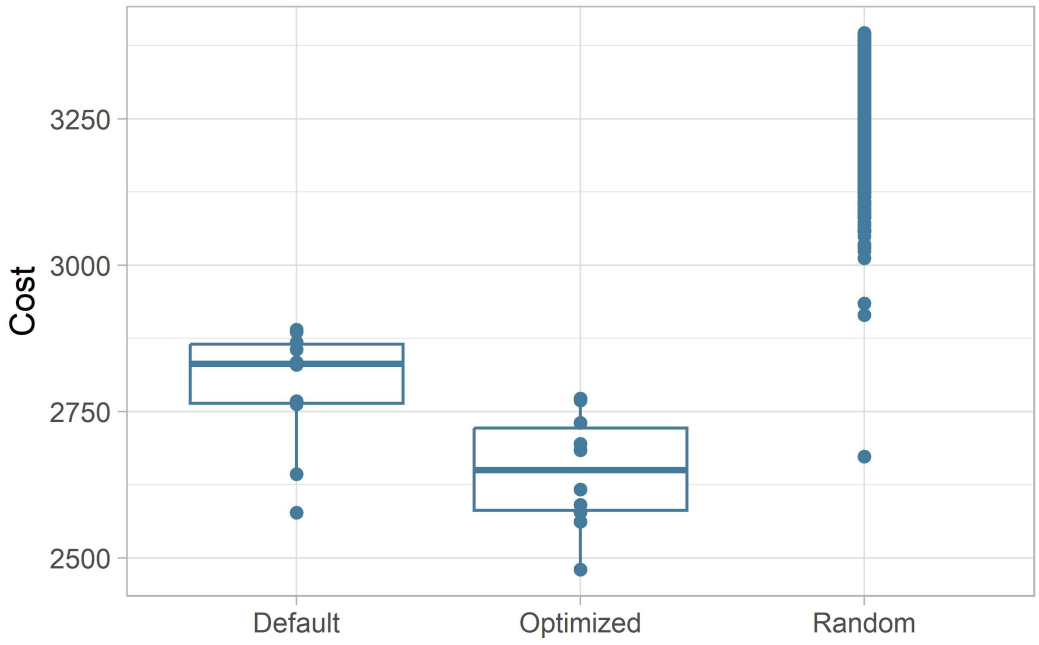

Figure 5.1: Genetic Algorithm with Elite

Scenario 3

scenario 3: 5v 3p

Scenario 4

also compare (average) diversity?

scenario 4: 18v 10p

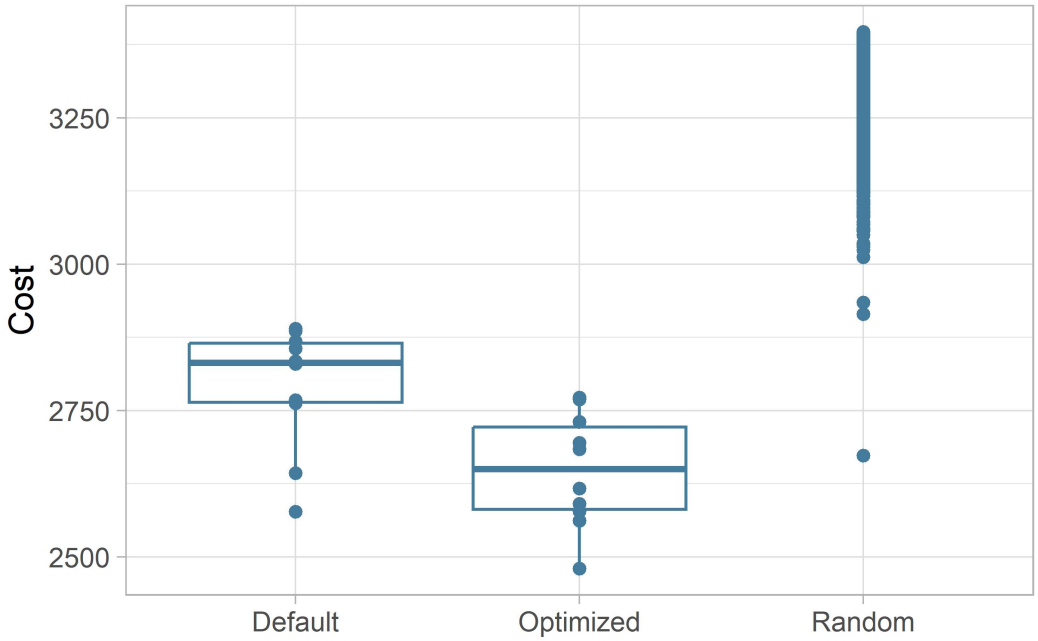

Figure 5.2: Genetic Algorithm with Elite

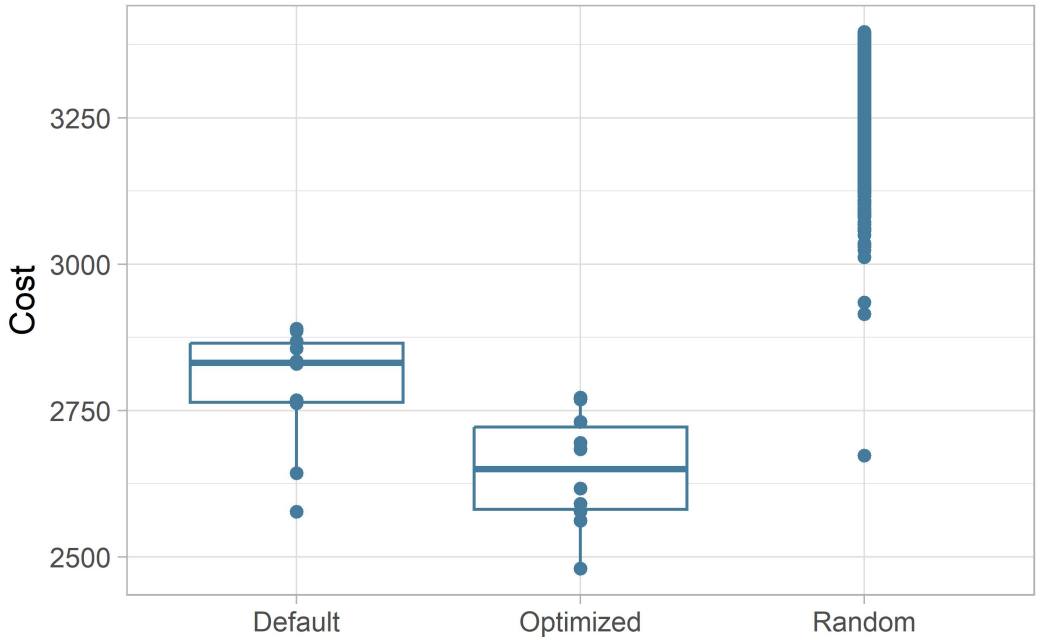

Figure 5.3: Genetic Algorithm with Elite

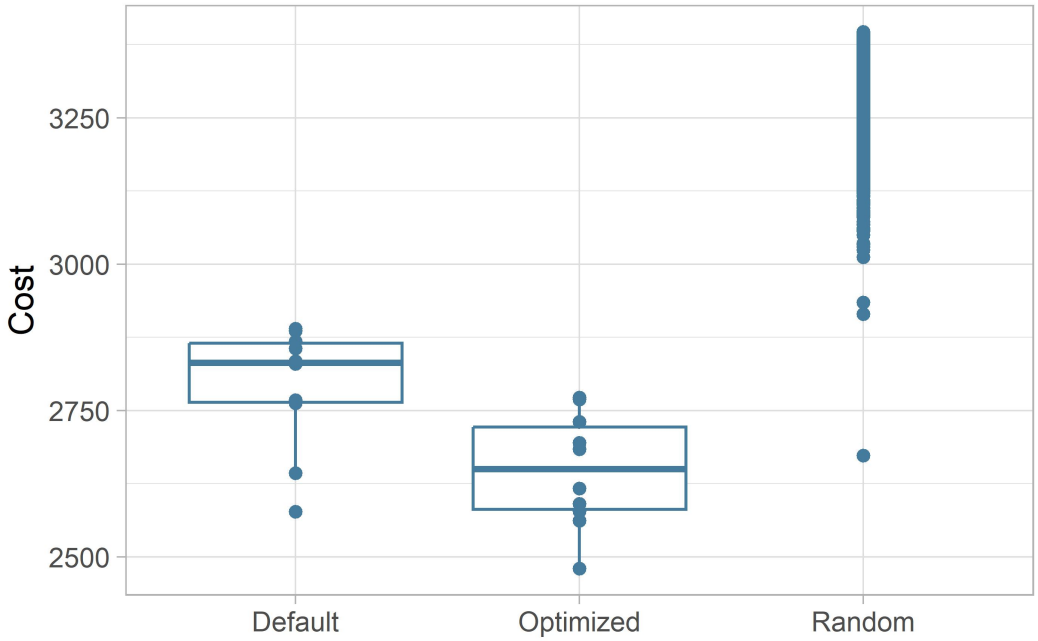

Figure 5.4: Genetic Algorithm with Elite

6 Conclusion

6.1 Future Work

6.1.1 Oracles

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

6.2 Final words

Appendix

Appendix A.

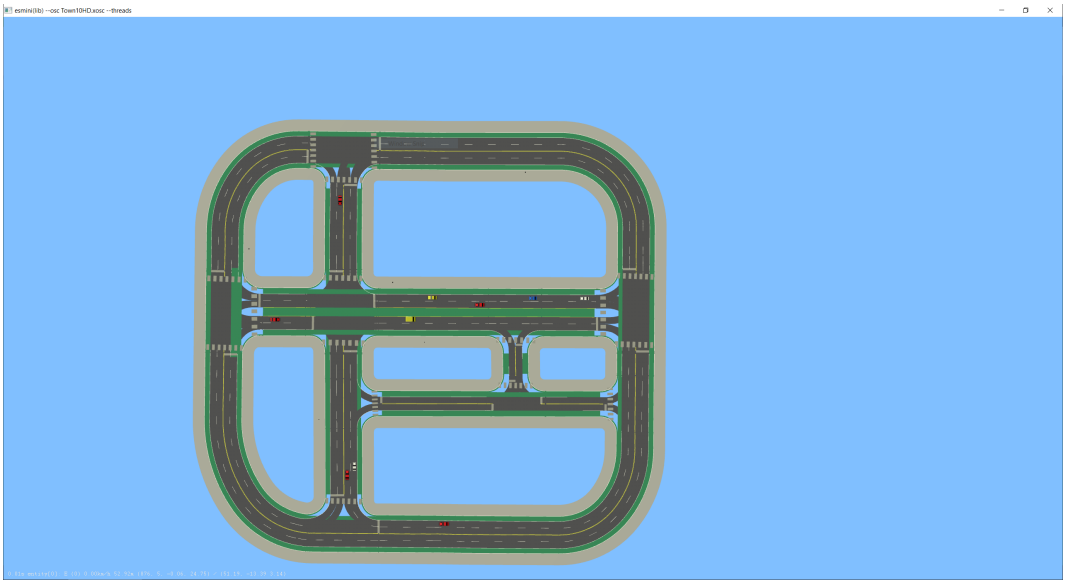

Figure 1: Start scenario 1

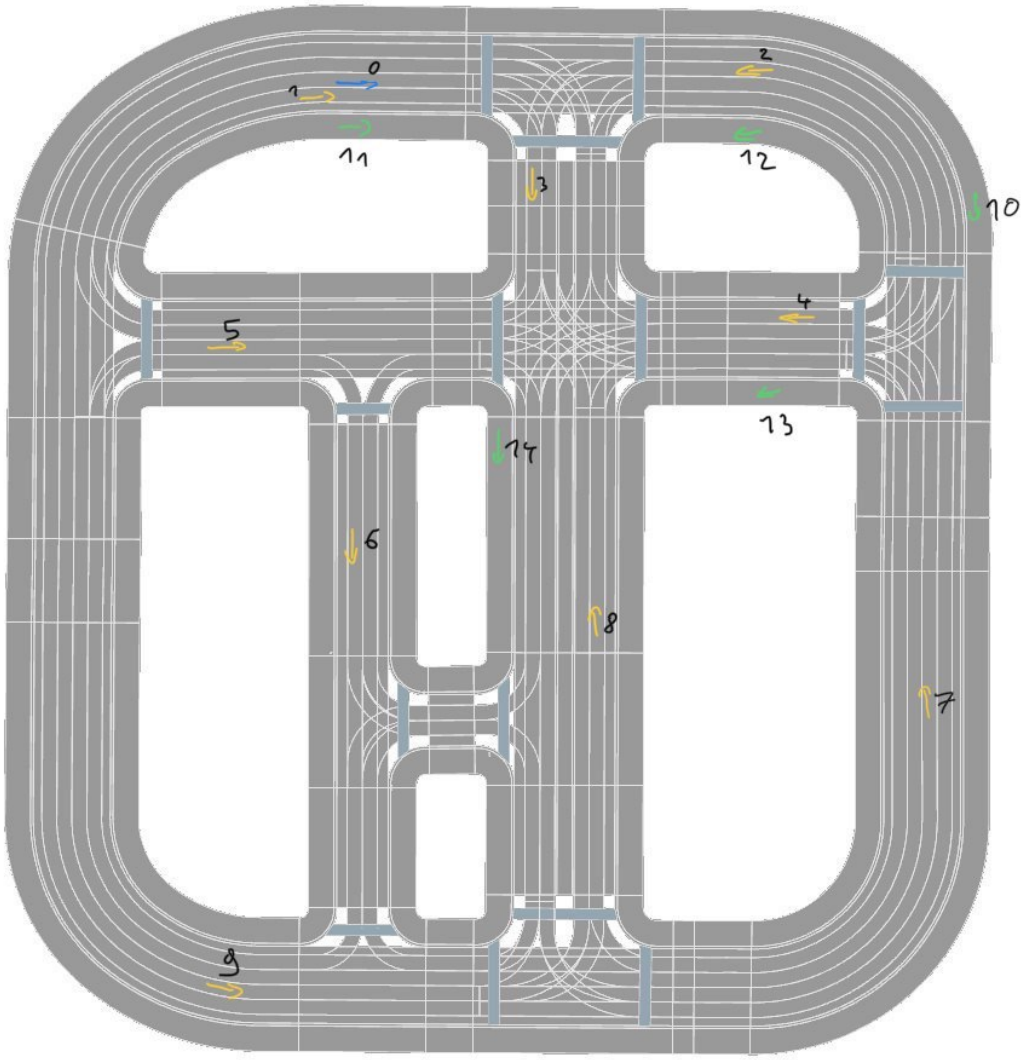

Figure 2: Start scenario 2

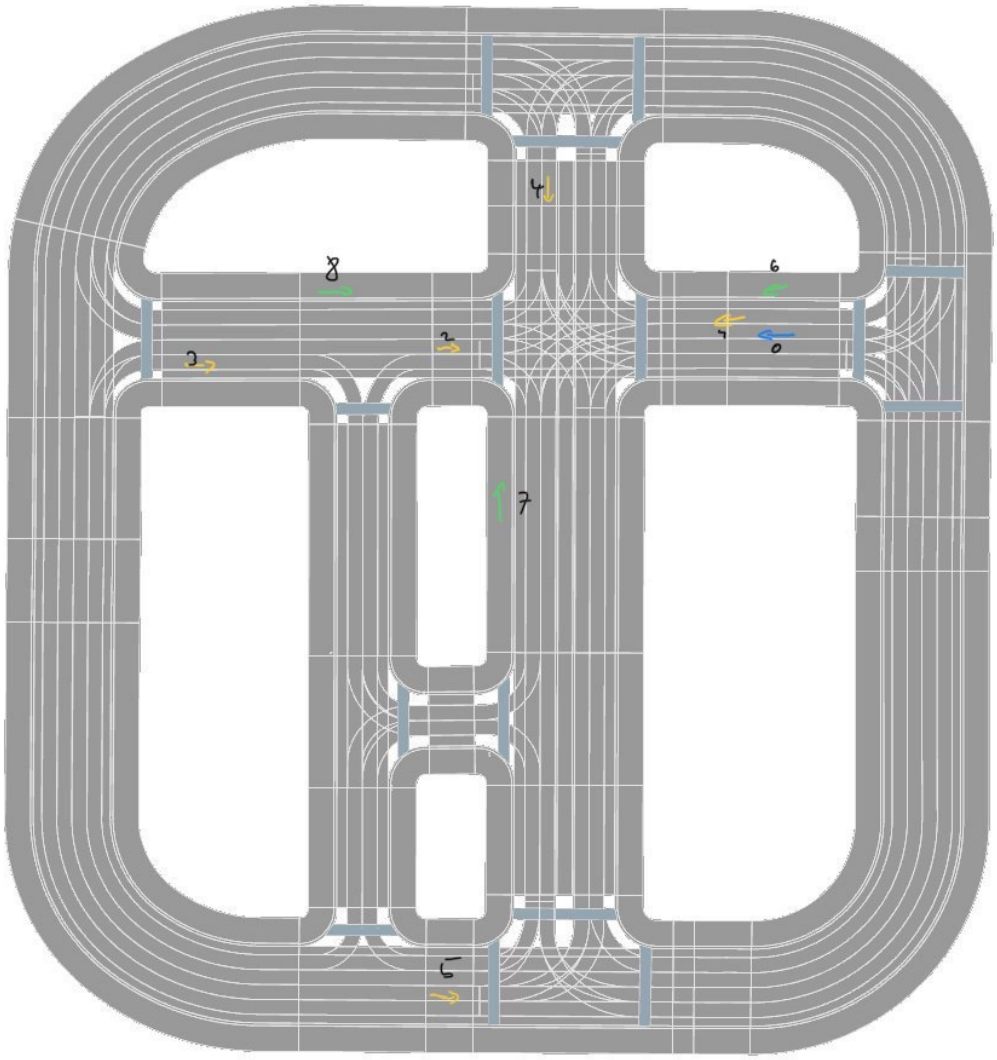

Figure 3: Start scenario 3

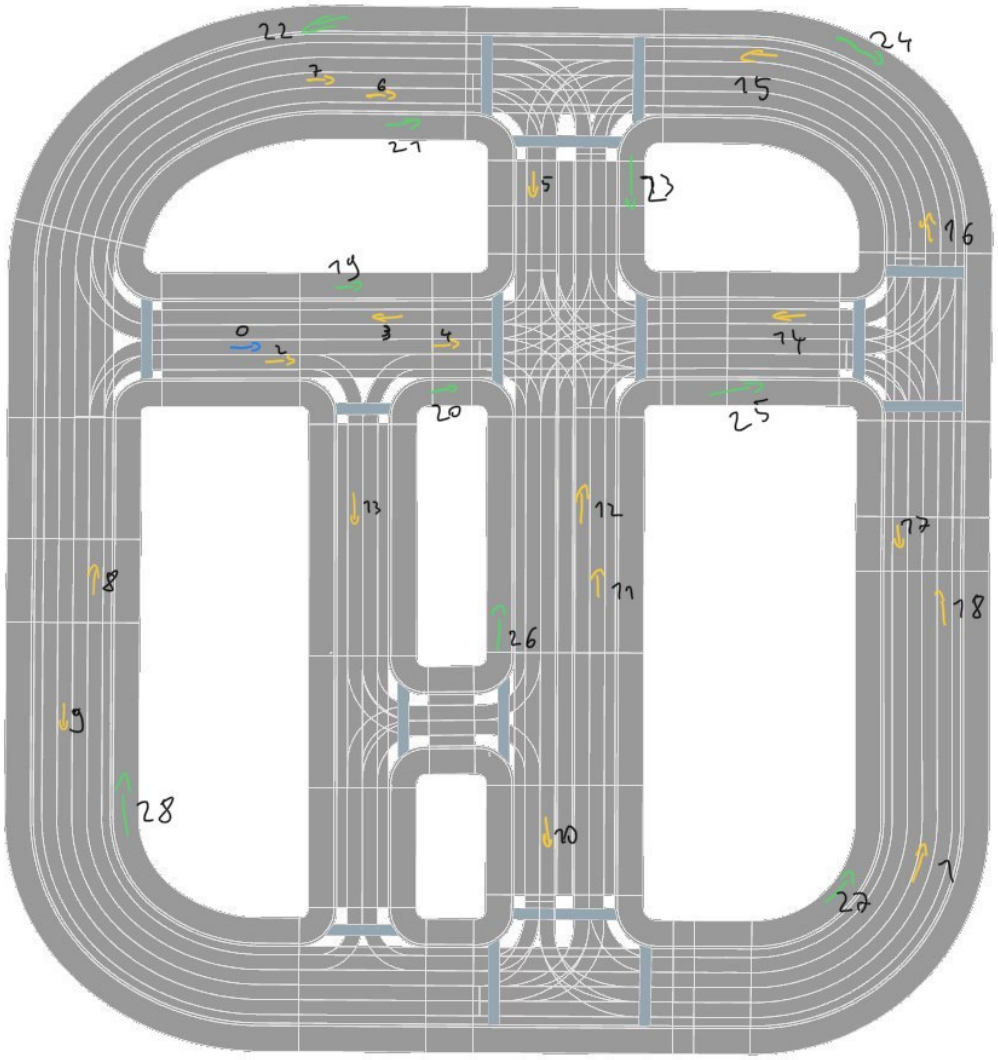

Figure 4: Start scenario 4

Appendix B.

NO.	rep1	rep2	rep3	rep4	rep5	rep6	rep7	rep8
1	3124	3110	3025	3077	3068	2925	3106	3105
2	2694	2980	3025	2996	3037	2921	3068	2900
3	2638	2711	2624	2856	2623	2832	2904	2778
4	2735	2805	2851	2965	2876	2703	2848	2858
5	3074	2955	3045	3080	3120	2971	2895	2798
6	2974	2979	2929	2941	2952	2936	3139	2953
7	3108	3099	3049	3020	3092	3057	3090	2895
8	2840	2931	2921	2921	2957	2997	2889	2895
9	3007	2995	2983	3009	2847	2996	2734	2927
10	2916	2828	3013	2787	2818	2926	3034	2822
11	3007	2879	3090	3033	2906	2981	3109	3104
12	2946	3016	2790	2917	2904	2983	2898	2606
13	2378	2712	2906	2800	2912	2795	2860	2834
14	2895	2760	2750	2849	2542	2997	2965	2991
15	2842	3065	3050	2779	2862	2923	2955	2892
16	3065	2834	2643	3056	3051	3011	2828	2963

Figure 1: List of results

Appendix C.

Insert information of Gene action probabitlities

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. 43).

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 pp. 44, 47).

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 pp. 11, 43, 65).

Dao, Son, Kazem Abhary, and Romeo Marian (Feb. 2016). "Maximising Performance of Genetic Algorithm Solver in Matlab." In: *Engineering Letters* 24 (cit. on pp. 41, 44, 47).

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. 41, 42).

De Jong, Kenneth Alan (1975). *An analysis of the behavior of a class of genetic adaptive systems.* University of Michigan (cit. on p. 43).

Erickson, Mark, Alex Mayer, and Jeffrey Horn (Jan. 2002). "Multi-objective optimal design of groundwater remediation systems: application of the niched Pareto genetic algorithm (NPGA)." In: *Advances in Water Resources* 25.1, pp. 51–65. ISSN: 03091708. DOI: 10.1016/S0309-1708(01)00020-3. URL: https://linkinghub.elsevier.com/retrieve/pii/S0309170801000203 (visited on 10/23/2023) (cit. on p. 28).

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. 44).

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. 5, 13, 14, 20, 21, 24, 42, 43).

Hamzaçebi, Coşkun (Mar. 24, 2021). "Taguchi Method as a Robust Design Tool." In: *Quality Control - Intelligent Manufacturing, Robust Design and Charts*. Ed. by Pengzhong Li, Paulo António Rodrigues Pereira, and Helena Navas. IntechOpen. ISBN: 978-1-83962-497-1 978-1-83962-498-8. DOI: 10.5772/intechopen.94908. URL: https://www.intechopen.com/books/quality-control-intelligent-manufacturing-robust-design-and-charts/taguchi-method-as-a-robust-design-tool (visited on 10/28/2023) (cit. on p. 48).

Hussain, Abid and Yousaf Shad Muhammad (Apr. 2020). "Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator." In: *Complex & Intelligent Systems* 6.1, pp. 1–14. ISSN: 2199-4536, 2198-6053. DOI: 10.1007/s40747-019-0102-7. URL: http://link.springer.com/10.1007/s40747-019-0102-7 (visited on 07/23/2023) (cit. on pp. 4, 15–17).

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 pp. 10, 18, 19, 43, 44).

Katoch, Sourabh, Sumit Singh Chauhan, and Vijay Kumar (Feb. 1, 2021). "A review on genetic algorithm: past, present, and future." In: *Multimedia Tools and Applications* 80.5, pp. 8091–8126. ISSN: 1573-7721. DOI: 10.1007/s11042-020-10139-6. URL: https://doi.org/10.1007/s11042-020-10139-6 (visited on 03/28/2023) (cit. on pp. 7, 8, 12, 14, 17, 18, 21, 22, 25, 27, 28).

Klampfl, Lorenz, Florian Klück, and Franz Wotawa (2023). "Using genetic algorithms for automating automated lane-keeping system testing." In: *Journal of Software: Evolution and Process* n/a (n/a). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/smr.2520, e2520. ISSN: 2047-7481. DOI: 10.1002/smr.2520. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2520 (visited on 03/29/2023) (cit. on pp. 3, 4, 13, 21, 24).

Majumdar, Abhishek and Debashis Ghosh (Oct. 15, 2015). "Genetic Algorithm Parameter Optimization using Taguchi Robust Design for Multiresponse Optimization of Experimental and Historical Data." In: *International Journal of Computer Applications* 127.5, pp. 26–32. ISSN: 09758887. DOI: 10.5120/ijca2015906383. URL: http://www.ijcaonline.org/research/volume127/number5/majumdar-2015-ijca-906383.pdf (visited on 10/27/2023) (cit. on pp. 7, 14, 17, 21, 25, 27, 29, 65).

Marsili Libelli, S. and P. Alba (July 2000). "Adaptive mutation in genetic algorithms." In: *Soft Computing* 4.2, pp. 76–80. ISSN: 1432-7643. DOI: 10.1007/s005000000042. URL: http://link.springer.com/10.1007/s005000000042 (visited on 11/21/2023) (cit. on pp. 9, 22, 25, 26).

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. 5, 7, 14, 26, 42–44, 48, 65).

NazanDanacıoğlu, F. ZehraMuluk (2005). "TAGUCHI TECHNIQUES FOR 2^k FRACTIONAL FACTORIAL EXPERIMENTS." In: *Journal* 34.1, pp. 83–93. ISSN: 2651-477X-2651-477X (cit. on p. 52).

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. 1, 47, 48, 50, 51, 53, 55–57, 60).

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 pp. 5, 6, 10–12, 19–21, 23, 24, 27, 43).

Taguchi, Genichi et al. (2005). *Taguchi's quality engineering handbook*. Hoboken, N.J. : Livonia, Mich: John Wiley & Sons ; ASI Consulting Group. 1662 pp. ISBN: 978-0-471-41334-9 (cit. on pp. 52, 53).

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. 43).

Xia, Xuemin et al. (Feb. 1, 2019). "Genetic algorithm hyper-parameter optimization using Taguchi design for groundwater pollution source identification." In: *Water Supply* 19.1, pp. 137–146. ISSN: 1606-9749, 1607-0798. DOI: 10.2166/ws.2018.059. URL: https://iwaponline.com/ws/article/19/1/137/39199/Genetic-algorithm-hyperparameter-optimization (visited on 10/06/2023) (cit. on p. 6).

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. 46–50, 52, 55–58).